Software Implementation of the Central Solar Battery Charging Station (CSBCS)

FARHA ISLAM **ID: 09221046**
FARHANA CHOWDHURY **ID: 09221035**
FARAH SHABNAM **ID: 10321059**

Department of Electrical and Electronic Engineering

Summer 2012

BRAC University, Dhaka, Bangladesh

DECLARATION

We do hereby declare that the thesis titled **"Software Implementation of the Central Solar Battery Charging Station"** is submitted to the Department of Electrical and Electronic Engineering of BRAC University in partial fulfillment of the Bachelor of Science in Electrical and Electronic Engineering. This is our original work and was not submitted elsewhere for the award of any other degree or any other publication.

Dhaka, Date 4th September, 2012

……

Signature of the Supervisor

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

FARHA ISLAM

STUDENT ID: 09221046

……..

FARHANA CHOWDHURY

STUDENT ID: 09221035

..……..……..

FARAH SHABNAM

STUDENT ID: 10321059

……

ACKNOWLEDGEMENT

Our admirable courtesy goes to our thesis coordinator Dr. AKM Abdul Malek Azad, Associate professor, Department of EEE, BRAC University for his valuable help and support throughout our entire thesis period. He extended his helping hand by providing us encouragement, inspiration, facilities and valuable feedback throughout the course of this thesis We are thankful to him as without his cooperation, moral and verbal support , we may not able to bring our dream to real success. Special gratitude to Dr. Shamim, Lecturer, Department of MNS for his coordination regarding renewable energy, panel array designing, calculation of power consumption etc. This project has been funded by EEE department of BRAC University to accomplish the final year thesis project. Special thanks to Control and Application Research Group (CARG) for its invaluable contribution and the project engineers, CARG, Mr. Tahsin Faraz and Mr. Shurjo for their immense advice and support towards our team.

ABSTRACT

SBCS is the beginner's outline of the Central Solar Battery Charging Station which can be worn as an alternation of the complementary power source or electricity for assembling the ever-increasing demand of our country, Bangladesh. Solar energy is a form of best renewable energy and by implementing this project as a prototype one our motto is to extend the voice for "Go Green-Save the Planet-work for existence". Software implementation is needed to monitor the charging status of the individual batteries in a station. It can also detect whether the battery has been damaged or not at a certain voltage level. Thus, we have deliberated software to monitor all the probable actions of the batteries of the SBCS in the software development for Central Solar Battery Charging Station. It indicates the charging status independently for each battery coupled to the prototype central charging station through a charge controller that deliver the requisite quantity of current to the battery and also connected superficially with a Data Acquisition card to examine the battery state with our software deployed using VISUAL STUDIO 2010. Consequently, the entire charging process of the central charging station is a solitary means of communication as we are going to provide input quantity directly from PV solar panel or diesel generator (unconventional source).

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Central Solar Battery Charging Station (CSBCS) provides power to trickle charging of batteries from stand-alone solar panels. People bring their own batteries or rent from the station for recharging up to a specific voltage level-which is monitored by the newly developed software dedicated for this project. CSBCS was initially conceived worldwide to bring the price per household of electrification within the capacity to pay of the rural poor, and to foster the establishment of community businesses supplying the modest electricity demands of end users far from the grid in an entrepreneur-based electrification model.

Considering the raising needs for electricity, it is quite impossible for all to bring individual solar home system for each house. Therefore, solar battery charging station is a good solution. When the solar batteries come into account, they get charged in a very short time period considering of the solar/sun/light hours per day, which is 5 hours in Bangladesh; whereas Diesel Battery Charging Stations (DBCS) take 1-2 days [1].

As, solar energy is abundant and free, it can be used to charge batteries used for any module or electrical kits which are obvious for daily usage through a well developed CSBCS. Our aim is to make solar energy popular through implementing software for Central Solar Battery Charging Station with a view to provide supplementary electricity without causing harm to the batteries. The electricity is instantaneously converted and then stored in the charging station which is consumed by the batteries. If the panels produce power which is not required instantly, customers can get hold of that energy in the outlook, whenever they oblige it. People whoever looking for savings and the future of the planet should indeed look into solar energy.

1.1 Background and Motivation

Crisis of electricity is a major problem in the present era. In some area outside the city side, there is general electricity service called 'PALLI BIDYUT' which can supply a very limited amount of electricity in those areas that is unable to cover up their basic demands. The existing electric grids are not capable of supplying the electric need. The great foreseen advantages of SBCS are security of payment for the electricity service and operation under much higher system final yields and capacity factors.

Considering cases on the successful implementation of SBCS in some foreign lands such as, Cambodia or Iraq or India, we are also motivated to use this concept too. It is one of the best alternatives for charging automotive batteries used either in solar home system or in solar Rickshaw/any solar powered vehicles or for charging purpose of lanterns or even can be used as an alternative of diesel or kerosene station with modifying the station.

Under the same project, a whole Central Solar Battery Charging Station (CSBCS) dividing into three separate groups along with the successful implementation of hardware part where the batteries are being charged from the PV cells through the charge controller. Implementation of SBCS also includes designing of a charge controller, making it capable of charging multiple batteries simultaneously and getting the desired current from the load.

Inclusion of the CSBCS we have effectively developed an unique GUI that represent all the activities visually that can also can be monitored and controlled from remote region. Resultantly, it will make the whole job of maintenance of the solar battery charging station even more effective. In our proto-type project, our aim is to charge one 48V 10 Ah automotive battery used in Solar Rickshaw and one 12V 80 Ah battery used in Solar Home System.

1.2 DESCRIPTION

Bangladesh is a tropical country where the amount of sunlight is mostly available to meet up the demand of producing electricity. This type of project is not new but for our country, this can be implemented successfully for commercial purpose. This project can bring a revolutionary change in the lifestyle and the economical prospectus that also can increase the GDP of Bangladesh. As Bangladesh is a massively power-deficient country with peak power shortages of around 25%. More than 60% of its people do not have access to the power grid. The country only produces 3500-4200 MW of electricity against a daily demand for 4000-5200 MW on average, according to official estimates [1][2]. Solar energy is an ideal solution as it can provide griddles power and is totally clean in terms of pollution and health hazards. Since it saves money on constructing electricity transmission lines, it is economical as well. The solar panel providers in Bangladesh are now expecting the price of batteries and accessories to drastically reduce. Moreover, after the current budget of 2012, the price for per unit electricity will be amplified. So the best alternative is development of SBCS in our country effectively.

1.3 REASONS BEHIND SOFTWARE IMPLEMENTATION OF SBCS

The motto of software implementation is to design a monitoring system, interfacing with the DAQ (ADVANTECH USB 4716) to receive signal that is being generated from the charge controller using the software that have been developed using Microsoft Visual Studio 2010. We have designed a primary GUI which includes multiple GUIs that works as individual software using the Active DAQ Pro functions (.dlls) of Advantech USB 4716. The software we built monitors the charging status of batteries by performing multiple tasks such as it shows the charging status in voltage, the time remaining to get fully charged, voltage level, the percentage charged and visual representation of charging status using progress bar for individual batteries; altogether for 16 batteries connected to 16 analog input channels of DAQ card where

Advantech USB 4716 Device manager can only display voltages from the corresponding input channels.

1.4 OBJECTIVE OF THE PROTO-TYPE PROJECT

Day by day our natural resources are decreasing. So the future of our next generation will be more heinous if we cannot implement any alternative for them. Our future is closely bonded with the modification and usage of modern technology. Therefore the first and foremost condition to develop our country through technological improvement is to supply the necessary amount of electricity. Thereby, the Implementation of Central Solar Battery Charging Station can be one of the best alternatives that can broaden the way of getting rid of electricity problem in many ways.

As for example:

- We can charge the batteries used in solar home system or in IPS in our station and our well developed monitoring software will save the batteries from further destructions caused by the system.
- The charging station can be used to charge Rickshaw battery or batteries used in Solar Home System either in rental or monthly payment basis.
- Electric lanterns used in village area can be charged as well.
- Moreover, our software is able to eliminate costs.
- The unemployed people including women of the village area can earn their livelihood by being employed in the charging station that can be implemented in near future following our prototype design.
- The load of national grid will be lessening resultantly.

- Moreover, followed by some other countries we can also replace kerosene station with Solar Battery Charging Station too with further modification.
- Lastly, our project is an environment friendly one.

1.5 OUTLINE OF THE THESIS PAPER

The entire paper reveals the up-coming possibilities that can be earned from using solar energy through building Solar Battery Charging Station with software implementation. Here details about the current development and procedure have been clearly notified. Chapter 2 is all about the overview of the proto-type project including the different parts correlated to clarify in details. Chapter 3 includes techniques of reading the battery state corresponds to different charging state and Chapter 4 describes about the battery and solar panel management system. In chapter 5 Data acquisition system has been described. Chapter 6 deals with the challenging issues- interfacing between software and hardware in windows environment. Chapter 7 is all about retrieving signal using visual studio. Lastly, in chapter 8 -conclusion and future works highlights the success as well as the boundaries of this hand-shaking project along with the upcoming development of our software.

CHAPTER 2

OVERVIEW OF THE PROTOTYPE PROJECT

2.1 OVERVIEW

A two-dimensional project with hardware and software implementation along with designing a microcontroller based solar charger for Central Solar Battery Charging Station has been designed. On behalf of the software part of the SBCS, we are responsible for maintenance of the batteries through monitoring of the central solar battery charging station. The whole project is not only internally connected but also responsible for all sorts of events as shown below:

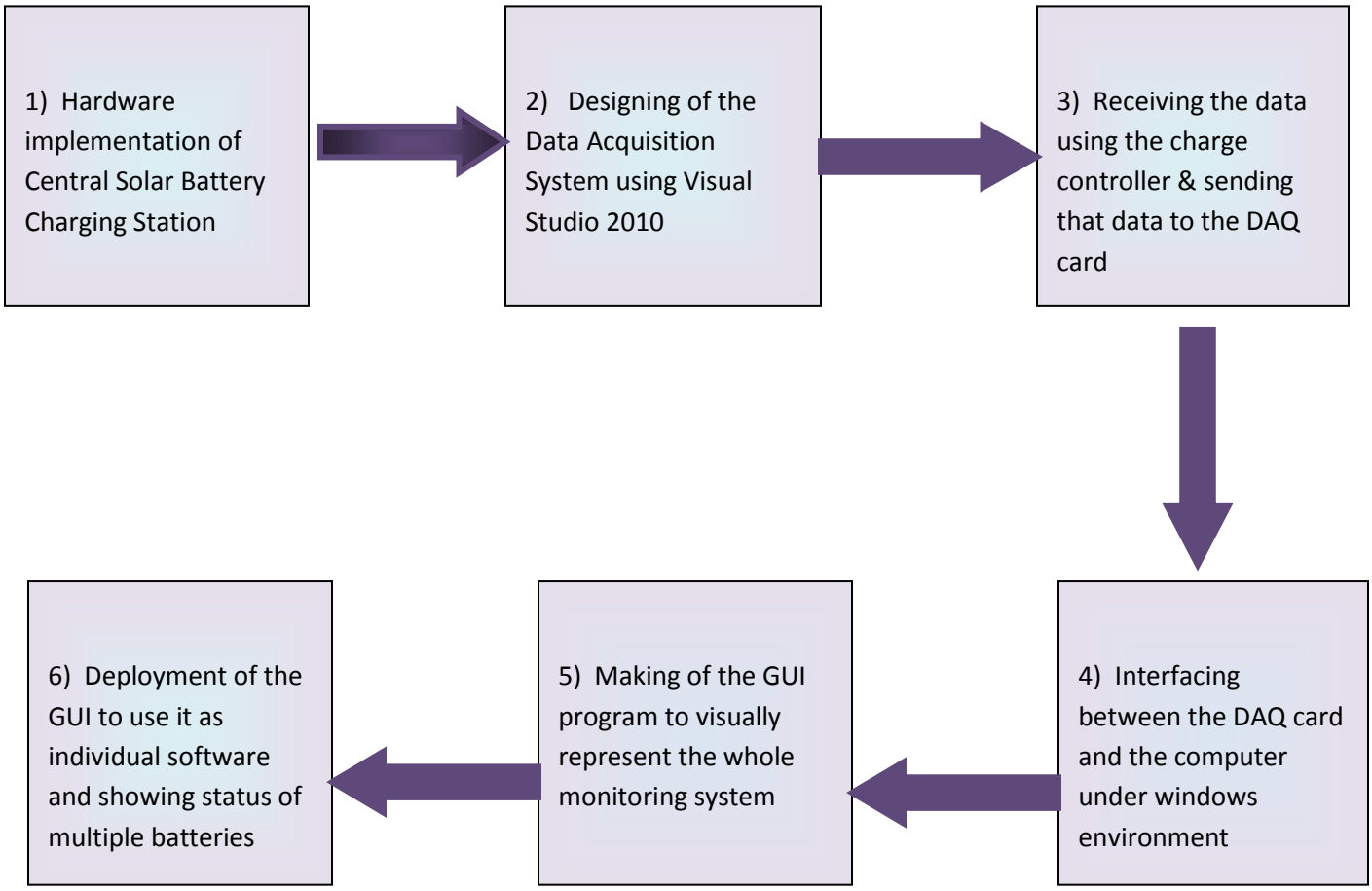


Fig 2.1: Flowchart Representation of the Whole Project

2.2 STANDARAD MODEL OF SBCS

The standard model of SBCS that we have followed to design our monitoring software is given below:

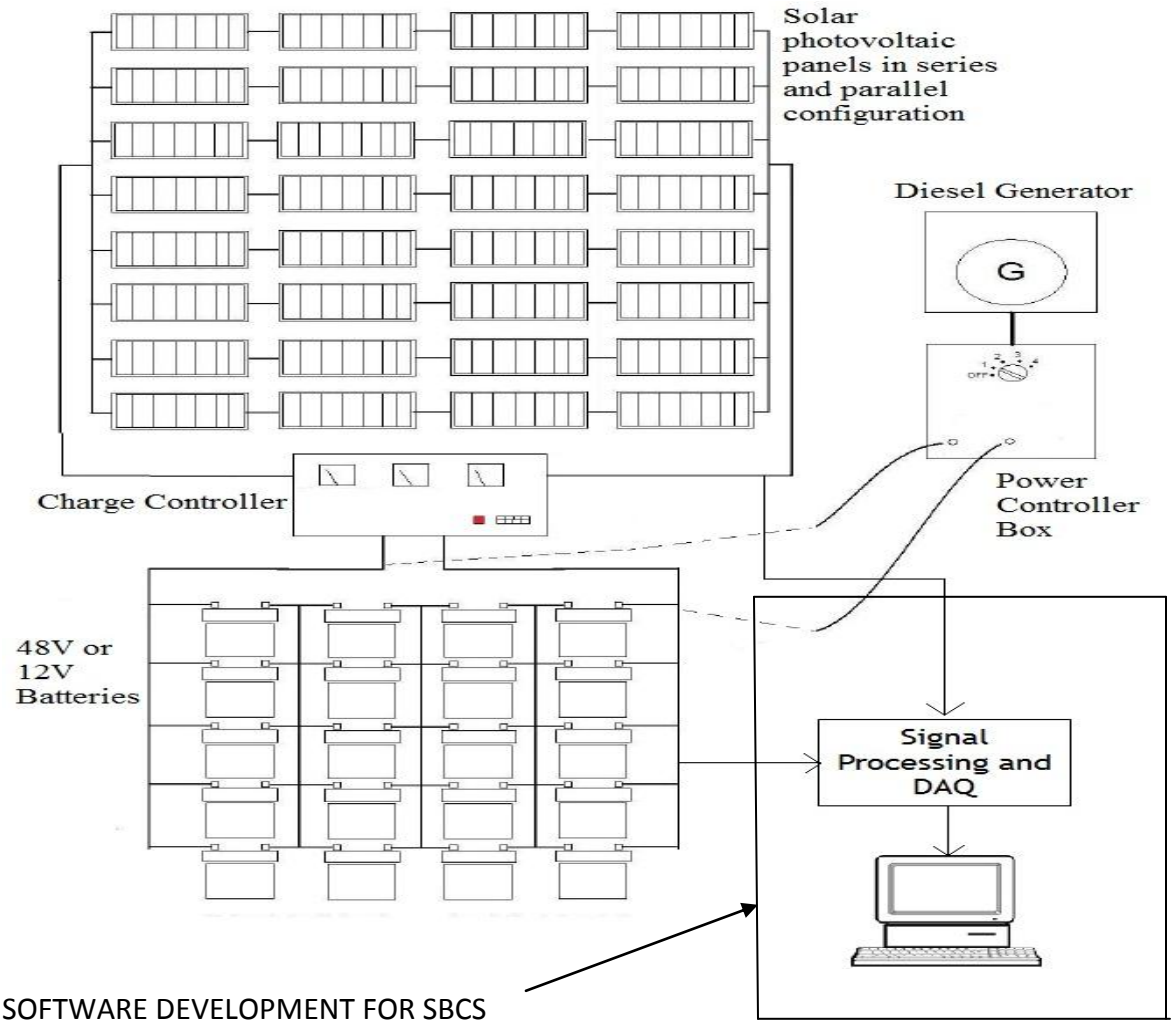


Fig 2.2: Standard model of SBCS

2.3 HARDWARE IMPLEMENTATION

Solar radiation produced by nuclear fusion reactions deep in the Sun's core. This is mainly done by a charge controller. It controls the battery charging according to the desired need. An efficient charge controller can be used to do the battery charging and discharging process faster and better. The existing electric grids are not capable of supplying the electric need. Whereas, the Solar Battery Charging Station (SBCS) has emerged to the rural areas in Bangladesh as well as in urban areas to change the scenario. Now, the required manpower and economic problem is less.

In the hardware implementation project following components are there:

1. A PV Generator composed of one or more PV modules, which are interconnected to form a DC power-producing unit.

2. A mechanical support structure for the PV generator.

3. A 12V lead acid battery.

4. A charge controller to prevent deep discharges and overcharges of the battery

5. Loads (LED lamps)

6. Wire connections (Cable, switches and connection box.)

Each component of the system must fulfill the quality and requirements. Size, voltage thresholds of the charge controller, the quality of installation etc directly effects the lifetime of batteries and lamps.

2.1.2 SOFTWARE IMPLEMENTATION

Software implementation of SBCS is vitally important to monitor the system state of charge and keep the batteries safe. While maintaining the batteries of the SBCS manually, there might occurs mistakes and batteries can get overcharged. But doing it using software is not only safe but also cost effective.

The main reasons that motivated us to implement efficient software are:

- ✓ A false practice to check charging status of battery using flashlight tends to damage battery due to overcharging
- ✓ The GUI of existing DAQ card has fewer features with few or no graphical representation of the voltage detected.

The software that we developed is a keen project where we had our full motivation to create a GUI using with VISUAL STUDIO 2010. We implemented our software based on all likely criterions that can arise in charging batteries in minimum time exclusive of causing any sort of damage. First of all we have design the Data Acquisition System using Data Acquisition Card (ADVANTECH USB 4716). This card is a 16 bit system with 5Volts-10Volts of reference to make the card resolution up to 0.7mv that is enough accurate for our development. Afterwards, the processing of the DAQ card is done; the data we received is send to the computer. Lastly we have to perform the deployment the GUI that we have made.

