MODELLING OF DEMAND RESPONSE SYSTEM (DRS) AN INTEGRAL OF SMART GRID

OSAGBEMI, DAVID SEGUN

2006/24424EE

A THESIS SUBMITTED TO THE DEPARTMENT OF ELECTRICAL / ELECTRONICS ENGINEERING,

FEDERAL UNIVERSITY OF TECHNOLOGY MINNA

IN PARTIAL FULFILMENT OF A BACHELORS DEGREE IN ELECTRICAL / ELECTRONICS ENGINEERING

November 2011

DEDICATION

I dedicate this project to God almighty appreciating him for his support, grace, faithfulness and mercy towards me.

DECLARATION

I OSAGBEMI, DAVID SEGUN with matriculation number 2006/24424EE, declare that this work was done by me and has never been presented elsewhere for the award of a degree. I also hereby relinquish the copyright to the Federal University of Technology, Minna.

OSAGBEMI DAVID SEGUN	Dr. JACOB TSADO
(Name of student)	(Name of supervisor)
[illegible] 18/11/2011	[illegible] 18/11/2011
(Signature and date)	(Signature and date)
ENGR A.G. RAJI	Engr Dr. G. I. [illegible]
(Name of H.O.D)	(Name of External Examiner)
[illegible] (March 15, 2012)	[illegible] 6/3/2012
(Signature and date)	(Signature and date)

ACKNOWLEDGEMENT

My gratitude to the Almighty God for the enablement and wisdom to carry out this project, through him all things are possible.

My heartfelt thanks to my supervisor Dr. Jacob Tsado, for the total support both morally and academically. I appreciate your encouragement, motivation, patience and all necessary assistance you gave to realize this project, thank you so much sir.

I also appreciate my parents Rev and Mrs. Osagbemi and my entire family for their support throughout my study. I appreciate the contributions of Temitope Oguntade, Bola, Idoko Sunday, Stephen Kolade and all my friends and colleagues. Thank you all.

ABSTRACT

This project presents the modeling of demand response system an integral of smart grid. The model consist of distribution software interfaced with hardware components, the software represents the Demand Response System (DRS) while the hardware components represent various load unit connected to the system. The distribution software is installed on a computer system which is interfaced with hardware circuit through the RS-232 communication port. The Demand Response System is capable of distributing power to meet consumers varying energy demand in an effective way. This project work is to make power distribution more efficient and reduce energy wastage.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1. General Overview :

Power systems engineering is a subfield of engineering that deals with the generation, transmission and distribution of electric power as well as the electrical devices connected to such systems including generators, motors and transformers[1].

The function of an electrical power system is to generate, transmit and distribute electrical energy in the most effective, secure and efficient possible way. To allow these aims to be met, the power system is interconnected from the generation to the consumer end of the system. This interconnection of the system is called the **Electric Grid**. The system carries the electric power from the generators over long distance to a distribution system, which brings the power to the customers [2].

The power system is a complex interconnected network which can be divided into four major parts:

1. Generation
2. Transmission
3. Distribution
4. Loads

The one line diagram below illustrates an interconnection of these parts in a typical power system.

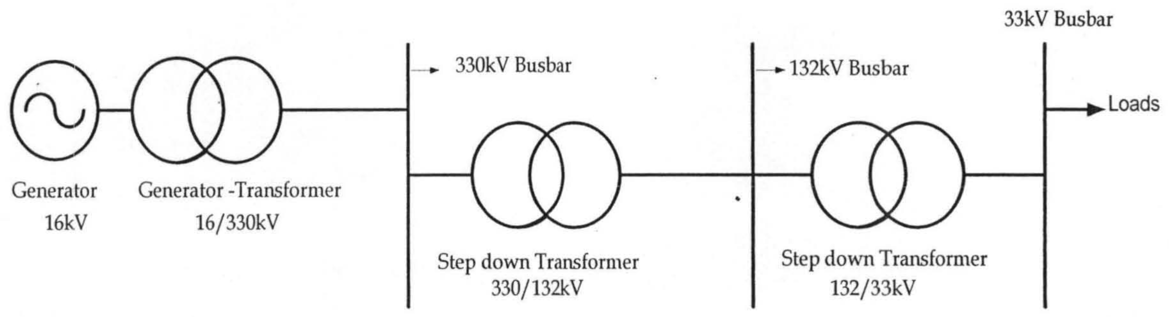


Fig1.1 Single line diagram of power system

Generation refers to the production of electricity from sources of energy, such as hydro, coal, thermal and natural gas. The transmission system transfers electric energy from various generating stations at various locations over long distances (usually at 330KV, 132KV, 33KV, 11KV) to the distribution systems. Distribution systems ultimately supplies power to the consumer loads which may be industrial, commercial and residential [3] . The electric grid involves interconnecting the various generating stations and the consumer loads such that total supply of electricity is coordinated from a fixed point, electric grid is also called the national electric grid or national grid (the Nigeria National Control Center is located at Oshogbo) [4].

Today's alternating current power grid evolved after 1896, based in part on Nikola Tesla's design published in 1888 [5]. Many implementation decisions that are still in use today were made for the first time using the limited emerging technology available 120 years ago. The electric grids available in developing countries are often antiquated. But even more modern systems, which are available in developed countries such as the United States, Germany Japan can be 50 years old or more. These systems are characterized by inefficiency, vulnerability to cyber attack, unreliable, polluting, and incompatible with renewable energy sources (Bio-fuel; Geothermal Energy; Solar Energy; Tidal Energy; and Wind Energy)[6].

The presence of these problems in the electric grid reveals the need for a better system with improved performance.

1.1.1 The Future Grid:

Following the limitations associated with the electric grid, a new system gradually evolved. This started as advancement in the existing grid by inculcating advancement in information technology to realize a system with highly improved performance. The future grid is envisioned to be more sophisticated and free of the problems involved in the existing system. This future grid forms what is now referred to as THE SMART GRID. The Smart Grid is an electricity network that can intelligently integrate the behavior and actions of all users connected to it (generators, consumers and those that do both) in order to efficiently deliver sustainable, economic and secure electricity supplies. A smart grid employs innovative products and services together with intelligent monitoring, control, communication, and self-healing technologies in order to meet consumer increasingly changing demand. Figure 1.2 shows smart grid concept.

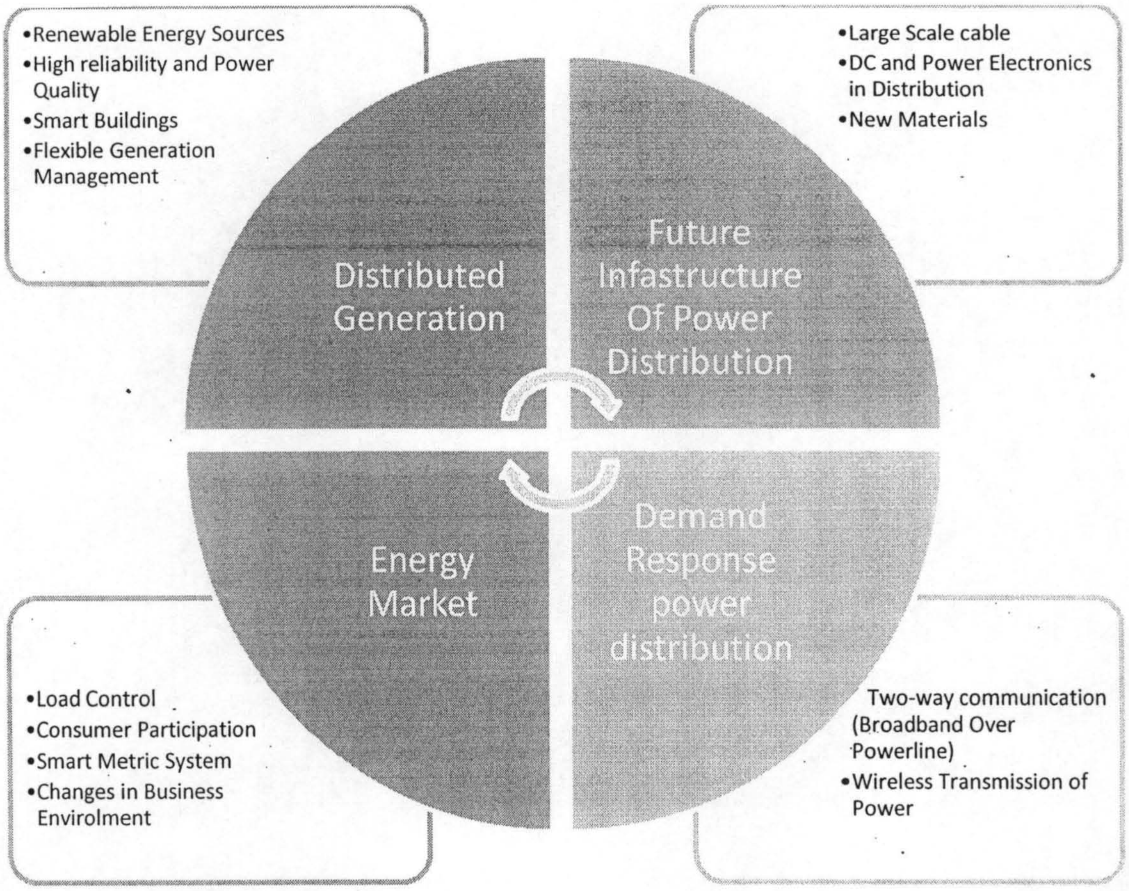

•Renewable Energy Sources
•High reliability and Power Quality
•Smart Buildings
•Flexible Generation Management
•Large Scale cable
•DC and Power Electronics in Distribution
•New Materials
Distributed Generation
Future Infastructure Of Power Distribution
Energy Market
Demand Response power distribution
•Load Control
•Consumer Participation
•Smart Metric System
•Changes in Business Envirolment
Two-way communication (Broadband Over Powerline)
•Wireless Transmission of Power

Fig 1.2 Smart grid concept

The Smart Grid promises many benefits, including increased energy efficiency, reduced carbon emissions, and improved power reliability.

Further futures of the smart grid include the following:

1. Better facilitate the connection and operation of generators of all sizes and technologies;
2. Allow consumers to play a part in optimizing the operation of the system;
3. Provide consumers with greater information and options for choice of supply;

- Significantly reduce the environmental impact of the whole electricity supply system;
- Maintain or even improve the existing level of system reliability, quality and security of supply;
- Maintain and improve the existing services.

