REAL TIME MONITORING OF SOLAR BATTERY CHARGING STATION (SBCS)

ABRAR ALVI CHOWDHURY – 12121073

MIRZA KARISHMA PRIYANKA – 12121019

BUSHRA MAHMUD – 12321019

Supervised By

Dr. AKM ABDUL MALEK AZAD

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

FALL – 2015

BRAC UNIVERSITY

DHAKA, BANGLADESH

DECLARATION

We hereby declare that this thesis paper titled "REAL TIME MONITORING OF SOLAR BATTERY CHARGING STATION (SBCS)" is submitted to Department of Electrical and Electronic Engineering of BRAC University in partial fulfillment of the Bachelor of Science in Electrical and Electronic Engineering. This paper, neither in whole nor in part, has ever been formerly submitted for any degree or publication elsewhere.

Dhaka, Date 17th December 2015

……
Signature of the supervisor

Dr. AKM ABDUL MALEK AZAD
PROFESSOR
DEPARTMENT of
ELECTRICAL & ELECTRONIC ENGINEERING
BRAC UNIVERSITY

ABRAR ALVI CHOWDHURY
ID: 12121073

……

MIRZA KARISHMA PRIYANKA

ID: 12121019

……

BUSHRA MAHMUD

ID: 12321019

……

ACKNOWLEDGEMENT

We are grateful to our supervisor DR. AKM ABDUL MALEK AZAD, Professor, Department of Electrical and Electronic Engineering of BRAC University for munificently giving us direction throughout our thesis work and providing us with valuable feedbacks that benefited us in every aspect of our thesis. His guidance regarding implementing the SBCS, analyzing readings and calculations, designing all circuitry systems and improvising logics all through the thesis period has helped us enormously. We are tremendously thankful to him for his support. Furthermore we would like to recognize the contribution of our project Engineer Sheri Jahan Chowdhury as she has been assisting us for all this time with every possible effort, hard work and commitment.

ABSTRACT

This thesis paper provides the process, design and implementation of solar battery charging station with real time monitoring system which can be defined as the unconventional energy source and an alternation of electricity. Today's world is moving towards environment friendly smart solutions. Since the concept of solar energy is very new in Bangladesh and applications for this source of energy is limited, new upgrades are required to use this renewable energy resource efficiently; such a resolution can be solar battery charging station. Moreover, real time monitoring makes it more gratified as it offers software monitoring system; displaying the solar voltage, solar current, the battery voltage, time remaining to charge the batteries and battery condition etc. The software is developed using Microsoft Visual Studio 2013 with the programing language C# to monitor the real time of solar based battery charging station using data acquisition (DAQ) card through which all the analog data can be converted to digital form and display them in three layered GUI of the software. All the information regarding a battery to be charged along with the solar condition can be perceived. Solar energy is generated from sun rays converted to direct current through photovoltaic panels; hence this direct current (DC) is stored in batteries. Therefore, monitoring the charging status of these batteries is essential. Furthermore, the paper demonstrates a design of making manual switching system when there is lack of solar radiation due to climatic change or if any blunder occurs in the panels then instantly switches to diesel generator.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

"The Stone Age did not end for lack of stone, and the Oil Age will end long before the world runs out of oil."

- Sheikh Zaki Yamani, Minister of Oil and Mineral Resources (1962-86), Saudi Arabia

The world today is going forward at an unbelievable pace thanks to our modern science and technology. The rapid development of human civilization has provided us with astonishing achievements and raised our hopes to conquer things that were considered impossible even a few decades ago. Unfortunately, almost every element of our technology directly or indirectly pends on fossil fuel, primarily petroleum [1], an energy source of which we have been yet to find an alternative source for. This has accelerated the speed of consuming fossil fuel into a totally new magnitude, causing the potential culmination of the so-called "Oil age" [2]. Therefore, finding an alternative source of energy has become a crying need. Renewable sources of energy, primarily solar energy has and is increasingly becoming popular worldwide as well as in Bangladesh [10][3]. An astonishing figure of 15 million people of Bangladesh is now directly or indirectly dependent on solar energy, with another 50,000 joining the community every month [3]. Bangladesh have gone past the phase of just getting introduced to solar panels. Now we are looking forward to use solar energy in transportation and other commercial purposes [4]. Solar energy is clean, effective to use and most importantly, free of cost and can be used to provide energy to assist solar powered vehicles like rickshaws and vans [7], solar home systems and so on. Most widely used method of storing solar energy is to use lead acid batteries. Solar Battery Charging Stations (SBCS) are very useful in charging those batteries using the solar energy. The concept of solar battery charging station is becoming popular all over the world due to its ability to provide free energy to off-grid areas [10][5]. Therefore, necessity of a proper system to monitor the solar battery charging station in real-time is beyond description. In our thesis, we have developed an effective system to monitor all the necessary parameters of a solar battery charging station.

1.2 Background and Motivation

Bangladesh is a tropical country which receives a very good amount of sunlight. Annually Bangladesh receives as high as 1700kwh/m^2 everyday [6], and proper usage of this energy is more than enough to cover the daily needs of the country. Proper implementation of SBCS and effective real-time monitoring can provide free energy and reduce the load from our national grid, thus decreasing pressure on fossil fuel, creating a green and pollution-free environment and ensure sustainable future for our country- these were our motivation for working with the project. Moreover, one of the major drawbacks of solar is that it cannot be used 24/7 hence storing solar energy in batteries is required for which we definitely need solar battery charging station (SBCS). Manual checking of each battery condition of a SBCS demands great hassle. For this reason then comes the idea of monitoring batteries in real time. Keeping these things in mind we were motivated to run a project which would provide us all these features accordingly.

1.3 Objective

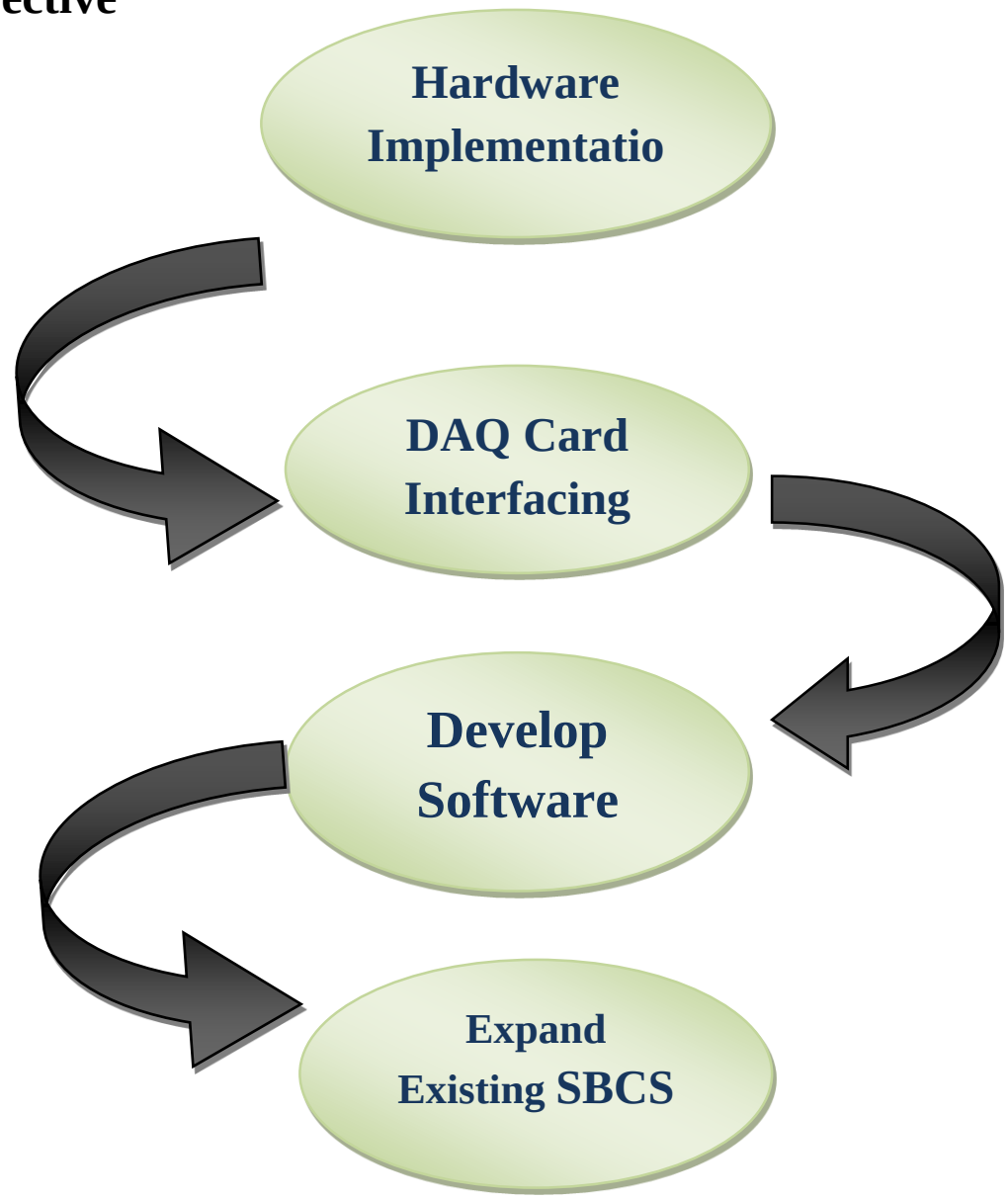

Figure 1.1: Objective of Our Thesis

1.4 Purpose of Our Thesis

Purpose is the main criteria for any project, without it one would find oneself in a maze. As we started our thesis our first target was to fix the goal. We have been attracted to this solar based project so that we can contribute to the society. The solar sector in Bangladesh is growing profoundly. Many solar electric vehicle have been put into work as both proto type and pilot project work, moreover companies are also investing in solar project to become more eco-friendly. Solar vehicle required batteries which has solar energy stored in them. Fossil fuel provides energy for vehicles likewise solar energy powers up these solar electric vehicles. Therefore as much as a gas station or petrol station is important, so as the charging station for solar vehicles. Henceforth we have come up to these purposes for our thesis:

1. Create a real-time monitored SBCS program.
2. Charge multiple battery sets efficiently.
3. Know battery and solar panels' status easily.
4. Identify errors.
5. Use one port to observe multiple battery sets

The following chapters discusses with elaborated information stating how we have done this project. Chapter 02 describes the overview of the project; chapter 03 contains hardware information about solar panel and charge controller, chapter 04 elaborately talks about how we have measured solar voltage and solar current, chapter 05 provides details about our main component battery with specifications and efficiency, chapter 06 illustrates another important component called Data Acquisition Card (DAQ), chapter 07 gives hardware and software interfacing of DAQ card, chapter 08 defines the software we have built for our real time monitoring system of SBCS, chapter 09 describes method of feeding more input in DAQ that is using one channel to charge multiple sets of batteries, chapter 10 provides the manual backup system design in case of low solar radiation, chapter 11 is our future work station how we could improve the efficiency and conclusion of our thesis paper.

CHAPTER 02

OVERVIEW OF THE PROJECT

2.1 Overall Concept of Our Project

The overall project consists of both hardware and software side, therefore implementing the whole project we had to work in both portions. Our thesis comprises the illustrates idea about building a solar battery charging station at the same time provides real time monitoring which demonstrates status of the batteries including status of the solar as well. We connected all the hardware equipment including solar PV modules, charge controller, batteries and DAQ (Data Acquisition Card) to setup our SBCS, followed by took measurements and readings of battery voltages, current, power and subsequently analyzing these results to determine our SOC chart for the batteries which afterwards helped us coding our software using programing language C# in Microsoft Visual Studio 2013. All the other circuitry system required to connect batteries to the DAQ, to measure solar current and solar voltage has been done correspondingly. The block diagram (Figure 2.1) is given below along with the brief description of every step we have performed for this thesis.

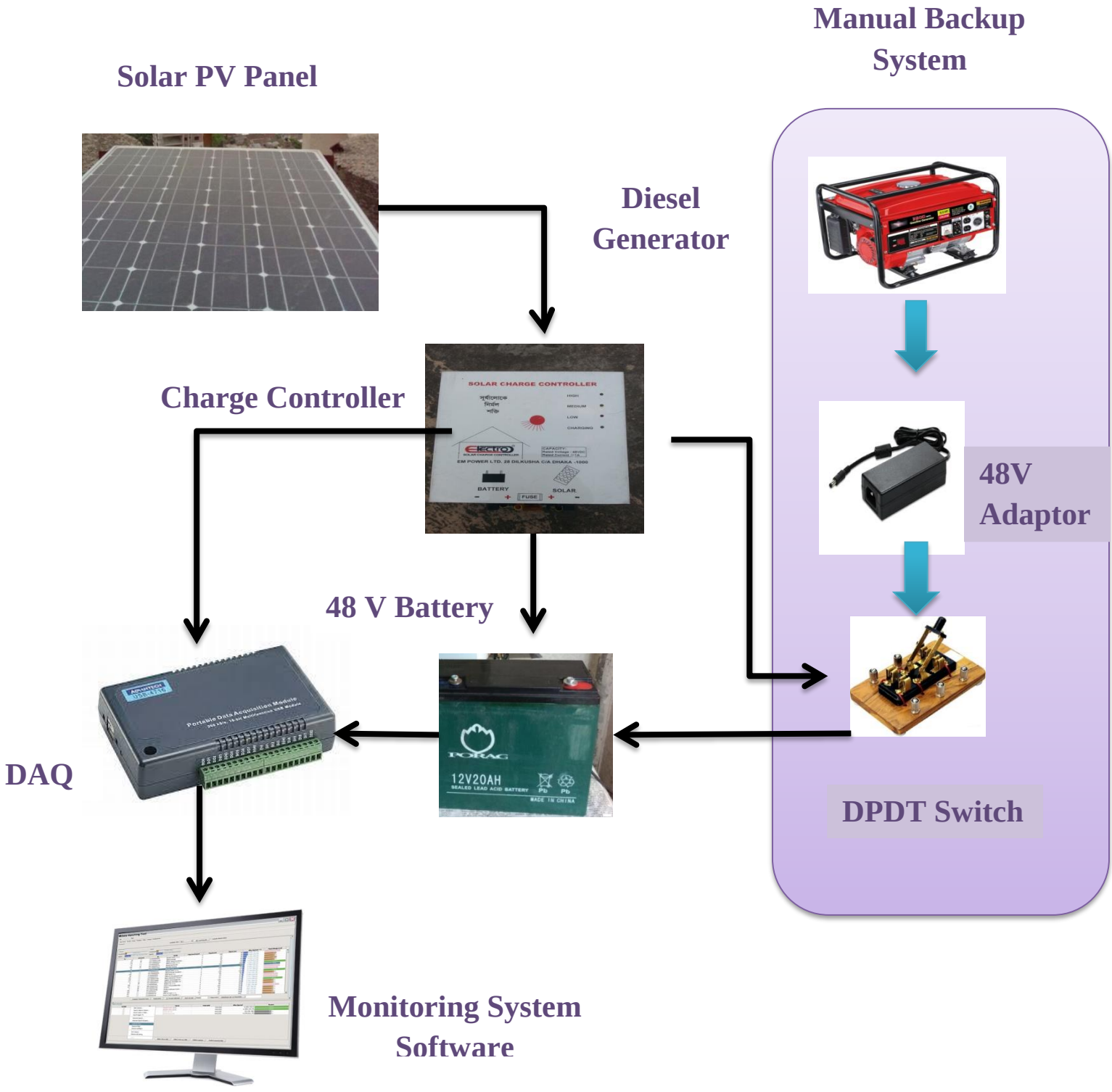

Figure 2.1: Overall Block Diagram of Our Project

2.2 Steps of Our Project:

- Solar PV modules are connected in series-parallel making it a total of 760 Watt-Power arrays and providing 48 V nominal DC voltages for charging our two different sets of 48 V batteries.

- The 12 V 20 Ah sealed lead acid batteries are attached in series to make 48 V 20 Ah; henceforth we have used two sets of 48 V batteries for our project.

- Two separate charge controllers of each 48 V 20 Ah is given connection in parallel with the solar panels in order to charge two sets simultaneously.

- Solar status which demonstrates us solar current and solar voltage individually has been measured using two different circuitry systems, where voltage divider rule provides us solar voltage and using power resistors we have been able to measure our solar current and display the result in our software as well.

- Measuring battery voltage is performed by evaluating voltage while charging the battery sets for several days and by studying those results we have come up with our SOC chart for 48 V 20 Ah battery sets.

- Henceforth those readings have been presented in our software using another voltage divider circuit through DAQ card. The output voltage had to be within a range of 0-15 V, cannot exceed 15 V keeping this in mind our DAQ protection circuit was designed to feed battery voltage as input and the output was fed into DAQ.

- In our software we have used multiplying factor to get the exact value of the battery voltage accordingly. Thus we presented our battery status in our software which is out main target. Each and every battery status in our software includes battery voltage, SOC percentage,

battery level (showing high, low, medium and damaged) and time remaining to get completely charged.

- In accordance with that we have provided a method of charging multiple sets of batteries using MUX since DAQ has limited number of pins, using MUX will help us use the same pin for charging two different sets simultaneously. The software has been programed in such a way that we can switch which battery to charge as the command comes from the software.

- Furthermore our software contains another exquisite feature called "Error Messages" which pops out when there will be changes in our solar panel due to any climatic reason or for when solar radiation is not available. When our solar voltage will be significantly low that will provide us with a message saying "Check Panel Status", and then we check whether there is any error occurred in our panels.

- We have provided a manual backup system since in off-grid areas people will face difficulties to charge batteries when sun is not available. In that condition our manual backup system will come in handy. We have used DPDT (Double Pole Double Throw) switch for it. The diesel generator will produce alternative electricity and adaptor will convert it to DC and that will be fed to charge the batteries through a DPDT switch.

Following all these steps we have been able to conclude this whole micro scale based project to make a complete SBCS with hardware and software implementation with the real time monitoring system. Every connection was done according to the block diagram provided. Here is the circuitry diagram of our overall project in figure 2.2.

