# Implementation of Efficient Fast Charger Using Pulse-Charging Method and Optimized Temperature

By

Saffat Newaz Sadmani
17121097
Md. Tanvir Hossain
17121100
Md. Taohidul Haque Emon
17121004
M Mustafizur Rahman
17121042

A thesis submitted to the Department of Electrical and Electronic Engineering in partial fulfillment of the requirements for the degree of
Bachelor of Science in Electrical and Electronic Engineering

Department of Electrical and Electronic Engineering
Brac University
June, 2021

## Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

| | |
|---|---|
| **Saffat Newaz Sadmani** | **Md. Tanvir Hossain** |
| 17121097 | 17121100 |
| **Md. Taohidul Haque Emon** | **M Mustafizur Rahman** |
| 17121004 | 17121042 |

# Approval

The thesis titled “Implementation of Efficient Fast Charger Using Pulse-Charging Method and Optimized Temperature” submitted by

1. Saffat Newaz Sadmani (17121097)
2. Md. Tanvir Hossain (17121100)
3. Md. Taohidul Haque Emon (17121004)
4. M Mustafizur Rahman (17121042)

of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering on 9th June, 2021.

**Examining Committee:**

Supervisor:
(Member)

_____

Dr. A. K. M. Abdul Malek Azad
Professor, Department of Electrical and Electronic Engineering
Brac University

Program Coordinator:
(Member)

_____

Dr. Abu S.M. Mohsin
Assistant Professor, Department of Electrical and Electronic Engineering
Brac University

Departmental Head:
(Chair)

_____

Dr. Md. Mosaddequr Rahman
Professor, Department of Electrical and Electronic Engineering
Brac University

## Abstract

Draining batteries at a high speed is one of the modern problems we face due to the electrical devices' power consumption rate. Moreover, as we need to use them frequently, charging them quickly is the necessity of the era where we cannot pass a single moment without wireless electrical devices. To fulfill the need, fast charging concept came to an existence where we inject a larger amount of current or voltage in a controlled way. In view of the properties of the lithium-ion battery, it is used widely for fast charging. Pulse method, CC-CV method, Multistage constant current (MCC) etc. are some of the fast-charging methods. Though CC-CV method is used widely, it has some drawbacks like heating the battery. So, to overcome the drawbacks we use the pulse method where the temperature does not increase much but the speed of the charging increases. In the pulse method, we are using two algorithms simultaneously, optimal frequency search and optimal duty cycle search. We are also searching for ambient temperature to fast charge as temperature may hamper fast charging performance.



**Keywords:** *Fast charging, Pulse method, Optimal frequency, Duty cycle, Temperature*

## Dedication

This work is dedicated mainly to our parents for all their efforts and encouragement they have invested in us up until now. We would also dedicate the work to our supervisor who has been the instrumental in the development of the final work output.

## Acknowledgement

We would like to thank our teachers, our supervisor Professor Dr. A. K. M. Abdul Malek Azad and our mentors who have been very supportive during the time of our work development. Our sincere appreciation is towards BRAC University EEE department for funding our project. And finally, each and every member of this team for having the dedication

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

| | |
|---|---|
| **CC-CV** | Constant current - constant voltage |
| **CCS** | Combined Charging System |
| **CP-CV** | Constant power - constant voltage |
| **ECM** | Equivalent circuit model |
| **EV** | Electric vehicle |
| **SoC** | State of charge |
| **MAC** | Maximum acceptable current |
| **MOSFET** | The metal–oxide–semiconductor field-effect transistor |
| **MCU** | Microcontroller |
| **PWM** | Pulse Width Modulation |
| **OCV** | Open circuit voltage |
| **DVPG** | Duty varied pulse generator |
| **VFPCS** | Variable frequency pulse charge system |
| **DVVPCS** | Duty-varied voltage pulse-charge strategy |
| **PSU** | Power supply unit |

# Chapter 1

# Introduction

## 1.1 Background and Motivation

While buying a smartphone nowadays, we focus on its battery life or how long the battery will last as well as how fast we can charge them as it is one of the most concerning the specification. Likewise, most of the electric devices we use in recent times have batteries but it was not like that always. Alessandro Volta is the inventor of the first battery which was a dry cell. Since then, we have discovered the way to use electric devices with mobility. As time goes on, we have seen many types of batteries to ease our life and expand our use of electrical devices. From the beginning of the invention of the battery, the aim was to increase the battery's lifetime so that it could be used for a longer period of time [1]. For that purpose, many developments were made in the battery world. However, the success in increasing the capacity of the battery leads to another question as to the batteries at that time cannot be used more than once. Moreover, heavy metals like nickel, cadmium, mercury were being used to produce the battery [2]. Since the use of the battery was increasing rapidly so does the chemical waste was increasing day by day. These heavy metals were regarded as environmental contamination. On top of that, they are highly dangerous for the human body. To limit environmental pollution, it was needed to produce less battery. But the demand for the battery was sky high at that time. To tackle this complex problem the concept of rechargeable batteries came into its existence. Now, with this rechargeable batteries' concept, people can use a single battery many times. Thus, the environment is getting lesser chemical waste.

As the technological progress was happening, we have seen devices that need more power as they need to run more complex work and for a longer time. The batteries which could give such powers cannot recharge as fast as it discharges. Moreover, electric vehicles were introduced to

the world as it was moving towards renewable energy from non-renewable energy. To fuel an electric vehicle a battery is needed and it is needed to charge fast like the traditional oil-gas fueled vehicles [3, 4]. So, the fast-charging concept has become the demand of this era where we want to use more renewable energy and preserve the nonrenewable energy sources. For fast charging, we need to inject a larger current or flow higher voltage into the battery [5]. This fast-charging process causes the battery to heat up. Due to the battery's chemical properties, excessive heat leads to its damage. The capacity of the battery is reduced if it is damaged by the heat. It is seen that the life of the lead-acid battery cuts in half by increasing the battery temperature by 8°C (15°F) [6]. Therefore, many methods are being used to fast charge the battery where they focus on fast charging as well as to increase the longevity of the battery. Implementing fast-charging methods, it is found that not all kinds of batteries are suitable for fast charging. Some get heated much faster, some do not have the ability to receive a larger amount of current. Because of the properties of the lithium-ion battery, it is used widely for fast charging [7]. In this paper, we are using Pulse Charging Method to fast charge the battery with the optimal frequency and optimal duty cycle.

Mass is using CC-CV as their fast-charging method to charge their devices. As using the convenient fast charging method increases the battery heat, due to the unawareness we notice many cases where the user or the surrounding people are injured due to battery explosion while charging the battery [8].

Again, we should limit the production of the battery because of the use of heavy metals. To do so, we need the battery to last longer than before. We can also turn to li-ion battery as it has fewer heavy metals than other batteries. The battery's longevity depends on the heat it produces. If the battery is heated frequently then we can be sure that it will not work for long. It is shown that available fast-charging produces much heat which gradually decreases the battery's longevity [5]. So, we need a method that will not only charge faster than the current

fast-chargers but also maintain the safety of the consumer by not increasing the heat of the battery. Furthermore, increase the longevity of the battery will cause less battery dumping. Thus, less environmental pollution than today.

For the above-mentioned reason, we are going to implement a fast-charger with pulse method on li-ion battery which will increase the longevity of the battery as it will create less heat than the traditional charging method.

## 1.2 Literature Review

[5] describes about the numerous fast-charging protocols that are available. The most common one is the CC-CV (Constant-Current Constant-Voltage). Other protocols are MCC-CV, pulse charging, boost charging.

But it is seen that those protocols cannot be implemented in all kinds of battery. By describing the characteristics of lithium-ion batteries such as high energy density, long cycle life, low self-discharge, no memory effect, [7] shows why among the existing battery technologies, the lithium-ion battery is widely used. For these characteristics of the li-ion battery, it is good for fast charging.

From the battery's equivalent circuits shown in [9], we can get the idea of the battery's impedance. It is an obstacle to fast charging as it reduces the current injection to a battery. It is one of the aims of fast charging to reduce battery impedance as much as possible so that the charger can inject the maximum amount of the current. [9] shows the relation between impedance and frequency in terms of SoC where we can see that the battery impedance is lower when the frequency is higher and at a certain SoC there is a certain frequency that causes the lowest battery impedance which is called optimal frequency.

Polarization voltage is not good for the battery. To do fast charging, [10] suggests a protocol needs to be set such that polarization voltage does not increase. It is seen that when SoC is

between 20% to 80% polarization voltage is relatively small. So, fast charging is done within that range of SoC.

In pulse charging duty cycle search mode focuses on finding an appropriate length for the relaxation period, which may be a value that is neither too short in diffusion nor too long in extending the charging time according to [10].

A battery produces heat in both cases while charging and discharging. Excessive temperature is the enemy of battery life [10]. Moreover, it increases the risk of a battery explosion.

If we want to get the most efficient and safe way to fast charge, we need to use both optimal frequency search mode and duty cycle search mode [10].

[10] also shows us the algorithm to obtain optimal frequency and duty cycle to pulse charge a battery which can be implemented by a microcontroller. This method reduces battery heating and charging time.

## 1.3 Problem Formulation

Among the numerous battery fast-charging strategies, CC-CV (constant current- constant voltage) is the most used. But it is not ideal in terms of charging time and temperature rise. There are strategies which can make a battery charge faster by injecting higher current but it results excessive heat in a battery which results gradual battery degradation [8].

Degraded battery loses its capacity thus it becomes unusable. As battery has heavy metal in it the disposal of the battery causes environmental pollution. Moreover, it produces higher temperature as degrading increases battery's internal resistance. At a certain point due to excessive heat battery caught into fire resulting fatal injuries.

So, these two simultaneous problems need to be solved in order to charge a li-ion battery in the most efficient way.

## 1.4 Overview of Entire Work

Introduction about fast charging and its methods were covered in the 2nd chapter. Discussion about the algorithm used to fast charge with pulse method where optimal frequency and optimal duty cycle is searched is in chapter 3. Then, chapter 4 discusses the hardware implementation of the project. Some field tests are described in chapter 5. And finally, chapter 6 concludes the work and describes the future work which can be done.

# Chapter 2

# Fast Charging Fundamentals

## 2.1 Introduction

Charging is essential for the battery to be used once it has lost all or some of its power. That's why we use a rechargeable battery instead of normal batteries which cannot be charged once its power is lost. SoC (state of charge) is used to determine whether a battery is fully charged or not. It is expressed in percentage form, from 0% to 100%. We charge the battery to increase the SoC. If we want to differentiate the charging with respect to charging speed, then it can be separated into two parts. One is slow charging and the other one is fast charging. Slow charging usually happens overnight taking 5 to 6 hours to charge. Charging the battery in lesser time is considered fast charging. It is happened by providing more voltage or current than it is given in normal charging. Many methods are being used to achieve this goal. Methods are also developing to shorten the charging time more and more. As a large amount of current is pushed into the battery, the battery gets heated which is the reason for the battery degradation. As a result, the methods which are being used are also taking this issue into the account when trying to implement fast charging. Polarization voltage is another important aspect to look for in the fast-charging method. Because of the polarization voltage, it not possible to fast charge a battery from 0% SoC to 100% SoC. The study shows polarization voltage is much higher when SoC is less than 20% and higher than 80%. In that state, if we want to fast charge the battery will get damaged.

## 2.2 Methods

From the above part, we get to know that there are several methods by which we can fast charge a battery. Some of them are Constant Current - Constant Voltage (CC-CV), Multi-State Constant Current - Constant Voltage (MCC-CV), pulse charging, CP-CV, boost charging,

VCP, etc. As CC-CV is the most widely used charging protocol used around the world for fast charging purposes, we will be covering this in another sub chapter. And as we are implementing pulse charging, we will be described later. An extended version of CC-CV is used in MCC-CV where the constant current phase is divided into several stages. Until the terminal voltage reaches a specified value in a certain stage, a constant current is pushed into the battery. Later CV phase starts and the charging ends when the current reaches zero. Though the method charges the battery faster than the traditional method it needs a complex calculation to determine each stage's charging current. In the CP-CV method, constant power is given at first then constant voltage was kept.

### 2.2.1 CC-CV Method

CC-CV is the method which is the most used method and by looking at the name we can assume that it is of two phases; constant current phase and constant voltage phase. In the first phase constant current is injected into the battery until it reaches up to the cut-off voltage. Then, the CC phase stops, and the CV or constant voltage phase starts. This time the voltage of the battery is kept constant while the current injection decreases. When the current stops flowing, it is implied that the battery is fully charged.

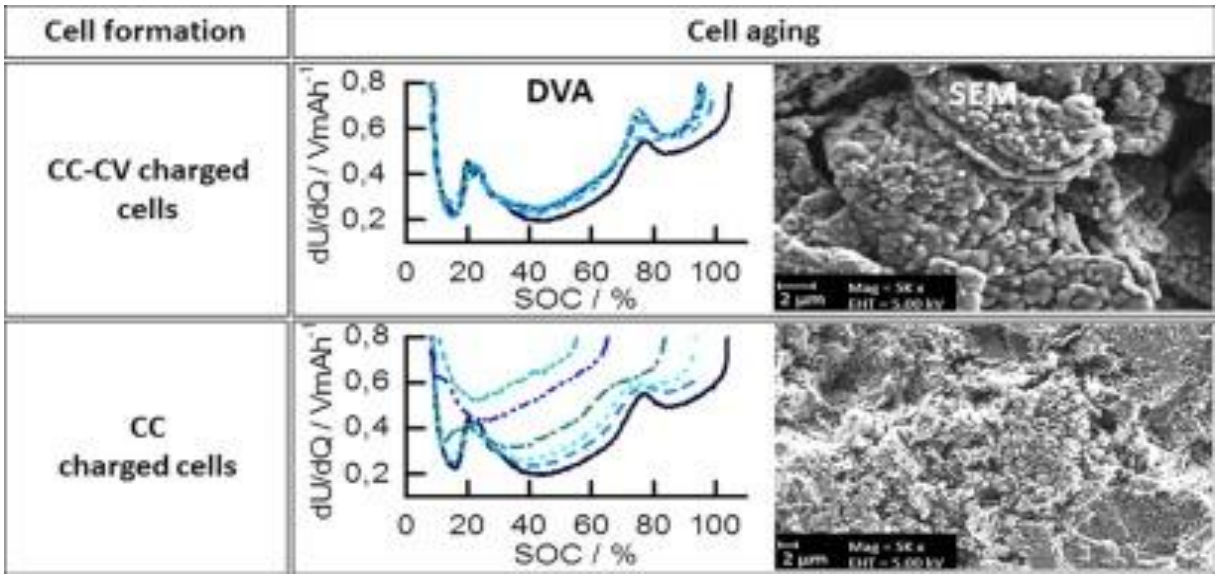


*Figure 2.1- Battery Degradation in CC-CV and CC [11]*

Usually, the CV phase takes most of the time of the charging yet it fills less SoC of the battery. This happens as the current decreases slowly. This phase is needed so that we may use the battery for a longer time. Without this phase lithium and graphite anode's cells which are able to cycle degrade rapidly.to cycle degrade rapidly. Because of this method's simplicity and easy implementation, it is used almost everywhere. It is used partially in other methods too. Though it is widely used it is not the best in terms of charging time. It takes few hours to fully charge the battery.

There is a positive correlation between increasing current and increasing time taken by the CV phase due to lithium plating. Above a certain C-rate no matter how much we inject current the charging time does not reduce. Another issue with the CC-CV method is it increases battery temperature much higher than other methods. As a result, the battery losses its longevity much faster in the CC-CV method.

### 2.2.2 Pulse Charging Method

Pulse charging is considered to be more efficient than the traditional CC-CV method. Furthermore, it has a high charging rate and a long battery life cycle. The mechanism of this charging strategy is to inject a continuous current pulse. The width and the amplitude of the pulse are controlled to inject enough current to fast charge the battery. Several algorithms are being used to do the pulse charging such as variable frequency pulse charge system (VFPCS) and duty-varied voltage pulse-charge strategy (DVVPCS). The duration of the charging is determined by the pulse frequency while the duty cycle determines the longevity.

## 2.3 Optimal Frequency

In the pulse charging method, pulse currents are injected into the battery which has various frequencies. Different frequency results in different current injection as frequency causes to change the battery impedance [12].

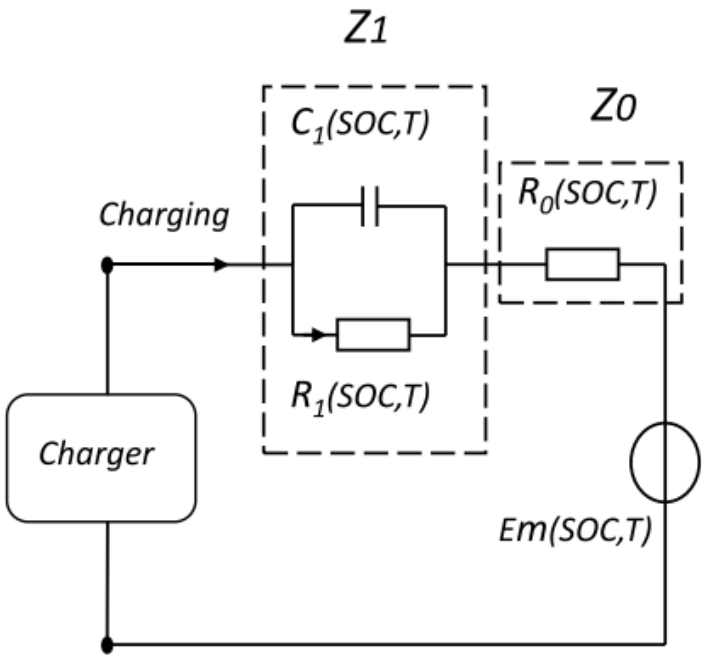


*Figure 2.2: Equivalent circuit model of li-ion battery [10]*

From the li-ion equivalent circuit, we can learn that the battery cell has capacitive characteristics and this characteristic shows different responses when different frequency pulse is injected into them.

From the circuit, we can get that the battery impedance is $Z_0 + Z_1$. Where $Z_0 = R_0$ and

$$Z_1 = \frac{R1 \frac{1}{j\omega C}}{R1 + \frac{1}{j\omega C}}$$

So total impedance, $Z = Z_0 + Z_1 = R_0 + \frac{R1 \frac{1}{j\omega C}}{R1 + \frac{1}{j\omega C}}$.

So, we can see that the frequency has negative relation with the battery impedance. The more frequency we will increase the less will be the battery impedance. Battery impedance is the reason why the battery cannot get the exact current from the power source. The optimal

frequency is the frequency that lowers the cell impedance such that the highest charging current can be pushed to the battery.

## 2.4 Optimal Duty Cycle

We cannot charge a battery with a constant charging current with pulse method rather we send current pulse to the battery. The pulse-charge strategy works on the electrochemical characteristics of the battery to fast charge a battery where it diffuses and distributes electrolyte's ions more evenly by sending the pulses. In this method, the battery is charged through the pulse current/voltage then it follows with a rest period so that the ions may diffuse and neutralize. Rest period lets the ions distribute evenly and change the concentration on the surface of the electrode. Duty of the pulse is determined by it resting period. When the resting period is much higher or in small duty, the battery charges slowly but results a higher exchange current density. On the high duty we see the opposite. So, to increase the charging efficiency a duty needs to be set in which the resting period is not so high which may result longer charging time and not so low that may result low exchange current density. That duty cycle is called optimal duty cycle. And it varies at different SoC. Study shows that the variable duty cycle pulse charging method is much efficient than the one where the duty cycle is fixed.

## 2.5 Temperature in Fast Charging

When we have a fever, that indicates that maybe our body is under the attack of the virus or bacteria. Likewise, when a battery heats up that indicates that there is something wrong with the battery. And if the heating lasts for long, it starts to decay the battery cell which results in low capacity and ultimately battery becomes dead [13]. In a circuit, if we pass current through it and if a resistor is present there then the resistor builds up heat. The battery is a circuit and it has properties like a resistor which tries to stop the flow of current and builds heat. They are polarization voltage and battery impedance. When the polarization voltage is higher, the

injection of the high amount of current results in a higher temperature. For this reason, fast charging is done when the SoC of a battery is between 20% to 80% as polarization voltage is higher outside of this SoC range. Various fast charging methods concerns specifically temperature as it is one of the core drawbacks in the fast-charging field. Excessive temperature not only decays the battery but also increases the risk of a battery explosion. In recent days, many accidents are occurring because of the battery explosion, mostly in smartphone batteries. This causes a serious threat to the body if the device is in contact with the person. As the battery is a collection of many cells and they stay close to one another, if one cell explodes others also explode because of that explosion. So, the temperature of the battery needs to be monitored and controlled so that it may not pass a certain range.

# Chapter 3

# Pulse and Temperature Based Fast Charger Algorithm

Fast charging has become one of the most demanding technology all over the world. The need of fast charging is now bigger than ever as almost all our devices use battery to power themselves. From giant electric truck to small TWS (true wireless) earbud, almost every electric device uses rechargeable battery. Li-ion batteries are the most popular rechargeable battery. There are many methods to charge a li-ion battery. Pulse based charging method is one of them. This method is recognized as a fast and efficient way to fast charge li-ion battery.

