Programmable Variable Load Module for UP Power System Simulation Laboratory

Undergraduate Project

by

Denisse DC. Robles
2010-16881
B.S. Electrical Engineering

Eldion Vincent H. Bartolo
2010-28167
B.S. Electrical Engineering

Adviser:

Wilbert Rey D. Tarnate
.

University of the Philippines, Diliman
June 2015

Abstract

Programmable Variable Load Module for UP Power System Simulation Laboratory

This paper presents the design and implementation of a Variable Load Module that can simulate load profiles and unbalanced loads. Moreover, the proposed load module can be varied in continuous manner unlike the existing loads in PSSL. The proposed load module which consists of lighting, inductive, and capacitive loads will have a programmable-variable apparent power consumption through the use of PWM control; PWM control will be used over the conventional phase-angle control of AC voltage to reduce or eliminate the total harmonic distortion introduced to the system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Simulation is the imitation of the operation of a real-world process or system over time[14]. The ability to simulate any real system gives planners a clear advantage in doing their job. With the use of simulators, planners will able to see how their system works and through that provide them necessary information useful in their operations. It is also in a simulation that probable solutions can be tested if it is viable to address a particular problem. Additionally, planners can predict the future performance of a system and allow them prepare for it.

Rapid growth of the population and economy results in a more complex and larger power system network. From this point of view, three-phase instantaneous-value based real-time power system simulators(in particular digital type simulators) are becoming useful and effective tools. In general, however, they are tend to very ideal and can't model the real phenomena occuring in live power systems. For such reasons, making a harware implementation of power system simulator becomes more important. In line with this, the goal of our project is to design a variable load module that can model the unbalanced and changing loads of the distribution system. It is necessary for the learning, training and appreciation of the students and other trainees about power systems.

1.1 Overview

Distribution system is the system of wires and associated facilities that delivers energy from the transmission system to end-users in a franchise area [6]. The end-users consume electric power and convert it to useable form though the electrical loads. It is composed of a variety of electrical equipments and appliances, as heating, cooking, lighting and air conditioner[11].

In modeling the distribution system, the diversity of load and its time varying characteristics is a major problem. The demand on electricity is not constant throughout the day. It

varies depending on the lifestyle of the consumers. A load profile shows this changing electrical load or demand over time. This information on the demand is very useful in determining the amount of electricity that should be generated. Furthermore, if there is frequent load variation in the distribution system then network voltage drop and light flicker can be observed.

In a three phase system, electrical loads can either be classified as balanced or unbalanced load. A balanced three phase load means the load impedances in all three phases are identical[15]. Unbalanced load on the other hand means that the load impedances are not equal and can be decomposed into the positive, negative or zero sequence. In reality, only unbalanced load is used to model distribution system. Unbalanced loading directly affects the equipment of costumers. For a three-phase induction motor; negative sequence current raises the motor's temperature, reduces the motor's torque, lowers the motor's full-load speed, and increase the motor's noise and vibration. Furthermore, unbalanced loading can increase the system loss, increase the ampacity of conductors, and increase harmonics [9].

Because of the necessity of modeling the wide-range power consumption of a distribution system, the effects of unbalanced loads, and the time varying electrical loads; our group proposes a continuous-valued, unbalanced loading and load profiling capable- load simulator module. The load simulator which will be integrated with the existing power system simulators in PSSL can be used to study the behaviour of the real distribution system, observe the effects of real distribution system on the generation and transmission system, and help the operator make appropriate decisions in handling distribution systems.

Chapter 2

Related Work

The TERCO PST 2240 Load Module shown in Fig.2.1 is one of the commercial load modules for power system simulation in the market. The characteristics and abilities of the load module are described as follows.

- Consists of 3-phase loads R, L and C controllable in 13 steps each, manually or by SCADA
- Can cover 0–150 % of nominal power
- Create single phase loads as well as other non-symmetrical loads
- Can be integrated with several motors to study the dynamics of the system as well as the mechanical load sharings between two or more generators.

Due to the cost and limitations of the commercial module; a low cost, continuous valued, load profile-capable, and unbalance operation-capable load simulator is proposed in our project.

Figure 2.1: Terco PST 2240 Load Module

2.1 Realizations on creating single phase variable loads

There are already papers that account for realizing single phase variable load simulators. This section presents the notable features of different realized load simulators.

2.1.1 Variable Impedance Loads

Knowing the phasor equivalent of ohm's law:

$$Z(\jmath\omega) = \frac{V_{rms}\angle\alpha}{I_{rms}\angle\beta} = \frac{V_{rms}}{I_{rms}}\angle\alpha - \beta \text{ or } \frac{V_{rms}}{I_{rms}}\angle\theta \tag{2.1}$$

we can adjust the impedance of a load by varying the ratio of $V_{rms}$to I_{rms}or adjust the phase difference θ between them.

This technique was implemented in the works of Guan-Chyun Hsieh, et al.(1993) in designing and implementing an AC Active Load Simulator(AC-ALS). Fig. 2.2 shows the building blocks of the AC-ALS , while Table 2.1 defines the function of each blocks.

