MINIMIZATION POWER CONSUMPTION USING LabVIEW

A minor project report submitted by

D. AKHIL KRISHNA **15J41A0212**

K. BHAVANI **15J41A0226**

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

ELECTRICAL AND ELECTRONICS ENGINEERING

Under the esteemed guidance of

Dr M. MAHESWARI
PROFESSOR

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
MALLAREDDY ENGINEERING COLLEGE (Autonomous)
(An Autonomous Institution, Approved by UGC and Affiliated to JNTUH, Approved by AICTE, Accredited by NAACwith 'A' Grade (II Cycle) and NBA, Recipient of World Bank Assistance under TEQIP phase – II S.C.1.1) Maisammaguda, Dhulapally (Post. Via. Kompally), Secunderabad – 500 100 Telangana State.

2015-19

MALLA REDDY ENGINEERING COLLEGE(Autonomous)

(An Autonomous Institution, Approved by UGC and Affiliated to JNTUH,
Approved by AICTE, Accredited by NAAC with 'A' Grade (II Cycle) and NBA,
Recipient of World Bank Assistance under TEQIP phase – II S.C.1.1)
Maisammaguda, Dhulapally (Post. Via. Kompally), Secunderabad – 500 100, Telangana State

DEPARTMENT OF _____

CERTIFICATE

This is to certify that this Minor Project work entitled "**Minimization of Power Consumption Using LabVIEW**" is the bonafide work of **D. Akhil Krishna (15J41A0212), K. Bhavani (15J41A0226)** in partial fulfillment for the award of **Bachelor of Technology** in ***Electrical & Electronics Engineering*** to the **Jawaharlal Nehru Technological University, Hyderabad** during the academic period **2015-2019** under my guidance and supervision. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

SIGNATURE
Dr. M. MAHESWARI
PROJECT GUIDE
Department of EEE
MREC(A)
Secunderabad – 500100.

SIGNATURE
Dr. N. RAJESWARAN
PROFESSOR & HEAD
Department of EEE
MREC(A)
Secunderabad – 500100.

Submitted for the Mini project viva voce held on……..

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We are very grateful to our Internal guide **Dr. M. MAHESWARI,** Professor, Department of Electrical & Electronics Engineering, Malla Reddy Engineering College (Autonomous), for her extensive patience and guidance for our Minor project.

We would like to thank **Ms. S. SUNANDA,** Assistant Professor & Minor project Coordinator, Department of Electrical & Electronics and Engineering, Malla Reddy Engineering College (Autonomous), for the timely suggestions and motivation during the course of this work.

We would like to thank **Dr. N. RAJESWARAN,** Professor & Head, Department of Electrical and Electronics Engineering, Malla Reddy Engineering College (Autonomous), for having provided the freedom to use all the facilities available in the department.

We are happy to express our deep sense of gratitude to **Dr. S. SUDHAKARA REDDY,** Principal, Malla Reddy Engineering College (Autonomous), for having provided us with adequate support to undergo Minor project.

We sincerely thank to all the teaching and non-teaching staff of the Department of Electrical and Electronics Engineering for their constant support during the course of this work.

We thank parents for their moral support and the Almighty for his bountiful mercy and graces showered on us during the period of our Minor project, without which nothing could have been possible.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at right time for the development and success of this work.

D. AKHIL KRISHNA **15J41A0212**

K. BHAVANI **15J41A0226**

CONTENTS

ABSTRACT

In Electrical Engineering, power consumption refers to the electrical energy per unit time supplied to operate something, such as a home appliance. Power consumption is usually measured in units of watts (W) or kilowatts. It is our responsibility to conserve electricity instead of wasting. We can conserve energy in many ways and at the many different levels of energy consumption. At home, simple actions such as turning off the lights and unplugging computers or turning off televisions can help us reduce our consumption and thereby conserve energy**.** At the commercial level, updating older buildings with new, more efficient technology can help reduce the amounts of electricity required to power a business. The Main Objective of this report is to minimize or to reduce the Extra Power consumed by two Sample Classes and Labs than the required by making use of day sequences and setting the time table for each day using **LabVIEW** Software and further extend it. Objectives and Concept are briefly explained further.

15J41A0212

15J41A0226

Internal Guide

(Dr M. Maheswari)

HOD/EEE

(Dr. N. Rajeswaran)

LIST OF FIGURES

CHAPTER-1

ENERGY CONSERVATION

1.1 INTRODUCTION

Energy conservation is the effort made to reduce the consumption of energy by using less of an energy service. This can be achieved either by using energy more efficiently (using less energy for a constant service) or by reducing the amount of service used (for example, by driving less). Energy conservation is a part of the concept of eco-sufficiency. Energy conservation reduces the need for energy services and can result in increased environmental quality, national security, personal financial security and higher savings. It is at the top of the sustainable energy hierarchy. It also lowers energy costs by preventing future resource depletion.

Energy can be conserved by reducing wastage and losses, improving efficiency through technological upgrades and improved operation and maintenance. On a global level energy use can also be reduced by the stabilization of population growth.

Energy can only be transformed from one form to other, such as heat energy to motive power in cars, or kinetic energy of water flow to electricity in hydroelectric power plants. However, machines are required to transform energy from one form to other. The wear and friction of the components of these machines while running cause loss of quadrillions of BTU and $500 billion in industries only in USA. It is possible to minimize these losses by adopting green engineering practices to improve life cycle of the components.

1.2 Energy tax

Some countries employ energy or carbon taxes to motivate energy users to reduce their consumption. Carbon taxes can force consumption to shift to nuclear power and other energy sources that carry different sets of environmental side effects and limitations. On the other hand, taxes on all energy consumption can reduce energy use across the board while reducing a broader array of environmental consequences arising from energy production. The state of California employs a tiered energy tax whereby every consumer receives a baseline energy allowance that carries a low tax. As usage increases above that baseline, the tax increases drastically. Such programs aim to protect poorer households while creating a larger tax burden for high energy consumers.

1.3 Ways to conserve energy

One of the primary ways to improve energy conservation in buildings is to perform an energy audit. An energy audit is an inspection and analysis of energy use and flows for energy conservation in a building, process or system with an eye toward reducing energy input without negatively affecting output. This is normally accomplished by trained professionals and can be part of some of the national programs discussed above. Recent development of smartphone apps enables homeowners to complete relatively sophisticated energy audits themselves.

Building technologies and smart meters can allow energy users, both commercial and residential, to visualize the impact their energy use can have in their workplace or homes. Advanced real-time energy metering can help people save energy by their actions.

In passive solar building design, windows, walls, and floors are made to collect, store, and distribute solar energy in the form of heat in the winter and reject solar heat in the summer. This is called passive solar design or climatic design because, unlike active solar heating systems, it does not involve the use of mechanical and electrical devices.

The key to designing a passive solar building is to best take advantage of the local climate. Elements to be considered include window placement and glazing type, thermal insulation, thermal mass, and shading. Passive solar design techniques can be applied most easily to new buildings, but existing buildings can be retrofitted.

Consumers are often poorly informed of the savings of energy efficient products. A prominent example of this is the energy savings that can be made by replacing an incandescent light bulb with a more modern alternative. When purchasing light bulbs, many consumers opt for cheap incandescent bulbs, failing to take into account their higher energy costs and lower lifespans when compared to modern compact fluorescent and LED bulbs. Although these energy-efficient alternatives have a higher upfront cost, their long lifespan and low energy use can save consumers a considerable amount of money. The price of LED bulbs has also been steadily decreasing in the past five years due to improvements in semiconductor technology. Many LED bulbs on the market qualify for utility rebates that further reduce the price of purchase to the consumer. Estimates by the U.S. Department of Energy state that widespread adoption

of LED lighting over the next 20 years could result in about $265 billion worth of savings in United States energy costs.

In this project the average energy consumed is minimized by syncing the loads with the respective time of usage. The user is controlling the action of loads at a place automatically in the specified time. An example of two Class rooms and 2 Labs with 16 loads is turned ON/OFF with a user defined time table is done in this program to reduce the energy consumed. By this type of method one can easily reduce energy consumed or the unnecessary energy being consumed by loads. The working model of minimization of Power Consumption in the form of simulation is done using LabVIEW Software.

CHAPTER-2

LabVIEW SOFTWARE

2.1 Introduction to LabVIEW

LabVIEW (**Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) is a graphical programming environment which has become prevalent throughout research labs, academia and industry. It is a powerful and versatile analysis and instrumentation software system for measurement and automation. Its graphical programming language called G programming is performed using a graphical block diagram that compiles into machine code and eliminates a lot of the syntactical details. LabVIEW offers more flexibility than standard laboratory instruments because it is software based. Using LabVIEW, the user can originate exactly the type of virtual instrument needed and programmers can easily view and modify data or control inputs. The popularity of the National Instruments LabVIEW graphical dataflow software for beginners and experienced programmers in so many different engineering applications and industries can be attributed to the software's intuitive graphical programming language used for automating measurement and control systems.

LabVIEW programs are called virtual instruments (VIs), because their appearance and operation imitate physical instruments like oscilloscopes. LabVIEW is designed to facilitate data collection and analysis, as well as offers numerous display options. With data collection, analysis and display combined in a flexible programming environment, the desktop computer functions as a dedicated measurement device. LabVIEW contains a comprehensive set of VIs and functions for acquiring, analyzing, displaying, and storing data, as well as tools to help you troubleshoot your code.