## 3.1 Previous Work on Pulse Charging

Pulse charging was first proposed back in 1900s [14]. This method was applied for lead acid battery as well. It helped to eliminate sulfation [15, 16, and 17].

Previously, variable frequency pulse charge system (VFPCS) and duty-varied voltage pulse-charge strategy (DVVPCS) was introduced to inject high current in the battery [18, 19]. Another charging system was developed in [10], where they implemented both in one algorithm.

### 3.1.1 VFPCS

VFPCS needs a variable frequency generator and a current sensor to measure current for different frequency. The frequency generator generates different frequency, and the current sensor measure the current sensor at the same time.

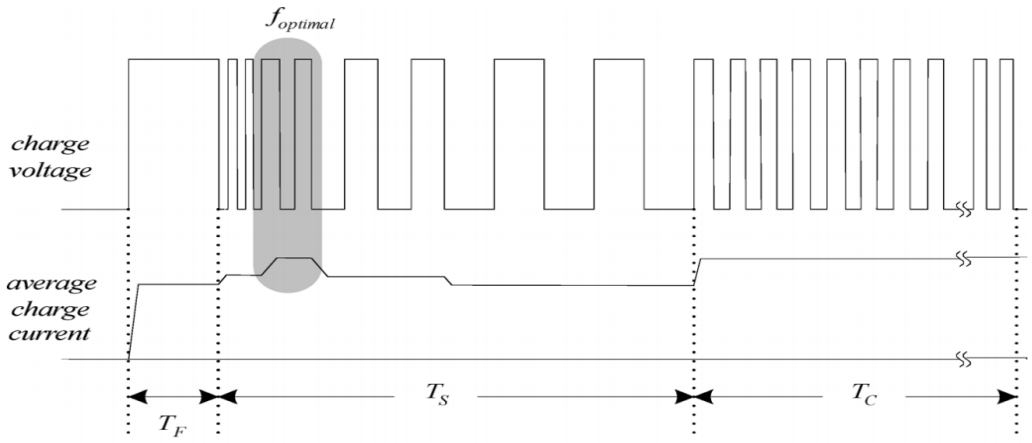


*Figure 3.1: Time sequence of VFPCS [18]*

This charging system used an optimal frequency detector to detect the most optimal frequency for charging. Then, frequency generator generates optimal frequency to charge the li-ion battery. This algorithm works in three steps. First, it finds out if the battery is full charged already to prevent from over charging. Then, it searches for the most optimal frequency to charge the battery and finally it charges the battery with the optimal frequency. Battery disconnects from the circuit after full charge. A microcontroller controls this whole procedure.

### 3.1.2 DVVPCS

DVVPCS adapted voltage pulse charger. It needs a duty varied pulse generator (DVPG) to generate different duty cycle and a current sensor to measure the current with respect to different duty cycle.

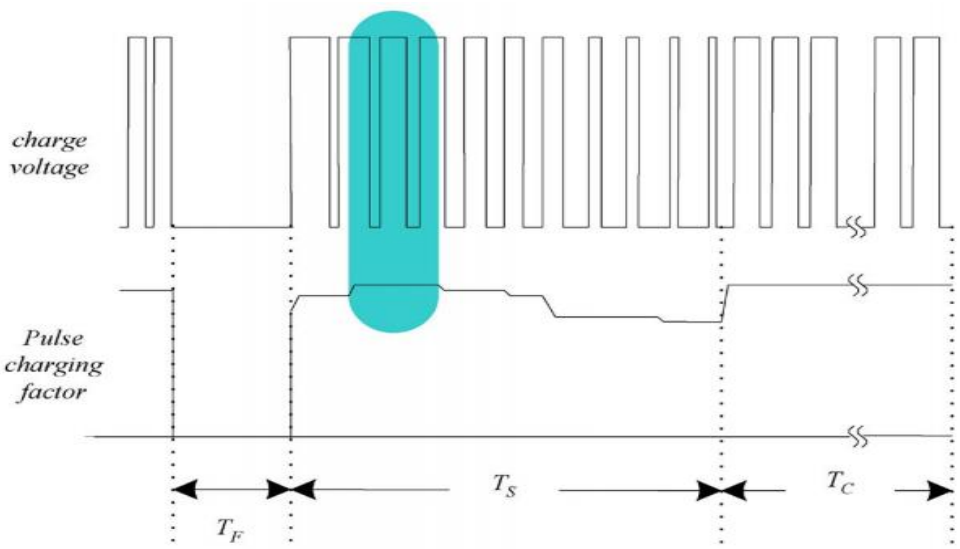


*Figure 3.2: Time sequence of DVVPCS [19]*

Initially, the algorithm finds out if the battery is full charged or not. If the battery is fully charged, it disconnects the battery from the charging circuit. Otherwise, it starts the search operation. The current sensor measures the current as the DVPG generates voltage pulse with different duty cycle. The algorithm searches for the duty cycle with maximum injected current and charge up the battery with that duty cycle. The battery gets disconnected after fully charged.

### 3.1.3 Variable Frequency and Duty Cycle

This algorithm is combination of VFPCS and DVVPCS. It proposed an algorithm where it find out a frequency for which the maximum current could be injected in the battery and maximum acceptable charging current by the optimal duty cycle search operation. Then it charges the battery with optimal frequency and optimal duty cycle according to the maximum acceptable charging current.

Two separate algorithms were used for searching optimal frequency and optimal frequency. Then, a microcontroller implemented both of the algorithm. The algorithm also used a battery management system to get the data about the battery. Their algorithm was able to charge the

battery within 3500 seconds. The battery model was 18650, its nominal voltage is 3.7v and capacity is 2500mA.

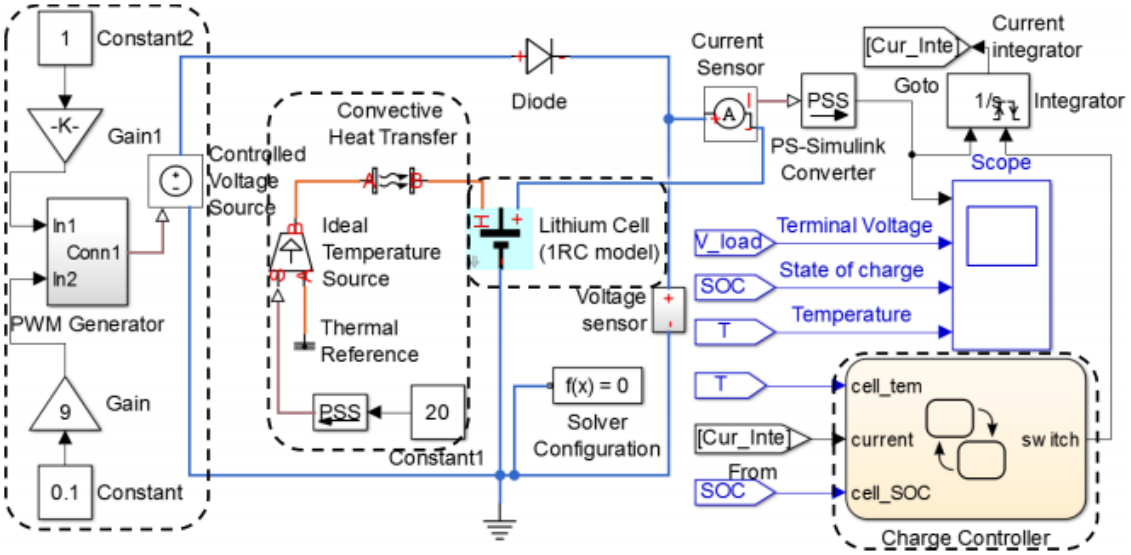


*Figure 3.3: Simulink-based model to evaluate the feasibility of the proposed charger system [10]*

## 3.2 Methodology

Fast charging is not as simple as charging with high voltage and injecting high current. Both the voltage and the current needs to be under control. Over voltage and over current both can be dangerous to the battery. It can shorten the lifespan of the battery, even totally damaged. In order to inject high current, at first, we have to determine how much current a battery can accept without damaging itself.

### 3.2.1 State of Charge (SoC)

State of charge, which is often written as SoC, is the indication of capacity left on a battery. SoC is 0% means, the battery is fully discharged and 100% means fully charged. State of charge is directly proportionate to open circuit voltage of a battery. But the relation is not linear.

| SoC | OCV |
|---|---|
| 0% | 2.80v |
| 20% | 3.50v |
| 40% | 3.65v |
| 50% | 3.72v |
| 55% | 3.77v |
| 60% | 3.81v |
| 65% | 3.86v |
| 70% | 3.9v |
| 75% | 3.95v |
| 80% | 4.0v |

*Table 1: SoC vs OCV of the test battery*

Our test battery is a lithium-ion battery. Its nominal voltage is 3.7v and the capacity is 2500mA. Table 1 shows the state of charge and open circuit voltage (OCV) of the battery. As we can see, the SoC increases as the OCV increase. But, the relation between them is not linear.

### 3.2.2 Polarization Phenomena

When high amount of current is injected in a battery, it faces polarization phenomena. This phenomenon occurs by the resistive and capacitive nature of the battery. Polarization voltage can be expressed by the following equation.

$$V_P = Vo - OCV_{SoC} - IR_O \qquad (1)$$

Here, VP represents polarization voltage of a battery. The terminal voltage is written as Vo. As we can see, the polarization voltage is drop out voltage of the battery. Depending on the state of charge of a battery, the polarization voltage changes accordingly. Figure 3.3 shows us the rise and fall of polarization voltage as the state of charge changes. At the beginning, when the state of charge is very low, the polarization voltage is relatively high. As the state of charge is rising, the polarization voltage is decreasing as well. The lowest polarization voltage is at 40% SoC. From 20% to 80% state of charge, the polarization voltage is most stable. Moreover, the polarization voltage increases as the charging current increases. For 0.5C, the polarization voltage is lowest. It increases as the C-rate rises.

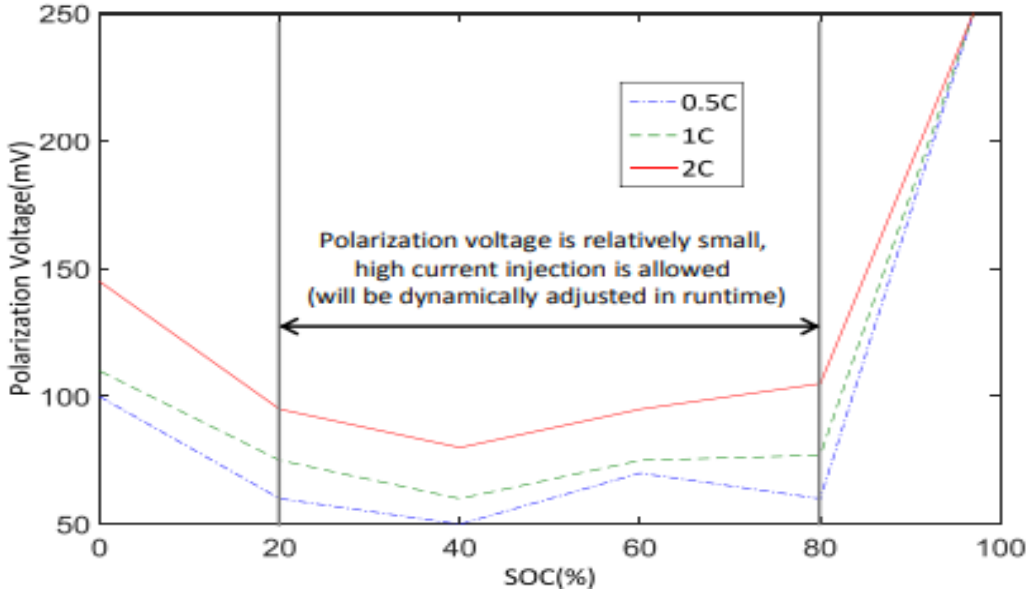


*Figure 3.4: Polarization voltage depending on charging current [10]*

Polarization voltage is not good for li-ion batteries. It increases the temperature of the battery. High temperature is not good for battery, since it decreases the lifespan of the battery. In Figure 3.5, we can see that temperature rises with the charging rate. It happens because of the polarization voltage. Since the lowest polarization voltage is observed when the state of charge is in the range of 20% to 40%, it is the most optimal range to inject high amount of current in the battery.

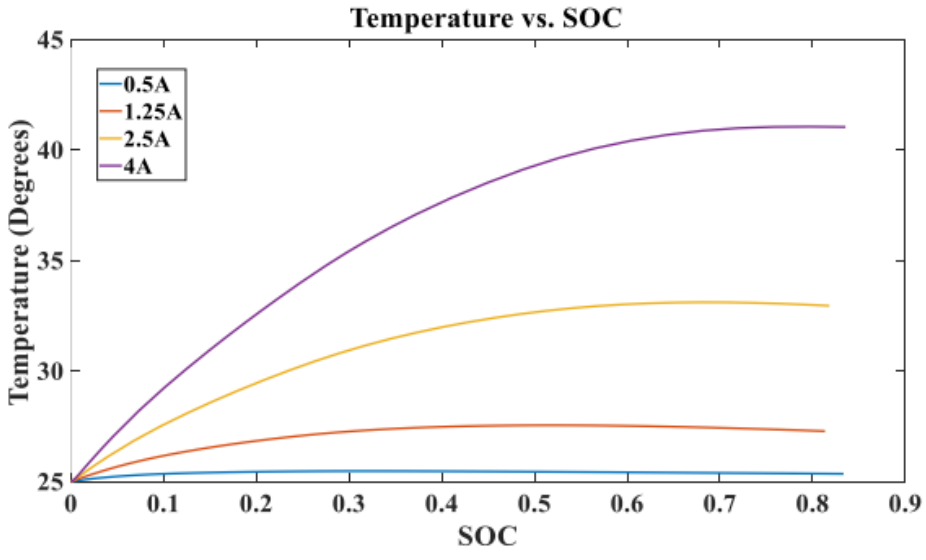


*Figure 3.5: Effects of temperature depending on C-rate [20]*

### 3.2.3 Maximum Acceptable Current

The maximum acceptable charging current for our test battery was calculated in [10] based on polarization phenomena. When the state of charge of our battery is minimal, we should not inject high current. After reaching 20% SoC we can inject high current. Then the maximum acceptable current (MAC) will be decreased as the SoC is rising. The data is given in the table 2.

| SoC (%) | MAC (mA) |
|---|---|
| 0-20 | 1250 |
| 21-40 | 4000 |
| 41-50 | 3858 |
| 51-55 | 3539 |
| 56-60 | 2979 |
| 61-65 | 2659 |
| 66-70 | 2319 |
| 71-75 | 1959 |
| 76-80 | 1500 |

*Table 2: MAC with respect to SoC [10]*

### 3.2.4 Optimal Frequency Search

The battery has internal impedance of the battery. It is dependent on frequency because of the capacitive nature. The impedance of the battery can be written as:

$$Z = Z_R + Z_I + Z_C \quad (2)$$

Here, $Z_R$ is the ohmic resistance, $Z_I$ and $Z_C$ are inductive and capacitive nature of the battery. This impedance causes energy loss in charging circuit, although for a certain frequency and for a certain state of charge, the internal impedance of the battery can be reduced to the minimal. That frequency is the optimal frequency. In our charging circuit, we have to use variable frequency to get the most optimal frequency for charging. From [14], we can see that the impedance increases exponentially in the range of 5000Hz to 100,000Hz. So, we can say that the most optimal range for frequency searching is 500Hz to 5000Hz.

The algorithm for optimal frequency search is to generate variable frequency from 500Hz to 5000Hz with a 500Hz increment. The equation for frequency search pattern can be described as the equation (3).

$$F = N * 500 \text{ (where } N=1, 2, 3\ldots) \qquad (3)$$

Then, the battery will get charged for one second. The injected current will be recorded and later it will compare with the other values for all frequencies. Frequency for highest current will be considered as optimal frequency and the current will be considered as optimal frequency current. The value of the frequency will be needed for next operation.

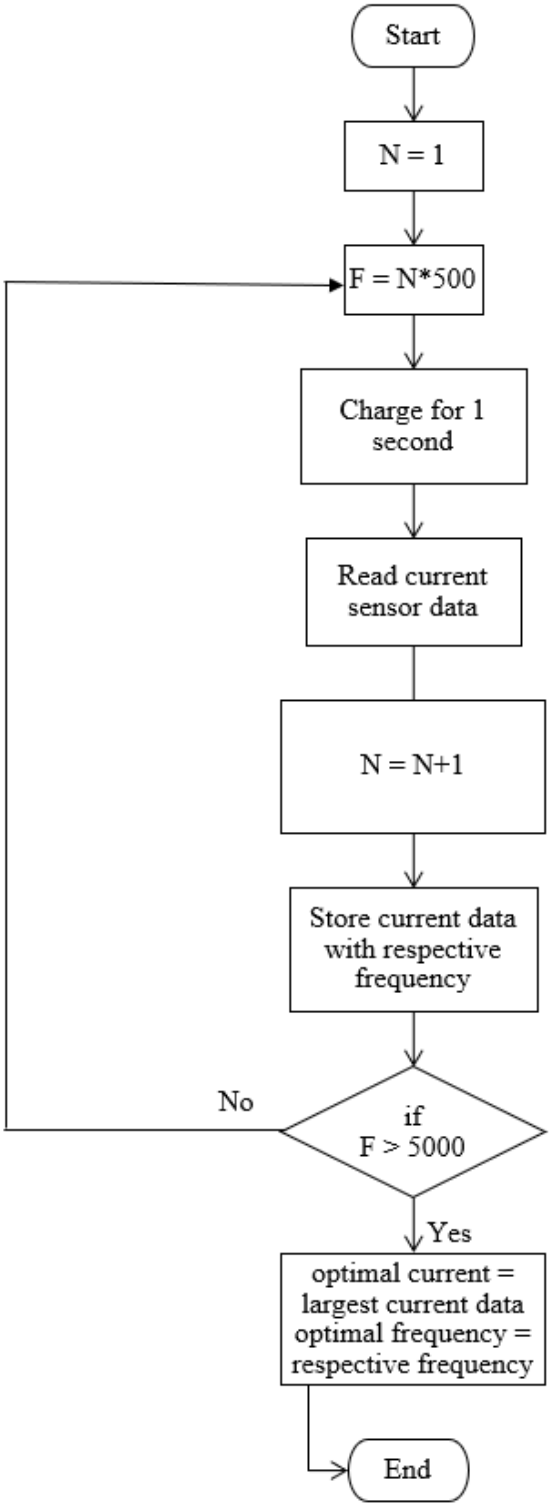


*Figure 3.6: Flowchart of Optimal Frequency Algorithm*

### 3.2.5 Optimal Duty Cycle Search

The duty cycle of pulse width modulation has effect during fast charging. The battery gets a relaxation period after injection of high current. In a whole time period of a single pulse, there will be a charging period, followed by a relaxation period.

$$T = T_C + T_R \quad (4)$$

In equation (4), the $T_C$ represents the charging period and $T_R$ represents the relaxation period. This provides a rest period for the ions inside the battery. It helps injecting high current without damaging the battery.

In order to find out the most optimal duty cycle for our battery to charge, we have to generate variable duty cycle. At first, this algorithm will get the optimal frequency from previous algorithm. Then it will generate variable duty cycle of that frequency from 10% to 90% with increment of 10%.

$$D = N * 10 \text{ (where } N=1, 2, 3\ldots) \quad (5)$$

After that it will read the open circuit voltage (OCV) of the battery and determine the state of charge (SoC). Then, it will read current for each duty cycle and store the data. After generating all the duty cycle, it will compare the current data with maximum acceptable current according to the SoC. The current data will the closest to MAC will be considered as optimal duty cycle current and the corresponding duty cycle value will be considered as optimal duty cycle.

