**SIMULATION OF ELECTRIC VEHICLE**

**CHARGING USING GB/T PROTOCOL**

**A project Report**

*Submitted in partial fulfilment of the requirements for the award of the degree*

*of*

BACHELOR OF TECHNOLOGY

*in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

*By*

**Y. MANISHA U. NAGA SAI VAMSI (18B91A04N9) (19B95A0423)**

**M.MOUNIKA T. NANDA KRSIHNA (19B95A0412) (19B95A0422)**

*Under the esteemed guidance of*

### Mr. P. KRISHNA KANTH VARMA

**M.tech*.***

**Assistant Professor, Department of E.C.E**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**S.R.K.R. ENGINEERING COLLEGE (AUTONOMOUS)**

**(Affiliated to JNTU, KAKINADA)**

**(Recognized by A.I.C.T.E., Accredited by N.A.A.C. with ‘A’ Grade, NEW DELHI)**

**CHINNA AMIRAM, BHIMAVARAM- 534204**

**(2021-2022)**

# S.R.K.R. ENGINEERING COLLEGE (A)

### (Affiliated to JNTU, KAKINADA)

**(Recognized by A.I.C.T.E., Accredited by N.A.A.C. with ‘A’ Grade, NEW DELHI)**

**CHINNA AMIRAM, BHIMAVARAM - 534204 (2018-2022)**

**DEPARTMENT OF**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**CERTIFICATE**

***This is to certify that Project work entitled***

**“Simulation of Electric Vehicle Charging Using GB/T protocol ”**

*This is to certify that the project work entitled* **“****SIMULATION OF ELECTRIC VEHICLE CHARGING USING GB/T PROTOCOL”** *is the bona fide work of*  **Y.MANISHA, M.MOUNIKA, U.NAGA SAI VAMSI, T.NANDA KRISHNA** bearing **18B91AO4N9, 19B95A0412, 19B95A0423, 19B95A0422** *in the partial fulfilment of the requirements for the award of the degree of* **BACHELOR OF TECHNOLOGY in *ELECTRONICS AND COMMUNICATION ENGINEERING* during** *the* *academic year* **2021-2022.**

| **Guide** | **Head of the Department** |
| --- | --- |
| **Mr. P. KRISHNA KANTH VARMA**  Assistant Professor**,** | **Dr. N. UDAYA KUMAR**  Professor |
| **M.Tech.** | **M.Tech, Ph.D., S.M.I.E.E.E.,F.I.E** |
| **Department of ECE** | **Department of ECE** |

**CERTIFICATE OF EXAMINATION**

This is to certify that we had examined the thesis and here by accord our approval of it as a study carried out and presented in a manned required for its acceptance in a fulfillment for the award of BACHELOR OF TECHNOLOGY in ELECTRONICS AND COMMUNICATION ENGINEERING for which it has been submitted. This approval does not endorse or accept every statement made, opinion expressed, or conclusion drawn as in report. It signifies acceptance of report for the purpose for which it is submitted.

External Examiner Internal Examiner

## DECLARATION

This is to certify that the project entitled “ELECTRIC VEHICLE CHARGING USING GB/T PROTOCOL” is submitted by Y. MANISHA (18B91A04N9), M. MOUNIKA (19B95A0412), U. NAGA SAI VAMSI (19B95A0423), T. NANDA KRISHNA (19B95A0422) in partial fulfillment of the requirement for the award of degree B. Tech in Electronics and Communication Engineering to S.R.K.R Engineering College, affiliated to JNTU KAKINADA. It comprises only our original work and the acknowledgement has been made in text to all other material used.

Date :

## ACKNOWLEDGEMENTS

The successful completion of any task would be incomplete without greeting those who made it possible and whose guidance and encouragement made the effort taken a success.

I express my sincere thanks to my beloved and esteemed guide **Mr. P. KRISHNA KANTH VARMA,** Department of Electronics and Communication Engineering, for his excellent guidance and constant encouragement throughout the project work. His timely guidance and motivation helped me to complete the project work.

I would like to express my sincere thanks to **Dr. N. UDAYA KUMAR,** Head of the Department of Electronics and Communication Engineering for his cooperation, encouragement and constant review during the project.

I would like to express my sincere thanks to **Dr. M JAGAPATHI RAJU**, Principal, SRKR Engineering College, Bhimavaram for giving me this opportunity for the successful completion of my project.

I also thank other teaching and non-teaching staff for their assistance and help extended. I thank one and all that have contributed directly or indirectly to our project.

### Project Associates

Y. MANISHA 18B91A04N9

M. MOUNIKA 19B95A0412

U. NAGA SAI VAMSI 19B95A0423

T. NANDA KRISHNA 19B95A0422

## CONTENTS

***Acknowledgement***

## *Acknowledgement*

[***List of Figure 1***](#_heading=h.gjdgxs)

[***List of Tables***](#_heading=h.30j0zll) ***2***

[***Abbreviations 3***](#_heading=h.1fob9te)

[***Abstract 5***](#_heading=h.3znysh7)

1. **INTRODUCTION 6-9**
   1. [Electric Vehicles 6](#_heading=h.2et92p0)
   2. Histrory of EV Charging technology 7
   3. [Types of EV chargers 8](#_heading=h.44sinio)
   4. [Battery Management System 8](#_heading=h.2jxsxqh)
2. **SOFTWARE 10-17**
   1. [Linux Shell Scripting 1](#_heading=h.3dy6vkm)0
   2. Python 10
      1. [Python libraries 1](#_heading=h.z337ya)1
      2. [Indentation 1](#_heading=h.3j2qqm3)3
      3. [Characteristics of python 1](#_heading=h.1y810tw)4
      4. [Features of python 1](#_heading=h.4i7ojhp)4
   3. HTML 14
   4. CSS 15
   5. [Apache server 1](#_heading=h.2xcytpi)6
3. **COMMUNICATION PROTOCOL 18-27**
   1. [Introduction 1](#_heading=h.4d34og8)8
   2. CAN BUS 18
      1. The [CAN standard 1](#_heading=h.2s8eyo1)8
      2. Standard CAN or Extended CAN 19
   3. GB/T PROTOCOL 21
      1. General GB/T 21
      2. Layers 22
      3. PDU format 22
   4. OSI modal 24
4. **BATTERY MANAGEMENT SYSTEM 28-37**
   1. [Introduction 2](#_heading=h.17dp8vu)8
   2. State of charge [2](#_heading=h.1ci93xb)9
   3. State of health [2](#_heading=h.3whwml4)9
   4. Design of BMS 30
   5. Estimation of SOC 30
      1. Electrical modal of battery 30
      2. Dual Extended Kalman Filter 31

4.6 Key technologies in BMS 35

1. **LITERATURE SURVEY 38-41**

| **6. METHODOLOGY**  6.1 [Virtual box 31](#_heading=h.35nkun2)  6.2 Ubuntu 42 [32](#_heading=h.2bn6wsx)  6.3 [Installation Process 42 32](#_heading=h.qsh70q)  6.4 Working 47 | **42- 52** |  |
| --- | --- | --- |
| 1. **RESULTS AND DISCUSSIONS** | **53-57** |  |
| 7.1. Charger HTML page 53  7.2 Battery Management system HTML page 53 |  |  |
| 7.3 At 50% SOC 54 |  |  |
| 7.4 At 75% SOC 55 |  |  |
| 7.5 At 90% SOC 56 |  |  |
| 7.6 Communication Between Charger and BMS 57  7.7 Output analysis 57 |  |  |
|  |  |  |
|  |  |  |
| **8. CONCLUSION** | **58** |  |
| [8.1 Conclusion](#_heading=h.3as4poj) 58 |  |  |
| 8.2 [Future Scope](#_heading=h.1pxezwc) 58 |  |  |
|  |  |  |
| **REFERENCES** | **59-60** |  |

## LIST OF FIGURES

| **Fig no** | **Title** | **Page no** |
| --- | --- | --- |
| 1.2 | Corner of battery room | 6 |
| 3.2.1 | The layered ISO 11898 standard architecture | 19 |
| 3.2.2 | Standard CAN :11-Bit Identifier | 19 |
| 3.2.3 | Extended CAN:29-Bit Identifier | 20 |
| 3.3.1 | GENERAL GB/T | 21 |
| 3.3.3 | Protocol Data Unit (PDU) | 22 |
| 3.4 | OSI MODEL | 27 |
| 3.4 | Big Endian | 31 |
| 3.4 | Little Endian | 31 |
| 4.1 | BATTERY MANAGEMENT SYSTEM | 32 |
| 4.5.1 | ELECTRICAL MODEL OF BATTERY | 34 |
| 4.5.1 | A TYPICAL OCV-SOC CHARACTERISTICS OF 18650 | 35 |
| 4.5.2 | SCHEMATICS OF CLOUD BMS | 37 |
| 4.6 | BATTERY ELECTRIC MODEL | 40 |
| 4.6 | THREE CLASSIFICATIONS OF BATTERY MODELLING | 41 |
| 4.6 | TRADITIONAL CHARGING APPROACHES FOR BATTERY IN EV | 42 |
| 5.1 | IMAGE OF VIRTUAL BOX | 44 |
| 5.2 | MEMORY ALLOCATION FOR UBUNTU | 44 |
| 5.3 | COMMANDS FOR UPDATING TERMINAL | 45 |
| 5.4 | INSTALLATION OF SCIPY AND CHROMIUM BROWSER | 46 |
| 5.5 | INSTALLATION OF TORNADO AND NODEJS | 47 |
| 5.6 | INSTALLATION OF NPM AND SOCKET CAN | 48 |
| 5.7 | COMMANDS FOR STARTING THE PROCESS OF EV CHARGING | 49 |
| 5.8 | OVERALL CHARGING PROCESS | 50 |
| 5.9 | OUTPUT PAGE | 53 |
| 5.10 | COMMUNICATION BETWEEN AND CHARGER AND BMS AND VICE VERSA | 54 |
| 7.1 | CHARGER HTML PAGE | 60 |
| 7.2 | BATTERY MANAGEMENT SYSTEM HTML PAGE | 60 |
| 7.3 | AT SOC 50% | 61 |
| 7.4 | AT SOC 75% | 62 |
| 7.5 | AT SOC 90% | 63 |
| 7.6 | COMMUNICATION BETWEEN CHARGER AND BMS | 64 |