The whole job of this software implementation can be showed with the flowchart below:

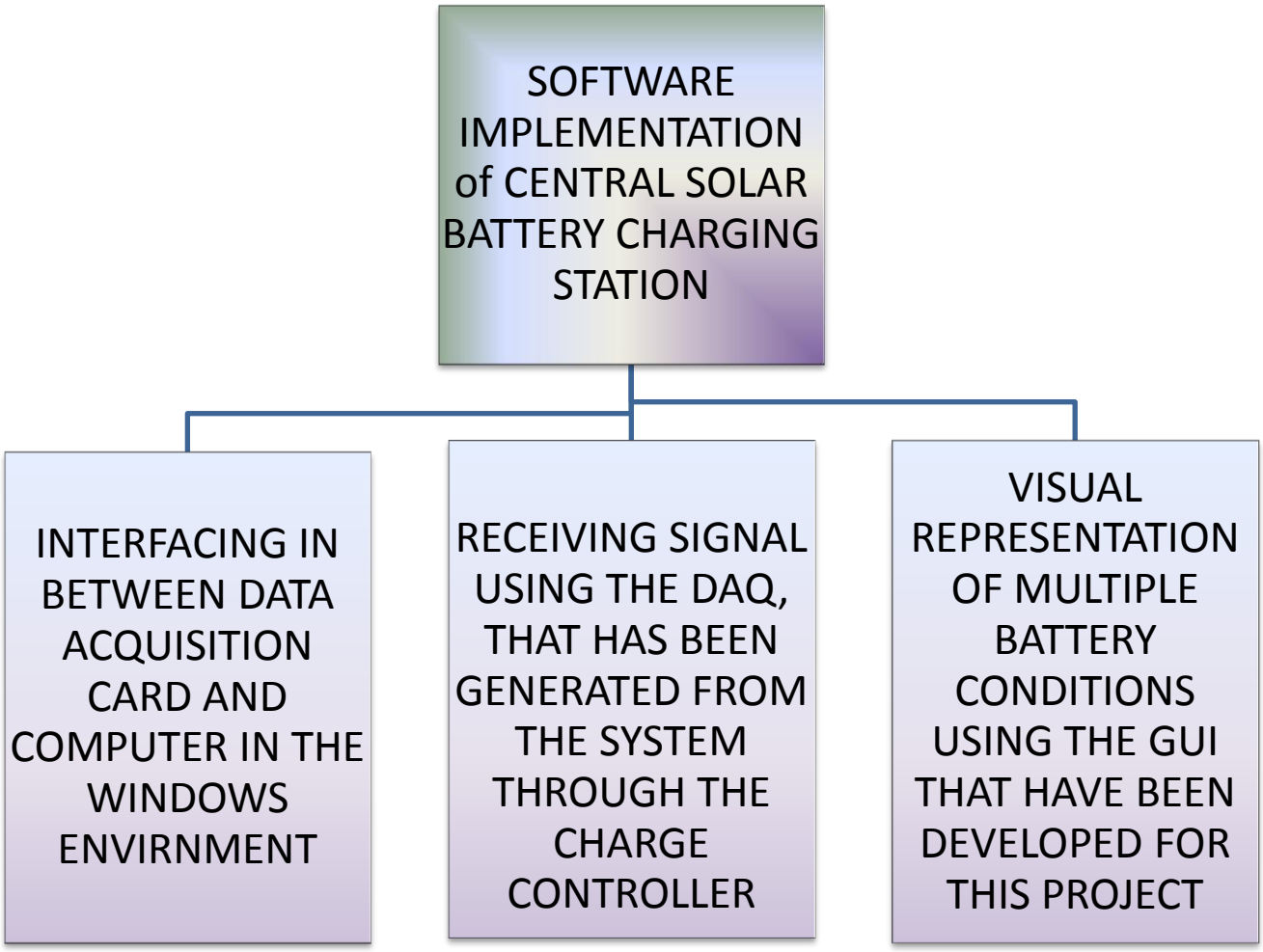


Fig 2.1.2: Block Diagram Representation

The software that we have made named “Solar Battery Charging Status Monitoring Panels” is more efficient than human as it can trace out all possible errors with manual maintenance. It

traces charging status in voltage, time remaining to get fully charged, voltage level, percentage charging and visual representation of charging status by progress bar for individual batteries simultaneously. Again if any of the battery is damaged, then our GUI only shows its status in voltage and in progress bar rather than performing any further calculation for that particular battery required for operating the controller. We can use this software either in rental basis or people can bring their own batteries and get those charged in exchange of monthly payment as like as paying electricity or phone bills monthly.

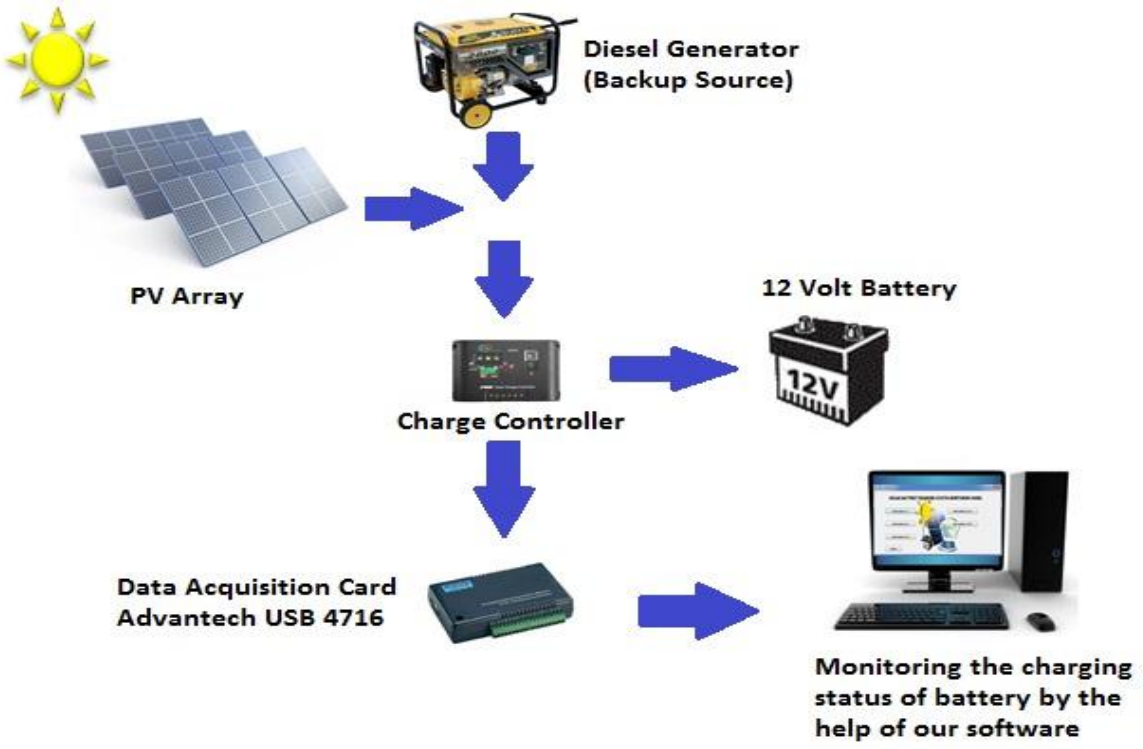


Fig2.1.3 Overview of the system

CHAPTER 3

TECHNIQUES OF READING THE BATTERY STATES

As we are dealing with charging different types of batteries using our embedded it is important for us to know the state of charge that is formerly known as SOC. Moreover we need to define corresponding voltage level for showing the charging state using our well designed software. Thereby it is important to define SOC clearly.

3.1 State of Charge (SOC):

A key parameter of a battery in use in a PV system is the battery state of charge (BSOC) or (SOC). Battery state of charge (BSOC or SOC) gives the ratio of the amount of energy presently stored in the battery to the nominal rated capacity. For example, for a battery at 80% SOC and with a 500 Ah capacity, the energy stored in the battery is 400 Ah. A common way to measure the BSOC is to measure the voltage of the battery and compare this to the voltage of a fully charged battery. However, as the battery voltage depends on temperature as well the state of charge of the battery, this measurement provides only a rough idea of battery state of charge.

3.2 Battery State

Knowing the amount of energy left in a battery compared with the energy it had when it was full gives the user an indication of how much longer a battery will continue to perform before it needs recharging. Using the analogy of a deep cycle battery, State of Charge (SOC) estimation is often called the "Gas Gauge" or "Fuel Gauge" function. The SOC is defined as the available capacity expressed as a percentage of some reference, sometimes its rated capacity but more likely its measure in Coulombs, kWh or Ah of the energy left in the battery which would be less confusing.

The preferred SOC reference should be the rated capacity of a new cell rather than the current capacity of the cell. This is because the cell capacity gradually reduces as the cell ages. For example, towards the end of the cell's life its actual capacity will be approaching only 80% of its rated capacity and in this case, even if the cell were fully charged, its SOC would only be 80% of its rated capacity. A fully charged cell, nearing the end of its life, could have an SOC of 100% but it would only have an effective capacity of 80% of its rated capacity and adjustment factors would have to be applied to the estimated capacity to compare it to its rated new capacity. Using the current capacity rather than the rated capacity is usually a design shortcut or compromise to avoid the complexity of determining and allowing for the age related capacity adjustments which are conveniently ignored.

3.3 SOC Accuracy Requirements

Battery management systems (BMS) are used to keep the battery within a safe operating window and to batteries need very precise control of the SOC for efficient and safe management of the energy flows. In EV applications the SOC is used to determine range. It should be an absolute value based on capacity of the battery when new not a percentage of current capacity which could result in an error of 20% or more due to battery ageing. Automotive fuel gauges are notoriously imprecise so an SOC accuracy of 5%, if it could be achieved, would probably be satisfactory for such applications.

3.4 WHY IS BATTERY MAINTANANCE SO IMPORTANT?

Using batteries in any purpose, the likelihood is we aren't getting what the lead acid battery is capable of. This is because without systematic lead acid battery maintenance, the chemical

change inside the battery results in a buildup of lead sulphate crystals. Monitoring is the best way for maintenance of the batteries that are being charged in the central solar battery charging station. Thereby we developed a software which is enable to monitor the batteries simultaneously and able to detect different condition of the batteries. In this way we are benefitted as compared to the manual maintenance of battery as mentioned below:

- To ensure effective lead acid battery maintenance for manual maintenance we have to follow the procedure mentioned here as example:

Disengage battery ⇨ check gravity with hydrometer ⇨ apply relevant chemicals
re-engage battery ⇨ repeat every 3 months

In recent times, the only proven ways to ensure battery maintenance, involved a lot of manual effort and potentially dangerous procedures being carried out up to four times a year. But while we are monitoring the system, less chances of failure is there as our software is tracking all possible changes of the batteries. So it is better to use monitoring device rather than manual maintenance. For battery maintenance using our software we need not to worry about any of these. What we need to do it only sitting in front of the monitor and check their status and handle them accordingly. Therefore before charging and monitoring we have to consider all the explained issues sensitively.

CHAPTER 4

BATTERY AND SOLAR PANEL MANAGEMENT SYSTEM

4.1 Techniques of Experiment and Result

As we are dealing with the software part, the main motto is to make the whole system more efficient and environment friendly too. Interference with the hardware part is made so that we can read the different state of battery properly using the charge controller to monitor their status. We have used USB ADVANTECH 4716 DAQ card to perform the interference via computer.

The following techniques can be successfully implemented to read input greater than the capability of DAQ card as below:

- Monitoring of the charging status by measuring voltage level
- Monitoring of the panel status by measuring current

4.1.1 Monitoring of the charging status by measuring voltage

Figure 4.1.1 shows the technique of monitoring the charging status of the battery by measuring the voltage. Battery terminals are connected with a voltage divider of 10K resistors to ensure safety of the DAQ card. For example when battery voltage is 12.7Volts, voltage divider gives 6.35Volts which is half of the battery voltage and it is fed to the DAQ card which ensures safety of DAQ card.

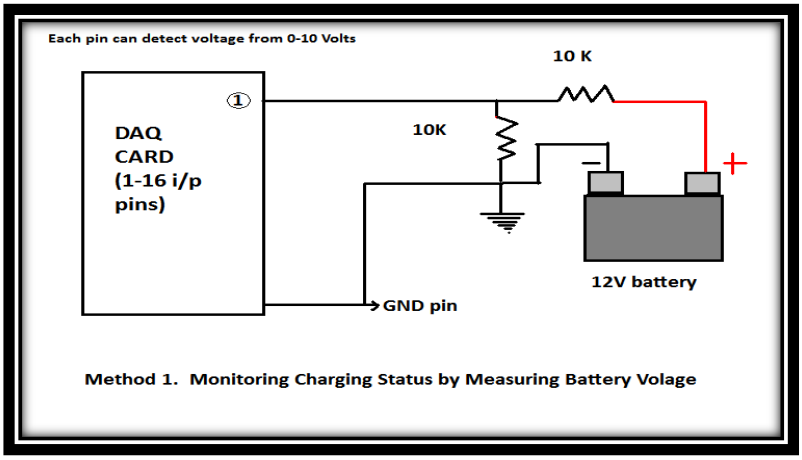


Fig 4.1.1: Monitoring of the charging status by measuring voltage

4.1.2 Monitoring of the PV Array Current Status

This method is applied as it ensures the panel performance. It is an efficient method as it checks individual panel performance with no or less human effort. Current Shunt The simplest method of determining the current is by measuring the voltage drop across a low ohmic value, high precision, series, sense resistor between the battery and the load known as a current shunt. This method of measuring current causes a slight power loss in the current path and also heats up the battery and is inaccurate for low currents. Due to time constraint, we couldn't implement this design but we will hope to do it in near future.

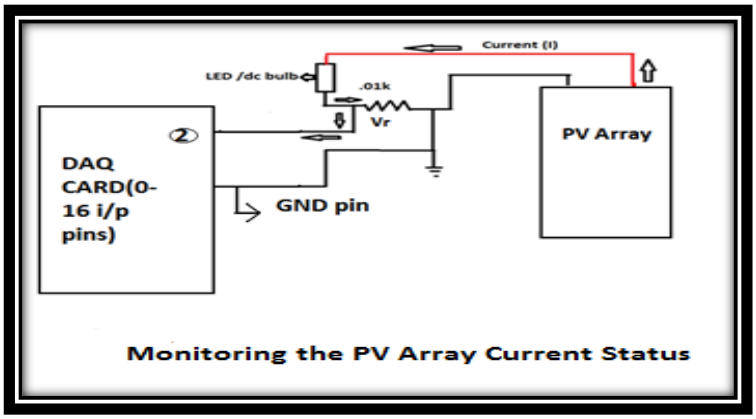


Fig 4.1.2: Monitoring the PV Array Current Status

4.2 Battery Condition and State of Charge Charts

To design the software for monitoring of the Central Solar Battery Charging Station, the battery condition and corresponding state of charge is collected from formerly used batteries for solar system. The following chart represents a clear idea about automotive battery condition that are generally used including charging and discharging process:

STATE OF CHARGE	12 V BATTERY
20%	11.58
30%	11.75
40%	11.9
50%	12.06
60%	12.20
70%	12.32
80%	12.42
90%	12.5
100%	12.7

Table 4.2 State of Charging correspond to voltage level

CHAPTER 5

DATA ACQUISITION SYSTEM

A computer-based data-acquisition system to monitor and control photovoltaic power generation systems using a novel method, based on general data-acquisition card and programming software has been designed and implemented. Prior to designing the data-acquisition system, a small-sized PV power generation system, consisting of Solar panel, batteries, a charge controller and a DAQ, has been assembled. 10K resistors have been utilized by voltage division method method to measure the amount of irradiance incident on the DAQ. If the power generation system is not producing the amount of power that it should, based on the readings we are using, the program would track the battery state to the user to inform him of the condition. For interfacing purpose, we need to use a Data Acquisition Card (DAQ). Its analog input Channels are for voltage measurements whose dynamic ranges can be set independently within the maximum ranges of ±10V. Since this range is lower than the voltages that are to be measured, external signal conditioning, based on voltage dividers and current shunts, is required to condition the measured values to fall within the required range.

5.1 EMBEDDED SYSTEM DESIGN

Embedded Systems – a combination of hardware and software for specific purposes – are in widespread applications in consumer electronics or communication products. Software development in this area concentrates on the delivery of creative, secure, and reliable functionality to improve the working procedure for the end users.

In our case, the system is a embedded one. It consists of hardware and software part as all other embedded system has and both parts are equally important in this project. Hardware part includes the data acquisition system and primary signal processing circuit. Solar charge controller that is connected to the DAQ card is a part of the hardware implementation from where we are retrieving the signal. Software system consists of operating system and developed software in windows environment. It is designed to monitor the battery properties, rather than be a general-purpose computer for multiple tasks. It is made based on real-time performance constraints that must be met, for reasons such as safety and usability; others may

have low or no performance requirements, allowing the system hardware to be simplified to reduce costs. The contract between the hardware and software system is given below:

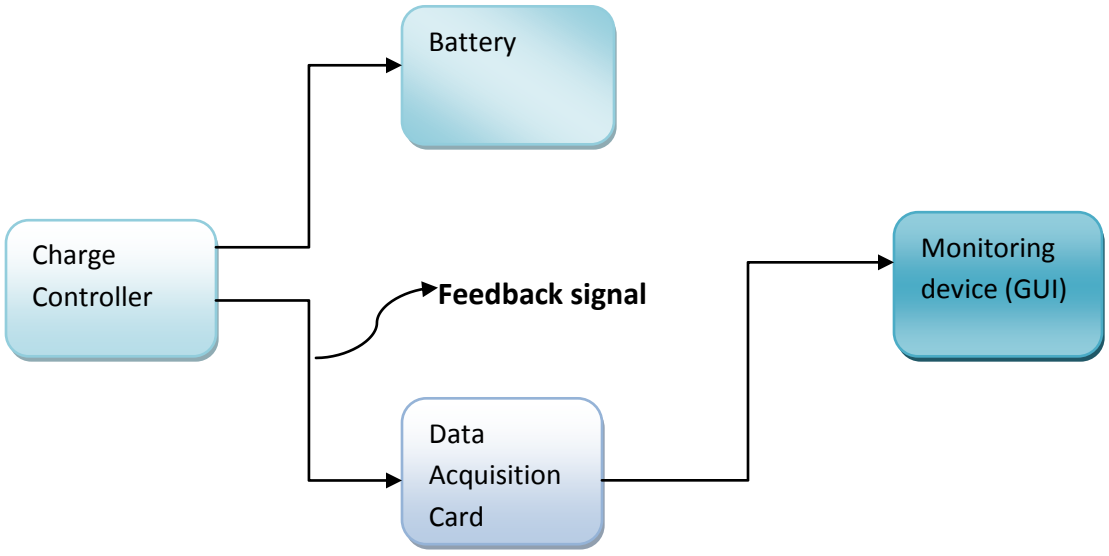


Figure 5.1: Signal flow into the software part

5.2 USB-4716 SPECIFICATIONS

The data acquisition board we are using for signal generation and communicating with computer is a DAQ from Advantech Company model no USB-4716. The USB-4716 is a true Plug & Play data acquisition device. No need to opening up the computer chassis, it is just needed to use the USB port for data acquisition. USB-4716 has 16 single-ended/ 8 differential inputs with 16-bit resolution, up to 200 kS/s throughput, 16 digital I/O lines and 1 user counter, add two 16-bit analog outputs [12]. It obtains all required power from the USB port, so no external power connection is ever required. The features of the USB-4716 are given along with the manual of this product attached in the APPENDIX section.

Figure 5.2: USB-4716

5.3 DATA ACQUISITION SYSTEM

Data acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. Data acquisition systems (abbreviated with the acronym DAS or DAQ) typically convert analog waveforms into digital values for processing. Graphical programming environments include ladder logic, Visual C++, Visual Basic, and Lab VIEW. For transmission purposes, single ended analog signals, which are more susceptible to noise, can be converted to differential signals.

5.4 IMPORTANCE OF DATA ACQUISITION CARD

The need of recording the measurements and process, the data collected for visualization has become increasingly important. There are several ways in which the data can be exchanged between instruments and a computer. Many instruments have a serial port which can exchange data to and from a computer or another instrument. Another way to measure signals and transfer the data into a computer is by using a Data Acquisition board

- ✓ Sampling: The rate at which the signal is sampled is known as sampling frequency. The sampling frequency determines the quality of the analog signal that is converted. Higher

sampling frequency achieves better conversion of the analog signals. Minimum sampling frequency required to represent the signal should at least be twice the maximum frequency of the analog signal under test (Nyquist rate).

- ✓ ADC: Once the signal has been sampled, we need to convert the analog samples into a digital code. This process is called analog to digital conversion. This is shown in most boards also have a multiplexer that acts a like a switch between different channels and the ADC. Therefore with one ADC, it is possible to have a multichannel input DAQ board. The DAQ board we are using has 16 channel analog inputs. This makes it possible to acquire up to 16 analog signals in parallel (however, the sampling frequency will be divided by the number of parallel channels).

- ✓ Resolution: Precision of the analog input signal converted into digital format is dependent upon the number of bits the ADC uses. The resolution of the converted signal is a function of the number of bits the ADC uses to represents the digital data. The higher the resolution, the higher the number of divisions the voltage range is broken into, and therefore, the smaller the detectable voltage changes. An 8 bit ADC gives 256 levels (2^8) compared to a 12 bit ADC that has 4096 levels (2^12). Hence, 12 bit ADC will be able to detect smaller increments of the input signals then a 8 bit ADC. If the full scale of the input signal is 10V than the LSB for a 3-bit ADC corresponds to 10/2^3=1.25V. That is not very good! However, for a 12 bit ADC the least significant bit will be 10/2^12=10/4096=2.44mV. If we need to detect smaller changes, one has to use a higher resolution ADC. Clearly, the resolution is an important characteristic of the DAQ board.