All test, measurement and control applications can be divided into three main components and the key to virtual instrumentation is the ability to acquire, analyze and present data. LabVIEW can acquire data using the devices like GPIB, Serial, Ethernet, VXI, PXI Instruments, Data Acquisition (DAQ), PCI extensions for Instrumentation (PXI), Image Acquisition (IMAQ), Motion Control, Real-Time (RT) PXI, PLC (through OPC Server), PDA, and Modular Instruments. To help you analyze your data LabVIEW includes analysis functions for Differential Equations, Optimization, Curve Fitting, Calculus, Linear Algebra, Statistics and so on. Express VIs are specifically designed for measurement analysis, including filtering and spectral analysis. Signal

Processing VIs for Filtering, Windowing, Transforms, Peak Detection, Harmonic Analysis, and Spectrum Analysis are provided. LabVIEW includes the following tools to help in presenting data on the computer; Graphs, Charts, Tables, Gauges, Meters, Tanks, 3D Controls, Picture Control, 3D Graphs and Report Generation. Over the Internet, Web Publishing Tools, Data socket (Windows Only), TCP/IP, VI Server, Remote Panels and Email are available to present data.

LabVIEW can communicate with hardware such as data acquisition, vision, and motion control devices, and GPIB, PXI, VXI, RS-232, and RS-485 devices. LabVIEW also has built-in features for connecting your application to the Web using the LabVIEW Web Server and software standards such as TCP/IP networking and ActiveX. Using LabVIEW, you can create test and measurement, data acquisitions, instrument control, datalogging, measurement analysis, and report generation applications. You also can create stand-alone executables and shared libraries, like DLLs, because LabVIEW is a true 32-bit compiler.

For new programmers, LabVIEW Express technology transforms common measurement and automation tasks into much higher-level, intuitive VIs. With Express technology, thousands of nonprogrammers have taken advantage of the LabVIEW platform to build automated systems quickly and easily. For experienced programmers, LabVIEW delivers the performance, flexibility and compatibility of a traditional programming language such as C or BASIC. In fact, the full-featured LabVIEW programming language has the same constructs that traditional languages have such as variables, data types, objects, looping and sequencing structures, as well as error handling. And with LabVIEW, programmers can reuse legacy code packaged as DLLs or shared libraries and integrate with other software using ActiveX, TCP and other standard technologies.

The LabVIEW Family consists of NI LabVIEW Graphical Programming Software for Measurement and Automation, LabVIEW Real-Time Module, LabVIEW FPGA Module, LabVIEW PDA Module, LabVIEW Datalogging and Supervisory Control Module.

2.2 Advantages of LabVIEW

The following are the advantages of LabVIEW:

- **Graphical user interface:** Design professionals use the drag-and-drop user interface library by interactively customizing the hundreds of built-in user objects on the control's palette.
- **Drag-and-drop built-in functions:** Thousands of built-in functions and IP including analysis and I/O, from the functions palette to create applications easily.
- **Modular design and hierarchical design:** Run modular LabVIEW VIs by themselves or as subVIs and easily scale and modularize programs depending on the application.
- **Multiple high-level development tools:** Develop faster with application specific development tools, including the LabVIEW State chart Module, LabVIEW Control Design and Simulation Module and LabVIEW FPGA Module.
- **Professional Development Tools:** Manage large, professional applications and tightly integrated project management tools; integrated graphical debugging tools; and standardized source code control integration.
- **Multi platforms:** The majority of computer systems use the Microsoft Windows operating system. LabVIEW works on other platforms like Mac OS, Sun Solaris and Linux. LabVIEW
  - applications are portable across platforms.
  - **Reduces cost and preserves investment:** A single computer equipped with LabVIEW is used for countless applications and purposes—it is a versatile product. Complete instrumentation libraries can be created for less than the cost of a single traditional, commercial instrument.
  - **Flexibility and scalability:** Engineers and scientists have needs and requirements that can change rapidly. They also need to have maintainable, extensible solutions that can be used for a long time. By creating virtual instruments based on powerful development software such as LabVIEW, you inherently design an open framework that seamlessly integrates software and hardware. This ensures that your applications not only work well today but that you can easily integrate new technologies in the future.
  - **Connectivity and instrument control:** LabVIEW has ready-to-use libraries for integrating stand-alone instruments, data acquisition devices, motion control and vision products, GPIB/IEEE 488 and serial/RS-232 devices, and PLCs to build a complete measurement and automation solution. Plug-and Play

instrument drivers access the industry's largest source of instrument drivers with several instruments from various vendors.

- **Open environment:** LabVIEW provides the tools required for most applications and is also an open development environment. This open language takes advantage of existing code; can easily integrate with legacy systems and incorporate third party software with .NET, ActiveX, DLLs, objects, TCP, Web services and XML data formats.
- **Distributed development:** Can easily develop distributed applications with LabVIEW, even across different platforms. With powerful server technology you can offload processor-intensive routines to other machines for faster execution, or create remote monitoring and control applications.
- **Visualization capabilities:** LabVIEW includes a wide array of built-in visualization tools to present data on the user interface of the virtual instrument as chart, graphs, 2D and 3D visualization. Reconfiguring attributes of the data presentation, such as colors, font size, graph types, and more can be easily performed.
- **Rapid development with express technology:** Use configuration-based Express VIs and I/O assistants to rapidly create common measurement applications without programming by using LabVIEW signal Express.
- **Compiled language for fast execution:** LabVIEW is a compiled language that generates optimized code with execution speeds comparable to compiled C and develops high-performance code.
- **Simple application distribution:** Use the LabVIEW application builder to create executables(exes) and shared libraries (DLLs) for deployment.
- **Target management:** Easily manage multiple targets, from real-time to embedded devices including FPGAs, microprocessors, microcontrollers, PDAs and touch panels.
- **Object–oriented design:** Use object-oriented programming structures to take advantage of encapsulation and inheritance to create modular and extensible code.
- **Algorithm design:** Develop algorithms using math-oriented textual programming and interactively debug .m file script syntax with LabVIEW Math Script.

CHAPTER-3

BLOCK DIAGRAM AND MAIN THEME OF PROJECT

3.1 Block Diagram Representation:

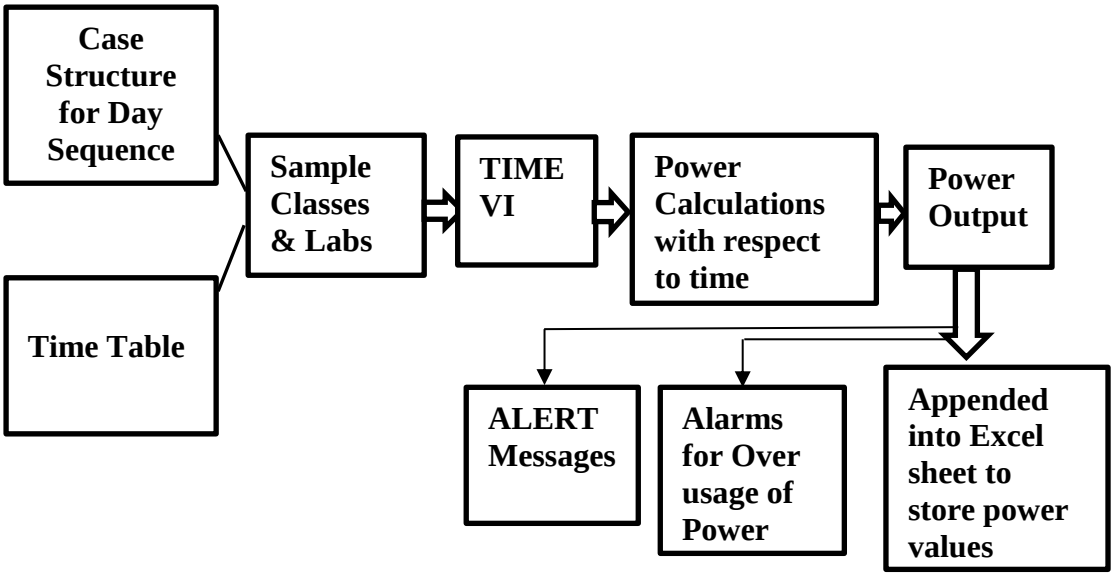


Fig:3.1 Block Diagram of Project

Above Block Diagram explains the whole concept of the Project. Firstly, Sample Classes and laboratories are considered and the lights and fans in each class are indicated by Boolean indicators. According to the Day Sequence and Time table the lights and fans are turned ON and power is Calculated every day with respect to time. This Calculated Power output is given to "Write to Measurement File" in order to store the calculated power values in the Excel Sheet. The Power Output is also given to Alarm Indicators. Incase if the obtained power exceeds the limit then an ALERT message is displayed using String Indicator and alarm is Enabled. Thus, by setting the Time table and Switching on the lights and fans only in the required class rooms can reduce the power consumption to some extent. By making use of Alarm Indicators we can ensure the Over consumption of Power and thus Power consumption can be reduced.