## LIST OF TABLES

| **TABLE NO.** | **NAME OF THE TABLE** | **PAGE NO.** |
| --- | --- | --- |
| 1 | PROTOCOL DATA UNIT | 24 |
| 4.6 | KEY TECHNOLOGIES FOR BMS | 38 |
| 5.1.1 | CLASSIFICATION OF MESSAGES DURING CHARGING HANDSHAKE STAGE | 50 |
| 5.1.2 | CLASSIFICATION OF MESSAGES DURING CHARGING PARAMETER CONFIGURATION STAGE | 50 |
| 5.1.3 | CLASSIFICATION OF MESSAGES DURING CHARGING STAGES | 51 |
| 5.1.4 | CLASSIFICATION OF MESSAGES DURING END CHARGE STAGE | 51 |

## ABBREVIATIONS

| EV | Electric Vehicle |
| --- | --- |
| BMS | Battery Management System |
| GB/T | Guobiao |
| CLI | Command Line-Interface |
| SOA | Safe Operating Area |
| CAN  GUI    HTML | Controller Area Network  Graphical User Interface    Hyper Text Markup Language |
| CSS | Cascading Style Sheet |
| ISO | International Standardization Organization |
| NUMPY  PIP  SCIPY | Numeric python  Package Installer for Python  Scientific Python |
| OSI    SSR    IFS    IDE    PDU    SOH    SOP    SOC    CBMS | Open Systems Interconnection  Substitute Remote request  Interframe Space  Identifier Extension  Identifier Extension  State Of Health    State Of Power  State Of Charge    Cloud Battery Management System |

## ABSTRACT

Communication among Electric Vehicle (EV), Electric Vehicle Supply Equipment (EVSE) and Battery Management System (BMS) is crucial for efficient and safe charging of an EV. To ensure safe charging process we should consider the parameters such as BMS effects, Communication between BMS and Charger i.e., GB/T protocol using CAN. Controller Area Network, which is a robust vehicle bus standard designed to allow microcontrollers and other devices to communicate with each other applications without any Host computer. In this project a simulation model for communication between the Charger and BMS is developed using GB/T standard and CAN bus protocol. By varying the SOC range bar in charger HTML page we observed the SOC levels changes in BMS HTML page.

## CHAPTER 1 INTRODUCTION

### ELECTRIC VEHICLES

An EV is a shortened acronym for an electric vehicle. EVs are vehicles that are either partially or fully powered on electric power**.** Electric vehicles have low running costs as they have less moving parts for maintaining and very environmentally friendly as they use little or no fossil fuels (petrol or diesel). While some EVs used lead acid or nickel metal hydride batteries, the standard for modern battery electric vehicles is now considered to be lithium-ion batteries as they have a greater longevity and are excellent at retaining energy, with a self-discharge rate of just 5% per month. Despite this improved efficiency, there are still challenges with these batteries as they can experience thermal runaway, which have, for example, caused fires or explosions in the Tesla model S, although efforts have been made to improve the safety of these batteries.

### ELECTRIC VEHICLE CHARGERS

An EV charger is the EV equivalent of a gas pump: it’s how you fill your EV’s “tank”. Except instead of a gas tank, your EV has a battery. And instead of using gasoline to power the engine of your car, an EV’s motor runs on electricity. EV chargers–often also called EVSE for electric vehicle supply equipment–come in a few different shapes and sizes. There are residential EV chargers, commercial EV chargers, and even portable EV chargers! But the two biggest differences between an EV charger and a traditional gasoline pump are: where you can put it and what comes out of it. A few men working in a factory

Description automatically generated with low confidence

**History of EV charging technology**

Since the first appearance of electric vehicles (EV) in mid-19th century, there has been a need for EV charging technology. The first EVs had disposable batteries and technology for „charging “was replacing „dead “batteries. Invention of rechargeable batteries meant that new models of EVs had the possibility to use the same batteries without replacement. All these early models of EVs were not mass made, thus, there was no need for commercial charging stations. Also, main problem was that until early 20th century, many homes were without electricity, and that meant that it was impossible to charge the vehicle in homes. Electrification of homes meant that EVs would be more accessible to public. This would mean that more and more people would buy and use EVs. At the beginning of 20th century, 38% of automobiles in USA were powered by electricity. These cars would charge either with batteries in the vehicle, or battery was removed from the vehicle, charged at another place, and then mounted back on vehicle.A picture containing text, outdoor, person, road

Description automatically generated

During these early EV days, a company called General Electric introduced first charging stations called „Electrant “. They were like phone boxes set up around major US cities, and EV users could use them to recharge the vehicle.

Due to evolution of automotive industry in 1920’s, improved road quality and quantity, EVs with their limited range were not more suitable for journeys, and this meant that their purpose was limited. Cheaper gasoline price during this period surely had even more influence on usage of internal combustion engine vehicles. This meant that until late 20th century, EVs were merely a quirk in automotive industry. In late 20th century, public became more and more aware of air pollution, and idea of producing EVs started to rise again. Automotive companies started to make models of EVs, and question of charging infrastructure followed. First models of these EVs were able to be charged at home, using a normal socket. At first, hybrid cars were considered as compromise between ICE vehicles and EV, and home charging was enough to recharge them. As soon as companies started to make plug in EVs, need for a proper public charging infrastructure grew. This brings us to 21st century and evolution of EVs and charging technology.

### TYPES OF CHARGERS

There are three different levels of EV charging, conveniently known as Level 1, Level 2 and Level 3 charging.

* **Level 1** charging is what you get when you charge your EV from a regular wall outlet. It’s typically 120 Volt charging, and works very, very slowly. This type of charging can work in a pinch but you’re unlikely to get more than 5 miles of range per hour of charge.
* **Level 2** charging is the most common type of charging used in homes. While Level 2 charging requires special equipment, it provides significantly faster charging times than a Level 1 charger: it operates at 208 or 240 Volts and charges up to 10x faster than a Level 1. Generally speaking, you can charge your car overnight with a Level 2 charger, though you will need a professional electrician to install it.
* **Level 3**charging is also often referred to as direct current (DC) supercharging. Typically, these operate at 400 Volts, which is a higher level of service than most homes have. It’s best to think of Level 3 as where you top off or fill up on road trips–these chargers provide a significant hit of energy quickly, helping you fill your battery in half an hour or less.
  1. **BATTERY MANAGEMENT SYSTEM**

Battery management system (BMS) is technology dedicated to the oversight of a battery pack, which is an assembly of battery cells, electrically organized in a row x column matrix configuration to enable delivery of targeted range of voltage and current for a duration of time against expected load scenarios. The oversight that a BMS provides usually includes:

* Monitoring the battery
* Providing battery protection
* Estimating the battery’s operational state
* Continually optimizing battery performance
* Reporting operational status to external devices

Here, the term “battery” implies the entire pack; however, the monitoring and control functions are specifically applied to individual cells, or groups of cells called modules in the overall battery pack assembly. Lithium-ion rechargeable cells have the highest energy density and are the standard choice for battery packs for many consumer products, from laptops to electric vehicles. While they perform superbly, they can be rather unforgiving if operated outside a generally tight safe operating area (SOA), with outcomes ranging from compromising the battery performance to outright dangerous consequences. The BMS certainly has a challenging job description, and its overall complexity and oversight outreach may span many disciplines such as electrical, digital, control, thermal, and hydraulic.

# CHAPTER 2

## SOFTWARE

### LINUX SHELL SCRIPTING

### A shell script is a text [file](https://www.techtarget.com/whatis/definition/file) that contains a sequence of commands for a [UNIX](https://www.techtarget.com/searchdatacenter/definition/Unix)-based [operating system](https://www.techtarget.com/whatis/definition/operating-system-OS). It is called a shell script because it combines a sequence of commands, that would otherwise have to be typed into the keyboard one at a time, into a single script. The [shell](https://www.techtarget.com/searchdatacenter/definition/shell) is the operating system's command-line interface (CLI) and interpreter for the set of commands that are used to communicate with the system. A shell script is usually created for command sequences in which a user has a need to use repeatedly in order to save time. Like other programs, the shell script can contain parameters, comments and subcommands that the shell must follow. Users initiate the sequence of commands in the shell script by simply entering the file name on a command line.