The operation of the AC-ALS is described as follows.

From the V-sensor, the period of the source voltage is measured by the ZCD while a sine

table of the ac source is tabulated by the A/D converter. These characteristics of the ac source are then sent to SCU - which sends a desired phase shift to the Delay Circuit. After getting the phase-shift the Delay Circuit sends the period of the desired current waveform to SCU [12].

The location of the accumulated current waveform in the SCU is then stored as an address in CWAMB. With a proper timing signal and the address from CWAMB, a current waveform with desired charateristics is created through the use CWG and D/A converter. This current waveform will now drive the ALD that models the programmed impedance[12].

To maintain the characteristics of the current, the current from I-sensor is digitized by the A/D converter. This digitized current waveform together with the digitized voltage source will be used by the SCU to correct the timing signals needed in making the desired phase shift[12].

Though the works of Guan-Chyun Hsieh can simulate different impedance load, their implementation has some drawbacks. First, it can handle 11VA only from the mains. Second, having three of these AC-ALSs so that unbalanced loading can be simulated, will be very complex and costly to implement. Third, The use of MOSFETs as a load will require very large heat sinks.

In our proposed project, we will also the use of micro-controllers as the major control unit, and use A/D converters extensively for measuring and feedback purposes.

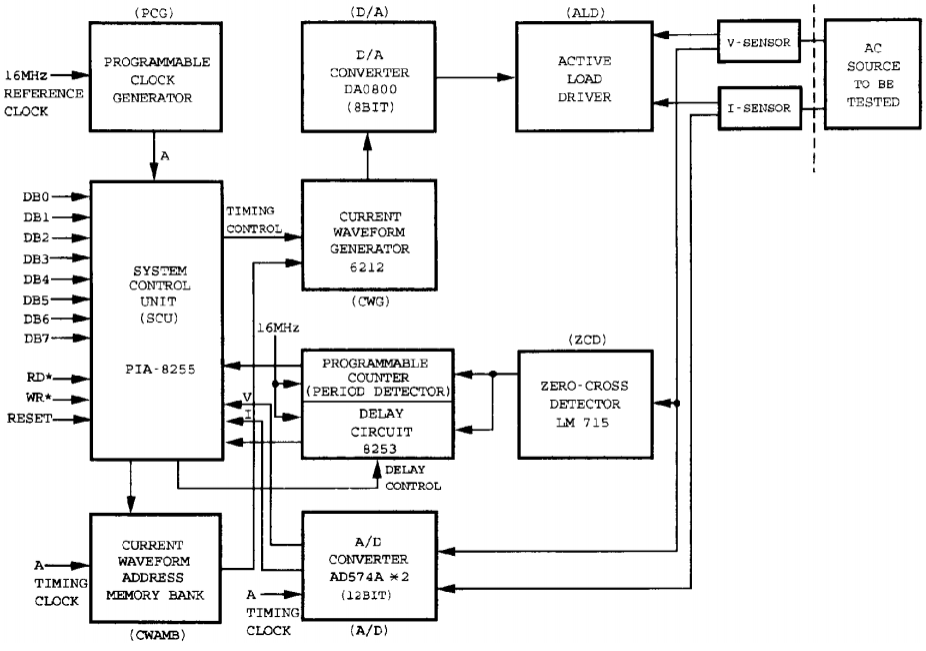

Figure 2.2: Block Diagram of the AC Active Load Simulator

Components	Function
SCU(System Control Unit)	continuously monitor frequency and amplitude of the ac source, program the current waveform
A/D(Analog to Digital) Converter	convert current and voltage inputs to digital output
ZCD(Zero Crossing Detector)	gets the frequency of the voltage source
Delay Circuit(Programmable Counter)	delay the phase of a reference period by counting a 16 MHz clock
D/A (Digital to Analog) Converter	convert the digital output of the current waveform generator to analog signals that drives the CWG
CWG(Current Waveform Generator)	achieves a set digital current waveforms through the timing signals of SCU and current waveform addresses form CWAMB
CWAMB(Current Waveform Address Memory Bank)	a RAM cell, memorizes the current waveform address from SCU
PCG(Programmable Clock Generator)	provides the reference frequency for the system as a time base, a programmable frequency divider that ca range from 31.25 kHz through 8MHz.
(ALD) Active Load Driver	a push-pull topology of MOSFETS used as the load sink

Table 2.1: Basic Components of the AC Active Load Simulator

2.1.2 AC Voltage Controllers

AC Voltage Controller as its name suggests, control the amount of AC voltage applied to a load. If the load is a constant impedance load then the amount of apparent power delivered to the load will be dependent on applied rms voltage. When ac voltage controller switches are designed to have low conduction loss, the input apparent power of the power supply will be approximately equal to the output apparent power of the load impedance.

The conventional AC Voltage controller using the method of phase-angle control is shown in Fig.2.3. Phase angle control changes the V_{rms} of a load by delaying the application of power supply at a certain firing angle. Hence phase-angle ac voltage controllers remove some of the source waveforms [8].The voltage waveforms for a resistive and resistive-inductive loads using phase-angle control are shown in Figs.2.4a and 2.4b respectively.