3.2 Main Theme of Project:

In Electrical Engineering, power consumption refers to the electrical energy per unit time supplied to operate something, such as a home appliance. Power consumption

is usually measured in units of watts (W) or kilowatts. It is our responsibility to conserve electricity instead of wasting. We can conserve energy in many ways and at the many different levels of energy consumption. At home, simple actions such as turning off the lights and unplugging computers or turning off televisions can help us reduce our consumption and thereby conserve energy**.** At the commercial level, updating older buildings with new, more efficient technology can help reduce the amounts of electricity required to power a business. The Main Theme of this project is to minimize or to reduce the Extra Power consumed by two Sample Classes and Labs than the required by making use of day sequences and setting the time table for each day using **LabVIEW** Software. Every Individual Class and Lab is provided with separate indicators, separate Alarms and Warning Displays.

3.3 Write to Measurement File:

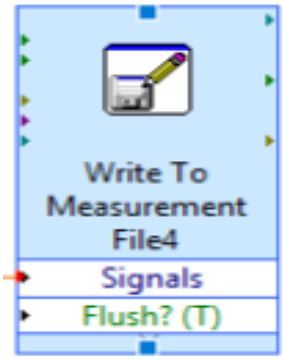


Fig:3.2 Write to Measurement File

Owning Palette: Output Express VIs

Requires: Base Development System

Writes data to text-based measurement files (.lvm), binary measurement files (.tdm or .tdms), or Microsoft Excel files (.xlsx). This Express VI operates similarly to the following VIs and functions:

- Open/Create/Replace File
- Write to Text File
- Write to Binary File
- Write to Spreadsheet File

3.4 FRONT PANEL:

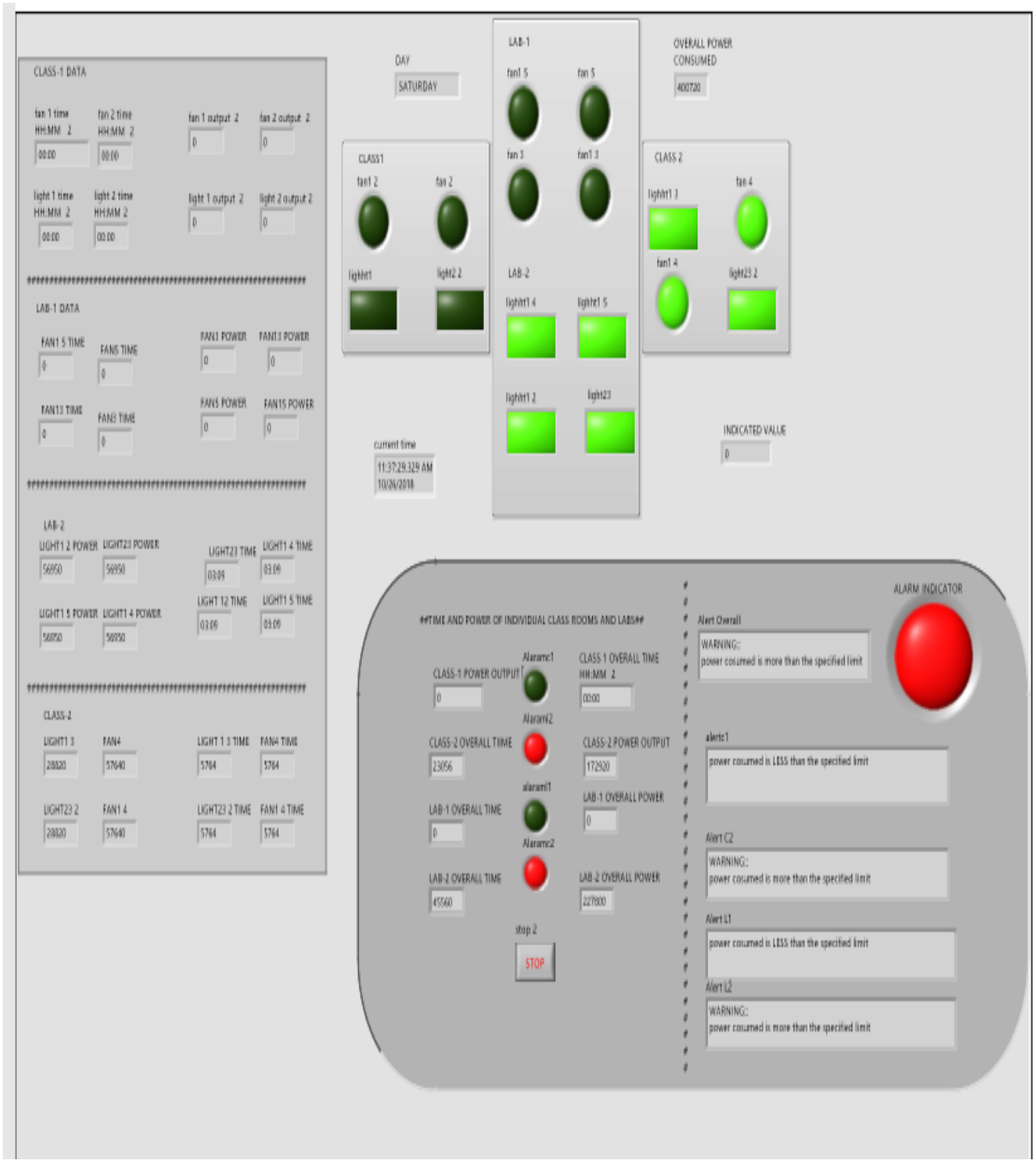


Fig 3.3 Front panel of VI

Above fig 3.3 indicates the front panel of the project. As soon as the program runs day sequence starts from Sunday and the loop repeats. As per the time table scheduled on that day the corresponding loads of class rooms/ Labs will be turned ON. For every class/labs power and time will be calculated and displayed in the Indicators. Daily the same procedure repeats. If the power consumed exceeds the limit alarm is enabled and is indicated by red light as shown in above fig and Warning messages are displayed.

3.5 BLOCK DIAGRAM:

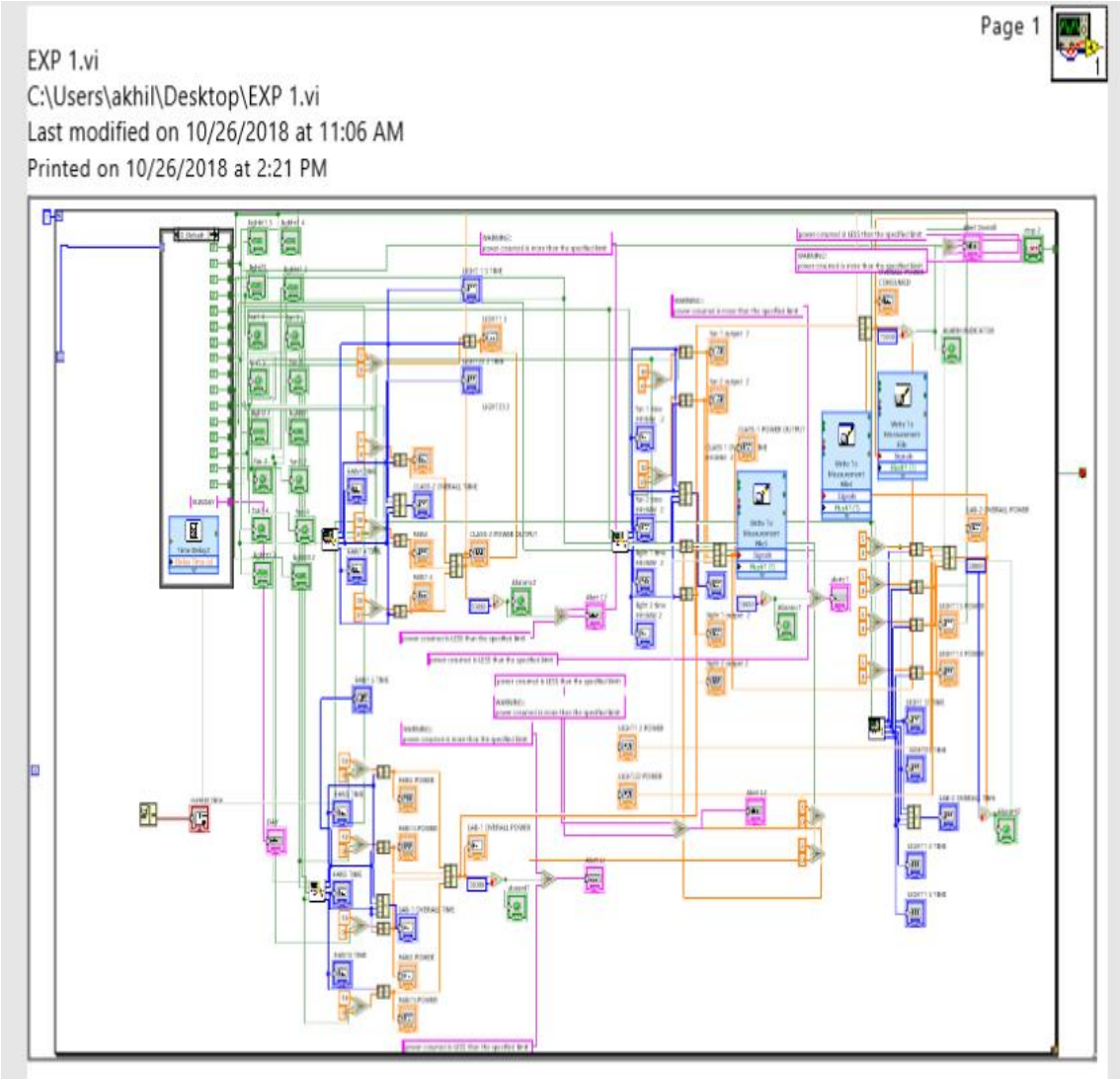


Fig 3.4 MAIN VI

Above fig 3.5 is the Main VI of the Project. 'while' loop is used to run the program N number of times until the condition becomes false. A 'For' loop is inserted inside the while loop which used to repeat the loop for seven times as there are seven days in a week. These are synchronized with Loads and the loads output is given to time VI. Then power and time are multiplied and these energy outputs are added and given to write to measurement file and thus data is stored. Alarms are enabled as per the limits.