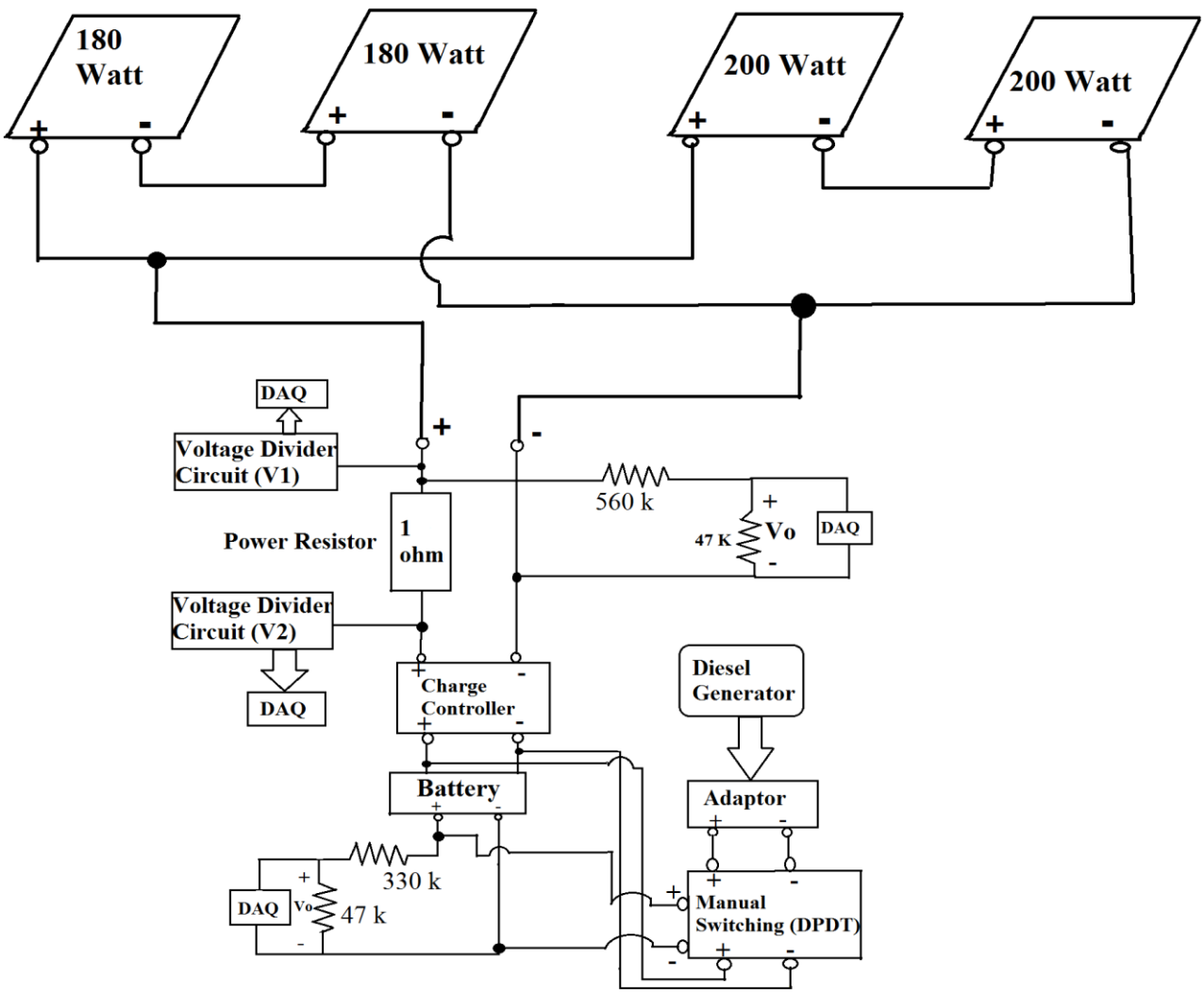

Figure 2.2: Overall Circuit Diagram

CHAPTER 3

HARDWARE COMPONENTS

3.1 Introduction of Hardware Connection

In this chapter we are basically discussing about the hardware equipment and how we have connected them with our software. We are stating details about each and every component we have used to build this project. Since this is our solar battery charging station monitoring these batteries is an essential part for which the DAC card plays an important role. However the DAC card cannot be connected to the actual circuit directly since it is very sensitive and the high voltage from the batteries can damage the DAC. Therefore when are making connections we are ought to be very careful. Moreover we also provided manual switching system for when there will be lack of solar power to charge the batteries; along with the swapping technique for our batteries. Therefore every hardware setup is mentioned in this chapter. For our overall setup for SBCS these apparatus are needed

1. Solar Photovoltaic Panels
2. Two 48 Volt charge controller
3. Two set of 48 V 20 Ah lead acid batteries
4. Advantech USB 4716 portable data acquisition card
5. 1Ω 5 W power resistors
6. Resistors of 560kΩ, 330kΩ and 47kΩ
7. Adaptor
8. DPDT (Double Pole Double Throw) Switch
9. Wire connections (Crocodile wires, bread board, multi-meter, clamp-meter, cables)
10. 2x1 Multiplexer for taking multiple readings through one channel
11. Battery swapping technique using adjustable trolley

3.2 Hardware Components

- 760 Watt Solar Panel
- 2 Set Charge Controller
- 2 Sets Battery,each 48 V 20 Ah
- DAQ Card

In this chapter, solar panel and charge controller will be described from the hardware components. Battery will be elaborated in chapter 5 and Data Acquisition Card (DAQ Card) will be discussed in chapter 6.The whole block diagram shows in Figure 3.1.

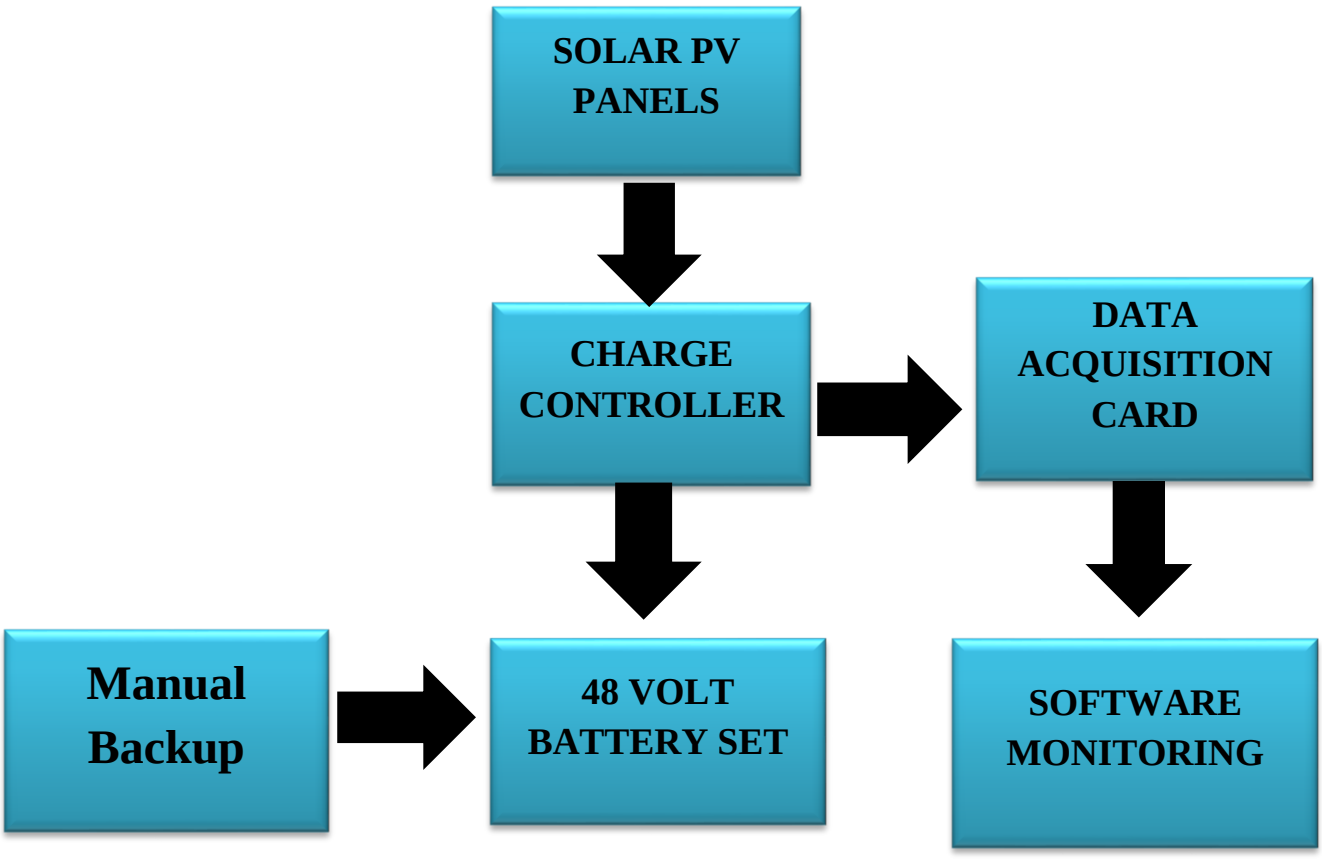

Figure 3.1: Block diagram of SBCS

The figure 3.1 illustrates how we will get our reading in our software. The process is to connect batteries to the panels through charge controller and connect the DAQ card with the battery will provide us with the battery status which will be shown in our software. However we cannot directly go to the software part without paying any heed to the hardware part for this reason hardware installtion and connection performance an essential role in SBCS.

3.3 Solar Photovoltaic Panels

In recent days the people are moving towards sustainable energy, therefore renewable every like solar play important role. Solar photovoltaic panels convert sun's energy into DC (direct current) voltage. Therefore panels are one of the most vital element for any solar based projects. For our thesis we have used monocrystalline panels since the efficiency is greater in monocrystalline panels. In our rooftop we have two 200 W panels and two 180 W panels in series-parallel connection making a total of 760 W. Here are the four panels we have used for our thesis.Figure 3.2 shows the actuall 4 panel placed in BRAC University rooftop.

Figure 3.2: Solar Panels

3.3.1 Features

Maximum Output Power	Pmax	200 watt
Nominal Operating Voltage	Vdc	24 V
Voltage at MPP		34.92 V
Current at MPP		5.70 A
Open Circuit Voltage	Voc	44.64 V
Short Circuit Current	Isc	6.2 A
Maximum System Voltage	DC	1000 V
Cell type	Monocrystalline	
Protection type	Class II	
Cell Temperature	25ºC	

Table 3.3: Specifications of a 200 W panel

3.3.2 Panel Connection

Connecting these four panels is the most important part since we have different watt panels; therefore, according to our requirements we have to connect them. We have connected 200 W panels in series and paralleled them with another 180 W panels which are in series with each other so that we get maximum current and voltage for our battery charging station. The diagram for our connection is given below in Figure 3.4.

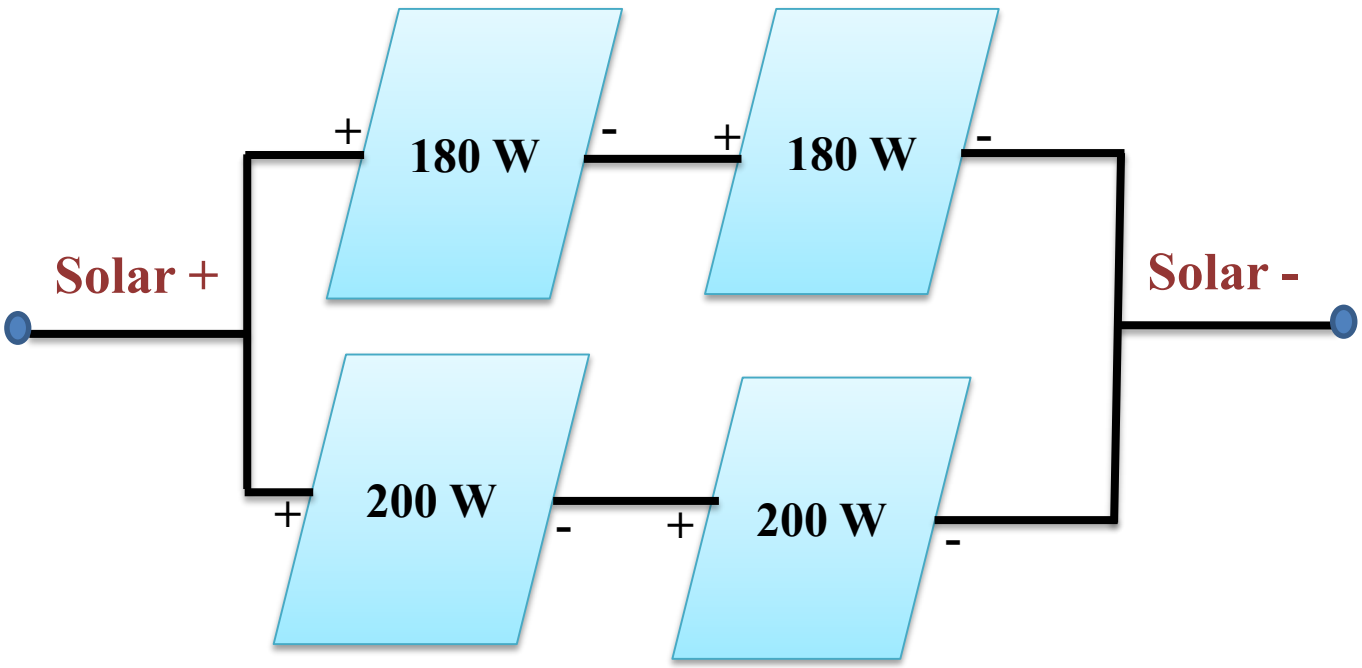

Figure 3.4 : Diagram of connecting solar panels

Our target is to charge two sets of batteries, due to that reason this setup is required. This overall setup will be able to fully charge two 48V 20 Ah batteries within a day. SBCS will provide charged batteries for solar vehicles hence we need our panels to be capable of charging them within a day whereas single PV module will not be able to charge the batteries this fast. Solar vehicles require fully charged batteries for various purpose, and whenever charged of these batteries is requisited SBCS performs a great work. Simultationsly batteries from different firlds will come and would call for charging. In the case our SBCS design has to be in such a way that the panels connected will provide ample amount of voltage to charge these batteries fast. Therefore, series-parallel connection is an effective for charging batteries within a day. There are two arrays of PV modules one of 360 Wp and another one is 400 Wp. These are connected in parallel delivers 48 V DC which is later on used for charging of two sets of 48 V 20 Ah batteries from 50% of their SOC through two 48 V 20 Ah charge controllers.

3.4 Charge Controller

A charge controller is a device that prevent the batteries from overcharging. It is one of the most important components for any solar based project. The high solar voltage coming from the panels will damage the batteries if they are connected directly; that is why charge controller is placed in between them and it can regulate the voltage coming from the panels that goes into the bank of the batteries for charging. Since we are charging two set of 48 V batteries, we require 48 V charge controller. Here is a picture of the charge controller we have used in our project. Figure 3.5 shows 2 charge controller we have used for our project.

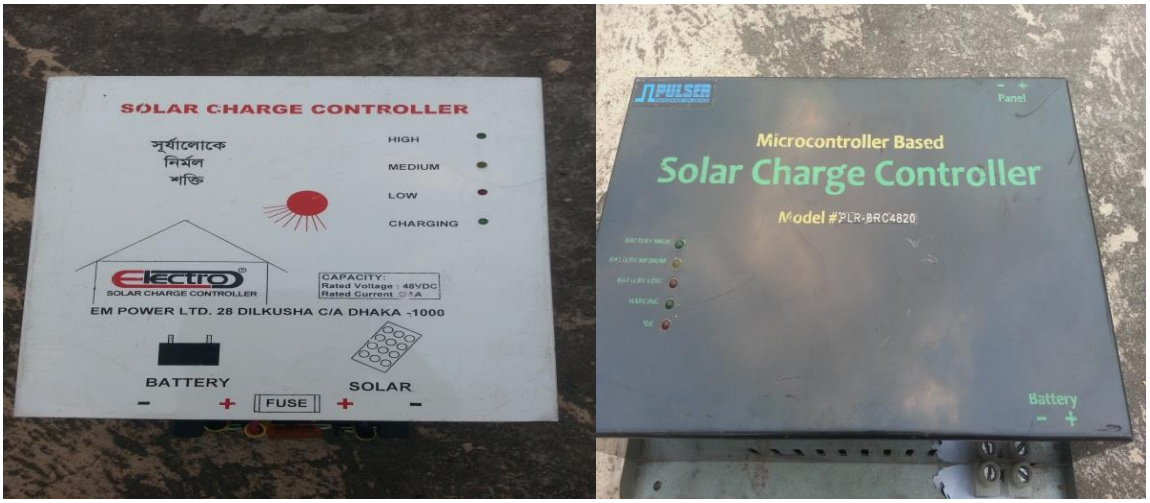

Figure 3.5: Our Solar Charge Controller 48 V Each

3.4.1 Features

1. Nominal output voltage is 48 volt
2. Solar charging current is 0 ~ 30 amps
3. Overvoltage and under voltage protection
4. LED indicates connected or disconnect and low, high and medium

3.4.2 Setup and Connection

The hardware setup was built on our BRAC University building rooftop where we had our four panels connected in series-parallel connection and since this is our micro-scale based SBCS project we had two sets of fully functional batteries. Every setup was done on the rooftop. The wires came from panels and through charge controller we connected them with our batteries for charging. The panels were connected in series-parallel connection as per described in the connection section after that the two wires coming from panels are labeled as S+ (Solar positive) and S- (Solar negative); they are fed into the charge controller's PV+ and PV- port. Similarly, S+ and S- is fed into another charge controller's ports through parallel connection. Parallel connection is important since we are trying to charge two different sets of batteries at the same time. The charge controllers have PV+ and PV- ports and on another side there are two Battery+ and Battery- ports. We connected the batteries on battery+ and battery- ports. This is how we have made our hardware connections for our SBCS. Here is a block diagram (in Figure 3.6) below showing how we have made our parallel connections for two sets of batteries using two charge controllers.

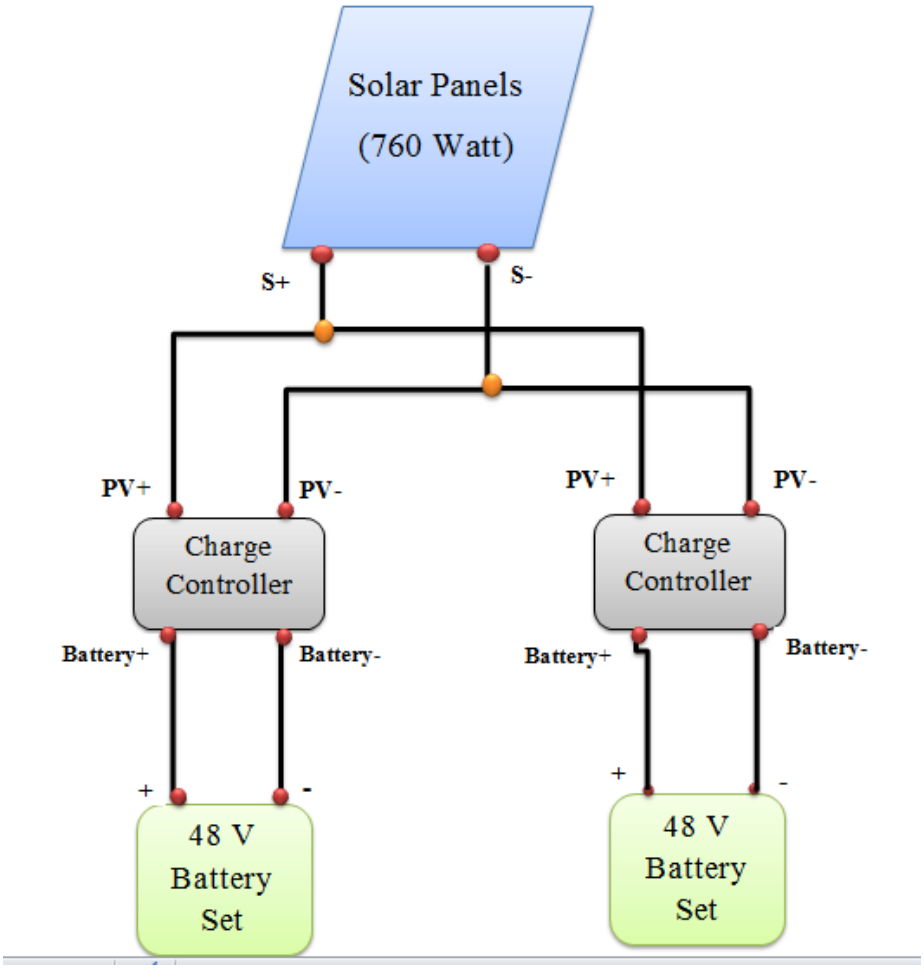

Figure 3.6: Functional Block Diagram For Parallel Connections of SBCS

3.5 Efficiency Calculations for Solar Panel and Charge Controller

Efficiency indicates a process of evaluating whether we get our desired output or how the output behaves comparing with the input. It's a process of getting the maximum usable output at the lowest input. The more the efficiency is, the better. We measure efficiency by diving output by

input. The results will demonstration that how effectively our device is performing hence calculating efficiency is a phenomenal effort. Here we have provided our efficiency calculations separately for solar panels and for charge controllers so that we can acknowledge how productive their performance is. We followed the basic formula for efficiency, that is: $\eta = \frac{P_o}{Pin} \times 100\ \%$

We calculted the efficiency of our panels. For claculating the input power for our panels the formula is to multipy 1000W/m² with the area of the panel. The length and width our one one solar photovoltaic module was 1.61 m and 0.81 m respectively. Therefore that gives us the area of one single module is = 1.3041 m². Similarly for our output power we calculated the current coming form the panel on a certain day and that current was I = 3.5 A amd according to these calculations we have measure our efficiency.

