National University of Computer and

Emerging Sciences

Final Year Project Thesis

Intelligent Unit of Hybrid System

Authors: Supervisor:

Arsalan Saghir (i110466) Engr. Aamer Munir

Arsalan Ahmed (i110504)

Department of Electrical Engineering

June 2015



**Developer’s Submission**

This report is being submitted to the Department of Electrical Engineering of the National University of Computer and Emerging Sciences in partial fulfillment of the requirements for the degree of BS in Electrical Engineering.

ii

**Developer’s Declaration**

“We take full responsibility of the project work conducted during the Final Year Project (FYP) titled “Intelligent Unit of Hybrid System”. We solemnly declare that the project work presented in the FYP report is done solely by us with no significant help from any other person; however, small help wherever taken is duly acknowledged. We have also written the complete FYP report by ourselves. Moreover, we have not presented this FYP (or substantially similar project work) or any part of the thesis previously to any other degree awarding institution within Pakistan or abroad.

We understand that the management of Department of Electrical Engineering of National University of Computer and Emerging Sciences has a zero tolerance policy towards plagiarism. Therefore, we as an author of the above-mentioned FYP report solemnly declare that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced. Moreover, the report does not contain any literal citing of more than 70 words (total) even by giving a reference unless we have obtained the written permission of the publisher to do so. Furthermore, the work presented in the report is our own work and we have positively cited the related work of the other projects by clearly differentiating our work from their relevant work.

We further understand that if we are found guilty of any form of plagiarism in our FYP report even after our graduation, the University reserves the right to withdraw our BS degree. Moreover, the University will also have the right to publish our names on its website that keeps a record of the students who committed plagiarism in their FYP reports.”

iii

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Name Student Name

BS(EE) Batch-Roll # BS(EE) Batch-Roll #

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Certified by Supervisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Verified by Plagiarism Cell Officer

Dated: \_\_\_\_\_\_\_\_\_\_\_

iv

**Abstract**

We were required to program a microcontroller for a hybrid system. Task of microcontroller was to do efficient switching between different available resources.  
By using Arduino DUE, power meters and switches we made an efficient hybrid system. This hybrid system utilizes intermittent source to its fullest. We were required to make our hybrid system more efficient, reliable and cost effective. This system needs minimal interaction with its user for modifications. The system designed is also capable of tackling load shedding conditions, what it does is it created loadsheding scheduler along with battery level at that time and by following an algorithm for scheduler it counter loadsheding condition by optimally charging the battery. System is also capable of data acquisition of incoming power resources with respect to time so that we do further analysis of hybrid system and analysis of when which and how much power source is available. For viewing the data from data logger a software on Matlab was designed that reads data from SD card and display graph of power values of different power resources, so now we can say it’s a step towards a smart home.

v

Table of Contents

Chapter 1

Introduction

**1.1 Background Information** 1

1.1.1 Why Was this Project Appealing? 1

1.1.2 Overview 1

1.1.3 Problem Statement2

Chapter 2

Solution/Design/Implementation

**2.**1 Division of Projects 3

2.1.1 Simulation of Hybrid System 3

2.1.2 Integration of Single Phase Hybrid System3

2.1.3 Intelligent Unit of Hybrid System3

2.1.4 Construction and Implementation of Load Unit4

# 2.2 Research 4

# Our Hybrid System 5

# 2.4 Selection of Microcontroller7

# 2.5 Real Time condition for decision 8

Case 18

Case 28

Case 39

Case 49

Case 59

Case 69

Case 79

# 2.6 Real Time Clock Module 12

# 2.7 SD card Module 12

# 2.8 Graphical User Interface13

# 2.9 Announced Load Shedding19

# 2.10 Problems Faced21

Chapter 3

**(Results/Conclusions/ Recommendations)**

# 3.1 Results 22

# 3.2 Recommendations23

References24

Appendix A24

Appendix B26

List of Figures

vi

Figure 2.1 Block Diagram 5

Figure 2.2 Intelligent Unit of Hybrid System 6

Figure 2.3 Arduino DUE 8

Figure 2.4 Unannounced Power Outage11

Figure 2.5 RTC DS1307 12

Figure 2.6 SD Card Modules12

Figure 2.7 SD card Process13

Figure 2.8 GUI Show all14

Figure 2.9 GUI grids Voltage15

Figure 2.10 GUI Solar Power16

Figure 2.11 GUI Battery levels17

Figure 2.12 GUI Load Consumption levels18

Figure 2.13 Announced power outage20

Figure 2.14 Serial Monitoring of Load Shedding Schedule21

List of Tables

vi

Table 2.1 Unannounced Load Shedding Switching Conditions10

vii

**Chapter 1 (Introduction)**

* 1. **Background Information**

1. **Why Was this Project Appealing?**

As we are progressing towards a modern era, the need for green energy is ever increasing. For this purpose Hybrid Systems are used which converts solar power into electric power. But unfortunately the available commercial hybrid systems are not as efficient as they should have been. Therefore we saw in this project a great opportunity to contribute to the hybrid industry by implementing a smart and efficient hybrid system.

1. **Overview**

Hybrid systems are those systems that can change an intermittent source of energy into electrical energy. As these sources are not available continuously therefore they can be only used to charge the battery and supply load for some time of the day. When these sources are not available then either battery or grid may supply power to our load.

As we all know that there are many types of commercial hybrid system present in the market. But most of those are dumb devices i.e. they don’t have the capability of making a decision at any level. They are hard coded to charge the battery if battery fall short of 100% by any means. Therefore these systems do not take into consideration the availability of solar energy, or the charge cycles of the battery.

As most of the commercially available hybrid systems were inefficient, so we proposed to make an intelligent unit of hybrid system, which will make a hybrid system more efficient. For this we would have to consider availability of the solar energy and maximize the utilization of it, furthermore our system will take into consideration the charge cycles of the battery so as to increase the life of the battery. It will work on real life scenarios of real life and will maximize the use of the

Intermittent source. It will enable the user to monitor the power values of Charge controller, Battery level, and Grid voltage and Load consumption. So that the user can track all of the information and see the performance of each and every device.

Chapter 1 Introduction

**1.3 Problem Statements**

We had to develop a system that would:

* Able to tell what energy sources are currently present.
* Based on the availability of sources it would be able to calculate the best and most efficient course of action.
* Be able to present power values to the user
* Have a database of all the previous values of incoming and outgoing power
* Be able to adjust itself according to the changing time of load shedding.
* Make the best use of battery and solar panels.

**Chapter 2 (Solution/Design/Implementation)**

# Division of Project

The production of product of hybrid system was divided into 4 parts.

1. Integration of a single phase hybrid system.
2. Intelligent unit of hybrid System.
3. Construction and implementation of Load Unit.
4. Simulation of Hybrid System
5. **Simulation of Hybrid System**

To check the optimal conditions on which our hybrid system would work, simulations were made on software. These simulation were there to prove that our hybrid system was efficient and low cost then the rest of commercially available hybrid systems.

1. **Integration of Single Phase Hybrid System**

This group had to make a customized hybrid system. This hybrid system would be able to send its power values of charge controller, inverter, and AC charger to the intelligent unit of hybrid system. So this group had to make power meters which could easily measure AC and DC values. Furthermore this group had to make different switches that would enable them to control the input/output of charge controller, inverter and AC charger.