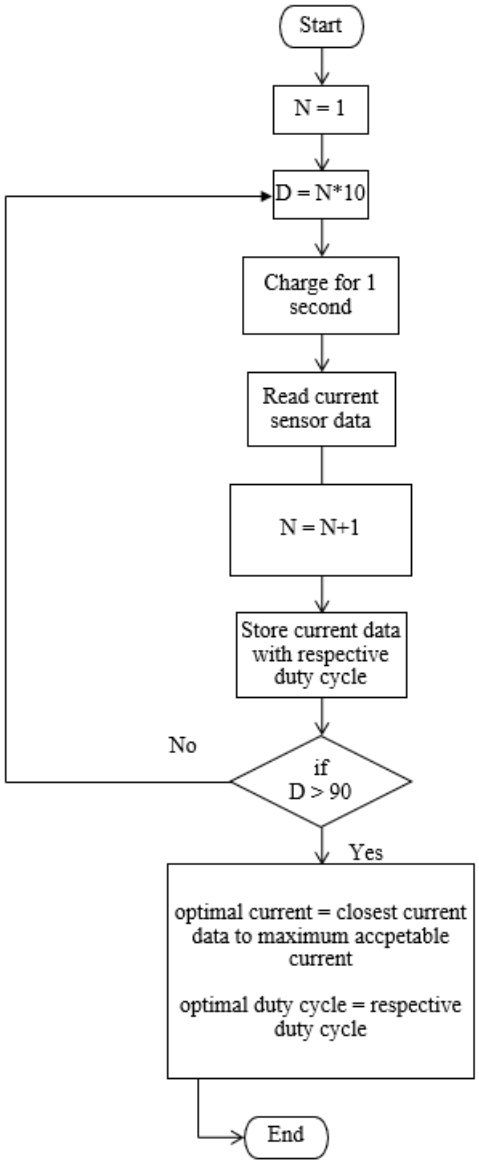


*Figure 3.7: Flowchart of Optimal Duty Cycle Algorithm*

### 3.2.6 Variable frequency and Duty Cycle Generation

In order to conduct the optimal frequency and duty cycle search, we need a variable frequency generator. A microcontroller was used in this operation. The Arduino Uno is the most suitable board for it. The TCCR1A and TCCR1B register (timer counter control register) of AtMega328P was used to generate the variable frequency and duty cycle. The TCCR1A has 8 bit, the first four of them is related to generating wave form, next two bit is to force compare and the last two is to select the mode. The TCCR1B has 8 bit as well. First two bit is related to

input compare and the last three is to select the clock. The values of the registers were set accordingly, so that it can generate variable PWM wave.

TCCR1A

| COM1A1 | COM1A0 | COM1B1 | COM1B0 | FOC1A1 | FOC1B | WGM11 | WGN00 |
|---|---|---|---|---|---|---|---|

TCCR1B

| ICNC1 | ICES2 | - | WGM13 | WGM12 | CS12 | CS11 | CS10 |
|---|---|---|---|---|---|---|---|

*Figure 3.8: Time counter control register*

The value of ICR1 register has to be set a fixed value to get a frequency and OCR1A register is responsible for duty cycle. The equation for determining the values of these two registers are given below:

$$ICR1 = 16000000 / (j * 500) \quad (6)$$

$$OCR1A = ICR1/2 \quad (7)$$

Here, the value of "j" is from 1 to 10. This equation calculates the value for ICR1 and OCR1A. Then the microcontroller generates the variable frequency and duty cycle.

## 3.3 The Algorithm

At first, it will measure the open circuit voltage (OCV) of the battery. It will help to determine the state of charge of the battery. Then, a temperature sensor will read temperature of the battery surface. If the SoC is 80% or more than that, the battery will get disconnected form the charging circuit. Also, if the temperature of the battery surface exceeds 45 degree Celsius, the battery will get disconnected from the charging circuit for one minute to cool down the battery. If the SoC is below 80% and the temperature is below 45 degree Celsius, the charger will start frequency search operation. This operation will get us the optimal frequency for the SoC. Then, it will start the duty cycle operation with the optimal frequency data. Then it will get the optimal duty cycle.

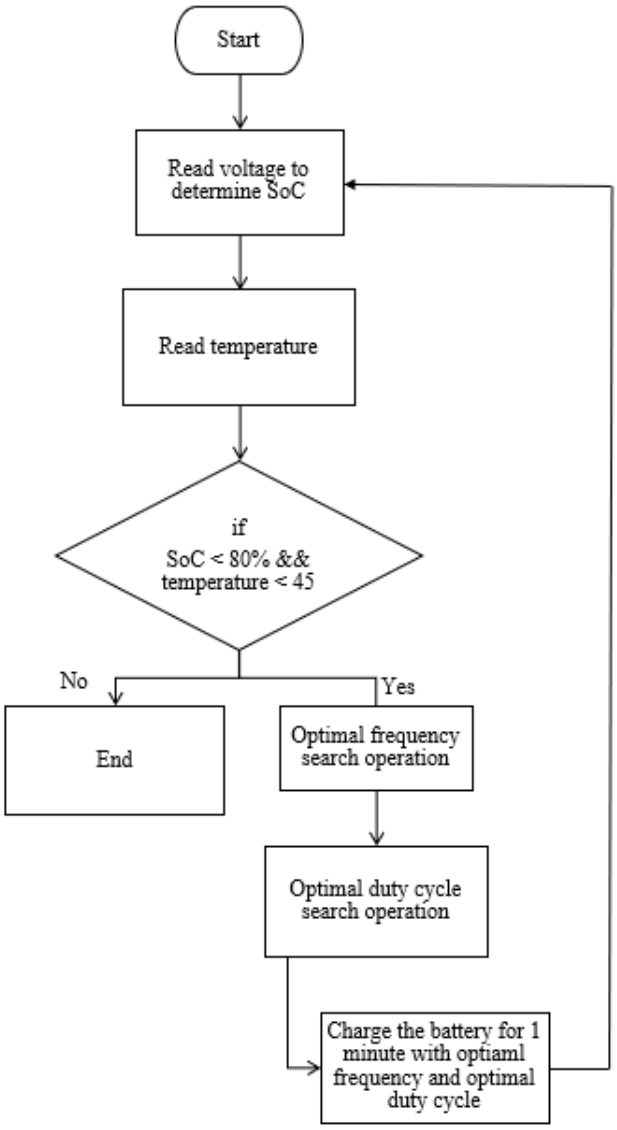


*Figure 3.9: Flowchart of Charging Algorithm*

Now, the charging algorithm knows both optimal frequency and duty cycle. It will charge the battery for one minute with optimal frequency and optimal duty cycle. After that, the battery will get disconnected from the charging circuit to measure the open circuit voltage. Voltage will determine the new SoC of the battery and do the frequency search and duty search accordingly. This procedure will go on till it reaches 80% SoC. The polarization voltage increases significantly after this range.

## 3.4 Sensor Reading

### 3.4.1 Voltage Reading

The voltage reading is a very important part of this algorithm, since it determines the SoC of the battery. The analog to digital converter (ADC) pin of the Arduino Uno reads the voltage reading of the battery. To get the accurate value, the reading is taken for 2000 times and averaged it.

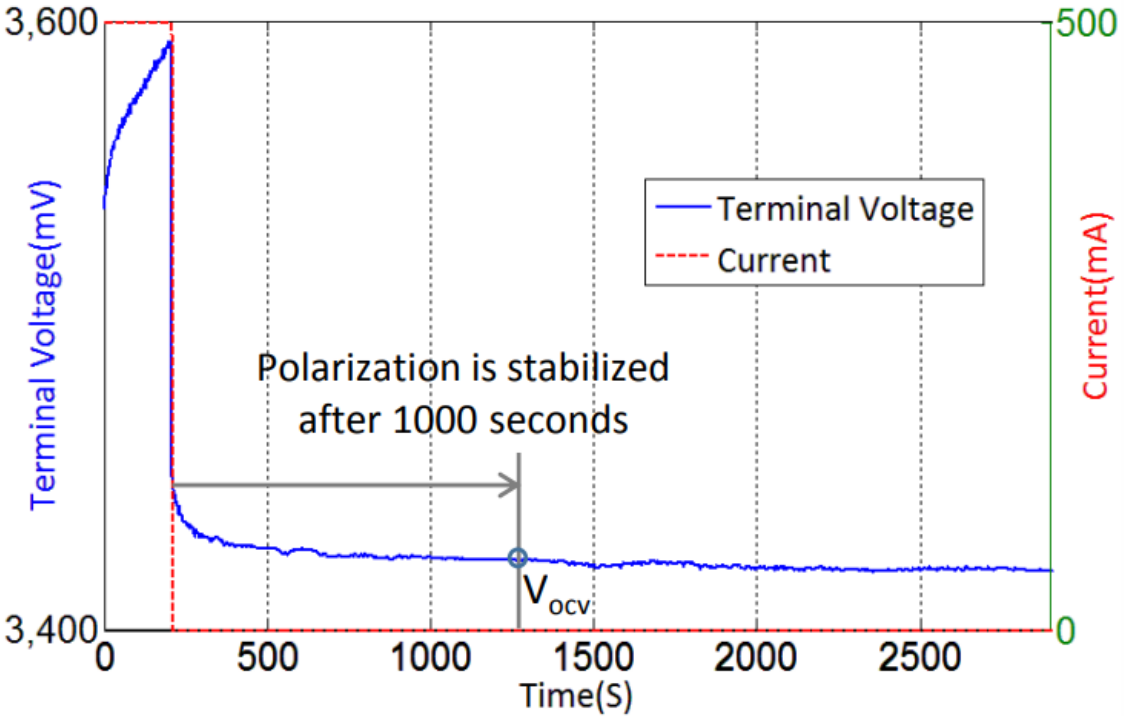


*Figure 3.10: Terminal voltage settle down after 1000 second [10]*

Since, it takes almost 1000 seconds to stabilize the terminal voltage of the battery, we have to wait this amount of time before reading the voltage. We tested the voltage drop of the lithium-ion battery terminal after one minute of charging. Right after the charge, the sensor reading was 3.38v. Then, after one minute the voltage dropped down to 3.32v. The voltage stabilizes at 3.27v over the time and dropped down even more at the very last moment and gave us the data of 3.26v.

The voltage data determines the state of charge of the battery. From this test, it is very much clear to us that the delay of 17 minutes before charging was very much vital for our experiment.

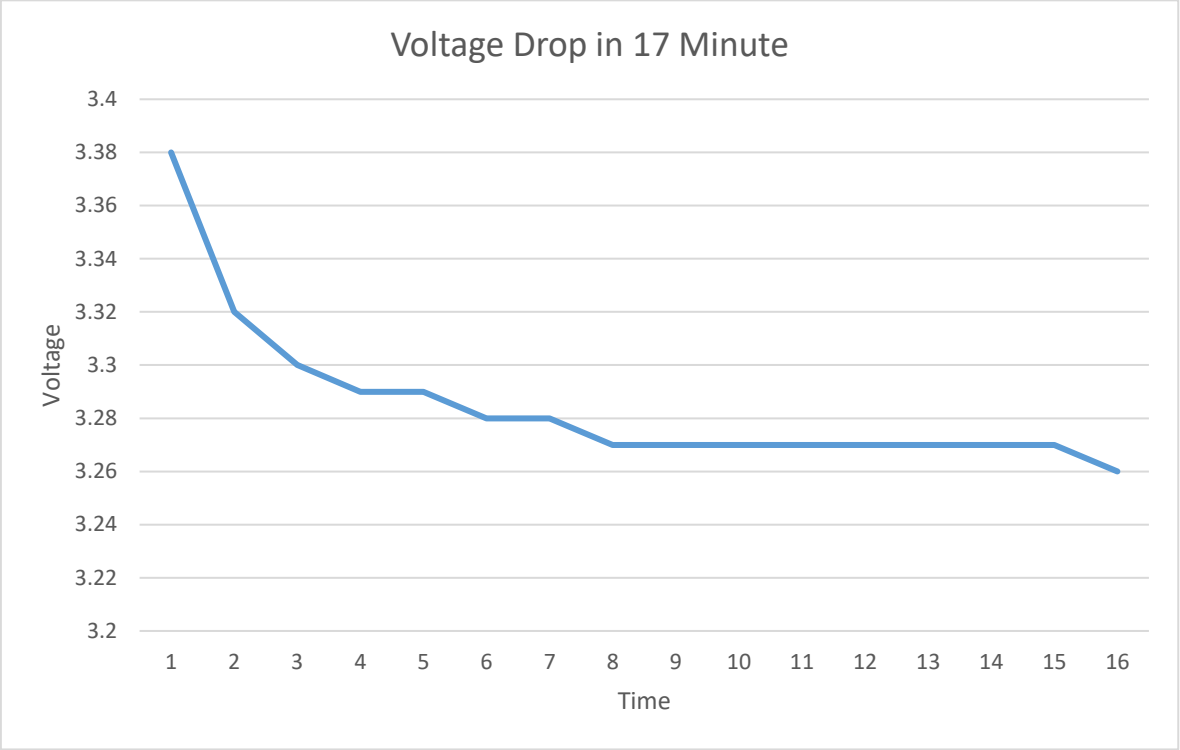


*Figure 3.11: Voltage drop with time*

### 3.4.2 Current Reading

The current reading was done by another analogue to digital converter (ADC) pin of Arduino Uno. The data was taken 1000 times and then averaged to get the most accurate data. Otherwise, the data fluctuates a lot.

The current reading part of the code was accessed multiple times. The frequency search operation, duty cycle search operation needed the current reading part. Furthermore, the charging current was monitored with it.

### 3.4.3 Temperature Reading

Another important part of the algorithm is temperature reading. Arduino's ADC was used for reading the temperature. This part of the code helped us to monitor the battery surface temperature. Also, when the temperature rises out of the control it will signal the

microcontroller to disconnect the battery from the charging circuit. The microcontroller disconnects the battery with the help of a relay. This operation helps to prevent an accident that can happen from overheating the battery.

# Chapter 4

# Hardware Implementation

## 4.1 Proposed Circuit Diagram

In order to justify the predesigned prototype of the proposed charging system, a sustainable hardware implementation needed to be established. Considering the hardware implementation, a proper circuit diagram with the details of various components specifications is developed. As the proposed charging system is offering the pulse charging method the hardware must be needed to run the operation at the most accurate appearance. To build a promising circuit to implement the pulse charging method.

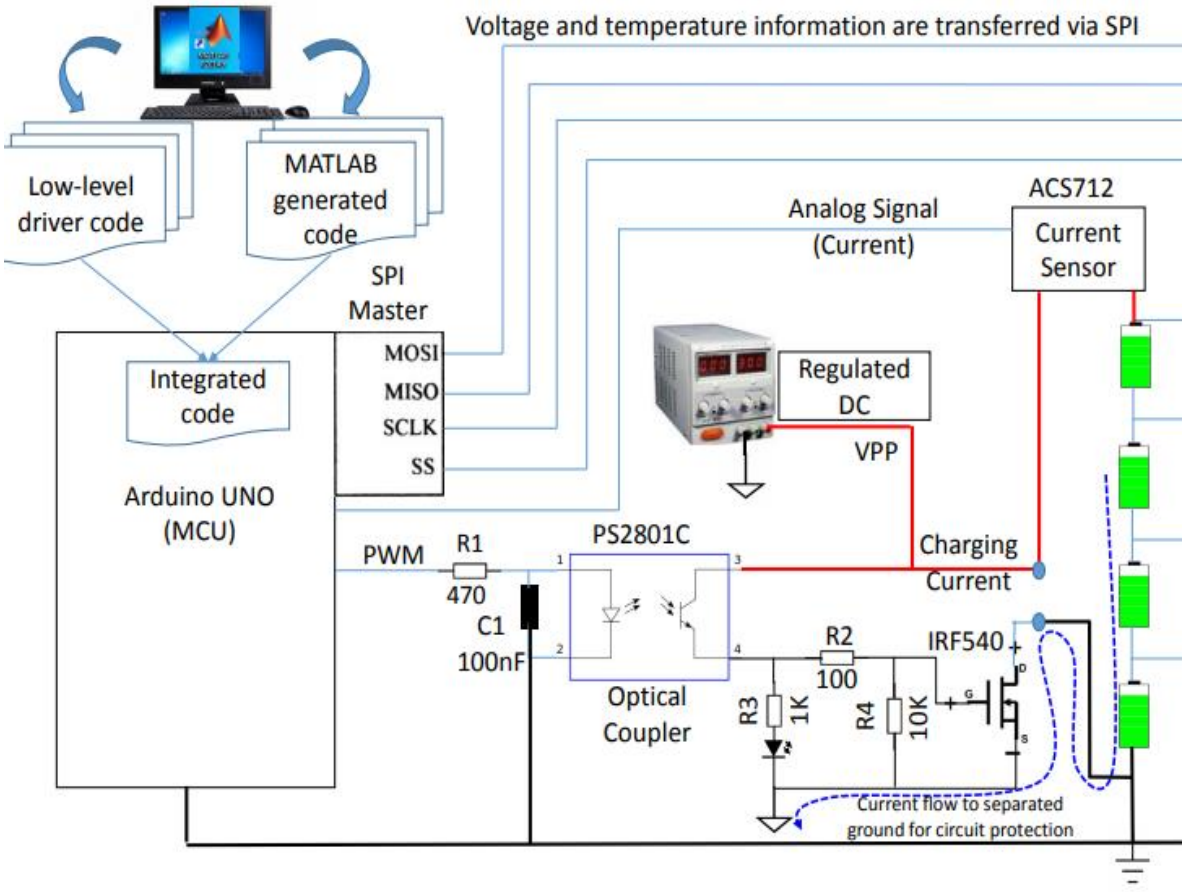


*Figure 4.1: Schematic diagram for pulse charging model [10]*

To charge the battery through the pulse charging method, as per the rating of different soc and current injection rating it is needed to change the pulse accordingly to the algorithm. So, to control the charging current, according to the pulse charging method firstly a microcontroller needs to be set as the host. The other different components are being synchronized through the microcontroller. The microcontroller is the main hardware component to control the charging parameters as per the algorithm. As the circuit is being designed based on delivering a prominent charging system with the pulse method, the circuit needs to ensure a switching capability at a very high speed. An uninterrupted switching mechanism is ensured by a switching Mosfet, which is synchronized with the host microcontroller. The ultimate target is to charge the battery in a very efficient time-saving manner and to accomplish it the current injection part is also moderate as per the algorithm. To supply the required current to the battery a prominent power supply unit is introduced in the circuit design. Along with that, to measure the ultimate rating such as soc, optimal frequency, optimal duty cycle and optimal injection current some sensors are a must prerequisite. In the circuit design, a voltage sensor and a current sensor perform actively to activate the algorithm along with the microcontroller simultaneously. In addition, a relay is also a very important part of the circuit diagram, as the relay plays a vital role to establish a secure connection to the power supply unit. As per reference, the ultimate circuit diagram is as followed to get the best outcome. A simplified block diagram is shown below.

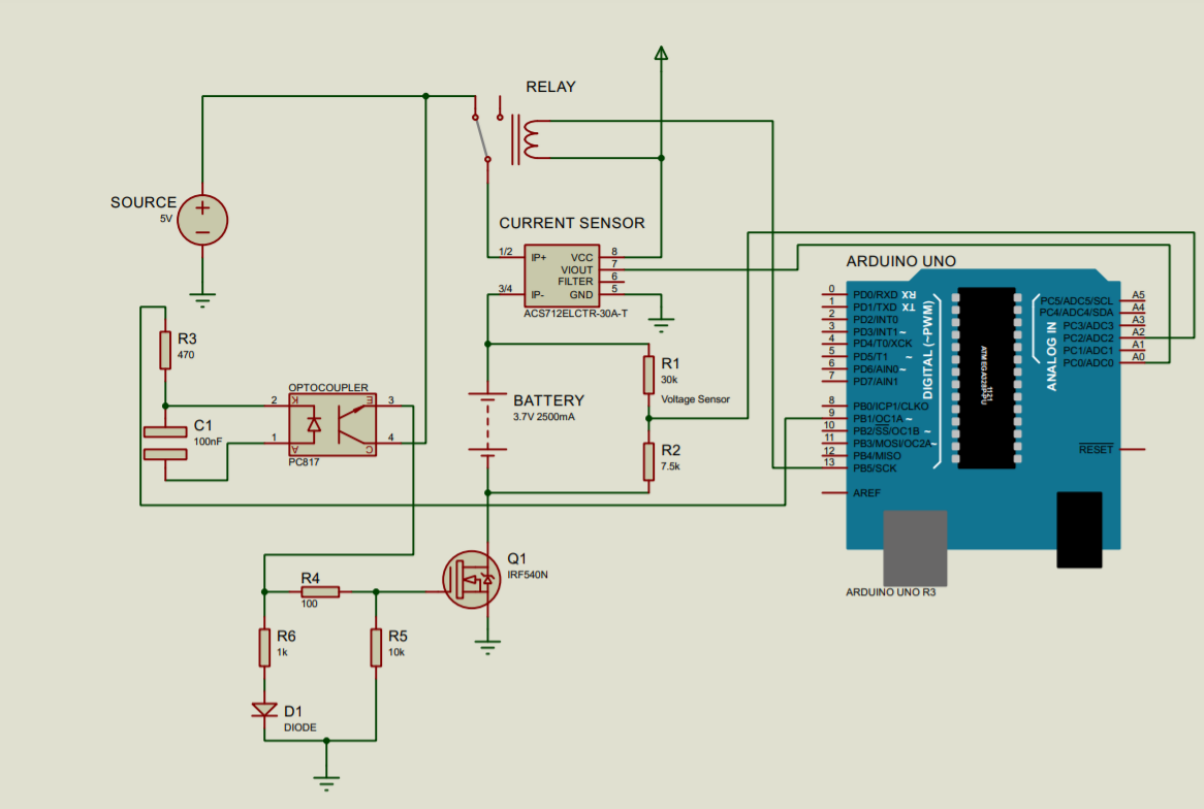


*Figure: 4.2: Proposed model*

In the hardware implementation all the sensor and relay are powered by 5V DC source. Basically, these equipments are synced with the host Arduino and Arduino itself provided the 5V dc supply. In the hardware implementation a secondary ground is established in order to secure the battery connection. The secondary ground is totally separate from the power supply unit or the ground provided by the host Arduino. The main purpose of the secondary ground is to converge a draining channel for the battery in case any over flow of the current.