Since the rms output voltage for a resistive load can be expressed as:

$$V_{o,rms} = \sqrt{\frac{1}{\pi}[\int_{\alpha}^{\pi} V_m \sin(\omega t)]^2 d\omega t} = \frac{V_m}{\sqrt{2}}\sqrt{1 - \frac{\alpha}{\pi} + \frac{\sin(2\alpha)}{2\pi}} \tag{2.2}$$

While the rms output current of the resistive-inductive load can be expressed as:

$$I_{o,rms} = \sqrt{\frac{1}{\pi}\int_{\alpha}^{\beta} i_o^2(\omega t) d(\omega t)} \tag{2.3}$$

$$\text{where} \begin{cases} i_o(\omega t) = \begin{cases} \frac{V_m}{Z}[\sin(\omega t - \theta) - \sin(\alpha - \theta)e^{\frac{(\alpha - \omega t)}{\omega \tau}}] & \text{for } \alpha \le \omega t \le \beta \\ 0 & \text{otherwise} \end{cases} \\ Z = \sqrt{R^2 + (\omega L)^2} \text{ and } \theta = \tan^{-1}\left(\frac{\omega L}{R}\right) \\ i_o(\beta) = 0 = \frac{V_m}{Z}[\sin(\beta - \theta) - \sin(\alpha - \theta)\, e^{\frac{(\alpha - \beta)}{\omega \tau}}] \text{ ...where } \beta \text{ is the extinction angle} \end{cases}$$

The the apparent power of the load ($VA = I_{o,rms}^2 \bullet Z = \frac{V_{o,rms}^2}{Z}$) can be adjusted by changing the values of firing angle α.

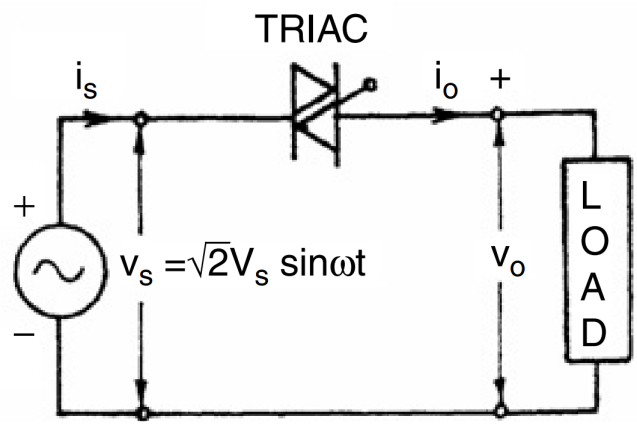

Figure 2.3: Basic phase-angle controlled AC voltage controller

taken from M. Rashid ;"Power Electronics Handbook: Devices, Circuits, and Applications" 2nd ed.

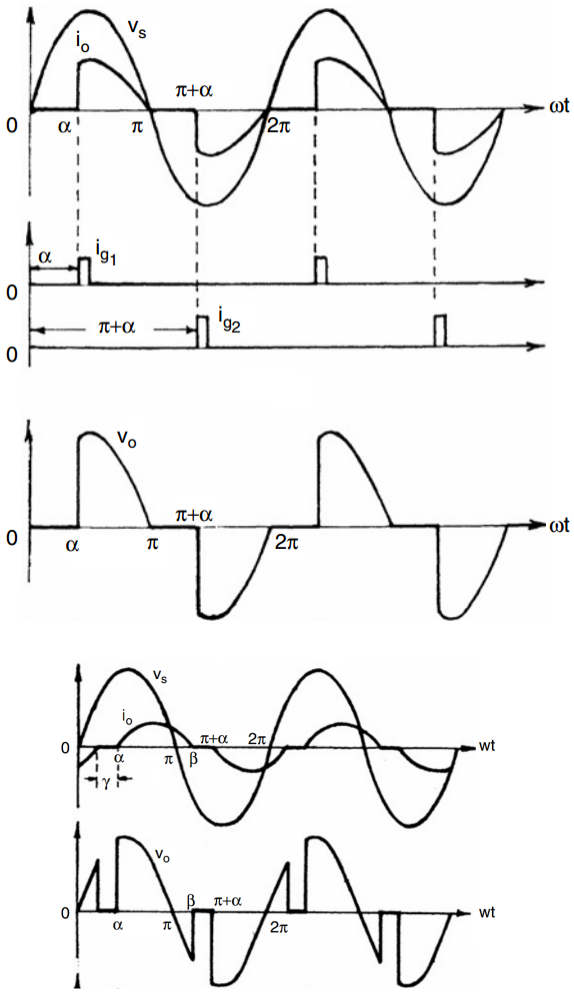

Figure 2.4: Waveforms of (a) resistive load and (b) resistive-inductive loads in response to phase-controlled AC Voltage Controllers

taken from M. Rashid "Power Electronics Handbook: Devices, Circuits and Applications" 2nd ed.

There are many applications of ac voltage controllers. Some of these which are related to our project are discussed in the following subsections.