1. **Intelligent Unit of Hybrid System**

To enable the hybrid system to take decisions based on the available sources, an intelligent unit was needed to be installed. This intelligent unit would control all of the decisions of the hybrid system. Furthermore it will also enable the user to see the power values of different devices and will be able to communicate with the user by using a keypad and LCD.

**2.1.4 Construction and Implementation of Load Unit**

In this project a variable load unit was to be made that could communicate with our Hybrid System. In this project the intelligent unit was to communicate with the dynamic load unit, and if we would achieve surplus energy than hybrid system would supply power to the dynamic load unit.

Chapter 2 Solution, Implementation, Design

# Intelligent Unit of Hybrid System

# Research

Doing research about the hybrid system marked the start of our project. First of all we had to know the basics of a hybrid system. Then after that we had to survey different commercially available hybrid systems. For links refer to appendix.

1. After the survey we found out that most of the hybrid systems were only acting as modified UPS systems. These hybrid systems were charging batteries in day in case of solar powered hybrid systems and then giving the stored power when the grid was down or when solar energy was not available.
2. None of the systems were considering the amount of charge cycles that were being wasted and also they were not considering the price impact of using the solar systems.
3. These systems were also using the solar energy when battery was charging or when the load could be supplied by our solar energy. The rest of the time solar energy was being wasted.

These results lead us to following improvements that must be made in a hybrid system

* A hybrid system should consider the charge cycle of the battery. This will enable us to maximize the life of the battery.
* A hybrid system should have a plan of action which it should follow in real life to minimize the cost per unit hour. Instead of blindly stuck on two or three condition.
* A hybrid system should adapt itself as the requirement around it changes.
* A hybrid system should have minimal human interaction.
* It should be able to show the different powers of incoming sources, batteries and load. So that the user can compare these values to see which is more efficient with respect to time.

Intelligent Unit of Hybrid System

# Our Hybrid System

After extensive research we finalized our block diagram which would enable us to complete all our goals. Figure 2.1 shows the block diagram of our project. In this block diagram we had 2 types of wires that were connected to our intelligent unit. First are the inputs of intelligent unit that constitute of output of power meters these power meter were sending analogue values from 0-3.3V which were then read by our microcontroller. Inputs also included values coming from the keypad to our controller. Second are the outputs, which constitute of switches used for switching our hybrid system’s various parts, furthermore the output from microcontroller is used to display text on LCD.

Figure 2.1 Block Diagram

In the figure 2.2 we see that there are 6 variable resisters present at the upper region of diagram. These variable resister corresponds to the different sensors that give us the value of current and voltages from charge controller, load, grid and battery. Our microcontroller depending on the incoming power, switches on or switches off our charge controller battery, grid and/or inverter shown in the diagram in the form of LEDs. The incoming powers can be seen in the Liquid Crystal Display. And these values are stored in the form of an excel sheet by our SD card module with respect to the real time by our RTC.

Chapter 2 Solution, Implementation, Design

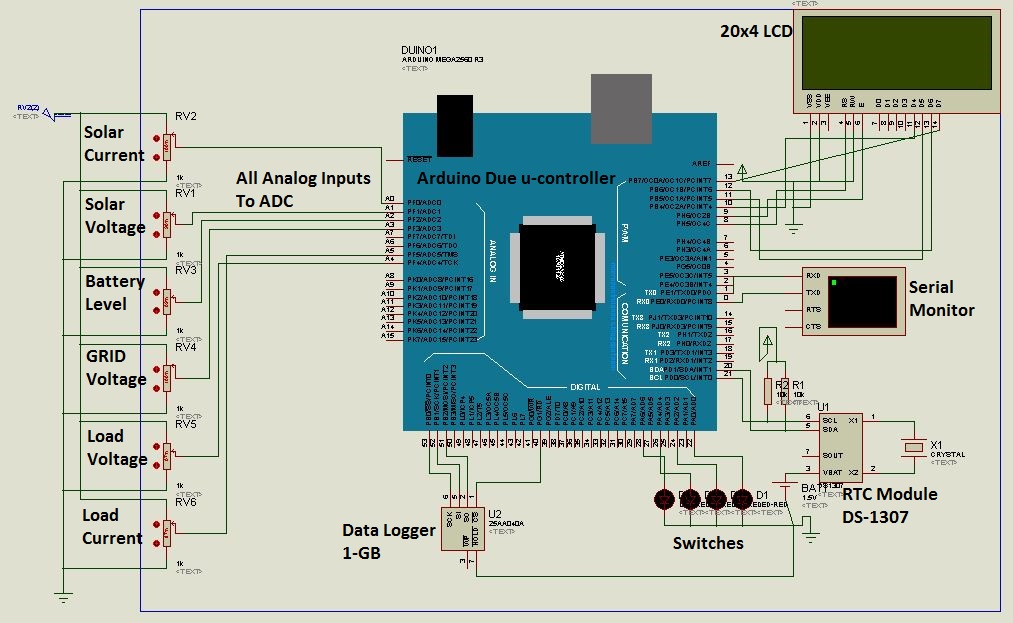


Figure 2.2 Intelligent Unit of Hybrid System

# Selection of Microcontroller

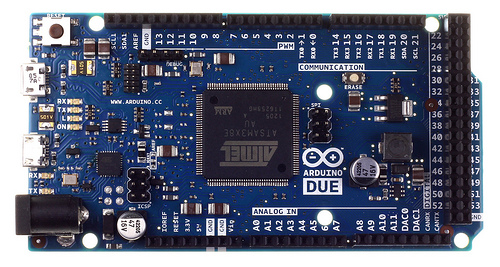
Intelligent Unit of Hybrid System

For our intelligent unit we needed to consider the inputs, outputs and the minimum speed that can easily sustain our computations without giving a delay that would affect our decision time. So the list of inputs and outputs we required from the microcontroller were:

* For inputs we required analogue pins which could vary their voltages. And as we required 7 of these corresponding to the different voltages and currents incoming from the power meters and required
* 11 digital pins for showing text on Liquid Crystal Display
* 7 digital pins for keypad
* 2 I2C, 1 for data logging, other for communication with load unit microcontroller
* 1 SPI for load unit microcontroller communication

Depending on these conditions we finalized Arduino DUE as our microcontroller board. Its main features are:

* DUE has a 32bit ARM core that can outperform a typical 8-bit microcontroller.
* A 32-bit core that allows operations on 4 bytes wide data within a single CPU clock.
* The microcontroller used in DUE is AT91SAM3X8E
* Operating voltage 3.3 V
* Input Voltage 7-12V
* Digital I/O pins 54
* Analogue pins 12
* Current for 3.3 V pins output 800mA
* DC current for 5 V 800mA
* Flash memory 512KB
* flash Memory 96KB
* Clock Speed 84 MHz



Chapter 2 Solution, Implementation, Design

Figure 2.3 Arduino DUE

# Real Time condition for decision

Our studies concluded that there were four variables that would play a huge role in deciding real time scenarios for our hybrid system. These four variables were:

1. Charge Controller
2. Battery level
3. Grid Voltage
4. Load Consumption

These four were the deciding factor upon which we built our real time scenarios. By considering these we were able to design cases which covered our daily life.

**Case 1:**

This case can be corresponded to, when the solar energy is available in ample amounts and load consumption is less than solar energy produced. Furthermore battery is greater than critical value but quite close to full battery. In this case our solar power is enough to charge the battery and also supply the load. So we will turn on the charge controller along with the inverter and keep the AC charger and grid switches off.