1.2 Aim and Objectives:

The aim of this research work is to design a model to illustrate a type of power distribution obtainable in the smart grid. This involves distribution of electrical power to various consumers base on their electrical load demand referred to as Demand Response System (DRS). The objectives of the project are as follows:

1. To design a power distribution software which is capable of responding instantaneously to consumer demand.
2. To design an electrical circuit representing different load units
3. To interface the distribution software to the electrical circuit

1.3 Methodology

The project comprises of two parts: The Hardware Module and Distribution Software. Steps to achieve each part are highlighted below

1.3.1 Hardware Module (Load Units)

i. Identify the major components for the Hardware module
ii. Design of the various components identified in (i) above
iii. Interface the modules with the Distribution Software

3. Interface the modules with the Distribution Software

1.3.2 Distribution Software

1. Design of a distribution algorithm

2. Distribution software programming

3. Interfacing the distribution software with the Hardware module

1.4 Scope of the Project

The project is basically a model to illustrate how the Demand Response System (DRS) can be implemented in a power distribution network the smart grid system. The model consists of distribution software interfaced with hardware components which represents the load units. The system is capable of providing a two-way communication between the power system and the load demanded by the consumer. The scope of this work does not include generation of electricity from its various sources.

CHAPTER TWO

LITERATURE REVIEW

2.1 Power Distribution System:

The power distribution system is the part of the power system which connects the distribution substation to consumers [7]. The system is usually composed of the primary distribution network and the secondary distribution network. The secondary distribution supplies the large industrial loads while the primary distribution (which includes the primary substation and primary distribution lines) feeds small industrial loads, commercial loads, and residential loads.

2.2 The Smart Grid Technology:

A smart grid is a form of electricity network in which power distribution is implemented via digital technology. The smart grid is an upgrade of 20th century power grids which generally "broadcast" power from a few central power generators to a large number of users, the upgrade is capable of routing power in better ways (instead of broadcast) to respond to a very wide range of conditions, and to charge a premium to those that use energy during peak hours .The system delivers electricity from suppliers to consumers using two-way digital communications to monitor electricity flowing in it and also control appliances at consumer homes [6]. The increased data transmission capacity made available by the growth of an extensive digital communication network for the internet has made it conceptually possible to apply sensing, measurement and control devices with two-way communications to electricity production, transmission, distribution and consumption parts of the power grid. This grid includes an intelligent monitoring system that keeps track of all electricity flowing in the system. When power is least expensive the user can allow the smart grid to turn on selected home appliances such as washing machines or factory processes that can run at arbitrary hours. At peak times it could turn off selected appliances to reduce demand.

2.2.1 History of Smart Grid:

Many power implementation decisions that are still in use today were made for the first time using the limited emerging technology available about 120 years ago, following Nikola Tesla's 19th century design. Specific obsolete power grid assumptions and features (like centralized unidirectional [1] electric power transmission, electricity distribution, and demand-driven control) represent a vision of what was thought possible in the 19th century.

Over the past 50 years, electricity networks have not kept pace with modern challenges, such as:

- Security threats, from either energy suppliers or cyber attack
- National goals to employ alternative power generation sources whose intermittent supply makes maintaining stable power significantly more complex
- Conservation goals that seek to lessen peak demand surges during the day so that less energy is wasted in order to ensure adequate reserves
- High demand for an electricity supply that is uninterruptible
- Digitally controlled devices that can alter the nature of the electrical load (giving the electric company the ability to turn off appliances in your home if they see fit) and result in electricity demand that is incompatible with a power system that was built to serve an "analog economy." For a simple example, timed Christmas lights can present significant surges in demand because they come on at near the same time (sundown or a set time).[9]

Smart grid technologies have emerged from earlier attempts at using electronic control, metering, and monitoring. In the 1980s, Automatic meter reading was used for monitoring loads from large customers, and evolved into the Advanced Metering Infrastructure of the 1990s, whose meters could store how electricity was used at different times of the day.[11] Smart meters add continuous communications so that monitoring can be done in real time, and can be used as a gateway to demand response-aware devices and "smart sockets" in the home.

2.3 Smart grid functions

In other to have a defined focus for design and research it is necessary to identify the expected functions of the smart grid. According to the United States Department of Energy's Modern Grid Initiative report [12] a modern smart grid must:

1. Be able to heal itself
2. Motivate consumers to actively participate in operations of the grid
3. Resist attack
4. Provide higher quality power that will save money wasted from outages
5. Accommodate all generation and storage options
6. Enable electricity markets to flourish
7. Run more efficiently
8. Enable higher penetration of intermittent power generation sources

2.3.1 Self-healing

Using real-time information from embedded sensors and automated controls to anticipate, detect, and respond to system problems, a smart grid can automatically avoid or mitigate power outages, power quality problems, and service disruptions.

As applied to distribution networks, there is no such thing as a "self healing" network. If there is a failure of an overhead power line, given that these tend to operate on a radial basis (for the most part) there is an inevitable loss of power. In the case of urban/city networks that for the most part are fed using underground cables, networks can be designed (through the use of interconnected topologies) such that failure of one part of the network will result in no loss of supply to end users. A fine example of an interconnected network using zoned protection is that of the Merseyside and North Wales Electricity Board (MANWEB) [13].

2.3.2 Consumer participation

A smart grid is in essence, an attempt to require consumers to change their behavior around variable electric rates or to pay vastly increased rates for the privilege of reliable electrical service during high-demand conditions. Historically, the intelligence of the grid in North America has been demonstrated by the utilities operating it in the spirit of public service and shared responsibility, ensuring constant availability of electricity at a constant price, day in and day out, in the face of any and all hazards and changing conditions. A smart grid incorporates consumer equipment and behavior in grid design, operation, and communication. This enables consumers to better control "smart appliances" and "intelligent equipment" in homes and businesses, interconnecting energy management systems in "smart buildings" and enabling consumers to better manage energy use and reduce energy costs. Advanced communications capabilities equip customers with tools to exploit real-time electricity pricing, incentive-based load reduction signals, or emergency load reduction signals.

2.3.3 Ability to Resist Attack

Smart grid technologies better identify and respond to man-made or natural disruptions. Real-time information enables grid operators to isolate affected areas and redirect power flows around damaged facilities.

One of the most important issues of resist attack is the smart monitoring of power grids, which is the basis of control and management of smart grids to avoid or mitigate the system-wide disruptions like blackouts. The traditional monitoring is based on weighted least square (WLS) which is very weak and prone to fail when gross errors (including topology errors, measurement errors or parameter errors) are present. New technology of state monitor is needed to achieve the goals of the smart grids.

2.3.4 High quality power

Outages and power quality issues cost US businesses more than $100 billion on average each year,[14] other developed countries incur similar cost (underdeveloped and developing countries like Nigeria suffers more cost due to frequent power outage). It is asserted that assuring more stable power provided by smart grid technologies will reduce downtime and prevent such high losses, but the reliability of complex systems is very difficult to analyze and guarantee.

2.3.5 Accommodate generation options

As smart grids continue to support traditional power loads they also seamlessly interconnect fuel cells, renewables, microturbines, and other distributed generation technologies at local and regional levels. Integration of small-scale, localized, or on-site power generation allows residential, commercial, and industrial customers to self-generate and sell excess power to the grid with minimal technical or regulatory barriers. This also improves reliability and power quality, reduces electricity costs, and offers more customer choice. It will be a long time before a smart grid is actually necessary to realize these benefits. The existing grid can typically accommodate an order of magnitude more than the existing small-scale localized generation without the benefit of the smart grid. Most obstacles to the integration of larger renewable projects, like wind farms, is due to limitations of traditional infrastructure.

2.3.6 Enable electricity market

Significant increases in bulk transmission capacity will require construction of new transmission lines before improvements in transmission grid management proposed by smart grids can make a difference. Such improvements are aimed at creating an open marketplace where alternative energy sources from geographically distant locations can easily be sold to customers wherever they are located.

Intelligence in distribution grids are not required to enable small producers to generate and sell electricity at the local level using alternative sources such as rooftop-mounted photo voltaic panels, small-scale wind turbines, and micro hydro generators.

2.4 Demand Response:

Demand response support allows generators and loads to interact in an automated fashion in real time, coordinating demand to flatten spikes. Eliminating the fraction of demand that occurs in these spikes eliminates the cost of adding reserve generators, cuts wear and tear and extends the life of equipment, and allows users to cut their energy bills by telling low priority devices to use energy only when it is cheapest.[15]

Currently, power grid systems have varying degrees of communication within control systems for their high value assets, such as in generating plants, transmission lines, substations and major energy users. In general information flows one way, from the users and the loads they control back to the utilities. The utilities attempt to meet the demand and succeed or fail to varying degrees (brownout, rolling blackout, uncontrolled blackout). The total amount of power demand by the users can have a very wide probability distribution which requires spare generating plants in standby mode to respond to the rapidly changing power usage. The smart grid tries to meet the varying consumer demands in the most effective way.

2.5 Smart meters

A smart grid replaces analog mechanical meters with digital meters that record usage in real time. Smart meters are similar to Advanced Metering Infrastructure meters and provide a communication path extending from generation plants to electrical outlets (smart socket) and other smart grid-enabled devices. By customer option, such devices can shut down during times of peak demand.

2.6 Improved interfaces and decision support

Smart grid uses information systems that reduce complexity so that operators and managers have tools to effectively and efficiently operate a grid with an increasing number of variables. Technologies include visualization techniques that reduce large quantities of data into easily understood visual formats, software systems that provide multiple options when systems operator actions are required, and simulators for operational training.