Input power, Pin = 1000 W/m² × Area of panel

= 1000 W/m² × 1.3041 m²

∴ Pin = 1304.1 W

Output power, P_o = V × I = 58.5 V × 3.5 A = 204.75 W

Efficiency for solar panel, $\eta_{panel} = \frac{P_o}{Pin} \times 100\ \% = = \frac{204.75\ W}{1304.1\ W} \times 100\ \% = 15.7\ \%$

Similarly for charger controller the output power of the panel will be the input power since it is fed into the charge controller. That is why

P_o,panel = Pin,charge controller = 204.75 W

P_o,charge contoller = V × I = 54.5× 3.5 = 190.75 W

Efficiency for charge contoller, $\eta_{c.c} = \frac{P_o}{Pin} \times 100\ \% = \frac{190.75\ W}{204.75\ W} \times 100\ \% = 93.17\%$

CHAPTER 4

MEASURING SOLAR VOLTAGE AND SOLAR CURRENT

4.1 Overview

To charge batteries, the most important part is solar radiation. If solar condition is not good enough, we cannot get the fully charged batteries as required. Hence, it is essential to monitor how solar radiation is available along with the solar current following through the entire panel. In that case, we have designed our software that includes solar status along with measuring battery voltages. The term solar status, we have used in software basically provides solar voltage and solar current separately.

4.2 Measuring Solar Voltage

The main purpose of measuring solar voltage is to get an idea from software that how much solar radiation is available in nature at that specific time. Since our software is able to monitor every analog data in real time, any change in amount of radiation due to any shed on panel, natural causes or unexpected disconnection of panel can be easily detect through the software instantly.

4.2.1 Hardware Setup for Measuring Solar Voltage

For this project, we have used 760 Watt panel to charge two sets of batteries each consist of 48 volt. We already know that between panel and batteries, a charge controller must be placed to regulate the excess voltage coming from solar panel. Hence, to measure solar voltage, we have connected an analog port of Data Acquisition Card (DAQ card) in between the solar positive wire and charge controller. The figure below (Figure 4.1) shows the process of measuring solar voltage.

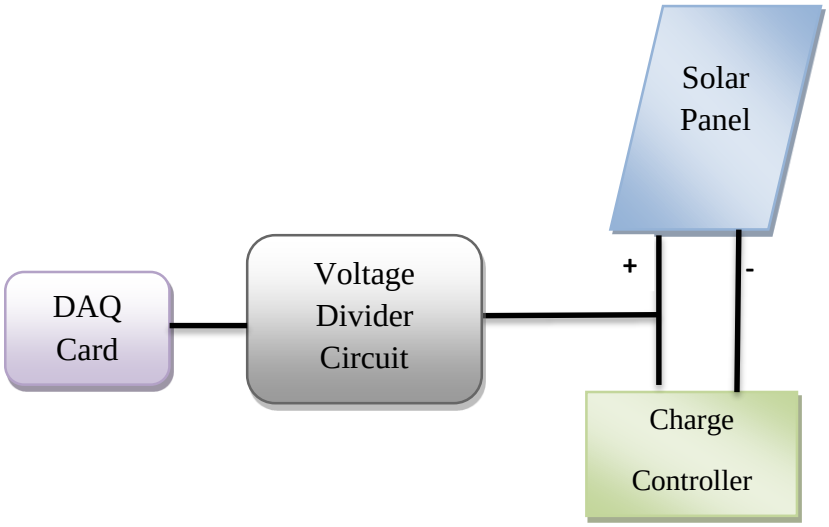

Figure 4.1: Measuring Solar Voltage

4.2.2 Voltage Divider Circuit

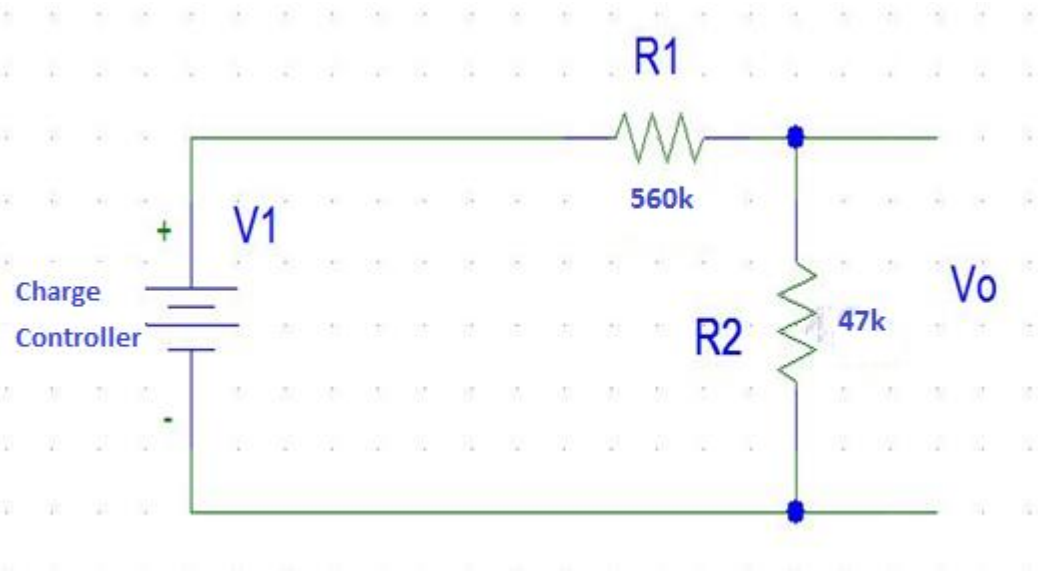

Figure 4.2: Voltage Divider Circuit for Measuring Solar Voltage

DAQ card is a device that converts the analog data into digital form. But it has a problem that the maximum input it can take is 15 volt, otherwise, it will be damaged. Since, normally the solar voltage is always above 50 volt, we have to make a protector circuit so that whatever the input is, DAQ card is always getting below 15 volt. In order to do so we have used this voltage divider circuit to get the low output voltage within 15 V range for DAQ. Figure 4.2 shows the block diagram for measuring solar voltage. However more details about DAQ card is discussed in DAQ card chapter.

4.3 Measuring Solar Current

Measuring solar current is a quite different process rather than measuring solar voltage and measuring battery. In both case of solar voltage and battery, we can connect the positive and negative terminal directly to DAQ card via voltage divider circuit and show them as voltage in the developed software. But for solar current we cannot determine directly how much current following throughout the circuit. However, with the help of 1 ohm 10 Watt power resistor, we are able to calculate the solar current.

4.3.1 Hardware Setup of Measuring Solar Current

According to ohm's law,

$V=I \times R$

Here, V=voltage, I= following Current and R=Resistance

Since we have used 1 ohm power/ceramic resistor, so that from ohm's law we found

$V= I \times 1$ **or V=I**

Therefore, the voltage drop on that power resister is actually the current following in the entire circuit. To measure the voltage drop, we have placed 1 ohm power resistor in between the positive terminal of solar panel and charge controller. After that, two terminal of that resistor is connected

to separate voltage divider circuit. Two different output of the circuit (V1 & V2) is fed to DAQ card to complete further step to measure solar current in Real Time Monitoring SBCS software and to do that, in our software, we have subtract (V1-V2) the inputs of DAQ card. This subtracted value is the solar current since it is the actual voltage drop on the power resistor. Here is the picture for the power resistor we have used for that purpose (Figure 4.3)

Figure 4.3: 10 W 1ohm power resistor

However, the power ratings of the ceramic/power resistor we have used are 10Watt. Therefore, to ensure proper current flow we have used four piece resistor connected in parallel to get exactly 1 ohm for our requirement

4.3.2 Why 1ohm Power Resistor is used

Normal 1 ohm resistor we did not use since solar panel provides high voltage and current. Normal 1 ohm resistor cannot bear it. That is why we have used 1 ohm power resistor. Figure 4.4 shows how we have implemented the circuit for measuring solar current; in addition the figure 4.5 illustrates the overall block diagram for measuring the solar voltage and current together.

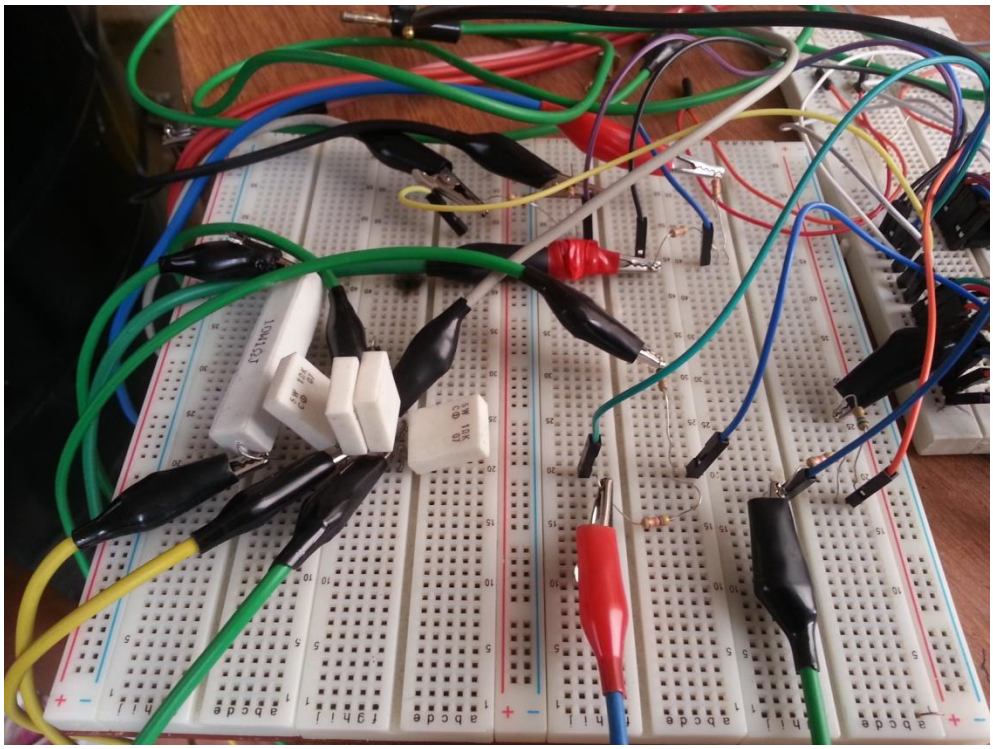

Figure 4.4: Circuit of measuring Solar Voltage and Current

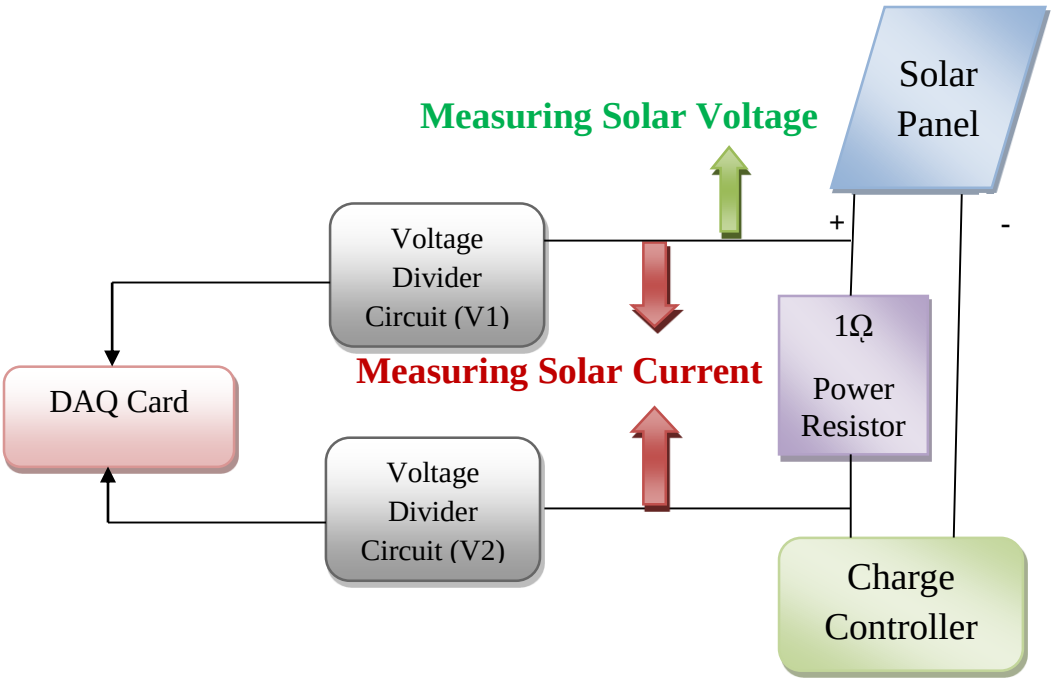

Figure 4.5: Overall Circuit for Measuring Solar Voltage and Solar Current

CHAPTER 5

BATTERY

5.1 Batteries used in SBCS

Our main aim is to provide real time monitoring system for SBCS hence the main functional element here is battery. We charge our batteries in order to use them for other vehicle uses and we monitor that charging status in our software. For any solar based project the first and foremost duty is to store the solar power into a battery so that afterwards it can be used as a main source or as a backup source when ideal solar radiation is not available. For our thesis, we have used 12 V 20 Ah lead acid batteries. We connected four batteries in series to make them a complete set of 48 V 20 Ah. Therefore, we have worked with two fully functional set of batteries, each 48 V 20 Ah. Here in a picture given of the battery we have worked with in Figure 5.1. Subsequently the pictures are provided of both sets of batteries (Figure 5.2) and how we have connected each set in series (Figure 5.3).

Figure 5.1: 12 V 20 Ah Lead Acid Battery

5.2 Reasons to Use Sealed Lead Acid (SLA) Battery

Lead Acid battery is the common solution for commercial application. Though it has some limitation like low energy density and unable to store at discharge condition sealed lead acid battery has some tremendous advantages that helps to what we have required for our work. We have worked with 12 V 20Ah batteries and for our requirements we have connected them in series to make it a set of 40V 20Ah. There are many types of batteries available in the market such as sealed lead acid, unsealed lead acid, deep cycle etc. However for our thesis we had to be very careful in case of choosing batteries. We are dealing with voltage of the batteries and providing methods of measuring them in real time in that case we chose sealed lead acid (SLA) batteries, hence as the voltage changes in real time from either charging or discharging our software will show that changes in real time accordingly. Here are the reasons for choosing our battery:

- Among all type of batteries, SLA batteries has lowest self-discharge rate.
- Lead acid battery is portable.
- Very less maintenance is required for this battery.
- Its capacity range is 0.2Ah to 30 Ah
- Typical charge time is 8-16 hours.
- It is capable of high discharge rate.
- Inexpensive
- Low self-discharge rate
- Mature technology

Figure 5.2: Two sets of 48 V batteries

Figure 5.3: Connecting them in series

5.3 SOC (State Of Charge)

For charging a battery, we need to know the SOC of a battery. An SOC is the state of charge means that maximum charge a battery can contain within itself. It is measured in percentage (%) and plays an important part in case of charging batteries because we cannot exceed the maximum or the minimum SOC for a certain battery. There are two types of batteries; one is deep cycle and another one is lead acid battery. For a deep cycle battery SOC 20% is the lowest that means practically we can use the battery up to where its SOC is 20%; however for a lead acid battery that is up to 50%, which means we can discharge up to 50% SOC. Theoretically, 0% = empty/damaged battery and 100% = Fully charged.

5.3.1 Methods of Measuring SOC

The two most common methods of measuring SOC:

1) Chemical method
2) Voltage method

The voltage method deals with voltage of the batteries, it converts the voltage of the battery to SOC percentage. According to this method the possible charge of the batter can be determined by only observing the battery voltage as in how long it takes to completely discharge from a successfully charged battery. We have used the voltage method to measure SOC because we are dealing with voltage and we need to measure battery voltage so that we can display them in our software. Another reason for using voltage method is our battery is sealed lead acid; therefore we cannot use the chemical method since that includes measuring pH or specific gravity of batteries' electrolyte. To get these SOC percentages we have charged these batteries for several days and took voltage readings and according to that made our SOC chart.

5.4 Setup and Connection for Charging Batteries

The connections of panels and charge controller are done as we have discussed in the previous chapter, then come the connections for batteries. At first the batteries we connected in series and made two different set of 48 V 20 Ah batteries, labeled them Set-01 and Set-02. The two wires coming from panels S+ and S- are connected the charge controller's PV+ and PV- port and through parallel networks both sets of batteries will be competent for charging. As a consequence we connected the battery set-01 with one charge controller and battery set-02 with another thus made our hardware system for charging the batteries. After this we measured voltage by charging these batteries for several days using multi-meter. Subsequently we evaluated our SOC chart for 48V 20 Ah batteries according to our voltage reading taken on each day. Figure 5.4 demonstrates the picture of how we have charged our batteries.