**Case 2:**

In this case the solar power available is less than the actual power required by the load, grid is available in this case and also battery is greater than the critical value. In this case as battery is greater than the critical point. So this means that it can supply the load. So we will enable the inverter so it can supply the load, charge controller is enabled to charge the battery if some solar power is available and grid is turned off.

**Case 3:**

Intelligent Unit of Hybrid System

In this case the solar power available is less than the actual power required by the load, grid is available in this case and also battery is less than the critical value. In this case as battery is less than the critical value. So we will enable the grid to supply the load. And the charge controller is turned on so it can charge the battery.

**Case 4:**

This case can be corresponded to, when the solar energy is available in ample amounts and load consumption is less than solar energy produced. Furthermore battery is less than critical value and grid is not available. In this case we will enable our charge controller and inverter. So that the charge controller can charge the battery and also supply the load.

**Case 5:**

In this case the solar power available is less than the actual power required by the load, grid is not available in this case and also battery is greater than the critical value. In this case as battery is greater than the critical value than we will supply battery to our load and charge controller will be turned on. For the case that if the solar power comes than it will charge our battery.

**Case 6:**

In this case the solar power available is greater than the actual power required by the load, grid is available in this case but battery is less than the critical value. In this case as battery is less than the critical value so we need to charge it. We will charge our battery by the surplus solar power that is available. And enable our inverter so it can also supply our load.

**Case 7:**

In this case the solar power available is less than the actual power required by the load, grid is not available in this case and also battery is less than the critical value. In this case as we have no available source. For this case our hybrid system will shut down after enabling the charge controller.

Chapter 2 Solution, Implementation, Design

All of these conditions can be seen tabulated below in table 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Conditions** | **GRID** | **AC Charger** | **Charge Controller** | **Inverter** |
| Charge Controller Power>load consumption  Battery Level >Critical Value  Grid > 0 | OFF | OFF | ON | ON |
| Charge Controller Power<load consumption  Grid > Min. Grid Volt.  Grid < Max. Grid Volt. |  |  |  |  |
| Battery Level > Minimum\_Battery\_level | ON | OFF | ON | OFF |
| Battery Level <= Minimum\_Battery\_level | ON | ON | OFF | OFF |
| Charge Controller Power>load consumption  Battery Level <Critical Value  Grid Not Available | OFF | OFF | ON | ON |
| Charge Controller Power<load consumption   Battery Level <Critical Value  Grid Not Available | OFF | OFF | ON | OFF |
| Charge Controller Power>load consumption   Battery Level <Critical Value  Grid > Min. Grid Volt. Grid < Max. Grid Volt | ON | OFF | ON | OFF |
| Charge Controller Power<load consumption  Battery Level >Critical Value  Grid Not Available | OFF | OFF | ON | ON |

Table 2.1 Unannounced Load Shedding Switching Conditions

Intelligent Unit of Hybrid System

We can see these conditions in the form of flow chart in Figure 2.4



Figure 2.4 Unannounced Power Outage

# Real time Clock Module

Chapter 2 Solution, Implementation, Design

For time based module we used a tiny RTC which enabled us to take real time clock data. This data is then used for logging our data with respect to time. It enabled us to log current time as well as give us future values so that we can perform operations on them. Figure depicts our RTC that we have used.

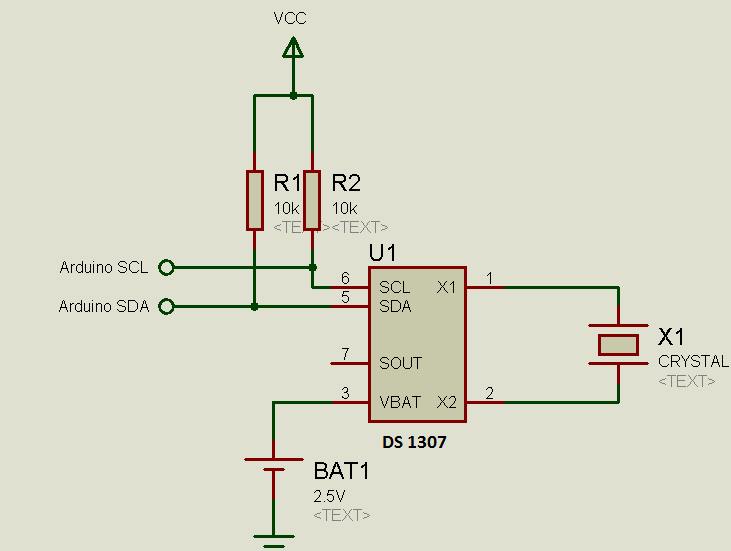


Figure 2.5 RTC DS1307

# SD Card Module

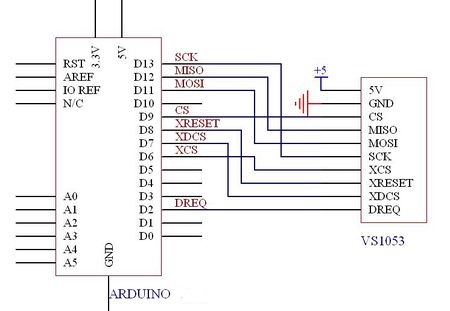
For the convenience of the user and also to perform some operation (that will be mentioned after) a SD card module was used. The SD card will store all of the data of the incoming power values of our grid and solar panel. Also it will enable us to store the battery level and the consumption of load. These reading will be stored on the SD card in the form of a “.CSV” format of excel sheet. The used SD card module is shown below in figure.

Figure 2.6 SD Card Module

Intelligent Unit of Hybrid System

The procedure a SD card module follows can be shown by the flow chart below



Figure 2.7 SD card Process

# Graphical User Interface

For the convenience of the user a Graphical User Interface was designed and implemented. This GUI was designed in MATLAB. This GUI picks up the data of the SD card when it is connected to a laptop or PC. It first translate the comma separated values to a Matrix and then these values can be saved into different vectors for the multiple operations that we require for our GUI. Given below is the GUI which not only can show the grid voltage, load consumption, solar power and battery level with respect to real time in a single graph but can also show each and every individual variable with respect to time in the same GUI with the push of a button. Given below are the figures of GUI for each and every case.