CHAPTER THREE

DESIGN AND IMPLEMENTATION

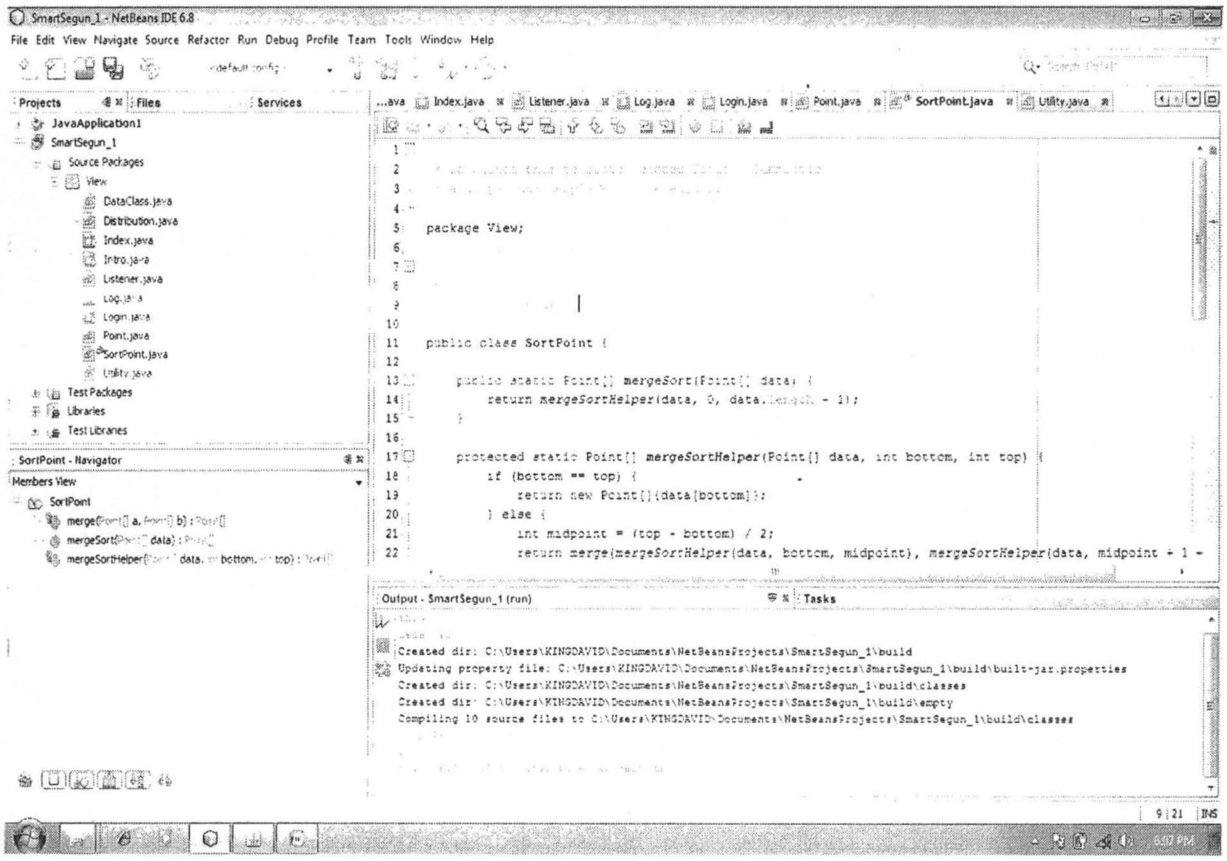
3.1 Design

This project consists of distribution software installed on a computer system and interfaced with an electrical circuit through the RS-232 communication port of the computer.

The design was carried out in two parts namely: software design and hardware design. The steps taking to achieve each part are explained below.

3.1.1 Software Design

The distribution software program was written in java language which was designed using the Netbeans IDE 6.8. Fig 3.1 shows the Netbeans window.



```java
package View;


public class SortPoint {

    public static Point[] mergeSort(Point[] data) {
        return mergeSortHelper(data, 0, data.length - 1);
    }

    protected static Point[] mergeSortHelper(Point[] data, int bottom, int top) {
        if (bottom == top) {
            return new Point[]{data[bottom]};
        } else {
            int midpoint = (top - bottom) / 2;
            return merge(mergeSortHelper(data, bottom, midpoint), mergeSortHelper(data, midpoint + 1 -
```

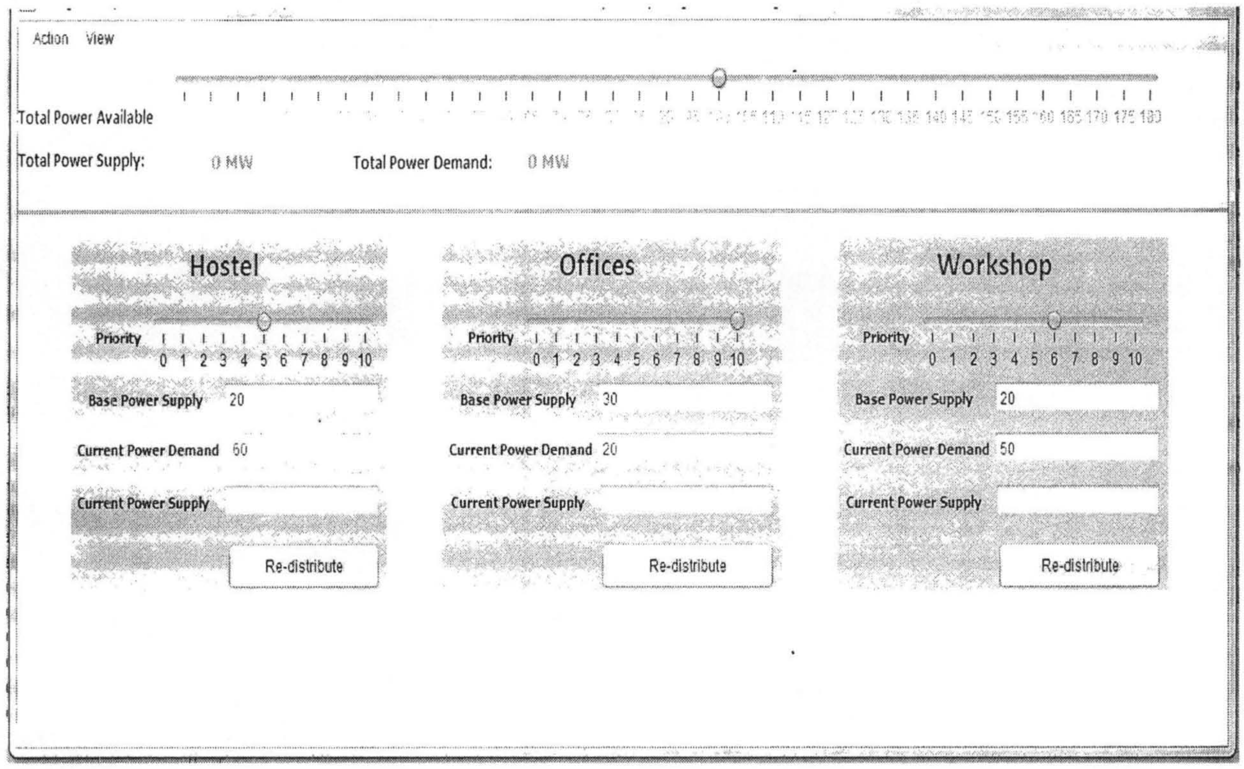
```
Created dir: C:\Users\KINGDAVID\Documents\NetBeansProjects\SmartSegun_1\build
Updating property file: C:\Users\KINGDAVID\Documents\NetBeansProjects\SmartSegun_1\build\built-jar.properties
Created dir: C:\Users\KINGDAVID\Documents\NetBeansProjects\SmartSegun_1\build\classes
Created dir: C:\Users\KINGDAVID\Documents\NetBeansProjects\SmartSegun_1\build\empty
Compiling 10 source files to C:\Users\KINGDAVID\Documents\NetBeansProjects\SmartSegun_1\build\classes
```

Fig 3.1 Netbeans IDE 6.8 window

The software performs the following functions:

1. Receives input from the hardware module and attends to this data in real time.
2. Displays the total available power and its allocation to each load unit, current value of power not used (if any), preset base power that can be allocated to each load unit and priority of supply to these load units.
3. Shows changes in total power available for supply when there is change in supply input to the system.
4. Update load distribution database with changes in power demand and corresponding supply to various units against the time of change.

The graphic user interface and flow chart diagram of the software is shown in the figures below



Total Power Available

Total Power Supply: 0 MW

Total Power Demand: 0 MW

Hostel

Base Power Supply 20

Current Power Demand 60

Current Power Supply

Offices

Base Power Supply 30

Current Power Demand 20

Current Power Supply

Workshop

Base Power Supply 20

Current Power Demand 50

Current Power Supply

Fig 3.2 Distribution Software Graphic User Interface

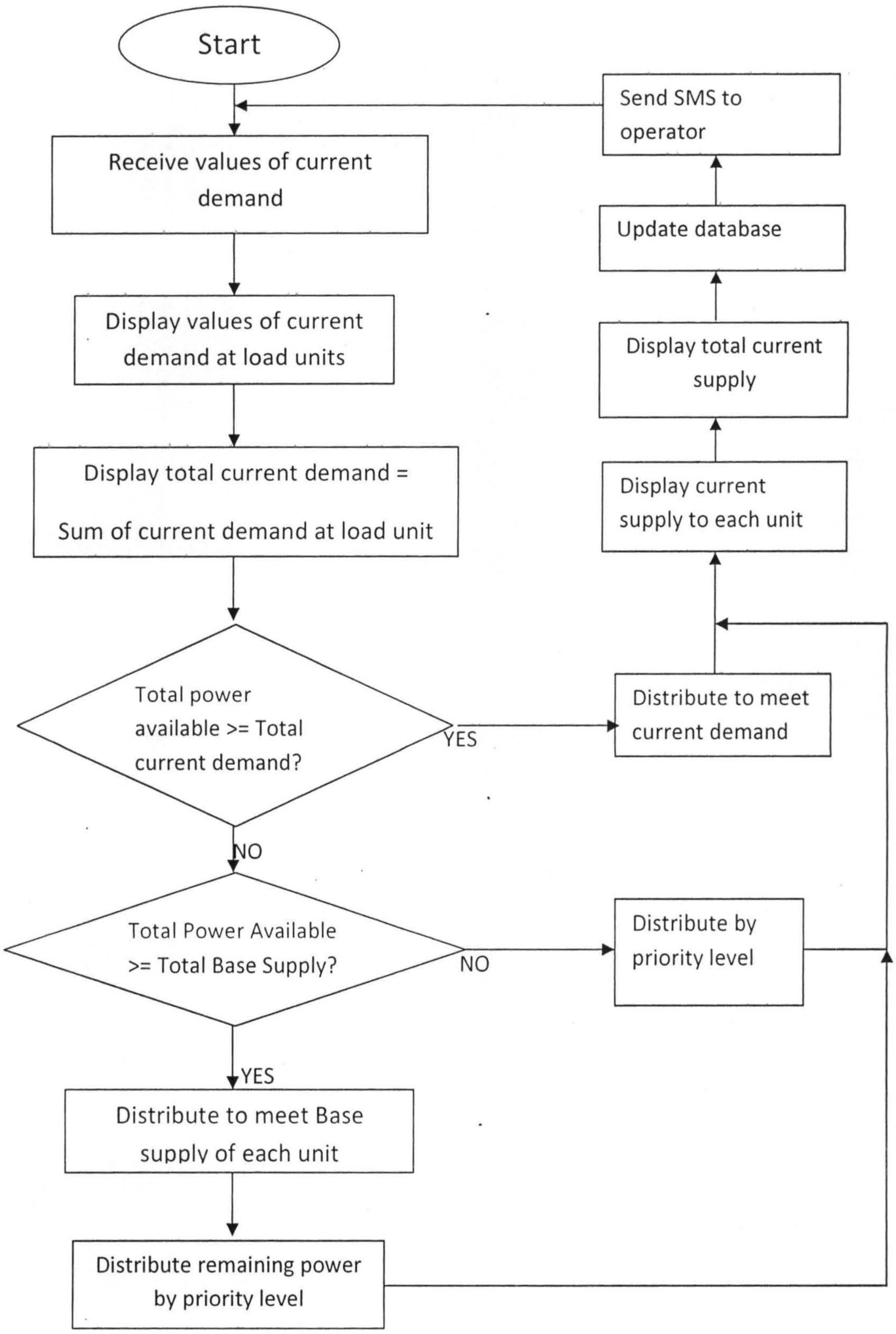

Start
Receive values of current demand
Display values of current demand at load units
Display total current demand = Sum of current demand at load unit
Total power available >= Total current demand?
YES
NO
Total Power Available >= Total Base Supply?
NO
YES
Distribute to meet Base supply of each unit
Distribute remaining power by priority level
Distribute by priority level
Distribute to meet current demand
Display current supply to each unit
Display total current supply
Update database
Send SMS to operator