### The basic steps involved with shell scripting are writing the script, making the script accessible to the shell and giving the shell execute permission. Shell scripts contain [ASCII](https://www.techtarget.com/whatis/definition/ASCII-American-Standard-Code-for-Information-Interchange) text and are written using a [text editor](https://www.techtarget.com/whatis/definition/text-editor), [word processor](https://www.techtarget.com/searchwindowsserver/definition/word-processor) or graphical user interface ([GUI](https://www.techtarget.com/whatis/definition/GUI)). The content of the script is a series of commands in a language that can be interpreted by the shell. Functions that shell scripts support include [loops](https://www.techtarget.com/whatis/definition/loop), variables, if/then/else statements, arrays and shortcuts. Once complete, the file is saved typically with a .txt or .sh extension and in a location that the shell can access.

Shell scripting is meant to be simple and efficient. It uses the same syntax in the script as it would on the shell command line, removing any interpretation issues. Writing code for a shell script is also faster and requires less of learning curve than other programming languages. However, if there is an error in a shell script, this can prove to be extremely costly if left unnoticed. Additionally, differing platforms associated with shell scripting may not be compatible. Shell scripts can also be slower to execute than individual commands.

**2.2 PYTHON**

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn’t specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. Python is commonly used for developing websites and software, task automation, data analysis, and data visualization.

Since it’s relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances. Python was designed for readability and has some similarities to the English language with influence from mathematics. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. Python relies on indentation, using whitespace, to define scope, such as the scope of loops, functions and classes.

**2.2.1 Python Libraries**

**SciPy**

SciPy, a scientific library for Python is an open source, BSD-licensed library for mathematics, science and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The main reason for building the SciPy library is that it should work with NumPy arrays. It provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. This is an introductory tutorial, which covers the fundamentals of SciPy and describes how to deal with its various modules.

The SciPy library supports integration, gradient optimization, special functions, ordinary differential equation solvers, parallel programming tools, and many more. We can say that SciPy implementation exists in every complex numerical computation.

**NumPy**

NumPy is often used along with packages like [SciPy](https://www.mygreatlearning.com/blog/scipy-tutorial/)(Scientific Python) and [Matplotlib](https://www.mygreatlearning.com/academy/learn-for-free/courses/python-matplotlib?post=23244)(plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language. It is open-source, which is an added advantage of NumPy.The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index. Every item in a ndarray takes the same size as the block in the memory. Each element in ndarray is an object of the data-type object (called dtype).

Standard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, pip.

Pip3 NumPy

The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python).

Tornado

Tornado is a Python web framework and asynchronous networking library, originally developed at [Friend Feed](https://en.wikipedia.org/wiki/FriendFeed). By using non-blocking network I/O, Tornado can scale to tens of thousands of open connections, making it ideal for [long polling](http://en.wikipedia.org/wiki/Push_technology#Long_polling), [WebSocket’s](http://en.wikipedia.org/wiki/WebSocket), and other applications that require a long-lived connection to each user. The Tornado web framework and HTTP server together offer a full-stack alternative to [WSGI](http://www.python.org/dev/peps/pep-3333/). While it is possible to use the Tornado HTTP server as a container for other WSGI frameworks ([**WSGIContainer**](https://www.tornadoweb.org/en/stable/wsgi.html#tornado.wsgi.WSGIContainer)), this combination has limitations and to take full advantage of Tornado you will need to use Tornado’s web framework and HTTP server together. A coroutine library ([**tornado.gen**](https://www.tornadoweb.org/en/stable/gen.html#module-tornado.gen)) which allows asynchronous code to be written in a more straightforward way than chaining callbacks. This is similar to the native coroutine feature introduced in Python 3.5 (async def).

Vim

[Vim](https://www.vim.org/) ([source code](https://github.com/vim/vim)), short for Vi Improved, is a configurable text editor often used as a Python development environment. Vim proponents commonly cite the numerous plugins, Vimscript and logical command language as major Vim strengths.  developers are more productive when they avoid taking their hands off the keyboard. Code should flow naturally from the developer's thoughts through the keyboard and onto the screen. Using a mouse or other peripheral is a detriment to the rate at which a developer's thoughts become code. This "efficiency by keyboard" [keeps Vim as one of the most popular text editors](https://pragmaticpineapple.com/how-did-vim-become-so-popular/) despite having been around for decades. Few programming tools have that kind of staying power.Vim has a logical, structured command language. When a beginner is learning the editor, she may feel like it is impossible to understand all the key commands. However, the commands stack together in a logical way so that over time the editor becomes predictable.

Nodejs

Node.js is an event-based, non-blocking, asynchronous I/O framework that uses Google's V8 JavaScript engine. It is used for developing applications that make heavy use of the ability to run JavaScript both on the client, as well as on server side and therefore benefit from the re-usability of code and the lack of context switching. It is open-source and cross-platform. Node.js applications are written in pure JavaScript and can be run within Node.js environment on Windows, Linux etcNode.js = Runtime Environment + JavaScript Library

All APIs of Node.js library is asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

**2.2.2 Indentation:**

Indentation is the leading whitespace ( spaces and tabs ) before any statement in [python](https://www.scaler.com/topics/python/). The reason why indentation is important in python is that the indentation serves another purpose other than code readability. Python treats the statements with the same indentation level (statements with an equal number of whitespaces before them) as a single code block. So whereas in languages like c, c++, etc. a block of code is represented by Curly braces { }, in python a block is a group of statements that have the same Indentation level i.e same number of leading whitespaces.

Indented statements should have an attaching statement; for instance, all the statements indented below form a block and belong to the if statement. This is applicable for while, for, functions, classes, etc. in python. The below example makes this point clear. The first line of python code cannot have an indentation. Indentation is mandatory in python to define the blocks of statements. The number of spaces must be uniform in a block of code.

It is preferred to use whitespaces instead of tabs to indent in python. Also, either use whitespace or tabs to indent; intermixing of tabs and whitespaces in indentation can cause wrong indentation errors.

**2.2.3characteristics of python**

* Easy to Learn and Use. Python is easy to learn as compared to other programming languages
* Expressive Language
* Interpreted Language
* Cross-platform Language
* Free and Open Source
* Object-Oriented Language
* Extensible
* Large Standard Library

**2.2.4Features of python**

* Easy to code
* Free and open source
* Object-Oriented language
* GUI programming support
* High-level Language
* Extensible feature
* Python is a portable language
* Interpreted language
* Large Standard Library
* Dynamically Typed language
* Automation is Easy

**2.3 HTML**

The Hypertext Markup Language or HTML is the standard [markup language](https://en.wikipedia.org/wiki/Markup_language) for documents designed to be displayed in a [web browser](https://en.wikipedia.org/wiki/Web_browser). It can be assisted by technologies such as [Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) and [scripting languages](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript).

[Web browsers](https://en.wikipedia.org/wiki/Web_browser) receive HTML documents from a [web server](https://en.wikipedia.org/wiki/Web_server) or from local storage and [render](https://en.wikipedia.org/wiki/Browser_engine) the documents into multimedia web pages. HTML describes the structure of a [web page](https://en.wikipedia.org/wiki/Web_page) [semantically](https://en.wikipedia.org/wiki/Semantic_Web) and originally included cues for the appearance of the document.

[HTML elements](https://en.wikipedia.org/wiki/HTML_element) are the building blocks of HTML pages. With HTML constructs, [images](https://en.wikipedia.org/wiki/HTML_element#Images_and_objects) and other objects such as [interactive forms](https://en.wikipedia.org/wiki/Fieldset) may be embedded into the rendered page. HTML provides a means to create [structured documents](https://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](https://en.wikipedia.org/wiki/Semantics) for text such as headings, paragraphs, lists, [links](https://en.wikipedia.org/wiki/Hyperlink), quotes and other items. HTML elements are delineated by *tags*, written using [angle brackets](https://en.wikipedia.org/wiki/Bracket#Angle_brackets).

Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) such as [JavaScript](https://en.wikipedia.org/wiki/JavaScript), which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[[2]](https://en.wikipedia.org/wiki/HTML#cite_note-deprecated-2) A form of HTML, known as [HTML5](https://en.wikipedia.org/wiki/HTML5), is used to display video and audio, primarily using the <canvas> element, in collaboration with JavaScript.

**2.4 CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents. In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL. In CSS, selectors declare which part of the markup a style applies to by matching tags and attributes in the markup itself. All elements of a specific typeid: an identifier unique within the document, identified with a hash prefix e.g. #idclass: an identifier that can annotate multiple elements in a document, identified with a period prefix e.g. classname. Classes and IDs are case-sensitive, start with letters, and can include alphanumeric characters, hyphens, and underscores. A class may apply to any number of instances of any elements. An ID may only be applied to a single element.