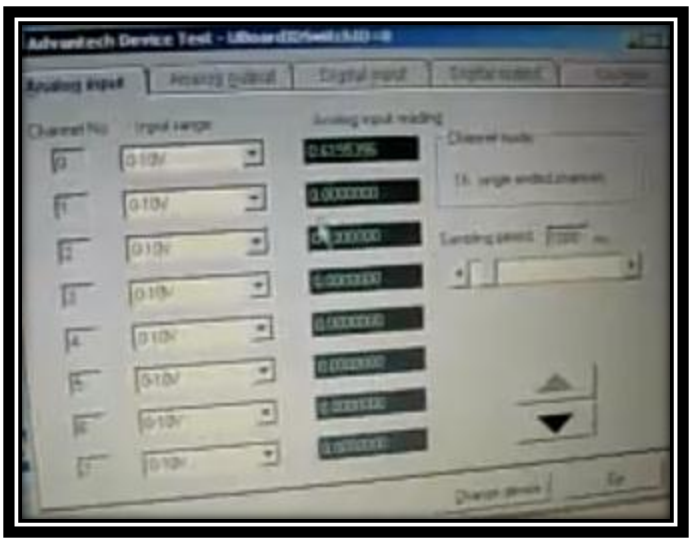
CHAPTER 6

INTERFACING IN BETWEEN SOFTWARE AND HARDWARE IN WINDOWS ENVIORNMENT

6.1 REASONS BEHIND CHOOSING WINDOWS

In our project for interfacing, we have used windows 2007 version as it is not only updated with recent modifications but also the GUI made under windows environment is able to operate in any operating system efficiently with a supporting .NET framework than the GUI made in LINUX or any other environment.

6.2 COMPARISON WITH EXISTING DATA ACQUISITION CARD SOFTWARE



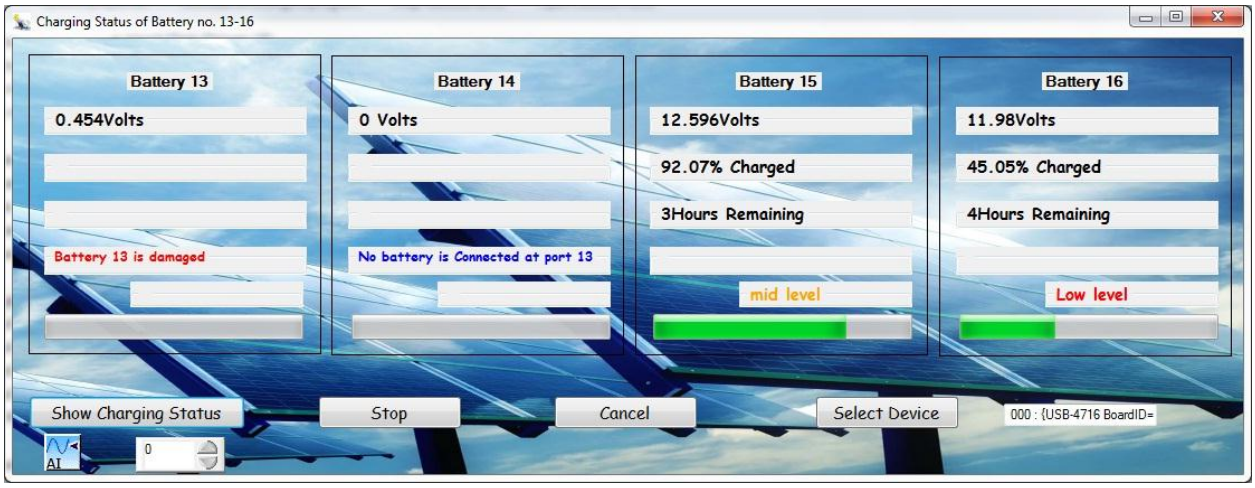


Fig 6.2 Built in Software of Advantech USB-4716 vs Newly Built Software

Fig 6.2 shows the software provided by DAQ card Advantech USB 4716 and the newly built software of ours sequentially. It has fewer features as it shows only voltage level of the batteries that is not sufficient to monitor the status of the battery.

6.3 REASONS BEHIND NOT USING Device manager

6.3.1 Device Manager Advantech USB 4716

Firstly, the device manager shows the voltage of the input only. As the channel input of the DAQ card is not the exact voltage of the battery according to the circuit design we used, it is quite meaningless to monitor the battery charging status using the Advantech GUI.

Secondly, Advantech device manager comes with only one feature and that is showing the voltage reading from each channel. Whereas our software has more attractive features to monitor the central solar battery charging status such as it has a primary battery selection panel which redirects to other GUIs (we have built four GUIs as Advantech comes with 16 analog channels only) each having four battery charging status each up to 20 batteries all together. Each of the GUIs shows battery voltage, percentage charged, time remaining to get fully charged, battery levels, battery condition and a visual representation of battery charging in progress bar.

6.3.2 WHY WE CREATED A NEW GUI USING THE EXISTING?

Instead of using the built in GUI we have made a new one using the existing .dll functions provided by the 'ACTIVE DAQ PRO'. The main reason is the limitations of that GUI which we had to overcome to make a sophisticated one which can perform multi functions simultaneously for the monitoring purpose to make the whole job of monitoring easier.

The interfacing is done between the DAQ card and the computer software. The data acquisition card USB-4716 provides us a device driver that can give different functionality of

system. The device driver software named ActiveDAQ Pro gives us different function to fetch data from DAQ card and represent the data in our software [3]. We have used the device control functions to manipulate the data coming through the DAQ card .We integrated the device control function to our GUI to control the data coming from the DAQ card.

6.4 DEVICE DRIVERS

The interfacing is done between the DAQ system and the computer software. The data acquisition card USB-4716 provides us a device driver that can give different functionality of system. The device driver software named ActiveDAQ Pro gives us different function to use the DAQ system and represent the data [14]. We have used the device control functions to manipulate the data coming through the DAQ card .We integrated the device control function to our GUI to control the data coming from the DAQ card. The device control functions categories are given below:

- Analog Input Control (AdvAI)- The control supports functions relating to AI: single channel or multi-channel sampling and fast speed sampling, such as interrupt or DMA transfer. Different from the previously released AI control, AdvAI is an independent control that can be directly used without the assistance of any other control.
- ADEVEDIT- Advantech ActiveDAQ Pro Number and Editor Control
- AxAdvAI- Advantech ActiveDAQ Pro AI Control
- AxADEVEDIT- Advantech ActiveDAQ Pro Number Editor and LED Controls
- Analog Output Control (AdvAO)-Supports all the functions related to analog output, including single data or quantity data high speed sampling.

6.5 PROPERTIES OF DIFFERENT FUNCTIONS

.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.

Table 6.6: Properties of .dll functions

6.6 HOW TO INTERFACE?

Interfacing between the DAQ card and our software that we built is most important part in the project. Microsoft Visual Studio that has been used for deployment process provides a utility to run the program just the way it performs when it is deployed. The card takes the data from the charge controller and sends the data to the computer. The computer gets a digital data and software takes the responsibility for further processing of the data and shows it in specific manner. It is versatile to ensure the communication between the software and the card thereby. The steps of performing the interfacing are given below:

- Although we are not using device manager provided by the manufacturer company Advantech, we have to make sure that it is working properly as we are not re-writing the built in GUI, we are just grabbing necessary signals from the built in GUI.

- Therefore, first step is to install the Advantech device drivers to make sure that the windows will recognize the hardware. It just recognizes the hardware and creates communication with the developed software.
- Choose the necessary .dll functions needed to process the signal via our newly built GUI using the provided one.
- Needs the analog signal processing function AdvAI. As we used C# language for the graphical user interface we added the specified functions in C# development environment as reference. Here we are using VISUAL STUDIO 2010 version.
- Afterwards select a device by calling SelectDevice() function. It makes sure we are using the correct version of the product which is for us USB-4716.
- Then needs to select device name and device number by using device name and device number properties
- Lastly using Data analog properties to control analog input data coming from the Central Solar Battery Charging Station.
- After getting the analog input data software processes it as needed then represents it visually using the GUIs which we converted into individual software afterwards.

6.7 GRAPHICAL USER INTERFACE (GUI)

Today's major operating systems provide a graphical user interface. Applications typically use the elements of the GUI that come with the operating system and add their own graphical user interface elements and ideas. A GUI sometimes uses one or more metaphors for objects familiar in real life, such as the desktop, the view through a window, or the physical layout in a building. Elements of a GUI include such things as: windows, pull-down menus, buttons, scroll bars, iconic images, wizards, the mouse, and no doubt many things that haven't been invented yet. Graphical user interface (GUI) is essential to represent the charging status of every four batteries of the SBCS in one form. The GUIs have been developed in visual studio 2010 by using C# language. The GUIs represent the data those are obtained by the software. The GUIs have different options which were discussed earlier section. We can use those options by using the buttons, textbox, groupbox, progress bar, timer etc given in the GUI. We have developed codes

for the primary GUI- “Battery Selection Panel” and also for the other GUIs redirected by the primary GUI depending on user’s selection. In Appendix A the codes are given.

6.8 CHARACTERISTICS OF GUIs

The primary GUI has multiple GUIs programmed under it. The GUI has been programmed to show the charging status of 16 batteries.

The GUIs built to show the charging statuses can measure the charging status of the batteries in voltage, time remaining to get fully charged, percentage charged. It also shows the voltage level (Low, medium and high), indicates whether no battery is connected at the port (when voltage detected is 0Volt) or battery is damaged or fully charged or has been disconnected from the charge controller (when battery is charged up to 14.4 Volts).
Five buttons have been added that will direct the user to different windows, each having four different battery statuses as shown in Figure 4. When clicked on “Show battery #number-#number” (For example “Show battery 1-4”), a window is popped up having four slots containing four battery charging status. In this way our GUI is able to monitor multiple batteries under the same window simultaneously. A pop up window containing four slots of battery charging status includes a ‘*Select Device*’ button which will select ADVANTECH USB 4716 and will fetch the analog data. Another feature is ‘*Show Charging Status*’ which will begin the process. A ‘Stop’ button is used to halt the system and its stops fetching data to the GUI thus freezing the previously collected data and finally a ‘Cancel’ button cross out the pop window of the GUI. To show the user that the process visually, a progress bar is used.

Charging Status is showed using following information and windows components:

- Voltage of the battery connected.
- Charging Status in percentage
- How much time is remaining to get it fully charged (considering the panel current to be 3A for our case)
- The battery condition indicator: Flashes the important conditions to make the user alert. Each GUI indicates the battery conditions as for example:

- Battery is not connected at port X No. , Battery is damaged, Battery is fully charged and Battery is auto-disconnected.
    - When any battery voltage is 0V- the condition is: Battery is not connected at port #no.
    - When Voltage is less than 11.6Volts –the condition is: Battery is damaged.
    - When Voltage is equal or greater than 12.7Volts-the condition is: Battery is fully charged.
    - When Voltage is 14.4 Volts- the condition is: Battery is disconnected.
- Progress bar: shows the progress of the charging visually.
- Level indicator: indicates different levels of charging: Low, medium and high.

Different use of colors in the GUI provides improved visualization.

CHAPTER 7

RETREIVING SIGNAL WITH VISUAL STUDIO

7.1 OVERVIEW AND FEATURES OF VISUAL STUDIO 2010

As we are using visual studio 2010 version and c# for programming we need to know the reason behind choosing this version of visual studio. Visual Studio 2010 is for developers who need to do application design, development, and testing. In Visual Studio 2010's simplified tier structure, each tier — aside from Test Professional (which is the only role-specific version left) — adds features to the previous tier.

Title	Description of the new features
What's New in Visual C# 2010	Describes new features in the C# language and Code Editor. The features include the dynamic type, named and optional arguments, enhanced Office programmability, and variance
Getting Started with Visual Studio	Describes how you can become familiar with this version of Visual Studio, whether you have used other versions or are new to the product
What's New in the .NET Framework 4	Contains information about key features and improvements in the .NET Framework 4.
Quick Tour of the Integrated Development Environment	Provides a brief overview of many of the application development features and tools that are included in Visual Studio.

What's New in the Visual Studio 2010 Editor Title	Describes the new and enhanced features in the Visual Studio editor. Description of the new features
What's New in Deployment	Describes how you can deploy applications and the latest version of the .NET Framework as a prerequisite in your Click Once or Windows Installer deployment packages. You can also deploy your applications by using Install Shield 2010 Limited Edition.
Installing and Managing Visual Studio Tools and Extensions	Provides an overview of Extension Manager, which you can use to install a variety of tools and extensions for Visual Studio.

Table 7.1: Features of VISUAL STUDIO 2010

7.2 RETRIEVEING DIFFERENT SIGNAL USING EQUATION FROM THE GUI

The next and the most important job of this software designing is to retrieve signal physically from batteries to test whether the software is working well enough to get desired level of error as expected. Therefore to test the software we have to interface the newly built software of ours with the charge controller that is connected to at least one battery via ADVANTECH USB 4716 to measure the performance. It seems quite easy but actually it is the toughest part of this project.

The condition we used for different level of the batteries is mentioned below (say, for channel 1):

- When channel input voltage = 0 Volts that means there is no battery connected to the specified channel.

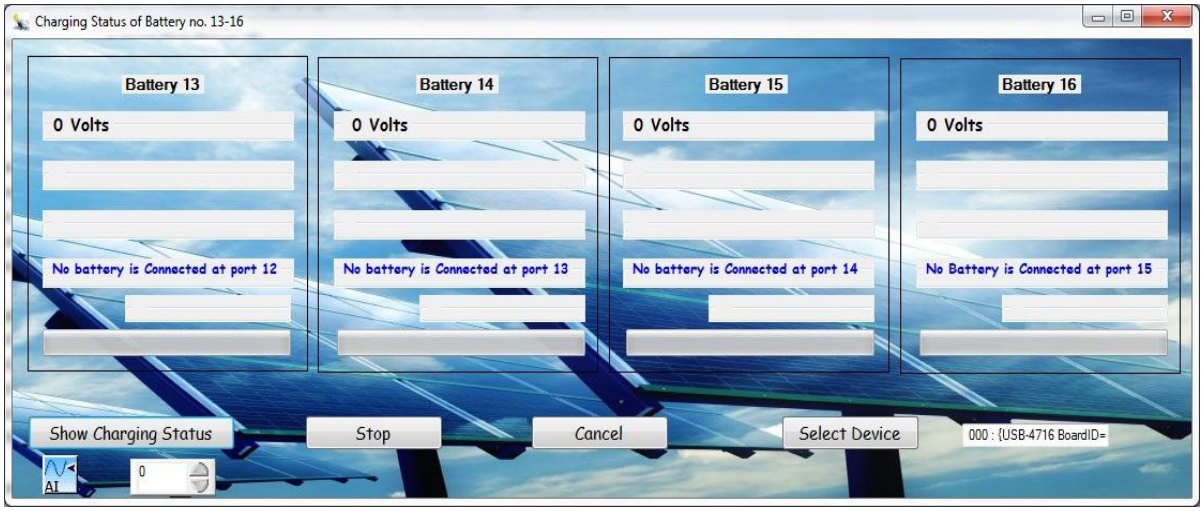


Fig 7.2.1: "NO BATTERIES IS CONNECTED AT PORT 12,13,14 and 15"

- When battery voltage is in greater than or equal to 11.6 but less or equal to 12.534 Volts, the level is denoted as 'Low level' and if voltage is less than 11.6 Volts then the battery condition is "Battery #no is damaged".

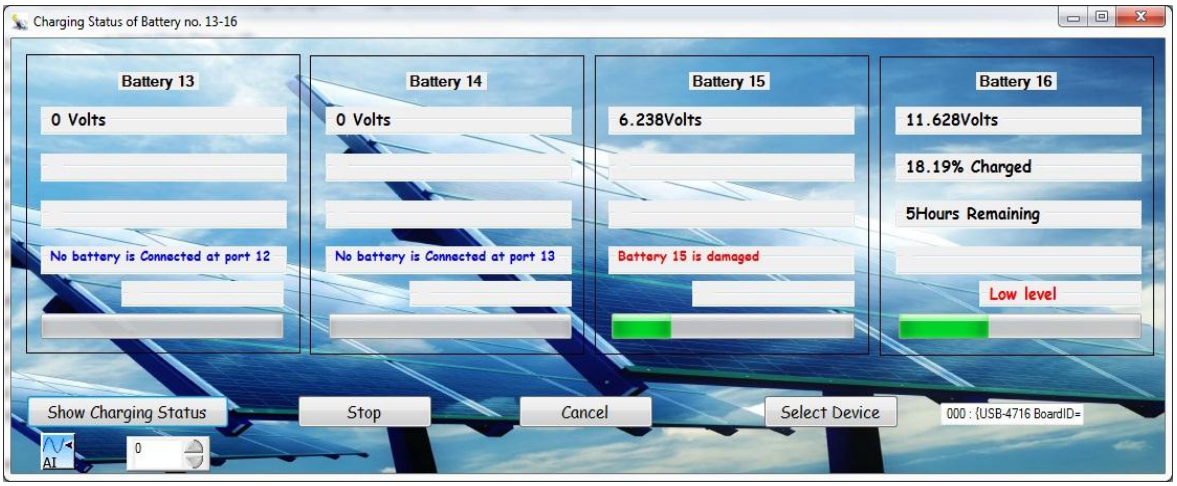


Fig 7.2.2: "BATTERY 16 IS AT LOW LEVEL"

- When battery voltage is in greater than or equal to 12.534 but less or equal to 12.6 Volts, the level is denoted as ‘Mid level’

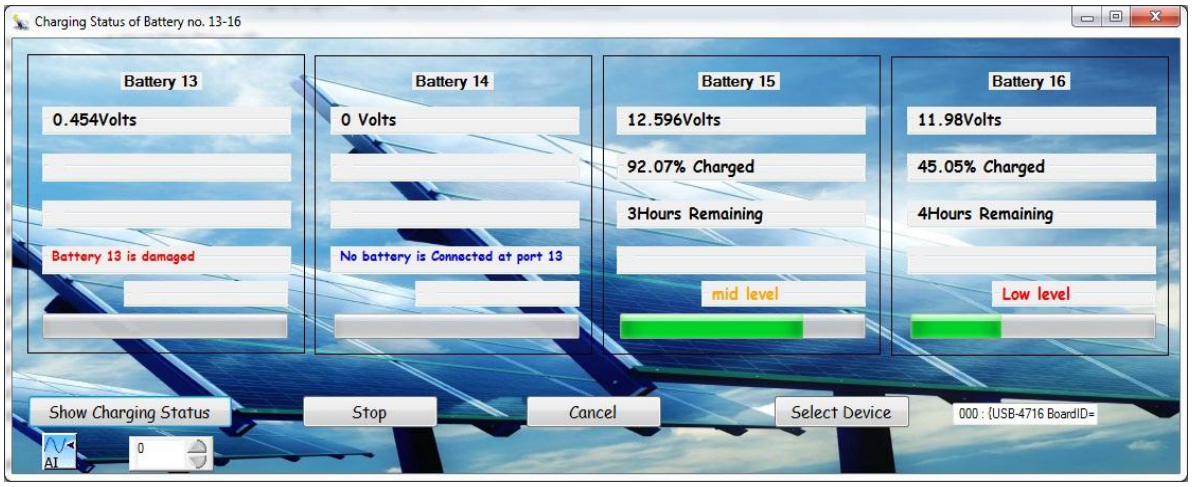


Fig 7.2.3: “BATTERY 15 IS AT MID LEVEL”

- When battery voltage is in greater than or equal 12.6 Volts, the level is denoted as ‘High level.