Fig 3.3 Software Flow Chart Diagram

The software was designed as a real-time operating system. A real-time operating system manages processes and resource allocation in a real-time system. It starts and stops processes so that stimuli can be handled and allocates memory and processor resources. The components of a real-time operating system depend on the size and complexity of the real-time system being developed. They usually include:

1. A real-time clock: This provides information to schedule processes periodically.

2. An interrupt handler: This manages aperiodic requests for service.

3. A scheduler: It examines the processes that can be executed and choose one for execution.

4. Resource manager: This allocates appropriate memory and processor resources.

5. A dispatcher: This component is responsible for starting the execution of a process.

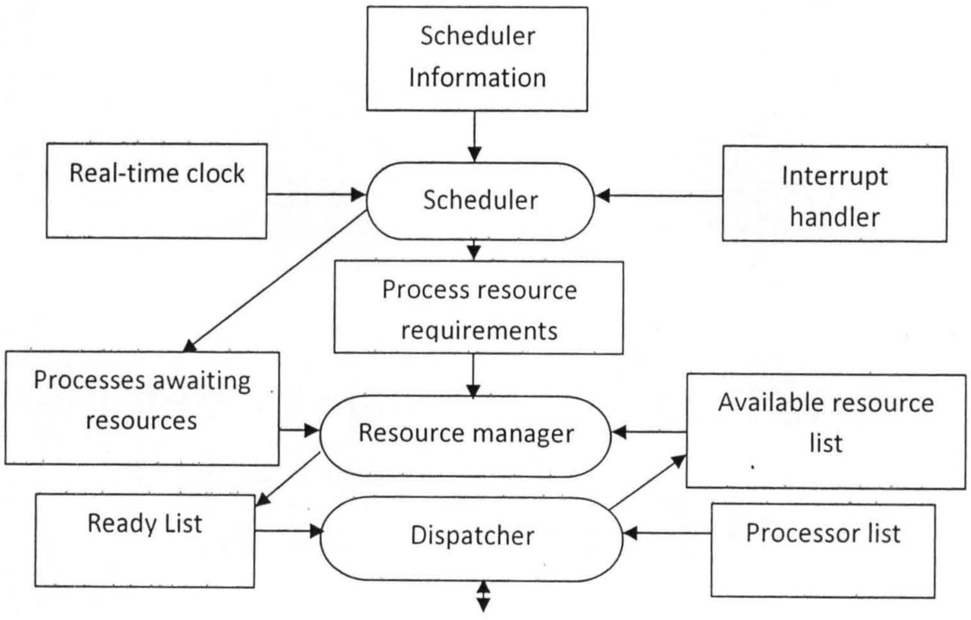


Fig 3.4 Components of a real-time operating system

3.1.2 Computer Input/Output

Peripheral devices are usually external to the computer. Printers, mice, video cameras, scanners, data/fax modems, plotters, robots, telephones, light switches, weather gauges, Palm Computing Platform devices, and many others exist beyond the confines of our desktop or server machine. These devices are accessed via the two communication ports on the computer system (the serial and parallel ports).

The Java Communications API cleverly unifies the programming model for dealing with a range of external devices. It supports both serial (RS232/434, COM) and parallel (printer, LPT) ports. Serial ports are used for modems and occasionally printers, and parallel ports are used for printers and sometimes (in the PC world) for Zip drives and other peripherals.

3.1.3 The Java Communications API

The choice of programming language for this project work is the Java Programming language. It is robust, scalable, simple, fast and has a very rich input/output and network library. It is the choice language due to the fact that this work is largely based on the effectiveness and efficiency of input/output, and the Java language is one of the most established in this domain.

The Communications API is centered on the abstract class CommPort and its two subclasses, SerialPort and ParallelPort, which describe the two main types of ports found on desktop computers. CommPort represents a general model of communications, and has general methods like getInputStream() and getOutputStream() that allow us to communicate with the device on that port.

However, the constructors for these classes are intentionally non-public. Rather than constructing them, we instead use the static factory method CommPortIdentifier.getPortIdentifiers() to get a list of ports, we choose a port from this list, and call this CommPortIdentifier's open() method to receive a CommPort object. We cast the CommPort object to a non-abstract subclass representing a particular communications device. At present, the subclass must be either SerialPort or ParallelPort.

Each of these subclasses has some methods that apply only to that type. For example, the SerialPort class has a method to set baud rate, parity, and the like, while the ParallelPort class has methods for setting the "port mode" to original PC mode, bidirectional mode, etc.

Both subclasses also have methods that allow us to use the standard Java event model to receive notification of events such as data available for reading, output buffer empty and type-specific events such as ring indicator for a serial port and out-of-paper for a parallel port.

3.1.4 Power Distribution Dynamics: the Knapsack Problem

The Knapsack Problem is a classic in computer science. In its simplest form it involves trying to fit items of different weights into a knapsack so that the knapsack ends up with a specified total weight. You don't need to fit in all the items. For example, suppose you want your knapsack to weigh exactly 20 pounds, and you have five items, with weights of 11, 8, 7, 6, and 5 pounds. For small numbers of items, humans are pretty good at solving this problem by inspection. So we can probably figure out that only the 8, 7, and 5 combination of items adds up to 20. If we want a computer to solve this problem, we'll need to give it more detailed instructions. Here's the algorithm:

1. If at any point in this process the sum of the items selected add up to the target, you're done.

2. Start by selecting the first item. The remaining items must add up to the knapsack's target weight minus the first item; this is a new target weight.

3. Try, one by one, each of the possible combinations of the remaining items. Notice, however, that you don't really need to try all the combinations, because whenever the sum of the items is more than the target weight, you can stop adding items.

4. If none of the combinations work, discard the first item, and start the whole process again with the second item.

5. Continue this with the third item and so on until you've tried all the combinations, at which point you know there is no solution.

In the example just described, we start with 11, Now we want the remaining items to add up to 9 (20 minus 11). Of these, we start with 8, which is too small. Now we want the remaining items to add up to 1 (9 minus 8). We start with 7, but that's bigger than 1, so we try 6 and then 5, which are also too big. We've run out of items, so we know that any combination that includes 8 won't add up to 9. Next we try 7, so now we're looking for a target of 2 (9 minus 7). We continue in the same way, as summarized here:

Items: 11, 8, 7, 6, 5

```
========================================
11          // Target = 20, 11 is too small
11, 8       // Target = 9, 8 is too small
11, 8, 7    // Target = 1, 7 is too big
11, 8, 6    // Target = 1, 6 is too big
11, 8, 5    // Target = 1, 5 is too big. No more items
11, 7       // Target = 9, 7 is too small
11, 7, 6    // Target = 2, 6 is too big
```

11, 7, 5	// Target = 2, 5 is too big. No more items
11, 6	// Target = 9, 6 is too small
11, 6, 5	// Target = 3, 5 is too big. No more items
11, 5	// Target = 9, 5 is too small. No more items
8,	// Target = 20, 8 is too small
8, 7	// Target = 12, 7 is too small
8, 7, 6	// Target = 5, 6 is too big
8, 7, 5	// Target = 5, 5 is just right. Success!

The following code snippet emphasizes a dynamic programming approach to the knapsack problem.

```
static int knap(int M)
  { int i, space, max, maxi = 0, t;
    if (maxKnown[M] != unknown) return maxKnown[M];
    for (i = 0, max = 0; i < N; i++)
      if ((space = M-items[i].size) >= 0)
        if ((t = knap(space) + items[i].val) > max)
          { max = t; maxi = i; }
    maxKnown[M] = max; itemKnown[M] = items[maxi];
    return max;
  }
```

By dynamic programming it reduces the running time from exponential to linear. It simply save any function values that it computes, then retrieves any saved values whenever it needs them (using a sentinel value to represent unknown values), rather than making recursive calls. It saves the index of the item so that, if it wishes, it can reconstruct the contents of the knapsack after the computation: itemKnown[M] is in the knapsack, the remaining contents are the same as for the optimal knapsack of size M-itemKnown[M].size, so itemKnown[M-items[M].size] is in the knapsack, and so forth.

However, in the case of power distributing among various load units, where there are more constraints than considered by the knapsack algorithm, we must consider the possible solution as a variation of this standard algorithm. The following are the aforementioned parameters:

1. The total power for distribution is a constant
2. The power allocated to each load unit has a preset maximum
3. If a load unit shed power below its preset maximum and another requests above its, there should be a dynamic allocation of power to incrementally meet up with this demand
4. If a load unit with a preset maximum that has previously shed power demands for it again, demand should be met in real-time by claiming from one or more of all those using above their preset maximum.
5. Each load unit has a priority value for supply

3.1.5 A Varied Knapsack

It is important to observe that knapsack splits the value in consideration to an *exact change.* Simply stated, consider a_i's are coins, you want to make an exact change of *K*. Maybe there are multiple ways you can do this, then you want to minimize (or maximize) the number of coins you use. Now the knapsack solution is no longer 0 or 1, it is exactly the minimum number of coins we need to hit k. However in this project work, *exact change values* are seldom required. This is due to the fact that more constraints are defined; hence the distribution algorithm will have to cater for more than is considered in the knapsack algorithm.

For instance, load units can demand for power values below their preset maximum. In addition, they may be allocated more than their preset maximum under the circumstance that other load units are running below their expected preset maximum or if there is an excess in the power supply value from the distribution point. All these constraints define undetermined distribution values that will always not equal the total power supply to the system.

3.1.6 System Database:

The system is connected to a database which stores the values of various distribution actions performed within the system against the time the action was performed. The database as the following features:

Database used: Apache Derby (Relational Database)

Language of Implementation: Structure Query Language (SQL)

The database program is shown below

```
create table log(
supplyhostel varchar(10),
supplyoffices varchar(10),
supplylab varchar(10),
demandhostel varchar(10),
demandoffices varchar(10),
demandlab varchar(10),
available varchar(10),
datee varchar(30) primary key  )
```

3.2 Hardware Module (Load Units):

The hardware module comprises of load units which are interfaced to the computer through the communication ports of the microcontroller. It performs the following functions:

1. Displays the load values for each unit via a connected Light Emitting Diode (LED)
2. Varies the load value with time so as to create a continuous simulation of the load consumption at each load units.
3. Sends these load values (in 2 above) to the serial port of the general purpose computer system.

The various functions above were actualized by taking advantage of the multipurpose ability provide by the microcontroller.

This project utilizes a single microcontroller (AT89C51) to perform the functions above by the use of seven displays, Light Emitting Diode (LED) and the RS232 DB-9 connector which are connected to it via its 40 pin ports.