**2.5 Apache server:-**

Apache is a popular open-source, cross-platform web server that is, by the numbers, the most popular web server in existence. It’s actively maintained by the Apache Software Foundation. Some high-profile companies using Apache include Cisco, IBM, Salesforce, General Electric, Adobe, VMware, Xerox, LinkedIn, Facebook, Hewlett-Packard, AT&T, Siemens, eBay, and many more (source).In addition to its popularity, it’s also one of the oldest web servers, with its first release all the way back in 1995. Many cPanel hosts utilize Apache today. Like other web servers, Apache powers the behind-the-scenes aspects of serving your website’s files to visitors. Because Apache doesn’t perform as well in some benchmarks, especially for static websites or websites with high traffic, Kinsta uses the NGINX web server instead of Apache. Though NGINX hasn’t been around for as long as Apache, it’s quickly grown in popularity and market share since its launch in 2004.Apache is just one component that is needed in a web application stack to deliver web content. One of the most common web application stacks involves LAMP, or Linux, Apache, MySQL, and PHP.Linux is the operating system that handles the operations of the application. Apache is the web server that processes requests and serves web assets and content via HTTP. MySQL is the database that stores all your information in an easily queried format. PHP is the programming language that works with apache to help create dynamic web content. While actual statistics may vary, it’s fair to say a large portion of web applications run on some form of the LAMP stack because it is easy to build and also free to use. For the most part, web applications tend to generally have similar architecture and structure even though they serve many different functions and purposes. Most web applications also benefit from Firewalls, Load Balancers, Web Servers, Content Delivery Networks, and Database Servers. Firewalls help protect the web application from both external threats and internal vulnerabilities depending on where the firewalls are configured. Load Balancers help distribute traffic across the web servers which handle the HTTP(S) requests (this is where Apache comes in) and application servers (servers that handle the functionality and workload of the web app.) We also have Database Servers, which handle asset storage and backups. Depending on your infrastructure, your database and application can both live on the same server although it’s recommended to keep those separate.

**How to launch a webpage: -**

Launching a website is what you need to do after you've designed and built a website (on your computer), and before you tell the world about your internet-accomplishments. This can be a tricky process if you're not very comfortable with how websites are placed and hosted on the internet, so take care to learn the right way to do it the first time.

You probably know that the content of your website is its most important aspect, but it's also very important that its intended audience are able to find it, and that visitors to your site can quickly locate the information they are seeking. It's also a good idea to check that it's thoroughly optimised to attract new visitors and track their activities.

A page load begins when a user selects a hyperlink, submits a form, or types a URL in a browser. This is also referred to as the initial request or the navigation start. The user's action sends a request across the network to the web application server. The request reaches the application for processing.

## CHAPTER 3 COMMUNICATION PROTOCOLS

### INTRODUCTION

The communication network is responsible for data exchange between different entities. The application level is responsible for the management and control of EVs' charging. The types of monitoring data, traffic volume, communication requirements and network ownership are defined and discussed.

### CAN BUS

* A CAN acronym is Controller Area Network, which is a robust vehicle bus standard designed to allow microcontrollers and other devices to communicate with each other applications without any Host computer.
* In general CAN bus protocol is used for internal communication between devices in vehicle.
* For EV charging ,we use 11-bit can identifier with default speed of 500 kbps.
* CAN was developed by Robert bosch in 1986,the main advantage of CAN bus is no host required.
* It was a message based protocol

**3.2.1 The CAN standard**

CAN is an International Standardization Organization (ISO) defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for high immunity to electrical interference and the ability to self-diagnose and repair data errors. These features have led to CAN’s popularity in a variety of industries including building automation, medical, and manufacturing. The CAN communications protocol, ISO-11898: 2003, describes how information is passed between devices on a network and conforms to the Open Systems Interconnection (OSI) model that is defined in terms of layers. Actual communication between devices connected by the physical medium is defined by the physical layer of the model. The ISO 11898 architecture defines the lowest two layers of the seven layer OSI/ISO model as the data-link layer and physical layer in Figure 1.

**3.2.2 Standard CAN or Extended** **CAN**Diagram

Description automatically generated

The CAN communication protocol is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP). CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message. CD+AMP means that collisions are resolved through a bit-wise arbitration, based on a preprogrammed priority of each message in the identifier field of a message. The higher priority identifier always wins bus access. That is, the last logic high in the identifier keeps on transmitting because it is the highest priority. Since every node on a bus takes part in writing every bit "as it is being written," an arbitrating node knows if it placed the logic-high bit on the bus.

Standard CANTable

Description automatically generated

The meaning of the bit fields of Figure 2 are:

• SOF–The single dominant start of frame (SOF) bit marks the start of a message, and is used to synchronize the nodes on a bus after being idle.

• Identifier-The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.

• RTR–The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way, all data being used in a system is uniform.

• IDE–A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.

• r0–Reserved bit (for possible use by future standard amendment).

• DLC–The 4-bit data length code (DLC) contains the number of bytes of data being transmitted. • Data–Up to 64 bits of application data may be transmitted.

• CRC–The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.

• ACK–Every node receiving an accurate message overwrites this recessive bit in the original message with a dominate bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message and the sending node repeats the message after rearbitration. In this way, each node acknowledges (ACK) the integrity of its data. ACK is 2 bits, one is the acknowledgment bit and the second is a delimiter.

• EOF–This end-of-frame (EOF), 7-bit field marks the end of a CAN frame (message) and disables bitstuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.

• IFS–This 7-bit interframe space (IFS) contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area.

**Extended CAN**

As shown in Figure 3, the Extended CAN message is the same as the Standard message with the addition of: • SRR–The substitute remote request (SRR) bit replaces the RTR bit in the standard message location as a placeholder in the extended format. • IDE–A recessive bit in the identifier extension (IDE) indicates that more identifier bits follow. The 18-bit extension follows IDE. • r1–Following the RTR and r0 bits, an additional reserve bit has been included ahead of the DLC bit.

A picture containing table

Description automatically generated

### GB/T PROTOCOL

This standard specifies the definitions of physical layer, data link layer and application

layer of the Control Area Network (CAN)-based communication between off-board

conductive charger (hereinafter referred to as "charger") and battery management

system (hereinafter referred to as "BMS") for electric vehicle.

This standard is applicable to the communication between charger and BMS of

charging mode 4 specified in GB/T 18487.1 or that between charger and vehicle

control units having charging control function.

3.3.1 General GB/T

* The communication network between charger and BMS adopts CAN 2.0B

communication protocol. The charging process refers to Appendix A.

* During the charging, the charger and BMS monitor such parameters as voltage,

current and temperature meanwhile BMS shall manage the whole charging process.

* CAN communication network between charger and BMS shall be composed of

two nodes, i.e. charger and BMS.

* Data information is transported in the priority of low byte.Diagram

  Description automatically generated
* Positive current represents discharging while negative current represents

charging.

* Charger and BMS conforming to this standard should be capable of forward

compatibility.

3.3.2 Layers

Physical layer- Physical Layer conforming to this standard shall refer to ISO 11898-1:2003 and SAE

J1939-11: 2006. The communication between charger and BMS in this standard shall

use the CAN interface independent to power assembly control system. The

communication rate between charger and BMS may choose 250 kbit/s.

Note: 50 kbit/s communication rate may be adopted as agreement between power equipment manufacturer and electric vehicles manufacturer in the bad communication environment (for example commercial vehicle charging station with longer communication distance).

Data Link layer- frame format Equipment complying with this standard shall use 29-bit identifier of CAN extended frame, and the corresponding definition of each specific bit allocation shall meet the requirements as given in SAE J1939-21:2006.

**3.3.3 PDU frame format**

Each CAN data frame contains a single protocol data unit (PDU), as detailed in Table

Table

Description automatically generated

The protocol data unit is composed of seven parts which respectively are priority,reserved bit, data page, PDU format, specific PDU, source address and data field

Data format requirements:

1. P is priority: set from the highest 0 to the lowest 7.

2. R is reserved bit: it is used for the development in future, and is set as 0 in this standard.

3. DP means the data page: it is used to select the auxiliary page for the description of parameter group, and is set as

0 in this standard.

4. PF means PDU format: it is used to determine the PDU format as well as the corresponding parameter group

number of data field.

5. PS means the specific PDU format: PS value depends on PDU format. In this standard, PDU1 format is adopted

and the PS value is the destination address.

6. SA means source address: it is the source address sending this message.

7. DATA means the data field: if the preset parameter group data length is less than or equal to 8 bytes, message

length specified in Chapter 9 is used in the transport. If the preset parameter group data length is 9 to 1,785 bytes,

multiple CAN data frames are needed for data transport, and the communication between multiple packets of

parameter groups is established and closed depending upon the connection management ability of transport protocol,

detailed in 6.5 of this standard.

8.In this table, Line 3 refers to the number of bits.

**Application layer**-

7.1The application layer is defined in manner of parameters and parameter group.

7.2Parameter group is numbered by PGN, and each node identifies the content of

data packet according to PGN.

7.3"Request PGN" is used to actively obtain the parameter groups of other nodes.

7.4Data are transported in the form of periodical transport and event-driven mode.

7.5In case that multiple PGN data need to be transmitted in order to realize one

function, it requires receiving multiple PGN messages of this definition to judge the

successful transmission of this function.

7.6When defining new parameter group, the parameters of one function, the

parameters of the same or similar refresh frequency and the parameters belonging to

one subsystem shall be put into one parameter group as much as possible;

meanwhile, on one hand the new parameter group shall make the best of the data

width of 8 bytes and the relevant parameters shall be put into one group as much as

possible, and on the other hand the expansibility of the new parameter group shall be

fully considered, one byte or bit shall be reserved for future modification.

7.7When modifying the defined parameter group as given in Chapter 9, the

definition of defined byte or bit shall not be modified; the newly-added parameters

shall be relevant to the original parameters in parameter group and the irrelevant

parameters shall not be added into the defined PGN for purpose of saving the number

of PGNs.