CHAPTER-4
CONVERTING VI TO SUB VI

4.1 How to create a VI:

LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution.

In LabVIEW, you build a user interface by using a set of tools and objects. The user interface is known as the front panel. You then add code using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. Every VI uses functions that manipulate input from the user interface or other sources and display that information or move it to other files or other computers.

A VI contains the following three components:

• **Front panel** — Serves as the user interface.

• **Block diagram** — Contains the graphical source code that defines the functionality of the VI.

• **Icon and connector pane** — Identifies the VI so that you can use the VI in another VI. A VI within another VI is called a sub VI. A sub VI corresponds to a subroutine in text-based programming languages.

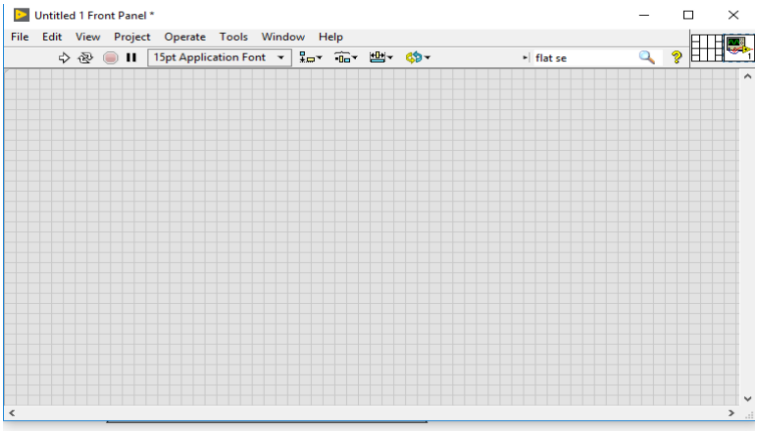


Fig 4.1: Front Panel

The front panel is the user interface of the VI. You build the front panel with controls and indicators, which are the interactive input and output terminals of the VI,

respectively. Controls are knobs, pushbuttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.

After you build the front panel, you add code using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code. Front panel objects appear as 0terminals on the block diagram. Additionally, the block diagram contains functions and structures from built-in LabVIEW VI libraries. Wires connect each of the nodes on the block diagram, including control and indicator terminals, functions, and structures.

4.2 LabVIEW Palettes:

LabVIEW palettes give you the options you need to create and edit the front panel and block diagram. The Tools palette is available on the front panel and the block diagram. A tool is a special operating mode of the mouse cursor. When you select a tool, the cursor icon changes to the tool icon. Use the tools to operate and modify front panel and block diagram objects.

Select **Window» Show Tools Palette** to display the Tools palette. You can place the Tools palette anywhere on the screen.

If automatic tool selection is enabled and you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the Tools palette. The Controls palette is available only on the front panel. The Controls palette contains the controls and indicators you use to create the front panel.

Select **Window» Show Controls Palette** or right-click the front panel workspace to display the Controls palette. You can place the Controls palette anywhere on the screen. The Functions palette is available only on the block diagram. The Functions palette contains the VIs and functions you use to build the block diagram. Select **Window» Show Functions Palette** or right-click the block diagram workspace to display the Functions palette. You can place the Functions palette anywhere on the screen.

4.3 Dataflow Programming:

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when all its inputs are available. When a node completes execution, it supplies data to its output terminals and passes the output data to the next node in the dataflow path.

4.4 Creating Sub VIs:

After you build a VI and create its icon and connector pane, you can use it in another VI. A VI called from the block diagram of another VI is called a sub VI. A sub VI corresponds to a subroutine in text-based programming languages. A sub VI node corresponds to a subroutine call in text-based programming languages. The node is not the sub VI itself, just as a subroutine call statement in a program is not the subroutine itself.

4.5 Setting up the Connector Pane:

To use a VI as a sub VI, you need to build a connector pane. The connector pane is a set of terminals that corresponds to the controls and indicators of that VI, similar to the parameter list of a function call in text-based programming languages. The connector pane defines the inputs and outputs you can wire to the VI so you can use it as a sub VI. Define connections by assigning a front panel control or indicator to each of the connector pane terminals. To define a connector pane, right-click the icon in the upper right corner of the front panel window and select Show Connector from the shortcut menu. The connector pane replaces the icon. Each rectangle on the connector pane represents a terminal. Use the rectangles to assign inputs and outputs. The number of terminals LabVIEW displays on the connector pane depends on the number of controls and indicators on the front panel.

The connector pane has, at most, 28 terminals. If your front panel contains more than 28 controls and indicators that you want to use programmatically, group some of them into a cluster and assign the cluster to a terminal on the connector pane. Select a different terminal pattern for a VI by right-clicking the connector pane and selecting Patterns from the shortcut menu. Select a connector pane pattern with extra terminals. You can leave the extra terminals unconnected until you need them. This flexibility enables you to make changes with minimal effect on the hierarchy of the VIs.

If you create a group of sub VIs that you use together often, give the sub VIs a consistent connector pane with common inputs in the same location to help you remember where to locate each input. If you create a sub VI that produces an output another sub VI uses as the input, align the input and output connections to simplify the wiring patterns. Place the error in clusters on the lower left corner of the front panel and the error out clusters on the lower right corner.

Setting Required, Recommended, and Optional Inputs and Outputs

You can designate which inputs and outputs are required, recommended, and optional to prevent users from forgetting to wire sub VI connections. Right-click a terminal in the connector pane and select This Connection Is from the shortcut menu. A checkmark indicates the terminal setting. Select Required, Recommended, or Optional. For terminal inputs, required means that the block diagram on which the sub VI is dropped will be broken if the required inputs are not wired. Required is not available for terminal outputs. For terminal inputs and outputs, recommended means that the block diagram on which the sub VI is dropped can be run if the input is not wired but the Warnings dialog box will generate a warning that the input has not been wired. Optional means that the block diagram on which the sub VI is dropped can be run and will not generate any warnings if the terminal input or output is not wired.

Inputs and outputs of VIs in vi.lib are already marked as Required, Recommended, or Optional. LabVIEW sets inputs and outputs of VIs you create to Recommended by default. Set a terminal setting to required only if the VI must have the input or output to run properly. In the Context Help window, required connections are bold, recommended connections are plain text, and optional connections are dimmed if you have the Detailed view selected or do not appear if you have the Simple view selected.

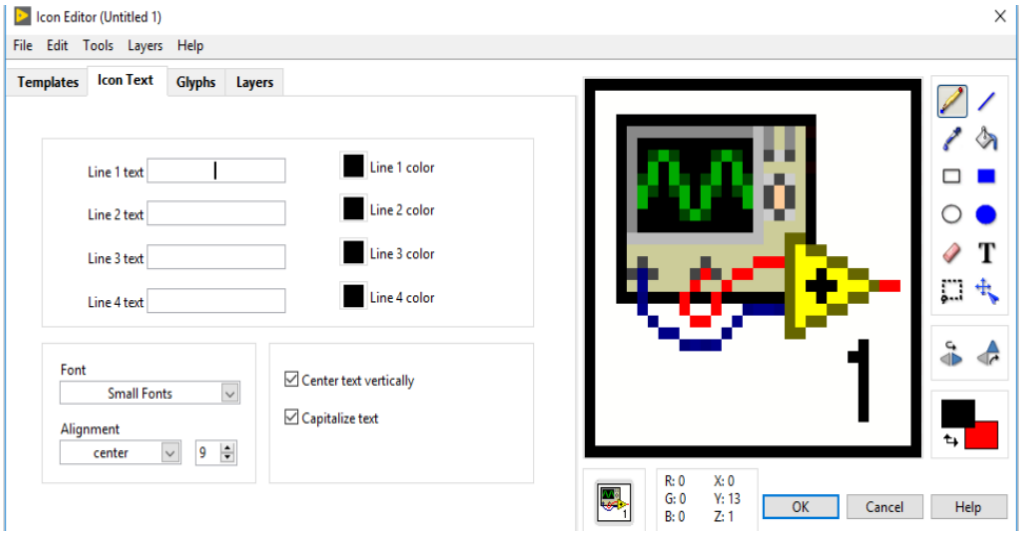


Fig 4.2: Sub VI icon

Every VI displays an icon in the upper right corner of the front panel and block diagram windows. An icon is a graphical representation of a VI. It can contain text, images, or a combination of both. If you use a VI as a sub VI, the icon identifies the sub VI on the block diagram of the VI. The default icon contains a number that indicates how many new VIs you have opened since launching LabVIEW. Create custom icons to replace

the default icon by right clicking the icon in the upper right corner of the front panel or block diagram and selecting Edit Icon from the shortcut menu or by double-clicking the icon in the upper right corner of the front panel. You also can drag a graphic from anywhere in your file system and drop it in the upper right corner of the front panel or block diagram. LabVIEW converts the graphic to a 32 X 32-pixel icon. Depending on the type of monitor you use, you can design a separate icon for monochrome, 16-color, and 256-color mode. LabVIEW uses the monochrome icon for printing unless you have a color printer.