A detailed description of how the various components are interconnected is shown in figure 3.1 below

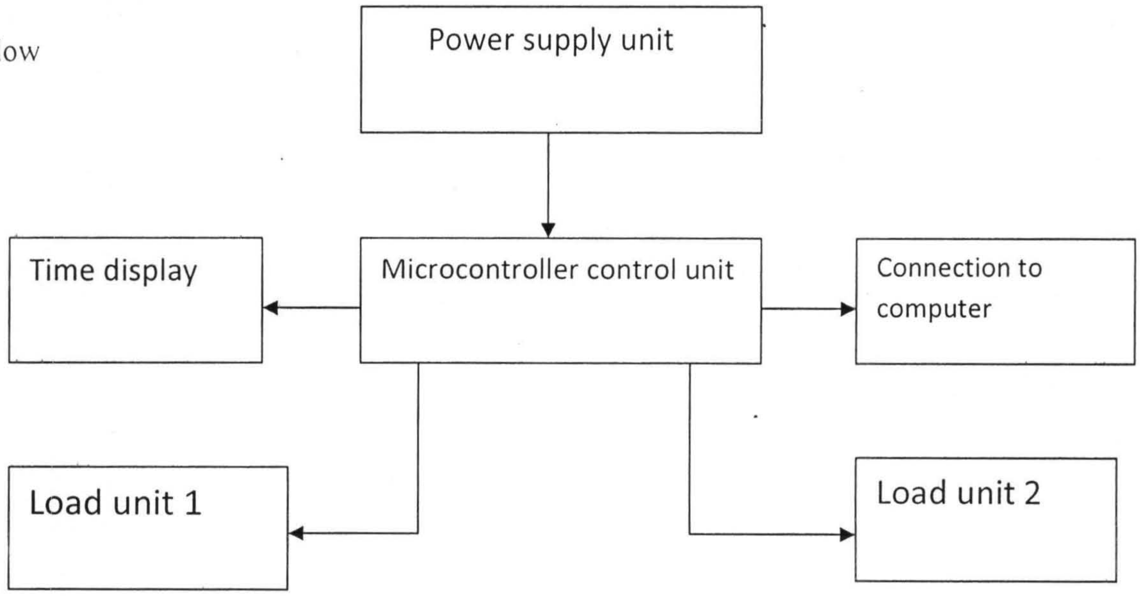


Fig 3.5 Block diagram of Hardware Module

As described above the hardware module can be seen to comprise of the following subsystems:

1. The microcontroller
2. The power supply unit
3. Time display
4. Load units 1 and 2
5. Connection to computer

3.2.2 The Microcontroller:

A microcontroller is a self-contained computer system built on a single silicon chip. It is based on Harvard architecture which offers greater speed for instruction execution. Microcontrollers contain a single microprocessor core, along with all necessary data and program memory. For the purpose of this project the Atmel AT89C2051 microcontroller is used which provides a highly flexible and cost-effective solution to many embedded control applications. The microcontroller program was written in Assembly language using the EdSim51 IDE, Fig 3.5 and Fig 3.6 shows the program flow chart and EdSim51 IDE window.

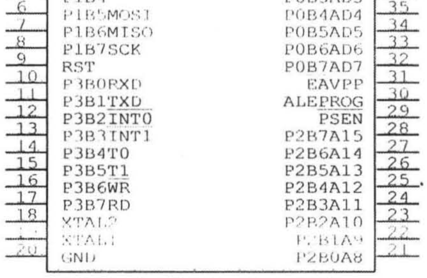


Fig 3.6 Pin description of AT89C51

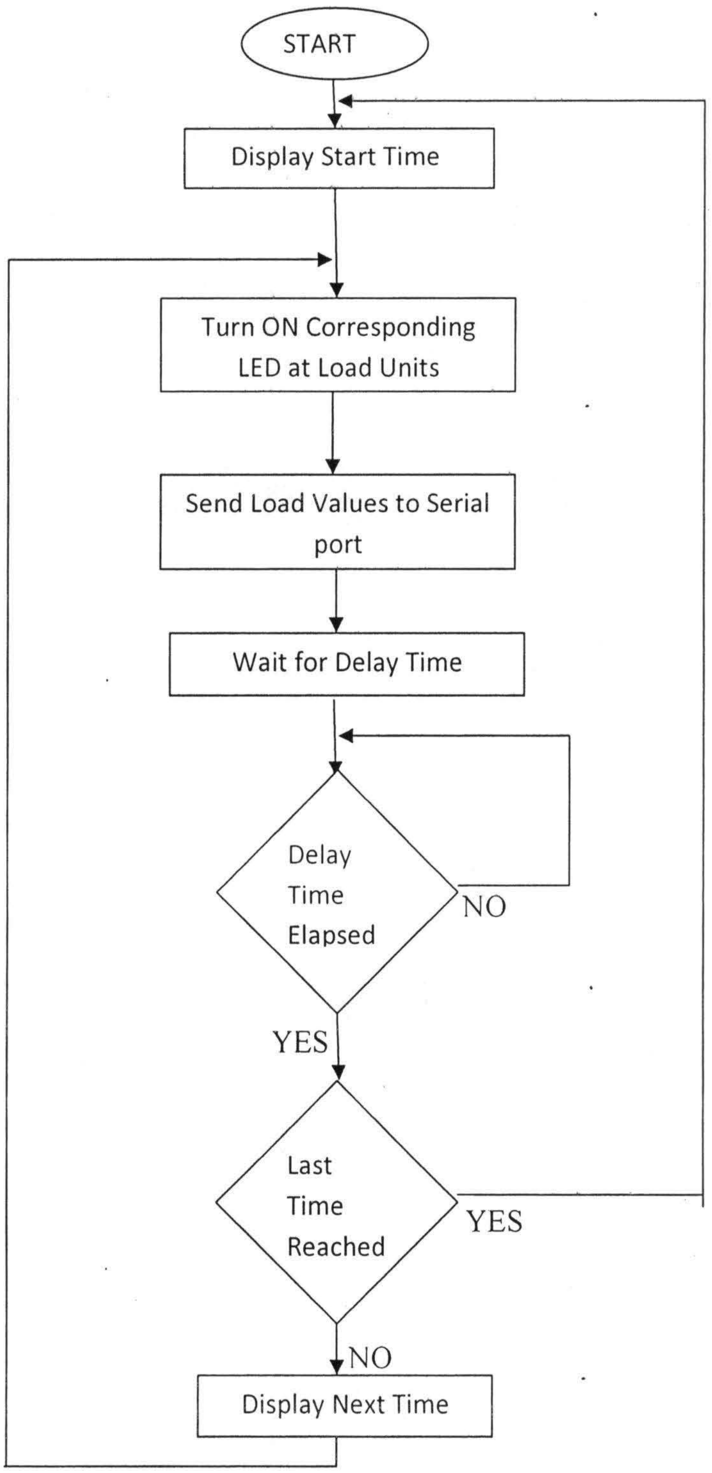
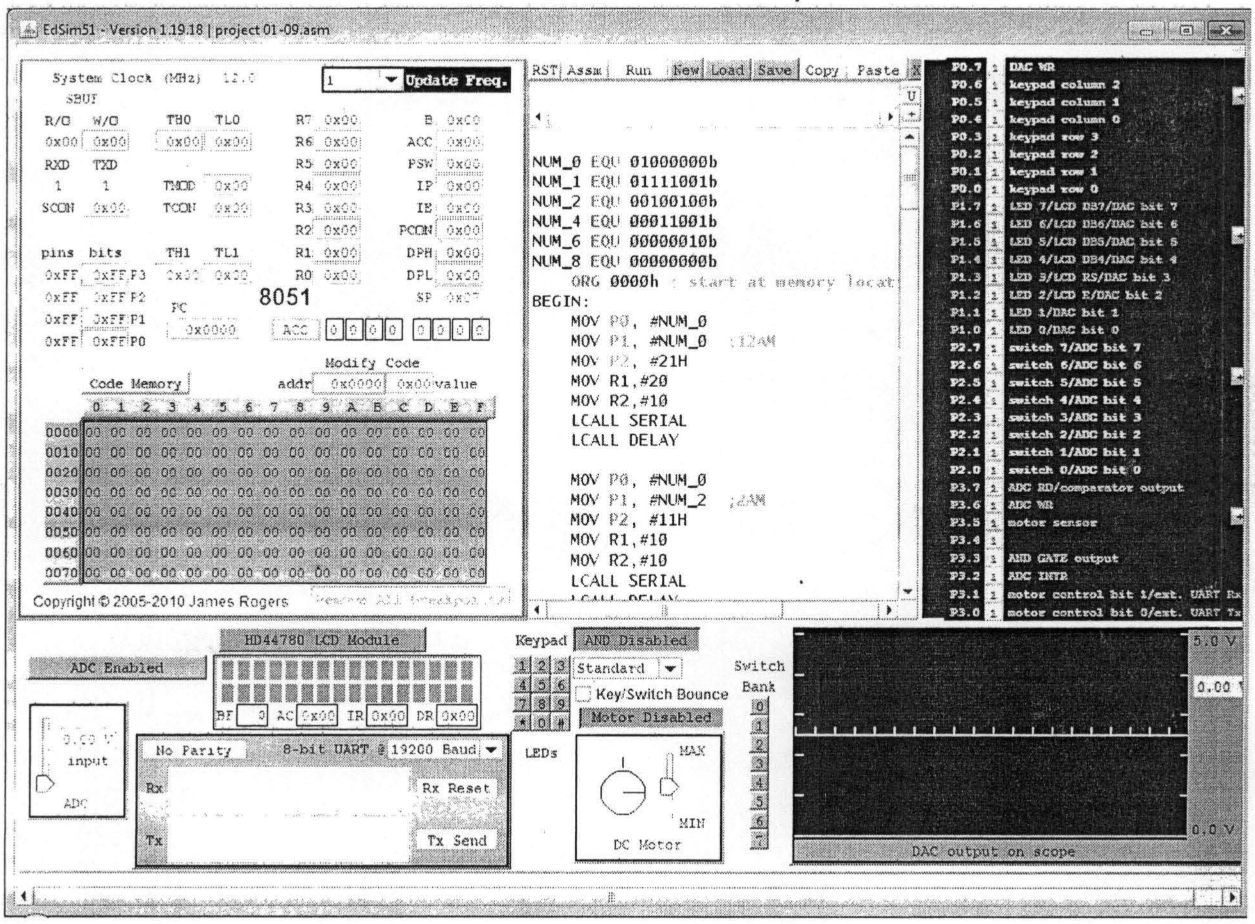