2.1.2.1 Light Dimmers

There are two main types of light dimmers in the market. First is the forward phase controlled dimmers or the conventional light dimmer shown in Fig.2.5. These dimmers produce audible noises in lamps due to the sharp turn-on waveform of current. So a large bulky filter-inductor is needed to be placed in series with the load to reduce the rate of change of current through the lamp.[3].

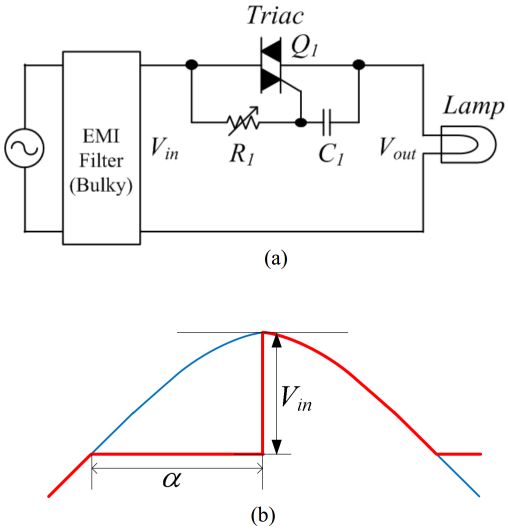

Figure 2.5: Typical phase-controlled dimmer. (a) circuit, (b) typical waveforms

taken from Jong-Hyun Kim, et al.."A simple dimmer using a MOSFET for AC driven lamp".IEEE conference

The second type of light dimmer is the reverse-phase controlled light dimmer shown in Fig.2.6. In the reverse phase-controlled dimmers, the triac is triggered into conduction immediately after the zero crossing of line voltage and then commutated off at some point during the ac half-cycle. When the reverse phase light dimmer is used, the bulky filter-inductor in the forward phase controlled dimmer is replaced by a less expensive capacitor in parallel with the power switches and the rate of change of current through the load after tum-off will be reduced [3].

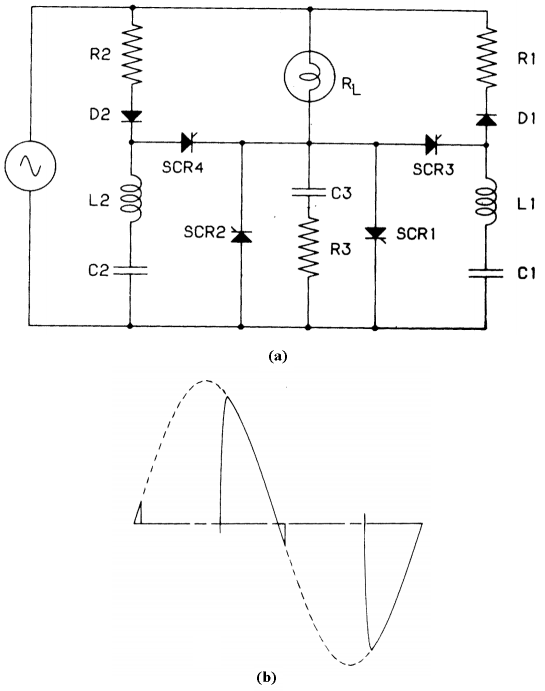

Figure 2.6: Reverse-Phase Cotrolled Dimmer. (a)Basic Circuit (b) Voltage Waveform

taken from R.Burkhart, D.Ostrodk; "A Reverse Phase-Controlled Dimmer for Incandescent Lighting";IEEE articles.

2.1.2.2 Static VAR Compensators

The circuit shown in Fig.2.7 is called a static var compensator using thyristors. Static Var Compensators are used to maintain a unity power factor for a connected load in the power system. Static Var compensators inject a specific amount of reactive power to meet the var requirements of lagging loads.

Reactive power compensation is achieved by by installing a fixed capacitance that supplies a fixed amount of reactive power, generally greater than required by the load. The amount of injected reactive power is limited by the parallel inductance which absorbs a variable amount of reactive power, depending on the delay angle of the SCRs [8].

Most SCRs are placed in the inductor branch rather than in the capacitor branch because very high currents could result from switching a capacitor with an SCR .[8].

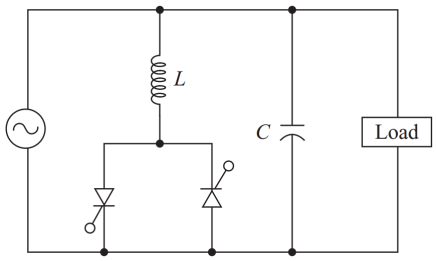

Figure 2.7: Static Reactive Power Compensator Circuit

taken from D. Hart; "Power Electronics"; 2011

2.2 Pulse Width Modulated AC Voltage Controllers

Though conventional phase-angle voltage controllers are simple to implement and use, they have many drawbacks. Some of these are low power factor, high content of low order harmonics in the supply and load side, and discontinuity of current in the load [2, 1]. To overcome some of these problems PWM AC Voltage controllers are used. The basic schematic diagram of PWM AC voltage controllers is shown in Fig. 2.8.