## 4.2 Arduino Uno

Arduino Uno is a microcontroller board. This is based on the ATmega 328. An Arduino Uno consists of almost every possible needed feature to support a microcontroller. The Arduini Uno has a memory of 32 kb, 2kb of SRAM and a combination of 1kb EEPROM. Arduino Uno is an open-source gadgets stage dependent on simple to-utilize equipment and programming. It can be guided by sending a bunch of code to the microcontroller on the board. To do so it is

needed to utilize the Arduino programming language, and the Arduino Software (IDE), in view of Processing. Arduino Uno is a very popular microcontroller for its unbeatable features. The required software for the Arduino Uno is really justified for both beginners and the advanced level of users. Whereas it supports the cross-platform operating system like Windows or Linux. One of the main and attractive characteristics of Arduino Uno is that it is an open-source platform and also supports extensible hardware. Along with that, an Arduino Uno offers a very cost-effective package. To implement the hardware part the microcontroller plays a vital role as a host. In this case, an Arduino Uno is the prime option. It supports all the possible features to create the ultimate version of the hardware part smoothly. It gives another advantage over the synchronization part for the other components like a different sensor. The user interface of the Arduino Uno is really compatible with both the hardware and coding part. So, the Arduino Uno makes the implementation much easier than the other microcontroller in this case. But there is a drawback also, as the memory is not really enriched like the other microcontroller like Raspberry Pi. For the sake of hardware implementation, a realistic pulse-based algorithm is developed which is needed to run continuously by a microcontroller. In that case, Arduino Uno is the best option in the class. Arduino Uno is used as a host microcontroller in order to run the algorithm continuously. It is connected to a pc to upload the algorithm and also for power up. Along with that, the property of the serial monitor is also monitored regularly for data accusation of the real-time hardware implementation. As per the algorithm, the Arduino Uno perform the controlling part over the other different components like sensor or power supply unit. The Arduino Uno take the reading of the battery open-circuit voltage from the voltage sensor in order to check the current stage of SOC. After that, it finds the optimal current and optimal duty cycle for the updated stage of SOC and then it operates the charging stage for the battery. Optimal current is determined by the reading received from the current sensor reading which is synchronized with the Arduino. Basically, the Arduino proceed a searching

function and the very next moment it enables the charging operation and it is responsible for all the controlling parts of the real-time charging procedure. For the implementation of the circuit setup, there are different type of connection is established. It can be clearly observed from the Figure 4.2 that some the pin of the Arduino is used for connecting the different components. A0, A1 and A3 are used for connecting the current sensor, voltage sensor and relay accordingly. Pin 9 is dedicated to the MOSFET IRF540. This pin is especially dedicated to PWM. Lastly, the ground of the Arduino is connected with the ground of the power supply unit.

## 4.3 Relay Module

Relays are switches that are used to close and open circuits both electronically and electromechanically. It regulates the opening and closing of an electronic circuit's circuit contacts. The relay is not energized with the open touch when the relay contact is open (NO). If it is closed (NC), however, the relay is not energized due to the closed communication. Relay module also maintains the same working procedure. A relay module is easy to sync with the Arduino Uno. A relay module is operated by different voltage. It could be 5v, 12v etc. This voltage source is used to energize the coil of the relay to trip the other portion of the relay. This procedure of energizing coil plays the mechanical trip for the relay and interchanged the connection between the normally closed and normally opened terminals. Thus, the interchanged secured the contact from the circuit connection.

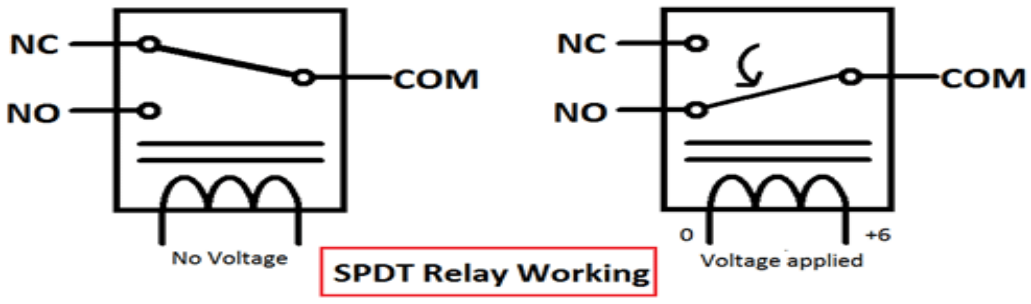


*Figure: 4.3: Working operation of Relay [21]*

A relay module is a more developed device for easy access to sync with the Arduino. It is equipped with some LED indicators, resistors and diode. Along with that, a terminal block is for connecting the circuit and input jumper for sync with the Arduino and provide the required power for operating the relay. From Figure 4.4, it is observed that the terminal block has three connection point these are normally closed, normally opened and common point. Normally closed means it will allow the current flow for all the time if the coil is not energized on the other hand the normally opened means it will not conduct any current to the circuit if the coil is not energized. However, if the relay coil is energized it will be performed just the opposite as mentioned now. The three input pins are designed for connecting with the Arduino and provide the power for the relay. From Figure 4.4 it can also mention that the pin Named In is connecting for the Arduino. It can be connected to any of the pins of the Arduino but it also needed to declare the right pin in the coding part and the other 5V and GND is for the power-up of the relay.

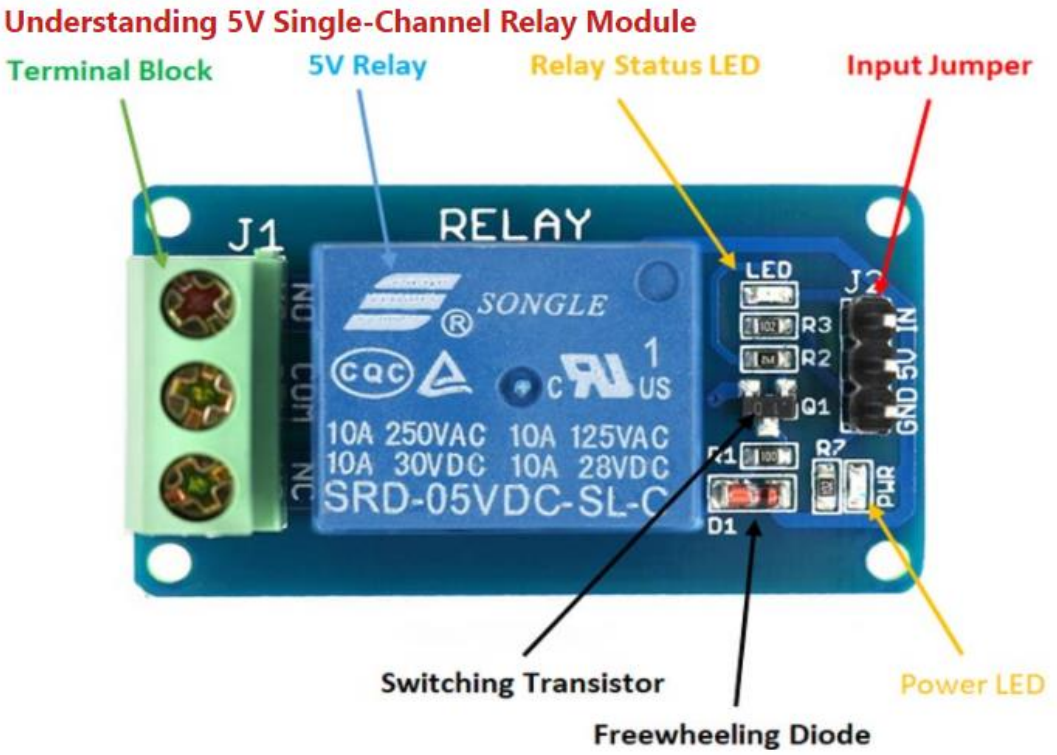


*Figure: 4.4: Relay [22]*

For the implementation of a pulse-based charging circuit, the relay plays a very crucial part. The relay maintains the current flow according to the algorithm. It must be ensured that the current flow will continue only based on the algorithm. As per the algorithm, there is a regular basis seventeen minutes delay after every charging period and at this time the current flow must be stopped completely. Thus, a relay is the best choice to stop the flowing of the current. It breaks the circuit completely by its mechanical performance and ensures the current flow stopped by 100%. After this delay, the relay turns on automatically and the microcontroller starts the charging process again and the run-up to the compellation of one charging cycle. For the implementation a 5v relay is used as the Arduino can serve a constant 5V supply. In that case, a 5V relay can be powered up simply using the Arduino dedicated 5V supply. Along with that, the required current rating is another important factor for choosing the correct relay. The 5V relay can bear up to 10A current without any complicacy. Whereas, as per the algorithm, the highest desired current rating is up to 4A. So, a 5v relay is the best option to implement the circuit.

## 4.4 Current Sensor

A current sensor senses electric live current in any wire and produced a signal proportional to the current. The produced signal can be used to show the measured current like an ammeter. There is different current sensing method available for a different purpose [23]. The magnitude, precision, bandwidth, robustness, cost, isolation, and size of a current sensing system are the key factors to consider when choosing one. An instrument may represent the current value directly, or it may be translated to digital form for use by dedicated monitoring or control device. For any current sensor, there are two different ways to measure the current, one is direct sensing and another is indirect sensing. In the case of direct sensing, the mechanism justifies the voltage drop appears in a wire followed by Ohm’s law [24]. On the other hand, indirect sensing is developed by the magnetic field followed by Faraday’s law. Whereas, the magnetic

field is obtained by transformer or Hall effect sensor. ACS712 is one of the most popular and available current sensor modules for using along with Arduino Uno. This current sensor is formed by developed an indirect current sensing system which is done by a Hall effect sensor. The Hall sensor developed a voltage that is proportional to the sensed magnetic field by the sensor is used to measure the current. This IC can sense both AC and DC, making it useful in a variety of situations. Peak detector circuits, circuits to maximize gain, rectification application for A to D converters, Overcurrent fault latch, among other applications use ACS712. There are some variants of the ACS712 is available. The variant is depending on different ratings of the current level such as 5A, 20A, 30A. Based on this current rating there are also some changes in the coding part for synchronizing the sensor with Arduino. ACS712 is also a cost-effective choice for the implementation of any hardware part.

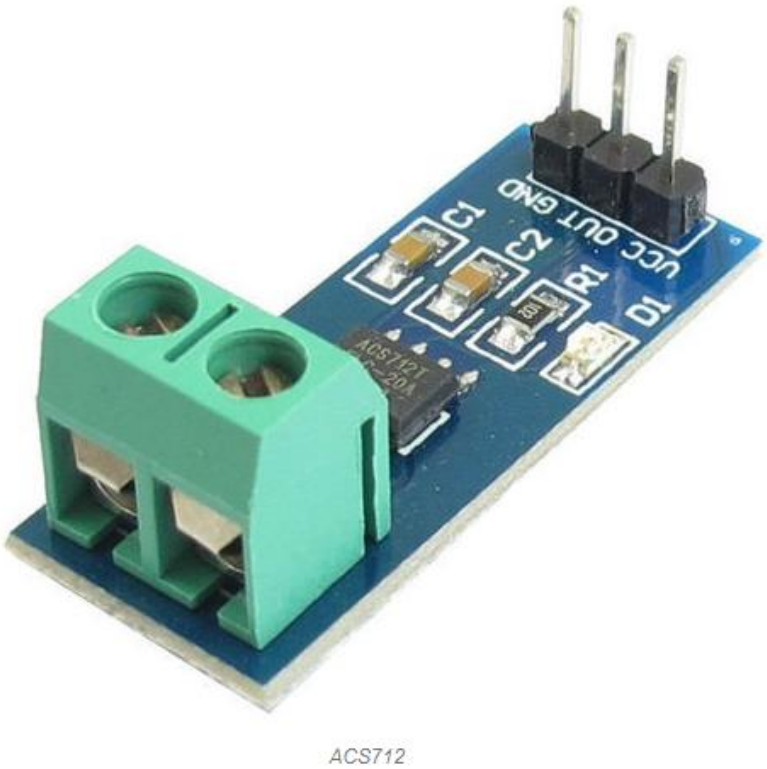


*Figure 4.5: ACS712 Current sensor*

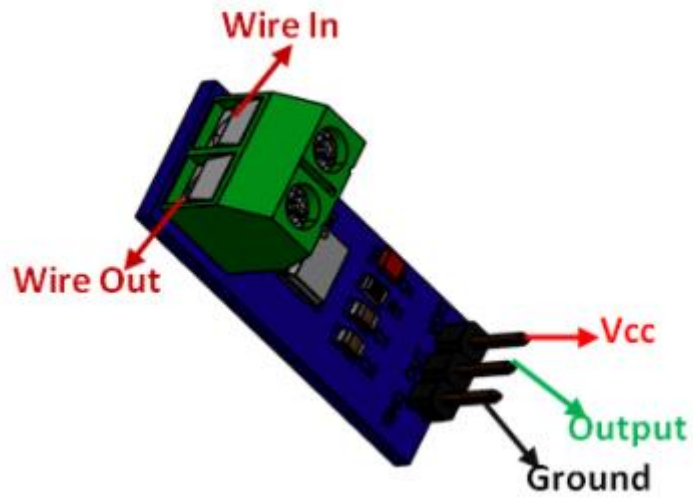


*Figure 4.6: Connection diagram of the current sensor [25]*

It is observed that the ACS712 has 2 pins for the current incoming and outgoing. These 2 pins are the wire out and wire in on Figure 4.6.

Along with that, ACS712 has an output pin that is for synchronizing with the Arduino. It also needs a 5V supply to operate. The $V_{cc}$ and GND are needed for the empowering of the sensor. There is also a LED available for the indication of the power on or off.

For the implementation of the pulsed based charging circuit current sensor is another important part. As per the algorithm, the microcontroller needs to check the rating of injection current frequently. Thus, the current rating data accusation is done by the current sensor. According to the algorithm the microcontroller firstly determines the SoC and then it needs to be determined the optimal current with a regulatory combination with the optimal frequency and optimal duty cycle. Along with that, the injection current rating is also measured at the charging period. All the current rating data is monitored in real-time optimization by the serial monitor of IDE. In this case, a 30A rated ACS712 is used for the best output. In Figure 4.2 it can be determined that the current sensor is connected to the Arduino by the A0 pin. For synchronizing the ACS712 any pin of the Arduino can be used.  Apart from this, the current incoming wire is fed

from the relay and the outgoing wire is directly connected to the battery positive terminal. Thus, it is clear that the sensor is liable to measure only the battery injecting current.

## 4.5 Voltage Sensor

A voltage sensor is a sensor which calculate and monitor the amount of voltage in an object or the amount of voltage passing through an element [26]. These types of sensors are equipped to track AC and DC voltage levels. The sensor takes input as voltage from the arrangement and gives outputs as switches, analog voltage signals, current signals or an audible signal. These types of sensors have become an excellent choice for conventional voltage measurement methods.

A voltage sensor can sense the supply of voltage by determining, monitoring, and measuring. It can measure the AC level or DC voltage level [27]. Depending on the model of the voltage sensor it gives the output in the form of analog voltage signals, switches, audible signals, analog current levels, frequency, or even frequency-modulated outputs.

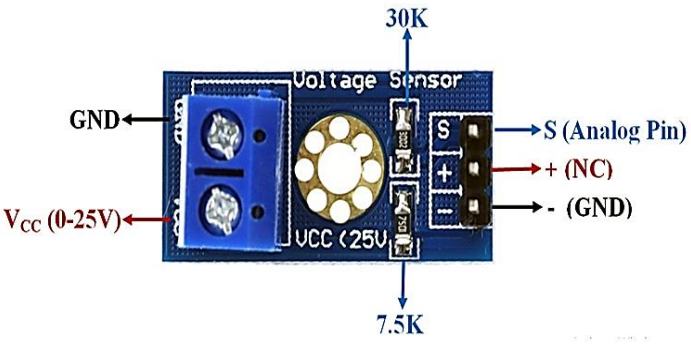


*Figure 4.7: Voltage sensor module pinout*

This is a 25V voltage sensor and it has five pins in total. From these five pins, three of them are male header pin and two are on the two-pin screw terminal. The marked screw terminal pins are VCC and Gnd. Most importantly VCC and Gnd must be connected to the external source of voltage that needs to be measured. Input voltage is 0-25V. On the other hand, three

male headers which is marked as S, + and -. Here, the S pin is the sense pin and it must be connected to the analog input of the Arduino. Also, the "- "pin must be connected to the ground of Arduino. As for the "+" pin, it is an N/C pin and it is not connected to anything. Moreover, this voltage sensor is a voltage divider consisting of two resistors with resistance of 30 KΩ and 7.5 KΩ.

For implementation of the pulsed based charging circuit voltage sensor is another important part. The voltage sensor is used here to read the terminal voltage of the battery.

## 4.6 Temperature Sensor

LM35 is an analogue temperature sensor that has an integrated circuit and 3 pins. It can measure the temperature of the surface of solid objects. LM35's output pin gives voltage as the output. As the output voltage has a linear relation with the temperature, we can get the result very easily. Apart from that, it is a small and cheap IC that can be found easily in the market. We can get the temperature in the Celsius scale and its measurement scale is from -55°C to 150°C. One of the features of this three-pin temperature sensor is, it is easily compatible with Arduino Uno [28].

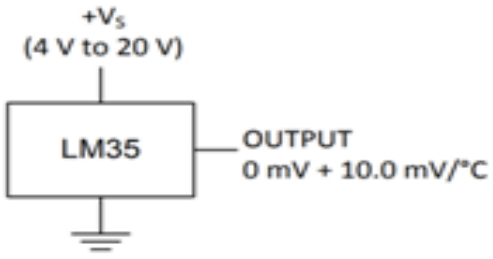


*Figure 4.8: Pins of LM35[29]*

Because of these properties, we have chosen to use this temperature sensor to measure the temperature of the surface of the battery.

To connect the sensor, we have to power the IC by applying a regulated voltage VS to the input pin and connect the ground pin to the ground of the circuit. Now, measurement of the temperature can be extracted in form of voltage as shown below.

Initially, the output voltage will be 0V according to the temperature. There will be a change of 10mV for every degree Celsius change in the temperature. The voltage can be converted into temperature using the below formulae.

$V_{OUT}$ = 10 mv/°C × T

where

- $V_{OUT}$ is the LM35 output voltage
- T is the temperature in °C

## 4.7 Battery

To do this project, we have used a specific model of li-ion battery which is Samsung 25R (INR18650-25R). We have selected the battery for few reasons. Firstly, the algorithm which we are following needs to have the maximum acceptable current data at every SoC of the battery. We have managed to get the data for this specific type of battery. Hence, we selected it. Moreover, this model is available in the market and is affordable. In the market, it is available with two shapes, cylindrical and rectangular. We were able to manage both shapes battery for our project.