Fig 3.7 Microcontroller Program Flow Chart


EdSim51 - Version 1.19.18 | project 01-09.asm
RST Assm Run New Load Save Copy Paste
8051
HD44780 LCD Module
ADC Enabled
AND Disabled
Motor Disabled
DC Motor
DAC output on scope
Copyright © 2005-2010 James Rogers

```
NUM_0 EQU 01000000b
NUM_1 EQU 01111001b
NUM_2 EQU 00100100b
NUM_4 EQU 00011001b
NUM_6 EQU 00000010b
NUM_8 EQU 00000000b
    ORG 0000h ; start at memory locat
BEGIN:
    MOV P0, #NUM_0
    MOV P1, #NUM_0   ;12AM
    MOV P2, #21H
    MOV R1,#20
    MOV R2,#10
    LCALL SERIAL
    LCALL DELAY

    MOV P0, #NUM_0
    MOV P1, #NUM_2   ;2AM
    MOV P2, #11H
    MOV R1,#10
    MOV R2,#10
    LCALL SERIAL
```

```
P0.7 1 DAC WR
P0.6 1 keypad column 2
P0.5 1 keypad column 1
P0.4 1 keypad column 0
P0.3 1 keypad row 3
P0.2 1 keypad row 2
P0.1 1 keypad row 1
P0.0 1 keypad row 0
P1.7 1 LED 7/LCD DB7/DAC bit 7
P1.6 1 LED 6/LCD DB6/DAC bit 6
P1.5 1 LED 5/LCD DB5/DAC bit 5
P1.4 1 LED 4/LCD DB4/DAC bit 4
P1.3 1 LED 3/LCD RS/DAC bit 3
P1.2 1 LED 2/LCD E/DAC bit 2
P1.1 1 LED 1/DAC bit 1
P1.0 1 LED 0/DAC bit 0
P2.7 1 switch 7/ADC bit 7
P2.6 1 switch 6/ADC bit 6
P2.5 1 switch 5/ADC bit 5
P2.4 1 switch 4/ADC bit 4
P2.3 1 switch 3/ADC bit 3
P2.2 1 switch 2/ADC bit 2
P2.1 1 switch 1/ADC bit 1
P2.0 1 switch 0/ADC bit 0
P3.7 1 ADC RD/comparator output
P3.6 1 ADC WR
P3.5 1 motor sensor
P3.4 1
P3.3 1 AND GATE output
P3.2 1 ADC INTR
P3.1 1 motor control bit 1/ext. UART Rx
P3.0 1 motor control bit 0/ext. UART Tx
```

Fig 3.8 Microcontroller Programming interface

3.5.2 The Power Supply Unit:

For proper functioning of the microcontroller, it is necessary to provide a stable source of supply (5V dc). The power supply unit consists of four stages as shown in fig 3.3. These stages are explained below

i. Transformer

   A 12V step-down transformer would be used to step down mains supply

ii. Rectifier

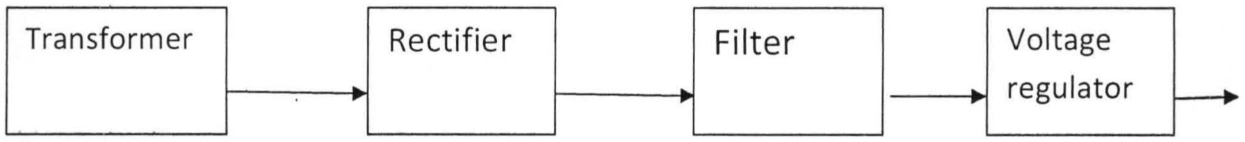
A full-bridge rectifier would be used to convert the 12V transformer output to pulsating 12V dc.

iii. Filter

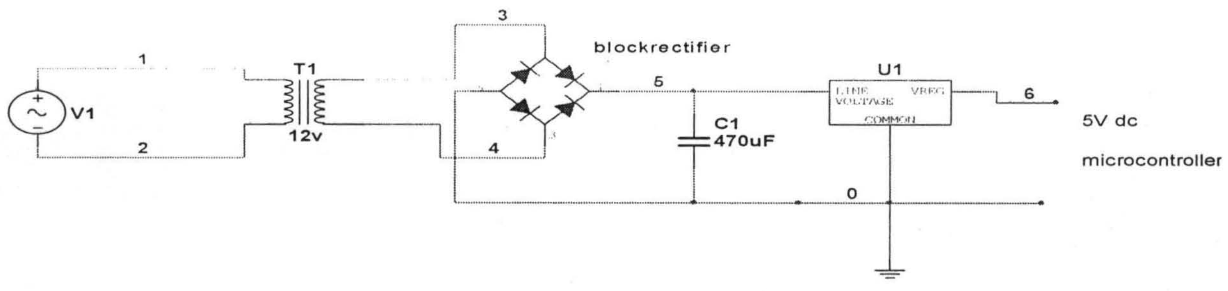
A 2200μF capacitor would be used to remove the fluctuations or ripples present in the output voltage supply of the rectifier.

iv. Voltage regulator

A voltage regulator, LM7805 would be used to keep the terminal voltage of the dc supply constant. This would ensured that voltage remain constant even when the AC input to the transformer varies or the load varies. The LM7805 also acts as voltage divider by reducing the 12V dc to 5V dc, which is required for the proper operation of the microcontroller.



a



b

Fig 3.9 Power Supply Unit

3.2.3 Time display:

This project involves a simulation which illustrates variation in load values at different load units with time over 24-hrs duration. A display of time values and how time is increasing is required. This is achieved by the use of two seven-segment LED display which is connected to the port 0 and port 1 of the microcontroller.

A seven segment display contains seven LED 'bars' that can be lit up in different combinations to show the ten digits 0 to 9. Each 'bar' would be connected to one microcontroller output port, this would use 7 of the 8 available pins of each port.

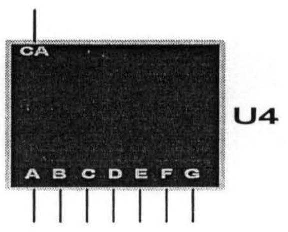


Fig 3.10 A Common-Anode seven segment display

3.2.4 Load Units:

This project illustrates power distribution between three buildings in the Federal University of Technology Minna Gidan-Kwano campus whose daily load distribution is considered as shown in fig 3.5. Each unit can have one of four load values (20, 40, 60 and 80) in megawatt (MW). These values are represented by Light Emitting Diodes (LED) which are connected to port 0 and port 3 of the microcontroller. At every time the microcontroller turns on a corresponding LED to indicate the load at the time.

Time (24hrs)	Hostels	Offices	Workshop
00	40	20	20
02	20	20	20
04	20	20	20
06	40	20	20
08	60	40	40
10	40	60	60
12	20	60	80
14	20	60	40
16	60	40	40
18	80	20	20
20	80	20	20
22	60	20	20
24	40	20	20

Table 3.1 Load table

Light Emitting Diodes are semiconductor components which allow unidirectional flow of current like other diodes but in addition give particular color of illumination when sufficient voltage is applied across its terminals.

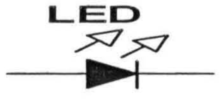


Fig 3.11 Light Emitting Diode

3.2.5 Connection to computer:

The microcontroller communicates with the PC (personal computer) via its two serial communication pins (pin 10 and pin 11). Pin 10 and 11 are used for receiving and transmitting respectively.

An interfacing standard RS232 was set by the Electronics Industries Association (EIA) in 1960 (mc assembly & c). The standard was set long before the advent of the TTL logic family, its input and output voltage levels are not TTL compatible. In RS232, a 1 is represented by -3 ~ -25 V, while a 0 bit is +3 ~ +25 V, making -3 to +3 undefined.

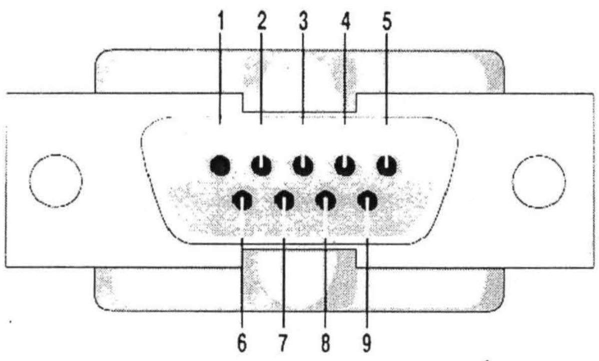


Fig 3.12 RS232 Connection DB-9

Pin	**Description**
1	Data carrier detect (-DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (-DSR)
7	Request to send (-RTS)