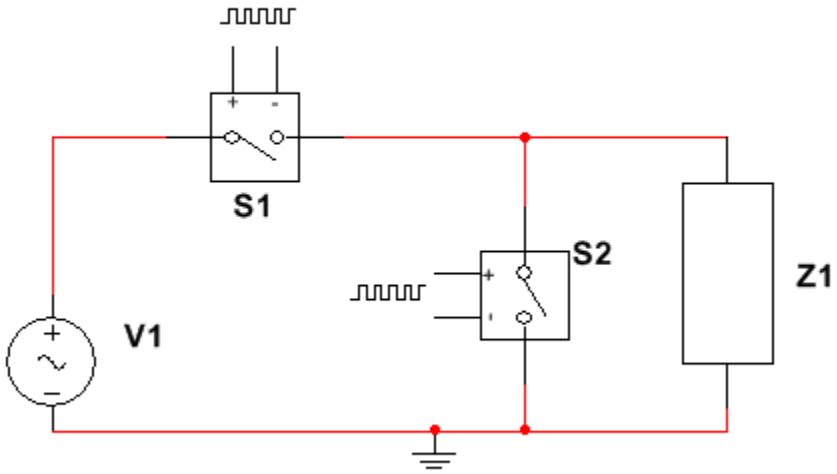

Figure 2.8: Basic PWM AC Voltage Controllers

It is composed of two bidirectional switches which are in series and parallel with the load. Bidirectional switches are switches that are capable of conducting currents and blocking voltages

in both polarities of a supply[16].

Unlike phase-angle control, the PWM control of AC voltage divides the sine wave into segments, thus the pwm control don't cause too much distortion in the power supply.

The rms voltage of the load is defined by the duty cycle of the switch S1 shown in Fig.2.8. On the other hand the function of switch S2 is to provide a free-wheeling path for an inductive load so that the inductor can release its stored energy and have continuous current whenever switch S1 is off [4].

The typical output voltage of the load with PWM control is shown in Fig. 2.9

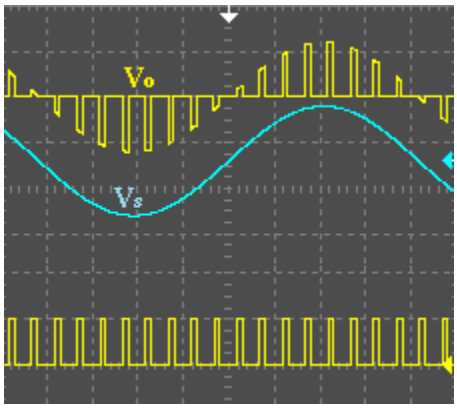

Figure 2.9: PWM control technique waveforms

2.2.1 Bidirectional Switches Realization

Fig. 2.10 shows the several topologies for realizing bidirectional switches using MOSFETS, IGBTs, etc.

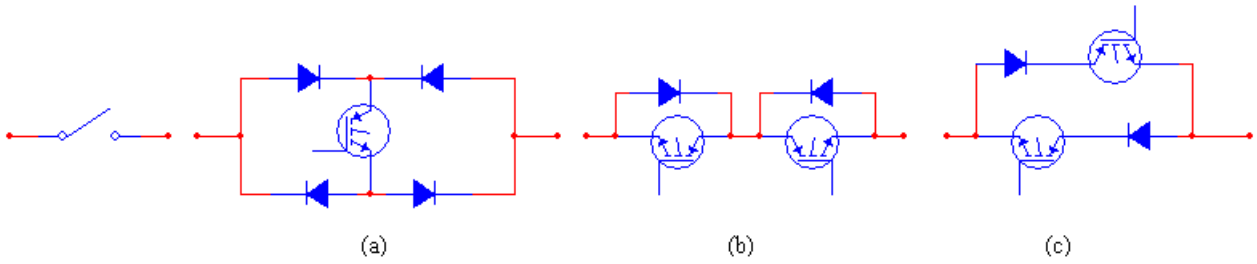

Figure 2.10: Bidirectional Switches (a) Diode Bridge, (b) anti- series active switches, (c) anti-parallel active switches

Fig.2.10.a shows a bidirectional switch that consists of one gated switch and four diodes. Though this topology is simple to implement because of having only one active device that needs

a PWM signal; it has disadvantages such as high conduction loss and unsafe commutation that causes over currents and over-voltages [5, 10].

The anti-series topology shown in fig.2.10.b shows a common collector configuration, but it can also be implemented in the common emitter mode. On the other hand, the topology shown in fig.2.10.c can can also be implemented without diodes by using NPT-IGBTs with reverse blocking capability. Both of these topologies are more advantageous over the the diode bridge because of low conduction loss and safe commutation.

2.2.2 PWM AC Voltage Controller as light dimmer

Forward and reverse phase-controlled dimmers chop the ac sine wave; they cause deformation in the output waveforms and discontinuity in the flow of current in the distribution system. Hence, they also introduce undesirable harmonics in the power system. To reduce the harmonics in phase cutting light dimmers, Jong-Hyun Kim, et al.(2011) made a simple dimmer using mosfets. Their PWM dimmer consists of only two active switches and an Electromagnetic Interference filter [3]. The schematic diagram and typical waveforms of their light dimmer is shown in Fig.2.11.