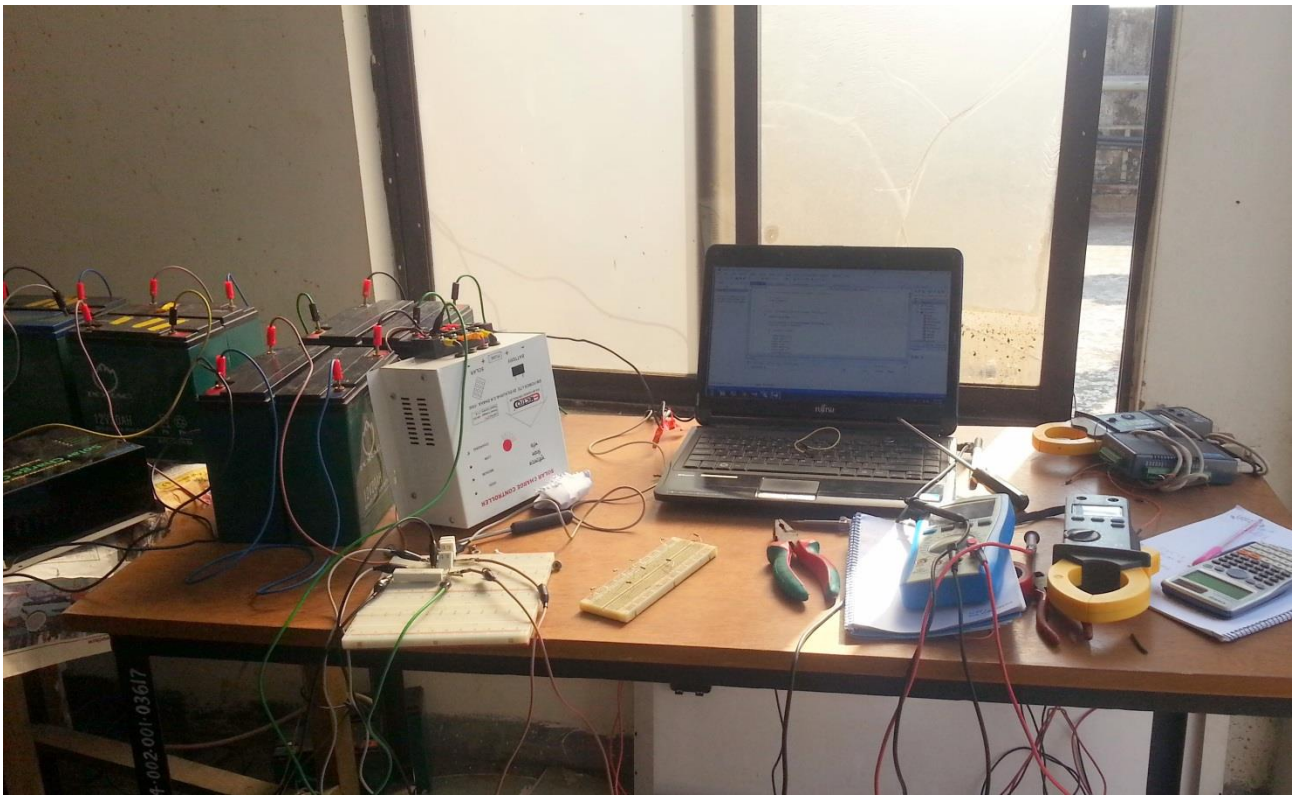

Figure 5.4: Charging batteries in BRAC University rooftop compartment

5.5 Measuring Voltage and Analysis

The voltage reading we have measured were noted down for many days. Each day we started at the peak hour of the day when the sun's radiation is at its highest in order to get the maximum efficiency for charging and also the voltage and current would be high as well. We took reading every 30 minutes for example we started at 12:00 PM and continued till late afternoon and every 30 minutes we measured the voltage in multi-meter and noted down. Therefore, we were able to come up with our chart and charging graph which shows our charging time and voltage respectively. However for our software we have used a reference SOC chart for coding purpose which is quite similar to our one. Table 5.5 shows our voltage reading chart and corresponding graph in given in Figure 5.6.

SET-1

Time	Solar Voltage	Solar Current	Battery Voltage
12:00 (initial)	68.9	2	48
12:30	66.1	2.4	49.7
1:05	66.9	3.8	50.2
1:35	67.1	1.9	50.5
2:07	66.2	1.7	50.6
2:36	67.3	2.1	50.9
3:06	67.2	2.1	50.9

Figure 5.5: Table for measuring voltages

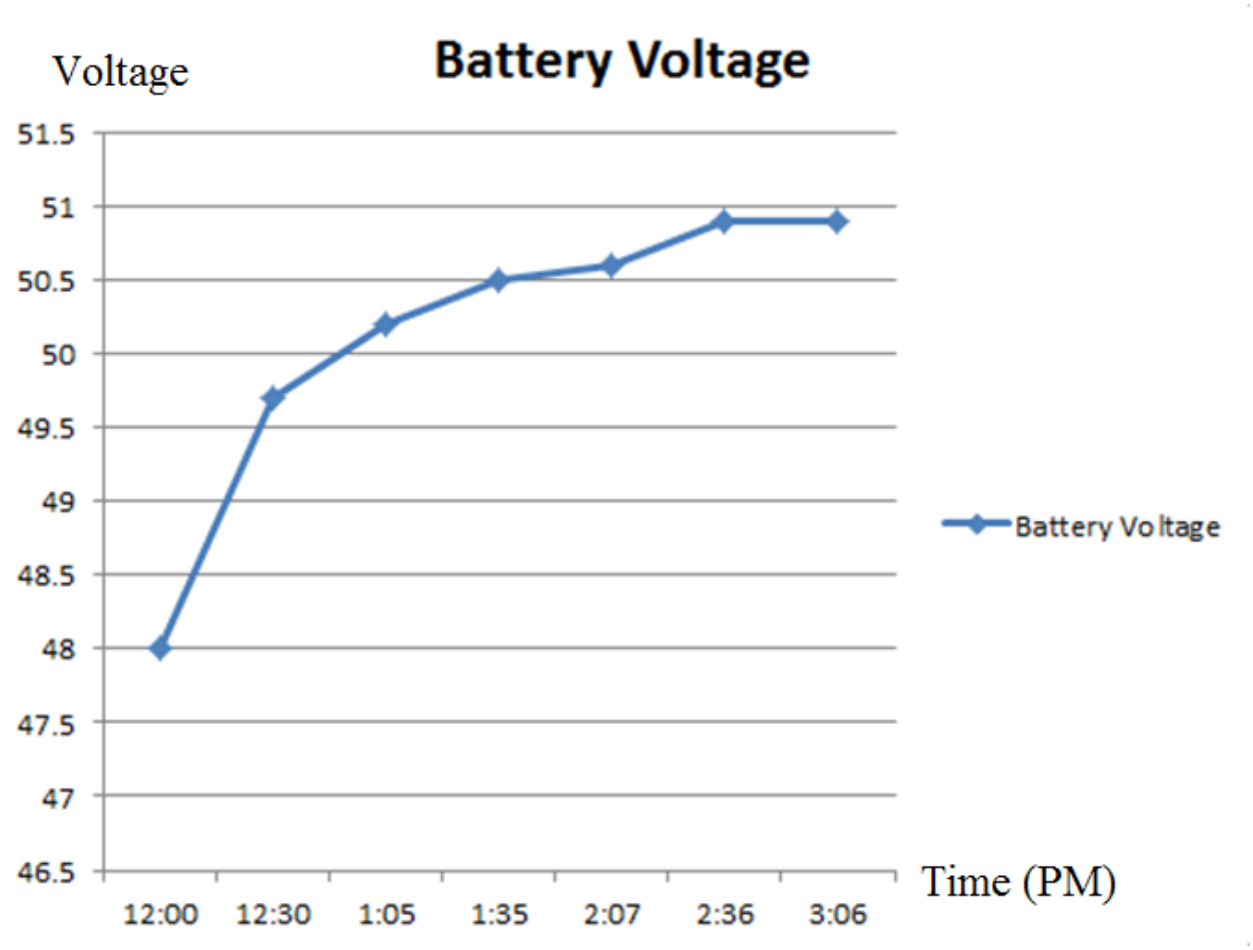

Figure 5.6: Charging Graph (V vs. t)

This charging graph illustrates gradually increasing of the voltage with respective time. As we have deliberated before that we started at the peak hour which is at 12:00 PM and continued till late afternoon. The x-axis expresses the time and the y-axis expressed the battery voltages.

5.6 Comparison with the Reference SOC Chart

For coding purpose in our software we have applied a reference SOC chart of 48V 20 Ah battery which is akin to our charging table. The 100% SOC and 50% SOC of both the charts expresses similar voltage at that time with a negligible deviation. It is taken form Solar Electricity article from homepower.com website. Here is the reference chart given along with the deviation calculations that we have prepared to depict that it matches with our chart. Table 5.7 is the reference SOC chart and the comparison we have calculation is shown in table 5.8.

Charge	12V Battery	48V Battery
100%	**12.73**	**50.92**
90%	**12.62**	**50.48**
80%	**12.50**	**50**
70%	**12.37**	**49.48**
60%	**12.24**	**48.96**
50%	**12.10**	**48.40**
40%	**11.96**	**47.84**
30%	**11.81**	**47.24**
20%	**11.66**	**46.64**
10%	**11.50**	**46.04**

REFERENCE

http://www.homepower.com/articles/solar-electricity/equipment-products/managing-your-batteries/page/0/2 Retrieved 10/06/2015

Table 5.7: Reference SOC chart

SOC	Battery Voltage	Reference	Deviation
100%	50.9	50.92	0.03%
90%	50.6	50.48	0.23%
80%	50.5	50	0.99%
70%	50.2	49.48	1.43%
60%	49.7	48.96	1.49%
50%	48	48.4	0.82%

Table 5.8: Table for deviation calculations

Therefore, the average deviation = 0.83% which is quite negligible hence this table is used for software to demonstrate battery status in our real time monitoring system. As each battery is changing its voltage that can be monitored in our software. The overall hardware setup using panels, charge controller and batteries are prepared considering all these requirements.

5.7 Analyzing Battery Efficiency

In the previous chapter we have shown the efficiency calculations of solar panels and charge controller. Here we will discuss about battery efficiency. To measure battery's efficiency we had to know the discharge time and readings according to that. In order to do so we have connected a load with the battery so that it gets fully discharged. We took readings every five minutes to measure voltage and current. Here is a table given of our discharge voltage and current (Table 5.9). The initial voltage was 51.4 V and current was 10.7 A.

Serial No.	Discharging Time (Min)	Voltage (V)	Current (A)
1.	5 min	51	10.7
2.	10 min	50.6	10.4
3.	15 min	49.4	10.2
4.	20 min	48.5	10.2
5.	25 min	47.8	10.2
6.	30 min	47.5	10.1
7.	35 min	47.2	10.1
8.	40 min	46.6	10.1

Table 5.9: Discharge Rate of Our Battery

Therefore, the output power of battery,

P_o = V × I (Discharge voltage and current, when battery is discharged till 50% of SOC)

= 46.6 V ×10.1 A = 470.66 W

Input power of the batter, P_{in} = V × I (Initial voltage and current when battery was fully charged)

= 51.4 V × 10.7 A = 549.98 W

Efficiency of battery, $\eta = \frac{P_o}{Pin} \times 100\% = = \frac{470.66\ W}{549.98\ W} \times 100\% = 85.58\ \%$

CHAPTER 6

DATA ACQUISITION CARD

(DAQ CARD)

6.1 Introduction to DAQ Card

From the term data acquisition, we understand a process through which real world signals or data are sampled and then converted into digital numeric value which can be manipulated by a computer. A data acquisition card or DAQ card is the device that used for this purpose. We have used Advantech USB-4716 which has 16 channels that are capable of dealing with 16 different signals simultaneously. Figure 6.1 is he picture of Advantech USB-4716 we have used.

Figure 6.1: Advantech USB 4716

As our main aim is to build software that can monitor solar based battery charging station, we work with a PC-based DAQ card to take the input signal as battery voltages. This DAQ system has some internal mechanism through which it can interface only with a specific processing system of a computer [8] [9]. The performance of a DAQ system depends significantly on the data transfer capabilities of a computer. For a real time processing of high frequency signals, we specifically needed a 32-bit processor of windows 7 operating system to satisfy our requirement.

6.2 Advantages of Advantech USB-4716

- Advantech USB-4716 is pretty accurate,
- Reliable
- Error free to record the data
- Plug & play system
- Only USB power is enough, no external power is required

6.3 How DAQ Card Works

Data acquisition card, Advantech USB-4716 follows some specific steps to communicate with the real world data [8]. The block diagram (Figure 6.2) illustrates the steps of DAQ Card.

.

Figure 6.2: Steps of DAQ Card

6.3.1 Elements of DAQ Card

Three basic system elements of a DAQ Card –

- **Transducer or Sensor**

  Transducer or sensor sense the physical phenomena and translate to electrical signal (or vice versa). In our case transducer sense battery voltage, solar voltage and solar current which is considered as variables to be observed in real time.

- **Signal conditioning**

  A signal sensed by transducer is optimized or converted that signal to form depending on the requirement of the DAQ card by amplifying, filtering, multiplexing, and isolation process (for safety of DAQ card). This is the stage where DAQ card make a signal prepared to be converted into digital values.

- **Analog to Digital converter**

  Analog to digital converter is built inside DAQ Card which basically converts the analog input to digital form so that it can be shown in monitor.

  In the converter, there are some control and logic registers to show the digital output on monitor and a comparator to compare the input with DAQ. Previously DAQ sets the logic to zero and starts counting up until it reaches the measured input voltage. After finishing the conversion, the final digital value is stored to register. This logic system is shown in Figure 6.3.

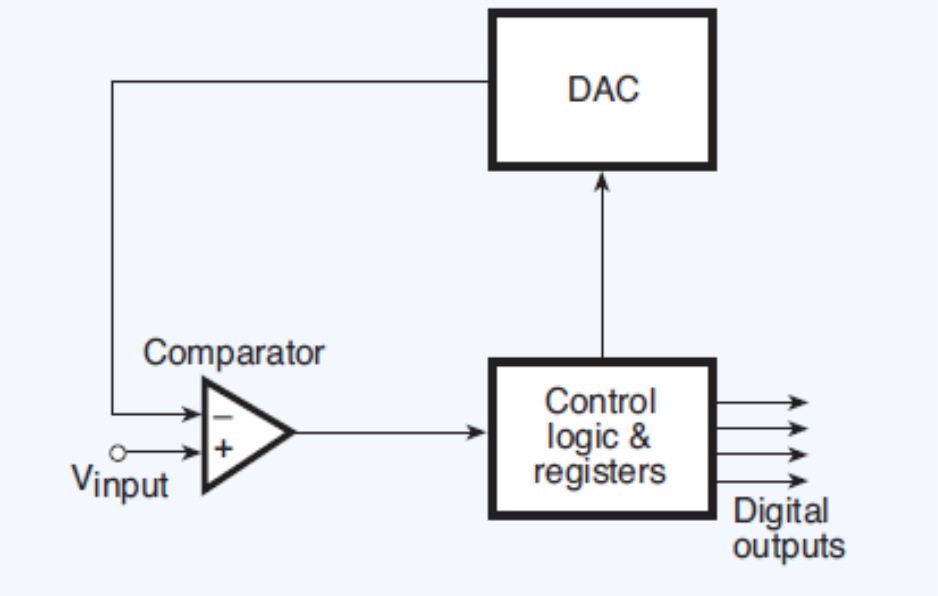

Figure 6.3: Logic of DAQ Card

6.3.2 Pin Configuration of DAQ Card

Advantech USB-4716 has 16 analog input channels from AI0 –AI15, only 2 analog output (AO0 & AO1) along with 8 digital inputs (DI0-DI7) and 8 digital outputs (DO0-DO7) ,shown in Figure 6.4. Every time when any channel is used corresponding AGND for analog and DGND for digital should be used as reference or ground. Apart from the input and output ports USB-4716 has two external trigger port, one external event output channel and one pulse output channel.

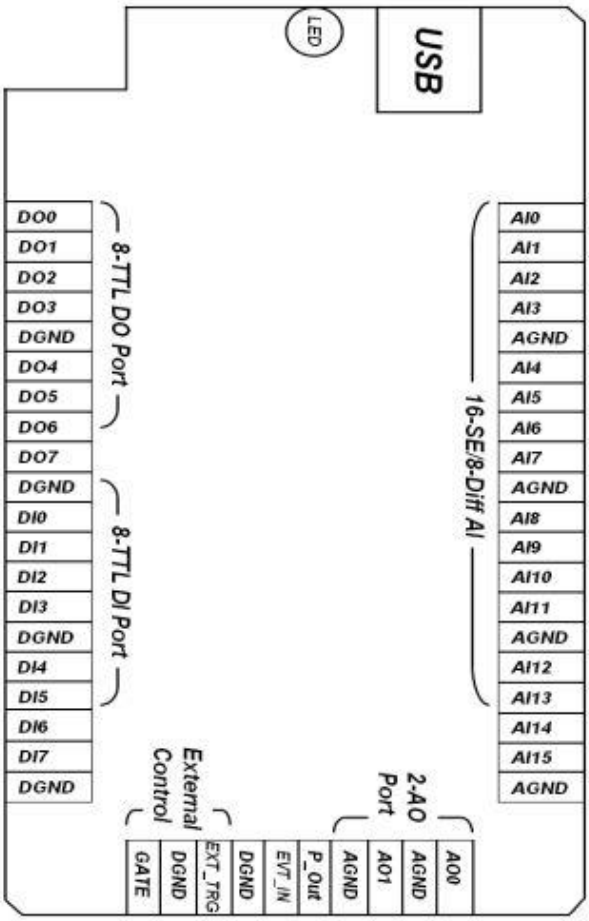

Figure 6.4: Internal Pin Configuration

To install and work with DAQ Card some terms are very important to know-

6.4 DAQ Hardware

DAQ hardware is mainly the interface between signal and a PC. It can be connected directly to slots in the motherboard or can interface to computer external ports like parallel, serial or USB ports. Advantech USB-4716 is specified to interface through USB ports to PC.

6.5 DAQ Software

Data acquisition card has its own software which is delivered with the hardware. To complete DAQ analysis and display in monitor DAQ software is a must. To understand how DAQ software is important, it can be said that DAQ hardware without DAQ software is of no use. On the other hand, DAQ hardware with poor software is almost useless.

6.6 DAQ Device Drivers

Most of the DAQ applications need its device driver software to work with the PC. It can be programmed directly from the register of DAQ hardware and has easy interrogation process from the computer. These drivers are very user friendly, hides low level and complicated details of programming to interface easily and understandable.

6.7 Connection of DAQ Card

DAQ card is connected to charge controller at the port where batteries are connected. Other terminal of the DAQ interfaces with computer through USB 2.0 port. Similarly, for measuring solar voltage and solar current, DAQ card is connected positive terminal of solar panel and other terminal with USB 2.0 port of computer. In both cases, analog input pin (AI0-AI15) is used. This hardware connection will work if DAQ Device Driver is installed in the computer. The software we have developed for Real Time Monitoring of SBCS uses this DAQ driver software to take the readings from real world. However, to install the DAQ Device Driver software for Advantech USB-4716, 32-bit, Windows 7 operating system is required.

6.8 Limitation of DAQ Card

- Cannot take more than 15 volts as inputs
- For industrial purpose, channel number is low

CHAPTER 7

HARDWARE AND SOFTWARE INTERFACING OF DAQ CARD

In our thesis we need to do two types of interfacing, the hardware interfacing and the software interfacing.

7.1 Hardware Interfacing

The hardware interfacing is the process of connecting the hardware properly to the DAQ card so that we could receive the signal from the hardware components. To do this properly, we need to implement a voltage divider circuit for every reading to be taken. As we know, the DAQ card cannot take more than 15 volts, therefore the voltage divider rule is implemented to ensure the safety of the card. In all these cases the voltage which is entered into the card will never exceed 15 volts thus ensuring the safety of the card. We can get the value of the voltages of the batteries, the panels and the solar voltage through this process.

To read battery voltages, individual panel voltages we have used 330k and 47k resistors in series, and took readings across the 47k resistors and fed it to the DAQ card. The 47k resistor takes only a fraction of the original signal but our program multiplies that fraction accordingly so that we can get the accurate value of the signal and display in.Figure 7.1.