7.8During charging process, the definition of various fault diagnosis for charger and

BMS shall comply with the requirements for CAN-bus diagnostic system as stated in

5.1 of SAE J1939-73:2006. The specification for definition of fault diagnosis

messages is detailed in Appendix B.

7.9The message options may be either mandatory or optional. If all the contents in

the same frame of message are optional, such message may not be transported; if

some contents in the same frame of message are optional, all the optional bits are

transported in the format as specified in this standard or filled with 1; the invalid bit or

field not specified in this standard is filled with 1. The bit or reserved bit not specified

in this standard is filled with 1.

**3.4** **OSI MODEL**

OSI stands for Open System Interconnection is a reference model that describes how information from a [software](https://www.javatpoint.com/software) application in one [computer](https://www.javatpoint.com/what-is-computer) moves through a physical medium to the software application in another computer, OSI consists of seven layers, and each layer performs a particular network function. OSI model was developed by the International Organization for Standardization (ISO) in 1984, and it is now considered as an architectural model for the inter-computer communications. OSI model divides the whole task into seven smaller and manageable tasks. Each layer is assigned a particular task. Each layer is self-contained, so that task assigned to each layer can be performed independently.Diagram

Description automatically generated

**Functions of the OSI Layers**

There are the seven OSI layers. Each layer has different functions. A list of seven layers are given below:

* **Physical Layer −** Its function is to transmit individual bits from one node to another over a physical medium.
* **Data Link Layer −** It is responsible for the reliable transfer of data frames from one node to another connected by the physical layer.
* **Network Layer −** It manages the delivery of individual data packets from source to destination through appropriate addressing and routing.
* **Transport Layer −**It is responsible for delivery of the entire message from the source host to destination host.
* **Session Layer −** It establishes sessions between users and offers services like dialog control and synchronization.
* **Presentation Layer −** It monitors syntax and semantics of transmitted information through translation, compression, and encryption.
* **Application Layer −** It provides high-level APIs (application program interface) to the users.

### Features Of The OSI Model

The various features of the OSI Model are enlisted below:

* Easy to understand the communication over wide networks through the OSI Reference Model architecture.
* Helps to know the details, so that we can get a better understanding of the software and hardware working together.
* Troubleshooting of faults is easier as the network is distributed in seven layers. Each layer has its own functionality; hence the diagnosis of the issue is easy and less time is taken.
* Understanding new technologies generation by generation becomes easier and adaptable with the help of the OSI Model.

**ENDIAN FORMAT**

Endianness is a term that describes the order in which a sequence of [bytes](https://www.techtarget.com/searchstorage/definition/byte) is stored in computer memory. Endianness can be either big or small, with the adjectives referring to which value is stored first. Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first, at the lowest storage address. Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first.

**HISTORY OF ENDIANNESS**

In 1980, computer scientist Danny Cohen applied the terms *big-endian* and *little-endian* to digital electronics in his article "On Holy Wars and a Plea for Peace."Big-endian and little-endian derive from Jonathan Swift's *Gulliver's Travels,*in which the Big Endian were a political faction that broke their eggs at the large end, or "the primitive way." They rebelled against the Lilliputian King, who required his subjects, the Little Endian, to break their eggs at the small end. While many mainframe computers are big-endian, most modern computers are little-endian. Endianness is initially an arbitrary decision by the semiconductor vendor that can have a long-term effect on a line of products. When vendors update their technology, they keep the existing endianness to help maintain backward compatibility. For example, the designers of the Motorola 68000 and the Intel 8086 (predecessor of the x86 family) chose their endianness in the 1970s and continue to use their respective endianness today.

**LITTLE-ENDIAN**

The little-endian convention is a type of addressing that refers to the order of data stored in memory. In this convention, the least significant bit (or "littlest" end) is first stored at address 0, and subsequent bits are stored incrementally. Little-endian is the opposite of big-endian, which stores the most significant bit first. Because they are opposites, it is difficult to integrate two systems that use different endian conventions.

Memory addresses may be viewed incrementally from left to right, with the leftmost address being the first address: address 0. In a little-endian system, the least significant bit is stored in address 0, and subsequent data is stored to the right in incremental address locations. Thus, data or bits stay within the same address where they are stored - even when new data is added.

For example, if data "0 1 2 3 4, 0" is stored in address 0, and each digit represents an incremental address, then 4 would be stored at address 4. This format is more logical for humans because we read, write and do most things from the left. The opposite is true for big-endian. Because the most significant bit should always be stored in address 0, all previous stored data that is of lesser significance is moved to the right to a higher address location.

## EXAMPLE

0x12674592 in 32-bit representation can be stored as –

To allow machines with different byte order conventions communicate with each other, the Internet protocols specify a canonical byte order convention for data transmitted over the network. This is known as Network Byte Order.

By these definitions, a 32-bit data pattern, which is regarded as a 32-bit unsigned integer. The "high- Order" byte is the one for the largest powers of 2: 231, ..., 224. The "low-order" byte is the one for the smallest powers of 2: 27, ..., 20.

zTable

Description automatically generated

## CHAPTER 4 BATTERY MANAGEMENT SYSTEM

### INTRODUCTION

### A battery management system (BMS) is any electronic system that manages a [rechargeable battery](https://en.wikipedia.org/wiki/Rechargeable_battery) ([cell](https://en.wikipedia.org/wiki/Electrochemical_cell) or [battery pack](https://en.wikipedia.org/wiki/Battery_pack)), such as by protecting the battery from operating outside its [safe operating area](https://en.wikipedia.org/wiki/Safe_operating_area), monitoring its state, calculating secondary data, reporting that data, controlling its environment, authenticating it and / or [balancing](https://en.wikipedia.org/wiki/Battery_balancing) it.

As an energy source of mobile system including robot, Li battery is extensively applied. To increase durability and efficient usage of battery, it is to require the rigorous management to battery controller, BMS, of them, the exact estimation of SOC is very important for correct usage.

Diagram

Description automatically generated

The major functions of BMS are to protect from over voltage, over current ,over temperature and under voltage ,to maintain voltage balance of battery cells and to estimate of internal states of battery such as State of charge(SOC),StateofHealth(SOH).

The major task of a battery management system (BMS) is to provide security and longevity of the battery. This can be done through continuous monitoring and control of the battery's state-of-charge . **4.2 STATE OF CHARGE (SOC)**

* The State of charge (SOC) has an important role in determining the remaining capacity of the battery pack. Accurate estimation of the SOC is very complex and difficult to implement, because of the limited battery model
* The SOC of a battery is defined as the ratio of its current capacity (Q(t)) to the nominal capacity(Qn). Nominal capacity, Qn, is the maximum amount of charge that can be stored in the battery. This rating is given by the manufacturer.

SOC(t)=Q(t)/Qn

* In the other words, state of charge means the ratio of the remaining charge of the battery to the total charge while the battery is fully charged at the same specific standard condition.
* During charging and discharging, battery internal parameters like resistance, temperature, etc,. Vary with SOC, so these parameters are shown useful for SOC estimation. SOC is expressed in percentage.

SOC = 100% battery fully charged

SOC = 0% battery fully discharged

**4.3 STATE OF HEALTH (SOH)**

* Battery State Of Health (SOH) is an important indicator of the battery’s life. SOH reflects the ability of a battery to deliver and receive energy and power.
* The SOH of a battery is defined as the ratio of its maximum instantaneous releasable capacity, (Qmax(t)) to the capacity of the new battery(Qnew).

SOH (t) = Qmax(t)/Qnew

* State of Health (SOH) is a figure of merit of the present condition of a battery cell( or a battery module, or a battery system), compared to its ideal conditions.
* The Soh is represented in percentage form. A SOH equal to 100% means it is a fresh/ new battery.
* The SOH could be derived by capacity and the internal resistance, and it could also be derived from other battery parameters like AC impedance, self-discharging rate, and power density.

## 4.4 Design of BMS

BMS for Li battery is basically to measure the physical values such as current, voltage and temperature from its sensor. Especially, in case of voltage, each cell voltage connected in series is measured from an exclusive chip made by some of company. This chip includes a circuit for balancing among cells. Fig. 1 shows a configuration of battery pack in the viewpoint of BMS.

The major functions of BMS are as follows.

To protect the critical operations points of battery which include overvoltage, under voltage, over current, temperature etc.

1. to control the cell balance and temperature.
2. to gauge the states through estimation algorithm: SOC, SOH, SOP.
3. to communicate with supervising controller for diagnosis and charging.

## 4.5 Estimation of SOC

### 4.5.1 Electrical model of battery

A general model of battery for state estimation consists of two components, OCV (open circuit voltage) and overpotential. OCV is a difference of potential between electrodes in the electrochemical equilibrium state. Overpotential is a voltage drop caused by internal resistances which is determining from the rate of electrochemistry reaction in the inside of battery.

This paper uses a simplified electrical model which still remains the major characteristics. It reduces the complexity.

Diagram, schematic

Description automatically generated

Fig. 2. An electrical model of battery

Where *Ri* denotes serial resistance of battery and *Vdiff* denotes a fast response such as the electric charge transfer between electrodes and electrolyte interface or a slow response.

On the one hand, OCV is noted as a steady state voltage source on the electrical model but actually a nonlinear component changing with SOC. The relationship between OCV and SOC is determined by the material of electrodes, etc and this paper uses the data obtained from 18650 battery as like Fig. 3.