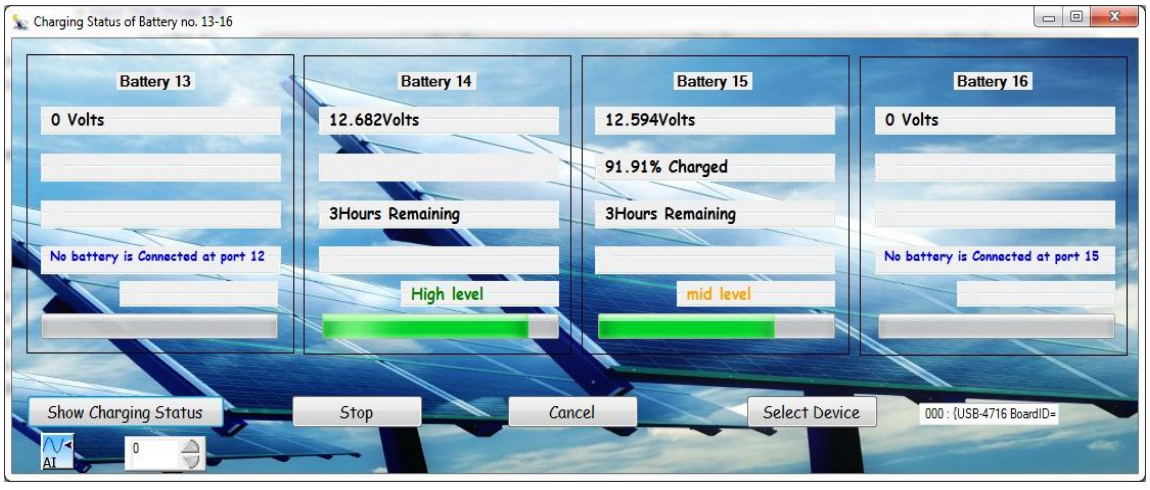
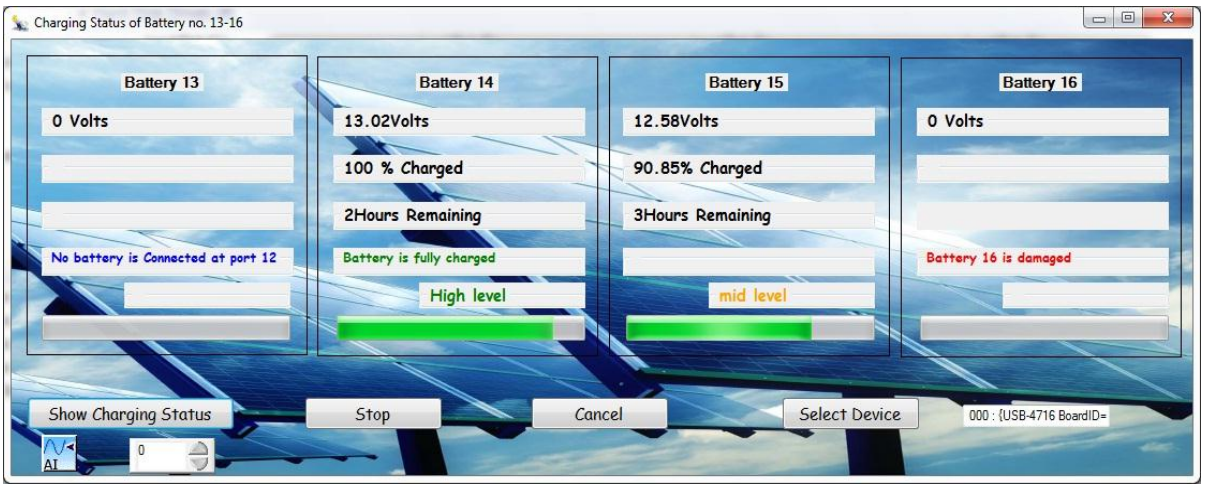


Fig 7.2.4: “BATTERY 14 is AT HIGH LEVEL”

- When battery voltage is in greater than or equal 12.7 Volts, the battery condition is" battery is fully charged"



. Fig 7.2.5: "BATTERY 14 IS FULLY CHARGED"

- When battery voltage is in greater than or equal 14.4 Volts, the battery condition is " battery is disconnected".

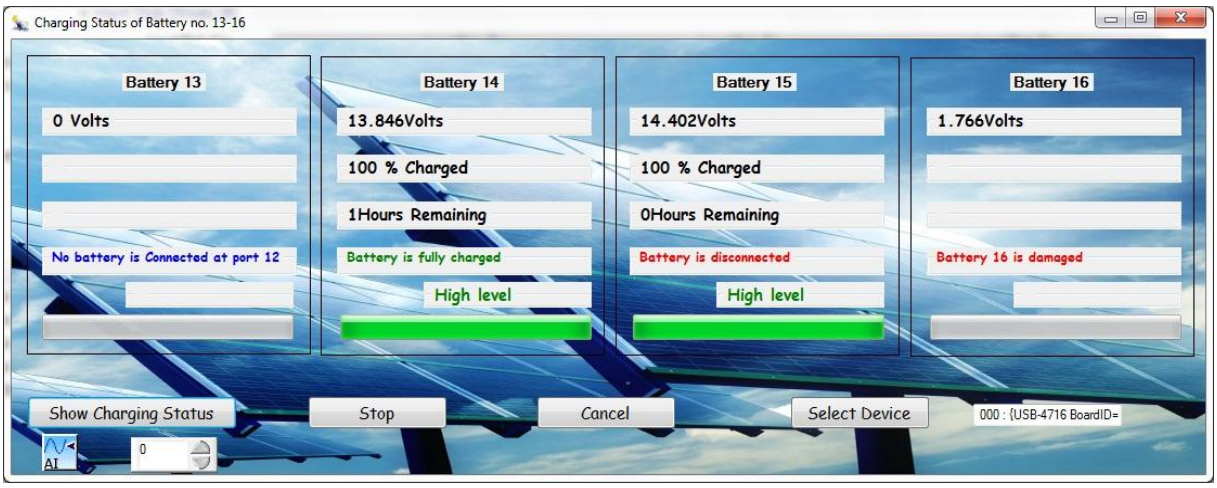


Fig 7.2.6: "BATTERY 15 IS DISCONNECTED"

Again we have used the following equations to find out the properties of the batteries using our GUI or the monitoring software that we built. Those are mentioned below:

7.2.1 MEASURING BATTERY VOLTAGE

As we have mentioned that we have used equation to measure battery voltage corresponding to different level of battery in charging condition which has been measured from real life interfacing. The corresponding equation that we have used stands for the data which is coming through channel X at a particular time and is rounded up to 3 precision points to get almost desired value that matches with experimental readings.

double channel1 = Math.Round(axAdvAI1.DataAnalog,3)

7.2.2 MEASURING PERCENTAGE CHARGED

We found out an equation that is Y = (.7632 * (X*2) - 8.7197) * 100 + 2.71, where Y =percentage charged and X stand for battery voltage to measure percentage charged of the battery from the following relationship of battery voltage and percentage[22]:

STATE OF CHARGE	12 V BATTERY
20%	11.58
30%	11.75
40%	11.9
50%	12.06
60%	12.20
70%	12.32
80%	12.42
90%	12.5
100%	12.7

Table: 7.2.2: Relationship between percentage charged and battery voltage[22]

7.2.3 MEASURING TIME REMAINING TO GET BATTERY FULLY CHARGED

To design efficient software that reduces the cost of manual monitoring of battery maintenance, we have used time function to calculate the time remaining for the battery to be fully charged using the equation below:

Time remaining to get fully charged: {80 Amp - (80 Amp/14.4Volts)*n volts}/3Amp;

Considering a constant panel current of 3 Amperes at 14.4V HVD [HIGH VOLTAGE DISCONNECT], n=battery voltage.

7.3 VISUALIZING CHARGING STATE OF BATTERY USING Progress Bar

The Windows Forms ProgressBar control indicates the progress of an action by displaying an appropriate number of rectangles arranged in a horizontal bar. When the action is complete, the bar is filled. To show the progress of charging of individual battery we have designed the progress bar which gradually changes with the change in the increasing voltage and for our case we denote the progress bar as p1, p1……p16 in our software which flashes gradually with the increasing or decreasing progress. It starts showing the charging progress from very beginning, also when any battery is damaged and goes on flashing till the battery get fully charged14.4 V.

7.4 IDENTIFIYING BATTERY STATE USING DIFFERENT COLOURS FROM THE GUI

In our newly built GUI we have used individual colors corresponding to different state of charging as mentioned below:

State of battery charging	Specified color for that level
No battery is Connected at port x	Blue
Battery x is damaged	Red
Low level	Red
Mid level	Orange
High level	Green
Battery is fully charged	Green
Battery is disconnected	Red

Table 7.4: Identifying battery state using different colors from the GUI

7.5 .NET FRAMEWORK4

The .NET Framework is an application development platform that provides services for building, deploying, and running desktop, web, and phone applications and web services. It consists of two major components: the common language runtime (CLR), which provides memory management and other system services, and an extensive class library, which includes tested, reusable code for all major areas of application development. .net framework 4 is a great platform creating by Microsoft for developing applications for windows environment. An application developed by .net framework can be run in any version of windows. The advantages of using .net framework are lot. We can develop the software in any language that supports .net such as visual basic++, c# etc. Another reason of using it is its drag and drop design option that is definitely an easier choice of any kind of software development. It has a gigantic library of code that we can use for different development. It also has a huge collection of device driver function. We have used C# language for our application development because it supports the

object oriented programming module. The .NET Framework 4 is highly compatible with applications that are built Visual Studio 2010 that improves security, standards compliance, correctness, reliability, and performance. The .NET Framework 4 does not automatically use its version of the common language runtime to run applications that are built with earlier versions of the .NET Framework. To run older applications with .NET Framework 4, you must compile your application with the target .NET Framework version specified in the properties for your project in Visual Studio, or you can specify the supported runtime with the <supported Runtime> Element in an application configuration file.

- Visual Studio 2010.NET and C# language features, such as implicit line continuations, dynamic dispatch, named parameters, and optional parameters.
- Support for Code Contracts.
- Inclusion of new types to work with arbitrary-precision arithmetic (System.Numerics.BigInteger) and complex numbers (System.Numerics.Complex).

7.6 DEPLOYEMENT OF THE SOFTWARE FOR MONITORING OF SBCS

Moving completed applications to other computers is generally referred to as deployment. The Microsoft development environment provides mechanisms for deployment; for more information, see Deploying Applications and Components. If one build and distribute mainly from the command line, he might need to consider other methods of deployment and redistributing dependencies. The procedure for deployment of the software is given below:

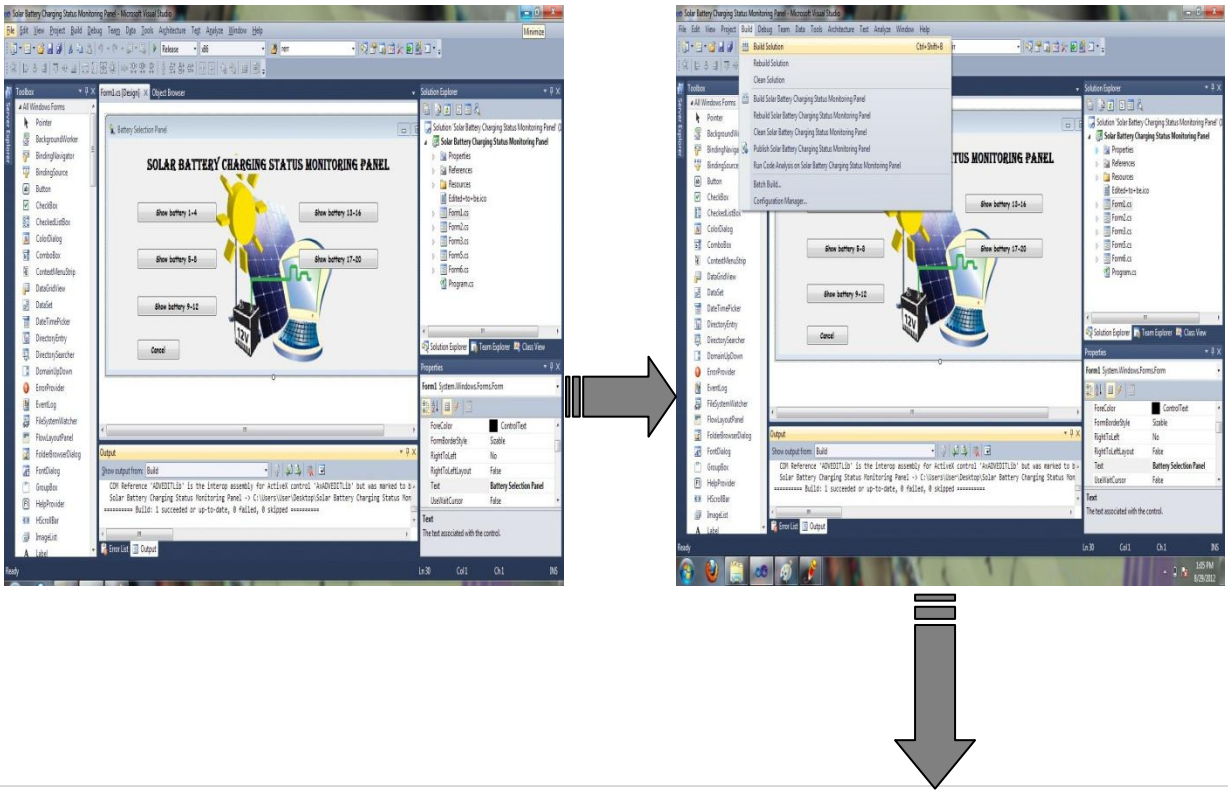
- Setting solution configuration in release mode
- Building of the solution
- Creating of a new project
    - ✓ Under the setup and deployment section in the visual studio installer selecting a setup wizard; we named it SBCS setup.msi to create a setup file for windows application

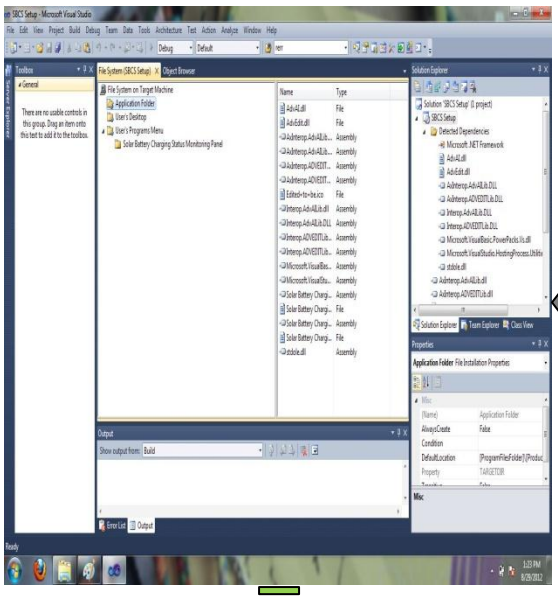
- ✓ Adding of the files from visual studio project (from release folder)
- ✓ In the application folder creating two shortcuts of “Solar Battery Charging Status Monitoring Panel” to the user desktop and to the user program menu
- ✓ Selecting a new file type-named it as SBCS -file extension (.far), command is “solar battery charging Status Monitoring Panel.exe”. We have also set an icon for our software here and also for the shortcuts in the application folder
- ✓ Building of the SBCS setup

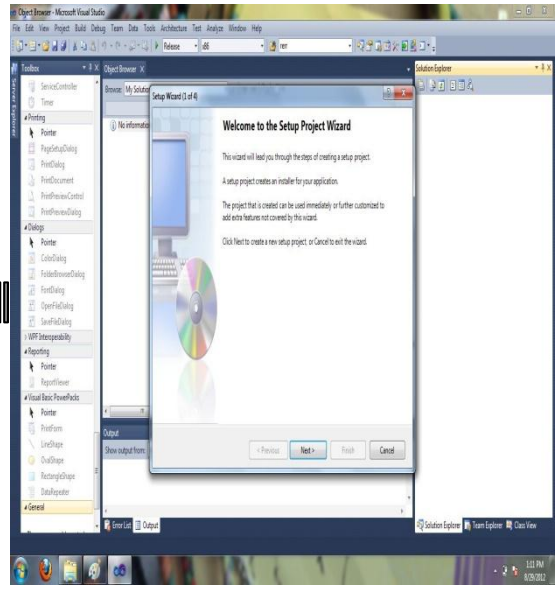
- Setting to the configuration to release
- Building the batch
  - ✓ Selecting both the debug and release configuration
  - ✓ Hitting the build button
    - Lastly installing of the setup file from the “SBCS setup” folder.

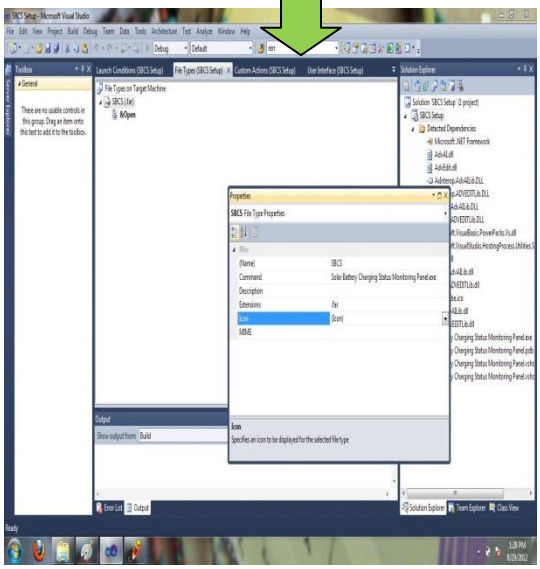
Here we named the file type as SBCS, for command of the file we used .exe. Also we used a new type of file extension .far for our software which is not used for any other application till now.

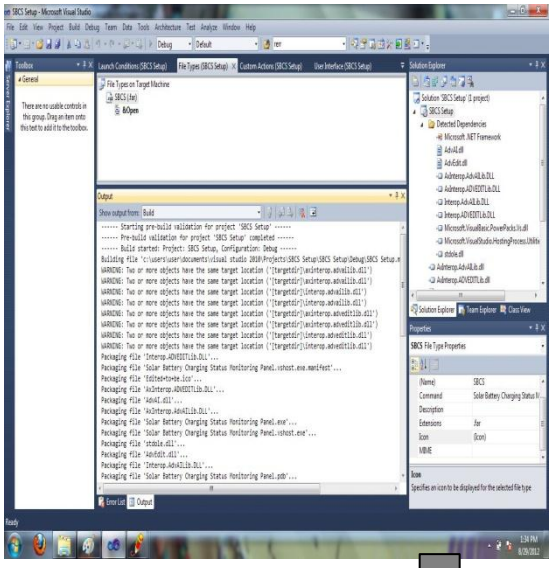
`






Welcome to the Setup Project Wizard





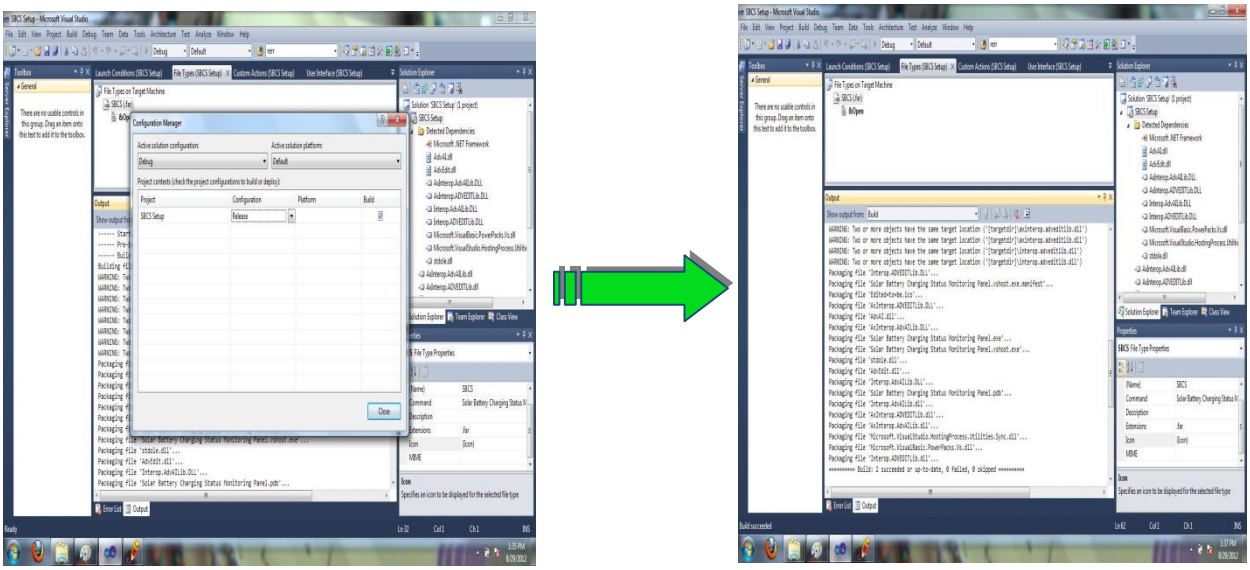


Fig 7.6 Details of Deployment

7.7 ICON OF THE SOFTWARE

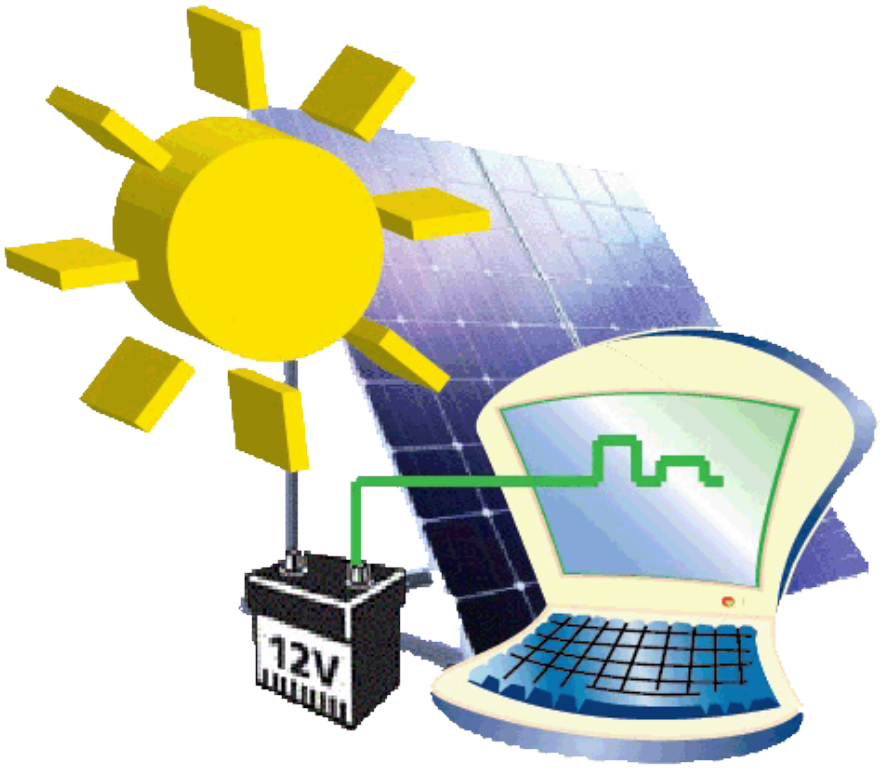


Fig. 7.7 : .ico file that has been used as the logo of the software

CHAPTER 8

CONCLUSION AND FUTURE WORKS

The emergent need for electricity has led to a countrywide propagation of solar energy based electricity generation systems that integrate battery storage through the use of Solar Home Systems (SHSs) and a large portion of the country's population is dependent on a strenuous means of livelihood that is rickshaw (tricycle) pulling [18]. To tackle the problem, implementation of Solar Battery Charging Station (SBCS) has emerged to the rural Bangladesh as well as in urban areas to change the scenario. Thereby, software implementation of SBCS is vitally important to monitor the system and keep the batteries safe. While maintaining the batteries of the SBCS manually, there might occurs mistakes and batteries can get overcharged. But doing it using software is not only safe but also time and cost effective. Thereby our motto is to make the cost-effective software for monitoring the station from remote region even-though. With the completion of our GUI we will be able to screen multiple batteries concurrently under the same monitor and will allow for the real time visualization of all types of readings, such as the voltage and percentage charge of each battery.