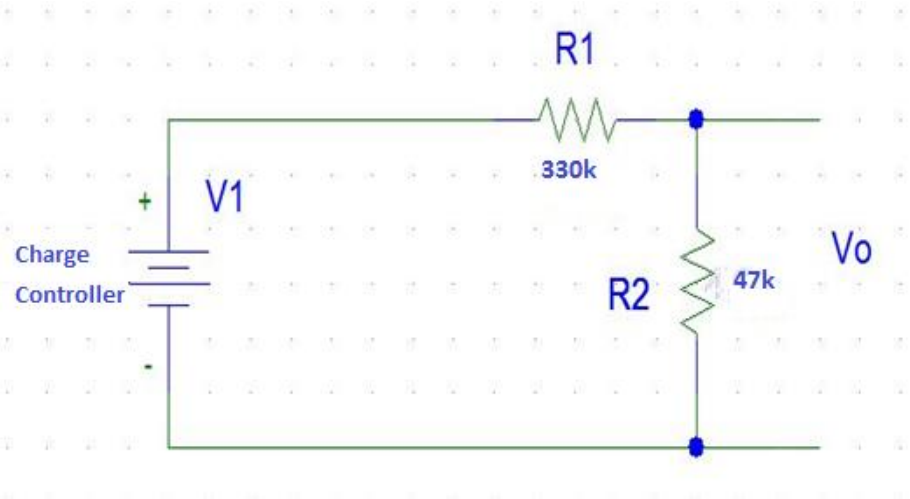

Figure 7.1: Voltage Divider Rule for Battery and Individual Panel Readings

In order to read Solar Panel voltage, we have used 560k and 47k resistors in series to make the voltage divider circuit. The reason behind choosing 560k resistors is to ensure the voltage across 47k resistor does not exceed 15 volts, and using 330k resistors was not enough to ensure it, because the value of the solar panel current was sometimes much higher. The voltage divider circuit diagram is shown in Figure 7.2.

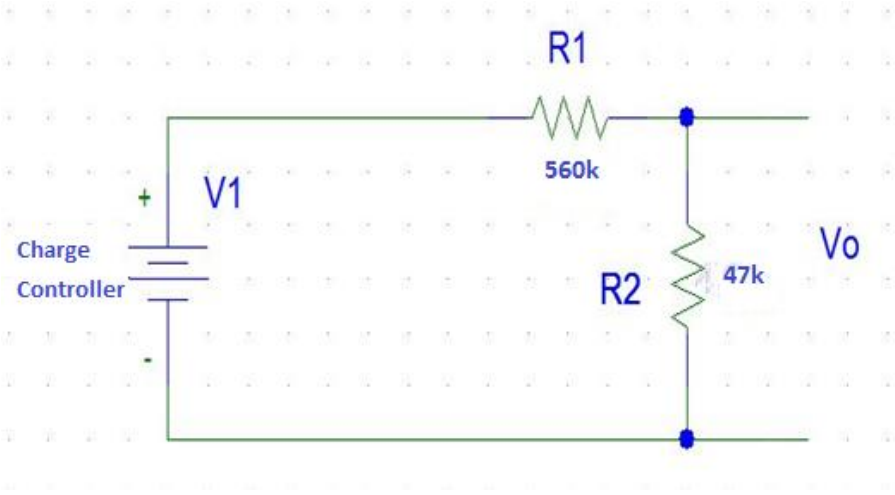

Figure 7.2: Voltage Divider Circuit for Solar Panel Voltage

To measure the solar current we have used a 1 ohm power resistor. The voltage across the resistor is fed through the card, which also goes through a voltage divider circuit into the DAQ, of resistors 560k and 47k. From Ohm's law we know that V=IR, so if I=1, then I=R, therefore the voltage drop across the power resistor is equal to the current flowing through the overall circuit. To determine this, we have used 1 pin for each terminal of the power resistor, and subtracted the higher potential from the lower potential.

The whole process is shown in the block diagram below (in Figure7.3).

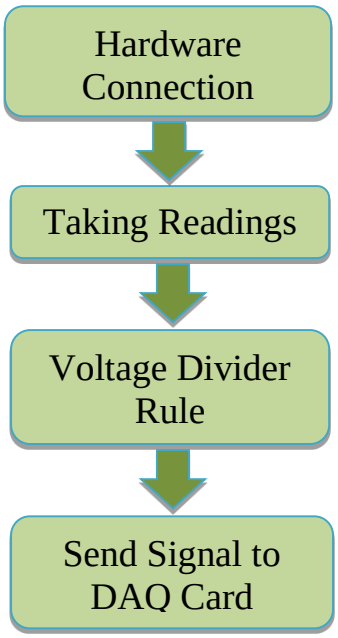

Figure 7.3: Block diagram of Hardware Connection

7.2 Software Interfacing

Software interfacing is needed to show the retrieved signal into the computer. Collecting the signal is not enough if we cannot show it in the computer. It is one of the most important parts in our thesis.

To ensure software interfacing properly, some steps needs to be followed.

- Advantech USB-4716 driver needs to be installed in the computer
- Connect the DAQ card with computer through USB port
- Copy the necessary .dll files from the driver to the system
- Select the correct input or output control functions. Some of these control functions are-
    - DemoAI- Analog Input
    - DemoAO- Analog output
    - DemoDI- Digital Input

- DemoAO- Digital output

- Take readings from the DAQ card by calling some functions from these control functions [8]. Some of these functions are-
  - DemoAI.SelectDevice() – Select correct device
  - DemoAI.ChannelNow()- Select correct analog input pin
  - DemoAO.DataPhysics()- Sending an analog voltage value to a specific analog output pin
  - ADEVEDIT- Advantech ActiveDAQ Pro Number and Editor Control

- Displaying these values according to our necessity in real time in our software.

The properties of functions and .dll files are given below (Figure 7.4).

.dll functions	Sets the device number for opening the specified AI device, or retrieves the device number of the current opened AI device.
Device Name	Retrieves the device name corresponding to the Device Number.
Channel Now	Sets or retrieves the currently selected output AI channel.
Data Analog	Retrieves the sampling data (float) from the current AI channel Now on the DAQ card.
Select Device	Selects a device that supports AI (analog input) functions from the installed Advantech device list in the system.

Figure 7.4: Properties of .dll Files and Functions

Software interfacing is very important because without it, we cannot show the values in real time in our software. Our software does not configure the driver of the DAQ card; rather it uses the driver's .dll functions to read the signals and configures it properly to show it in our software.

CHAPTER 8

SOFTWARE FOR REAL TIME MONITORING OF SBCS

In the real-time monitoring of the Solar Battery Charging Station (SBCS), observing the voltage in not enough, as there is much more to be observed. In our thesis, we have used the device USB-4716, which comes with a driver software as well as an observation software, which can be used to monitor voltages up to a certain limit, which is 10 volts. Moreover, it has some other imitations. To eliminate these limitations, we created our own software which could provide these necessary information for proper observation of the SBCS.

8.1 Reasons for Developing New Software

The reason for creating the new software is to overcome the limitations of the software provided by Advantech, as well as to include more parameters to be observed which are imperative in an SBCS. Although Advantech USB-4716 driver software is able to show voltages up to a certain limit, showing the voltages is not enough for an SBCS. We also need to monitor other attributes, for instance, magnitude of the current, State of Charge (SOC) of batteries, approximate time remaining and so forth. Therefore, it was essential to create a software which could exhibit all these necessary attributes to monitor an SBCS properly. We have used C# programming language in the Microsoft Visual Studio 2013 to develop the software.

8.2 Advantages of Microsoft Visual Studio 2013

There are some certain reasons for selecting Microsoft Visual Studio 2013 and C# programming language. They gave us some certain advantages over other software developing software, as well as other languages. The advantages are mentioned below (Table 8.1).

Advantages	Description
New facilities of Visual Studio 2013	Visual Studio 2013 is the updated version of Visual Studio 2012, and it has a better and easier work environment, increased number supported functions and options, and most importantly, it was the latest version of Visual Studio when we started our project. It also supports .Net Framework 4.5
Advantages of .NET Framework 4.5	Code marking, zip facility, profile optimization and improves startup performance, along with some other improvements from the previous version (.NET Framework 4.0). It also supports 54 languages.
Reasons for using C#	Full COM/Platform support for existing code integration, robustness through garbage collection and type safety, security provided through intrinsic code trust mechanisms, full support of extensible metadata concepts.

Table 8.1: Advantages of Visual Studio 2013 and C# Programming Language

8.3 Drawbacks of Advantech USB-4716 Software

As discussed above, there are certain drawbacks of Advantech USB-4716 software, which is in fact not suitable for SBCS monitoring at all. The limitations of the software are briefly pointed out below.

- The USB-4716 driver software only shows the battery voltages, which is just one of many components which are needed to observe the charging status of the Solar Battery Charging Station (SBCS).
- It cannot show the SOC.
- It can measure only voltage, not the current.
- It cannot sense voltage more than 15 volts.
- It is unable to show approximate time left for the batteries to be fully charged.
- In order to eliminate these restrictions and to show more precise details of the batteries connected to the solar battery charging station, we made our own software using Microsoft Visual Studio 2013 in C# programming language. The features of the newly created software by us are as follows:
- It can detect more than 10 volts. We have used the voltage divider rule to do this, and are measuring and observing charging status of 48 volt batteries, as well as the solar panel voltage.
- We were able to figure out the solar panel's current. We did this using 1 ohm power resistor. We connected it series so that the voltage drop in it will be equal to the current flowing through it, as its resistance is 1 ohm.
- As we were able figure out the solar panel's current, we successfully determined the required time for the batteries to be fully charged.
- We could measure the batteries' SOC, and show it.
- Several warning boxes show if there are any errors.

8.4 The New Software

Now that the necessity of new software is inevitable, we have created the software. There are several Graphical User Interfaces (GUIs) in the software, where each one of them serves a different purpose. The GUIs and their features are briefly discussed in the next section.

- The first Graphical User Interface (GUI), which is our main or primary GUI (Figure 8.1), contains 5 buttons, which are used for selecting the device, showing the batteries, showing the individual panels' status, showing the solar voltage and current and stop showing that,

which will freeze showing the last value it captured. Moreover, it has 2 error message boxes that, in normal condition, does not show any message, but when the solar panels' voltage is lower than battery voltage, depending on how low, shows two error messages, "Manual Backup Recommended" (Figure 8.3), when solar voltage is between 1% and 99% SOC, and shows "Error! Check Panel Status" (Figure 8.2), when panel voltage is significantly low (Lower than 0% of SOC). Furthermore, it also has a select device button. We have created our program in such a way that it can support multiple Data Acquisition (DAQ) cards, therefore, in this GUI, as well as in every other GUI, it is imperative that we select the appropriate device to take data readings.

Figure 8.1: The First GUI

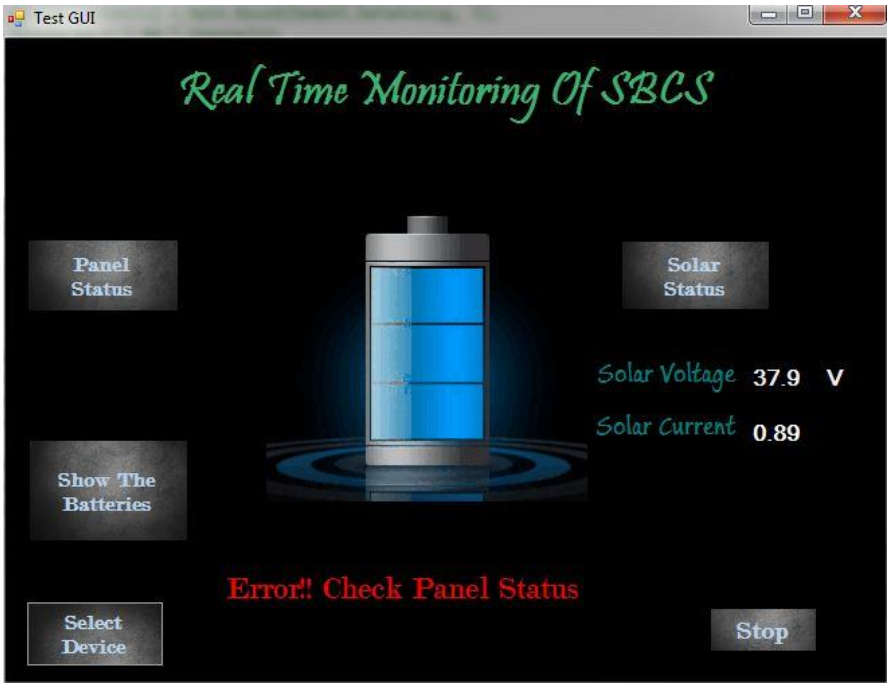

Figure 8.2: Error Message of Panel Error

Figure 8.3: Error Message of Manual Backup

- When we press the “Show The Batteries” button, another GUI opens up, which has one button for every individual battery set, and a select device button, a “Show All Values” button, a “Stop” button, and a “Next >>” button.

  - When the “Show All Values” button is pressed, two text boxes, situated under every individual battery sets’ button, activates and shows SOC and time remaining (Figure 8.4).

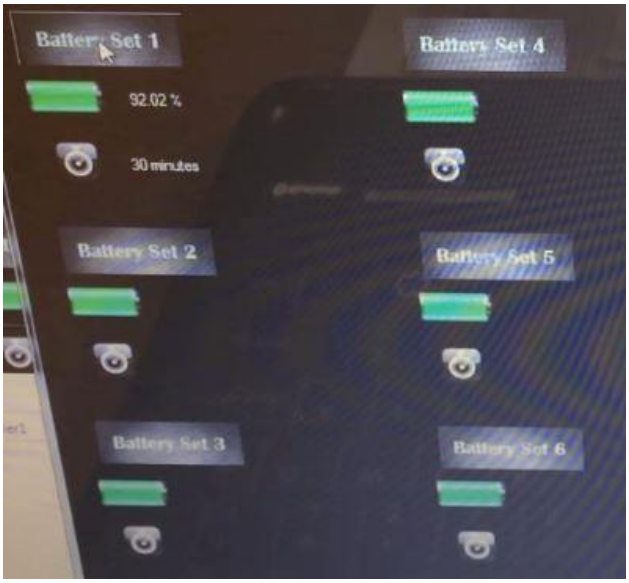

Figure 8.4: SOC and Time Remaining

  - “Stop” button halts the process of reading data

  - The functionality of the “Next>>” button is discussed in section 9.1

- Pressing any battery sets’ button (For example: Battery Set 1) from the above mentioned GUI will result in opening the individual battery sets’ GUI, as shown in figure 8.5, each GUI will display battery voltage, SOC, approximate time remaining to be fully charged and battery charge level (high, medium or low)

Figure 8.5: Individual battery sets' readings

- Again, in the primary GUI, we observe that there is a button called "Panel Status". When this button is pressed, another GUI opens up (Figure 8.7), which shows individual panels' status. It also contains 2 text boxes for each individual panel, one to show the panel voltage and another one to show if there are any errors. As we worked with only 2 panels, we can observe these 2 units of panels separately in this GUI. If any panel is less than 15% than the other panel, the lower panel's error box will show error in that panel (For example, Error in Panel 1). To detect panel errors, we have put shades on a unit of panels (Figure 8.6), and it was successful (Figure 8.7).

Figure 8.6: Panel Shading to Detect Error

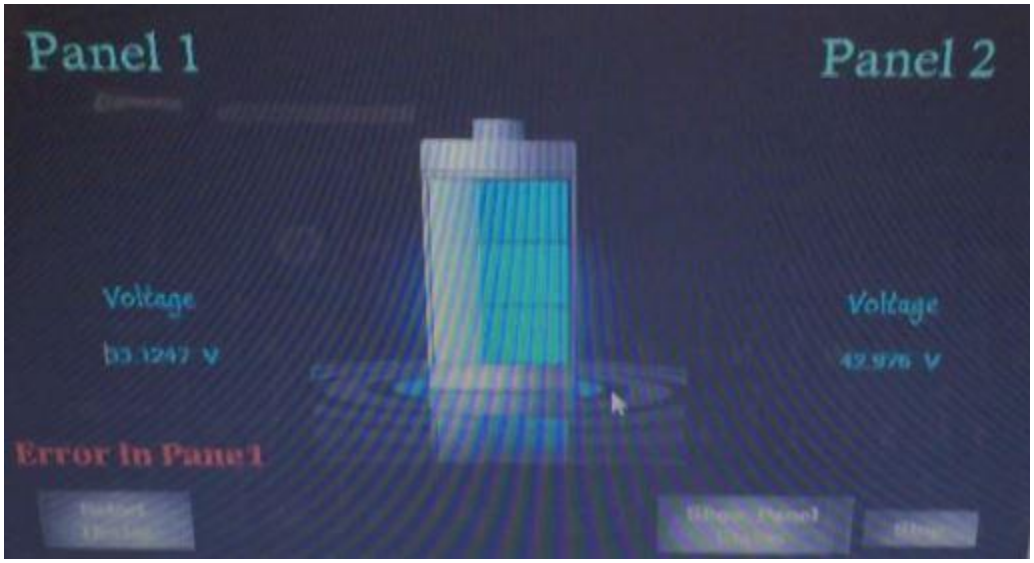

Figure 8.7: Individual Panel Error Detection

8.5 How to Determine Panels' Status Individually

There are 2 units of panels, who can charge one set of batteries individually, and 2 sets when connected in parallel. When they are in parallel that cannot be separated as individual units, but when the parallel connection is open then we can separate them and take individual readings. To take the reading we have used voltage divider rule as we have already used in chapter 7 (Figure 7.1). We use a 330k resistor and a 47 K resistor to build the voltage divider circuit. Voltage across the 47 K resistor is taken and fed into our software which multiplies it accordingly so that we can get individual panels voltages accurately.

8.6 Why New Software

Alongside the device driver software for the Advantech USB-4716, software is given to show the voltage readings in the computer. So, the question may arise that why it is necessary to build new software for real-time monitoring purpose. A comparison of the default software vs. our built software is given below.

8.6.1 Advantech Default Software

The default Advantech software shows only the voltage readings. Moreover, it is unable to display battery SOC, approximate time remaining, battery charge level etc. The default software is not programmable internally so that we can display the voltage reading of 48 volt batteries using the voltage divider rule by not giving input of 10 volts per analog input port. To eliminate these major problems, it was inevitable that a new software was required for the purpose of proper real-time monitoring of the SBCS.

8.6.2 The New Software

The new software that was built for the sole purpose of real-time monitoring of SBCS eliminates all these problems and adds attractive Graphical User Interfaces (GUIs) with real-time animated effects. It shows the voltage of a 48 volt battery without taking input of 48 volts, rather a small fraction of the voltage, and in the programming it is multiplied accordingly to ensure we observe the correct voltage. We also observe the SOC, approximate time remaining, charge level.

Our newly built program supports multiple DAQ cards readings on a single computer. Therefore, it is required, in every GUI, to select the correct device before displaying the values in the GUI.

8.7 Comparison between Existing SBCS Software and Our Software

Software implementations of SBCS are not new, in fact, there has been a thesis already done on Software Implementation of SBCS, where software was created to monitor battery voltages. A detailed comparison between our software and the existing software is discussed in the next section.

8.7.1 Existing SBCS Software

The existing SBCS software shows voltage values of 12 volts batteries. It only shows battery voltages using all analog input ports of the DAQ card. It also shows the SOC, approximate time remaining and battery level. In the voltage reading GUIs, one GUI shows the status of 4 batteries.

Figure 8.8: GUI of Existing SBCS

It is ideal for taking readings of 12 volts batteries only, and highest readable units is 16.

8.7.2 Our New Software

The software that we have developed for our project shows voltages of 48 volt batteries, as commercial solar vehicles use it. Along with the battery voltage, it also informs us about the SOC, approximate time remaining and battery level. Moreover, it also shows us the solar panel voltage and solar panel current. It exhibits individual panels' voltages, and if there are any error in any stage, i.e. damaged battery, low solar radiation, individual panel error etc., there are separate error messages for each problem. We have used, among the 16 analog input pins, 4 pins of DAQ card for solar panel voltage, solar panel current, and 2 pins for 2 individual solar panel units. So, 12 pins are left to measure battery voltages. We have implemented 2-to-1 multiplexers to use one pin to take readings of 2 battery sets, therefore our software is able to read 24 battery sets readings using only 12 pins. Moreover, we have implemented a manual backup system in our project to provide uninterruptable power to the batteries, when required.