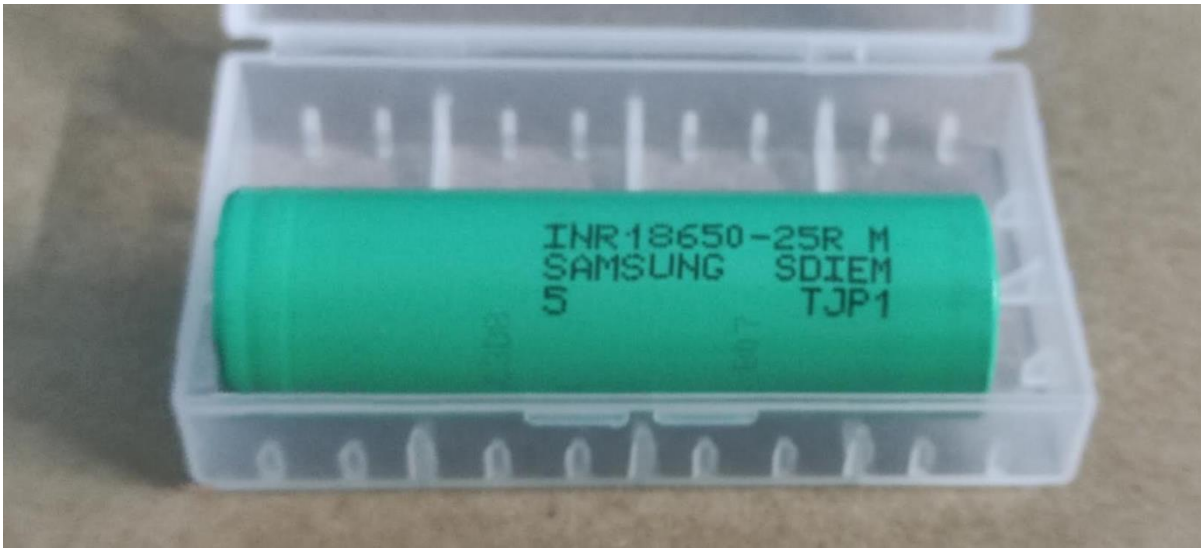


*Figure 4.9: Samsung 25R (INR18650-25R) Battery of cylindrical shape*

From the battery model, we get to know that the battery is manufactured by Samsung. 18650 signifies the battery's size. It is a mid-power range cell having the capacity of 2500mAh.

| | |
|---|---|
| Manufacturer: | Samsung SDI |
| Model: | INR18650-25R (25R) |
| Size: | 18650 |
| Nominal Capacity: | 2500mAh |
| Discharge Current: | 20A Maximum Continuous |
| Nominal Voltage: | 3.6V |
| Maximum Voltage: | 4.2V |
| Cutoff Voltage: | 2.5V |
| Approximate Dimensions: | 18.33m x 64.85mm |
| Approximate Weight: | 45g |

*Table 3: Nominal Specification of Samsung 25R (INR18650-25R). [30]*

## 4.8 PSU

Power supply unit is the key source of power for any kind of hardware implementation. To have a sustainable and reliable hardware project a promising power supply unit is a must. Along with that, the power supply unit also need to maintain the required criteria in terms of the rating of the different components a particular hardware project. Basically, for the Arduino based project the AC-DC power supply unit is mostly used one as most of the Arduino components are run by DC current. AC-DC converters take the AC power from the normal supply source like 220v 50 Hz source and convert it to unregulated DC [31]. It supplies incorporate transformers that change the voltage of the AC that comes through normal supply, rectifiers to save it from AC to DC and a channel that eliminates clamor from the pinnacles and pledges of the AC power waves. Ordinarily, the voltage will be ventured somewhere near the transformer to the voltage needed by the gadget being provided. In the initial step of changing AC over to DC, the voltage is redressed utilizing a progression of diodes. This changes the sinusoidal AC wave into a progression of positive pinnacles utilizing a rectifier. Now, in any case, there is still waveform vacillation - the time between the pinnacles - that should be taken

out. It also converts the voltage level according to the required one. For the implementation of the pulse-based charging circuit it is required a power supply unit with the ratings of at 5v and at least equipped with 4A rated current. As per the algorithm the power supply unit have to have the ability to inject at least 4A current constantly for any instance. Thus, for the implementation the circuit was equipped with a power supply unit of 5v and 10A rated. The ratings of the power supply unit ensure that it is good enough to deliver a constant 5v and 4A rated current for any instance as per the algorithm. Beside it, it can also able to deliver a higher current up to 10A if the circuit needs though.


*Figure 4.10: Power Supply (5V10A)*

# Chapter 5

# Test and Result Analysis

## 5.1 Objective

The objective of our test is to determine the time needed to charge our battery from 0% SoC to 80% SoC with our prototype.

## 5.2 Test Setup

We developed this prototype according to the block diagram and algorithm. The microcontroller is responsible for implementing the algorithm and reading sensor data. It controlled the n-channel MOSFET to implement frequency and duty cycle accordingly.

We used multiple sensors to monitor the state of the battery. The algorithm is controlled by these sensor data as well. The current sensor was used to measure injected current in the battery. The voltage sensor is to read the terminal voltage of the battery. The temperature sensor helps to monitor the temperature of the battery.

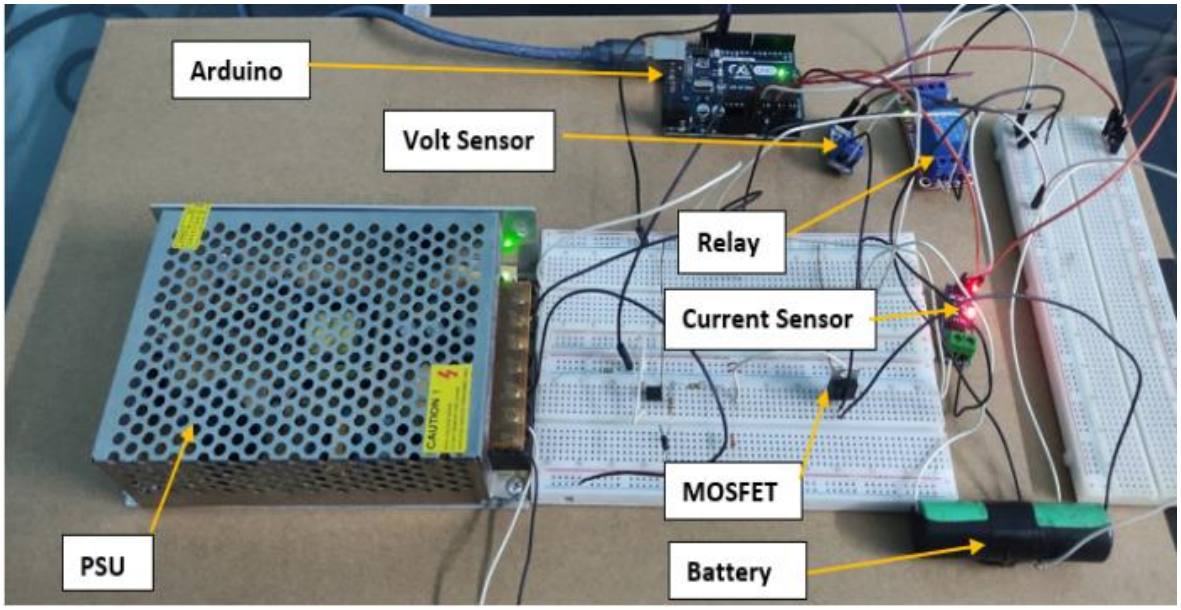


*Figure 5.1: Test Setup*

Moreover, a relay was used and controlled by the microcontroller to detach the battery from the charging circuit. The battery will disconnect from the circuit when it reaches 80% SoC. Also, this relay works as an overheat protection. If the temperature of the battery rises more than 45 degrees, the relay will disconnect the battery from the charging circuit. The voltage reading operation is done when the battery is disconnected from the charging circuit with the help of a relay.

The power supply was the source of our charging circuit. It supplied the voltage and current accordingly. Some other component was used for the short circuit protection of the battery as well

## 5.3 Data Acquisition

The sensors read the data and send it to the microcontroller. Then, the microcontroller sends the data to the host computer. The voltage data was sent after a minute of charging. The current data was sent with a one second interval. The temperature data was sent with one minute interval.

## 5.4 Test Result without Delay

In this test, we charged the battery without any delay. The circuit continuously searched for the optimal frequency and duty cycle according to the SoC and charged the battery. For this test, we took measurement of charging time, current, frequency, duty cycle and temperature. All the data are presented below with graphs and proper explanation.

### 5.4.1 Charging Time

Our prototype charged up the battery within 91 minutes or almost 1 hour and 31 minutes. In this graph, the x axis represents time and y axis represents the terminal voltage of the battery. The terminal voltage represents the SoC as well.

| State of Charge | Terminal Voltage |
|---|---|
| 0 - 20% | 2.40v – 3.50v |
| 21 – 40% | 3.51v – 3.65v |
| 41 – 50% | 3.66v - 3.72v |
| 51 – 55% | 3.73v - 3.77v |
| 56 – 60% | 3.78v - 3.81v |
| 61 – 65% | 3.82v - 3.86v |
| 66 – 70% | 3.87v - 3.90v |
| 71 – 75% | 3.91v – 3.95v |
| 76 – 80% | 3.96v – 4.00v |

*Table 4: SoC vs Terminal Voltage*

At the very beginning, the terminal voltage rises very fast. Because the early voltage levels represent lower state of charge. But once the terminal voltage rises to 3.4 volt, which is close to 20% SoC, the rise in voltage reduced significantly. It happened as the battery reached to higher SoC levels.

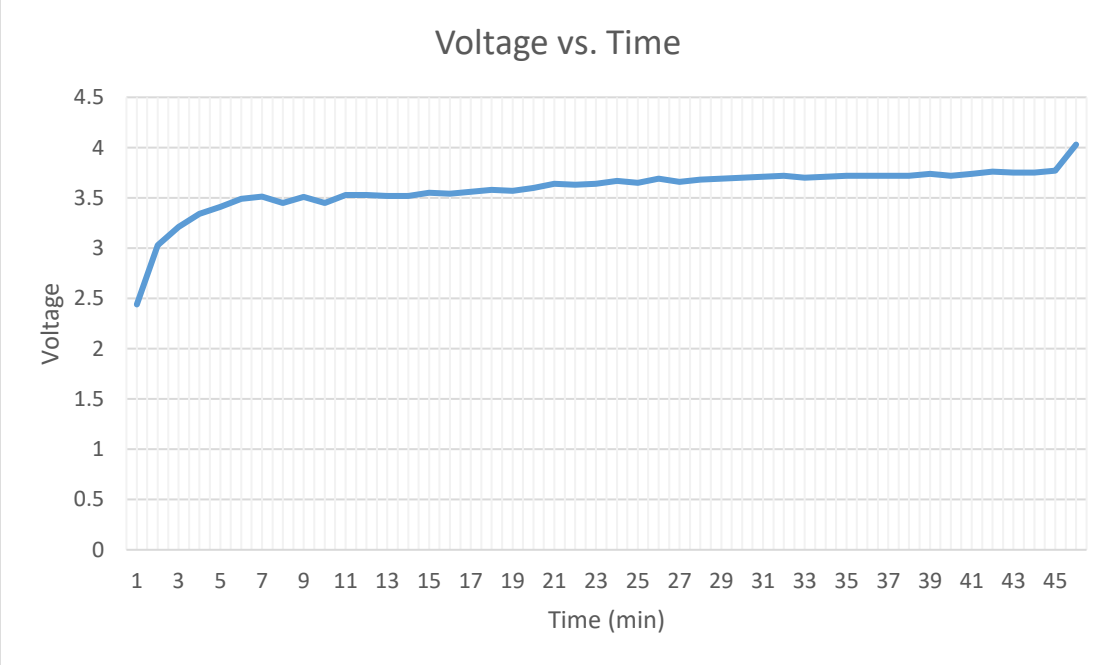


*Figure 5.2: Charging time with respect to terminal voltage without delay*

Here in the graph, we can see that the voltage fell down couple of times. This happened because we did not implement any delay before reading the voltage sensor reading. The reading we got was not stable enough to give us correct reading.

### 5.4.2 Injected Current

In this graph, the x axis represents time and the y axis represents the injected current into the battery. Initially the terminal voltage of the battery was very low, the current supplied by the power supply was high. According to the algorithm, the charging current should have been 1.25 amps. But in the searching part of the algorithm, low current was not found. So, the prototype charged the battery with high current, which was closer to the marginal current (1.25A), as per as the instruction. The current injection was reducing with time. As a result, our charging circuit took a while to reach 80% SoC.

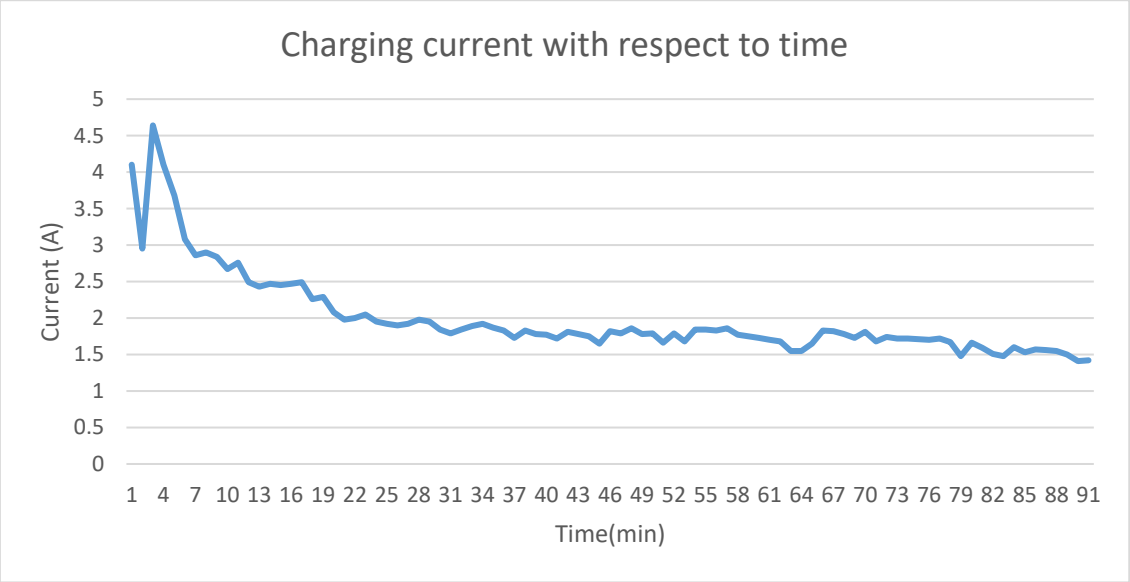


*Figure 5.3: Charging current with respect to time without delay*

### 5.4.3 Optimal Frequency Data

The algorithm searched for the most optimal frequency before searching for duty cycle. It searched for 500 to 5000 Hz with an increment of 500 Hz. In this searching procedure, optimal

frequency was set when the current sensor measured the highest current. Then, the algorithm started the search for optimal duty cycle.

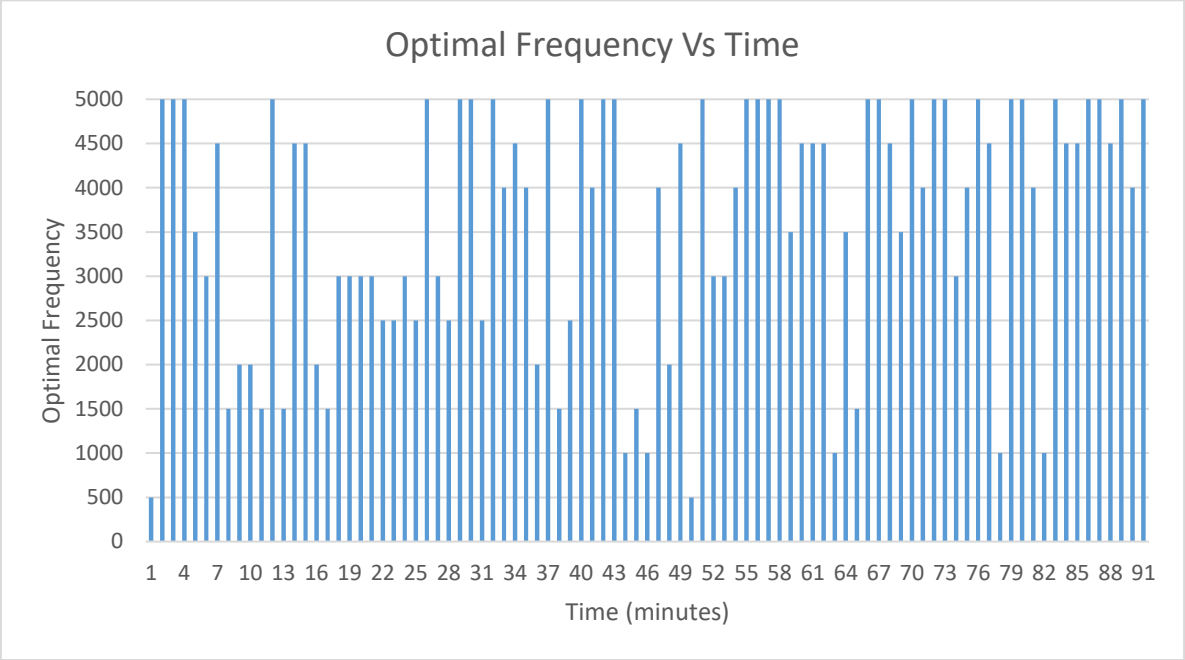


*Figure 5.4: Optimal frequency for each minute of charging without delay*

Here, in this graph, the frequency is in x axis and the time is in y axis. As we can see, the frequency did not follow any particular pattern. When the current sensor sensed highest current, the microcontroller set that frequency value as optimal frequency.

### 5.4.4 Optimal Duty Cycle Data

The next operation for the microcontroller is to search for the most optimal duty cycle. For this operation the frequency is set to the optimal frequency which was obtained earlier. A margin was considered in order to search for optimal duty cycle. A li-ion battery has a maximum acceptable current according to the state of charge. Algorithm searched for a duty cycle for which the current was closest to that maximum acceptable current. As a result, polarization voltage would be minimum even with high charging current.

The graph shows us time in y axis and duty cycle in x axis. Initially duty cycle was high then it settles down to 10% for the most of the time.

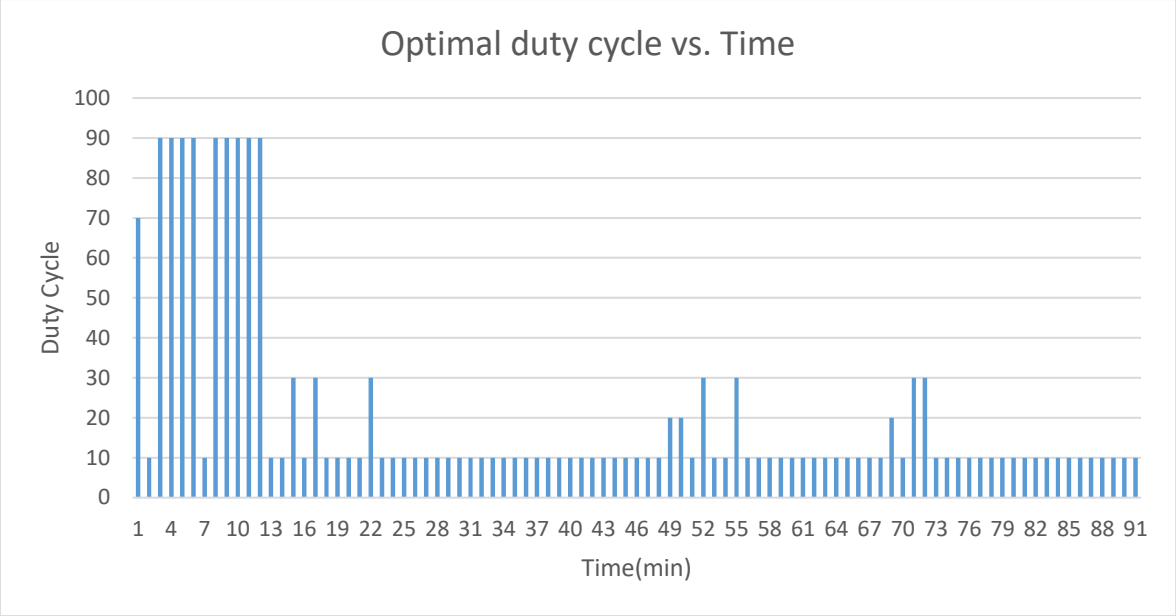


*Figure 5.5: Optimal duty cycle for each minute of charging without delay*

### 5.4.5 Temperature Data

High current always introduces high temperature. In our prototype, there is no difference. The ambient temperature for our test was 30 degree Celsius. A temperature sensor was directly attached to the surface of the battery to measure the battery temperature so that we can get the exact temperature of the battery with respect to time and current injection.

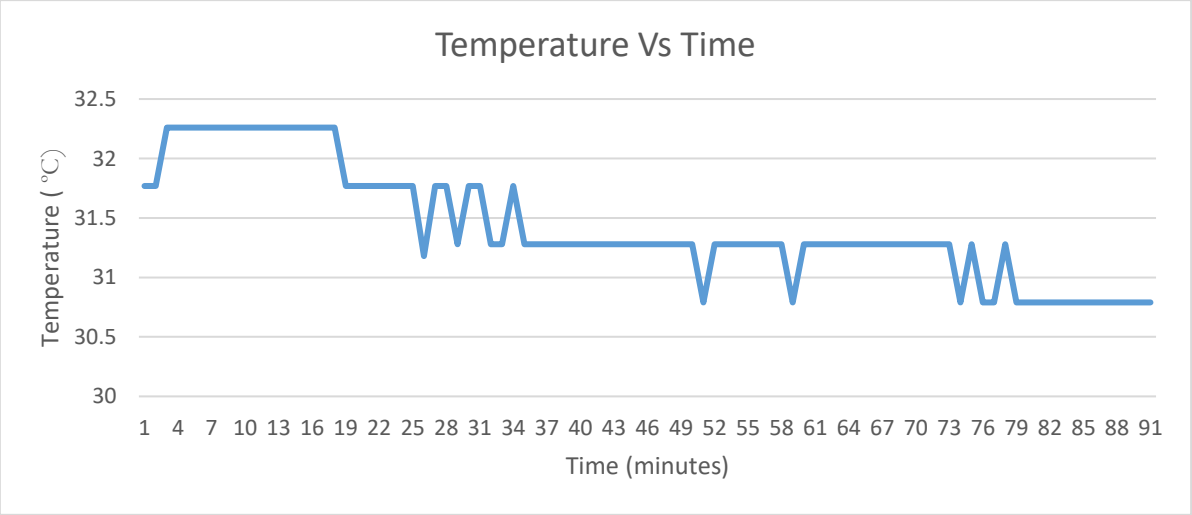


*Figure 5.6: Temperature reading from the sensor without delay*

From the charging current graph (figure 5.2), we can see that the initial current was high. As a result, the battery got hot and the temperature rose up to 32.26 degree Celsius. The battery was getting cool as the current injection was decreasing. It was stable at 31.28 degree Celsius. Then, the temperature got as low as 30.79 degree Celsius as the current inject decreased even more.