8.1 SUCCESS OF THE HAND-SHAKING PROJECT

Our present work can be counted as a successful one as we have fruitfully deployed the GUI of ours that can smoothly make communication for monitoring battery status, which we initially targeted. Though, our entire goal is not done yet due to time constraint. Some of the important aspects of our success are mentioned below:

- Screening multiple batteries concurrently in the same monitor
- The main GUI has multiple GUIs programmed under it
- Showing battery status with multi functions; say voltage level that changes gradually with charging, charging status, level indicator etc.
- Again our GUI has got individual progress bar correspond to individual battery to show the charging progress throughout the whole period of charging.

- Development of Data processing in Graphical user interface for central solar battery charging station (SBCS) in windows environment; minimum system required to run our software is windows XP, .net faramework4.
- Interfacing DAQ system in windows 2007

8.2 BOUNDARIES OF THE CURRENT JOB

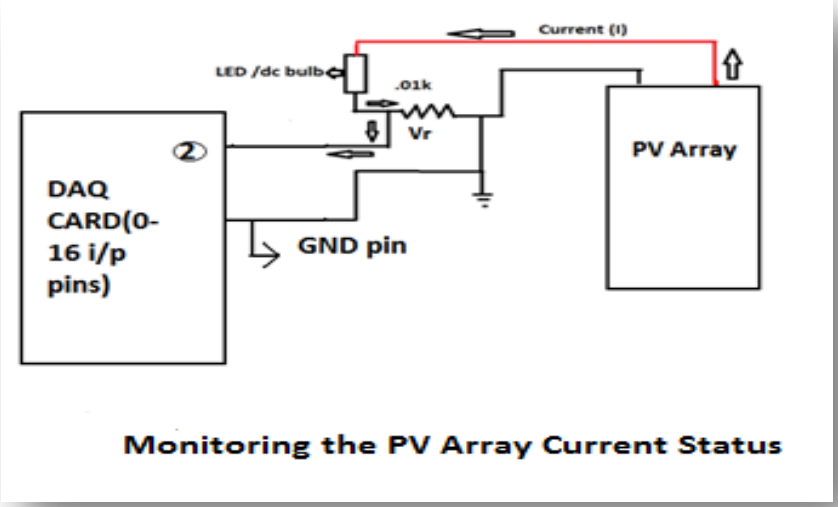
Though we have successfully implemented our project, we also got few boundaries that should be mentioned here along with the success of ours:

- We could interface the DAQ system only with windows environment only.
- During the testing period for performance, our software did not get exactly same value that we got from physical calculations. May be because of instrumental errors all those occurred. Thereby, we added some level of negligible precision points to perform it efficiently in the program we have written.
- Our entire system is mostly reliant on USB-4716. Any other data acquisition system can diminish or enlarge the accuracy level too.
- Again we have considered our maximum current limit as 3A for 12V-80Ah-batteries connected to 75W solar panel. Resultantly our GUI is needed to be changed for monitoring of batteries.
- Again there can be occurrences of backflow of current too and we have not still dealt with this part till now.

8.3 FUTURE WORKS

As we are not done with the whole of our plan as a part of this software implementation, we have to fulfill those in near future. Those are mentioned below:

- ✓ Additional development of out the software in windows by accumulating further options specifically updating for with panel monitoring.
- ✓ Making of log file: In future the software can also be modified with an addition of a Log file that will store the data of the charging status for example the voltage once it start running .
- ✓ Measuring the current of the solar panels to monitor the solar panels as well implementing the following design which we couldn't due to time constraint.



- ✓ Incorporation of alarm management system for well-organized handling of batteries
- ✓ Addition of voltage and current curves for individual batteries.

REFERENCES

1. http://bdoza.wordpress.com/2009/05/11/solar-energy-alternative-source-of-energy-for-bangladesh/
2. http://en.wikipedia.org/wiki/Electricity_sector_in_Bangladesh
3. http://www.advantech.com/products/ActiveDAQ/mod_6BAFEB9C-9227-4F85-87F5-3006403539E1.aspx
4. http://www.advantech.gr/products/ActiveDAQ-Pro/mod_1-2MLC76.aspx.htm
5. URL:http://www.idcol.org/prjshsm2004.php
6. http://www.brac.net/content/brac-solar#.UAzEMXpdAgo
7. http://www.kamworks.com/projects/solar-battery-station.html
8. http://www.msdn.microsoft.com/en-us/library/system.drawing.bitmap.asp
9. http://en.wikipedia.org/wiki/Real_time_computing
10. http://en.wikipedia.org/wiki/Data_acquisition
11. M. Burgess (2002). A Short Introduction to Operating System." December 29, 2002.
12. Web.(http://support.elmark.com.pl/advantech/pdf/iag/USB-4716-manual.pdf)
13. Advantech. "Active DAQ pro user guide."
14. Rachaen M. Huq et al, Thesis on "Development of Torque Sensor Based Electrically Assisted Hybrid Rickshaw," CARG Project, BRAC University
15. Web.(http://www.windsun.com/Batteries/Battery_FAQ.html)
16. Kazi Mohammed Razin et al, Thesis on 'DATA PROCESSING THROUGH BIOSENSORS AND DEVELOPMENT OF SIMULATION SOFTWARE IN WINDOWS & RT-LINUX'
17. Web.(http://en.wikipedia.org/wiki/Software_deployment)
18. Infrastructure Development Company Limited (IDCOL) Bangladesh. URL: http://www.idcol.org/energyProject.php

19. The World Bank Asia Sustainable and Alternative Energy Program ASTAE, URL:http://siteresources.worldbank.org/INTEAPASTAE/Resources/TimorLeste_RuralEenergyPolicy.pdf

20. S. M. Sunny, O. Ahmed et al, “Solar Battery Charging Station (Design and Development of Hardware Part),” CARG Project, BRAC University

21. Advantech USB Manual

22. http://www.windsun.com/Batteries/battery_faQ.htm

APPENDIX

CODES FOR THE SOFTWARE

(BUILT IN VISUAL STUDIO UNDER C# ENVIRONMENT)

Code for the primary GUI :

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

namespace Solar_Battery_Charging_Status_Monitoring_Panel
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }


      private void Form1_Load(object sender, EventArgs e)
      {

      }

      private void button6_Click(object sender, EventArgs e)
      {

         this.Close();
      }

      private void button1_Click(object sender, EventArgs e)
      {
```

```
        Form f2 = new Form2();

        f2.Show();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form f3 = new Form3();
        f3.Show();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        Form f4 = new Form5();
        f4.Show();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Form f5 = new Form6();
        f5.Show();
    }



    }
}
```

Code for the windows Form2 that has been initialized as f2:

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

namespace Solar_Battery_Charging_Status_Monitoring_Panel
{
  public partial class Form2 : Form
  {
    public Form2()
    {
      InitializeComponent();

    }
    private void Form2_Load(object sender, EventArgs e)
    {
```

```
}
private bool set = false;
private double p = 0;
private double current;
private int v;
private int t = 0;
private double pi = 0;
private double current1;
private int v1;
private int t1;
private int i = 0;
private double pii = 0;
private double current2;
private int v2;
private int t2;
private double piii = 0;
private double current3;
private int v3;
private int t3;
private void cmdSelectDevice_Click(object sender, EventArgs e)
{
   // selecting device
   axAdvAI1.SelectDevice();
   txtDeviceName.Text = axAdvAI1.DeviceName;
}
private void timer1_Tick(object sender, EventArgs e)
{
   axAdvAI1.ChannelNow = 0;


   double channel1 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 1;


   double channel2 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 2;
   double channel3 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 3;
   double channel4 = Math.Round(axAdvAI1.DataAnalog, 3);



   if (channel1 < 5.8 && channel2 < 5.8 && channel3 < 5.8 && channel4 < 5.8)
   {

      g13.Refresh();
```

```
g14.Refresh();
g15.Refresh();
g16.Refresh();
g1.Refresh();
g2.Refresh();
g3.Refresh();



g7.Refresh();
g8.Refresh();
g9.Refresh();
g17.Refresh();



gi.Refresh();
g5.Refresh();
g6.Refresh();



g10.Refresh();
g11.Refresh();
g12.Refresh();


g2.Text = " ";
g3.Text = " ";



g5.Text = " ";
g6.Text = " ";

g8.Text = " ";
g9.Text = " ";

g11.Text = " ";
g12.Text = " ";
g17.Text = " ";
g18.Refresh();
g18.Text = " ";
g19.Refresh();
g19.Text = " ";

g20.Refresh();
```

```
g20.Text = " ";



if (channel1 == 0)
{
   g13.Refresh();
   g13.ForeColor = Color.Blue;
   g13.Text = "No battery is Connected at port 0";
   g1.Text = "0 Volts";
   p1.Value = 0;
}
else if (channel1 <= 2.9 && channel1 > 0)
{
   g13.Refresh();
   g13.ForeColor = Color.Red;
   g13.Text = "Battery 1 is damaged";
   g1.Text = Math.Round((channel1 * 2 ), 3).ToString() + "Volts";
   p1.Value = 1;
}
else
{
   g13.Refresh();
   g13.ForeColor = Color.Red;
   g13.Text = "Battery 1 is damaged";
   g1.Text = Math.Round((channel1 * 2 ), 3).ToString() + "Volts";
   p1.Value = 2;
}// channel 1 progress bar

if (channel2 == 0)
{
   g14.Refresh();
   g14.ForeColor = Color.Blue;
   g14.Text = "No battery is Connected at port 1";
   gi.Text = " 0 Volts";
   p2.Value = 0;
}
else if (channel2 <= 2.9 && channel2 > 0)
{
   g14.Refresh();
   g14.ForeColor = Color.Red;
   g14.Text = "Battery 2 is damaged!";
   gi.Text = Math.Round((channel2 * 2 ), 3).ToString() + "Volts";
   p2.Value = 1;
}
else
{
   g14.Refresh();
   g14.ForeColor = Color.Red;
   g14.Text = "Battery 2 is damaged!";
```

```
    gi.Text = Math.Round((channel2 * 2 ), 3).ToString() + "Volts";
    p2.Value = 2;
}//channel 2 progress bar

if (channel3 == 0)
{
    g15.Refresh();
    g15.ForeColor = Color.Blue;
    g15.Text = "No battery is Connected at port 2";
    g7.Text = "0 Volts";
    p3.Value = 0;
}
else if (channel3 <= 2.9 && channel3 > 0)
{
    g15.Refresh();
    g15.ForeColor = Color.Red;
    g15.Text = "Battery 3 is damaged!";
    g7.Text = Math.Round((channel3 * 2 ), 3).ToString() + "Volts";
    p3.Value = 1;
}
else
{
    g15.Refresh();
    g15.ForeColor = Color.Red;
    g15.Text = "Battery 3 is damaged!";
    g7.Text = Math.Round((channel3 * 2 ), 3).ToString() + "Volts";
    p3.Value = 2;
}//Channel 3 reading progress bar

if (channel4 == 0)
{
    g16.Refresh();
    g16.ForeColor = Color.Blue;
    g16.Text = "No Battery is Connected at port 3";
    g10.Text = "0 Volts";
    p4.Value = 0;
}
else if (channel4 <= 2.9 && channel4 > 0)
{
    g16.Refresh();
    g16.ForeColor = Color.Red;
    g16.Text = "Battery 4 is damaged!";
    g10.Text = Math.Round((channel4 * 2), 3).ToString() + "Volts";
    p4.Value = 1;
}
else
{
    g16.Refresh();
    g16.ForeColor = Color.Red;
    g16.Text = "Battery 4 is damaged!";
```

```
        g10.Text = Math.Round((channel4 * 2 ), 3).ToString() + "Volts";
        p4.Value = 2;
    }// channel 4 reading progress bar




}
else
{
    if (channel1 < 5.8)
    {
        g1.Refresh();
        g17.Refresh();
        g17.Text = " ";


        g2.Refresh();
        g2.Text = " ";
        g3.Refresh();
        g3.Text = " ";

        if (channel1 == 0)
        {
            g13.Refresh();
            g13.ForeColor = Color.Blue;
            g13.Text = "No battery is Connected at port 0";
            g1.Text = "0 Volts";
            p1.Value = 0;
        }
        else if (channel1 <= 2.9 && channel1 > 0)
        {
            g13.Refresh();
            g13.ForeColor = Color.Red;
            g13.Text = "Battery 1 is damaged";

            g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
            p1.Value = 1;
        }
        else
        {
            g13.Refresh();
            g13.ForeColor = Color.Red;
            g13.Text = "Battery 1 is damaged";

            g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
            p1.Value = 2;
```

```
    }


}
if (channel1 >= 5.8)
{
    g13.Refresh();
    g13.Text = " ";



    double p = (.7632 * (channel1 * 2 ) - 8.7197) * 100 + 2.71;
    p = Math.Round(p, 2);
    current = 5.55555556 * (channel1 * 2);//(80A/14.4V)*nVolts=current
    v = Convert.ToInt32(current);
    t = (80 - v) / 3;//3~A/h considering }


    if (channel1 <= 6.267)
    {

        g17.Refresh();
        g17.ForeColor = Color.Red;
        g17.Text = "Low level";
        if (channel1 >= 5.8 && channel1 <= 6.0355)
        {
            p1.Value = 3;

        }
        else
        {
            p1.Value = 4;
        }
    }

    else if (channel1 <= 6.3 && channel1 > 6.267)
    {


        g17.Refresh();

        g17.ForeColor = Color.Orange;
        g17.Text = "mid level";

        if (channel1 > 6.267 && channel1 <= 6.3235)
        {
            p1.Value = 5;

        }
        else
```

```
    {
        p1.Value = 6;

    }
}
else
{
    g17.Refresh();
    g17.ForeColor = Color.Green;

    g17.Text = "High level";

    if (channel1 > 6.3 && channel1 <= 6.75)
    {
        p1.Value = 7;
        if (channel1 >= 6.35)
        {
            g13.Refresh();
            g13.ForeColor = Color.Green;
            g13.Text = "Battery is fully charged";
        }
        else
        {
            g13.Refresh();
            g13.Text = " ";
        }

    }
    else
    {
        p1.Value = 8;
        g13.Refresh();
        if (channel1 >= 7.2)
        {

            g13.Refresh();
            g13.ForeColor = Color.Red;
            g13.Text = "Battery is disconnected";

        }
        else
        {

            g13.Refresh();
            g13.ForeColor = Color.Green;
            g13.Text = "Battery is fully charged";
        }


    }
```

```
    }
    g1.Refresh();
    g2.Refresh();
    g3.Refresh();
    g1.Text = (channel1 * 2).ToString() + "Volts";
    if (channel1 >= 6.35)
    {
       g2.Text = "100 % Charged";
    }
    else
    {
       g2.Refresh();
       g2.Text = p.ToString() + "% Charged";
    }
    g3.Text = t.ToString() + "Hours Remaining";

    g1.Refresh();
    g2.Refresh();
    g3.Refresh();
    g13.Refresh();

}///channel 1 ends

if (channel2 < 5.8)
{
    gi.Refresh();
    g5.Refresh();
    g5.Text = " ";


    g6.Refresh();
    g6.Text = " ";
    g18.Refresh();
    g18.Text = " ";

    if (channel2 == 0)
    {
       g14.Refresh();
       g14.ForeColor = Color.Blue;
       g14.Text = "No battery is Connected at port 1";
       gi.Text = "0 Volts";
       p2.Value = 0;
    }
    else if (channel2 <= 2.9 && channel2 > 0)
    {
       g14.Refresh();
       g14.ForeColor = Color.Red;
       g14.Text = "Battery 2 is damaged";
```

```
            gi.Text = Math.Round(channel2 * 2,3 ).ToString() + "Volts";
            p2.Value = 1;
        }
        else
        {
            g14.Refresh();
            g14.ForeColor = Color.Red;
            g14.Text = "Battery 2 is damaged";

            gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
            p2.Value = 2;
        }


    }
    if (channel2 >= 5.8)
    {
        g14.Refresh();
        g14.Text = " ";



        double pi = (.7632 * (channel2 * 2 ) - 8.7197) * 100 + 2.71;
        pi = Math.Round(pi, 2);
        current1 = 5.55555556 * (channel2 * 2);//(80A/14.4V)*nVolts=current
        v1 = Convert.ToInt32(current1);
        t1 = (80 - v1) / 3;//3~A/h considering }


        if (channel2 <= 6.267)
        {

            g18.Refresh();
            g18.ForeColor = Color.Red;
            g18.Text = "Low level";
            g14.Refresh();
            g14.Text = " ";
            if (channel2 >= 5.8 && channel2 <= 6.0355)
            {
                p2.Value = 3;

            }
            else
            {
                p2.Value = 4;

            }
        }

        else if (channel2 <= 6.3 && channel2 > 6.267)
```

```
{


   g18.Refresh();

   g18.ForeColor = Color.Orange;
   g18.Text = "mid level";
   g14.Refresh();
   g14.Text = " ";

   if (channel2 > 6.267 && channel2 <= 6.2835)
   {
      p2.Value = 5;


   }
   else
   {
      p2.Value = 6;

   }
}
else
{
   g18.Refresh();
   g18.ForeColor = Color.Green;

   g18.Text = "High level";

   if (channel2 > 6.3 && channel2 <= 6.75)
   {
      p2.Value = 7;
      if (channel2 >= 6.35)
      {
         g14.Refresh();
         g14.ForeColor = Color.Green;
         g14.Text = "Battery is fully charged";
      }
      else
      {
         g14.Refresh();
         g14.Text = " ";
      }

   }
   else
   {
      p2.Value = 8;

      if (channel2 >= 7.2)
```

```
            {

               g14.Refresh();
               g14.ForeColor = Color.Red;
               g14.Text = "Battery is disconnected";
            }
            else
            {

               g14.Refresh();
               g14.ForeColor = Color.Green;

               g14.Text = "Battery is fully charged";
            }


         }

      }
      gi.Refresh();
      g5.Refresh();
      g6.Refresh();
      gi.Text = Math.Round(channel2 * 2 , 3).ToString() + "Volts";
      if (channel2 >= 6.35)
      {
         g5.Text = "100 % Charged";
      }
      else
      {
         g5.Refresh();
         g5.Text = pi.ToString() + "% Charged";
      }
      g6.Text = t1.ToString() + "Hours Remaining";

      gi.Refresh();
      g5.Refresh();
      g6.Refresh();

   }//////////channel 2 ends here
   if (channel3 < 5.8)
   {
      g7.Refresh();
      g8.Refresh();
      g8.Text = " ";


      g9.Refresh();
      g9.Text = " ";
      g19.Refresh();
      g19.Text = " ";
```

```
    if (channel3 == 0)
    {
       g15.Refresh();
       g15.ForeColor = Color.Blue;
       g15.Text = "No battery is Connected at port 2";
       g7.Text = "0 Volts";
       g19.Refresh();
       g19.Text = " ";
       p3.Value = 0;
    }
    else if (channel3 <= 2.9 && channel3 > 0)
    {
       g15.Refresh();
       g15.ForeColor = Color.Red;
       g19.Refresh();
       g19.Text = " ";
       g15.Text = "Battery 3 is damaged";

       g7.Text = Math.Round(channel3 * 2 , 3).ToString() + "Volts";
       p3.Value = 1;
    }
    else
    {
       g15.Refresh();
       g15.ForeColor = Color.Red;
       g15.Text = "Battery 3 is damaged";

       g7.Text = Math.Round(channel3 * 2 , 3).ToString() + "Volts";
       p3.Value = 2;
    }


}
if (channel3 >= 5.8)
{
   g15.Refresh();
   g15.Text = " ";



   double pii = (.7632 * (channel3 * 2 ) - 8.7197) * 100 + 2.71;
   pii = Math.Round(pii, 2);
   current2 = 5.55555556 * (channel3 * 2 );//(80A/14.4V)*nVolts=current
   v2 = Convert.ToInt32(current2);
   t2 = (80 - v2) / 3;//3~A/h considering }


   if (channel3 <= 6.267)
   {
```

```
    g19.Refresh();
    g19.ForeColor = Color.Red;
    g19.Text = "Low level";
    g15.Refresh();
    g15.Text = " ";
    if (channel3 >= 5.8 && channel3 <= 6.0355)
    {
        p3.Value = 3;

    }
    else
    {
        p3.Value = 4;

    }
}

else if (channel3 <= 6.3 && channel3 > 6.267)
{


    g19.Refresh();

    g19.ForeColor = Color.Orange;
    g19.Text = "mid level";
    g15.Refresh();
    g15.Text = " ";

    if (channel3 > 6.267 && channel3 <= 6.2835)
    {
        p3.Value = 5;


    }
    else
    {
        p3.Value = 6;

    }
}
else
{
    g19.Refresh();
    g19.ForeColor = Color.Green;

    g19.Text = "High level";

    if (channel3 > 6.3 && channel3 <= 6.75)
    {
```

```
            p3.Value = 7;
            if (channel3 >= 6.35)
            {
                g15.Refresh();
                g15.ForeColor = Color.Green;
                g15.Text = "Battery is fully charged";
            }
            else
            {
                g15.Refresh();
                g15.Text = " ";
            }

        }
        else
        {
            p3.Value = 8;

            if (channel3 >= 7.2)
            {
                g15.Refresh();
                g15.ForeColor = Color.Red;

                g15.Text = "Battery is disconnected";


            }
            else
            {

                g15.Refresh();
                g15.ForeColor = Color.Green;
                g15.Text = "Battery is fully charged";


            }


        }

    }
    g7.Refresh();
    g8.Refresh();
    g9.Refresh();
    g7.Text = (channel3 * 2 ).ToString() + "Volts";

    if (channel3 >= 6.35)
    {
        g8.Text = "100 % Charged";
    }
```

```
    else
    {
        g8.Refresh();
        g8.Text = pii.ToString() + "% Charged";
        g8.Refresh();
    }
    g9.Text = t2.ToString() + "Hours Remaining";

    g7.Refresh();
    g8.Refresh();
    g9.Refresh();

}
//channel 3 ends


if (channel4 < 5.8)
{
    g10.Refresh();
    g11.Refresh();
    g11.Text = " ";


    g12.Refresh();
    g12.Text = " ";
    g20.Refresh();
    g20.Text = " ";

    if (channel4 == 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Blue;
        g16.Text = "No battery is Connected at port 3";
        g10.Text = "0 Volts";
        g20.Refresh();
        g20.Text = " ";
        p4.Value = 0;
    }
    else if (channel4 <= 2.9 && channel4 > 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Red;
        g20.Refresh();
        g20.Text = " ";
        g16.Text = "Battery 4 is damaged";

        g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
        p4.Value = 1;
    }
    else
```

```
    {
        g16.Refresh();
        g16.ForeColor = Color.Red;
        g16.Text = "Battery 4 is damaged";

        g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
        p4.Value = 2;
    }


}
if (channel4 >= 5.8)
{
    g16.Refresh();
    g16.Text = " ";



    double piii = (.7632 * (channel4 * 2) - 8.7197) * 100 + 2.71;
    piii = Math.Round(piii, 2);
    current3 = 5.55555556 * (channel4 * 2);//(80A/14.4V)*nVolts=current
    v3 = Convert.ToInt32(current3);
    t3 = (80 - v3) / 3;//3~A/h considering }


    if (channel4 <= 6.267)
    {

        g20.Refresh();
        g20.ForeColor = Color.Red;
        g20.Text = "Low level";
       g16.Refresh();
        g16.Text = " ";
        if (channel4 >= 5.8 && channel4 <= 6.0355)
        {
            p4.Value = 3;

        }
        else
        {
            p4.Value = 4;

        }
    }

    else if (channel4 <= 6.3 && channel4 > 6.267)
    {


        g20.Refresh();
```

```
    g20.ForeColor = Color.Orange;
    g20.Text = "mid level";
    g16.Refresh();
    g16.Text = " ";

    if (channel4 > 6.267 && channel4<= 6.2835)
    {
       p4.Value = 5;


    }
    else
    {
       p4.Value = 6;

    }
}
else
{
    g20.Refresh();
    g20.ForeColor = Color.Green;

    g20.Text = "High level";

    if (channel4 > 6.3 && channel4 <= 6.75)
    {
       p4.Value = 7;
       if (channel4 >= 6.35)
       {
          g16.Refresh();
          g16.ForeColor = Color.Green;
          g16.Text = "Battery is fully charged";
       }
       else
       {
          g16.Refresh();
          g16.Text = " ";
       }

    }
    else
    {
       p4.Value = 8;

       if (channel4 >= 7.2)
       {
          g16.Refresh();
          g16.ForeColor = Color.Red;
```

```
                g16.Text = "Battery is disconnected";


            }
            else
            {

                g16.Refresh();
                g16.ForeColor = Color.Green;
                g16.Text = "Battery is fully charged";


            }


        }
        g10.Refresh();
        g11.Refresh();
        g12.Refresh();
        g10.Text = (channel4 * 2).ToString() + "Volts";

        if (channel4 >= 6.35)
        {
            g11.Text = "100 % Charged";
        }
        else
        {
            g11.Refresh();
            g11.Text = piii.ToString() + "% Charged";
            g11.Refresh();
        }
        g12.Text = t3.ToString() + "Hours Remaining";

        g10.Refresh();
        g11.Refresh();
        g12.Refresh();

    }



  }
}

private void buttonStart_Click(object sender, EventArgs e)
{
  if (!this.set)
  {
```