CHAPTER 9

METHOD OF FEEDING MORE INPUTS IN DAQ CARD

9.1 Overview

One of the major problems of DAQ card is its limitation of number of channel. It has only 16 analog inputs (AI), 2 analog outputs (AO), 8 digital inputs (DI) and 8 digital outputs (DO) pins.

To build a commercial SBCS, we should have a decent number of battery sets and to analyze and monitor it in real time using our software; we need more available pins in DAQ card.

9.1.1 Why we need Multiplexing in the DAQ inputs

- To monitor more number of batteries at a time
- More than one battery can be monitored using one channel
- To increase the efficiency to DAQ Card
- To minimize the cost of DAQ Card

9.2 Multiple Sets Using One Channel

If we want to build solar battery charging station (SBCS), there will be number of sets of batteries to be charged and monitored in real time through our software. In this process, DAQ card plays an important role since we can monitor number of batteries depending on number of channels of DAQ card are available. The data acquisition card, USB-4716 has 16 analog input channels in which two channels are used for separate panel detection and other two are occupied for solar voltage and solar current measuring in software. Therefore, the remaining 12 channels are used for battery, which is of course not enough for any commercial purpose. However, considering this value to increase the number of battery through one DAQ card, we have come to a solution to use multiple sets of battery in one channel. To do that, we have used a digitally controlled analog multiplexer, CD4053B, 2-to-1 MUX which has very low ON impedance and very low OFF leakage current.

9.2.1 The Multiplexing IC

This IC CD4053B has 16 pin and triple MUX inside it but we have used one2-to-1 MUX as we considered only two sets of batteries. In that case, all unused pin considered as grounded whereas VDD is set as biasing +9V. Two inputs X and Y are our two battery sets that are selected by the selection pin A.B.C. This selection pin is basically controlled from the software according to our command. Initially, battery set 1 to 12 readings are default inputs from DAQ card. Using MUX, battery set 13 to 24 can be shown in another GUI of the software through a selection pin, connected to the analog output of the DAQ card. So, when we want to see the status of battery set 13 from channel AI0, we have to command from the software to analog output pin of the DAQ card, which is already been connected with the selection pin, this pin switches according to the command and transmit the status to battery set 13 to DAQ card. The original picture of the 2x1 MUX IC-CD4053B (Figure 9.1) along with the internal pic configuration of the IC (Figure 9.2) and how we have connected the IC with DAQ card (Figure 9.3) figures are given below.

Figure 9.1: Picture of CD4053B

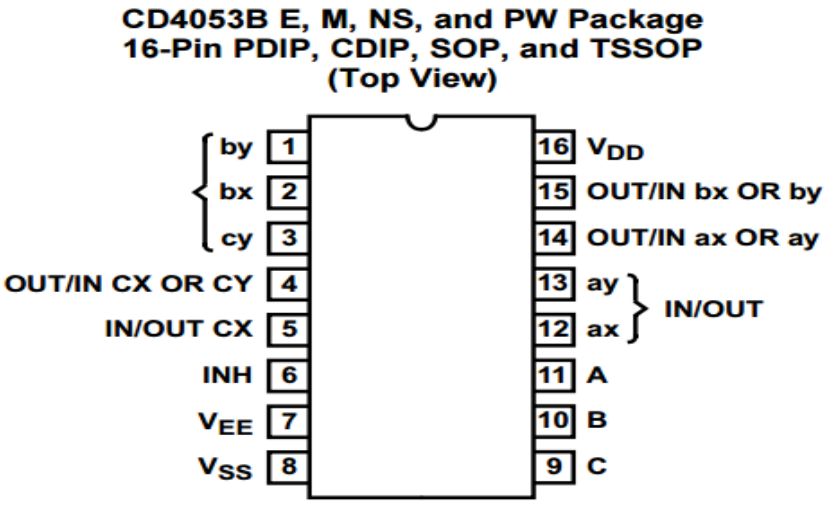

Figure 9.2: Internal Pin Configuration of CD4053B

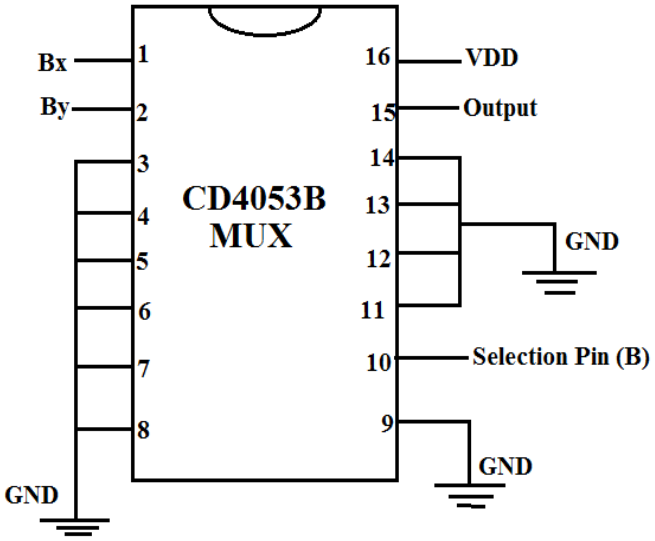

Figure 9.3: How We Have Connected the MUX with DAQ

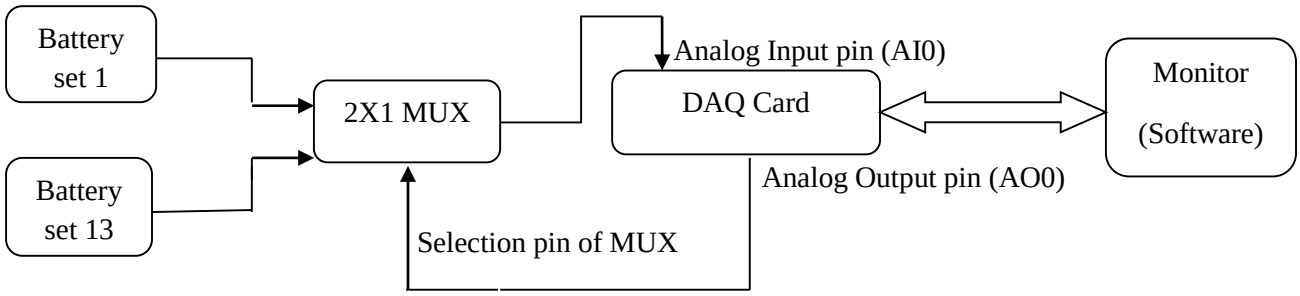

Figure 9.4: Overall Functional Block for Multiplexing

This overall blocked diagram (Figure 9.4) elaborates the process we have used in our project for charging multiple sets. Therefore, with this process using 2x1 MUX, we are able to show 24 sets of battery using 12 channels. However, with this similar process if we use 4x1 or 8x1 MUX we can show the 48 or 96 different sets of batteries respectively at a time. All we have to consider the number of analog pin of DAQ card since they will act as the selection pin. For 2x1 MUX, we need only one selection pin but if the number of inputs increases, the selection pin will also increase accordingly.

9.3 Implementation of MUX in Our Software

In our software's second GUI, mentioned in section, we have stated that there is a "NEXT>>" button. There are 12 available pins for batteries, and we have used 2-to-1 MUX, and we have programmed in such a way that pin number 1 to 12 (AI0 to AI11) will take readings of battery set 1 to 12, and after multiplexing, the same pins will take readings of battery sets 13 to 24. Therefore, analog input pin no. 1 (AI0) will take input of battery sets 1 and 13; pin number 2 will take battery sets 2 and 14 and so forth.

- When the selection pin of the MUX is 0, the software will show battery set 1 in pin no.1. Figure 9.5 shows the block diagram and Figure 9.6 shows the GUI that will pop out.

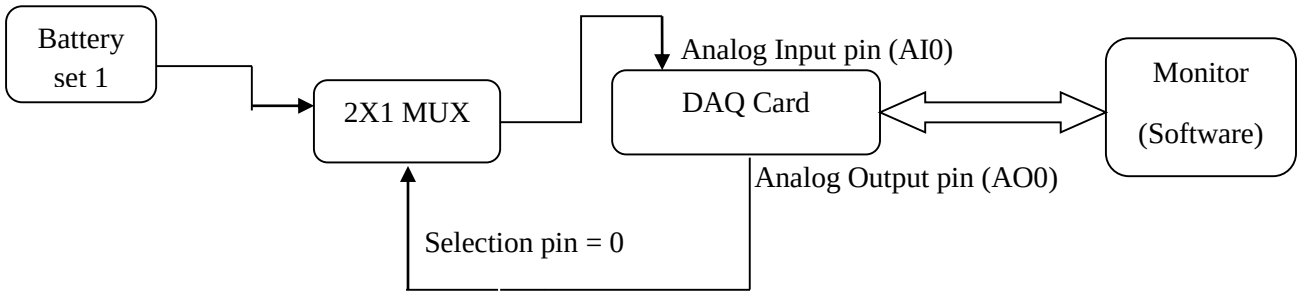

Figure 9.5: Multiplexing Block Diagram (01)

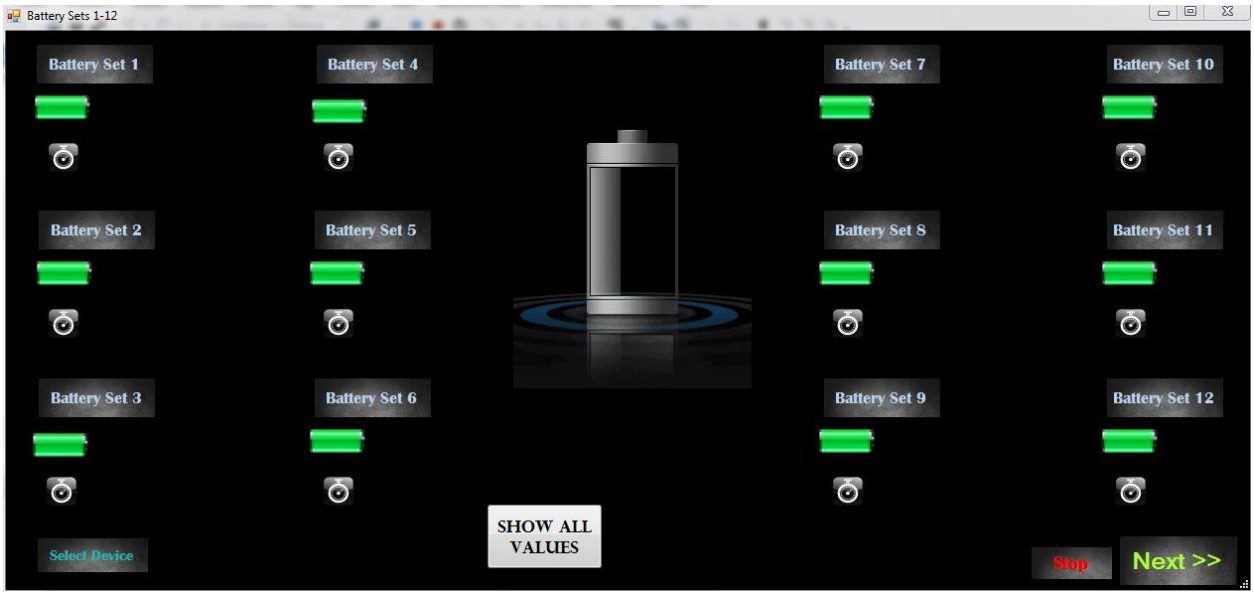

Figure 9.6: Multiplexing GUI (01)

- When the selection pin is selected to become 1, battery set 13 will be shown. Figure 9.7 shows the block diagram and Figure 9.8 shows the second GUI that will pop out as a result of multiplexing.

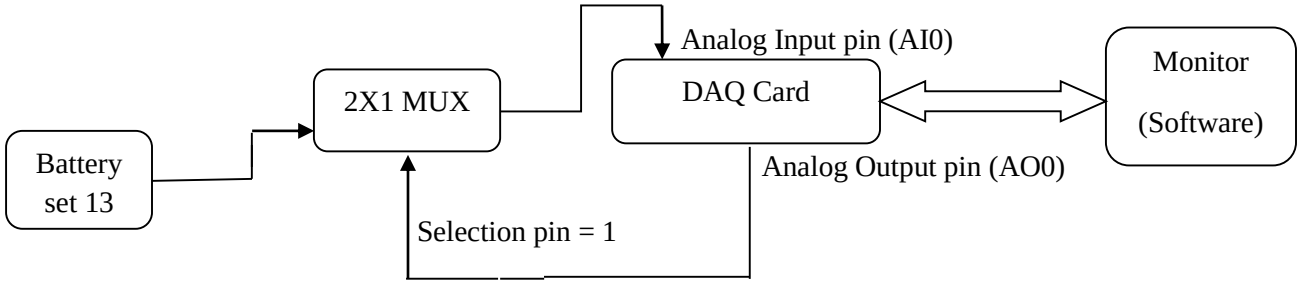

Figure 9.7: Multiplexing Block Diagram (02)

Figure 9.8: Multiplexing GUI (02)

CHAPTER 10

MANUAL BACKUP

10.1 Introduction to Manual Backup System

Solar Energy is clean and emission free and has the greatest availability compared to other energy sources. The solar energy incident on the earth in one day is sufficient to power the total energy needs of the earth for one year. However, due to the low efficiency in the conversion process, we cannot utilize this advantage.

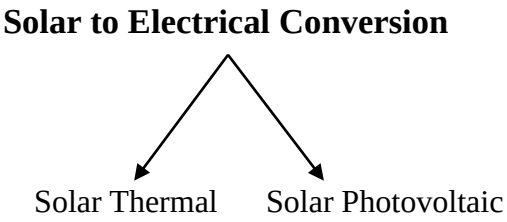

Figure 10.1: Classification of solar-electric conversion

Among the two process of converting solar energy to electrical, our one is solar photovoltaic which converts the light energy, absorbed from incident sunshine, into DC electricity. Therefore,
Solar Battery Charging Station (SBCS) totally depends on solar radiation. Without the radiation of sun, no battery can be charged. So, at night or in any natural situation, where there is no sunlight, we need a backup system to charge those barriers.

10.2 Why Manual Backup Is Required

Bangladesh is indeed a developing and growing country, but in terms providing facility of electricity in every corner of the country is not yet satisfactory. Only 62% of total population has access to electricity. Many places are still out of this facility. Moreover, despite of having national grid, in some places people suffer from heavy load shedding for hours. In that case, batteries of SBCS can provide alternative energy in those areas. As the privilege of using national grid is not always possible due to heavy load shedding, we have to come up with a solution that can be controlled manually when required. Figure 10.2 illustrates the process of manual backup.

10.3 Why Not Automated Backup

We have considered manual backup instead of automated switching as this project is designed for off-grid areas. In that case, we do not need to charge at all times, but when we need in critical or no solar condition we can manually control it and take power from generator. Our software also shows if we do not get appropriate solar power to charge the batteries efficiently.

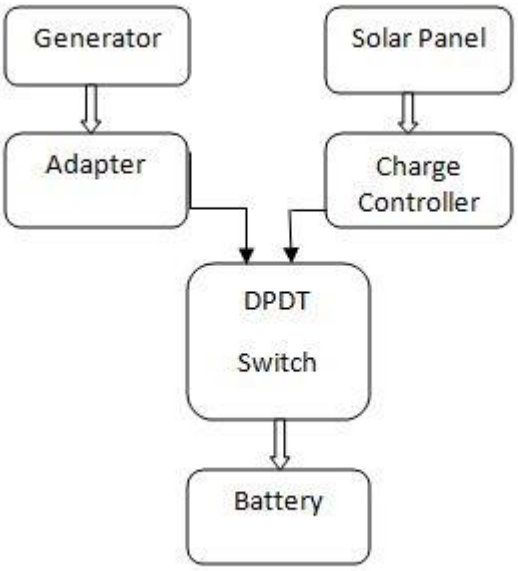

Figure 10.2: Block Diagram for Manual Backup

To provide backup power into our system manually, we have used a double pole double throw (DPDT) which operates such as switch, can be powered according to requirement. A generator of same power, compared to solar panel we are using, can take place as a backup source. A generator generates AC current. However, through an adapter we can make it DC to charge 48V batteries. A DPDT switch is used to determine whether we want to use solar source or generator as a backup.

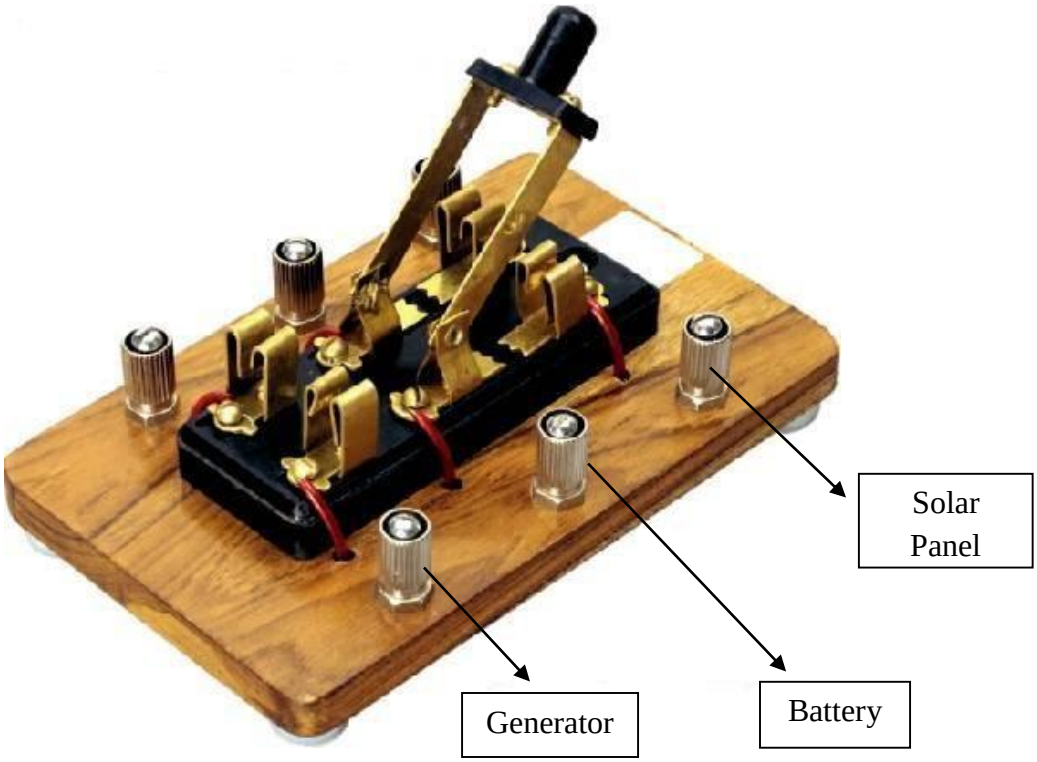

Figure 10.3: Connection of DPDT Switch

Figure 10.3 is the DPDT switch. The positive and negative sides of both adapter and charge controller are connected to the opposite poles and the battery should be connected with the middle pole of the DPDT switch. According to our desire we can change the throw to select the appropriate power source to charge the batteries.