4.6 Creating Sub VIs from Sections of a VI

Convert a section of a VI into a sub VI by using the Positioning tool to select the section of the block diagram you want to reuse and selecting **Edit» Create» Sub VI**. An icon for the new sub VI replaces the selected section of the block diagram. LabVIEW creates controls and indicators for the new sub VI and wires the sub VI to the existing wires. Creating a sub VI from a selection is convenient but still requires careful planning to create a logical hierarchy of VIs. Consider which objects to include in the selection and avoid changing the functionality of the resulting VI.

CHAPTER-5
TIME SYNCHRONIZATION

5.1 ALGORITHM:

In this chapter, a VI is created that can be used later as a sub-VI.

1.Start

2. Convert the Output of tick count to Double precision float by DBL palette [to get float values].

3. To the select switch in which true statement is '0' false statement is time lapse.

4. From the tick count the initial value of time in milli seconds is subtracted from the final value of time in milli seconds(ms).

5. Boolean switch enables the time lapse of particular load to pass through while loop to indicator.

6. Pass "t" Statement to while loop.

7. End.

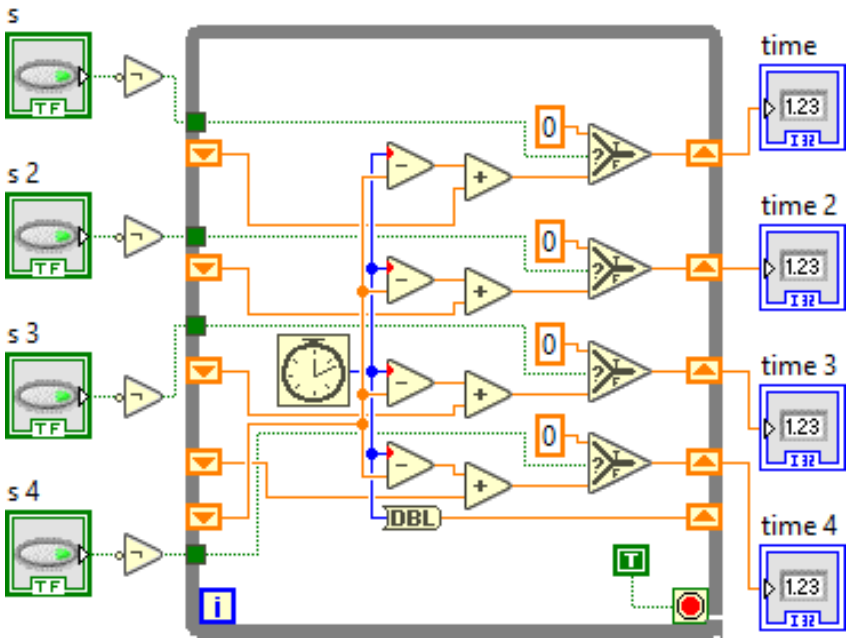


Fig 5.1 Time VI

CHAPTER-6

TIME TABLE AND CASE STRUCTURE FOR DAY SEQUENCE

6.1 TIME TABLE:

Project runs according to the time table which is defined by the User. This Project runs according to the following time table.

List of classes	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Class 1	0	1	0	1	1	0	0
Lab 1	0	0	1	0	1	1	0
Lab2	0	0	0	1	0	1	1
Class 2	0	1	1	0	0	0	1

Fig 6.1 Time table

Above Time table is set such that '1' represents that the corresponding day the lights and fans present in the corresponding class rooms or labs should be turned ON.'0' represents that the corresponding day lights and fans present in the corresponding class rooms or labs should be turned OFF. This Time table should be Synchronized with the Day Sequence Case Structure. In Order to understand this first one should know about the Working of the Case Structure and how it is used.

6.2 Case Structure:

Owning Palette: Structures

Requires: Base Development System.

Contains one or more sub diagrams, or cases, exactly one of which executes when the structure executes. The value wired to the case selector determines which case to execute.

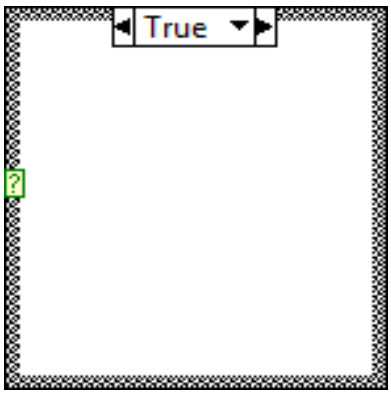


Fig 6.2 Case Structure

A Case Structure Consists of following components:

- **Selector label**—Displays the value(s) for which the associated case executes. You can specify a single value or a range of values. You also can use the selector label to specify a default case.
- **Sub diagram(case)**—Contains the code that executes when the value wired to the case selector matches the value that appears in the selector label. To modify the number or order of sub diagrams, right-click the border of the Case structure and select the appropriate option.
- **Case selector**—Selects which case to execute based on the value of the input data. The input data can be a Boolean, string, integer, enumerated type or error cluster. The data type you wire to the case selector determines the allowed cases you can enter in the selector label.

6.3 CASE STRUCTURE FOR DAY SEQUENCE:

In this Chapter we are going discuss about how the Time table is Synchronized with the Day sequence. Here we are going to have so many cases depending upon the day and Corresponding Classes and labs which have to Turned ON automatically.

There are Seven Cases as there are Seven Days in a week. Using these cases, we can explain how power minimization can be done in this project.

6.4 ALGORITHM:

1. Start the Program

2. As there are seven cases, the execution of the program starts from the default case.

3. In each case, the time delay and Boolean constants are present.

4. When we run the program, the time delay is executed first because the duration of the case depends on the Delay we set.

5. Boolean Constants should be set according to the above time table in fig 6.1

6. If the Boolean constants are set ‘true’ it means that the lights and fans are Turned ON in that class or lab. If ‘False’ then lights and fans are Turned ‘OFF’.

7. Stop the Program after getting the Expected output.

Case default: all classes loads are Turned off

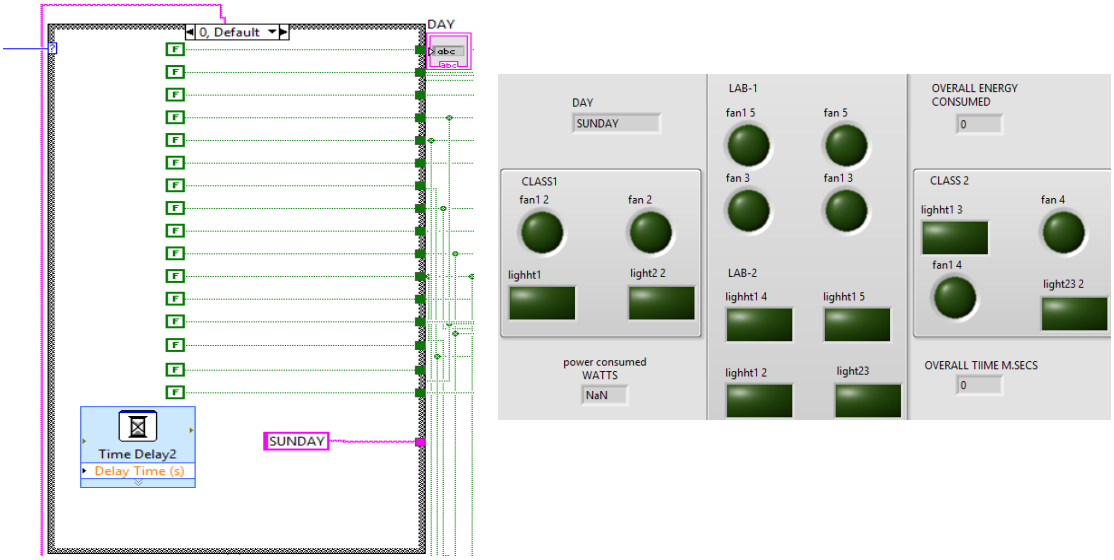


Fig 6.3 Day1 Time table synchronized with day sequence

In this case, All the lights and fans in every class and lab is Turned OFF. Because Sunday is not a working day there are no classes Scheduled on that day.

Case1: Class-1 and Class-2 Loads are turned ON

Fig 6.4 Day2 Time table synchronized with day sequence

In this case, the lights and fans in the Class-1 and Class-2 are Turned ON as scheduled in the Time table.