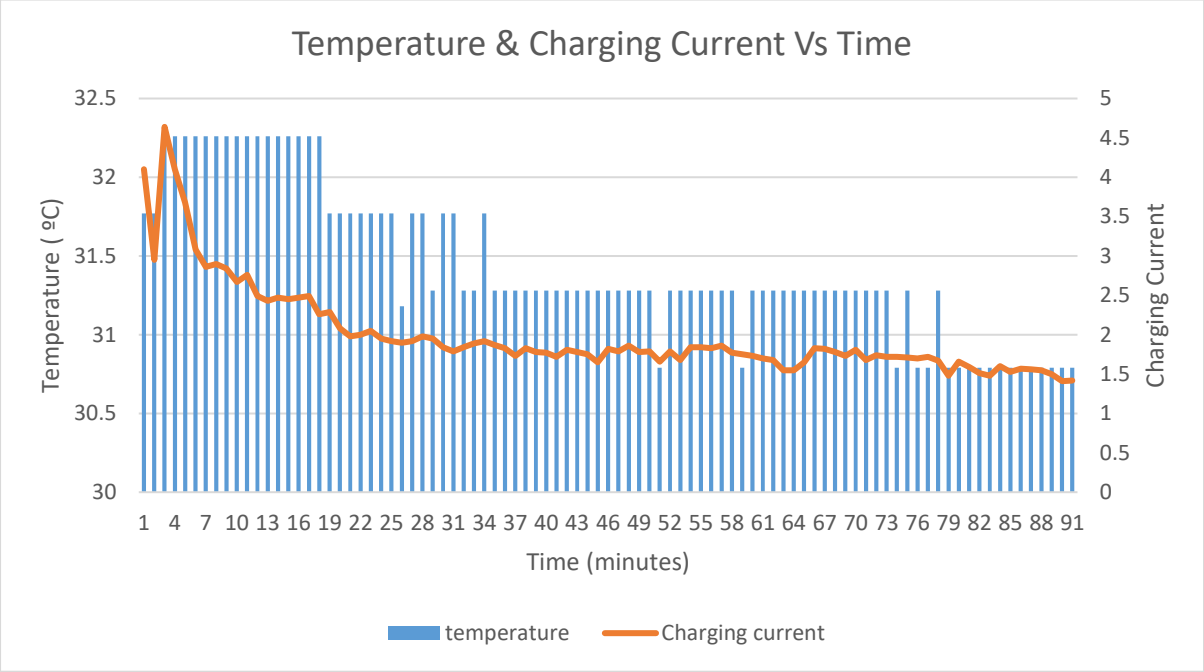


*Figure 5.7: Temperature analysis with current injection without delay*

## 5.5 Test Result with Delay

As we know, the lithium-ion battery needs time to settle down the terminal voltage. The battery needs 1000 seconds [10] to settle down its terminal voltage. In our algorithm, we measured the voltage to determine the SoC. If there are not 1000 seconds of delay before measuring the terminal voltage, chances are the sensor will not give us the perfect value. So, in this test, we included a 17 minutes delay before reading voltage sensor data. This process took much time to charge the battery as we had to wait for 17 minutes before reading voltage sensor data each time. The data retrieved from the sensor is discussed below.

### 5.5.1 Charging Time

The charging current dramatically increases when we use that 17-minute delay for voltage measurement. The x-axis of the graph represents time and the y axis represents injected current. Surprisingly, this time the charging time is quite low. Our prototype was able to charge the battery in only 24 minutes. The 17 minutes delay for each voltage reading was not considered, since the battery was disconnected the whole time with the help of a relay. The battery started with a terminal voltage of 2.77 which is 0% SoC. Then it reached 4.0v within 24 minutes, which is 80% SoC. Initially, the terminal voltage was rising with sharper slop but then it started to rise with linear slop.

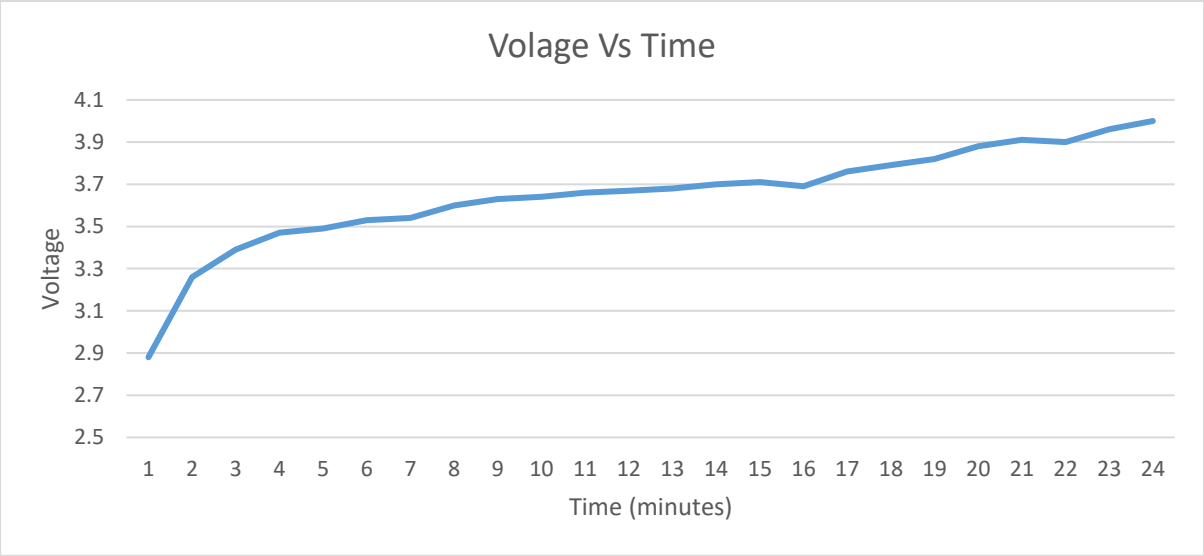


*Figure 5.8: Charging time with respect to voltage with delay*

### 5.5.2 Charging Current

The x-axis of the graph is time and the y axis represents current injected in the battery. It was measured by a current sensor which was connected in between the battery and the power supply. At the beginning of the charging, the injected current was high. Then, the current got stable in the 3$^{rd}$ minute of charging at around 4 amps. Current started to rise again at 14$^{th}$ minute

and got much lower after 22 minutes of charging. Altogether it took 24 minutes to charge the battery with our prototype.

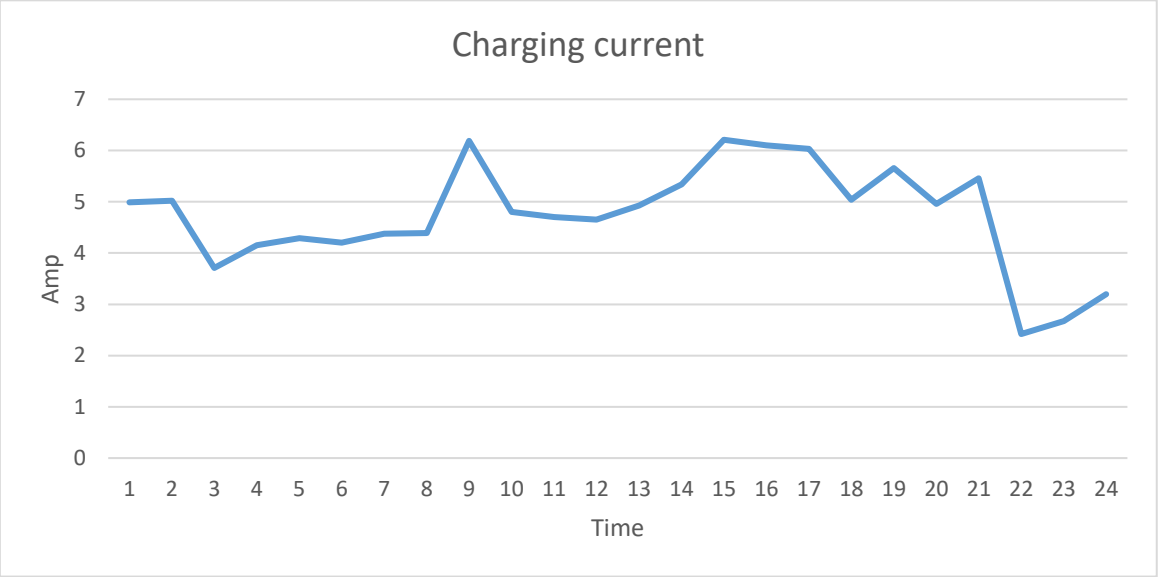


*Figure 5.9: Charging current with respect to time with delay*

According to our maximum acceptable current chart, the current was supposed to be much lower than what we got. The algorithm could not find any lower current neither in frequency search nor in duty cycle search. So, the prototype charged the battery with the closest current that could be found in the searching procedure.

### 5.5.3 Optimal Frequency Data

The algorithm searched for a frequency for which the power supply will be able to inject the highest amount of current. It searched for it in the range of 500 to 5000 Hz. Initially, the optimal frequency was found in lower frequency. At the end of the charge, the frequency was relatively high. If we compare it to the current injection, the current was high as well during this time, except for the last couple of minutes of charging.

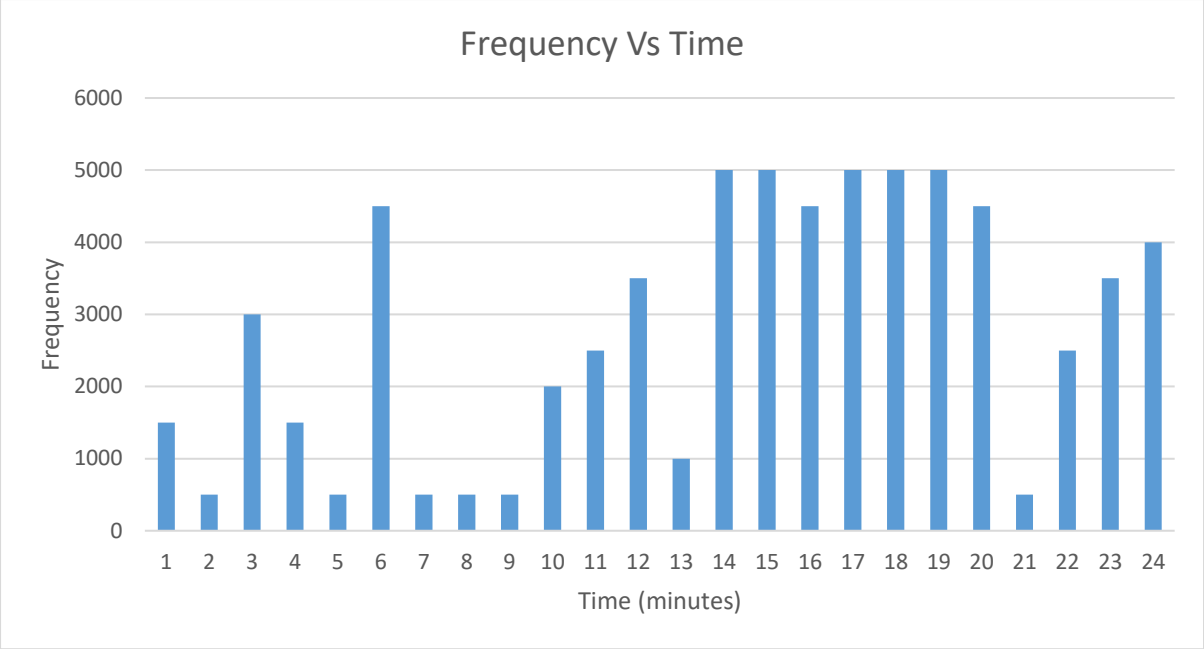


*Figure 5.10: Optimal frequency for each minute of charging with delay*

### 5.5.4 Optimal Duty Cycle Data

Duty cycle is another vital part for our prototype. With the optimal frequency, the algorithm search for optimal duty cycle. With each pulse, the battery gets charged and gets relaxation period as well for a short period of time.

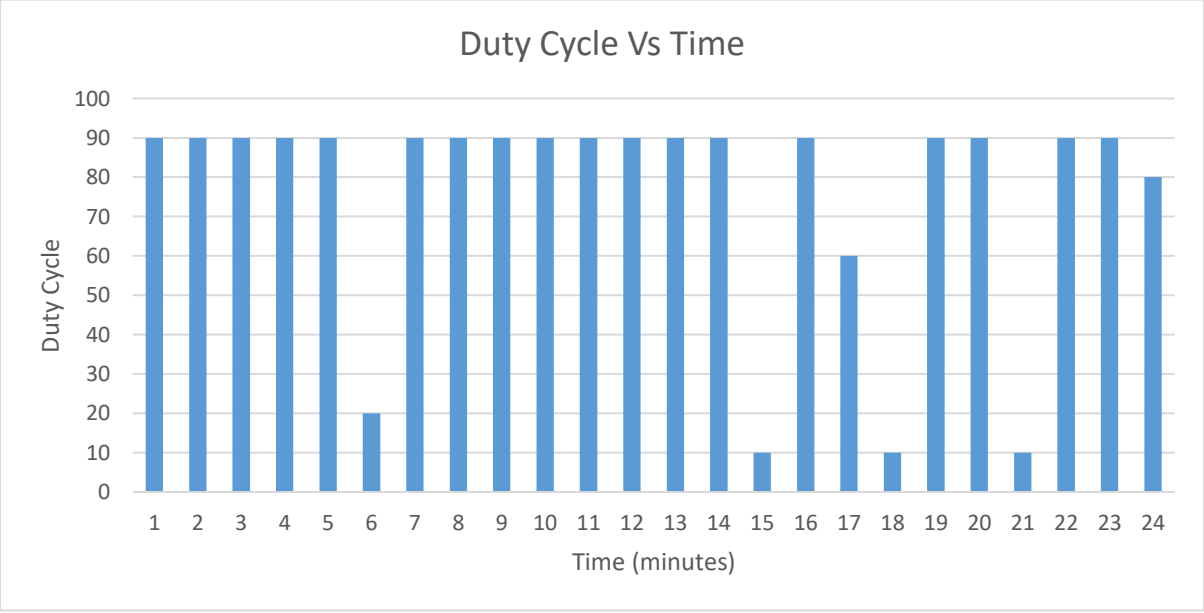


*Figure 5.11: Optimal duty cycle for each minute of charging with delay*

The algorithm search from 10% to 90% duty cycle with an increment of 10%. The closest current data according to the margin is considered as optimal charging current and the respective duty cycle is considered as the optimal duty cycle. In this graph, we can observe the duty cycle change with each minute of time. Most of the time the optimal duty cycle was 90%.

### 5.5.5 Temperature

The ambient temperature for this test was 30 degree Celsius. Although the current was very high, the battery did not heat up at all. This happened because of the 17 minutes of delay we put for measuring the terminal voltage of the battery. After the delay, the battery got charged only for 1 minute. This is very low compare to the delay. So, the battery could not get hot at all. The sensor, which is attached directly to the battery surface, showed us the same data all the time. The temperature did not increase or decrease at all.

## 5.6 Result Analysis

We tested our prototype with two different approaches. At first, we tried to charge the battery without any delay for battery terminal voltage reading. Instead, we disconnected the battery for a brief period of time to get the voltage reading. As we know, the lithium-ion battery needs a thousand second to settle down its terminal voltage. As a result, we could not inject a high current according to our maximum acceptable current chart. Our charging time was high compared to the simulation done by [20].

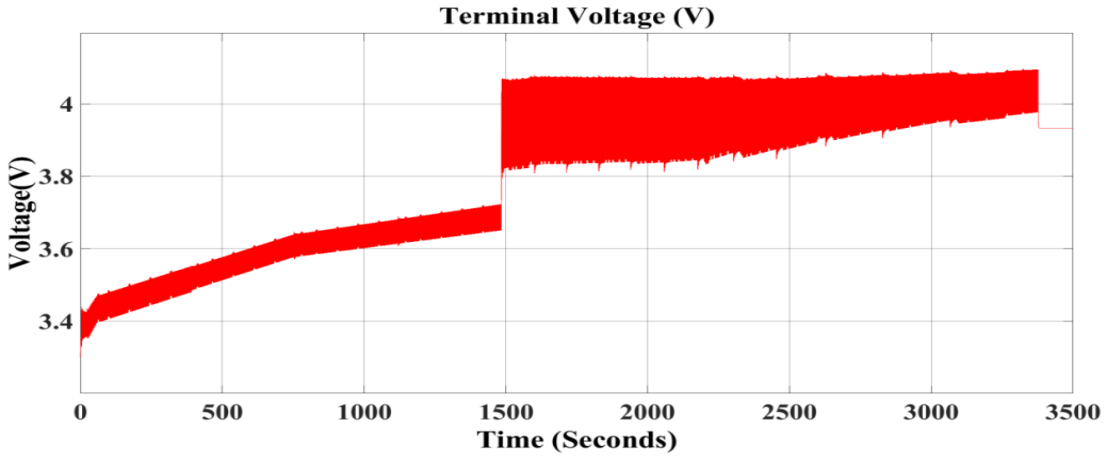


*Figure 5.12: Simulation charging time against the terminal voltage of the battery [20]*

Then, we tried to charge the battery with that thousand-second delay so that the terminal voltage could settle down before charging. This time we succeed. We were able to charge the battery within only 24 minutes, where the simulation took 58 minutes to charge the battery to 80% of SoC. It is 34 minutes faster than the simulation [20]. We have also made progress in charging time by 32 minutes from the hardware implementation done in [10]. We were able to inject even higher current in the battery with our prototype.

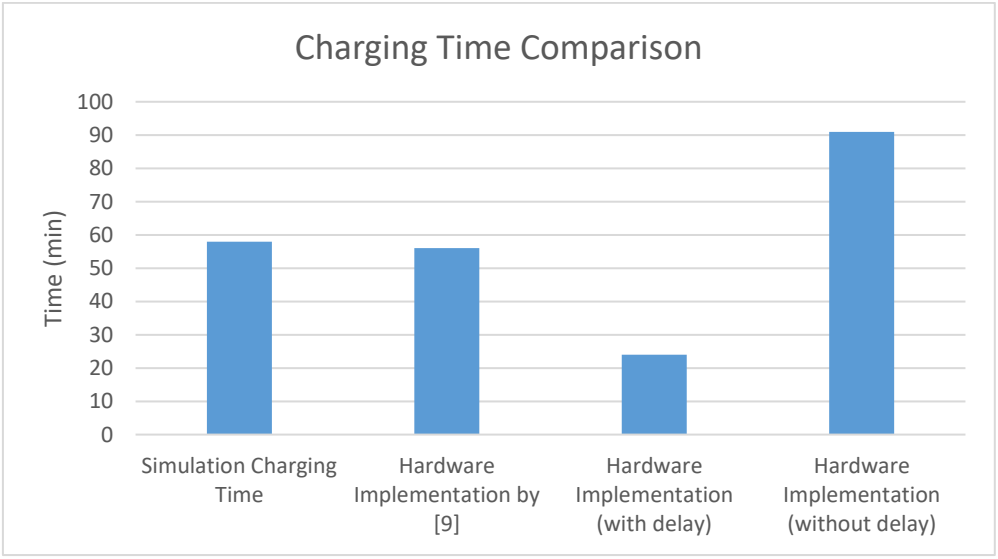


*Figure 5.13: Comparison between simulation and hardware implementations*

Even though we were able to reduce the charging time significantly, it needed 17 minutes of delay before each charge. The temperature did not rise as the battery got time to cool down.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

With the motivation to charge the battery faster than the CC-CV method, we implemented the project with pulse method. As we could not get the lab facility in this pandemic situation, we did the project in our house environment and with limited resources. The partial data collecting on the pulse-based charging algorithm is the beginning of this project. By analyzing the data, we hope that we can charge better. The results also show very low temperature rise of the battery. With the lab environment and availability of the components we believe that we could reach our goal of charging battery in fast and secure way.

## 6.2 Future Work

One of the limitations of our prototype was we could not inject high current with continuous charging. On the other hand, we were able to inject high current when we inserted a delay before each minute of charge to get the reading of the terminal voltage. Now we were able to charge the battery in 24 minutes where the previous method needed 91 minutes. So, we should work on how to inject a high current into the battery without any delay in between.

The study we did was only for a specific type of battery. We need to do the same test with different battery to learn more about the behavior of the lithium-ion battery for pulse charging. A database should be created to make one universal charger for all the lithium-ion battery. The charger will detect the specification. Then, it will obtain the maximum acceptable current data and charge the battery accordingly. This will be revolutionary for the lithium-ion battery charger. We will be able to charge any lithium-ion battery with one single universal battery. This will be more user friendly. Moreover, it will save the industry a huge amount of investment. They will not need to design different types of battery charger any more.