Chapter 2 Solution, Implementation, Design

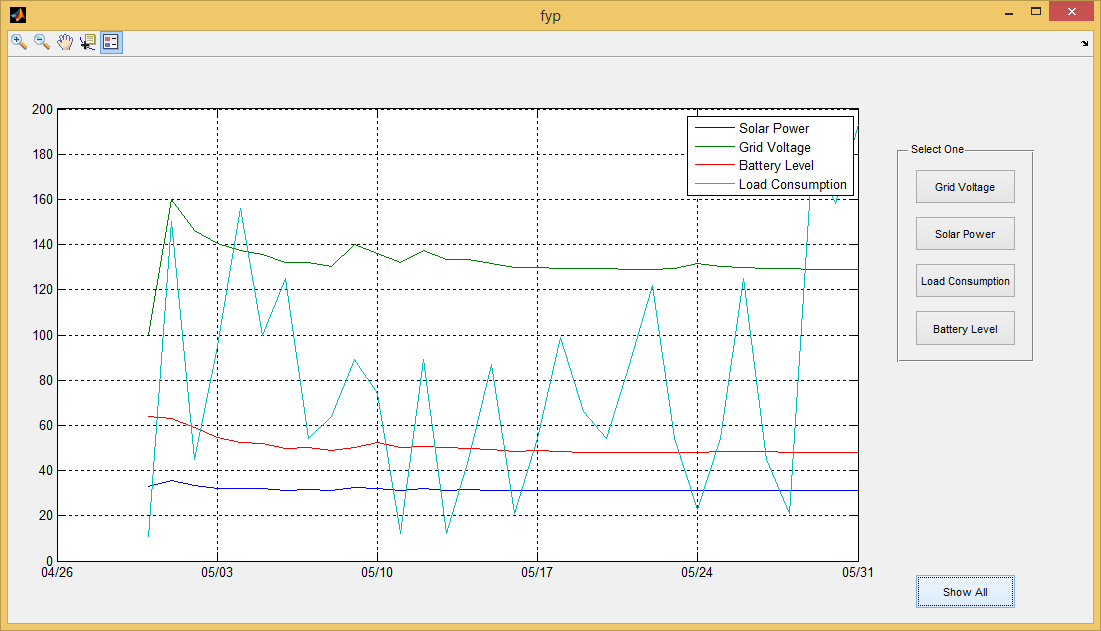


Figure 2.8 GUI Show all

In this GUI we see the initial display window that popes up when our GUI is run. This GUI takes all of the variables in our comma separated file and places them in the form of a graph. A legend is also given at the top of the graph which tells us which line represents which variable. If we see this current graph then we can see that

* Dark blue represents our solar power in KWH
* Green represents our grid voltage in Volts
* Read represents our battery level in percentage
* Light blue represents our load consumption in KWH

Intelligent Unit of Hybrid System

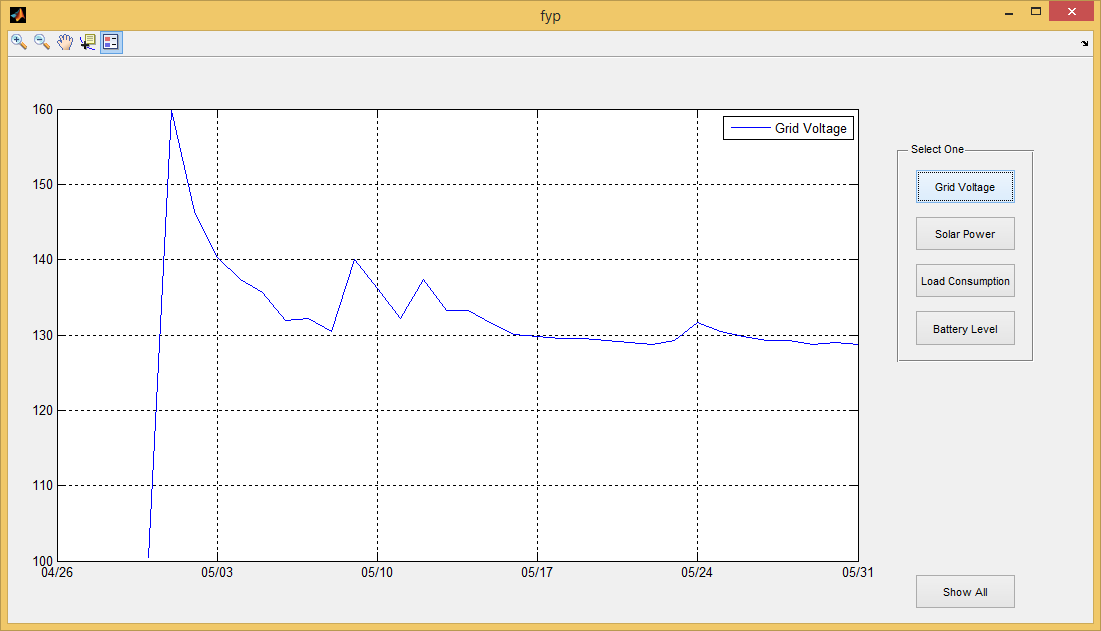
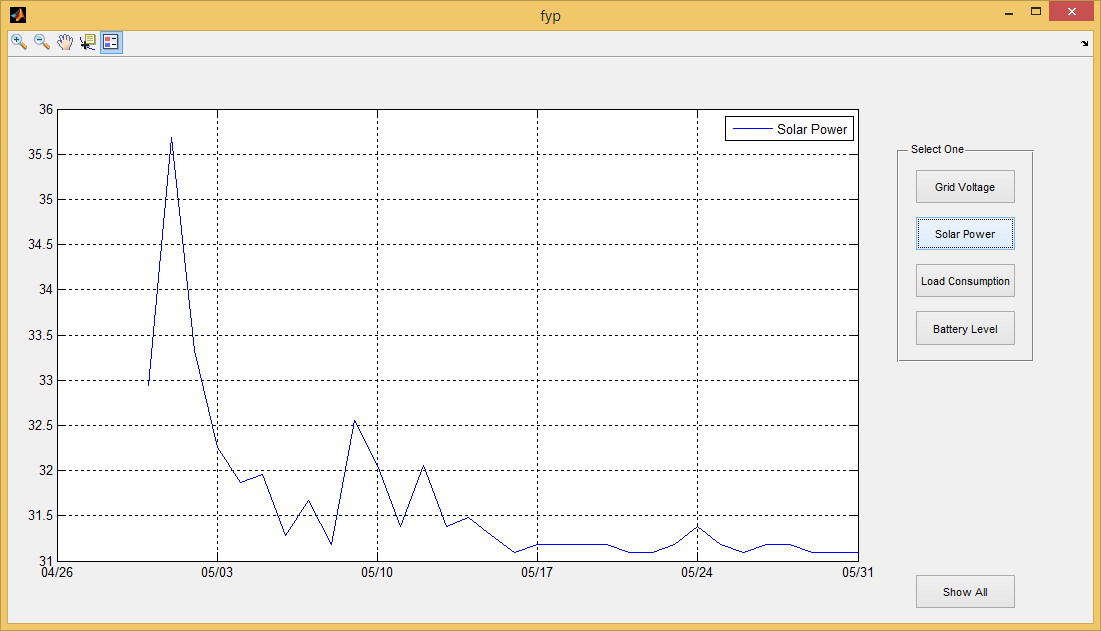


Figure 2.9 GUI grid Voltage

Y-axis Volts

X-axis time

This figure gives us an accurate view of the change of voltage with respect to time.