Case 2: Class-2 and Lab-1 loads are turned ON

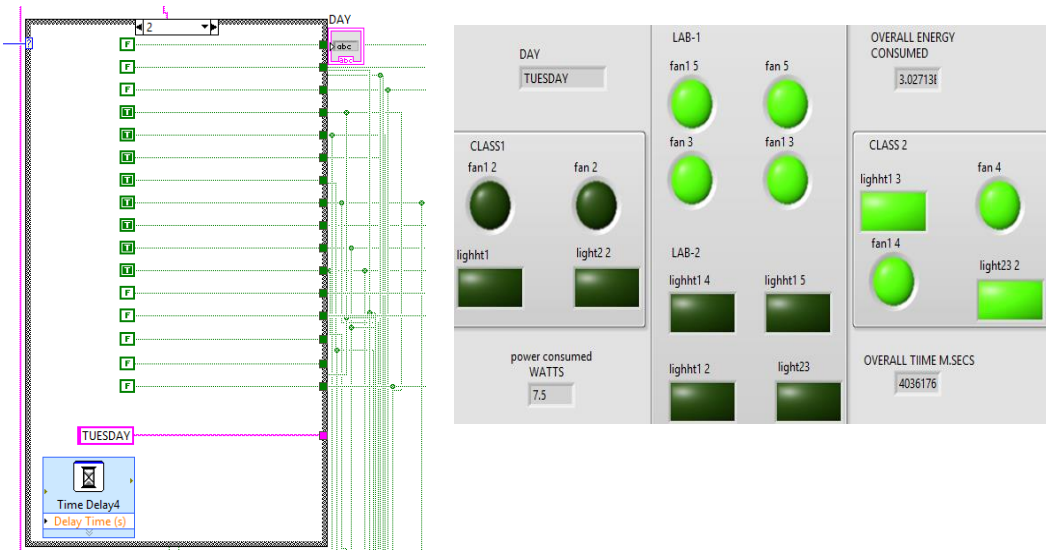


Fig 6.5 Day3 Time table synchronized with day sequence

In this case, the lights and fans in the Class -2 and lab-1 are Turned ON as scheduled in the Time table.

Case 4: Class-1 and Lab-2 Loads are turned ON

Fig 6.6 Day4 Time table synchronized with day sequence

In this case, the lights and fans in the Class-1 and lab-2 are Turned ON as scheduled in the Time table.

Case 4: Class-1 and Lab-1 Loads are turned ON

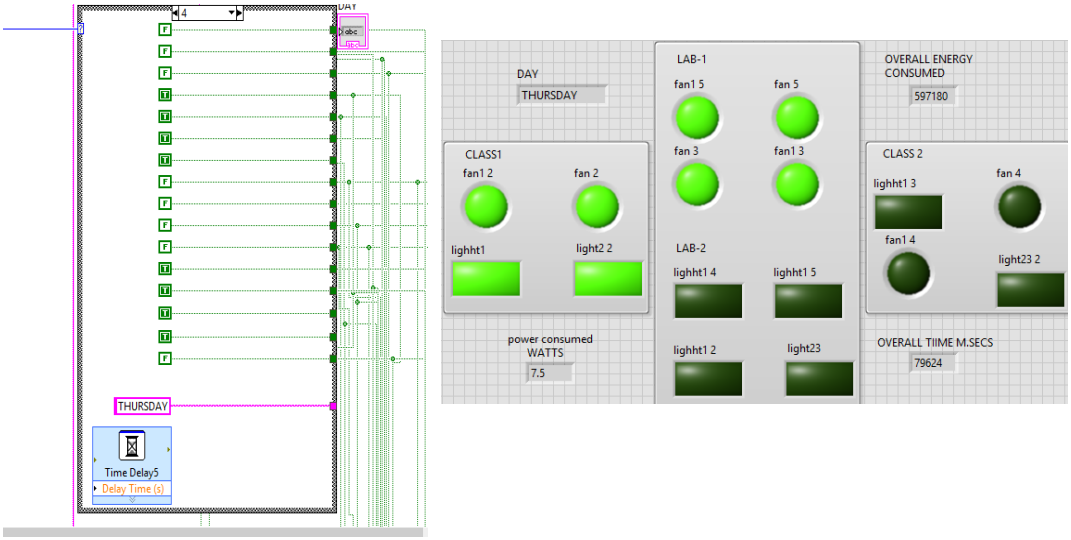


Fig 6.7 Day5 Time table synchronized with day sequence

In this case, the lights and fans in the Class -1 and lab-1 are Turned ON as scheduled in the Time table.

Case 5: Lab-1 and Lab-2 loads are turned ON

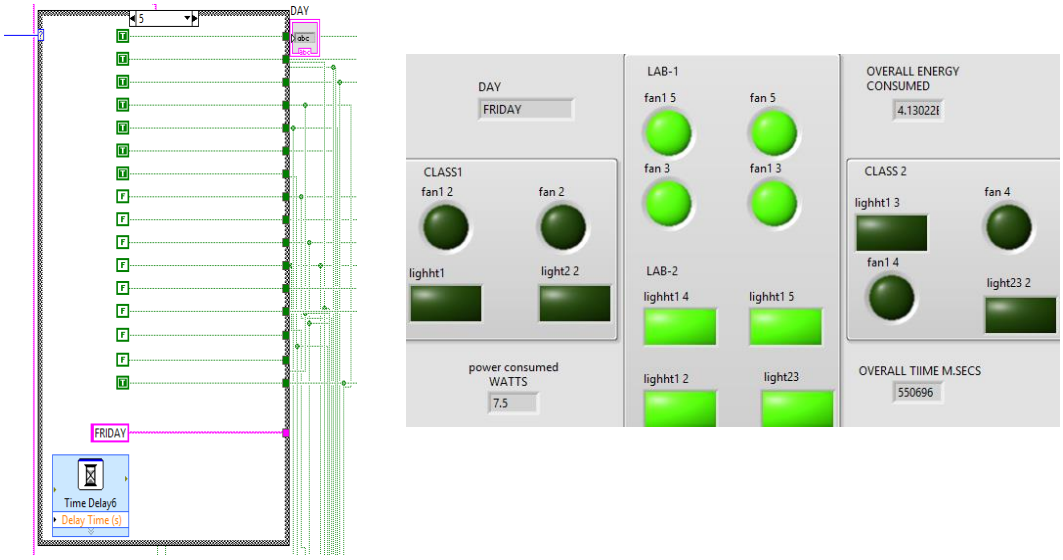


Fig 6.8 Day6 Time table synchronized with day sequence

In this case, the lights and fans in the Lab-1 and Lab-2 are Turned ON as scheduled in the Time table.

Case 6: Class-2 and Lab-2 loads are turned ON

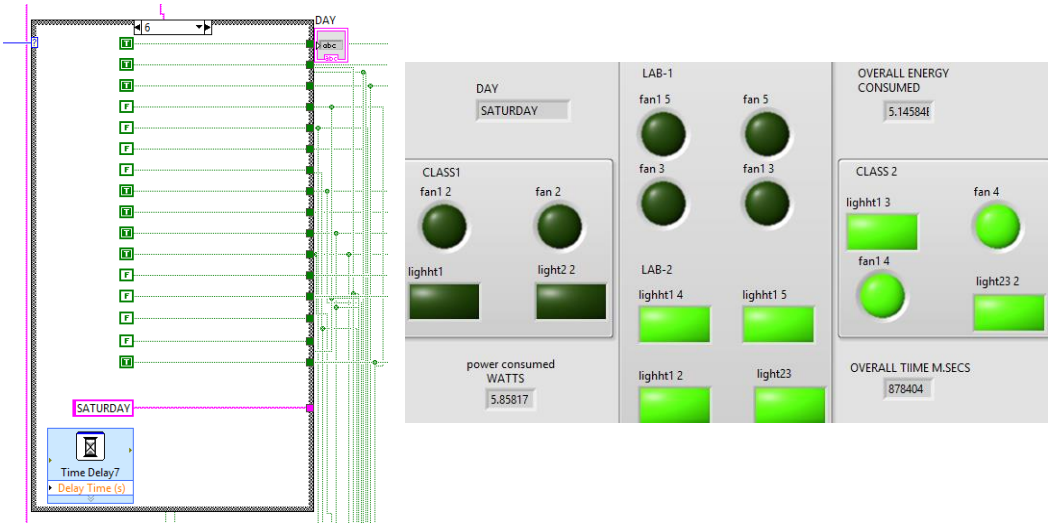


Fig 6.9 Day7 Time table synchronized with day sequence

In this case, the lights and fans in the Class-2 and Lab-2 are Turned ON as scheduled in the Time table

CHAPTER-7

POWER CALCULATIONS

7.1 ALGORITHM:

1. Start the program.
2. Turn ON the Boolean Controllers.
3. When the Boolean Controller is set '1' the condition is executed as "TRUE". If the Boolean Controller is set '0' the condition is executed as "FALSE".
4. Give the Output of the Boolean to the SELECT switch. It checks for the condition, whether Output is true or false. If true then it will return true statement. This true statement is the default ratings of Lights and Fans. This rating is multiplied by time duration or time lapse in order to get the energy consumed.

   i.e., Energy = Power * Time

5. Thus the Energy obtained by individual class or lab is indicated in a Numeric Indicator.
6. The Energy Obtained by all class rooms is added using Compound Arithmetic. This will be the Overall energy Consumed.
7. This is shown in the Numeric Indicator. The Obtained Energy Should be divided by overall time to get Over all Power consumed per day which is displayed in Numeric Indicator.
8. Stop the program.

7.2 Select Function:

Fig 7.1: Select Function

It returns the value wired to the **t** input or **f** input, depending on the value of **s**. If **s** is TRUE, this function returns the value wired to **t**. If **s** is FALSE, this function returns the value wired to **f**. The connector pane displays the default data types for this polymorphic function.

t is the value that this function returns if **s** passes a TRUE value. **t** and **f** must be of the same type, but they can have different numeric representations. **s** determines whether the function returns the value of **t** or **f** in **s? t: f.** If you wire an error cluster to **s** and an error occurs, the error cluster passes a TRUE value to the function. Otherwise, the error cluster passes a FALSE value to the function. **f** is the value that this function returns if **s** passes a FALSE value. **t** and **f** must be of the same type, but they can have different numeric representations.