To implement our prototype for commercial purpose, the industry should not use all the sensor that we used. Instead, they should charge the battery according to the data we got from the test. A new charger should be designed which change the frequency and duty will cycle with respect to the state of charge of the battery.

Nowadays a new industry trend is going on. The battery of the smartphone is being divided into two pieces. Our study will help them as well.

# References


[1] J. C. Burns, A. Kassam, N. N. Sinha, L. E. Downie, L. Solnickova, B. M. Way, and J. R. Dahn, “Predicting and Extending the Lifetime of Li-Ion Batteries,” *Journal of The Electrochemical Society*, vol. 160, no. 9, 2013.

[2] Kuchhal P, Sharma UC. Battery waste management. In: Gurjar BR, Sharma UC, Singh N, editors. Environmental Science and Engineering Vol. 5 Solid Waste Management, Texas: Studium Press LLC; 2017, p. 141−155.

[3] M. Shen, M. Ren, Y. Wang, F. Shen, R. Du, L. Quan, Y. Wei, T. Zhang, J. Li, G. Yan, J. Peng, and Z. Cao, “Identifying dust as the dominant source of exposure to heavy metals for residents around battery factories in the Battery Industrial Capital of China,” *Science of The Total Environment*, vol. 765, p. 144375, 2021.

[4] K. Asakura, M. Shimomura, and T. Shodai, “Study of life evaluation methods for Li-ion batteries for backup applications,” *Journal of Power Sources*, vol. 119-121, pp. 902–905, 2003.

[5] A. Tomaszewska, Z. Chu, X. Feng, S. O'Kane, X. Liu, J. Chen, C. Ji, E. Endler, R. Li, L. Liu, Y. Li, S. Zheng, S. Vetterlein, M. Gao, J. Du, M. Parkes, M. Ouyang, M. Marinescu, G. Offer, and B. Wu, “Lithium-ion battery fast charging: A review,” eTransportation, vol. 1, p. 100011, 2019.

[6] X. Hu, L. Xu, X. Lin, and M. Pecht, “Battery Lifetime Prognostics,” *Joule*, vol. 4, no. 2, pp. 310–346, 2020.

[7] J. A. Carcone, "Performance of lithium-ion battery systems", *Proc. WESCON Idea/Microelectron. Conf. Rec.*, pp. 242-248, 1994-Sep.

[8] M. Ye, H. Gong, R. Xiong, and H. Mu, “Research on the Battery Charging Strategy with Charging and Temperature Rising Control Awareness,” IEEE Access, vol. 6, pp. 64193–64201, 2018.

[9] J. Amanor-Boadu, A. Guiseppi-Elie, and E. Sánchez-Sinencio, “The Impact of Pulse Charging Parameters on the Life Cycle of Lithium-Ion Polymer Batteries,” Energies, vol. 11, no. 8, p. 2162, 2018.

[10] M. Yin, J. Cho, and D. Park, “Pulse-Based Fast Battery IoT Charger Using Dynamic Frequency and Duty Control Techniques Based on Multi-Sensing of Polarization Curve,” Energies, vol. 9, no. 3, p. 209, 2016.

[11] V. Müller, R. Kaiser, S. Poller, and D. Sauerteig, “Importance of the constant voltage charging step during lithium-ion cell formation,” *Journal of Energy Storage*, vol. 15, pp. 256–265, 2018.

[12] A. Cuadras and O. Kanoun, “SoC Li-ion battery monitoring with impedance spectroscopy,” *2009 6th International Multi-Conference on Systems, Signals and Devices*, 2009.

[13] B. K. Purushothaman and U. Landau, “Rapid Charging of Lithium-Ion Batteries Using Pulsed Currents,” *Journal of The Electrochemical Society*, vol. 153, no. 3, 2006.

[14] David H Wilson, "Method of charging storage batteries," US Patent US1126667A, June 27. 1914.

[15] F.B. Diniz, L.E.P. Borges and B.d.B. Neto, "A comparative study of pulsed current formation for positive plates of automotive lead acid batteries," Journal of Power Sources, vol. 109, pp. 184-188, 2002.

[16] L.T. Lam, H. Ozgun, O.V. Lim, J.A. Hamilton, L.H. Vu, D.G. Vella and D.A.J. Rand, "Pulsed-current charging of lead/acid batteries — a possible means for overcoming premature capacity loss?" Journal of Power Sources, vol. 53, pp. 215-228, 1995.

[17] H. Ikeda, S. Minami, S.J. Hou, Y. Onishi and A. Kozawa, "Nobel High Current Pulse Charging Method for Prolongation of Lead-acid Batteries," Journal of Asian Electric Vehicles, vol. 3, pp. 681-687, 2005.

[18] L.-R. Chen, "A Design of an Optimal Battery Pulse Charge System by Frequency-Varied Technique," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 398–405, 2007.

[19] Chen, L.R. Design of Duty-Varied Voltage Pulse Charger for Improving Li-Ion Battery-Charging Response. IEEE Trans. Ind. Electron. 2009, 56, 480–487

[20] AKM Ferdous, M. Morshed, P. Saha and K. F. Reaz, "Designing a Fast Charging System for Lithium-Ion Batteries Based on Double Pulse Method Using Optimized Regulation of Frequency and Duty Cycle" B.S. Thesis, Dept. Elect. Eng., Brac Univ., Dhaka, Bangladesh, 2020.

[21] "Basic Working Principle of Relay - Construction and Types," *Circuit Digest*, 17-Nov-2020. [Online]. Available: https://circuitdigest.com/article/relay-working-types-operation-applications. [Accessed: 19-May-2021].

[22] "5V Single-Channel Relay Module - Pin Diagram, Specifications, Applications, Working, "*Components101*. [Online]. Available: https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet.

[23] Y. N. Ning, Z. P. Wang, A. W. Palmer, K. T. Grattan, and D. A. Jackson, “Recent progress in optical current sensing techniques,” Review of Scientific Instruments, vol. 66, no. 5, pp. 3097–3111, 1995.

[24] B. Schweber, “Using Resistors for Current Sensing: It’s More Than Just I = V/R,” *StackPath*, 29-Mar-2018. [Online]. Available: https://www.powerelectronics.com/technologies/power-management/article/21864130/using-resistors-for-current-sensing-its-more-than-just-i-vr. [Accessed: 20-May-2021].

[25] “ACS712 Current Sensor Module Pinout, Specifications, Circuit & Datasheet,” *Components101*. [Online]. Available: https://components101.com/sensors/acs712-current-sensor-module. [Accessed: 20-May-2021].

[26] Electrical4U, “Voltage Sensor: What is it And How Does it Work? ” *Electrical4U*, 31-Jan-2021. [Online]. Available: https://www.electrical4u.com/voltage-sensor/. [Accessed: 20-May-2021].

[27] “Voltage Sensor,” *Voltage Sensor - an overview | ScienceDirect Topics*. [Online]. Available: https://www.sciencedirect.com/topics/engineering/voltage-sensor. [Accessed: 20-May-2021].

[28] S. S. Thomas, A. Saraswat, A. Shashwat, and V. Bharti, “Sensing heart beat and body temperature digitally using Arduino,” *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 2016.

[29] E. G. Projects, “LM35 Temperature Sensor Pin out, Interfacing guide, Circuit Construction and Working Principals,” *Engineers Garage*, 10-May-2021. [Online]. Available: https://www.engineersgarage.com/electronic-projects/lm35-description-and-working-principal/. [Accessed: 20-May-2021].

[30] “INR18650-25R Battery. Datasheet pdf. Equivalent,” *INR18650-25R Battery Datasheet pdf - Lithium-ion Battery. Equivalent, Catalog*. [Online]. Available: https://datasheetspdf.com/pdf/839321/Samsung/INR18650-25R/1. [Accessed: 19-May-2021].

[31] E. Granath “What's the difference between AC-DC and DC-DC power supplies?” [Online]. Available: https://www.power-and-beyond.com/whats-the-difference-between-ac-dc-and-dc-dc-power-supplies-a-915209/ [Accessed May. 20, 2021].