Chart

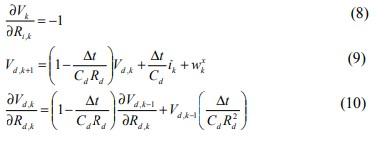
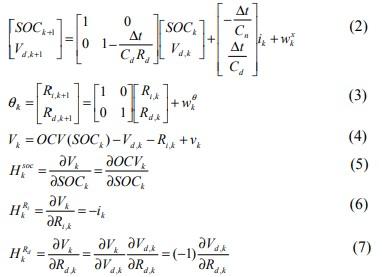
Description automatically generated with medium confidence

Fig.3. a typical OCV-SOC characteristics of 18650

### 4.5.2 Duel Extended Kalman Filter

From the electrical model of battery, state variables are *x* =[*SOC*( )*k* ;*Vdiff* ( )*k* ;*Rd k*( );*Cn* ]but, since the characteristics of

OCV-SOC is non-linear, an EKF (extended Kalman filter) which does linearization for each interval of SOC is needed. Additionally, due to computational complexity problem by 4-dimentional state space, the state space is divided by two state space, *x*1 =[*SOC*( )*k* ;*Vdiff* ( )*k* ] and *x*2 =[*Rd k*( );*Cn* ] and EKF is applied to each state space and each result output is provided as necessary information to each other. This is DEKF called in this paper[3,4,5].

**Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation:** With the rapid advances in energy storage technologies, the battery system has emerged as one of the most popular energy storage systems in stationary and mobile applications to reduce global carbon emissions. However, without proper monitoring and controlling of the batteries by a battery management system (BMS), problems concerning safety, reliability, durability, and cost will appear. The state-of-the-art BMS includes two main modules: BMS-Master and BMS-Slave, which can be combined in one embedded system or designed as two separate systems with wiring communication. While the BMS-Slave is responsible for monitoring the battery cells with signal acquisition and filtering, more advanced functions, such as battery diagnostic algorithms, which require high computation power, are implemented in the BMS-Master.

**Cloud battery management system**

By bridging the physical and the virtual world, digital twin allows the virtual entity to the battery systems simultaneously with the seamless transmission of data. Compared with the onboard BMS, the cloud BMS has advantages in both hardware and software, as summarized in Table. In terms of hardware, the cloud BMS has high computation power, enormous data storage capability, and high system reliability. These features further support the application of advanced algorithms in software. On the one hand, the performance of the functions, which already exist in onboard BMSs, can be further improved with more advanced algorithms. On the other hand, new functions, such as data-based lifetime prediction and system optimization, which are hard to be implemented in onboard BMSs, can be implemented in the cloud.

**Summary of advantages of cloud BMS**

Advantages

Hardware

High computation power

Enormous data storage capability

High system reliability

Software

Accurate monitoring and diagnostics

Reliable prognostics and optimization

Graphical user interface

Description automatically generated

Schematic of the cloud BMS, which consists of six subsystems: the battery systems for data generation, the BMS-Slave for data sensing, IoT component for data collection, cloud for data storage, application programming interface (API) for data analytics and user interface (UI) for data visualization.

**BMS-Slave**

The main components in the developed BMS-Slave for data sensing are multi-cell battery monitors LTC 6804G-2 and LPC 11C14F/301. The LTC measures up to 12 battery cells connected in series with a total measurement error of less than 1.2 mV, which can also be connected in series to monitor high-voltage battery strings. The LPC is a 32-bit ARM Cortex-M0 based microcontroller, which sends the measurement commands to the LTC and receives the measured data in hexadecimal values from the LTC. The BMS-Slave performs data acquisition by measuring the voltage, current, and temperature of the battery cells with sensors at different sampling rates.

**IoT Component**

The basic idea behind IoT is to make the devices, i.e., the stationary and mobile battery systems embedded with electronics and connected with the internet, communicate and interact with others to be monitored and controlled remotely. Therefore, a stable internet connection is vital for a stable real-time data transfer between the battery system and the digital twin. In order to increase the reliability of the whole system, the functions which are required at each time point during operation should also run locally, guaranteeing the system safety. An advanced version of these functions will run in the cloud with advanced algorithms, which provide higher accuracy while requiring high computation power. The functions which require historical operation data can only run in the cloud. The digital twin will communicate with the battery system and update the model parameters to improve the system performance and restrain battery degradation.

**Cloud**

As IoT devices usually have limited storage and computation capabilities and do not allow complex data processing, cloud computing with virtually unlimited storage capability and processing power enables the scalable and real-time data analysis of the IoT devices. The proposed cloud platform in the cloud BMS consists of a data logger and a database hosted in Germany. The data logger captures a massive amount of non-structured or semi-structured data produced by the battery systems and enables a secure gateway and data transfer into the cloud database, which was extensively certified and has distributed denial-of-service (DDoS) protection and multi-redundant connections. Only the owners and operators of the battery systems have access to the data, and data privacy and security are guaranteed.

**A brief review on key technologies in the battery management system of electric vehicles**

Electric vehicles (EVs) and hybrid electric vehicles (HEVs) have been widely regarded as the most promising solutions to replace the conventional internal combustion (IC) engine-based vehicles, and the recent years have seen

Battery types and key technologies for BMS In EV applications, many types of battery can be adopted as the power supply. There are number of functional modules in the BMS. Some popular battery types and key technologies for BMS are analysed and summarized in this paper.

Battery types in EVs

Batteries are generally grouped into two categories based on the ability of recharging: Primary and secondary battery. The primary battery can be just used once after being fully discharged, and the secondary battery is capable of being recharged after discharging process. For the applications of EVs and HEVs, the secondary battery with long cycle life, small energy loss, high power density and enough safety level is indispensably required. Some popular types of batteries used in EVs include lithium-ion (Li-ion), lead acid, nickel-cadmium (NiCd) and nickel metal hydride (NiMH), etc. Table 1 illustrates some key characteristics for these popular battery types. It is clearly shown that Li-ion battery is significantly better than other types of battery, especially in terms of large cycle life which is key to long service of EVs (e.g., 6–10 years’ services life). Besides, Li-ion battery is also composed of eco-friendly materials without toxic gassing problem and has high safety level. Therefore, Li-ion battery becomes a most popular power supply for EVs

**4.6 Key technologies for BMS**

| Battery type | Service life/cycle | Nominal voltage/V | Energy density  /(W$h$kg–1) | Power density  /(W$kg–1) | Charging efficiency/% | Self-discharge rate  /(%$month–1) | Charging temperature/oC | Discharging temperature/oC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Li-ion battery | 600–3000 | 3.2–3.7 | 100–270 | 250–680 | 80–90 | 3–10 | 0 to 45 | –20 to 60 |
| Lead acid battery | 200–300 | 2.0 | 30–50 | 180 | 50–95 | 5 | –20 to 50 | –20 to 50 |
| NiCd  battery | 1000 | 1.2 | 50–80 | 150 | 70–90 | 20 | 0 to 45 | –20 to 65 |
| NiMH  battery | 300–600 | 1.2 | 60–120 | 250–1000 | 65 | 30 | 0 to 45 | –20 to 65 |

# Battery modelling

Building a proper model is usually the starting point for BMS design, control and optimization. Over the years, numerous battery models with various levels of accuracy and complexity have been developed. These models can be primarily categorized as the battery electric model, battery thermal model, and battery coupled model, which are detailed in Fig. 2. Other model types such as battery kinetic models that are far less used in BMS are not covered in this paper.Diagram

Description automatically generated

**Battery electric model**

**The relation of key technologies in the BMS**

Graphical user interface, application

Description automatically generated

Three classifications of battery modelling

**Traditional battery charging approach**

There are some traditional but popular charging approaches to solve battery charging problem with numerous objectives and termination conditions. Four traditional charging approaches that have been widely utilized to charge batteries in EVs are listed in Fig. 4. These typical approaches can be mainly classified as constant current (CC) charging, constant-voltage (CV) charging, constant-current-constant-voltage (CC-CV) charging and multi-stage constant-current (MCC) charging. In the following, a particular emphasis is place up on the CCCV charging and MCC charging approaches.

The CC charging is a simple but rough approach which adopts a small constant current rate to charge battery during the whole charging process.

A picture containing text, businesscard

Description automatically generated

Traditional charging approaches for battery in EVs

## CHAPTER 5

## LITERATURE SURVEY

**[1] Communication Protocols between Off-Board Conductive Charger and Battery Management System for Electric Vehicle.** **China Electricity Council, Xuji Group Corporation, China Electric Power Research Institute.**

This standard is drafted in accordance with the rules given in GB/T 1.1-2009. This standard replaces GB/T 27930-2011 Communication Protocols between Off-board Conductive Charger and Battery Management System for Electric Vehicle. The charger and BMS conforming to this standard should be capable of forward compatibility. Overview flowchart is changed Communication, handshake message BHM and CHM ,8bytes are reserved for BRM. From this paper we came to know that how communication is going, what are the formats for communication and what are the messages the charger and BMS send to each other in communication process.