8 Clear to send (-CTS)

9 Ring Indicator (RI)

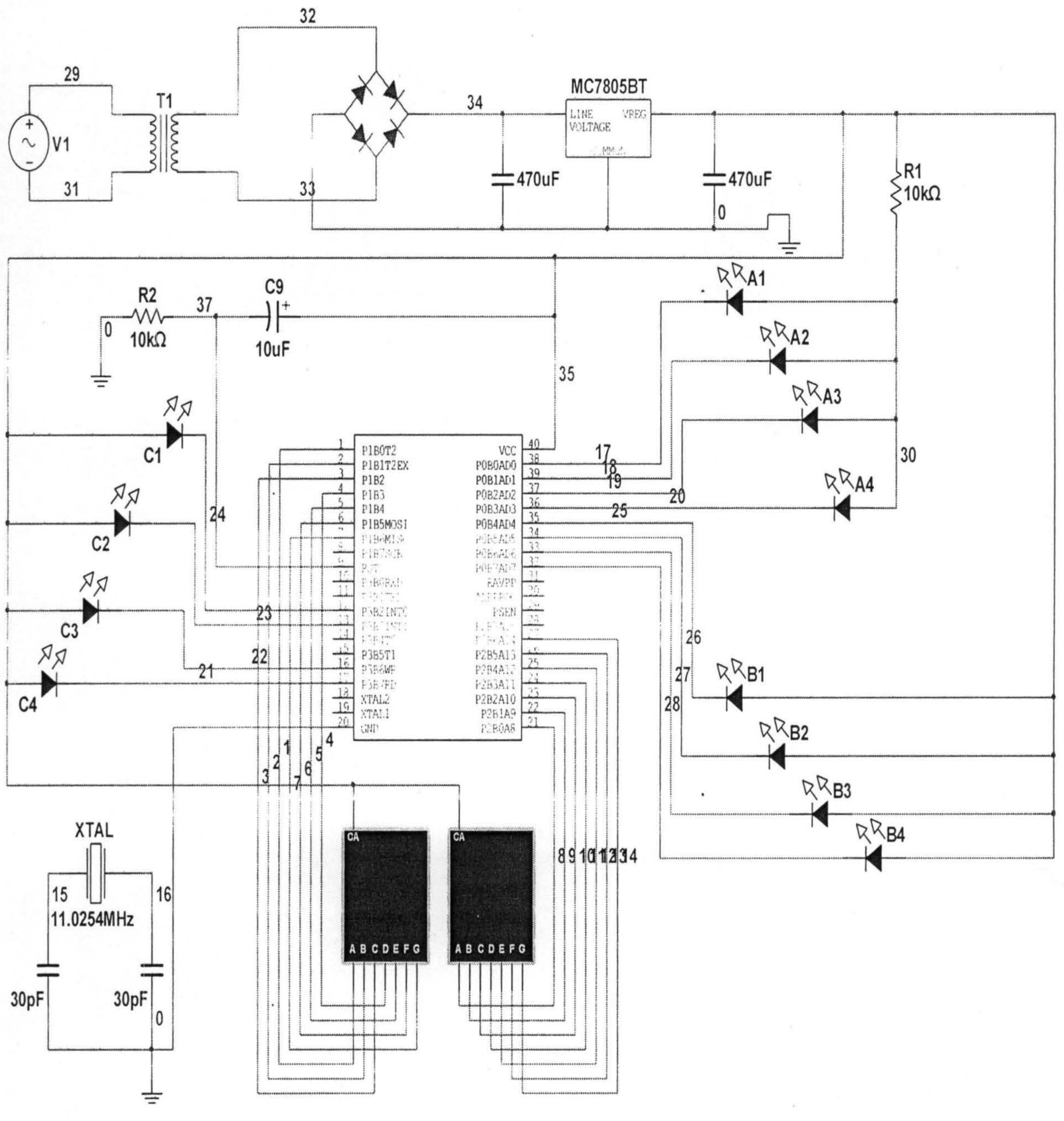


Fig 3.13 Main Circuit Diagram

CHAPTER FOUR

Construction, Testing and Results Discussion

4.1 Circuit Construction:

The construction part of the circuit followed the design steps. Great care was taking in connecting the components in accordance with the circuit diagram. The construction was achieved by dividing the total circuit into units. Each unit was executed one after the other after which the units were put together as a working construction.

4.2 Case Construction:

The casing of the constructed circuit was made with plastic. The material was carefully selected and cut to size for parts of the circuit. The setup was quite flexible to allow for easy mobility and connection to the computer system.

4.3 Software Installation and Interfacing:

The distribution software was installed on a computer system with the Java Development Kit (JDK) since the software was developed in a java language. The system was interfaced with the hardware counterpart via the RS-232 port on the computer system. The software Graphic User Interface (GUI) is shown in fig 3.2.

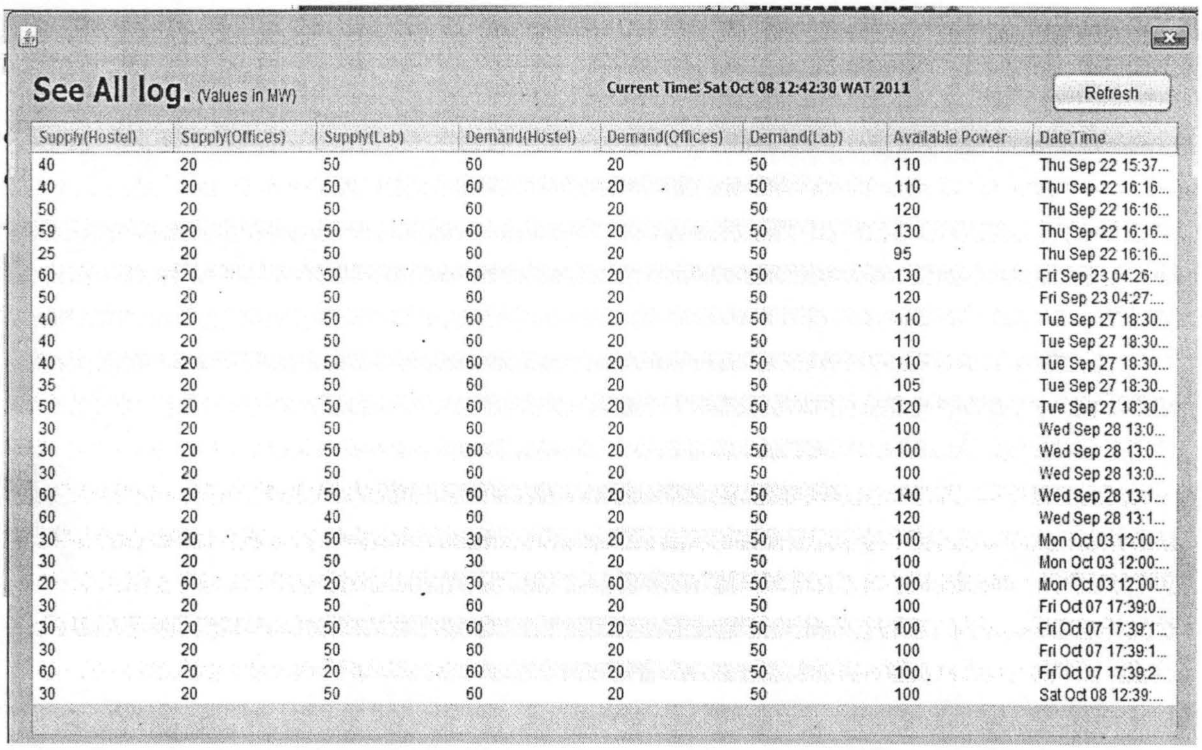
4.4 Testing:

After caring out the various stages of design and construction, the testing was carried out in two stages:

Stage 1: this stage involves the setting up of the circuit and power unit. The various voltage outputs were measured (12V ac Transformer secondary output, 5V dc regulated supply to VCC).

The displays for time and load values at the load units were observed for correctness. Also the software GUI display was observed to correspond to the hardware counterpart.

Stage 2: The testing at this stage was carried out after the circuits have being completed, coupled and interfaced to the computer system. The system software was then initialized and the hardware counterpart activated. The power distribution was obtained for different load values at the load units at different times and the database was updated. The database result is shown below.



See All log. (Values in MW)   Current Time: Sat Oct 08 12:42:30 WAT 2011

Supply(Hostel)	Supply(Offices)	Supply(Lab)	Demand(Hostel)	Demand(Offices)	Demand(Lab)	Available Power	DateTime
40	20	50	60	20	50	110	Thu Sep 22 15:37...
40	20	50	60	20	50	110	Thu Sep 22 16:16...
50	20	50	60	20	50	120	Thu Sep 22 16:16...
59	20	50	60	20	50	130	Thu Sep 22 16:16...
25	20	50	60	20	50	95	Thu Sep 22 16:16...
60	20	50	60	20	50	130	Fri Sep 23 04:26:...
50	20	50	60	20	50	120	Fri Sep 23 04:27:...
40	20	50	60	20	50	110	Tue Sep 27 18:30...
40	20	50	60	20	50	110	Tue Sep 27 18:30...
40	20	50	60	20	50	110	Tue Sep 27 18:30...
35	20	50	60	20	50	105	Tue Sep 27 18:30...
50	20	50	60	20	50	120	Tue Sep 27 18:30...
30	20	50	60	20	50	100	Wed Sep 28 13:0...
30	20	50	60	20	50	100	Wed Sep 28 13:0...
30	20	50	60	20	50	100	Wed Sep 28 13:0...
60	20	50	60	20	50	140	Wed Sep 28 13:1...
60	20	40	60	20	50	120	Wed Sep 28 13:1...
30	20	50	30	20	50	100	Mon Oct 03 12:00:...
30	20	50	30	20	50	100	Mon Oct 03 12:00:...
20	60	20	30	60	50	100	Mon Oct 03 12:00:...
30	20	50	60	20	50	100	Fri Oct 07 17:39:0...
30	20	50	60	20	50	100	Fri Oct 07 17:39:1...
30	20	50	60	20	50	100	Fri Oct 07 17:39:1...
60	20	20	60	20	50	100	Fri Oct 07 17:39:2...
30	20	50	60	20	50	100	Sat Oct 08 12:39:...

Fig 4.1 System Database

4.5 Result Discussion:

The time and load displays were obtained as expected from the load table in fig 3.5 and the software corresponding power distribution were obtained for each time and load value. These result shows that power can better be distributed to consumers based on there instantaneous demand instead of the static distribution available in the existing electric grid. These will reduce energy wastage incurred over power distributed to consumer when they have no demand for it and also reduce running cost on the system, reduce consumer bills among other advantages such as increased supply capability, improved reliability improved system performance.

CHAPTER FIVE

Conclusions

The project work is the modeling of Demand Response System (DRS) an integral of smart grid. The Demand Response System is a software which is able to utilize the two-way communication feature provided by the smart grid to receive consumer varying power demand and distribute power to consumers based on there instantaneous demand.

The design considers power distribution to three units which represents student hostel, office building and workshop. The idea of the project is to be able to vary power supplied to each unit such that despite assigning a fixed value to a unit, the supply to that unit can be reduced if the unit is not using up to the assigned value. Hence, the system can use the available power to supply other units whose current demand is higher than there assigned value.

Nevertheless, a unit can reclaim assigned value when its demand is increased as long as the demand is within assigned limit. Also the system gives priority to some units which are of high criticality such as Network Operating Centre (NOC), this priority can also be varied by the operator at anytime to meet the demand of areas with highest priority at the time.

5.1 Problems Encountered

1. It was very challenging obtaining a suitable algorithm that will efficiently implement the power distribution due to the complex nature of the distribution parameters.

2. The number of load units was limited to the available ports on the microcontroller.

3. There were few reference materials that provide required information because the smart grid is a new idea in the field of power engineering.

5.2 Recommendation

The smart grid technology is a new idea in power system engineering and the world at large. Various aspect of the power system have been considered and modernize in this new grid, this project only considers one of them which is the Demand Response. Other aspects include load control, renewable source of energy, consumer participation and several others

For further research into this project the following improvements are recommended.

1. The system should show the effect the varying power distribution on an hardware system

2. The system can be modified to include load control at individual units (load units)

3. A better illustration of load values at the units can be made to include varying voltage levels

4. The demand response system should illustrate more numbers of load units

5. The system should be able to increase its capacity to accommodate new units.

REFERENCES

1. Power System Engineering (ECE523) lecture note. Department of Electrical/ Computer Engineering ,F.U.T minna,2011, pp.2-3s

2. Kathy Kowalenko "The Smart Grid : A Primer", IEEE The Institute journal ,vol. 34 No. 4 December 2010, pp.5-6

3. Power System Engineering (ECE523) lecture note. Department of Electrical/ Computer Engineering ,F.U.T minna,2011, pp.3

4. Power System Engineering (ECE523) lecture note. Department of Electrical/ Computer Engineering ,F.U.T minna,2011, pp.5

5. http//en.wikipedia.org/wiki/War of Currents#history

6. Bruce Hamilton and Mohammad Shahidehpour "working together: international smart grid collaboration" IEEE Power & energy magazine, vol. 9 No. 1 January/February 2011

7. Power System Engineering (ECE523) lecture note. Department of Electrical/ Computer Engineering ,F.U.T minna,2011, pp.8

8. http://southwesternontario.ctv.ca/news.php?id=6369

9. Smart Grid Working Group (2003-06) (pdf). Challenge and Opportunity: Charting a New Energy Future, Appendix A:.Working Group Reports. Energy Future Coalition. http://www.energyfuturecoalition.org/files/webfmuploads/EFC_Report/EFCReport.pdf. Retrieved 2008-11-27.