# Appendix A.

## Code

```
const int currentSensor = A0;
float a = 0.0;

const int voltageSensor = A2;
float v = 0.0;

const int tempSensor=A1;
float tempc=0.0;

const int relayPin = 13;

int outputpin=9; /* Assign symbolic name outputpin to D9 PWM pin of Arduino */

float soc=0;
float current=0;

int chargingTime;

float optimal_frequency_20;
float optimal_dutycycle_20;
float optimal_dutycycle_value_20;
float optimal_frequency_current_20;
float optimal_dutycycle_current_20;

float optimal_frequency_40;
float optimal_dutycycle_40;
float optimal_dutycycle_value_40;
float optimal_frequency_current_40;
float optimal_dutycycle_current_40;

float optimal_frequency_50;
float optimal_dutycycle_50;
float optimal_dutycycle_value_50;
float optimal_frequency_current_50;
float optimal_dutycycle_current_50;

float optimal_frequency_55;
float optimal_dutycycle_55;
float optimal_dutycycle_value_55;
float optimal_frequency_current_55;
float optimal_dutycycle_current_55;

float optimal_frequency_60;
float optimal_dutycycle_60;
float optimal_dutycycle_value_60;
float optimal_frequency_current_60;
float optimal_dutycycle_current_60;

float optimal_frequency_65;
float optimal_dutycycle_65;
float optimal_dutycycle_value_65;
float optimal_frequency_current_65;
float optimal_dutycycle_current_65;
```

```
float optimal_frequency_70;
float optimal_dutycycle_70;
float optimal_dutycycle_value_70;
float optimal_frequency_current_70;
float optimal_dutycycle_current_70;

float optimal_frequency_75;
float optimal_dutycycle_75;
float optimal_dutycycle_value_75;
float optimal_frequency_current_75;
float optimal_dutycycle_current_75;

float optimal_frequency_80;
float optimal_dutycycle_80;
float optimal_dutycycle_value_80;
float optimal_frequency_current_80;
float optimal_dutycycle_current_80;

//maximum accepatable current
float mac_20=1.25;//1.188; //1.25; //2.8-3.5
float mac_40=4.0;//3.8; //4.0; //3.5-3.65
float mac_50=3.865;//3.665; //3.858; //3.65-3.72
float mac_55=3.539;//3.3621; //3.539; //3.72-3.77
float mac_60=2.979;//2.83; //2.979; //3.77-3.81
float mac_65=2.659;//2.526; //2.659; //3.81-3.86
float mac_70=2.319;//2.203; //2.319; //3.86-3.9
float mac_75=1.959;//1.861; //1.959; //3.9-3.95
float mac_80=1.5;//1.425; //1.5; //3.95-4

void setup() {
  Serial.begin(9600);
  pinMode(outputpin, OUTPUT); /* set as a output put */
  TCCR1A=(1<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (1<<WGM11) | (0<<WGM10);
  TCCR1B=(0<<ICNC1) | (0<<ICES1) | (1<<WGM13) | (1<<WGM12) | (0<<CS12) | (0<<CS11) | (1<<CS10);
  TCNT1=0;
  pinMode(relayPin, OUTPUT);
  pinMode(tempSensor,INPUT);
  //Serial.println("CLEARDATA");
  Serial.println("LABEL,Date,Time,Open Circuit VOltage,Current,Frequency,Duty Cycle");
}

float voltage_read(){
  for(int j=0;j<1;j++){
    float vOUT = 0.0;
    float vIN = 0.0;
    float vSUM = 0.0;
    float R1 = 30000.0; //29600
    float R2 = 7500.0; //7590
    float value = 0;
    //delay (60000);
    digitalWrite(relayPin, LOW);
    for (int i=0;i<2000;i++){
      value = analogRead(voltageSensor);
      vOUT = (value * 5.0) / 1024.0;
      vIN = vOUT / (R2/(R1+R2));
      vSUM = vSUM + vIN;
      delay (1);
    }
    v = (vSUM/2000.0);
```

```
    Serial.print("Volts = "); // shows the voltage measured
    Serial.println(v,2); // the '2' after voltage allows you to display 2 digits after decimal point
   }

  digitalWrite(relayPin, HIGH);
  return v;

}

float current_read(){
  //float a = 0.0;
  float mVperAmp = 66; // use 100 for 20A Module and 66 for 30A Module
  float RawValue= 0.0;
  float ACSoffset = 2500.0;
  float Voltage = 0.0;
  float Amps = 0.0;
  float AmpsSum = 0.0;
  for(int i=0;i<1000;i++){
    RawValue = analogRead(currentSensor);
    Voltage = (RawValue / 1024.0) * 5000.0; // Gets you mV
    Amps = ((Voltage - ACSoffset) / mVperAmp);
    AmpsSum = AmpsSum + Amps;
    delay (1);
  }
  a = AmpsSum/1000.0;

  if(a<0){
    a=-a;
  }

  Serial.println("This is current reading part of the code");
  //Serial.print("charging current = "); // shows the cuurent measured
  //Serial.println(a,2); // the '2' after current allows you to display 2 digits after decimal point
  //Serial.print("freq = "); // shows the voltage measured
  //Serial.println(ICR1);
  //Serial.print("dc = "); // shows the voltage measured
  //Serial.println(OCR1A);

  return a;
}

float temperatureRead(){
  float vout;
  vout=analogRead(tempSensor);
  vout=(vout*500)/1023;

  Serial.print("Temperature = ");
  Serial.println(vout,2);

  return vout;
}

void batteryParameters(){
  soc=voltage_read();
  Serial.print("DATA,DATE,TIME,");
  Serial.print(soc,2);
  Serial.print(",");
  Serial.println(a,2);
  //Serial.print(",");
}
```

```
void frequency_search_20(){
  float frequency_current_array_20[10];
  float frequency_icr1_array_20[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_20[i]=current;
    frequency_icr1_array_20[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
    if(frequency_current_array_20[0]<frequency_current_array_20[i]){
      frequency_current_array_20[0]=frequency_current_array_20[i];
      frequency_icr1_array_20[0]=frequency_icr1_array_20[i];
    }
  }
  optimal_frequency_current_20=frequency_current_array_20[0];
  optimal_frequency_20=frequency_icr1_array_20[0];
  Serial.print("Optimal Current for frequency = ");
  Serial.println(optimal_frequency_current_20,2);
  Serial.print("Optimal Frequency = ");
  Serial.println(16000000/optimal_frequency_20);
}

void dutycycle_search_20(){
  float dutycycle_current_array_20[9];
  float dutycycle_ocr1a_array_20[9];
  float diff[9];
  for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_20;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_20[i]=current;
    dutycycle_ocr1a_array_20[i]=OCR1A;
    diff[i]=dutycycle_current_array_20[i]-mac_20;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_20[0]=dutycycle_ocr1a_array_20[i];
      dutycycle_current_array_20[0]=dutycycle_current_array_20[i];
    }
  }
  optimal_dutycycle_20=dutycycle_ocr1a_array_20[0];
  optimal_dutycycle_current_20=dutycycle_current_array_20[0];
```

```
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_20,2);
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_20=((optimal_dutycycle_20/optimal_frequency_20)*100);
  Serial.println(optimal_dutycycle_value_20);
}

void charging_20(){
  while(1){
    ICR1=optimal_frequency_20;
    OCR1A=optimal_dutycycle_20;
    Serial.print("optimal_frequency_20=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_20=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    ////batteryParameters();
    if(voltage_read()>3.50){
      break;
    }
    frequency_search_20();
    dutycycle_search_20();
  }
}

void frequency_search_40(){
  float frequency_current_array_40[10];
  float frequency_icr1_array_40[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_40[i]=current;
    frequency_icr1_array_40[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
    if(frequency_current_array_40[0]<frequency_current_array_40[i]){
      frequency_current_array_40[0]=frequency_current_array_40[i];
      frequency_icr1_array_40[0]=frequency_icr1_array_40[i];
    }
  }
  optimal_frequency_current_40=frequency_current_array_40[0];
  optimal_frequency_40=frequency_icr1_array_40[0];
  Serial.print("Optimal Current for frequency = ");
```

```
  Serial.println(optimal_frequency_current_40,2);
  Serial.print("Optimal Frequency = ");
  Serial.println(16000000/optimal_frequency_40);
}

void dutycycle_search_40(){
  float dutycycle_current_array_40[9];
  float dutycycle_ocr1a_array_40[9];
  float diff[9];
  for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_40;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_40[i]=current;
    dutycycle_ocr1a_array_40[i]=OCR1A;
    diff[i]=dutycycle_current_array_40[i]-mac_40;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_40[0]=dutycycle_ocr1a_array_40[i];
      dutycycle_current_array_40[0]=dutycycle_current_array_40[i];
    }
  }
  optimal_dutycycle_40=dutycycle_ocr1a_array_40[0];
  optimal_dutycycle_current_40=dutycycle_current_array_40[0];
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_40,2);
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_40=((optimal_dutycycle_40/optimal_frequency_40)*100);
  Serial.println(optimal_dutycycle_value_40);
}

void charging_40(){
  while(1){
    ICR1=optimal_frequency_40;
    OCR1A=optimal_dutycycle_40;
    Serial.print("optimal_frequency_40=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_40=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
```

```
    //batteryParameters();
    if(voltage_read()>3.65){
      break;
    }
    frequency_search_40();
    dutycycle_search_40();
  }
}

void frequency_search_50(){
  float frequency_current_array_50[10];
  float frequency_icr1_array_50[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_50[i]=current;
    frequency_icr1_array_50[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
    if(frequency_current_array_50[0]<frequency_current_array_50[i]){
      frequency_current_array_50[0]=frequency_current_array_50[i];
      frequency_icr1_array_50[0]=frequency_icr1_array_50[i];
    }
  }
  optimal_frequency_current_50=frequency_current_array_50[0];
  optimal_frequency_50=frequency_icr1_array_50[0];
  Serial.print("Optimal Current for frequency = ");
  Serial.println(optimal_frequency_current_50,2);
  Serial.print("Optimal Frequency = ");
  Serial.println(16000000/optimal_frequency_50);
}

void dutycycle_search_50(){
  float dutycycle_current_array_50[9];
  float dutycycle_ocr1a_array_50[9];
  float diff[9];
  for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_50;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_50[i]=current;
    dutycycle_ocr1a_array_50[i]=OCR1A;
    diff[i]=dutycycle_current_array_50[i]-mac_50;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
```

```
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_50[0]=dutycycle_ocr1a_array_50[i];
      dutycycle_current_array_50[0]=dutycycle_current_array_50[i];
    }
  }
  optimal_dutycycle_50=dutycycle_ocr1a_array_50[0];
  optimal_dutycycle_current_50=dutycycle_current_array_50[0];
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_50,2);
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_50=((optimal_dutycycle_50/optimal_frequency_50)*100);
  Serial.println(optimal_dutycycle_value_50);
}

void charging_50(){
  while(1){
    ICR1=optimal_frequency_50;
    OCR1A=optimal_dutycycle_50;
    Serial.print("optimal_frequency_50=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_50=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    //batteryParameters();
    if(voltage_read()>3.72){
      break;
    }
    frequency_search_50();
    dutycycle_search_50();
  }
}

void frequency_search_55(){
  float frequency_current_array_55[10];
  float frequency_icr1_array_55[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_55[i]=current;
    frequency_icr1_array_55[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
```

```
    if(frequency_current_array_55[0]<frequency_current_array_55[i]){
      frequency_current_array_55[0]=frequency_current_array_55[i];
      frequency_icr1_array_55[0]=frequency_icr1_array_55[i];
    }
  }
  optimal_frequency_current_55=frequency_current_array_55[0];
  optimal_frequency_55=frequency_icr1_array_55[0];
  Serial.print("Optimal Current for frequency = ");
  Serial.println(optimal_frequency_current_55,2);
  Serial.print("Optimal Frequency = ");
  Serial.println(16000000/optimal_frequency_55);
}

void dutycycle_search_55(){
  float dutycycle_current_array_55[9];
  float dutycycle_ocr1a_array_55[9];
  float diff[9];
  for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_55;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_55[i]=current;
    dutycycle_ocr1a_array_55[i]=OCR1A;
    diff[i]=dutycycle_current_array_55[i]-mac_55;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_55[0]=dutycycle_ocr1a_array_55[i];
      dutycycle_current_array_55[0]=dutycycle_current_array_55[i];
    }
  }
  optimal_dutycycle_55=dutycycle_ocr1a_array_55[0];
  optimal_dutycycle_current_55=dutycycle_current_array_55[0];
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_55,2);
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_55=((optimal_dutycycle_55/optimal_frequency_55)*100);
  Serial.println(optimal_dutycycle_value_55);
}

void charging_55(){
  while(1){
    ICR1=optimal_frequency_55;
    OCR1A=optimal_dutycycle_55;
    Serial.print("optimal_frequency_55=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_55=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
```

```
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    //batteryParameters();
    if(voltage_read()>3.77){
      break;
    }
    frequency_search_55();
    dutycycle_search_55();
  }
}

void frequency_search_60(){
  float frequency_current_array_60[10];
  float frequency_icr1_array_60[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_60[i]=current;
    frequency_icr1_array_60[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
    if(frequency_current_array_60[0]<frequency_current_array_60[i]){
      frequency_current_array_60[0]=frequency_current_array_60[i];
      frequency_icr1_array_60[0]=frequency_icr1_array_60[i];
    }
  }
  optimal_frequency_current_60=frequency_current_array_60[0];
  optimal_frequency_60=frequency_icr1_array_60[0];
  Serial.print("Optimal Current for frequency = ");
  Serial.println(optimal_frequency_current_60,2);
  Serial.print("Optimal Frequency = ");
  Serial.println(16000000/optimal_frequency_60);
}

void dutycycle_search_60(){
  float dutycycle_current_array_60[9];
  float dutycycle_ocr1a_array_60[9];
  float diff[9];
  for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_60;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_60[i]=current;
    dutycycle_ocr1a_array_60[i]=OCR1A;
    diff[i]=dutycycle_current_array_60[i]-mac_60;
    if(diff[i]<0){
```

```
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_60[0]=dutycycle_ocr1a_array_60[i];
      dutycycle_current_array_60[0]=dutycycle_current_array_60[i];
    }
  }
  optimal_dutycycle_60=dutycycle_ocr1a_array_60[0];
  optimal_dutycycle_current_60=dutycycle_current_array_60[0];
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_60,2);
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_60=((optimal_dutycycle_60/optimal_frequency_60)*100);
  Serial.println(optimal_dutycycle_value_60);
}

void charging_60(){
  while(1){
    ICR1=optimal_frequency_60;
    OCR1A=optimal_dutycycle_60;
    Serial.print("optimal_frequency_60=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_60=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    //batteryParameters();
    if(voltage_read()>3.81){
      break;
    }
    frequency_search_60();
    dutycycle_search_60();
  }
}

void frequency_search_65(){
  float frequency_current_array_65[10];
  float frequency_icr1_array_65[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_65[i]=current;
```

```
    frequency_icr1_array_65[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
   }
   for(int i=1;i<10;i++){
    if(frequency_current_array_65[0]<frequency_current_array_65[i]){
     frequency_current_array_65[0]=frequency_current_array_65[i];
     frequency_icr1_array_65[0]=frequency_icr1_array_65[i];
    }
   }
   optimal_frequency_current_65=frequency_current_array_65[0];
   optimal_frequency_65=frequency_icr1_array_65[0];
   Serial.print("Optimal Current for frequency = ");
   Serial.println(optimal_frequency_current_65,2);
   Serial.print("Optimal Frequency = ");
   Serial.println(16000000/optimal_frequency_65);
}

void dutycycle_search_65(){
   float dutycycle_current_array_65[9];
   float dutycycle_ocr1a_array_65[9];
   float diff[9];
   for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_65;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_65[i]=current;
    dutycycle_ocr1a_array_65[i]=OCR1A;
    diff[i]=dutycycle_current_array_65[i]-mac_65;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
   }
   for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
     dutycycle_ocr1a_array_65[0]=dutycycle_ocr1a_array_65[i];
     dutycycle_current_array_65[0]=dutycycle_current_array_65[i];
    }
   }
   optimal_dutycycle_65=dutycycle_ocr1a_array_65[0];
   optimal_dutycycle_current_65=dutycycle_current_array_65[0];
   Serial.print("Optimal Current for duty cycle = ");
   Serial.println(optimal_dutycycle_current_65,2);
   Serial.print("Optimal Duty Cycle = ");
   optimal_dutycycle_value_65=((optimal_dutycycle_65/optimal_frequency_65)*100);
   Serial.println(optimal_dutycycle_value_65);
}

void charging_65(){
   while(1){
    ICR1=optimal_frequency_65;
```

```
    OCR1A=optimal_dutycycle_65;
    Serial.print("optimal_frequency_65=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_65=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    //batteryParameters();
    if(voltage_read()>3.86){
      break;
    }
    frequency_search_65();
    dutycycle_search_65();
  }
}

void frequency_search_70(){
  float frequency_current_array_70[10];
  float frequency_icr1_array_70[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_70[i]=current;
    frequency_icr1_array_70[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
    if(frequency_current_array_70[0]<frequency_current_array_70[i]){
      frequency_current_array_70[0]=frequency_current_array_70[i];
      frequency_icr1_array_70[0]=frequency_icr1_array_70[i];
    }
  }
  optimal_frequency_current_70=frequency_current_array_70[0];
  optimal_frequency_70=frequency_icr1_array_70[0];
  Serial.print("Optimal Current for frequency = ");
  Serial.println(optimal_frequency_current_70,2);
  Serial.print("Optimal Frequency = ");
  Serial.println(16000000/optimal_frequency_70);
}

void dutycycle_search_70(){
  float dutycycle_current_array_70[9];
  float dutycycle_ocr1a_array_70[9];
  float diff[9];
  for(int i=0;i<9;i++){
```

```
    int j=i+1;
    ICR1=optimal_frequency_70;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_70[i]=current;
    dutycycle_ocr1a_array_70[i]=OCR1A;
    diff[i]=dutycycle_current_array_70[i]-mac_70;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_70[0]=dutycycle_ocr1a_array_70[i];
      dutycycle_current_array_70[0]=dutycycle_current_array_70[i];
    }
  }
  optimal_dutycycle_70=dutycycle_ocr1a_array_70[0];
  optimal_dutycycle_current_70=dutycycle_current_array_70[0];
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_70,2);
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_70=((optimal_dutycycle_70/optimal_frequency_70)*100);
  Serial.println(optimal_dutycycle_value_70);
}

void charging_70(){
  while(1){
    ICR1=optimal_frequency_70;
    OCR1A=optimal_dutycycle_70;
    Serial.print("optimal_frequency_70=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_70=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    //batteryParameters();
    if(voltage_read()>3.9){
      break;
    }
    frequency_search_70();
    dutycycle_search_70();
  }
}

void frequency_search_75(){
```

```
  float frequency_current_array_75[10];
  float frequency_icr1_array_75[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_75[i]=current;
    frequency_icr1_array_75[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
    if(frequency_current_array_75[0]<frequency_current_array_75[i]){
      frequency_current_array_75[0]=frequency_current_array_75[i];
      frequency_icr1_array_75[0]=frequency_icr1_array_75[i];
    }
  }
  optimal_frequency_current_75=frequency_current_array_75[0];
  optimal_frequency_75=frequency_icr1_array_75[0];
  Serial.print("Optimal Current for frequency = ");
  Serial.println(optimal_frequency_current_75,2);
  Serial.print("Optimal Frequency = ");
  Serial.println(16000000/optimal_frequency_75);
}

void dutycycle_search_75(){
  float dutycycle_current_array_75[9];
  float dutycycle_ocr1a_array_75[9];
  float diff[9];
  for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_75;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_75[i]=current;
    dutycycle_ocr1a_array_75[i]=OCR1A;
    diff[i]=dutycycle_current_array_75[i]-mac_75;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_75[0]=dutycycle_ocr1a_array_75[i];
      dutycycle_current_array_75[0]=dutycycle_current_array_75[i];
    }
  }
  optimal_dutycycle_75=dutycycle_ocr1a_array_75[0];
  optimal_dutycycle_current_75=dutycycle_current_array_75[0];
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_75,2);
```

```
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_75=((optimal_dutycycle_75/optimal_frequency_75)*100);
  Serial.println(optimal_dutycycle_value_75);
}

void charging_75(){
  while(1){
    ICR1=optimal_frequency_75;
    OCR1A=optimal_dutycycle_75;
    Serial.print("optimal_frequency_75=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_75=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    //batteryParameters();
    if(voltage_read()>3.95){
      break;
    }
    frequency_search_75();
    dutycycle_search_75();
  }
}

void frequency_search_80(){
  float frequency_current_array_80[10];
  float frequency_icr1_array_80[10];
  for(int i=0;i<10;i++){
    int j=i+1;
    ICR1=16000000/(j*500);
    OCR1A=ICR1/2;
    current=current_read();
    frequency_current_array_80[i]=current;
    frequency_icr1_array_80[i]=ICR1;
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("freq = ");
    Serial.print(j*500);
    Serial.println("hz");
  }
  for(int i=1;i<10;i++){
    if(frequency_current_array_80[0]<frequency_current_array_80[i]){
      frequency_current_array_80[0]=frequency_current_array_80[i];
      frequency_icr1_array_80[0]=frequency_icr1_array_80[i];
    }
  }
  optimal_frequency_current_80=frequency_current_array_80[0];
  optimal_frequency_80=frequency_icr1_array_80[0];
  Serial.print("Optimal Current for frequency = ");
  Serial.println(optimal_frequency_current_80,2);
  Serial.print("Optimal Frequency = ");
```

```
  Serial.println(16000000/optimal_frequency_80);
}

void dutycycle_search_80(){
  float dutycycle_current_array_80[9];
  float dutycycle_ocr1a_array_80[9];
  float diff[9];
  for(int i=0;i<9;i++){
    int j=i+1;
    ICR1=optimal_frequency_80;
    OCR1A=ICR1*(j*0.1);
    current=current_read();
    dutycycle_current_array_80[i]=current;
    dutycycle_ocr1a_array_80[i]=OCR1A;
    diff[i]=dutycycle_current_array_80[i]-mac_80;
    if(diff[i]<0){
    diff[i]=-diff[i];
    }
    Serial.print("Amps = ");
    Serial.println(current,2);
    Serial.print("dc = ");
    Serial.print(j*10);
    Serial.println("%");
  }
  for(int i=1;i<9;i++){
    if(diff[i]<diff[0]){
      dutycycle_ocr1a_array_80[0]=dutycycle_ocr1a_array_80[i];
      dutycycle_current_array_80[0]=dutycycle_current_array_80[i];
    }
  }
  optimal_dutycycle_80=dutycycle_ocr1a_array_80[0];
  optimal_dutycycle_current_80=dutycycle_current_array_80[0];
  Serial.print("Optimal Current for duty cycle = ");
  Serial.println(optimal_dutycycle_current_80,2);
  Serial.print("Optimal Duty Cycle = ");
  optimal_dutycycle_value_80=((optimal_dutycycle_80/optimal_frequency_80)*100);
  Serial.println(optimal_dutycycle_value_80);
}

void charging_80(){
  while(1){
    ICR1=optimal_frequency_80;
    OCR1A=optimal_dutycycle_80;
    Serial.print("optimal_frequency_80=");
    Serial.println(ICR1);
    Serial.print("optimal_DutyCycle_80=");
    Serial.println(OCR1A);
    Serial.println("Charging for 1 minute");
    //delay(60000);
    for(int i=0;i<60;i++){
      current=current_read();
      Serial.print("Charging Current = ");
      Serial.println(current,2);
    }
    chargingTime=chargingTime+1;
    Serial.print("Charge time: ");
    Serial.print(chargingTime);
    Serial.println(" minute(s)");
    //batteryParameters();
    if(voltage_read()>3.9){
```

```
      break;
    }
    frequency_search_80();
    dutycycle_search_80();
  }
}

void loop() {
  soc=voltage_read();
  tempc=temperatureRead();

  /*//maximum accepatable current
  float mac_20=1.25; //2.8-3.5
  float mac_40=4.0; //3.5-3.65
  float mac_50=3.858; //3.65-3.72
  float mac_55=3.539; //3.72-3.77
  float mac_60=2.979; //3.77-3.81
  float mac_65=2.659; //3.81-3.86
  float mac_70=2.319; //3.86-3.9
  float mac_75=1.959; //3.9-3.95
  float mac_80=1.5; //3.95-4*/

  if(soc<=3.50 && soc>=2.50 && tempc<=45){
    frequency_search_20();
    dutycycle_search_20();
    charging_20();
  }
  if(soc<=3.65 && soc>3.50 && tempc<=45){
    frequency_search_40();
    dutycycle_search_40();
    charging_40();
  }
  if(soc<=3.72 && soc>3.65 && tempc<=45){
    frequency_search_50();
    dutycycle_search_50();
    charging_50();
  }
  if(soc<=3.77 && soc>3.72 && tempc<=45){
    frequency_search_55();
    dutycycle_search_55();
    charging_55();
  }
  if(soc<=3.81 && soc>3.77 && tempc<=50){
    frequency_search_60();
    dutycycle_search_60();
    charging_60();
  }
  if(soc<=3.86 && soc>3.81 && tempc<=45){
    frequency_search_65();
    dutycycle_search_65();
    charging_65();
  }
  if(soc<=3.9 && soc>3.86 && tempc<=45){
    frequency_search_70();
    dutycycle_search_70();
    charging_70();
  }
  if(soc<=3.95 && soc>3.9 && tempc<=45){
    frequency_search_75();
    dutycycle_search_75();
```

```
  charging_75();
 }
 if(soc<=4.0 && soc>3.95 && tempc<=45){
  frequency_search_80();
  dutycycle_search_80();
  charging_80();
 }
 if(tempc>45){
  digitalWrite(relayPin, LOW);
  delay (60000);
 }
 if(soc>4.0){
  digitalWrite(relayPin, LOW);
 }
}
```

**Table 1:** data of the charging with delay

| voltage | Charging current | time | Duty cycle | Frequency |
|---|---|---|---|---|
| 2.88 | 4.99 | 1 | 90 | 1500 |
| 3.26 | 5.02 | 2 | 90 | 500 |
| 3.39 | 3.71 | 3 | 89.99 | 3000 |
| 3.47 | 4.15 | 4 | 90 | 1500 |
| 3.49 | 4.29 | 5 | 90 | 500 |
| 3.53 | 4.2 | 6 | 20 | 4500 |
| 3.54 | 4.38 | 7 | 90 | 500 |
| 3.6 | 4.39 | 8 | 90 | 500 |
| 3.63 | 6.19 | 9 | 90 | 500 |
| 3.64 | 4.8 | 10 | 90 | 2000 |
| 3.66 | 4.7 | 11 | 90 | 2500 |
| 3.67 | 4.65 | 12 | 89.98 | 3500 |
| 3.68 | 4.93 | 13 | 90 | 1000 |
| 3.7 | 5.34 | 14 | 90 | 5000 |
| 3.71 | 6.21 | 15 | 10 | 5000 |
| 3.69 | 6.1 | 16 | 89.99 | 4500 |
| 3.76 | 6.03 | 17 | 60 | 5000 |
| 3.79 | 5.04 | 18 | 10 | 5000 |
| 3.82 | 5.66 | 19 | 90 | 5000 |
| 3.88 | 4.96 | 20 | 89.99 | 4500 |
| 3.91 | 5.46 | 21 | 10 | 500 |
| 3.9 | 2.42 | 22 | 90 | 2500 |
| 3.96 | 2.67 | 23 | 89.98 | 3500 |
| 4 | 3.2 | 24 | 80 | 4000 |

**Table 2:** data of the charging without delay

| time | optimal frequency | optimal duty cycle | temperature | Charging current |
|---|---|---|---|---|
| 1 | 500 | 70 | 31.77 | 4.1 |
| 2 | 5000 | 10 | 31.77 | 2.95 |
| 3 | 5000 | 90 | 32.26 | 4.64 |
| 4 | 5000 | 90 | 32.26 | 4.1 |
| 5 | 3500 | 90 | 32.26 | 3.68 |
| 6 | 3000 | 90 | 32.26 | 3.08 |
| 7 | 4500 | 10 | 32.26 | 2.86 |
| 8 | 1500 | 90 | 32.26 | 2.9 |
| 9 | 2000 | 90 | 32.26 | 2.84 |
| 10 | 2000 | 90 | 32.26 | 2.67 |
| 11 | 1500 | 90 | 32.26 | 2.76 |
| 12 | 5000 | 90 | 32.26 | 2.49 |
| 13 | 1500 | 10 | 32.26 | 2.43 |
| 14 | 4500 | 10 | 32.26 | 2.47 |
| 15 | 4500 | 30 | 32.26 | 2.45 |
| 16 | 2000 | 10 | 32.26 | 2.47 |
| 17 | 1500 | 30 | 32.26 | 2.49 |
| 18 | 3000 | 10 | 32.26 | 2.26 |
| 19 | 3000 | 10 | 31.77 | 2.29 |
| 20 | 3000 | 10 | 31.77 | 2.08 |
| 21 | 3000 | 10 | 31.77 | 1.98 |
| 22 | 2500 | 30 | 31.77 | 2 |
| 23 | 2500 | 10 | 31.77 | 2.05 |
| 24 | 3000 | 10 | 31.77 | 1.95 |
| 25 | 2500 | 10 | 31.77 | 1.92 |
| 26 | 5000 | 10 | 31.18 | 1.9 |
| 27 | 3000 | 10 | 31.77 | 1.92 |
| 28 | 2500 | 10 | 31.77 | 1.98 |
| 29 | 5000 | 10 | 31.28 | 1.95 |
| 30 | 5000 | 10 | 31.77 | 1.84 |
| 31 | 2500 | 10 | 31.77 | 1.79 |
| 32 | 5000 | 10 | 31.28 | 1.84 |
| 33 | 4000 | 10 | 31.28 | 1.89 |
| 34 | 4500 | 10 | 31.77 | 1.92 |
| 35 | 4000 | 10 | 31.28 | 1.87 |
| 36 | 2000 | 10 | 31.28 | 1.83 |
| 37 | 5000 | 10 | 31.28 | 1.73 |
| 38 | 1500 | 10 | 31.28 | 1.83 |
| 39 | 2500 | 10 | 31.28 | 1.78 |
| 40 | 5000 | 10 | 31.28 | 1.77 |
| 41 | 4000 | 10 | 31.28 | 1.72 |
| 42 | 5000 | 10 | 31.28 | 1.81 |
| 43 | 5000 | 10 | 31.28 | 1.78 |
| 44 | 1000 | 10 | 31.28 | 1.75 |
| 45 | 1500 | 10 | 31.28 | 1.65 |
| 46 | 1000 | 10 | 31.28 | 1.82 |
| 47 | 4000 | 10 | 31.28 | 1.79 |
| 48 | 2000 | 10 | 31.28 | 1.86 |
| 49 | 4500 | 20 | 31.28 | 1.78 |
| 50 | 500 | 20 | 31.28 | 1.79 |

| | | | | |
|---|---|---|---|---|
| 51 | 5000 | 10 | 30.79 | 1.66 |
| 52 | 3000 | 30 | 31.28 | 1.79 |
| 53 | 3000 | 10 | 31.28 | 1.68 |
| 54 | 4000 | 10 | 31.28 | 1.84 |
| 55 | 5000 | 30 | 31.28 | 1.84 |
| 56 | 5000 | 10 | 31.28 | 1.83 |
| 57 | 5000 | 10 | 31.28 | 1.86 |
| 58 | 5000 | 10 | 31.28 | 1.77 |
| 59 | 3500 | 10 | 30.79 | 1.75 |
| 60 | 4500 | 10 | 31.28 | 1.73 |
| 61 | 4500 | 10 | 31.28 | 1.7 |
| 62 | 4500 | 10 | 31.28 | 1.68 |
| 63 | 1000 | 10 | 31.28 | 1.55 |
| 64 | 3500 | 10 | 31.28 | 1.55 |
| 65 | 1500 | 10 | 31.28 | 1.65 |
| 66 | 5000 | 10 | 31.28 | 1.83 |
| 67 | 5000 | 10 | 31.28 | 1.82 |
| 68 | 4500 | 10 | 31.28 | 1.78 |
| 69 | 3500 | 20 | 31.28 | 1.73 |
| 70 | 5000 | 10 | 31.28 | 1.81 |
| 71 | 4000 | 30 | 31.28 | 1.68 |
| 72 | 5000 | 30 | 31.28 | 1.74 |
| 73 | 5000 | 10 | 31.28 | 1.72 |
| 74 | 3000 | 10 | 30.79 | 1.72 |
| 75 | 4000 | 10 | 31.28 | 1.71 |
| 76 | 5000 | 10 | 30.79 | 1.7 |
| 77 | 4500 | 10 | 30.79 | 1.72 |
| 78 | 1000 | 10 | 31.28 | 1.67 |
| 79 | 5000 | 10 | 30.79 | 1.48 |
| 80 | 5000 | 10 | 30.79 | 1.66 |
| 81 | 4000 | 10 | 30.79 | 1.59 |
| 82 | 1000 | 10 | 30.79 | 1.51 |
| 83 | 5000 | 10 | 30.79 | 1.48 |
| 84 | 4500 | 10 | 30.79 | 1.6 |
| 85 | 4500 | 10 | 30.79 | 1.53 |
| 86 | 5000 | 10 | 30.79 | 1.57 |
| 87 | 5000 | 10 | 30.79 | 1.56 |
| 88 | 4500 | 10 | 30.79 | 1.55 |
| 89 | 5000 | 10 | 30.79 | 1.5 |
| 90 | 4000 | 10 | 30.79 | 1.41 |
| 91 | 5000 | 10 | 30.79 | 1.42 |