7.3 Compound Arithmetic:

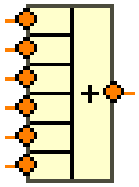


Fig 7.2: Compound Arithmetic

Performs arithmetic on one or more numeric, array, cluster, or Boolean inputs. To select the operation (Add, Multiply, AND, OR, or XOR), right-click the function and select **Change Mode** from the shortcut menu. When you select this function from the Numeric palette, the default mode is Add. When you select this function from the Boolean palette, the default mode is OR. The connector pane displays the default data types for this polymorphic function.

value 0...n-1 can be a number or Boolean value, an array of numbers or Boolean values, a cluster, array of clusters, and so on. You can wire a waveform to only one **value** input. If an input is a waveform, you can have an unlimited number of scalar inputs of varying sizes. If **value** is an error cluster, only the **status** parameter of the error cluster passes to the input terminal.

result returns the result of the selected operation applied to the **value 0...n-1**. For AND, OR, or XOR, **result** returns the bitwise operations on numeric inputs and logical operations on Boolean inputs.

7.4 CALCULATION OF POWER BLOCK DIAGRAM:

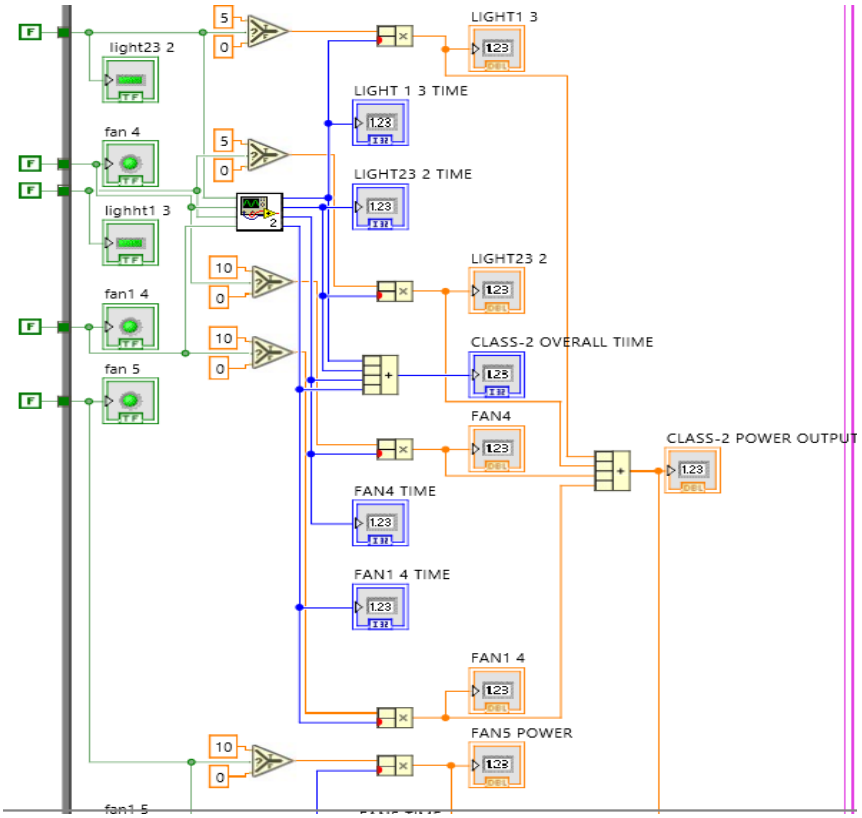


Fig 7.3 Power Calculation Block diagram

Above fig 7.3 is the block diagram for power calculations. The Output of fans and lights is given to select switch. If condition is true then the rated power is multiplied with time and total loads energy consumption is added. Thus, Power is calculated for every instant of time for every class / lab.

CHAPTER-8

STORING OBTAINED DATA THROUGH WRITE TO MEASUREMENT FILE

8.1 Filename:

Displays the full path to the file to which you want to write data. The Express VI writes data to the file that this parameter specifies only if the **Filename** input is unwired. If you wire the **Filename** input, the VI writes data to the file that this input specifies instead.

8.2 File Format:

Contains the following options:

- **Text (LVM)**—Sets the file format to text-based measurement file (.lvm) and the file extension in **Filename** to. lvm.
- **Binary (TDMS)**—Sets the file format to binary measurement file (.tdms) and the file extension in **Filename** to. tdms. If you select this option, the **Delimiter** section and the **No headers** option in the **Segment Headers** section are not available.
- **Binary with XML Header (TDM)**—(Windows) Sets the file format to binary measurement file (.tdm) and the file extension in **Filename** to. tdm. If you select this option, the **Delimiter** section and the **No headers** option in the **Segment Headers** section are not available. When you select this file format, you enable the **Lock file for faster access** checkbox. Selecting this checkbox makes reading and writing significantly faster (at the expense of the ability to multitask certain activities). It is recommended that you use this option in most cases. When this option is enabled, no two Express VIs can access the same file at the same time when one of them is writing a "series of files".

- **Microsoft Excel (.xlsx)**—Sets the file format to Microsoft Excel (.xlsx) and the file extension in **Filename** to .xlsx. If you select this option, the **Delimiter** section and the **Segment Headers** section are unavailable. This option does not require Microsoft Excel installed on the local computer.

8.3 ACTION:

Contains the following options:

- **Save to one file**—Saves all the data to one file.
- **Ask user to choose file**—Displays a dialog box that prompts users to select a file. This option is available only when you select the **Save to one file** option.
- **Ask only once**—Prompts users to select a file only once. This option is available only when you place a checkmark in the **Ask user to choose file** checkbox.
- **Ask each iteration**—Prompts users to select a file each time the Express VI runs. This option is available only when you place a checkmark in the **Ask user to choose file** checkbox.
- **Save to series of files (multiple files)**—Saves the data to multiple files. If **Reset** is TRUE, the VI starts at the first file in the series. For example, if test_001.lvm has been saved to test_004.lvm, test_001.lvm might be renamed, overwritten, or skipped based on the value of the **Existing Files** option in the Configure Multi-file Settings dialog box.
- **Settings**—Displays the Configure Multi-file Settings dialog box. This option is available only when you select the **Save to series of files (multiple files)** option.

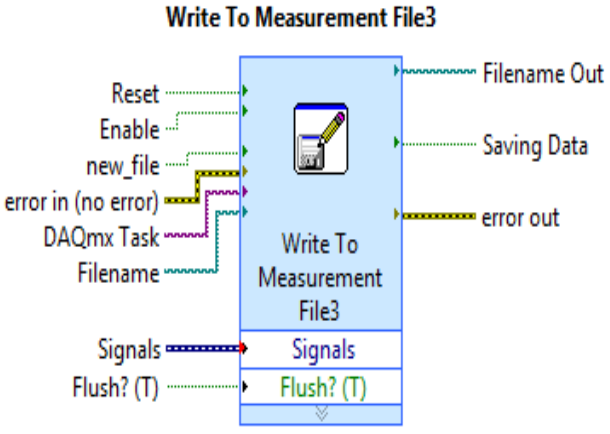


Fig 8.1 write to measurement file

8.4 ALGORITHM:

1. Start.

2. Calculate the power of the classes and labs

3. The calculated output power value is sent to the input of the write to measurement file through signals

4. In the write to measurement file choose the path to store the file in excel format and save it to the series of multiple files option.