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

    }

    private void buttonStop_Click(object sender, EventArgs e)
    {

        this.set = false;
        timer1.Stop();
    }

    private void button2_Click(object sender, EventArgs e)
    {

        timer1.Stop();
        this.Close();
    }


  }
}
```

Code for the windows Form3 that has been initialized as f3:

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

namespace Solar_Battery_Charging_Status_Monitoring_Panel
{
  public partial class Form3 : Form
  {
    public Form3()
    {
      InitializeComponent();
    }
    private bool set = false;
    private double p = 0;
    private double current;
    private int v;
    private int t = 0;
```

```
private double pi = 0;
private double current1;
private int v1;
private int t1;
private int i = 0;
private double pii = 0;
private double current2;
private int v2;
private int t2;
private double piii = 0;
private double current3;
private int v3;
private int t3;
private void cmdSelectDevice_Click(object sender, EventArgs e)
{
   // selecting device
   axAdvAI1.SelectDevice();
   txtDeviceName.Text = axAdvAI1.DeviceName;
}

private void Form3_Load(object sender, EventArgs e)
{

}

private void timer1_Tick(object sender, EventArgs e)
{
   axAdvAI1.ChannelNow = 4;


   double channel1 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 5;


   double channel2 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 6;
   double channel3 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 7;
   double channel4 = Math.Round(axAdvAI1.DataAnalog, 3);



   if (channel1 < 5.8 && channel2 < 5.8 && channel3 < 5.8 && channel4 < 5.8)
   {

      g13.Refresh();
      g14.Refresh();
```

```
g15.Refresh();
g16.Refresh();
g1.Refresh();
g2.Refresh();
g3.Refresh();



g7.Refresh();
g8.Refresh();
g9.Refresh();
g17.Refresh();



gi.Refresh();
g5.Refresh();
g6.Refresh();



g10.Refresh();
g11.Refresh();
g12.Refresh();


g2.Text = " ";
g3.Text = " ";



g5.Text = " ";
g6.Text = " ";

g8.Text = " ";
g9.Text = " ";

g11.Text = " ";
g12.Text = " ";
g17.Text = " ";
g18.Refresh();
g18.Text = " ";
g19.Refresh();
g19.Text = " ";

g20.Refresh();
g20.Text = " ";
```

```
if (channel1 == 0)
{
   g13.Refresh();
   g13.ForeColor = Color.Blue;
   g13.Text = "No battery is Connected at port 4";
   g1.Text = "0 Volts";
   p5.Value = 0;
}
else if (channel1 <= 2.9 && channel1 > 0)
{
   g13.Refresh();
   g13.ForeColor = Color.Red;
   g13.Text = "Battery 5 is damaged";
   g1.Text = Math.Round((channel1 * 2 ), 3).ToString() + "Volts";
   p5.Value = 1;
}
else
{
   g13.Refresh();
   g13.ForeColor = Color.Red;
   g13.Text = "Battery 5 is damaged";
   g1.Text = Math.Round((channel1 * 2 ), 3).ToString() + "Volts";
   p5.Value = 2;
}// channel 1 progress bar

if (channel2 == 0)
{
   g14.Refresh();
   g14.ForeColor = Color.Blue;
   g14.Text = "No battery is Connected at port 5";
   gi.Text = " 0 Volts";
   p6.Value = 0;
}
else if (channel2 <= 2.9 && channel2 > 0)
{
   g14.Refresh();
   g14.ForeColor = Color.Red;
   g14.Text = "Battery 6 is damaged!";
   gi.Text = Math.Round((channel2 * 2 ), 3).ToString() + "Volts";
   p6.Value = 1;
}
else
{
   g14.Refresh();
   g14.ForeColor = Color.Red;
   g14.Text = "Battery 6 is damaged!";
   gi.Text = Math.Round((channel2 * 2 ), 3).ToString() + "Volts";
```

```
    p6.Value = 2;
}//channel 2 progress bar

if (channel3 == 0)
{
    g15.Refresh();
    g15.ForeColor = Color.Blue;
    g15.Text = "No battery is Connected at port 6";
    g7.Text = "0 Volts";
    p7.Value = 0;
}
else if (channel3 <= 2.9 && channel3 > 0)
{
    g15.Refresh();
    g15.ForeColor = Color.Red;
    g15.Text = "Battery 7 is damaged!";
    g7.Text = Math.Round((channel3 * 2 ), 3).ToString() + "Volts";
    p7.Value = 1;
}
else
{
    g15.Refresh();
    g15.ForeColor = Color.Red;
    g15.Text = "Battery 7 is damaged!";
    g7.Text = Math.Round((channel3 * 2 ), 3).ToString() + "Volts";
    p7.Value = 2;
}//Channel 3 reading progress bar

if (channel4 == 0)
{
    g16.Refresh();
    g16.ForeColor = Color.Blue;
    g16.Text = "No Battery is Connected at port 7";
    g10.Text = "0 Volts";
    p8.Value = 0;
}
else if (channel4 <= 2.9 && channel4 > 0)
{
    g16.Refresh();
    g16.ForeColor = Color.Red;
    g16.Text = "Battery 8 is damaged!";
    g10.Text = Math.Round((channel4 * 2), 3).ToString() + "Volts";
    p8.Value = 1;
}
else
{
    g16.Refresh();
    g16.ForeColor = Color.Red;
    g16.Text = "Battery 8 is damaged!";
    g10.Text = Math.Round((channel4 * 2 ), 3).ToString() + "Volts";
```

```
        p8.Value = 2;
    }// channel 4 reading progress bar




}
else
{
    if (channel1 < 5.8)
    {
        g1.Refresh();
        g17.Refresh();
        g17.Text = " ";


        g2.Refresh();
        g2.Text = " ";
        g3.Refresh();
        g3.Text = " ";

        if (channel1 == 0)
        {
            g13.Refresh();
            g13.ForeColor = Color.Blue;
            g13.Text = "No battery is Connected at port 4";
            g1.Text = "0 Volts";
            p5.Value = 0;
        }
        else if (channel1 <= 2.9 && channel1 > 0)
        {
            g13.Refresh();
            g13.ForeColor = Color.Red;
            g13.Text = "Battery 5 is damaged";

            g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
            p5.Value = 1;
        }
        else
        {
            g13.Refresh();
            g13.ForeColor = Color.Red;
            g13.Text = "Battery 5 is damaged";

            g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
            p5.Value = 2;
        }
```

```
}
if (channel1 >= 5.8)
{
   g13.Refresh();
   g13.Text = " ";


   double p = (.7632 * (channel1 * 2 ) - 8.7197) * 100 + 2.71;
   p = Math.Round(p, 2);
   current = 5.55555556 * (channel1 * 2);//(80A/14.4V)*nVolts=current
   v = Convert.ToInt32(current);
   t = (80 - v) / 3;//3~A/h considering }


   if (channel1 <= 6.267)
   {

      g17.Refresh();
      g17.ForeColor = Color.Red;
      g17.Text = "Low level";
      if (channel1 >= 5.8 && channel1 <= 6.0355)
      {
         p5.Value = 3;

      }
      else
      {
         p5.Value = 4;
      }
   }

   else if (channel1 <= 6.3 && channel1 > 6.267)
   {


      g17.Refresh();

      g17.ForeColor = Color.Orange;
      g17.Text = "mid level";

      if (channel1 > 6.267 && channel1 <= 6.3235)
      {
         p5.Value = 5;

      }
      else
      {
```

```
            p5.Value = 6;

        }
    }
    else
    {
        g17.Refresh();
        g17.ForeColor = Color.Green;

        g17.Text = "High level";

        if (channel1 > 6.3 && channel1 <= 6.75)
        {
            p5.Value = 7;
            if (channel1 >= 6.35)
            {
                g13.Refresh();
                g13.ForeColor = Color.Green;
                g13.Text = "Battery is fully charged";
            }
            else
            {
                g13.Refresh();
                g13.Text = " ";
            }

        }
        else
        {
            p5.Value = 8;
            g13.Refresh();
            if (channel1 >= 7.2)
            {

                g13.Refresh();
                g13.ForeColor = Color.Red;
                g13.Text = "Battery is disconnected";

            }
            else
            {

                g13.Refresh();
                g13.ForeColor = Color.Green;
                g13.Text = "Battery is fully charged";
            }


        }
```

```
    }
    g1.Refresh();
    g2.Refresh();
    g3.Refresh();
    g1.Text = (channel1 * 2).ToString() + "Volts";
    if (channel1 >= 6.35)
    {
       g2.Text = "100 % Charged";
    }
    else
    {
       g2.Refresh();
       g2.Text = p.ToString() + "% Charged";
    }
    g3.Text = t.ToString() + "Hours Remaining";

    g1.Refresh();
    g2.Refresh();
    g3.Refresh();
    g13.Refresh();

}///channel 1 ends

if (channel2 < 5.8)
{
    gi.Refresh();
    g5.Refresh();
    g5.Text = " ";


    g6.Refresh();
    g6.Text = " ";
    g18.Refresh();
    g18.Text = " ";

    if (channel2 == 0)
    {
       g14.Refresh();
       g14.ForeColor = Color.Blue;
       g14.Text = "No battery is Connected at port 5";
       gi.Text = "0 Volts";
       p6.Value = 0;
    }
    else if (channel2 <= 2.9 && channel2 > 0)
    {
       g14.Refresh();
       g14.ForeColor = Color.Red;
       g14.Text = "Battery 6 is damaged";

       gi.Text = Math.Round(channel2 * 2,3 ).ToString() + "Volts";
```

```
            p6.Value = 1;
        }
        else
        {
            g14.Refresh();
            g14.ForeColor = Color.Red;
            g14.Text = "Battery 6 is damaged";

            gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
            p6.Value = 2;
        }


    }
    if (channel2 >= 5.8)
    {
        g14.Refresh();
        g14.Text = " ";



        double pi = (.7632 * (channel2 * 2 ) - 8.7197) * 100 + 2.71;
        pi = Math.Round(pi, 2);
        current1 = 5.55555556 * (channel2 * 2);//(80A/14.4V)*nVolts=current
        v1 = Convert.ToInt32(current1);
        t1 = (80 - v1) / 3;//3~A/h considering }


        if (channel2 <= 6.267)
        {

            g18.Refresh();
            g18.ForeColor = Color.Red;
            g18.Text = "Low level";
            g14.Refresh();
            g14.Text = " ";
            if (channel2 >= 5.8 && channel2 <= 6.0355)
            {
                p6.Value = 3;

            }
            else
            {
                p6.Value = 4;

            }
        }

        else if (channel2 <= 6.3 && channel2 > 6.267)
        {
```

```
    g18.Refresh();

    g18.ForeColor = Color.Orange;
    g18.Text = "mid level";
    g14.Refresh();
    g14.Text = " ";

    if (channel2 > 6.267 && channel2 <= 6.2835)
    {
       p6.Value = 5;


    }
    else
    {
       p6.Value = 6;

    }
}
else
{
    g18.Refresh();
    g18.ForeColor = Color.Green;

    g18.Text = "High level";

    if (channel2 > 6.3 && channel2 <= 6.75)
    {
       p6.Value = 7;
       if (channel2 >= 6.35)
       {
          g14.Refresh();
          g14.ForeColor = Color.Green;
          g14.Text = "Battery is fully charged";
       }
       else
       {
          g14.Refresh();
          g14.Text = " ";
       }

    }
    else
    {
       p6.Value = 8;

       if (channel2 >= 7.2)
       {
```

```
            g14.Refresh();
            g14.ForeColor = Color.Red;
            g14.Text = "Battery is disconnected";
        }
        else
        {

            g14.Refresh();
            g14.ForeColor = Color.Green;

            g14.Text = "Battery is fully charged";
        }


    }

  }
  gi.Refresh();
  g5.Refresh();
  g6.Refresh();
  gi.Text = Math.Round(channel2 * 2 , 3).ToString() + "Volts";
  if (channel2 >= 6.35)
  {
     g5.Text = "100 % Charged";
  }
  else
  {
     g5.Refresh();
     g5.Text = pi.ToString() + "% Charged";
  }
  g6.Text = t1.ToString() + "Hours Remaining";

  gi.Refresh();
  g5.Refresh();
  g6.Refresh();

}//////////channel 2 ends here
if (channel3 < 5.8)
{
  g7.Refresh();
  g8.Refresh();
  g8.Text = " ";


  g9.Refresh();
  g9.Text = " ";
  g19.Refresh();
  g19.Text = " ";
```

```
    if (channel3 == 0)
    {
       g15.Refresh();
       g15.ForeColor = Color.Blue;
       g15.Text = "No battery is Connected at port 6";
       g7.Text = "0 Volts";
       g19.Refresh();
       g19.Text = " ";
       p7.Value = 0;
    }
    else if (channel3 <= 2.9 && channel3 > 0)
    {
       g15.Refresh();
       g15.ForeColor = Color.Red;
       g19.Refresh();
       g19.Text = " ";
       g15.Text = "Battery 7 is damaged";

       g7.Text = Math.Round(channel3 * 2 , 3).ToString() + "Volts";
       p7.Value = 1;
    }
    else
    {
       g15.Refresh();
       g15.ForeColor = Color.Red;
       g15.Text = "Battery 7 is damaged";

       g7.Text = Math.Round(channel3 * 2 , 3).ToString() + "Volts";
       p7.Value = 2;
    }


}
if (channel3 >= 5.8)
{
    g15.Refresh();
    g15.Text = " ";



    double pii = (.7632 * (channel3 * 2 ) - 8.7197) * 100 + 2.71;
    pii = Math.Round(pii, 2);
    current2 = 5.55555556 * (channel3 * 2 );//(80A/14.4V)*nVolts=current
    v2 = Convert.ToInt32(current2);
    t2 = (80 - v2) / 3;//3~A/h considering }


    if (channel3 <= 6.267)
    {
```

```
    g19.Refresh();
    g19.ForeColor = Color.Red;
    g19.Text = "Low level";
    g15.Refresh();
    g15.Text = " ";
    if (channel3 >= 5.8 && channel3 <= 6.0355)
    {
        p7.Value = 3;

    }
    else
    {
        p7.Value = 4;

    }
}

else if (channel3 <= 6.3 && channel3 > 6.267)
{


    g19.Refresh();

    g19.ForeColor = Color.Orange;
    g19.Text = "mid level";
    g15.Refresh();
    g15.Text = " ";

    if (channel3 > 6.267 && channel3 <= 6.2835)
    {
        p7.Value = 5;


    }
    else
    {
        p7.Value = 6;

    }
}
else
{
    g19.Refresh();
    g19.ForeColor = Color.Green;

    g19.Text = "High level";

    if (channel3 > 6.3 && channel3 <= 6.75)
    {
        p7.Value = 7;
```

```
        if (channel3 >= 6.35)
        {
            g15.Refresh();
            g15.ForeColor = Color.Green;
            g15.Text = "Battery is fully charged";
        }
        else
        {
            g15.Refresh();
            g15.Text = " ";
        }

    }
    else
    {
        p7.Value = 8;

        if (channel3 >= 7.2)
        {
            g15.Refresh();
            g15.ForeColor = Color.Red;

            g15.Text = "Battery is disconnected";


        }
        else
        {

            g15.Refresh();
            g15.ForeColor = Color.Green;
            g15.Text = "Battery is fully charged";


        }


    }

}
g7.Refresh();
g8.Refresh();
g9.Refresh();
g7.Text = (channel3 * 2 ).ToString() + "Volts";

if (channel3 >= 6.35)
{
    g8.Text = "100 % Charged";
}
else
```

```
    {
        g8.Refresh();
        g8.Text = pii.ToString() + "% Charged";
        g8.Refresh();
    }
    g9.Text = t2.ToString() + "Hours Remaining";

    g7.Refresh();
    g8.Refresh();
    g9.Refresh();

}
//channel 3 ends


if (channel4 < 5.8)
{
    g10.Refresh();
    g11.Refresh();
    g11.Text = " ";


    g12.Refresh();
    g12.Text = " ";
    g20.Refresh();
    g20.Text = " ";

    if (channel4 == 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Blue;
        g16.Text = "No battery is Connected at port 7";
        g10.Text = "0 Volts";
        g20.Refresh();
        g20.Text = " ";
        p8.Value = 0;
    }
    else if (channel4 <= 2.9 && channel4 > 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Red;
        g20.Refresh();
        g20.Text = " ";
        g16.Text = "Battery 8 is damaged";

        g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
        p8.Value = 1;
    }
    else
    {
```

```
            g16.Refresh();
            g16.ForeColor = Color.Red;
            g16.Text = "Battery 8 is damaged";

            g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
            p8.Value = 2;
        }


    }
    if (channel4 >= 5.8)
    {
        g16.Refresh();
        g16.Text = " ";



        double piii = (.7632 * (channel4 * 2) - 8.7197) * 100 + 2.71;
        piii = Math.Round(piii, 2);
        current3 = 5.55555556 * (channel4 * 2);//(80A/14.4V)*nVolts=current
        v3 = Convert.ToInt32(current3);
        t3 = (80 - v3) / 3;//3~A/h considering }


        if (channel4 <= 6.267)
        {

            g20.Refresh();
            g20.ForeColor = Color.Red;
            g20.Text = "Low level";
            g16.Refresh();
            g16.Text = " ";
            if (channel4 >= 5.8 && channel4 <= 6.0355)
            {
                p8.Value = 3;

            }
            else
            {
                p8.Value = 4;

            }
        }

        else if (channel4 <= 6.3 && channel4 > 6.267)
        {


            g20.Refresh();
```

```
    g20.ForeColor = Color.Orange;
    g20.Text = "mid level";
    g16.Refresh();
    g16.Text = " ";

    if (channel4 > 6.267 && channel4<= 6.2835)
    {
       p8.Value = 5;


    }
    else
    {
       p8.Value = 6;

    }
}
else
{
    g20.Refresh();
    g20.ForeColor = Color.Green;

    g20.Text = "High level";

    if (channel4 > 6.3 && channel4 <= 6.75)
    {
       p8.Value = 7;
       if (channel4 >= 6.35)
       {
          g16.Refresh();
          g16.ForeColor = Color.Green;
          g16.Text = "Battery is fully charged";
       }
       else
       {
          g16.Refresh();
          g16.Text = " ";
       }

    }
    else
    {
       p8.Value = 8;

       if (channel4 >= 7.2)
       {
          g16.Refresh();
          g16.ForeColor = Color.Red;

          g16.Text = "Battery is disconnected";
```

```
            }
            else
            {

               g16.Refresh();
               g16.ForeColor = Color.Green;
               g16.Text = "Battery is fully charged";


            }


         }

      }
      g10.Refresh();
      g11.Refresh();
      g12.Refresh();
      g10.Text = (channel4 * 2).ToString() + "Volts";

      if (channel4 >= 6.35)
      {
         g11.Text = "100 % Charged";
      }
      else
      {
         g11.Refresh();
         g11.Text = piii.ToString() + "% Charged";
         g11.Refresh();
      }
      g12.Text = t3.ToString() + "Hours Remaining";

      g10.Refresh();
      g11.Refresh();
      g12.Refresh();

   }


}
}
 private void buttonStart_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)
    {
        this.set = false;
        timer1.Stop();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        timer1.Stop();
        this.Close();
    }

   //timer1 ends


  }
}
```