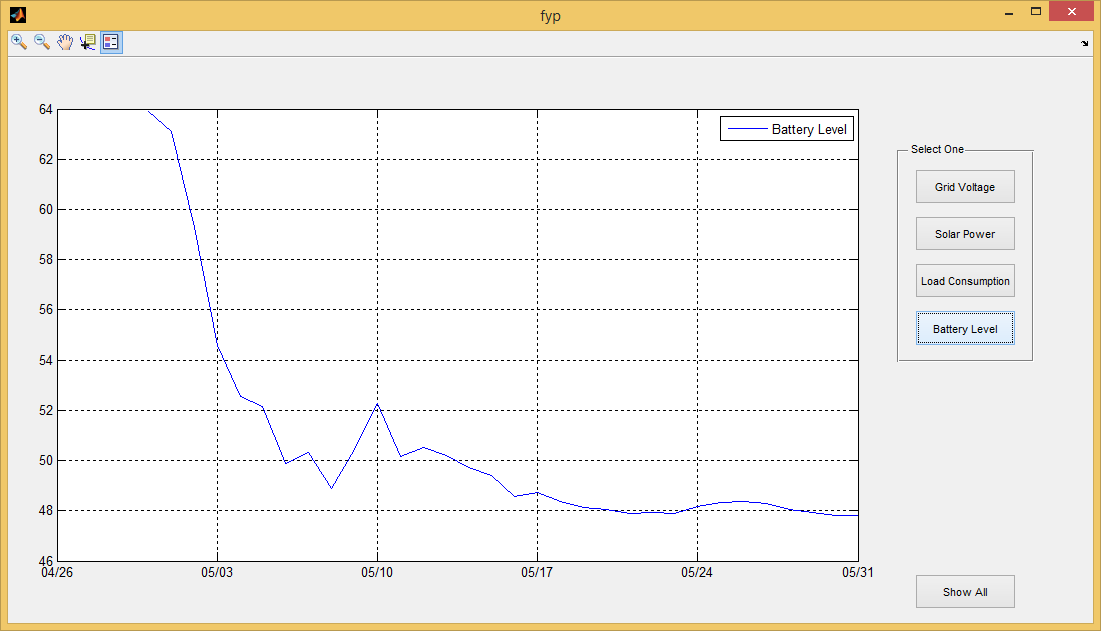


Chapter 2 Solution, Implementation, Design

Figure 2.10 GUI Solar Power

Y-axis KWH

X-axis time

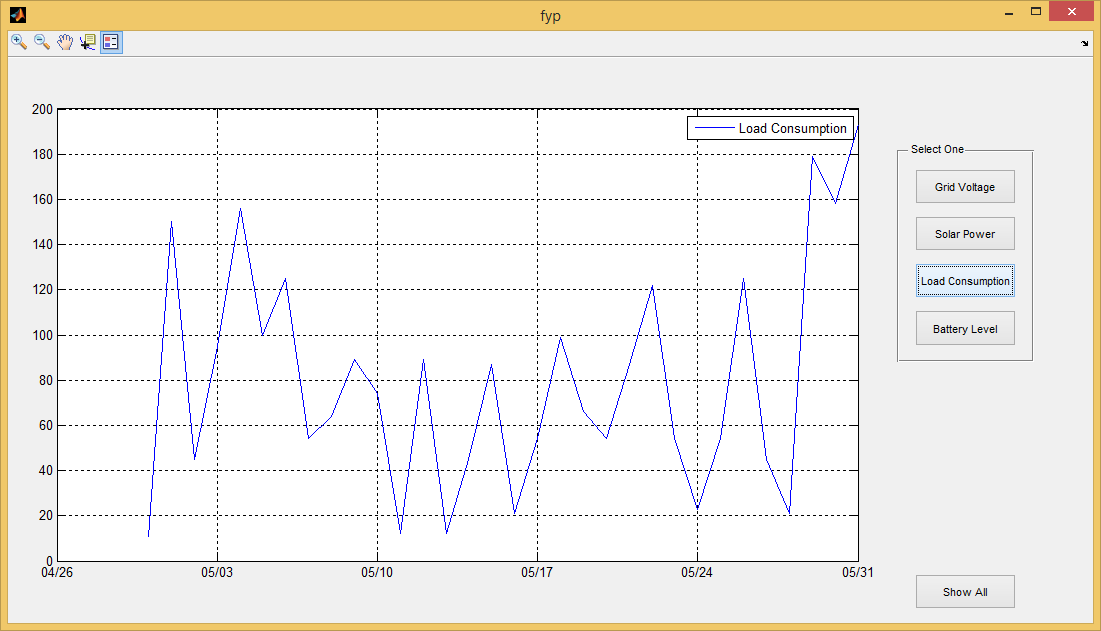


Intelligent Unit of Hybrid System

Figure 2.11 GUI Battery level

Y-axis Battery level (%)

X-axis time



Chapter 2 Solution, Implementation, Design

Figure 2.12 GUI Load Consumption

Y-axis KWH

X-axis time

Intelligent Unit of Hybrid System

# Announced Load Shedding

# As Pakistan is in the crisis of electricity, we have scheduled power outages. These times are announced and occur without any delay from one hour to four hours at a time. As this is scheduled so these future times can be predicted and they can be handled before power outage occurs.

We handled these power outages by first measuring the time at which power outage occurs. And we store this value if it is greater than or equal to sixty minutes. We also measure the corresponding battery level before and after the power outage. This gives us a rough estimate of the load consumption during the power outage. At midnight these values are stored in a load shedding table in which all the load shedding times are stored. A temporary variable is created in Arduino which stores the start time of one hour before load shedding occurs. This value is then compared continuously with the RTC. When these values match then our hybrid system checks whether our battery can support the load for the power outage time or not. Arduino calculates if battery level becomes less than critical value of battery after the power outage period than Arduino charges the battery until it reaches one hundred percent. If the Arduino calculates that battery stays above the critical point after the power outage period than it does not charge the battery. Because battery can then support the load for that time. After the power outage period is over Arduino finds the next time at which power outage is to occur and repeats the whole process again. This process can be seen in the below flowchart.