**[2] BMS Influence on Li-ion Packs Characterization and Modeling S. Castano, D. Serrano-Jimenez, J. Sanz Electrical Engineering Department Carlos III University of Madrid Leganes, Spain.**

In this paper the effect of the BMS on the performance of a Li-ion battery module by means of charge, discharge and frequency response tests. The BMS effects on Li-ion battery pack characterization and modeling has been analyzed. For this purpose, charge, discharge and EIS tests have been carried out. From these experimental results a pack model that considers BMS effect has been realized. Besides, a pack model based on a single cell approach has been calculated. In order to determinate what model can reproduce the pack behavior a HIL simulation of an electric vehicle has been performed. Simulation results show that the model which includes BMS effect reproduces the pack voltage response more accurately than the model based on a single cell approach. Thus, it can be concluded that BMS effects over the module performance should always be considered whenever a high degree of accuracy is required

**[3] Kailong LIU, Kang LI, Qiao PENG, Cheng ZHANG A brief review on key technologies in the battery management system of electric vehicles.**

In this paper we came to know that batteries have been widely applied in many high-power applications, such as electric vehicles (EVs) and hybrid electric vehicles, where a suitable battery management system (BMS) is vital in ensuring safe and reliable operation of batteries. This paper aims to give a brief review on several key technologies of BMS, including battery modelling, state estimation and battery charging. First, popular battery types used in EVs are surveyed, followed by the introduction of key technologies used in BMS. Various battery models, including the electric model, thermal model and coupled electro-thermal model are reviewed. Then, battery state estimations for the state of charge, state of health and internal temperature are comprehensively surveyed. Finally, several key and traditional battery charging approaches with associated optimization methods are discussed.

**[4] SOC Estimation and BMS Design of Li-ion Battery Pack for Driving Youngryul Kim, Sung Hyun Yun and Junho Lee, June 28 - July 1, 2017.**

In this paper as an energy source of mobile system including robot, Li battery is extensively applied. To increase durability and efficient usage of battery, it is to require the rigorous management to battery controller, BMS, of them, the exact estimation of SOC is very important for correct usage. In this introducing the design of BMS simply and, as an estimation method of SOC. It introduced the hardware configuration of BMS for a Li battery pack and proposed DEKF as a state estimation method. The proposed method has verified its validity through simulation with using the real data on current and voltage of battery. Especially, the proposed algorithm is implemented to do real time calculation for real controller.

**[5] Cloud battery management system with online state-of-charge and state-of-health estimation Weihan Li, a, b, Monika Rentemeister, Julia Badedae, Dominik Jösta,b , Dominik Schultef , Dirk Uwe Sauer ,volume 30,August 2020.**

Battery management is critical to enhancing the safety, reliability, and performance of the battery systems. This paper presents a cloud battery management system for battery systems to improve the computational power and data storage capability by cloud computing. With the Internet of Things, all battery relevant data are measured and transmitted to the cloud seamlessly, building up the digital twin for the battery system, where battery diagnostic algorithms evaluate the data and open the window into battery’s charge and aging level.  The application of equivalent circuit models in the digital twin for battery systems is explored with the development of cloud-suited state-of-charge and state-of-health estimation approaches.

**[6] Intelligent algorithms and control strategies for battery management system in electric vehicles: Progress, challenges and future outlook M.S. Hossain Lipu, M.A. Hannan, Tahia. Karim, Aini Hussain, Mohamad Hanif Md Saad , Afida Ayoub , Md. Sazal Miah , T.M. Indra Mahlia .**

This paper tells us about the research on battery technology in electric vehicle applications is advancing tremendously to address the carbon emissions and global warming issues. The effectiveness of electric vehicles depends on the accurate assessment of key parameters as well as proper functionality and diagnosis of the battery storage system. However, poor monitoring and safety strategies of the battery storage system can lead to critical issues such as battery overcharging, over-discharging, overheating, cell unbalancing, thermal runaway, and fire hazards. To address these concerns, an effective battery management system plays a crucial role in enhancing battery performance including precise monitoring, charging-discharging control, heat management, battery safety, and protection. This paper delivers a comprehensive review of different intelligent approaches and control schemes of the battery management system in electric vehicle applications. In line with that, the review evaluates the intelligent algorithms in battery state estimation concerning their features, structure, configuration, accuracy, advantages, and disadvantages. Moreover, the review explores the various controllers in battery heating, cooling, equalization, and protection highlighting categories, characteristics, targets, achievements, benefits, and shortcomings.

**[7] A Study on the Open Circuit Voltage and State of Charge Characterization of High Capacity Lithium-Ion Battery Under Different Temperature Ruifeng Zhang, Bizhong Xia, Baohua Li, Libo Cao , Yongzhi Lai, Weiwei Zheng , Huawen Wang, Wei Wang and Mingwang Wang.**

This paper focuses on the study of the OCV-SOC characteristics of high-capacity batteries under the influence of different temperatures. The result shows the OCV-SOC characteristic curve is greatly influenced by the temperature change. The polynomial fitting of the model is clear and simple so that it is widely applied in engineering. In the battery modeling, exponential, polynomial, sum of sin functions model, and Gaussian model are compared. In these models, accurate fitting of OCV-SOC curves in low SOC interval is a key and difficult point in battery state estimation, which has a great influence on the accuracy of battery state estimation.

**[8] Standardization of protocol for charging Infrastructure, Dr. Ashok Jhunjhunwala, Pravin Agrawal,15th May 2017.**

In this paper we came to know about what EV charger is and what are the different types of EV chargers. What the EV charger do. AS we are having different types of chargers such as private charger, public charger, Ac chargers and Dc charger etc., Communication for Chargers, also called EV Supply Equipment (EVSE) is explained in detail. Explains about system structure and what are the requirements needed for the EV charger.

## CHAPTER 6

## METHODOLOGY

**5.1 VIRTUAL BOX**

VirtualBox is open-source software for virtualizing the x86 computing architecture. It acts as a hypervisor, creating a VM (virtual machine) where the user can run another OS (operating system). The operating system where VirtualBox runs is called the "host" OS. The operating system running in the VM is called the "guest" OS. VirtualBox supports Windows, Linux, or macOS as its host OS.

VirtualBox was originally developed by Innotek GmbH, and released on January 17, 2007, as an open-source software package. The company was later purchased by Microsystems. On January 27, 2010, Oracle Corporation purchased Sun and took over development of VirtualBox.

**5.2 UBUNTU**

Linux is an operating system in much the same way that Windows is an operating system (and there any similarities between Linux and Windows end). The term operating system is used to describe the software which acts as a layer between the hardware in a computer and the applications that we all run on a daily basis. When programmers write applications, they interface with the operating system to perform such tasks as writing files to the hard disk drive and displaying information on the screen.

Richard Stallman at the Free Software Foundation, a strong advocate of free and open-source software, was working on an open-source operating system of his own. Rather than focusing initially on the kernel, Stallman decided to begin by developing all the tools, utilities and compilers necessary to use and maintain an operating system.

**5.3 INSTALLATION PROCESS**

STEP 1: Install VirtualBox software using the provided link below

<https://download.virtualbox.org/virtualbox/6.1.34/VirtualBox-6.1.34-150636-Win.exe>.

**Fig 5.1 Image of VirtualBox**Graphical user interface, diagram

Description automatically generated

STEP 2: Install ubuntu20.04 software using the link provided below

<https://ubuntu.com/wsl>.

STEP 3: After installing the VirtualBox and ubuntu in the system we have to give some system specifications about the how much memory and what type of VirtualBox and ubuntu are needed.Graphical user interface, text, application, email

Description automatically generated

**Fig 5.2 Memory allocation for ubuntu**

STEP 4: Now start the ubuntu machine after finishing all the initial setup.

STEP 5: After setting up click on ubuntu start and then going into ubuntu click on apps then search for terminal.

STEP 6: Now we have to do scripting in the terminal for this we have to right the script line to update the setup. we have to the script as sudo apt -get update to update the setup.

STEP 7: After the system get updated, now we have to install can utils in the terminal by using the script sudo apt-get install can-utils.Text

Description automatically generated

**Fig 5.3 Image of Commands for updating terminal**

STEP 8: After the installation of can utils, lets install python3.8 in the system by using the script code sudo apt-get install python3.8-dev. This installation takes a little bit more time for this entire process we should have good internet connection.

STEP 9: Next in the process of installation we have install the python pip which is used to install packages. Package Installation Using Pip in Python, pip is a tool to install packages for later usage in a program. It works by connecting with an online repository that includes public and private packages and is known as the Python Package Index. The script for this installation is sudo apt-get install python3-pip.

STEP 10: The next step includes the installation of python-can. The can package provides controller area network support for Python developers, providing common abstractions to different hardware devices, and a suite of utilities for sending and receiving messages on a can bus. The script for this installation is sudo pip3 install python-can.

STEP 11: Next step is to install idle python3.8 this for extracting and running the python codes in the system. The script used here is sudo apt-get install idle-python3.8.

STEP 12: First of all place all the codes in a folder and open the terminal from the python codes folder.

STEP 13: Now we have to install the SciPy library. The SciPy library of Python is built to work with NumPy arrays and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install and are free of charge. Script is sudo pip3 install Scipy.

STEP 14: In this process next we have to install the chromium browser. Chromium for Linux is an open-source web browser project started by Google, to provide the source code for the proprietary Google Chrome browser.Text

Description automatically generated

**Fig 5.4 Image of installation of SciPy and chromium browser**

STEP 15: Next we have to install tornado. By using non-blocking network, I/O, Tornado can scale to tens of thousands of open connections, making it ideal for long polling, Web Sockets, and other applications that require a long-lived connection to each user. Tornado is one of the most prominent libraries in Python for developers who build high-performance, low latency web applications. Script for this is sudo pip3 install tornado.