CHAPTER 11

FUTURE WORK

11.1 Overview

In spite of being a developing country, some places of Bangladesh are still under developed in terms of availability of national grid. For those areas, Real Time Monitoring of SBCS is decent solution to meet up their demand of electricity. Through the mass implementation of SBCS, we can provide the required energy that can be full filled that demand for an entire area. Real Time Monitoring SBCS is a complete solution but it is not limited in our project. There are many sectors in which it can be advanced in software to help real world. In terms of hardware connection, existing SBCS which works with only two sets of batteries can be extended to 24 sets of batteries using 2x1 multiplexer. Moreover, different types of sensor and sun tracking system can be installed to increase efficiency.

11.2 Future Works

➢ Mass Implementation

Real Time Monitoring of SBCS can be extended to mass implementation to help the rural area. IDCOL (Infrastructure Development Company Limited) and CARC (Control and Research Centre) are operating a pilot project of 15 solar electric vehicle funded by IDCOL. For this project, we are implementing a charging station which can be monitored in real time of those 15 solar electric vehicles. In order to do so, we have made an estimated budget for 15 sets.

Panel Calculation for 15 sets

Panels:

For one set of batteries – 400 W

∴ For 15 sets = 15 × 400 = 6000 W

Panel per watt = 52 TK

∴ For 15 sets = 6000 × 20 = 3, 12,000 TK for panels

DAQ Card:

Quantity-01

Price – 50,000 TK (Approximately)

Battery:

For 15 sets, batteries required = 15 × 4 = 60 Pieces

Price for one piece of battery = 3000 TK

∴for 15 sets = 3000 × 60 = 1, 80,000 TK

Charge Controller:

One piece = 5000 TK

For 15 sets = 15 × 5000 = 75,000 TK

Therefore, for 15 sets of batteries, total amount = **6, 17,000** TK

➢ Install various sensor

We are working on installing various types of senor in our software to enhance reliability and safety. We are planning a fire alarm sensor and smoke that alarms if any unwanted incident occurs in the charging station.

➢ Sun Tracking System

Sun tracking system is the movement of panel according to the movement of sun. It basically helps to get the maximum light of sun. Hence, if this sun tracking can be installed, we will be beneficial in the sense that our battery will be charged quickly and efficiently.

➢ Increasing Efficiency

Using 4x1 MUX, we can take battery voltage reading of 48 sets which will be more helpful larger implementation of SBCS. Similarly, using 8x1 MUX will further increase the efficiency.

➢ Battery Swapping Technique

The main purpose of our thesis is to provide off-grid solution in rural areas by solar battery charging station where people can come and easily charge their set of batteries and take them back home to use it in daily basis. However the batteries we are using is quite heavy to lift up from home and carry it to the station and also the other way around. In charging station they are kept at a place to charge them through solar panels but when we need to take them for various uses such a for solar van or solar tri-wheeler etc. it may be challenging since sealed lead acid batteries are very heavy. Moreover we connected four 12 V 20 Ah batteries in series to make 48 V 20 Ah battery set; therefore apparently it would require strong muscle power to lift them up. Hence we have used a useful swapping technique for our charging station to move these heavy sets of batteries along effortlessly.

The common way for transferring heavy loads from one place to another is by using a trolley. We have modified this trolley for the purpose or our charging station. Adding a new roller system to the trolley will enhance its ability to swap batteries from charging station to the van or ambulance and vice versa. First of all we have built a roller system using chain, bearing, wheel and belt. The roller structure is 20 inch in length and 8 inch in width (Figure 11.1). Chains are attached to the bearing and bearings can move along the trolley. We have place a belt above the chains to make a platform where the batteries will be positioned and the wheel attached can rotate both in right-handed and left-handed direction so that when we rotate it the batteries can simply slide to the platform both inwards and outwards direction.

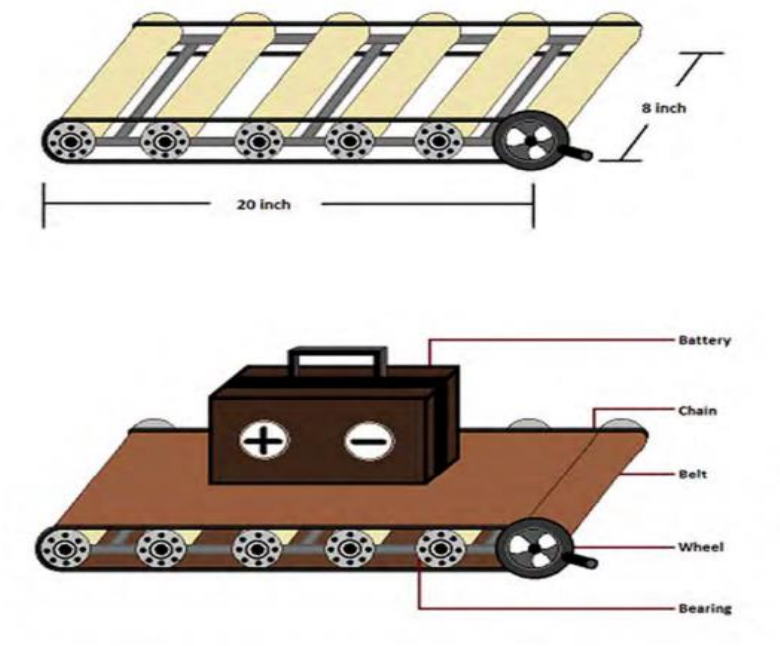

Figure 11.1: Overall roller system design

The whole roller system is placed on the trolley (Figure 11.2). The four stands of the trolley is also attached with one another by wielded iron rod so that it can sustenance the loads of the batteries and likewise the stands becomes firm this way. The trolley is 20.5 inch in height and it would be 30 inch having the batteries placed on the podium. The four stands have wheels underneath them and can move along on the surface.

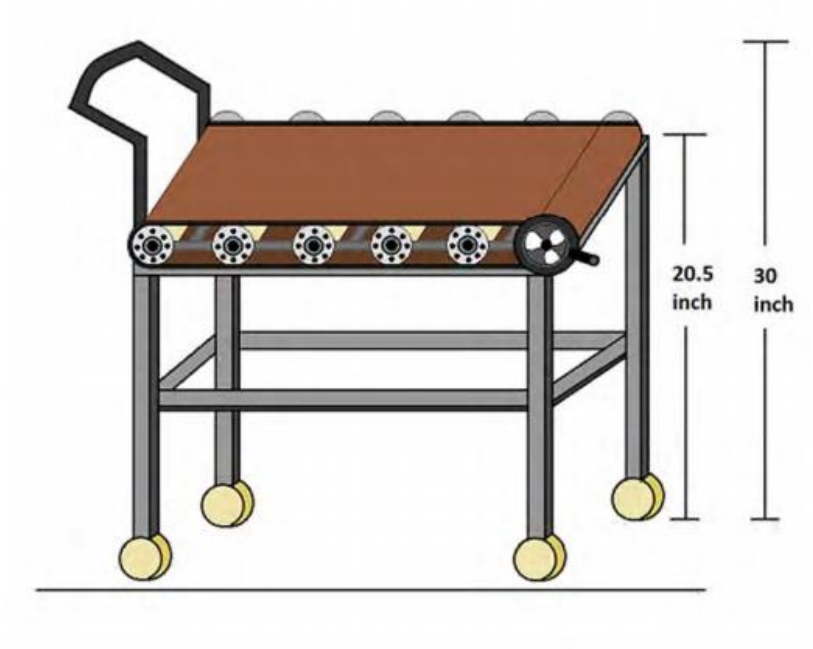

Figure 11.2: The Roller System Based Adjustable Trolley

The swapping would be done in both ways, such as from the station to the van or ambulance and the other way round as well. Therefore, it is a must to have a roller system built in the van or ambulance likewise. When we want to swap batteries we take the trolley to the van and then keep them aligned together then if we rotate the wheel on the trolley in right handed direction (clockwise) the chains and belt will move forward and would take the discharged batteries from the van onto the podium by sliding and when we perform it in anti-clockwise direction it will simply deliver the batteries place on the podium to the van. This simple mechanism used in this swapping system requires less manual labor making it more effective and providing tranquil solution to the swapping method.

CHAPTER 12

CONCLUSION

CONCLUSION

Bangladesh now moving towards renewable energy since being a tropical country it gets ample amount of solar radiation. Now-a-days solar based projects are being implemented all over the country. As a consequence, we were greatly interested in this sector and anticipated to contribute in solar based projects. Our endeavor was to establish a real time monitoring system for solar battery charging station which can display all the current status of the battery so that it can help us observe battery conditions instantaneously. However we have implemented it as a pilot project, it can be successfully expanded of which we have provided the calculations above. In spite of focusing mainly on the battery status feature our software is built in such a way that it adds extra features and makes it more user friendly. In the end we would like to conclude declaring some of the advantages of our thesis-

- All status & measuring can be monitored in real time.
- Any problem can be detected quickly without any difficulties.
- Can easily determine when manual backup is needed.
- Status of multiple battery sets can be shown through one channel at a time.
- Along with the battery, solar status can be acknowledged.
- Helpful for off-grid areas.

Reference

[1] Economist, “The end of the Oil Age,” *The Economist*, 23-October-2003.

[2] The Guardian, “The end of oil is closer thank you think”, 21-April-2005

[3] Reuters, “Bangladesh aims to be world’s 'first solar nation'

[4] AKM Abdul Malek Azad, "The Solar Assisted Rickshaw Van: A Complete Off-Grid Solution" published in IEEE Transportation Electrification newsletter, January/February 2015, USA.

[5] Internet. https://energypedia.info/wiki/Solar_Battery_Charging_Stations

[6] Khan, S.H., Rahman, T., Hossain, S.; “A BRIEF STUDY OF THE PROSPECT OF SOLAR ENERGY IN GENERATION OF ELECTRICITY IN BANGLADESH”, published in Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Renewable and Sustainable Energy (JRSE), June Edition, 2012

[7] Rachaen M. Huq, Thesis on “Development of Torque Sensor Based Electrically Assisted Hybrid Rickshaw,” CARG Project, BRAC University

[8] Advantech. “Active DAQ pro user guide.”

[9] Farhana Chowdhury, Farah Shabnam, Farha Islam, Thesis on “Software Implementation of the Central Solar Battery Charging Station (CSBCS)”, BRAC University

[10] Chhunn Chhim, Nipon Ketjoy, and Tawat Suriwong, “Techno-Economic Analysis of PV Battery Charging Station in Kampot, Cambodia”, Journal of Clean Energy Technologies, Vol. 2, No. 4, October 2014

APPENDIX

Codes used in our software

1st GUI:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SolarGUI
{
    public partial class mainForm : Form
    {
        public mainForm()
        {
            InitializeComponent();
        }

        private bool set = false;


        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void mainForm_Load(object sender, EventArgs e)
        {

        }

        private void btnBatteries_Click(object sender, EventArgs e)
        {
            showBat f2 = new showBat();
            f2.ShowDialog();

        }

        private void button2_Click(object sender, EventArgs e)
        {

            if (!this.set)
{
```

```
this.set = true;
timer1.Enabled = true;
timer1.Start();
}//timer1 starts



        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {

        }




        private void timer1_Tick(object sender, EventArgs e)
        {

            DemoAI.ChannelNow = 14;
            double channel14 = Math.Round(DemoAI.DataAnalog, 3);
            double k = channel14;
            double joker = 12.96 * k;
            double riddler = Math.Round(joker, 2); //Solar V

            text4.Text = " ";
            text5.Text = " ";

            double v1;
            String v;
            v1 = Math.Round(riddler, 2);
            v = Convert.ToString(v1);
            text3.Text = "";
            text3.Refresh();
            text3.Text = v + "    V";  //Showing solar V
            text3.Refresh();

            if (v1 < 51 && v1 > 44)
            {
                text4.Text = "Manual Backup Recommended";


            }
            if (v1 < 43.9)
            {
                text5.Text = "Error!! Check Panel Status ";


            }
            DemoAI.ChannelNow = 15;
            double channel15 = Math.Round(DemoAI.DataAnalog, 1);
```

```
        double l = channel15;

        double m = (k - l) * 6.15002000; //solar I



        w1 = Math.Round(m, 2);
        double w2 = 2.3;
        string w = Convert.ToString(w2);
        textbox1.Text = "";
        textbox1.Refresh();
        textbox1.Text = w + "    A"; //showing solar I
        textbox1.Refresh();




        //solar panel status
         DemoAI.ChannelNow = 12;
         double channel12 = Math.Round(DemoAI.DataAnalog, 3);
         double k1 = 7.90 * channel12;



        DemoAI.ChannelNow = 12;
        double channel12 = Math.Round(DemoAI.DataAnalog, 3);
        double k1 = 12.96 * channel12;
        string kk = Convert.ToString(k);

        DemoAI.ChannelNow = 13;
        double channel113 = Math.Round(DemoAI.DataAnalog, 3);
        double l1 = channel113 * 12.96;
        string ll = Convert.ToString(l);
    }

        double SV = k1 + l1;
        string svv = Convert.ToString(SV);
        text3.Text = svv;

        double a = l1 * (15 / 100);
        double x = l1 + a;
        double y = l1 - a;


        if (k1 > x)
        {
            text5.Text = "Error! Check Panel Status";
        }
        else if (k1 < y)
        {
            text5.Text = "Error! Check Panel Status";
        }

        if (w1 < 4) {

            text4.Text = "Manual Backup Recommended";
```

```
            }



        private void button1_Click_1(object sender, EventArgs e)
        {

            this.set = false;

            text4.Text = " ";
            text5.Text = " ";

            timer1.Stop();
        }

        private void label6_Click(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            Form19 f19 = new Form19();
            f19.ShowDialog();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            DemoAI.SelectDevice();
            String k = Convert.ToString(DemoAI.DeviceNumber);
            text7.Text = k;
            text6.Text = DemoAI.DeviceName;

        }
    }
}
```

2nd GUi:

Battery 13-24

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SolarGUI
{
    public partial class Form22 : Form
    {
        public Form22()
        {
            InitializeComponent();
        }

        private void btn1_Click(object sender, EventArgs e)
        {
            Form20 f23 = new Form20();
            f23.ShowDialog();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            DemoAI.SelectDevice();
            String k = Convert.ToString(DemoAI.DeviceNumber);
            text1.Text = k;
            text2.Text = DemoAI.DeviceName;

        }

        private bool set = false;

        private void button4_Click(object sender, EventArgs e)
        {
            if (!this.set)
            {
                this.set = true;
                timer1.Enabled = true;
                timer1.Start();
            }//timer1 starts


        }

        private void button2_Click(object sender, EventArgs e)
        {
```

```
    DemoAO.ChannelNow = 0;
    DemoAO.DataAnalog = 5;
    text3.Text = "Batteries 13 to 24 Activated";
}

private void button5_Click(object sender, EventArgs e)
{
    DemoAO.ChannelNow = 0;
    DemoAO.DataPhysics = 0;

    this.Close();
}

private void button3_Click(object sender, EventArgs e)
{

}

private void timer1_Tick(object sender, EventArgs e)
{
    DemoAI.ChannelNow = 0;

    double channel0 = Math.Round(DemoAI.DataAnalog, 2);

    double k = 7.90 * channel0;

    if (channel0 >= 0 && channel0 < 0.025)
    {
        textt1.Refresh();
        textt1.Text = "Disconnected";
        textt2.Refresh();
        textt2.Text = "";

        String v;
        v = Convert.ToString(k);
    }

    if (channel0 >= 0.025 && channel0 < 9.8)
    {



        if (channel0 < 5.69)
        {
            textt1.Refresh();
            textt1.Text = "";
            textt2.Refresh();
            textt2.Text = "";

            String v;
            v = Convert.ToString(k);
        }


        else if (channel0 >= 5.7)
        {
            textt1.Refresh();
            textt2.Refresh();
```

```
  if (channel0 >= 6.35)
  {
      String v;
      v = Convert.ToString(k);

      textt1.Refresh();

      textt1.Text = "100%";
      textt2.Refresh();


  }


  else if (channel0 >= 6.29 && channel0 < 6.35)
  {

      String v;
      v = Convert.ToString(k);

      textt2.Refresh();
      textt2.Text = "";
  }

  else if (channel0 >= 6.23 && channel0 < 6.29)
  {

      String v;
      v = Convert.ToString(k);
      textt2.Refresh();
      textt2.Text = "";

  }

  else if (channel0 >= 6.16 && channel0 < 6.23)
  {

      String v;
      v = Convert.ToString(k);
      textt2.Refresh();
      textt2.Text = "";
textt2.Text = "2 hours 30 minutes";
  }

  else if (channel0 >= 6.10 && channel0 < 6.16)
  {

      String v;
      v = Convert.ToString(k);
      textt2.Refresh();
      textt2.Text = "";
      textt2.Text = "3 hours";

  }

  else if (channel0 >= 6.03 && channel0 < 6.10)
  {
```

```
            String v;
            v = Convert.ToString(k);
            textt1.Refresh();
            textt2.Refresh();
            textt2.Text = "";
          textt2.Text = "3 hours 30 minutes";
            textt2.Refresh();

        }

        else if (channel0 >= 5.96 && channel0 < 6.03)
        {

            String v;
            v = Convert.ToString(k);
            textt2.Refresh();
            textt2.Text = "";
            textt2.Text = "4 hours";

        }

        else if (channel0 >= 5.88 && channel0 < 5.96)
        {

            String v;
            v = Convert.ToString(k);
            textt2.Refresh();
            textt2.Text = "";
            textt2.Text = "4 hours 30 minutes";

        }

        else if (channel0 >= 5.81 && channel0 < 5.88)
        {

            String v;
            v = Convert.ToString(k);
            textt2.Refresh();
            textt2.Text = "";
            textt2.Text = "5 hours";

        }

        else if (channel0 >= 5.74 && channel0 < 5.81)
        {

            String v;
            v = Convert.ToString(k);
            textt2.Refresh();
            textt2.Text = "";
            textt2.Text = "5 Hours";

        }

    }

    if (k >= 45 && k <= 47)
```

```
{
    double z1 = (k - 45) * (100 / 9);
    double z12 = Math.Round(z1, 2);
    string z11 = Convert.ToString(z12);

    textt1.Refresh();
    textt1.Text = z11 + " %";

    if (z12 >= 100)
    {
        textt1.Text = "100 %";
    }
    if (z12 >= 100)
    {
        textt1.Text = "100 %";
        textt2.Text = "Fully Charged";
    }
    if (z12 < 100 && z12 > 90)
    {
        textt2.Text = "30 minutes";
    }
    if (z12 < 90 && z12 > 80)
    {
        textt2.Text = "1 Hour";
    }
    if (z12 < 80 && z12 > 70)
    {
        textt2.Text = "1 Hr 30 min";
    }
    if (z12 < 70 && z12 > 60)
    {
        textt2.Text = "2 Hours";
    }
    if (z12 < 60 && z12 > 50)
    {
        textt2.Text = "2 Hr 30 min";
    }
    if (z12 < 50 && z12 > 40)
    {
        textt2.Text = "3 Hours";
    }
    if (z12 < 40 && z12 > 30)
    {
        textt2.Text = "3 Hr 30 min";
    }
    if (z12 < 30 && z12 > 20)
    {
        textt2.Text = "4 Hours";
    }
    if (z12 < 20 && z12 > 10)
    {
        textt2.Text = "4 Hr 30 min";
    }
    if (z12 < 10 && z12 > 0)
    {
        textt2.Text = "5 Hours";
    }
    if (z12 >= 100)
```

```
        {
            //text4.Text = "100 %";
            textt2.Text = "Fully Charged";
        }
    }

    else if (k > 47 && k <= 52)
    {
        double z1 = (k - 45) * (100 / 6.3);
        double z12 = Math.Round(z1, 2);
        string z11 = Convert.ToString(z12);

        textt1.Refresh();
        textt1.Text = z11 + " %";

        if (z12 >= 100)
        {
            textt1.Text = "100 %";
        }
        if (z12 >= 100)
        {
            textt1.Text = "100 %";
            textt2.Text = "Fully Charged";
        }
        if (z12 < 100 && z12 > 90)
        {
            textt2.Text = "30 minutes";
        }
        if (z12 < 90 && z12 > 80)
        {
            textt2.Text = "45 minutes";
        }
        if (z12 < 80 && z12 > 70)
        {
            textt2.Text = "1 Hour";
        }
        if (z12 < 70 && z12 > 60)
        {
            textt2.Text = "1 hr 15 min";
        }
        if (z12 < 60 && z12 > 50)
        {
            textt2.Text = "1 Hr 30 min";
        }
        if (z12 < 50 && z12 > 40)
        {
            textt2.Text = "1 Hr 45 min";
        }
        if (z12 < 40 && z12 > 30)
        {
            textt2.Text = "2 Hours";
        }
        if (z12 < 30 && z12 > 20)
        {
            textt2.Text = "2 Hr 15 min";
        }
        if (z12 < 20 && z12 > 10)
        {
```

```
                    textt2.Text = "2 Hr 30 min";
                }
                if (z12 < 10 && z12 > 0)
                {
                    textt2.Text = "3 Hours";
                }
                if (z12 >= 100)
                {

                }
            }

            else if (k > 52)
            {
                textt1.Text = "100 %";
                textt2.Text = "Fully Charged";
            }

            DemoAI.ChannelNow = 0;

   double channel1 = Math.Round(DemoAI.DataAnalog, 2);

   double k2 = 7.90 * channel0;

   if (channel0 > 0 && channel0 < 0.025)
   {
       textt3.Refresh();
       textt3.Text = "";
       textt4.Refresh();
       textt4.Text = "";

       String v2;
       v2 = Convert.ToString(k);
}

   if (channel0 >= 0.025 && channel0 < 9.8)
   {



       if (channel0 < 5.69)
       {
           textt3.Refresh();
           textt3.Text = "";
           textt4.Refresh();
           textt4.Text = "";

           String v2;
           v2 = Convert.ToString(k);
           }


       else if (channel0 >= 5.7)
       {
           textt3.Refresh();
           textt4.Refresh();

           if (channel0 >= 6.35)
```

```
{
    String v2;
    v2 = Convert.ToString(k);

    textt3.Refresh();

    textt3.Text = "100%";
    textt4.Refresh();

    textt4.Text = "Fully Charged";
}


else if (channel0 >= 6.29 && channel0 < 6.35)
{

    String v;
    v = Convert.ToString(k);

    textt2.Refresh();
    textt2.Text = "";
    textt2.Text = "1 hour";
}

else if (channel0 >= 6.23 && channel0 < 6.29)
{

    String v;
    v = Convert.ToString(k);
    textt2.Refresh();
    textt2.Text = "";
    textt2.Text = "2 hours";

}

else if (channel0 >= 6.16 && channel0 < 6.23)
{

    String v;
    v = Convert.ToString(k);
    textt2.Refresh();
    textt2.Text = "";
    textt2.Text = "2 hours 30 minutes";
                    }

else if (channel0 >= 6.10 && channel0 < 6.16)
{

    String v;
    v = Convert.ToString(k);
    textt2.Refresh();
    textt2.Text = "";
    textt2.Text = "3 hours";

}

else if (channel0 >= 6.03 && channel0 < 6.10)
{
```

```
        String v;
        v = Convert.ToString(k);
        textt1.Refresh();
        textt2.Refresh();
        textt2.Text = "";
        textt2.Text = "3 hours 30 minutes";
        textt2.Refresh();

    }

    else if (channel0 >= 5.96 && channel0 < 6.03)
    {

        String v;
        v = Convert.ToString(k);
        textt2.Refresh();
        textt2.Text = "";
        textt2.Text = "4 hours";

    }

    else if (channel0 >= 5.88 && channel0 < 5.96)
    {

        String v;
        v = Convert.ToString(k);
        textt2.Refresh();
        textt2.Text = "";
        textt2.Text = "4 hours 30 minutes";

    }

    else if (channel0 >= 5.81 && channel0 < 5.88)
    {

        String v;
        v = Convert.ToString(k);
        textt2.Refresh();
        textt2.Text = "";
        textt2.Text = "5 hours";

    }

    else if (channel0 >= 5.74 && channel0 < 5.81)
    {

        String v;
        v = Convert.ToString(k);
        textt2.Refresh();
        textt2.Text = "";
        textt2.Text = "5 Hours";

    }

}

if (k >= 45 && k <= 47)
```

```
        {
            double z1 = (k - 45) * (100 / 9);
            double z12 = Math.Round(z1, 2);
            string z11 = Convert.ToString(z12);

            textt1.Refresh();
            textt1.Text = z11 + " %";

            if (z12 >= 100)
            {
                textt1.Text = "100 %";
            }

        }

        else if (k > 47 && k <= 52)
        {
            double z1 = (k - 45) * (100 / 6.3);
            double z12 = Math.Round(z1, 2);
            string z11 = Convert.ToString(z12);

            textt1.Refresh();
            textt1.Text = z11 + " %";

            if (z12 >= 100)
            {
                textt1.Text = "100 %";
            }
        }

        else if (k > 52)
        {
            textt1.Text = "100 %";
        }


             }
         }

         private void textt2_TextChanged(object sender, EventArgs e)
         {

         }
    }
}
```