Chapter 2 Solution, Implementation, Design

Figure 2.13 Announced power outage

Intelligent Unit of Hybrid System

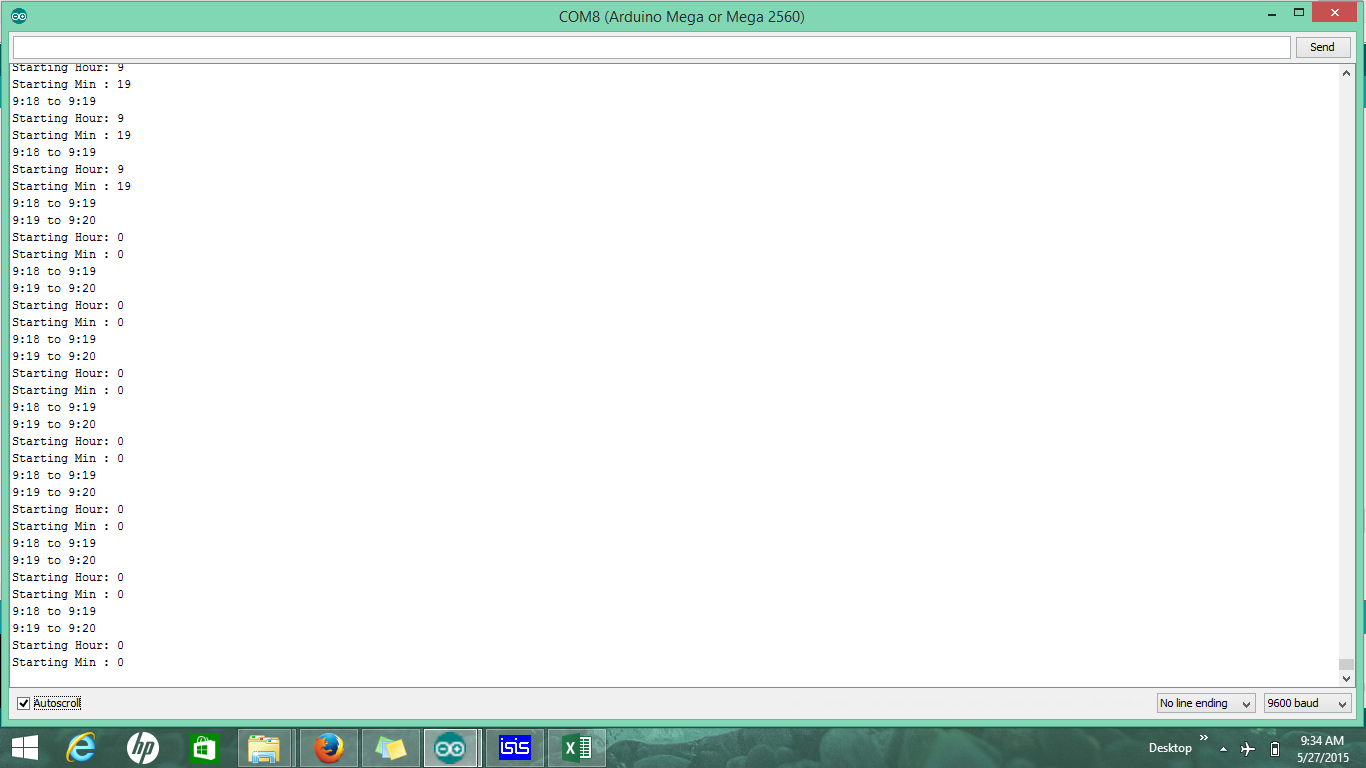
****

Figure 2.14 Serial Monitoring of Load Shedding Schedule

## 2.10. Problems Faced

During the implementation there were few problems that’s we faced

1. Initially we come up with 16 different switching conditions but later we found out some of the conditions were repeated, some were contradictory and while some conditions make our system unfeasible.
2. Microcontroller work on pin voltage of 3.3 volts so all the sensors and external hardware had to work on 3.3 volts.
3. Arduino has designed data logger library for data acquisition in text file only since we need to read that data to show it in a graph there we need to make changed to make system logged to data in excel file which is readable my Matlab.
4. Initial algorithm of scheduler that we came up for tackling load shedding conditions was not feasible therefore after some research we included battery level check in that algorithm as well.

**Chapter 3**

**(Results/Conclusions/ Recommendations)**

* 1. **Results**

We calculated the MIPs of different microcontroller by making assumptions of code that will run the hybrid system and on the basis of MIPs calculation we came to conclusion that Arduino Due will be suitable for our hybrid system. We studied working of different hybrid system and their switching condition. For our hybrid system we figure out 16 possible switching condition out of which we took 7 non-contradictory and non-repetitive conditions. System was running fine on all possible scenarios during testing and debugging stage. Battery cycles were saved by keeping the battery above thresh hold value. Switching conditions were made different from normal commercial hybrid system in a way that our switching condition were based on comparison of incoming load vs load consumption and battery level vs load consumption.

When it comes to load shedding hours, we keep a threshold value to 50 minutes and above and if load shedding occurs it saves the schedule i.e. starting time of loading shedding & ending time of load shedding and battery level. This schedule is followed the next day and it works perfectly. Data acquisition of power value coming from power meter with respect to time and date was taking place at a very high rate. These values were written in excel sheet and using Matlab GUI power values vs time graph was generated.

* 1. **Recommendations**

Chapter 3 Results, Conclusion, Recommendation

1. This Microcontroller should be integrated with read external hardware for further analysis.
2. Data acquisition values could be used for further up gradation of hybrid system according to need.
3. System should be upgraded and tested by adding more power resources.
4. Instead of Microcontroller System should be implemented on some pocket sized PC for example beagle bone or raspberry pi.

**References:**

Intelligent Unit of Hybrid System

Research and commercial hybrid systems:

1. <http://www.nizamsolar.com/hybrid-solar-system-solar-panels-inverters>
2. <http://energypotentialinc.com/pages/hybridsolar>
3. <http://www.livesolar.com.au/hybrid-solar-system>
4. <http://ebr-energy.com/pakistan/solar-wind-hybrid-solution>
5. <http://www.pvresources.com/PVSystems/HybridSystems.aspx>
6. <http://www.belectric.com/fileadmin/MASTER/pdf/brochures/BEL_SKW_Hybrid_Systems_EN_www.pdf>
7. <http://smart-prototyping.com/image/data/2_components/Arduino/100845%20Arduino%20Tiny%20RTC%20I2C%20real%20time%20clock%20module%2024C32%20storage%20%20DS1307/1.jpg>

**Appendix A**

**Code for GUI in MATLAB**

function varargout = fyp(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @fyp\_OpeningFcn, ...

'gui\_OutputFcn', @fyp\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function fyp\_OpeningFcn(hObject, eventdata, handles, varargin)

fid = fopen('data.csv');

cts = textscan(fid, '%s %f %f %f %f', 'Delimiter', ',');

d = datenum(cts{1});

v = [cts{2} cts{3} cts{4} cts{5}] ;

fclose(fid);

plot(d, v), datetick('x')

grid on

Appendix B: Code

legend('Solar Power','Grid Voltage' ,'Battery Level','Load Consumption')

handles.output = hObject;

guidata(hObject, handles);

unction varargout = fyp\_OutputFcn(hObject, eventdata, handles)

arargout{1} = handles.output;

function pushbutton1\_Callback(hObject, eventdata, handles)

fid = fopen('data.csv');

cts = textscan(fid, '%s %f %f %f %f', 'Delimiter', ',');

d = datenum(cts{1});

v = cts{4} ;

fclose(fid);

plot(d, v), datetick('x')

grid on

legend('Battery Level')

cts = textscan(fid, '%s %f %f %f %f', 'Delimiter', ',');

d = datenum(cts{1});

v = cts{5} ;

fclose(fid);

plot(d, v), datetick('x')

grid on

legend('Load Consumption')

function pushbutton3\_Callback(hObject, eventdata, handles)

fid = fopen('data.csv');

cts = textscan(fid, '%s %f %f %f %f', 'Delimiter', ',');

d = datenum(cts{1});

v = cts{2};

fclose(fid);

plot(d, v), datetick('x')

grid on

legend('Solar Power')

function pushbutton4\_Callback(hObject, eventdata, handles)

fid = fopen('data.csv');

cts = textscan(fid, '%s %f %f %f %f', 'Delimiter', ',');

d = datenum(cts{1});

v = cts{3} ;

fclose(fid);

plot(d, v), datetick('x')

grid on

legend('Grid Voltage')

function pushbutton5\_Callback(hObject, eventdata, handles)

fid = fopen('data.csv');

cts = textscan(fid, '%s %f %f %f %f', 'Delimiter', ',');

d = datenum(cts{1});

v = [cts{2} cts{3} cts{4} cts{5}] ;

Intelligent Unit of Hybrid System

fclose(fid);

plot(d, v), datetick('x')

grid on

legend('Solar Power','Grid Voltage' ,'Battery Level','Load Consumption')

.

function radiobutton2\_Callback(hObject, eventdata, handles)

fid = fopen('data.csv');

cts = textscan(fid, '%s %f %f %f %f', 'Delimiter', ',');

d = datenum(cts{1});

v = [cts{2} cts{5}] ;

fclose(fid);

plot(d, v), datetick('x')

grid on

legend('Solar Power','Grid Voltage' ,'Battery Level','Load Consumption')

**Appendix B**

**Main Code for Arduino DUE**

#include <LiquidCrystal.h>

#include <Wire.h>

#include "RTClib.h"

#include <SPI.h>

#include <SD.h>

int a;

RTC\_DS1307 rtc;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

const int chipSelect = 53;

int term;

int S\_GRID = 22;

int S\_CC = 24;

int S\_AC\_charger = 26;

int S\_inverter = 28;

int rem;

int Minimum\_Battery\_level;

int Min\_Grid\_Voltage ;

int Max\_Grid\_Voltage;

int Battery\_Critical\_point ;

int Battery\_losses;

int inverter\_losses;