Knowing that$V_{out} = V_{in} * D$ or $V_{out} = V_{in} * \frac{t_{on}}{T}$ for pwm ac voltage controllers, light dimming is implemented by varying the duty cycle of the pwm signal. Jong-Hyun Kim's experimental results show that the THD(Total Harmonic Disorder) of their designed light dimmer is only 3.4%, under an input voltage of 220V and output voltage of 195V; unlike conventional light dimmer with 73.33% THD[3].

In our proposed project we will also implement light dimming using the anti-series topology of mosfets. This is because of the topology's simplicity(consists of one gate signal only) and low harmonic content.

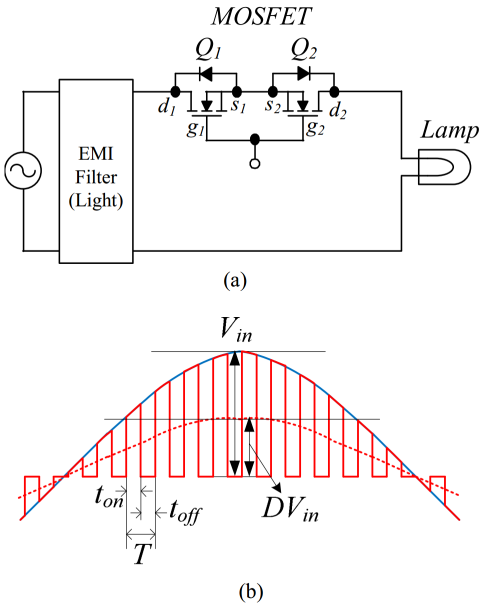

Figure 2.11: A simple dimmer for using MOSFET for AC driven lamp. (a) circuit, (b) voltage waveform

2.2.3 PWM AC Voltage Controllers as Static Var Compensators

Like in light dimming, phase-angle control is also used extensively in Static Var compensators. Which means high content of low order harmonics are introduced in the sytem. To reduce the harmonics, PWM control was also used in designing Static Var Compensators [13].

To be able to control the inductive load in the static var compensators some notable works on the pwm controlled ac voltage choppers dealing with inductive load are discussed as follows.

N.A.Ahmed, et al.(2000) as shown in fig.2.12 used pwm ac chopper to control a single phase induction machine. The authors considered the necessary three modes of pwm ac choppers to properly control inductive loads. The modes are (1) Active mode- that corresponds to the on-state of switch S1 or S2; (2) Free-wheeling mode- that corresponds to discharge of inductive current though switch S3 or S4; and (3) Dead time mode - that correponds to turning on of a pair of switches(either S1 and S3 or S2 and S4) during positive and negative half cycle, which is necessary to avoid voltage and current spikes. From these modes, the necessary gating signals for each switches are known [1].

Deniz Yildirim et al.(2008) as shown in fig.2.13, used a damped input filter to eliminate input harmonic distortions, and pulse transformer to isolate the driving circuit of pwm ac chopper used in controlling an induction motor. Though the works of Deniz Yildirim is simple to implement,

the duty cycle is limited from 0.1 to 0.9 [7].

V.K.Mehta, et al.(2013) like in [1] uses four IGBT switches while having an output filter, and driving circuit isolation in controlling an RL load. The implemented ac chopper can have duty cycle from 0 to 1[4].

In our proposed project we will implement the same method used by Mehta because of its capability to have a duty cycle of 0 to 1.

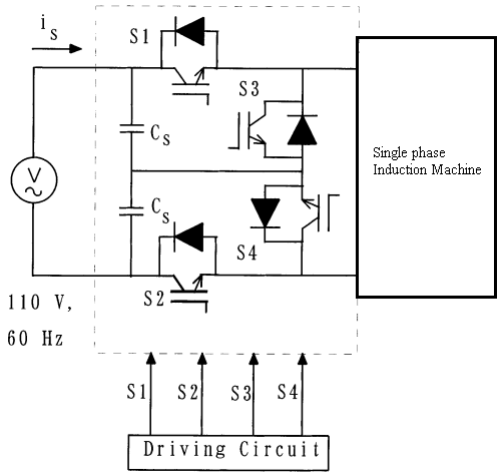

Figure 2.12: Schematic Diagram of [1]

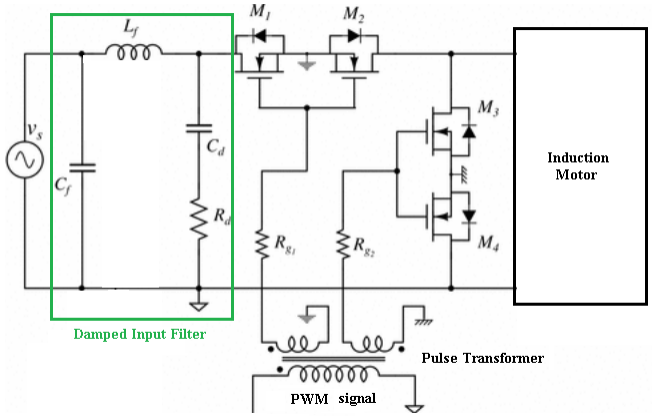

Figure 2.13: Schematic Diagram of [7]