Code for the windows Form5 that has been initialized as f4:

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

namespace Solar_Battery_Charging_Status_Monitoring_Panel
{
  public partial class Form5 : Form
  {
    public Form5()
    {
      InitializeComponent();
    }

    private void Form5_Load(object sender, EventArgs e)
    {

    }
    private bool set = false;
    private double p = 0;
    private double current;
    private int v;
```

```
private int t = 0;
private double pi = 0;
private double current1;
private int v1;
private int t1;
private int i = 0;
private double pii = 0;
private double current2;
private int v2;
private int t2;
private double piii = 0;
private double current3;
private int v3;
private int t3;

private void cmdSelectDevice_Click(object sender, EventArgs e)
{
   // selecting device
   axAdvAI1.SelectDevice();
   txtDeviceName.Text = axAdvAI1.DeviceName;
}

private void timer1_Tick(object sender, EventArgs e)
{

   axAdvAI1.ChannelNow = 8;


   double channel1 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 9;


   double channel2 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 10;
   double channel3 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 11;
   double channel4 = Math.Round(axAdvAI1.DataAnalog, 3);



   if (channel1 < 5.8 && channel2 < 5.8 && channel3 < 5.8 && channel4 < 5.8)
   {

      g13.Refresh();
      g14.Refresh();
      g15.Refresh();
      g16.Refresh();
```

```
g1.Refresh();
g2.Refresh();
g3.Refresh();



g7.Refresh();
g8.Refresh();
g9.Refresh();
g17.Refresh();



gi.Refresh();
g5.Refresh();
g6.Refresh();



g10.Refresh();
g11.Refresh();
g12.Refresh();


g2.Text = " ";
g3.Text = " ";



g5.Text = " ";
g6.Text = " ";

g8.Text = " ";
g9.Text = " ";

g11.Text = " ";
g12.Text = " ";
g17.Text = " ";
g18.Refresh();
g18.Text = " ";
g19.Refresh();
g19.Text = " ";

g20.Refresh();
g20.Text = " ";
```

```
if (channel1 == 0)
{
   g13.Refresh();
   g13.ForeColor = Color.Blue;
   g13.Text = "No battery is Connected at port 8";
   g1.Text = "0 Volts";
   p9.Value = 0;
}
else if (channel1 <= 2.9 && channel1 > 0)
{
   g13.Refresh();
   g13.ForeColor = Color.Red;
   g13.Text = "Battery 9 is damaged";
   g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
   p9.Value = 1;
}
else
{
   g13.Refresh();
   g13.ForeColor = Color.Red;
   g13.Text = "Battery 9 is damaged";
   g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
   p9.Value = 2;
}// channel 1 progress bar

if (channel2 == 0)
{
   g14.Refresh();
   g14.ForeColor = Color.Blue;
   g14.Text = "No battery is Connected at port 9";
   gi.Text = " 0 Volts";
   p10.Value = 0;
}
else if (channel2 <= 2.9 && channel2 > 0)
{
   g14.Refresh();
   g14.ForeColor = Color.Red;
   g14.Text = "Battery 10 is damaged!";
   gi.Text = Math.Round((channel2 * 2), 3).ToString() + "Volts";
   p10.Value = 1;
}
else
{
   g14.Refresh();
   g14.ForeColor = Color.Red;
   g14.Text = "Battery 10 is damaged!";
   gi.Text = Math.Round((channel2 * 2), 3).ToString() + "Volts";
   p10.Value = 2;
}//channel 2 progress bar
```

```
if (channel3 == 0)
{
   g15.Refresh();
   g15.ForeColor = Color.Blue;
   g15.Text = "No battery is Connected at port 10";
   g7.Text = "0 Volts";
   p11.Value = 0;
}
else if (channel3 <= 2.9 && channel3 > 0)
{
   g15.Refresh();
   g15.ForeColor = Color.Red;
   g15.Text = "Battery 11 is damaged!";
   g7.Text = Math.Round((channel3 * 2), 3).ToString() + "Volts";
   p11.Value = 1;
}
else
{
   g15.Refresh();
   g15.ForeColor = Color.Red;
   g15.Text = "Battery 11 is damaged!";
   g7.Text = Math.Round((channel3 * 2), 3).ToString() + "Volts";
   p11.Value = 2;
}//Channel 3 reading progress bar

if (channel4 == 0)
{
   g16.Refresh();
   g16.ForeColor = Color.Blue;
   g16.Text = "No Battery is Connected at port 11";
   g10.Text = "0 Volts";
   p12.Value = 0;
}
else if (channel4 <= 2.9 && channel4 > 0)
{
   g16.Refresh();
   g16.ForeColor = Color.Red;
   g16.Text = "Battery 12 is damaged!";
   g10.Text = Math.Round((channel4 * 2), 3).ToString() + "Volts";
   p12.Value = 1;
}
else
{
   g16.Refresh();
   g16.ForeColor = Color.Red;
   g16.Text = "Battery 12 is damaged!";
   g10.Text = Math.Round((channel4 * 2), 3).ToString() + "Volts";
   p12.Value = 2;
}// channel 4 reading progress bar
```

```
}
else
{
   if (channel1 < 5.8)
   {
      g1.Refresh();
      g17.Refresh();
      g17.Text = " ";


      g2.Refresh();
      g2.Text = " ";
      g3.Refresh();
      g3.Text = " ";

      if (channel1 == 0)
      {
         g13.Refresh();
         g13.ForeColor = Color.Blue;
         g13.Text = "No battery is Connected at port 8";
         g1.Text = "0 Volts";
         p9.Value = 0;
      }
      else if (channel1 <= 2.9 && channel1 > 0)
      {
         g13.Refresh();
         g13.ForeColor = Color.Red;
         g13.Text = "Battery 9 is damaged";

         g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
         p9.Value = 1;
      }
      else
      {
         g13.Refresh();
         g13.ForeColor = Color.Red;
         g13.Text = "Battery 9 is damaged";

         g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
         p9.Value = 2;
      }
```

```
}
if (channel1 >= 5.8)
{
   g13.Refresh();
   g13.Text = " ";


   double p = (.7632 * (channel1 * 2) - 8.7197) * 100 + 2.71;
   p = Math.Round(p, 2);
   current = 5.55555556 * (channel1 * 2);//(80A/14.4V)*nVolts=current
   v = Convert.ToInt32(current);
   t = (80 - v) / 3;//3~A/h considering }


   if (channel1 <= 6.267)
   {

      g17.Refresh();
      g17.ForeColor = Color.Red;
      g17.Text = "Low level";
      if (channel1 >= 5.8 && channel1 <= 6.0355)
      {
         p9.Value = 3;

      }
      else
      {
         p9.Value = 4;
      }
   }

   else if (channel1 <= 6.3 && channel1 > 6.267)
   {


      g17.Refresh();

      g17.ForeColor = Color.Orange;
      g17.Text = "mid level";

      if (channel1 > 6.267 && channel1 <= 6.3235)
      {
         p9.Value = 5;

      }
      else
      {
         p9.Value = 6;
```

```
    }
}
else
{
    g17.Refresh();
    g17.ForeColor = Color.Green;

    g17.Text = "High level";

    if (channel1 > 6.3 && channel1 <= 6.75)
    {
        p9.Value = 7;
        if (channel1 >= 6.35)
        {
            g13.Refresh();
            g13.ForeColor = Color.Green;
            g13.Text = "Battery is fully charged";
        }
        else
        {
            g13.Refresh();
            g13.Text = " ";
        }

    }
    else
    {
        p9.Value = 8;
        g13.Refresh();
        if (channel1 >= 7.2)
        {

            g13.Refresh();
            g13.ForeColor = Color.Red;
            g13.Text = "Battery is disconnected";

        }
        else
        {

            g13.Refresh();
            g13.ForeColor = Color.Green;
            g13.Text = "Battery is fully charged";
        }


    }

}
g1.Refresh();
```

```
    g2.Refresh();
    g3.Refresh();
    g1.Text = (channel1 * 2).ToString() + "Volts";
    if (channel1 >= 6.35)
    {
       g2.Text = "100 % Charged";
    }
    else
    {
       g2.Refresh();
       g2.Text = p.ToString() + "% Charged";
    }
    g3.Text = t.ToString() + "Hours Remaining";

    g1.Refresh();
    g2.Refresh();
    g3.Refresh();
    g13.Refresh();

}///channel 1 ends

if (channel2 < 5.8)
{
    gi.Refresh();
    g5.Refresh();
    g5.Text = " ";


    g6.Refresh();
    g6.Text = " ";
    g18.Refresh();
    g18.Text = " ";

    if (channel2 == 0)
    {
       g14.Refresh();
       g14.ForeColor = Color.Blue;
       g14.Text = "No battery is Connected at port 9";
       gi.Text = "0 Volts";
       p10.Value = 0;
    }
    else if (channel2 <= 2.9 && channel2 > 0)
    {
       g14.Refresh();
       g14.ForeColor = Color.Red;
       g14.Text = "Battery 10 is damaged";

       gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
       p10.Value = 1;
    }
```

```
    else
    {
        g14.Refresh();
        g14.ForeColor = Color.Red;
        g14.Text = "Battery 10 is damaged";

        gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
        p10.Value = 2;
    }


}
if (channel2 >= 5.8)
{
    g14.Refresh();
    g14.Text = " ";



    double pi = (.7632 * (channel2 * 2) - 8.7197) * 100 + 2.71;
    pi = Math.Round(pi, 2);
    current1 = 5.55555556 * (channel2 * 2);//(80A/14.4V)*nVolts=current
    v1 = Convert.ToInt32(current1);
    t1 = (80 - v1) / 3;//3~A/h considering }


    if (channel2 <= 6.267)
    {

        g18.Refresh();
        g18.ForeColor = Color.Red;
        g18.Text = "Low level";
        g14.Refresh();
        g14.Text = " ";
        if (channel2 >= 5.8 && channel2 <= 6.0355)
        {
            p10.Value = 3;

        }
        else
        {
            p10.Value = 4;

        }
    }

    else if (channel2 <= 6.3 && channel2 > 6.267)
    {
```

```
    g18.Refresh();

    g18.ForeColor = Color.Orange;
    g18.Text = "mid level";
    g14.Refresh();
    g14.Text = " ";

    if (channel2 > 6.267 && channel2 <= 6.2835)
    {
       p10.Value = 5;


    }
    else
    {
       p10.Value = 6;

    }
}
else
{
    g18.Refresh();
    g18.ForeColor = Color.Green;

    g18.Text = "High level";

    if (channel2 > 6.3 && channel2 <= 6.75)
    {
       p10.Value = 7;
       if (channel2 >= 6.35)
       {
          g14.Refresh();
          g14.ForeColor = Color.Green;
          g14.Text = "Battery is fully charged";
       }
       else
       {
          g14.Refresh();
          g14.Text = " ";
       }

    }
    else
    {
       p10.Value = 8;

       if (channel2 >= 7.2)
       {

          g14.Refresh();
```

```
            g14.ForeColor = Color.Red;
            g14.Text = "Battery is disconnected";
        }
        else
        {

            g14.Refresh();
            g14.ForeColor = Color.Green;

            g14.Text = "Battery is fully charged";
        }


    }

}
gi.Refresh();
g5.Refresh();
g6.Refresh();
gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
if (channel2 >= 6.35)
{
    g5.Text = "100 % Charged";
}
else
{
    g5.Refresh();
    g5.Text = pi.ToString() + "% Charged";
}
g6.Text = t1.ToString() + "Hours Remaining";

gi.Refresh();
g5.Refresh();
g6.Refresh();

}////////////channel 2 ends here
if (channel3 < 5.8)
{
    g7.Refresh();
    g8.Refresh();
    g8.Text = " ";


    g9.Refresh();
    g9.Text = " ";
    g19.Refresh();
    g19.Text = " ";

    if (channel3 == 0)
    {
```

```
            g15.Refresh();
            g15.ForeColor = Color.Blue;
            g15.Text = "No battery is Connected at port 10";
            g7.Text = "0 Volts";
            g19.Refresh();
            g19.Text = " ";
            p11.Value = 0;
        }
        else if (channel3 <= 2.9 && channel3 > 0)
        {
            g15.Refresh();
            g15.ForeColor = Color.Red;
            g19.Refresh();
            g19.Text = " ";
            g15.Text = "Battery 11 is damaged";

            g7.Text = Math.Round(channel3 * 2, 3).ToString() + "Volts";
            p11.Value = 1;
        }
        else
        {
            g15.Refresh();
            g15.ForeColor = Color.Red;
            g15.Text = "Battery 11 is damaged";

            g7.Text = Math.Round(channel3 * 2, 3).ToString() + "Volts";
            p11.Value = 2;
        }


    }
    if (channel3 >= 5.8)
    {
        g15.Refresh();
        g15.Text = " ";



        double pii = (.7632 * (channel3 * 2) - 8.7197) * 100 + 2.71;
        pii = Math.Round(pii, 2);
        current2 = 5.55555556 * (channel3 * 2);//(80A/14.4V)*nVolts=current
        v2 = Convert.ToInt32(current2);
        t2 = (80 - v2) / 3;//3~A/h considering }


        if (channel3 <= 6.267)
        {

            g19.Refresh();
            g19.ForeColor = Color.Red;
```

```
    g19.Text = "Low level";
    g15.Refresh();
    g15.Text = " ";
    if (channel3 >= 5.8 && channel3 <= 6.0355)
    {
        p11.Value = 3;

    }
    else
    {
        p11.Value = 4;

    }
}

else if (channel3 <= 6.3 && channel3 > 6.267)
{


    g19.Refresh();

    g19.ForeColor = Color.Orange;
    g19.Text = "mid level";
    g15.Refresh();
    g15.Text = " ";

    if (channel3 > 6.267 && channel3 <= 6.2835)
    {
        p11.Value = 5;


    }
    else
    {
        p11.Value = 6;

    }
}
else
{
    g19.Refresh();
    g19.ForeColor = Color.Green;

    g19.Text = "High level";

    if (channel3 > 6.3 && channel3 <= 6.75)
    {
        p11.Value = 7;
        if (channel3 >= 6.35)
        {
```

```
            g15.Refresh();
            g15.ForeColor = Color.Green;
            g15.Text = "Battery is fully charged";
        }
        else
        {
            g15.Refresh();
            g15.Text = " ";
        }

    }
    else
    {
        p11.Value = 8;

        if (channel3 >= 7.2)
        {
            g15.Refresh();
            g15.ForeColor = Color.Red;

            g15.Text = "Battery is disconnected";


        }
        else
        {

            g15.Refresh();
            g15.ForeColor = Color.Green;
            g15.Text = "Battery is fully charged";


        }


    }

}
g7.Refresh();
g8.Refresh();
g9.Refresh();
g7.Text = (channel3 * 2).ToString() + "Volts";

if (channel3 >= 6.35)
{
    g8.Text = "100 % Charged";
}
else
{
    g8.Refresh();
```

```
        g8.Text = pii.ToString() + "% Charged";
        g8.Refresh();
    }
    g9.Text = t2.ToString() + "Hours Remaining";

    g7.Refresh();
    g8.Refresh();
    g9.Refresh();

}
//channel 3 ends


if (channel4 < 5.8)
{
    g10.Refresh();
    g11.Refresh();
    g11.Text = " ";


    g12.Refresh();
    g12.Text = " ";
    g20.Refresh();
    g20.Text = " ";

    if (channel4 == 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Blue;
        g16.Text = "No battery is Connected at port 11";
        g10.Text = "0 Volts";
        g20.Refresh();
        g20.Text = " ";
        p12.Value = 0;
    }
    else if (channel4 <= 2.9 && channel4 > 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Red;
        g20.Refresh();
        g20.Text = " ";
        g16.Text = "Battery 12 is damaged";

        g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
        p12.Value = 1;
    }
    else
    {
        g16.Refresh();
        g16.ForeColor = Color.Red;
```

```
        g16.Text = "Battery 12 is damaged";

        g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
        p12.Value = 2;
    }


}
if (channel4 >= 5.8)
{
    g16.Refresh();
    g16.Text = " ";



    double piii = (.7632 * (channel4 * 2) - 8.7197) * 100 + 2.71;
    piii = Math.Round(piii, 2);
    current3 = 5.55555556 * (channel4 * 2);//(80A/14.4V)*nVolts=current
    v3 = Convert.ToInt32(current3);
    t3 = (80 - v3) / 3;//3~A/h considering }


    if (channel4 <= 6.267)
    {

        g20.Refresh();
        g20.ForeColor = Color.Red;
        g20.Text = "Low level";
        g16.Refresh();
        g16.Text = " ";
        if (channel4 >= 5.8 && channel4 <= 6.0355)
        {
            p12.Value = 3;

        }
        else
        {
            p12.Value = 4;