10. Federal Energy Regulatory Commission staff report (2006-08) (pdf). Assessment of Demand Response and Advanced Metering (Docket AD06-2-000). United States Department of Energy. p. 20. http://www.ferc.gov/legal/staff-reports/demand-response.pdf. Retrieved 2008-11-27.

11. National Energy Technology Laboratory (2007-07-27) (pdf). A Vision for the Modern Grid. United States Department of Energy. p. 5. http://www.netl.doe.gov/moderngrid/docs/A%20Vision%20for%20the%20Modern%20Grid_Final_v1_0.pdf. Retrieved 2008-11-27.

12. Anderson, Roger; A. Boulanger, J. A. Johnson and A. Kressner (2008 ISBN 978-1-59370-157-4). p 333. Computer-Aided Lean Management for the Energy Industry.

13. Energy Future Coalition, "Challenge and Opportunity: Charting a New Energy Future," Appendix A: Working Group Reports, Report of the Smart Grid Working Group. http://www.energyfuturecoalition.org/pubs/app_smart_grid.pdf

14. Energy Future Coalition, "Challenge and Opportunity: Charting a New Energy Future," Appendix A: Working Group Reports, Report of the Smart Grid Working Group. http://www.energyfuturecoalition.org/pubs/app_smart_grid.pdf

APPENDIX

Microcontroller Source Code

```
ORG 0000H

LJMP MAIN

MAIN:

MOV TMOD, #00100000B

MOV TMOD,#00100010B

MOV TL0,#6

MOV TH0,#6

MOV TH1, #0E6H

MOV TL1, #0E6H

SETB TR0

SETB TR1

MOV SCON, #01010000B

;-------------------------------------------

START:

MOV P1, #0FFH

MOV P2, #0FFH

MOV P3, #0FFH
```

```
;----------------------------------------------

DIS1:

MOV P1, #11000000B

MOV P2, #11000000B

CLR P0.1

CLR P0.4

CLR P3.4

MOV 30H, #'B'

MOV 31H, #'A'

MOV 32H, #'A'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.1

SETB P0.4

SETB P3.4
```

```
;-----------------------------------------------

DIS2:

MOV P1, #11000000B

MOV P2, #10100100B

CLR P0.0

CLR P0.4

CLR P3.4

MOV 30H, #'A'

MOV 31H, #'A'

MOV 32H, #'A'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.0

SETB P0.4

SETB P3.4

;---------------------------------------------

DIS3:

MOV P1, #11000000B

MOV P2, #10011001B

CLR P0.0

CLR P0.4

CLR P3.4

MOV 30H, #'A'

MOV 31H, #'A'

MOV 32H, #'A'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.0

SETB P0.4

SETB P3.4

;---------------------------------------------

DIS4:

MOV P1, #11000000B

MOV P2, #10000010B
```

```
CLR P0.1

CLR P0.4

CLR P3.4

MOV 30H, #'B'

MOV 31H, #'A'

MOV 32H, #'A'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.1

SETB P0.4

SETB P3.4

;-----------------------------------------------

DIS5:

MOV P1, #11000000B

MOV P2, #10000000B

CLR P0.2

CLR P0.5

CLR P3.5

MOV 30H, #'C'

MOV 31H, #'B'

MOV 32H, #'B'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.2

SETB P0.5

SETB P3.5

;-------------------------------------------

DIS6:

MOV P1, #11111001B

MOV P2, #11000000B

CLR P0.1

CLR P0.6

CLR P3.6
```

```
MOV 30H, #'B'
MOV 31H, #'C'
MOV 32H, #'C'
MOV 33H, #' '
MOV 34H, #0
LCALL SEND
LCALL DELAY1
SETB P0.1
SETB P0.6
SETB P3.6
;------------------------------------------------
DIS7:
MOV P1, #11111001B
MOV P2, #10100100B
CLR P0.0
CLR P0.6
CLR P3.7
MOV 30H, #'A'
MOV 31H, #'C'
MOV 32H, #'D'
MOV 33H, #' '
MOV 34H, #0
LCALL SEND
LCALL DELAY1
SETB P0.0
SETB P0.6
SETB P3.7
;----------------------------------------------
;------------------------------------------------
DIS9:
MOV P1, #11111001B
MOV P2, #10011001B
CLR P0.2
CLR P0.5
CLR P3.5
MOV 30H, #'A'
MOV 31H, #'C'
MOV 32H, #'D'
MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.2

SETB P0.5

SETB P3.5

;----------------------------------------------

DIS10:

MOV P1, #11111001B

MOV P2, #10000010B

CLR P0.3

CLR P0.4

CLR P3.4

MOV 30H, #'C'

MOV 31H, #'B'

MOV 32H, #'B'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.3

SETB P0.4

SETB P3.4

;---------------------------------------------------

DIS11:

MOV P1, #11111001B

MOV P2, #10000000B

CLR P0.3

CLR P0.4

CLR P3.4

MOV 30H, #'D'

MOV 31H, #'A'

MOV 32H, #'A'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1
```

```
SETB P0.3

SETB P0.4

SETB P3.4

;-----------------------------------------------

DIS12:

MOV P1, #10100100B

MOV P2, #11000000B

CLR P0.2

CLR P0.4

CLR P3.4

MOV 30H, #'D'

MOV 31H, #'A'

MOV 32H, #'A'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.2

SETB P0.4

SETB P3.4

;-----------------------------------------------

DIS13:

MOV P1, #10100100B

MOV P2, #10100100B

CLR P0.1

CLR P0.4

CLR P3.4

MOV 30H, #'C'

MOV 31H, #'A'

MOV 32H, #'A'

MOV 33H, #' '

MOV 34H, #0

LCALL SEND

LCALL DELAY1

SETB P0.1

SETB P0.4

SETB P3.4

JMP START
```

```
SEND:

MOV R0, #30H

again:

        MOV A, @R0                    ; move
from location pointed to by R0 to the accumulator

        JZ finish                  ;      if      the
accumulator contains 0, no more data to be sent,
jump to finish

        ;MOV C, P                     ;
otherwise, move parity bit to the carry

        ;MOV ACC.7, C           ; and move the
carry to the accumulator MSB

        MOV SBUF, A                   ; move
data to be sent to the serial port

        INC R0                        ;
increment R0 to point at next byte of data to be sent

        JNB TI, $                     ; wait
for TI to be set, indicating serial port has finished
sending byte

        CLR TI

        CALL TX_DELAY

        ; clear TI

        JMP again                           ; send
next byte

FINISH:

RET

;********************************

TX_DELAY:    MOV R7,#4

             MOV TL0,#0

TX_LOOP:     CLR TF0

             JNB TF0,$

             DJNZ R7,TX_LOOP

             RET

;********************************

DELAY1:

MOV R7, #35

OLA:

LCALL DELAY2

DJNZ R7, OLA

RET

DELAY2:

MOV R2, #10
```

```
M:

MOV R3, #200

O:

MOV R4, #250

DJNZ R4, $

DJNZ R3, O

DJNZ R2, M

RET

END
```

SOFTWARE SOURCE CODE

```
package View;

import java.sql.Statement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Calendar;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DataClass {//169.254.87.46
final String DATABASE_URL = "jdbc:derby://localhost:1527/Segun";
    final String username = "root";
    final String password = "water";
    static final String Driver = "com.mysql.jdbc.Driver";

    StringBuilder str = new StringBuilder();

    Connection con=null;
    Statement stat=null;
    ResultSet result = null;
    PreparedStatement ps = null;

    public void connect()
    {
     try {
        //Class.forName( Driver );
con=DriverManager.getConnection(DATABASE_URL, username, password);
        stat = con.createStatement();
        }
    catch (SQLException ex) {

Logger.getLogger(DataClass.class.getName()).log(L
evel.SEVERE, null, ex);

       }

   }

   public void closeWithResultset()

   {

    try{

       stat.close();//stat = null;

       con.close(); //con = null;

       result.close();// result = null;

    }

    catch(SQLException ex)

    {

    ex.getMessage();

    }

   }

   public void close()

   {

    try{

       stat.close();// stat = null;

       con.close(); // con = null;

    }

    catch(SQLException ex)

    {

    ex.getMessage();

    }

   }


   public        boolean        SignUp(String
senderUsername,String   recipientUsername,String
message)//wrong parameters

   {

    boolean bool = false;

   try{

   connect();

   String  query  =  "INSERT  INTO  Inbox
(Username,SenderUsername,Message,TimeReceived
)                                    VALUES
('"+recipientUsername+"','"+senderUsername+"','"+m
essage+"',"'+message.toString()+"')"; // confirm this
query

    try {

        int                    rst                    =
stat.executeUpdate(query);//System.out.println("done
");

        if(rst == 1){bool = true;}

        }

        catch (SQLException ex) {

        }

    close();

    }catch(Exception ex)

    {

    }

    return bool;

    }



    public void deleteLogin(String username)

    {

    // connect();

        // boolean bool = false;

    try{

    connect();

    String query = "Delete from login where
username = '"+username+"'"; // confirm this query

    try {

        int                    rst                    =
stat.executeUpdate(query);//System.out.println("done
");

        if(rst == 1){

            //bool = true;

        }

        }

        catch (SQLException ex) {

        }

    close();



    }catch(Exception ex)

    {
```

```
package View;

/**
 *
 * @author KINGDAVID
 */
public class Point {

    int priority;
    int maxDemand;
    int curDemand;
    int curSupply;
    int type;

    public int getType() {
        return type;
    }

    public void setType(int type) {
        this.type = type;
    }

    public int getCurDemand() {
        return curDemand;
    }

    public void setCurDemand(int curDemand) {
        this.curDemand = curDemand;
    }

    public int getCurSupply() {
        return curSupply;
    }

    public void setCurSupply(int curSupply) {
        this.curSupply = curSupply;
    }
    public int getMaxDemand() {
        return maxDemand;
    }

    public void setMaxDemand(int maxDemand) {
        this.maxDemand = maxDemand;
    }

    public int getPriority() {
        return priority;
    }

    public void setPriority(int priority) {
        this.priority = priority;
    }
}
    }
package View;
```

```
/**
 *
 * @author KINGDAVID
 */
public class SortPoint {

    public static Point[] mergeSort(Point[] data) {
        return mergeSortHelper(data, 0, data.length - 1);
    }

    protected static Point[] mergeSortHelper(Point[]
data, int bottom, int top) {
        if (bottom == top) {
            return new Point[]{data[bottom]};
        } else {
            int midpoint = (top + bottom) / 2;
            return merge(mergeSortHelper(data, bottom,
midpoint), mergeSortHelper(data, midpoint + 1,
top));
```