Qualities	Works							
	AC Active Load Simulator	Light Dimmer (phase-angle)	Static Var Compensator (phase-angle)	Light Dimmer (PWM)	PWMAC Chopper for Single IM (2000)	PWMAC Chopper for Single IM (2008)	PWMAC Chopper for RL Load (2013)	Proposed Simulator
Consumes Real Load	✓	✓		✓				✓
Consumes Reactive Load	✓		✓		✓	✓	✓	✓
Low THD	✓			✓	✓	✓	✓	✓
Feedback	✓							✓
Measurement	✓							✓
Isolation of control signals	✓					✓	✓	✓
Duty cycle of PWM is from 0 to 1		✓	✓	✓	✓		✓	✓
High Power		✓	✓	✓	✓	✓	✓	✓
Applicable for 3-phase loads		✓	✓	✓	✓	✓	✓	✓
Simple		✓✓	✓✓	✓	✓	✓	✓	✓

Table 2.2: Summary of RRW

Chapter 3

Problem Statement and Objectives

3.1 Problem Statement

Previous load modules are not capable of simulating load profiles and unbalanced three phase systems. Thus we have the following objectives.

3.2 Objectives

The Objectives of this project is to create a load module that is:

1. Able to adjust the real and reactive power absorbed by the load in a continuous manner
2. Able to simulate a time varying load (load profile)
3. Able to simulate balanced and unbalanced load
4. Able to display the real and reactive power, voltage, frequency and power factor

The motivation for these objectives is to increase the similarities between a real distribution system and the load module for PSSL.

Chapter 4

Methodology

The block diagram of the proposed load simulator module is shown in Fig. 4.1. The functions of the blocks are discussed below:

- LD(Light Dimmer) - will adjust the amount of real power absorbed by the incandescent lamps
- SVC(Static Var Controller)- will adjust the amount of reactive power absorbed or injected by the load module.
- CU(Control Unit) - will set the apropriate operations of the switching controllers connected to the LD and SVC. Likewise the operation of the CU will be based on the timing diagram,real, and reactive power set by the user. The control unit will be composed of micro-controllers that have many PWM ouput pins.
- PM(Power Meter)- will measure the voltage,current,phase difference, real and reactive power seen at the load side. The power meter will also provide a feedback for the CU to adjust the correct real and reactive power absorbed by the load module.
- I/O(Input-Output units) - will display and record the values of the power, voltage, and power factor of the load module; and will accept the necessary inputs from user.

The load simulator can be accomplished by the following activities: (1) Designing the light dimmer, (2) Designing the Static Var Controller, (3) Designing the power meter, and (4) Designing the input and output units of the load simulator module.

Figure 4.2 shows our project flowchart.