Appendix B: Code

int extra\_batt = 5;

float battery\_level;

float load\_voltage ;

float load\_current;

int load\_consumption;

float grid\_voltage;

float grid\_current;

float grid\_power;

float DC\_CC\_voltage;

float DC\_CC\_current;

int CC\_power ;

int losses;

int batterybuff1;

int batterybuff2;

int temp ;

int temp2;

int tempo;

int onetemphr;

int onetempmin;

int twotemphr;

int twotempmin;

int Loadshedinghr[10];

int Loadshedingmin[10];

int TLoadshedinghr[10];

int TLoadshedingmin[10];

int batterybuff[10];

int Tbatterybuff[10];

int Loadshedinghr1[10];

int Loadshedingmin1[10];

int TLoadshedinghr1[10];

int TLoadshedingmin1[10];

int batterybuff0[10];

int Tbatterybuff0[10];

int timediff;

int count ;

int timediff1;

int timediff2;

int differ1;

Intelligent Unit of Hybrid System

int differ2;

int battdiffer;

int count2;

DateTime future;

DateTime now;

void setup()

{ future = NULL;

  Serial.begin(9600);

  Minimum\_Battery\_level = 50;

  Min\_Grid\_Voltage = 210;

  Max\_Grid\_Voltage = 250;

  Battery\_Critical\_point = 75;

  Battery\_losses = 10;

  inverter\_losses = 15;

  SPI.begin();

  Wire.begin();

  rtc.begin();

  if (!(rtc.isrunning()))

  {

    Serial.println("RTC not Running");

    rtc.adjust(DateTime(\_\_DATE\_\_, \_\_TIME\_\_));

  }

  lcd.begin(20, 4);

  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

  pinMode(S\_GRID, OUTPUT);

  pinMode(S\_CC, OUTPUT);

  pinMode(S\_AC\_charger, OUTPUT);

  pinMode(S\_inverter, OUTPUT);

  term  = 0;

  int A1 = 0;

  int A2 = 0;

  int A3 = 0;

  int A5 = 0;

Appendix B: Code

  int A6 = 0;

  int A4 = 0;

  SD.begin(chipSelect);

  pinMode(53, OUTPUT);

  temp = 0;

  tempo = 0;

  temp2 = 0;

  batterybuff1 = 0;

  batterybuff2 = 0;

  temp = 0;

  tempo = 0;

  onetemphr = 0;

  onetempmin = 0;

  twotemphr = 0;

  twotempmin = 0;

  Loadshedinghr[10] = 0;

  Loadshedingmin[10] = 0;

  TLoadshedinghr[10] = 0;

  TLoadshedingmin[10] = 0;

  Loadshedinghr1[10] = 0;

  Loadshedingmin1[10] = 0;

  TLoadshedinghr1[10] = 0;

  TLoadshedingmin1[10] = 0;

  batterybuff[10] = 0;

  Tbatterybuff[10] = 0;

  batterybuff0[10] = 0;

  Tbatterybuff0[10] = 0;

  timediff = 0;

  count = 0;

  timediff1 = 0;

  timediff2 = 0;

  differ1 = 0;

  differ2 = 0;

  battdiffer = 0;

  rem = 1;

}

void loop()

Intelligent Unit of Hybrid System

{

  DateTime now = rtc.now();

  battery\_level = analogRead(A2);

  battery\_level = (battery\_level / 1023) \* 100;

  load\_voltage = analogRead(A4);

  load\_current = analogRead(A3);

  load\_voltage = (load\_voltage / 1023) \* 250;

  load\_current = (load\_current / 1023) \* 2;

  load\_consumption = load\_voltage \* load\_current;

  grid\_voltage = analogRead(A5);

  grid\_voltage = (grid\_voltage / 1023) \* 250;

  DC\_CC\_voltage = analogRead(A1);

  DC\_CC\_current = analogRead(A6);

  DC\_CC\_voltage = (DC\_CC\_voltage / 1023) \* 15;

  DC\_CC\_current = (DC\_CC\_current / 1023) \* 33;

  CC\_power  = DC\_CC\_voltage \* DC\_CC\_current;

  losses = Battery\_losses + inverter\_losses ;

  if (!(grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage) &&

(temp == 0))

  {

    onetemphr = int(now.hour());

    onetempmin = int(now.minute());

    twotemphr = now.hour();

    twotempmin = now.minute();

    batterybuff1 = battery\_level;

    batterybuff2 = battery\_level;

    temp = 1;

  }

  if ((grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage) &&

(temp == 1))

  {

    twotemphr = now.hour();

    twotempmin = now.minute();

    batterybuff2 = battery\_level;

Appendix B: Code

    temp = 0;

    temp2 = 1;

  }

  if (twotemphr > onetemphr)

  {

    timediff1 = (twotemphr - onetemphr) \* 60;

  }

  else if (twotemphr < onetemphr)

  {

    timediff1 = (onetemphr - twotemphr) \* 60;

  }

  if (twotempmin > onetempmin)

  {

    timediff2 = twotempmin - onetempmin;

  }

  else if (onetempmin > twotempmin)

  {

    timediff2 = onetempmin - twotempmin;

  }

  timediff = timediff1 + timediff2;

  if ((timediff >= 1) && (temp2 == 1) )

  {

    Loadshedinghr[count] = int(onetemphr);

    Loadshedinghr1[count] = int(twotemphr);

    Loadshedingmin[count] = int(onetempmin);

    Loadshedingmin1[count] = int(twotempmin);

    batterybuff[count] = batterybuff1;

    batterybuff0[count] = batterybuff2;

    temp2 = 0;

    onetemphr = 0;

    twotemphr = 0;

    onetempmin = 0;

    twotempmin = 0;

    count++;

  }

  count2 = 0;

  if ((now.hour() == 24 ) && now.minute() == 59 && now.second() == 57 &&

Intelligent Unit of Hybrid System

tempo == 0)

  {

    DateTime future (now.unixtime() + 86400L + 1);

    for (int k = 0; k <= count ; k++) {

      TLoadshedinghr[count] = Loadshedinghr[count];

      TLoadshedinghr1[count] =  Loadshedinghr1[count];

      TLoadshedingmin[count] = Loadshedingmin[count];

      TLoadshedingmin1[count] = Loadshedingmin1[count];

      Tbatterybuff[count] = batterybuff[count];

      Tbatterybuff0[count] = batterybuff0[count];

    }

    tempo = 1;

  }

  if (now.hour() == 00 && now.minute() == 00 && tempo == 1)

  {

    tempo = 0;

  }

  if (now.day() == future.day())

  {

    for (count2 = 0; count2 <= count ; count2++)

    {

      if ( TLoadshedinghr1[count2] > TLoadshedinghr[count2])

      {

        differ1 = TLoadshedinghr1[count2] - TLoadshedinghr[count2];

      }

      else

      {

        differ1 = TLoadshedinghr[count2] - TLoadshedinghr1[count2];

      }

      if ( TLoadshedingmin1[count2] > TLoadshedingmin[count2])

      {

        differ2 = TLoadshedingmin1[count2] - TLoadshedingmin[count2];

      }

      else

      {

        differ2 = TLoadshedingmin[count2] - TLoadshedingmin1[count2];

      }

Appendix B: Code

    }

    if (differ1 > 0)

    {

      differ2 = 0;