        }
    }

    else if (channel4 <= 6.3 && channel4 > 6.267)
    {


        g20.Refresh();

        g20.ForeColor = Color.Orange;
        g20.Text = "mid level";
```

```
    g16.Refresh();
    g16.Text = " ";

    if (channel4 > 6.267 && channel4 <= 6.2835)
    {
        p12.Value = 5;


    }
    else
    {
        p12.Value = 6;

    }
}
else
{
    g20.Refresh();
    g20.ForeColor = Color.Green;

    g20.Text = "High level";

    if (channel4 > 6.3 && channel4 <= 6.75)
    {
        p12.Value = 7;
        if (channel4 >= 6.35)
        {
            g16.Refresh();
            g16.ForeColor = Color.Green;
            g16.Text = "Battery is fully charged";
        }
        else
        {
            g16.Refresh();
            g16.Text = " ";
        }

    }
    else
    {
        p12.Value = 8;

        if (channel4 >= 7.2)
        {
            g16.Refresh();
            g16.ForeColor = Color.Red;

            g16.Text = "Battery is disconnected";
```

```
            }
            else
            {

               g16.Refresh();
               g16.ForeColor = Color.Green;
               g16.Text = "Battery is fully charged";

            }

         }

      }
      g10.Refresh();
      g11.Refresh();
      g12.Refresh();
      g10.Text = (channel4 * 2).ToString() + "Volts";

      if (channel4 >= 6.35)
      {
         g11.Text = "100 % Charged";
      }
      else
      {
         g11.Refresh();
         g11.Text = piii.ToString() + "% Charged";
         g11.Refresh();
      }
      g12.Text = t3.ToString() + "Hours Remaining";

      g10.Refresh();
      g11.Refresh();
      g12.Refresh();

   }
  }
}

private void button1_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)
{
```

```
        this.set = false;
        timer1.Stop();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        timer1.Stop();
        this.Close();
    }



  }
}
```

Code for the windows Form6 that has been initialized as f5:

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

namespace Solar_Battery_Charging_Status_Monitoring_Panel
{
  public partial class Form6 : Form
  {
    public Form6()
    {
      InitializeComponent();
    }

    private void Form6_Load(object sender, EventArgs e)
    {

    }
    private bool set = false;
    private double p = 0;
    private double current;
    private int v;
    private int t = 0;
    private double pi = 0;
    private double current1;
    private int v1;
    private int t1;
```

```
private int i = 0;
private double pii = 0;
private double current2;
private int v2;
private int t2;
private double piii = 0;
private double current3;
private int v3;
private int t3;

private void cmdSelectDevice_Click(object sender, EventArgs e)
{
   // selecting device
   axAdvAI1.SelectDevice();
   txtDeviceName.Text = axAdvAI1.DeviceName;
}


private void timer1_Tick(object sender, EventArgs e)
{
   axAdvAI1.ChannelNow = 12;


   double channel1 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 13;


   double channel2 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 14;
   double channel3 = Math.Round(axAdvAI1.DataAnalog, 3);

   axAdvAI1.ChannelNow = 15;
   double channel4 = Math.Round(axAdvAI1.DataAnalog, 3);



   if (channel1 < 5.8 && channel2 < 5.8 && channel3 < 5.8 && channel4 < 5.8)
   {

      g13.Refresh();
      g14.Refresh();
      g15.Refresh();
      g16.Refresh();
      g1.Refresh();
      g2.Refresh();
      g3.Refresh();
```

```
g7.Refresh();
g8.Refresh();
g9.Refresh();
g17.Refresh();



gi.Refresh();
g5.Refresh();
g6.Refresh();



g10.Refresh();
g11.Refresh();
g12.Refresh();


g2.Text = " ";
g3.Text = " ";



g5.Text = " ";
g6.Text = " ";

g8.Text = " ";
g9.Text = " ";

g11.Text = " ";
g12.Text = " ";
g17.Text = " ";
g18.Refresh();
g18.Text = " ";
g19.Refresh();
g19.Text = " ";

g20.Refresh();
g20.Text = " ";



if (channel1 == 0)
{
   g13.Refresh();
   g13.ForeColor = Color.Blue;
```

```
    g13.Text = "No battery is Connected at port 12";
    g1.Text = "0 Volts";
    p13.Value = 0;
}
else if (channel1 <= 2.9 && channel1 > 0)
{
    g13.Refresh();
    g13.ForeColor = Color.Red;
    g13.Text = "Battery 13 is damaged";
    g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
    p13.Value = 1;
}
else
{
    g13.Refresh();
    g13.ForeColor = Color.Red;
    g13.Text = "Battery 13 is damaged";
    g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
    p13.Value = 2;
}// channel 1 progress bar

if (channel2 == 0)
{
    g14.Refresh();
    g14.ForeColor = Color.Blue;
    g14.Text = "No battery is Connected at port 13";
    gi.Text = " 0 Volts";
    p14.Value = 0;
}
else if (channel2 <= 2.9 && channel2 > 0)
{
    g14.Refresh();
    g14.ForeColor = Color.Red;
    g14.Text = "Battery 14 is damaged!";
    gi.Text = Math.Round((channel2 * 2), 3).ToString() + "Volts";
    p14.Value = 1;
}
else
{
    g14.Refresh();
    g14.ForeColor = Color.Red;
    g14.Text = "Battery 14 is damaged!";
    gi.Text = Math.Round((channel2 * 2), 3).ToString() + "Volts";
    p14.Value = 2;
}//channel 2 progress bar

if (channel3 == 0)
{
    g15.Refresh();
    g15.ForeColor = Color.Blue;
```

```
    g15.Text = "No battery is Connected at port 14";
    g7.Text = "0 Volts";
    p15.Value = 0;
}
else if (channel3 <= 2.9 && channel3 > 0)
{
    g15.Refresh();
    g15.ForeColor = Color.Red;
    g15.Text = "Battery 15 is damaged!";
    g7.Text = Math.Round((channel3 * 2), 3).ToString() + "Volts";
    p15.Value = 1;
}
else
{
    g15.Refresh();
    g15.ForeColor = Color.Red;
    g15.Text = "Battery 15 is damaged!";
    g7.Text = Math.Round((channel3 * 2), 3).ToString() + "Volts";
    p15.Value = 2;
}//Channel 3 reading progress bar

if (channel4 == 0)
{
    g16.Refresh();
    g16.ForeColor = Color.Blue;
    g16.Text = "No Battery is Connected at port 15";
    g10.Text = "0 Volts";
    p16.Value = 0;
}
else if (channel4 <= 2.9 && channel4 > 0)
{
    g16.Refresh();
    g16.ForeColor = Color.Red;
    g16.Text = "Battery 16 is damaged!";
    g10.Text = Math.Round((channel4 * 2), 3).ToString() + "Volts";
    p16.Value = 1;
}
else
{
    g16.Refresh();
    g16.ForeColor = Color.Red;
    g16.Text = "Battery 16 is damaged!";
    g10.Text = Math.Round((channel4 * 2), 3).ToString() + "Volts";
    p16.Value = 2;
}// channel 4 reading progress bar
```

```
}
else
{
   if (channel1 < 5.8)
   {
      g1.Refresh();
      g17.Refresh();
      g17.Text = " ";


      g2.Refresh();
      g2.Text = " ";
      g3.Refresh();
      g3.Text = " ";

      if (channel1 == 0)
      {
         g13.Refresh();
         g13.ForeColor = Color.Blue;
         g13.Text = "No battery is Connected at port 12";
         g1.Text = "0 Volts";
         p13.Value = 0;
      }
      else if (channel1 <= 2.9 && channel1 > 0)
      {
         g13.Refresh();
         g13.ForeColor = Color.Red;
         g13.Text = "Battery 13 is damaged";

         g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
         p13.Value = 1;
      }
      else
      {
         g13.Refresh();
         g13.ForeColor = Color.Red;
         g13.Text = "Battery 13 is damaged";

         g1.Text = Math.Round((channel1 * 2), 3).ToString() + "Volts";
         p13.Value = 2;
      }


   }
   if (channel1 >= 5.8)
   {
      g13.Refresh();
      g13.Text = " ";
```

```
double p = (.7632 * (channel1 * 2) - 8.7197) * 100 + 2.71;
p = Math.Round(p, 2);
current = 5.55555556 * (channel1 * 2);//(80A/14.4V)*nVolts=current
v = Convert.ToInt32(current);
t = (80 - v) / 3;//3~A/h considering }


if (channel1 <= 6.267)
{

   g17.Refresh();
   g17.ForeColor = Color.Red;
   g17.Text = "Low level";
   if (channel1 >= 5.8 && channel1 <= 6.0355)
   {
      p13.Value = 3;

   }
   else
   {
      p13.Value = 4;
   }
}

else if (channel1 <= 6.3 && channel1 > 6.267)
{


   g17.Refresh();

   g17.ForeColor = Color.Orange;
   g17.Text = "mid level";

   if (channel1 > 6.267 && channel1 <= 6.3235)
   {
      p13.Value = 5;

   }
   else
   {
      p13.Value = 6;

   }
}
else
{
   g17.Refresh();
```

```
    g17.ForeColor = Color.Green;

    g17.Text = "High level";

    if (channel1 > 6.3 && channel1 <= 6.75)
    {
       p13.Value = 7;
       if (channel1 >= 6.35)
       {
          g13.Refresh();
          g13.ForeColor = Color.Green;
          g13.Text = "Battery is fully charged";
       }
       else
       {
          g13.Refresh();
          g13.Text = " ";
       }

    }
    else
    {
       p13.Value = 8;
       g13.Refresh();
       if (channel1 >= 7.2)
       {

          g13.Refresh();
          g13.ForeColor = Color.Red;
          g13.Text = "Battery is disconnected";

       }
       else
       {

          g13.Refresh();
          g13.ForeColor = Color.Green;
          g13.Text = "Battery is fully charged";
       }


    }

}
g1.Refresh();
g2.Refresh();
g3.Refresh();
g1.Text = (channel1 * 2).ToString() + "Volts";
if (channel1 >= 6.35)
{
```

```
            g2.Text = "100 % Charged";
        }
        else
        {
            g2.Refresh();
            g2.Text = p.ToString() + "% Charged";
        }
        g3.Text = t.ToString() + "Hours Remaining";

        g1.Refresh();
        g2.Refresh();
        g3.Refresh();
        g13.Refresh();

    }///channel 1 ends

    if (channel2 < 5.8)
    {
        gi.Refresh();
        g5.Refresh();
        g5.Text = " ";


        g6.Refresh();
        g6.Text = " ";
        g18.Refresh();
        g18.Text = " ";

        if (channel2 == 0)
        {
            g14.Refresh();
            g14.ForeColor = Color.Blue;
            g14.Text = "No battery is Connected at port 13";
            gi.Text = "0 Volts";
            p14.Value = 0;
        }
        else if (channel2 <= 2.9 && channel2 > 0)
        {
            g14.Refresh();
            g14.ForeColor = Color.Red;
            g14.Text = "Battery 14 is damaged";

            gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
            p14.Value = 1;
        }
        else
        {
            g14.Refresh();
            g14.ForeColor = Color.Red;
            g14.Text = "Battery 14 is damaged";
```

```
        gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
        p14.Value = 2;
    }


}
if (channel2 >= 5.8)
{
    g14.Refresh();
    g14.Text = " ";



    double pi = (.7632 * (channel2 * 2) - 8.7197) * 100 + 2.71;
    pi = Math.Round(pi, 2);
    current1 = 5.55555556 * (channel2 * 2);//(80A/14.4V)*nVolts=current
    v1 = Convert.ToInt32(current1);
    t1 = (80 - v1) / 3;//4.19~A/h considering }


    if (channel2 <= 6.267)
    {

        g18.Refresh();
        g18.ForeColor = Color.Red;
        g18.Text = "Low level";
        g14.Refresh();
        g14.Text = " ";
        if (channel2 >= 5.8 && channel2 <= 6.0355)
        {
            p14.Value = 3;

        }
        else
        {
            p14.Value = 4;

        }
    }

    else if (channel2 <= 6.3 && channel2 > 6.267)
    {


        g18.Refresh();

        g18.ForeColor = Color.Orange;
        g18.Text = "mid level";
        g14.Refresh();
```

```
    g14.Text = " ";

    if (channel2 > 6.267 && channel2 <= 6.2835)
    {
       p14.Value = 5;


    }
    else
    {
       p14.Value = 6;

    }
}
else
{
    g18.Refresh();
    g18.ForeColor = Color.Green;

    g18.Text = "High level";

    if (channel2 > 6.3 && channel2 <= 6.75)
    {
       p14.Value = 7;
       if (channel2 >= 6.35)
       {
          g14.Refresh();
          g14.ForeColor = Color.Green;
          g14.Text = "Battery is fully charged";
       }
       else
       {
          g14.Refresh();
          g14.Text = " ";
       }

    }
    else
    {
       p14.Value = 8;

       if (channel2 >= 7.2)
       {

          g14.Refresh();
          g14.ForeColor = Color.Red;
          g14.Text = "Battery is disconnected";
       }
       else
       {
```

```
            g14.Refresh();
            g14.ForeColor = Color.Green;

            g14.Text = "Battery is fully charged";
          }


        }

      }
      gi.Refresh();
      g5.Refresh();
      g6.Refresh();
      gi.Text = Math.Round(channel2 * 2, 3).ToString() + "Volts";
      if (channel2 >= 6.35)
      {
        g5.Text = "100 % Charged";
      }
      else
      {
        g5.Refresh();
        g5.Text = pi.ToString() + "% Charged";
      }
      g6.Text = t1.ToString() + "Hours Remaining";

      gi.Refresh();
      g5.Refresh();
      g6.Refresh();

    }//////////channel 2 ends here
    if (channel3 < 5.8)
    {
      g7.Refresh();
      g8.Refresh();
      g8.Text = " ";


      g9.Refresh();
      g9.Text = " ";
      g19.Refresh();
      g19.Text = " ";

      if (channel3 == 0)
      {
        g15.Refresh();
        g15.ForeColor = Color.Blue;
        g15.Text = "No battery is Connected at port 14";
        g7.Text = "0 Volts";
        g19.Refresh();
```

```
            g19.Text = " ";
            p15.Value = 0;
        }
        else if (channel3 <= 2.9 && channel3 > 0)
        {
            g15.Refresh();
            g15.ForeColor = Color.Red;
            g19.Refresh();
            g19.Text = " ";
            g15.Text = "Battery 15 is damaged";

            g7.Text = Math.Round(channel3 * 2, 3).ToString() + "Volts";
            p15.Value = 1;
        }
        else
        {
            g15.Refresh();
            g15.ForeColor = Color.Red;
            g15.Text = "Battery 15 is damaged";

            g7.Text = Math.Round(channel3 * 2, 3).ToString() + "Volts";
            p15.Value = 2;
        }


    }
    if (channel3 >= 5.8)
    {
        g15.Refresh();
        g15.Text = " ";



        double pii = (.7632 * (channel3 * 2) - 8.7197) * 100 + 2.71;
        pii = Math.Round(pii, 2);
        current2 = 5.55555556 * (channel3 * 2);//(80A/14.4V)*nVolts=current
        v2 = Convert.ToInt32(current2);
        t2 = (80 - v2) / 3;//3~A/h considering }


        if (channel3 <= 6.267)
        {

            g19.Refresh();
            g19.ForeColor = Color.Red;
            g19.Text = "Low level";
            g15.Refresh();
            g15.Text = " ";
            if (channel3 >= 5.8 && channel3 <= 6.0355)
            {
```

```
        p15.Value = 3;

    }
    else
    {
        p15.Value = 4;

    }
}

else if (channel3 <= 6.3 && channel3 > 6.267)
{


    g19.Refresh();

    g19.ForeColor = Color.Orange;
    g19.Text = "mid level";
    g15.Refresh();
    g15.Text = " ";

    if (channel3 > 6.267 && channel3 <= 6.2835)
    {
        p15.Value = 5;


    }
    else
    {
        p15.Value = 6;

    }
}
else
{
    g19.Refresh();
    g19.ForeColor = Color.Green;

    g19.Text = "High level";

    if (channel3 > 6.3 && channel3 <= 6.75)
    {
        p15.Value = 7;
        if (channel3 >= 6.35)
        {
            g15.Refresh();
            g15.ForeColor = Color.Green;
            g15.Text = "Battery is fully charged";
        }
        else
```

```
        {
            g15.Refresh();
            g15.Text = " ";
        }

    }
    else
    {
        p15.Value = 8;

        if (channel3 >= 7.2)
        {
            g15.Refresh();
            g15.ForeColor = Color.Red;

            g15.Text = "Battery is disconnected";


        }
        else
        {

            g15.Refresh();
            g15.ForeColor = Color.Green;
            g15.Text = "Battery is fully charged";


        }


    }

}
g7.Refresh();
g8.Refresh();
g9.Refresh();
g7.Text = (channel3 * 2).ToString() + "Volts";

if (channel3 >= 6.35)
{
    g8.Text = "100 % Charged";
}
else
{
    g8.Refresh();
    g8.Text = pii.ToString() + "% Charged";
    g8.Refresh();
}
g9.Text = t2.ToString() + "Hours Remaining";
```

```
    g7.Refresh();
    g8.Refresh();
    g9.Refresh();

}
//channel 3 ends

if (channel4 < 5.8)
{
    g10.Refresh();
    g11.Refresh();
    g11.Text = " ";


    g12.Refresh();
    g12.Text = " ";
    g20.Refresh();
    g20.Text = " ";

    if (channel4 == 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Blue;
        g16.Text = "No battery is Connected at port 15";
        g10.Text = "0 Volts";
        g20.Refresh();
        g20.Text = " ";
        p16.Value = 0;
    }
    else if (channel4 <= 2.9 && channel4 > 0)
    {
        g16.Refresh();
        g16.ForeColor = Color.Red;
        g20.Refresh();
        g20.Text = " ";
        g16.Text = "Battery 16 is damaged";

        g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
        p16.Value = 1;
    }
    else
    {
        g16.Refresh();
        g16.ForeColor = Color.Red;
        g16.Text = "Battery 16 is damaged";

        g10.Text = Math.Round(channel4 * 2, 3).ToString() + "Volts";
        p16.Value = 2;
    }
}
```

```
if (channel4 >= 5.8)
{
   g16.Refresh();
   g16.Text = " ";


   double piii = (.7632 * (channel4 * 2) - 8.7197) * 100 + 2.71;
   piii = Math.Round(piii, 2);
   current3 = 5.55555556 * (channel4 * 2);//(80A/14.4V)*nVolts=current
   v3 = Convert.ToInt32(current3);
   t3 = (80 - v3) / 3;//3~A/h considering }


   if (channel4 <= 6.267)
   {

      g20.Refresh();
      g20.ForeColor = Color.Red;
      g20.Text = "Low level";
      g16.Refresh();
      g16.Text = " ";
      if (channel4 >= 5.8 && channel4 <= 6.0355)
      {
         p16.Value = 3;

      }
      else
      {
         p16.Value = 4;

      }
   }

   else if (channel4 <= 6.3 && channel4 > 6.267)
   {

      g20.Refresh();

      g20.ForeColor = Color.Orange;
      g20.Text = "mid level";
      g16.Refresh();
      g16.Text = " ";

      if (channel4 > 6.267 && channel4 <= 6.2835)
      {
         p16.Value = 5;


      }
```

```
    else
    {
        p16.Value = 6;

    }
}
else
{
    g20.Refresh();
    g20.ForeColor = Color.Green;

    g20.Text = "High level";

    if (channel4 > 6.3 && channel4 <= 6.75)
    {
        p16.Value = 7;
        if (channel4 >= 6.35)
        {
            g16.Refresh();
            g16.ForeColor = Color.Green;
            g16.Text = "Battery is fully charged";
        }
        else
        {
            g16.Refresh();
            g16.Text = " ";
        }

    }
    else
    {
        p16.Value = 8;

        if (channel4 >= 7.2)
        {
            g16.Refresh();
            g16.ForeColor = Color.Red;

            g16.Text = "Battery is disconnected";

        }
        else
        {
          g16.Refresh();
            g16.ForeColor = Color.Green;
            g16.Text = "Battery is fully charged";

        }
    }
}
```

```
            g10.Refresh();
            g11.Refresh();
            g12.Refresh();
            g10.Text = (channel4 * 2).ToString() + "Volts";

            if (channel4 >= 6.35)
            {
               g11.Text = "100 % Charged";
            }
            else
            {
               g11.Refresh();
               g11.Text = piii.ToString() + "% Charged";
               g11.Refresh();
            }
            g12.Text = t3.ToString() + "Hours Remaining";

            g10.Refresh();
            g11.Refresh();
            g12.Refresh();

         }

      }
   }


   private void button1_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)
   {
      this.set = false;
      timer1.Stop();
   }

   private void button3_Click(object sender, EventArgs e)
   {
      timer1.Stop();
      this.Close();
   }

}}
```

TABLE LIST

FIGURE LIST