Panel 1 and panel 2:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SolarGUI
{
    public partial class Form19 : Form
    {
        public Form19()
        {
            InitializeComponent();
        }

        public bool set = false;

        private void button1_Click(object sender, EventArgs e)
        {
            if (!this.set)
            {
                this.sat = true;
                timer1.Enabled = true;
                timer1.Start();
            }//timer1 starts

        }

        private void button2_Click(object sender, EventArgs e)
        {
            DemoAI.SelectDevice();
            String k = Convert.ToString(DemoAI.DeviceNumber);
            text5.Text = k;
            text6.Text = DemoAI.DeviceName;

        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            text3.Text = " ";
            text4.Text = " ";
            DemoAI.ChannelNow = 12;
            double channel12 = Math.Round(DemoAI.DataAnalog, 3);
            double k = 7.90 * channel12;
            string kk = Convert.ToString(k);
            text1.Text = k + "  V"; //1st panel V

            DemoAI.ChannelNow = 13;
            double channel13 = Math.Round(DemoAI.DataAnalog, 3);
            double l = channel13 * 7.90;
```

```
            string ll = Convert.ToString(l);
            text2.Text = ll + "  V";//2nd panel v

            if (k > (l + 6))
            {
                text4.Text = "Error In Panel 2";
            }
            else if  (k <( l - 6))
                //text3.Text = " ";

            {
                text3.Text = "Error In Pane1 ";
            }
                else

              text4.Text = " ";









        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.set = false;
            timer1.Stop();

        }

        private void text3_TextChanged(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Battery set 1-12:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SolarGUI
{
    public partial class Form19 : Form
    {
        public Form19()
        {
            InitializeComponent();
        }

        public bool set = false;

        private void button1_Click(object sender, EventArgs e)
        {
            if (!this.set)
            {
                this.sat = true;
                timer1.Enabled = true;
                timer1.Start();
            }//timer1 starts

        }

        private void button2_Click(object sender, EventArgs e)
        {
            DemoAI.SelectDevice();
            String k = Convert.ToString(DemoAI.DeviceNumber);
            text5.Text = k;
            text6.Text = DemoAI.DeviceName;

        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            text3.Text = " ";
            text4.Text = " ";
            DemoAI.ChannelNow = 12;
            double channel12 = Math.Round(DemoAI.DataAnalog, 3);
            double k = 7.90 * channel12;
            string kk = Convert.ToString(k);
            text1.Text = k + "  V"; //1st panel V

            DemoAI.ChannelNow = 13;
            double channel13 = Math.Round(DemoAI.DataAnalog, 3);
            double l = channel13 * 7.90;
            string ll = Convert.ToString(l);
            text2.Text = ll + "  V";//2nd panel v

            if (k > (l + 6))
```

```
    {
        text4.Text = "Error In Panel 2";
    }
    else if  (k <( l - 6))
        //text3.Text = " ";

    {
        text3.Text = "Error In Pane1 ";
    }
        else

      text4.Text = " ";
```

```
 double a = l * (5 / 100); //tol
 double x = l + a;
        double y = l - a;
    double p = l+l*(50/100);
    double q = l-l*(50/100);


    if (k > p)
    {
        text4.Text = "Error In Panel 2";
        text3.Text = " ";
    }

    if (k < q)
    {
        text3.Text = "Error In Panel 1";
        text4.Text = " ";
    }

    if (k <= p && k >= q)
    {
        text4.Text = " ";
        text3.Text = " ";

    }
```

```
}

private void button3_Click(object sender, EventArgs e)
{
    this.set = false;
    timer1.Stop();
```

```
        }

        private void text3_TextChanged(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Battery set 1 GUI:

Set 1:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SolarGUI
{
    public partial class form1 : Form
    {

        public form1()
        {
            InitializeComponent();
        }

        private void form1_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            DemoAI.SelectDevice();
            String k = Convert.ToString(DemoAI.DeviceNumber );
            text1.Text = k;
            text2.Text = DemoAI.DeviceName;
```

```
        }

        private bool set = false;

        private void button2_Click(object sender, EventArgs e)
        {
            if (!this.set)
            {
                this.set = true;
                timer1.Enabled = true;
                timer1.Start();
            }//timer1 starts


        }

        private void text3_TextChanged(object sender, EventArgs e)
        {


        }

        private void DemoAI_OnTerminate(object sender,
AxAdvAILib._IAdvAIEvents_OnTerminateEvent e)
        {
            DemoAI.SelectDevice();
            double channel0 = Math.Round(DemoAI.DataAnalog, 3);


        }

        private void text1_TextChanged(object sender, EventArgs e)
        {

        }

        private void text2_TextChanged(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void text4_TextChanged(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {
```

```
}

private void text5_TextChanged(object sender, EventArgs e)
{

}

private void label4_Click(object sender, EventArgs e)
{

}

private void text6_TextChanged(object sender, EventArgs e)
{

}

private void text7_TextChanged(object sender, EventArgs e)
{

}

private void timer1_Tick(object sender, EventArgs e)
{
    DemoAI.ChannelNow = 0;

    double channel0 = Math.Round(DemoAI.DataAnalog, 2);

    double k = 7.86 * channel0;

    if (channel0 >= 0 && channel0 < 0.025)
    {
        text4.Refresh();
        text4.Text = "";
        text5.Refresh();
        text5.Text = "";
        text6.Refresh();
        text6.Text = "";
        text7.Refresh();
        text7.Text = "No Battery Is Connected At Port 0";

        String v;
        v = Convert.ToString(k);

        text3.Refresh();
        text3.Text = 0 + "  V";
        text3.Refresh();
    }

    if (channel0 >= 0.025 && channel0 < 9.8)
    {



        if (channel0 < 5.69)
        {
            text4.Refresh();
```

```
    text4.Text = "";
    text5.Refresh();
    text5.Text = "";
    text6.Refresh();
    text6.Text = "";
    text7.Refresh();
    text7.Text = "Battery Is Damaged!!";

    String v;
    v = Convert.ToString(k);
    text3.Text = "";
    text3.Refresh();
    text3.Text = v + "  V";
    text3.Refresh();
}


else if (channel0 >= 5.7)
{
    text3.Refresh();
    text4.Refresh();
    text5.Refresh();
    text6.Refresh();
    text7.Refresh();



    if (channel0 >= 6.35)
    {
        String v;
        v = Convert.ToString(k);


        text3.Refresh();
        text3.Text = v + "  V";

        text4.Refresh();

        text4.Text = "100%";
        text5.Refresh();


        text6.Refresh();

        text6.Text = "High";
        text7.Text = " ";
        text7.Refresh();

    }


    else if (channel0 >= 6.29 && channel0 < 6.35)
    {

        String v;
        v = Convert.ToString(k);
        text3.Text = "";
        text3.Refresh();
```

```
            text3.Text = v + "  V";
            text3.Refresh();
            text5.Refresh();
            text5.Text = "";
            text6.Refresh();
            text6.Text = "";
            text6.Text = "High";
            text7.Text = " ";
            text7.Refresh();
            text7.Text = "";


        }

        else if (channel0 >= 6.23 && channel0 < 6.29)
        {

            String v;
            v = Convert.ToString(k);
            text3.Text = "";
            text3.Refresh();
            text3.Text = v + "  V";
            text3.Refresh();

            //text4.Refresh();
            //text4.Text = "";
            //text4.Text = "80%";
            text5.Refresh();
            text5.Text = "";
//            text5.Text = "2 hours";
            text6.Refresh();
            text6.Text = "";
            text6.Text = "Medium";
            text7.Refresh();
            text7.Text = "";


        }

        else if (channel0 >= 6.16 && channel0 < 6.23)
        {

            String v;
            v = Convert.ToString(k);
            text3.Text = "";
            text3.Refresh();
            text3.Text = v;
            text3.Refresh();

            text5.Refresh();
            text5.Text = "";
            text6.Refresh();
            text6.Text = "";
            text6.Text = "Medium";
            text7.Refresh();
            text7.Text = "";
```

```
}

else if (channel0 >= 6.10 && channel0 < 6.16)
{

    String v;
    v = Convert.ToString(k);
    text3.Text = "";
    text3.Refresh();
    text4.Text = v + "  V";
    text3.Refresh();

    text5.Refresh();
    text5.Text = "";

    text6.Refresh();
    text6.Text = "";
    text6.Text = "Medium";
    text7.Refresh();
    text7.Text = "";


}

else if (channel0 >= 6.03 && channel0 < 6.10)
{

    String v;
    v = Convert.ToString(k);
    text3.Text = "";
    text3.Refresh();
    text3.Text = v + "  V";
    text3.Refresh();

    text4.Refresh();
    text5.Refresh();
    text5.Text = "";
    text5.Refresh();
    text6.Refresh();
    text6.Text = "";
    text6.Text = "Low";
    text6.Refresh();
    text7.Refresh();
    text7.Text = "";
    text7.Refresh();


}

else if (channel0 >= 5.96 && channel0 < 6.03)
{

    String v;
    v = Convert.ToString(k);
    text3.Text = "";
    text3.Refresh();
    text3.Text = v + "  V";
    text3.Refresh();
```

```
    text5.Refresh();
    text5.Text = "";
    text6.Refresh();
    text6.Text = "";
    text6.Text = "Low";
    text7.Refresh();
    text7.Text = "";


}

else if (channel0 >= 5.88 && channel0 < 5.96)
{

    String v;
    v = Convert.ToString(k);
    text3.Text = "";
    text3.Refresh();
    text3.Text = v + "  V";
    text3.Refresh();


    text5.Refresh();
    text5.Text = "";
    text6.Refresh();
    text6.Text = "";
    text6.Text = "Low";
    text7.Refresh();
    text7.Text = "";


}

else if (channel0 >= 5.81 && channel0 < 5.88)
{

    String v;
    v = Convert.ToString(k);
    text3.Text = "";
    text3.Refresh();
    text3.Text = v + "  V";
    text3.Refresh();

    text5.Refresh();
    text5.Text = "";
    text6.Refresh();
    text6.Text = "";
    text6.Text = "Low";
    text7.Refresh();
    text7.Text = "";


}

else if (channel0 >= 5.74 && channel0 < 5.81)
{
```

```
            String v;
            v = Convert.ToString(j);
            text3.Text = "";
            text3.Refresh();
            text3.Text = v + "  V";
            text3.Refresh();


            text5 .Refresh();
            text5.Text = "";
           // text5.Text = "5 Hours";
            text6.Refresh();
            text6.Text = "";
            text6.Text = "Low";
            text7.Refresh();
            text7.Text = "";


        }

        if (channel0 > 0 && channel0 < 0.025)
        {
            text7.Text = "No Battery Is Connected At Port 0";
        }

    }

    if (k >= 45 && k <= 47)
    {
        double z1 = (k - 45) * (100 / 9);
        double z12 = Math.Round(z1, 2);
        string z11 = Convert.ToString(z12);

        text4.Refresh();
        text4.Text = z11 + " %";

        if (z12 < 100 && z12 > 90)
        {
        text5.Text = "30 minutes";
        }
        if (z12 < 90 && z12 > 80)
        {
        text5.Text = "45 minutes";
        }
        if (z12 < 80 && z12 > 70)
        {
        text5.Text = "1 Hour";
        }
        if (z12 < 70 && z12 > 60)
        {
        text5.Text = "1 hr 15 min";
        }
        if (z12 < 60 && z12 > 50)
        {
        text5.Text = "1 Hr 30 min";
        }
        if (z12 < 50 && z12 > 40)
        {
```

```
        text5.Text = "1 Hr 45 min";
        }
        if (z12 < 40 && z12 > 30)
        {
        text5.Text = "2 Hours";
        }
        if (z12 < 30 && z12 > 20)
        {
        text5.Text = "2 Hr 15 min";
        }
        if (z12 < 20 && z12 > 10)
        {
        text5.Text = "2 Hr 30 min";
        }
        if (z12 < 10 && z12 > 0)
        {
        text5.Text = "3 Hours";
        }
        if (z12 >= 100)
        {
            text4.Text = "100 %";
            text5.Text = "Fully Charged";
        }

    }

    else if (k > 47 && k <= 52)
    {
        double z1 = (k - 45) * (100 / 6.3);
        double z12 = Math.Round(z1, 2);
        string z11 = Convert.ToString(z12);

        text4.Refresh();
        text4.Text = z11 + " %";

        if (z12 >= 100)
        {
            text4.Text = "100 %";
            text5.Text = "Fully Charged";
        }
        if (z12 < 100 && z12 > 90)
        {
            text5.Text = "30 minutes";
        }
        if (z12 < 90 && z12 > 80)
        {
            text5.Text = "45 minutes";
        }
        if (z12 < 80 && z12 > 70)
        {
            text5.Text = "1 Hour";
        }
        if (z12 < 70 && z12 > 60)
        {
            text5.Text = "1 hr 15 min";
        }
        if (z12 < 60 && z12 > 50)
        {
```

```
                    text5.Text = "1 Hr 30 min";
                }
                if (z12 < 50 && z12 > 40)
                {
                    text5.Text = "1 Hr 45 min";
                }
                if (z12 < 40 && z12 > 30)
                {
                    text5.Text = "2 Hours";
                }
                if (z12 < 30 && z12 > 20)
                {
                    text5.Text = "2 Hr 15 min";
                }
                if (z12 < 20 && z12 > 10)
                {
                    text5.Text = "2 Hr 30 min";
                }
                if (z12 < 10 && z12 > 0)
                {
                    text5.Text = "3 Hours";
                }
                if (z12 >= 100)
                {
                    text4.Text = "100 %";
                    text5.Text = "Fully Charged";
                }
            }

            else if (k > 52)
            {
                text4.Text = "100 %";
                text5.Text = "Fully Charged";
            }



        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        this.set = false;
        timer1.Stop();

    }

    private void form1_Load_1(object sender, EventArgs e)
    {

    }


}
}
```