5. Two columns of data are stored in excel sheet

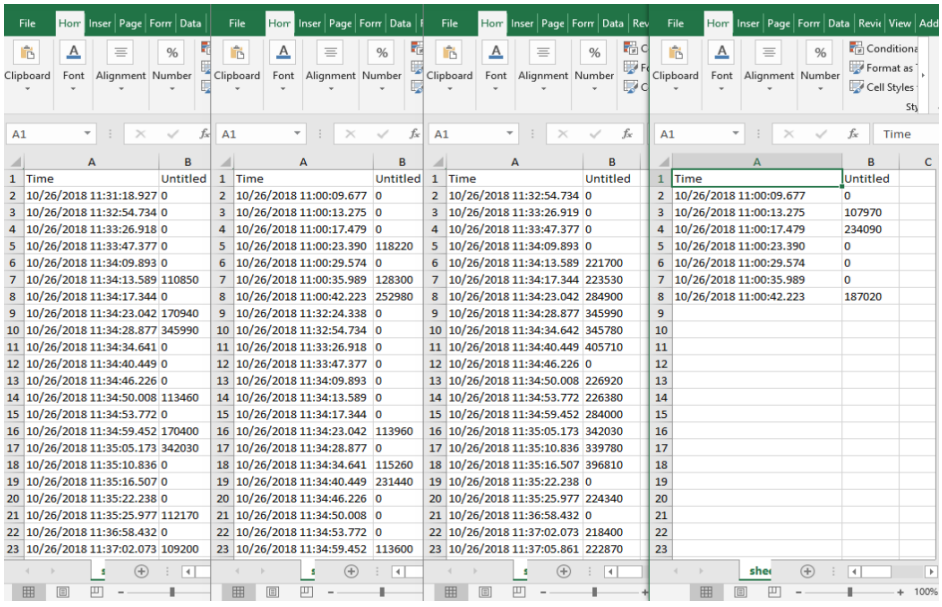
   (i) accurate time value

   (ii) calculated power value

6. If the data of 'n' values is to be stored in locations multiple files must be selected

7. End.

8.5 Data Stored in Excel Sheet:



	A	B
1	Time	Untitled
2	10/26/2018 11:31:18.927	0
3	10/26/2018 11:32:54.734	0
4	10/26/2018 11:33:26.918	0
5	10/26/2018 11:33:47.377	0
6	10/26/2018 11:34:09.893	0
7	10/26/2018 11:34:13.589	110850
8	10/26/2018 11:34:17.344	0
9	10/26/2018 11:34:23.042	170940
10	10/26/2018 11:34:28.877	345990
11	10/26/2018 11:34:34.641	0
12	10/26/2018 11:34:40.449	0
13	10/26/2018 11:34:46.226	0
14	10/26/2018 11:34:50.008	113460
15	10/26/2018 11:34:53.772	0
16	10/26/2018 11:34:59.452	170400
17	10/26/2018 11:35:05.173	342030
18	10/26/2018 11:35:10.836	0
19	10/26/2018 11:35:16.507	0
20	10/26/2018 11:35:22.238	0
21	10/26/2018 11:35:25.977	112170
22	10/26/2018 11:36:58.432	0
23	10/26/2018 11:37:02.073	109200

	A	B
1	Time	Untitled
2	10/26/2018 11:00:09.677	0
3	10/26/2018 11:00:13.275	0
4	10/26/2018 11:00:17.479	0
5	10/26/2018 11:00:23.390	118220
6	10/26/2018 11:00:29.574	0
7	10/26/2018 11:00:35.989	128300
8	10/26/2018 11:00:42.223	252980
9	10/26/2018 11:32:24.338	0
10	10/26/2018 11:32:54.734	0
11	10/26/2018 11:33:26.918	0
12	10/26/2018 11:33:47.377	0
13	10/26/2018 11:34:09.893	0
14	10/26/2018 11:34:13.589	0
15	10/26/2018 11:34:17.344	0
16	10/26/2018 11:34:23.042	113960
17	10/26/2018 11:34:28.877	0
18	10/26/2018 11:34:34.641	115260
19	10/26/2018 11:34:40.449	231440
20	10/26/2018 11:34:46.226	0
21	10/26/2018 11:34:50.008	0
22	10/26/2018 11:34:53.772	0
23	10/26/2018 11:34:59.452	113600

	A	B
1	Time	Untitled
2	10/26/2018 11:32:54.734	0
3	10/26/2018 11:33:26.919	0
4	10/26/2018 11:33:47.377	0
5	10/26/2018 11:34:09.893	0
6	10/26/2018 11:34:13.589	221700
7	10/26/2018 11:34:17.344	223530
8	10/26/2018 11:34:23.042	284900
9	10/26/2018 11:34:28.877	345990
10	10/26/2018 11:34:34.642	345780
11	10/26/2018 11:34:40.449	405710
12	10/26/2018 11:34:46.226	0
13	10/26/2018 11:34:50.008	226920
14	10/26/2018 11:34:53.772	226380
15	10/26/2018 11:34:59.452	284000
16	10/26/2018 11:35:05.173	342030
17	10/26/2018 11:35:10.836	339780
18	10/26/2018 11:35:16.507	396810
19	10/26/2018 11:35:22.238	0
20	10/26/2018 11:35:25.977	224340
21	10/26/2018 11:36:58.432	0
22	10/26/2018 11:37:02.073	218400
23	10/26/2018 11:37:05.861	222870

	A	B	C
1	Time	Untitled	
2	10/26/2018 11:00:09.677	0	
3	10/26/2018 11:00:13.275	107970	
4	10/26/2018 11:00:17.479	234090	
5	10/26/2018 11:00:23.390	0	
6	10/26/2018 11:00:29.574	0	
7	10/26/2018 11:00:35.989	0	
8	10/26/2018 11:00:42.223	187020	
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			

Fig 8.2 Data stored

CHAPTER-9

OVER POWER USAGE INDICATORS AND ALERTS

9.1 ALGORITHM:

1. Start

2. A select switch is attached to the values of total power of the individual classes and overall power of the program

3. Select switch with two strings one connected to the true statement and other to the false statement.

4. For individual classes power output and over all power output

   i. A limit is set according to the no. of loads connected through greater than comparator.

   ii. If obtained value is greater than the given input value LED glows showing the warning from the true statement.

5. END

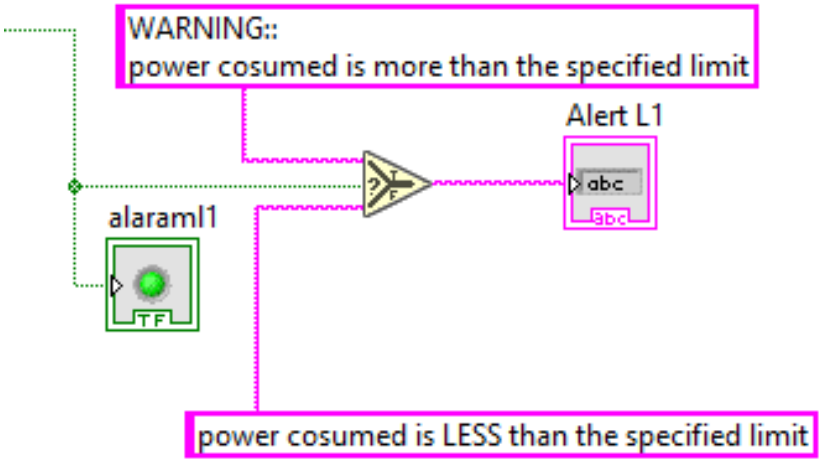


Fig 9.1 Block diagram for alerts and alarms

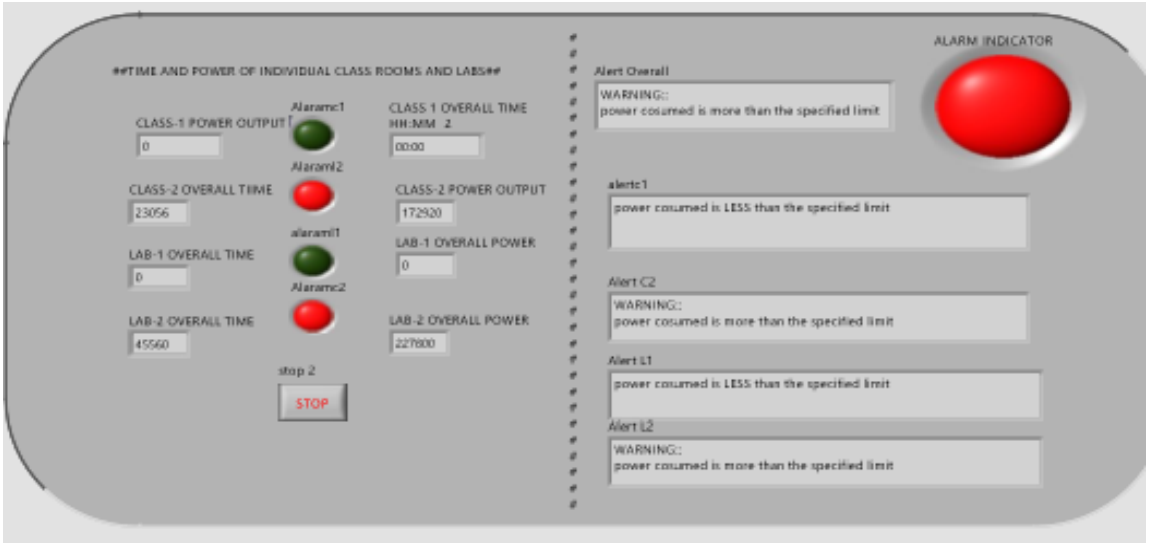


Fig 9.2: Front panel to show alert messages and alarms

In the above fig in the front panel red light indicates that the power consumed is greater the specified limit. Then an Alert message is displayed that “power consumed is more than the specified limit.

CHAPTER-10

CONCLUSION AND FUTURE SCOPE

Hence by using this method the power consumption is reduced in a Smart way. If this program is applied in real time then about 25-30% of extra power consumed is reduced. This is one of the best ways to conserve the electricity in the industries, institutions and at big infrastructures for commercial use.

Here in this Program, the minimization of power theme is justified by optimizing the electrical loads and syncing them with the time table. This makes the electricity to flow through the loads only when required. As the system is automatically responds with the values set in time table, one can easily modify the load in any section of the room at any time if required. The Power wastage at every commercial loads and institutions loads can be easily reduced by applying by

i. System inputs at regular intervals of time.

ii. By setting over usage limit.

Therefore, over usage of loads at any commercial or institutions can be easily known and can be optimized by this project.

REFERENCES

1.http://www.ni.com/en-in.html

2.http://search.ni.com/nisearch/app/main/p/bot/no/ap/global/lang/en/pg/1/q/dbl%20converter/

3. http://www.ni.com/getting-started/labview-basics/execution-structures

4. http://zone.ni.com/reference/en-XX/help/371361L-01/glang/

5.https://en.wikipedia.org/wiki/Energy_conservation#/media/File:USEnFlow02-quads.gif