STEP 16: In this step we have to install NodeJS npm. Node.js is an open-source, cross-platform JavaScript runtime environment with a focus on server-side and networking applications. Node.js allows developers to build fast, scalable network applications using easy-to-understand code. It runs on Windows OS, Mac OSX, Linux, Unix, and other operating systems.

npm is two things: first and foremost, it is an online repository for the publishing of open-source Node.js projects; second, it is a command-line utility for interacting with said repository that aids in package installation, version management, and dependency management.

Script for this is sudo apt-get install nodejs npm.Text

Description automatically generated

**Fig 5.5 Image of installation of tornado and nodejs**

STEP 17: Now we have to install ws in npm. The ws npm package is the de facto web socket library for Node.js. The ws package also includes a web socket client, which is useful for testing.

Script to write is npm install ws.

STEP 18: Next step includes the installation of socket can. The Socket CAN package is an implementation of CAN protocols (Controller Area Network) for Linux. CAN is a networking technology which has widespread use in automation, embedded devices, and automotive fields. While there have been other CAN implementations for Linux based on character devices, socket CAN uses the Berkeley socket API, the Linux network stack and implements the CAN device drivers as network interfaces. The CAN socket API has been designed as similar as possible to the TCP/IP protocols to allow programmers, familiar with network programming, to easily learn how to use CAN sockets. Script for npm install socket can.

**Fig 5.6 Image of installation if npm ws and socket can**Text

Description automatically generated

STEP 19: Now we have to install vim. Vim is a Unix text editor that's included in Linux, BSD, and macOS. It's known for being fast and efficient, in part because it's a small application that can run in a terminal (although it also has a graphical interface), but mostly because it can be controlled entirely with the keyboard with no need for menus or a mouse. Script is sudo apt vim.

STEP 20: The installation process is completed. We have installed the required software’s for this process.

**5.4 WORKING**

STEP 21: Now we will setup the process for starting the can bus for communication between charger and BMS. The sh command provides a secure encrypted connection between two hosts over an insecure network. This connection can also be used for terminal access, file transfers, and for tunneling other applications. The command for this is sh canbus.sh.

STEP 22: In this we will start the programming in Linux for the execution of the program. And the command for this is sh startchr.sh.

**Fig 5.7 commands for starting the process of EV charging**Text

Description automatically generated

STEP 23: Now let’s see how the overall charging process goes on. The whole charging process comprise six stages: completion of physical connection, auxiliary charging of low voltage, charging handshake stage, charging parameter configuration stage, charging stage and end-of-charging stage.

STEP 24: The charging process starts with physical connection of charger gun placing properly in the charger socket of the car.

STEP 25: After the physical connection of the charger gun, the communication is started in the system between charger and BMS. First off, all the communication initiates with charger handshake message from the charger to BMS. The charging handshake stage is classified into handshake initiation stage and handshake recognition stage. Start low-voltage auxiliary power supply when the charger and BMS have been in physical connection and charged, and enter the handshake initiation stage, at which, send the handshake message and carry out insulation monitoring. Then enter handshake recognition stage, at which they send the recognition message and determine the necessary information for the battery and charger. The CHM message and BHM message are added to be compatible to the products and used for the charger and BMS to judge the received text used by them at handshake initiation stage. Below table shows the what are the messages during the charger.

**Fig 5.8 overall charging process**

A screenshot of a phone

Description automatically generated with medium confidence

STEP 26: After completion of charging handshake stage, both charger and BMS will enter into the charging parameter configuration stage. During this stage, the charger will send message of its maximum output capability to BMS, and BMS will judge whether charging is feasible according to the charger's maximum output capability. In each stage, if the charger or BMS does not receive message from the other party or does not receive correct message within the stipulated time limit, it will be judged as timeout.Table

Description automatically generated

**Table 5.1.1 Classification of Messages During Charging Handshake Stage**

**Table 5.1.2 Classification of Messages During Charging Parameter Configuration Stage**Table

Description automatically generated

STEP 27: After completion of charging configuration stage, both charger and BMS will enter into charging stage. Throughout the charging stage, BMS will send battery charging demand to the charger, and the charger will regulate the charging voltage and charging current according to battery charging demand in order to ensure the normal proceeding of charging process. During the charging process, charger and BMS will mutually send their respective charging state.Table

Description automatically generated

**Table 5.1.3 Classification of Messages during Charging Stage**

STEP 28: After charger and BMS stop charging, they both will enter into the end-of-charging stage. During this stage, BMS will send the charging statistical data (including: initial SOC, final SOC, and minimum voltage and maximum voltage of battery) in whole charging process to the charger; after the charger receives the charging statistical data from BMS, it will send the output electric quantity of whole charging process, the cumulative charging time and other information to BMS, and finally stop the output of low-voltage auxiliary power supply.Table

Description automatically generated

**Table 5.1.4 Classification of Messages During End-of-Charging Stage**

STEP 24: Now the execution of the program is started. we will get a page which will give the temperature reading, current reading and percentage of battery in the car charger.Graphical user interface, application

Description automatically generated

**fig 5.9 output page**

STEP 25: If we want to see how the communication is going on we can use the command candump vcan0.which will show us the communication between charger and BMS.

**Fig 5.10 communication between charger and BMS vice versa**Table

Description automatically generated with low confidence

This is how the overall process takes place.

**CHAPTER-7**

**RESULTS AND DISCUSSIONS**

* 1. **CHARGER HTML PAGE**

Graphical user interface, text, application

Description automatically generated

* 1. **BATTERY MANAGEMENT SYSTEM HTML PAGE** Graphical user interface, application

     Description automatically generated

**7.3 AT SOC 50%**

Graphical user interface, application

Description automatically generated

Diagram

Description automatically generated

**7.4AT SOC 75%**

Graphical user interface, application

Description automatically generated

### A picture containing graphical user interface Description automatically generated

### AT SOC 90%

### Graphical user interface, application Description automatically generated

### Graphical user interface Description automatically generated

* 1. **COMMUNICATION BETWEEN CHARGER AND BMS**Table

     Description automatically generated with low confidence

**7.7 OUTPUT ANALYSIS**

The above images consists of Charger and Battery output html pages ,the outputs are obtained at different SOC levels, by varying the range bar in charger page ,such we applied the SOC level of BMS ,hence it charges from those Soc levels. We can also find the Voltage demand and Current demand, measured voltage, measured current in the output page. Whenever the charger is connected to the BMS ,the voltage, current demand was taken as inputs to charger and provides required voltage and current to BMS.

**CHAPTER 8**

**CONCLUSION**

Here we simulated the overall charging process of Electric vehicle, the output here we considered as state of charge and communication between the charger and Battery Management system. By following the process, the communication was established between the charger and BMS ,once the charging process has been initiated the messages start exchanging in between them, each message has been transferred as equivalent coded formats, Hence we achieved the communication between the Charger and BMS .By varying the SOC range bar in charger HTML page we observed the SOC levels changes in BMS HTML page.

**8.1Future Scope**

The future of the EV charging system can be more improved by integrating cloud in these process, hence we can know the total overview ,progress, results from anywhere by simply accessing the application or webpage .Now in simulation ,we had given inputs from the range bar of charger html page ,but in practical the car and charger has once connected ,if the car was connected to cloud or any network, we can easily access all the data which were related to the BMS and Charger.

# REFERENCES

[1] Communication Protocols between Off-Board Conductive Charger and Battery Management System for Electric Vehicle. China Electricity Council, Xuji Group Corporation, China Electric Power Research Institute.

[2] BMS Influence on Li-ion Packs Characterization and Modeling S. Castano, D. Serrano-Jimenez, J. Sanz Electrical Engineering Department Carlos III University of Madrid Leganes, Spain.

[3] Kailong LIU, Kang LI, Qiao PENG, Cheng ZHANG A brief review on key technologies in the battery management system of electric vehicles.

[4] SOC Estimation and BMS Design of Li-ion Battery Pack for Driving Youngryul Kim, Sung Hyun Yun and Junho Lee, June 28 - July 1, 2017.

[5] Cloud battery management system with online state-of-charge and state-of-health estimation Weihan Li, a, b, Monika Rentemeister, Julia Badedae, Dominik Jösta,b , Dominik Schultef , Dirk Uwe Sauer ,volume 30,August 2020.

[6] Intelligent algorithms and control strategies for battery management system in electric vehicles: Progress, challenges and future outlook M.S. Hossain Lipu, M.A. Hannan, Tahia. Karim, Aini Hussain, Mohamad Hanif Md Saad , Afida Ayoub , Md. Sazal Miah , T.M. Indra Mahlia .

7] A Study on the Open Circuit Voltage and State of Charge Characterization of High Capacity Lithium-Ion Battery Under Different Temperature Ruifeng Zhang, Bizhong Xia, Baohua Li, Libo Cao , Yongzhi Lai, Weiwei Zheng , Huawen Wang, Wei Wang and Mingwang Wang.

[8] Standardization of protocol for charging Infrastructure, Dr. Ashok Jhunjhunwala, Pravin Agrawal,15th May 2017.