    }

    battdiffer = Tbatterybuff[count2] - Tbatterybuff0[count2];

    if ((battery\_level - (battdiffer + extra\_batt)) > Battery\_Critical\_point)

    {

      goto unannounced;

    }

    else

    {

      if ((now.hour() >= (TLoadshedinghr[count2] - differ1)) && (now.minute() >= TLoadshedingmin[count2] - differ2) && (now.hour() <= (TLoadshedinghr1[count2] - differ1)) && (now.minute() <= TLoadshedingmin1[count2] - differ2))

      {

        if ((CC\_power - losses) > load\_consumption && battery\_level < 100 && (grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

        {

          digitalWrite(S\_GRID, LOW);

          digitalWrite(S\_AC\_charger, HIGH);

          digitalWrite(S\_CC, LOW);

          digitalWrite(S\_inverter, HIGH);

          goto announced;

        }

        else if ((CC\_power - losses) < load\_consumption && battery\_level < 100 && (grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

        {

          digitalWrite(S\_GRID, HIGH);

          digitalWrite(S\_AC\_charger, HIGH);

          digitalWrite(S\_CC, LOW);

          digitalWrite(S\_inverter, LOW);

          goto announced;

Intelligent Unit of Hybrid System

        }

        else {

          goto unannounced;

        }

      }

    }

    goto unannounced;

  }

unannounced :

  if ((CC\_power - losses) > load\_consumption && battery\_level > Battery\_Critical\_point && !(grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

  {

    digitalWrite(S\_GRID, LOW);

    digitalWrite(S\_AC\_charger, LOW);

    digitalWrite(S\_CC, HIGH);

    digitalWrite(S\_inverter, HIGH);

  }

  else if ((CC\_power - losses) < load\_consumption && (grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

  {

    if (battery\_level > Minimum\_Battery\_level)

    {

      digitalWrite(S\_GRID, HIGH);

      digitalWrite(S\_AC\_charger, LOW);

      digitalWrite(S\_CC, HIGH);

      digitalWrite(S\_inverter, LOW);

    }

    else if (battery\_level <= Minimum\_Battery\_level)

    {

      digitalWrite(S\_GRID, HIGH);

      digitalWrite(S\_AC\_charger, HIGH);

      digitalWrite(S\_CC, LOW);

      digitalWrite(S\_inverter, LOW);

    }

  }

  else if ((CC\_power - losses) > load\_consumption && battery\_level < Battery\_Critical\_point && (grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

Appendix B: Code

  {

    digitalWrite(S\_GRID, HIGH);

    digitalWrite(S\_AC\_charger, LOW);

    digitalWrite(S\_CC, HIGH);

    digitalWrite(S\_inverter, LOW);

  }

  else if ((CC\_power - losses) < load\_consumption && battery\_level < Battery\_Critical\_point && (grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

  {

    if (battery\_level <= Minimum\_Battery\_level)

    {

      digitalWrite(S\_GRID, HIGH);

      digitalWrite(S\_AC\_charger, HIGH);

      digitalWrite(S\_CC, LOW);

      digitalWrite(S\_inverter, LOW);

    }

    else if (battery\_level > Minimum\_Battery\_level)

    {

      digitalWrite(S\_GRID, HIGH);

      digitalWrite(S\_AC\_charger, LOW);

      digitalWrite(S\_CC, HIGH);

      digitalWrite(S\_inverter, LOW);

    }

  }

  else if ((CC\_power - losses) > load\_consumption && battery\_level < Battery\_Critical\_point && !(grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

  {

    digitalWrite(S\_GRID, LOW);

    digitalWrite(S\_AC\_charger, LOW);

    digitalWrite(S\_CC, HIGH);

    digitalWrite(S\_inverter, HIGH);

  }

Intelligent Unit of Hybrid System

  else if ((CC\_power - losses) < load\_consumption && battery\_level < Battery\_Critical\_point && !(grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

  {

    digitalWrite(S\_GRID, LOW);

    digitalWrite(S\_AC\_charger, LOW);

    digitalWrite(S\_CC, HIGH);

    digitalWrite(S\_inverter, LOW);

  }

  else if ((CC\_power - losses) < load\_consumption && battery\_level > Battery\_Critical\_point && !(grid\_voltage > Min\_Grid\_Voltage && grid\_voltage < Max\_Grid\_Voltage))

  {

    digitalWrite(S\_GRID, LOW);

    digitalWrite(S\_AC\_charger, LOW);

    digitalWrite(S\_CC, HIGH);

    digitalWrite(S\_inverter, HIGH);

  }

  goto announced;

announced :

  lcd.setCursor(0, 0);

  lcd.print("Load :");

  lcd.setCursor(14, 0);

  lcd.print(load\_consumption);

  lcd.setCursor(0, 1);

  lcd.print("Solar Power");

  lcd.setCursor(14, 1);

  lcd.print(CC\_power );

  lcd.setCursor(0, 2);

  lcd.print("Battery Level");

  lcd.setCursor(14, 2);

  lcd.print(battery\_level);

  lcd.setCursor(0, 3);

  lcd.print("GRID Voltage");

  lcd.setCursor(14, 3);

  lcd.print(grid\_voltage);

  // make a string for assembling the data to log:

Appendix B: Code

  String dataString2 = "";

  String dataString6 = "";

  String dataString8 = "";

  String dataString10 = "";

  String Year = "";

  String Month = "";

  String Days = "";

  String Hours = "";

  String Minute = "";

  String dataString11 = "";

  int A = now.year(); int B = now.month(); int C = now.day(); int D = now.hour(); int E = now.minute();

  dataString2 += String(CC\_power );

  dataString6 += String(grid\_voltage);

  dataString8 += String(battery\_level);

  dataString10 += String(load\_consumption);

  Year += String(A, DEC);

  Month += String(B, DEC);

  Days += String(C, DEC);

  Hours += String(D, DEC);

  Minute += String(E, DEC);

  dataString11 += String(" ");

  /////////////////////////////////////

  // open the file. note that only one file can be open at a time,

  // so you have to close this one before opening another.

  File dataFile = SD.open("datalog.csv", FILE\_WRITE);

  // if the file is available, write to it:

  if (dataFile) {

    dataFile.print(Year);

    dataFile.print(":");

    dataFile.print(Month);

    dataFile.print(":");

    dataFile.print(Days);

    dataFile.print(" ");

    dataFile.print(Hours);

    dataFile.print(":");

    dataFile.print(Minute);

Intelligent Unit of Hybrid System

    dataFile.print(",");

    dataFile.print(dataString8);

    dataFile.print(",");

    dataFile.print(dataString6);

    dataFile.print(",");

    dataFile.print(dataString2);

    dataFile.print(",");

    dataFile.print(dataString10);

    dataFile.println(dataString11);

    dataFile.close();

    // print to the serial port too:

  }

  SD.begin(chipSelect);

  for (int j = 0; j <= count - 1; j++)

  {

    Serial.print(Loadshedinghr[j]);

    Serial.print(":");

    Serial.print(Loadshedingmin[j]);

    Serial.print(" to ");

    Serial.print(Loadshedinghr1[j]);

    Serial.print(":");

    Serial.println(Loadshedingmin1[j]);

  }

  Serial.print("Starting Hour: ");

  Serial.println(onetemphr);

  Serial.print("Starting Min : ");

  Serial.println(onetempmin);

  timediff1 = 0;

  timediff2 = 0;

  timediff = 0;