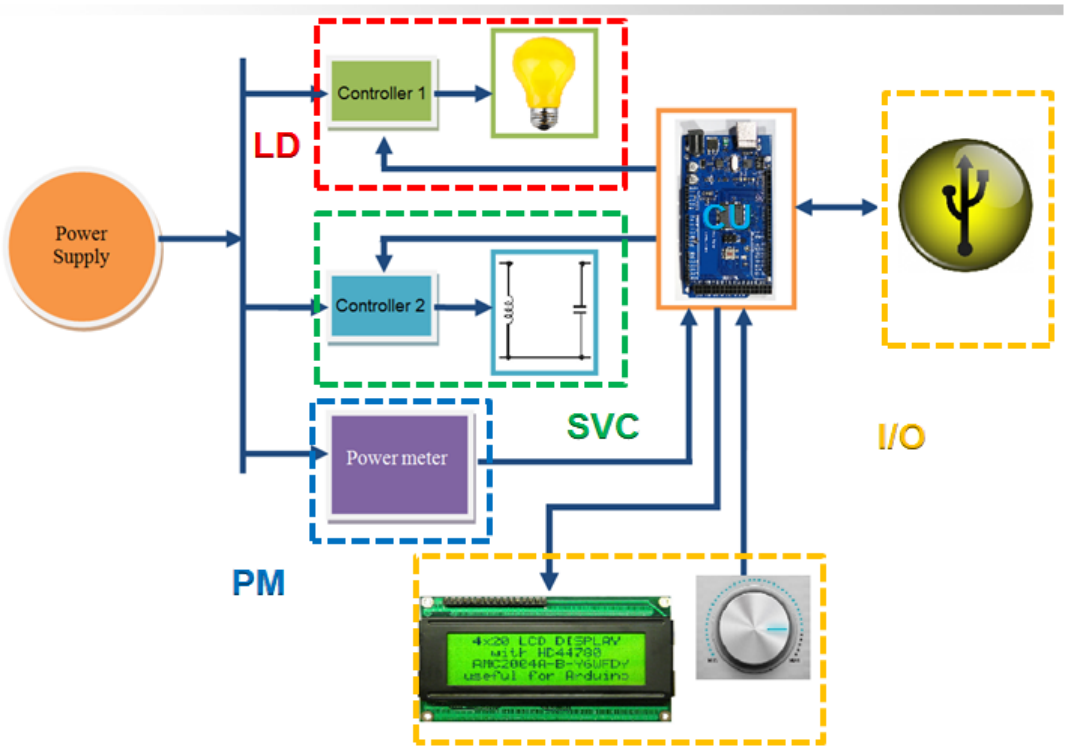

Figure 4.1: Load Module Block Diagram

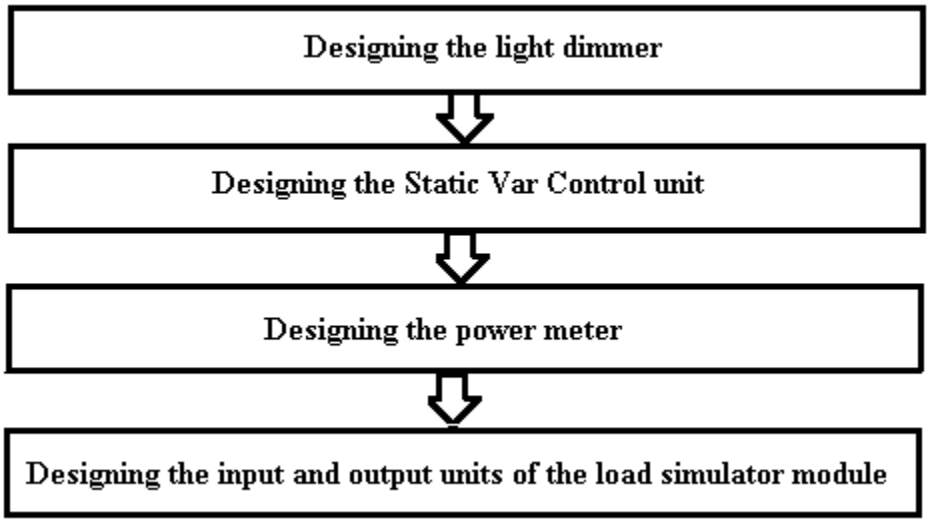

Figure 4.2: Project Flowchart

The implementation of the following activities is discussed in the succeding sections of the methodology.

4.1 Designing the light dimmer

The purpose of the light dimmer is to control the amount of power absorbed by the lighting load unit, so that the lighting load can have a variable real power consumption.

The final design of our light dimmer consists of an isolation circuits and bidirectional switches.

A single phase representation of the light dimmer circuit is shown in Fig **??**.

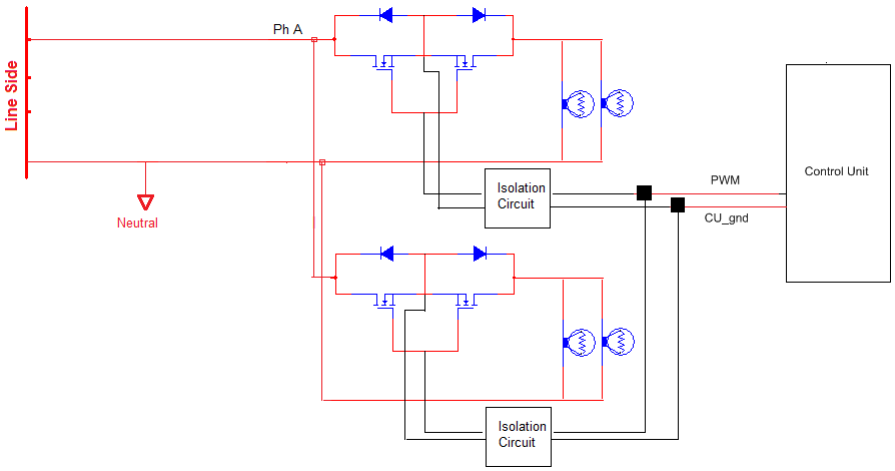

Figure 4.3: Single phase representation the light dimmer

4.1.1 Software Simulation

The Software we used in simulating the light dimmer is PSIM.

4.1.2 Hardware Creation

4.1.2.1 Isolation circuit

The schematics of isolation circuit is shown in Fig. 4.4. The isolation circuit consists of an optocoupler for isolation of the arduino pwm signals from the gate drive, and a voltage buffer to eliminate the loading effect of mosfet's gate. The optocoupler and op amp is chosen based on their availability in local electronics store and in the lab. Though working at the typical 500 Hz frequency of arduino's pwm ouput, our design of isolation circuit is not applicable for high switching purposes. Our isolation circuit can achieve a maximum switching frequency of 3kHz before destroying the control signals. The benefits of having low frequency switching are the reduction of harmonics introduced in the system and the negligibilty of the the gate capacitance of the mosfet. If the

designer is interested in operating at much higher frequency the major considerations would be buying a more expensive and non-locally available fast response opto-coupler (e.g. HCPL-4503), and a high slew rate operational amplifier (e.g. MC34071).

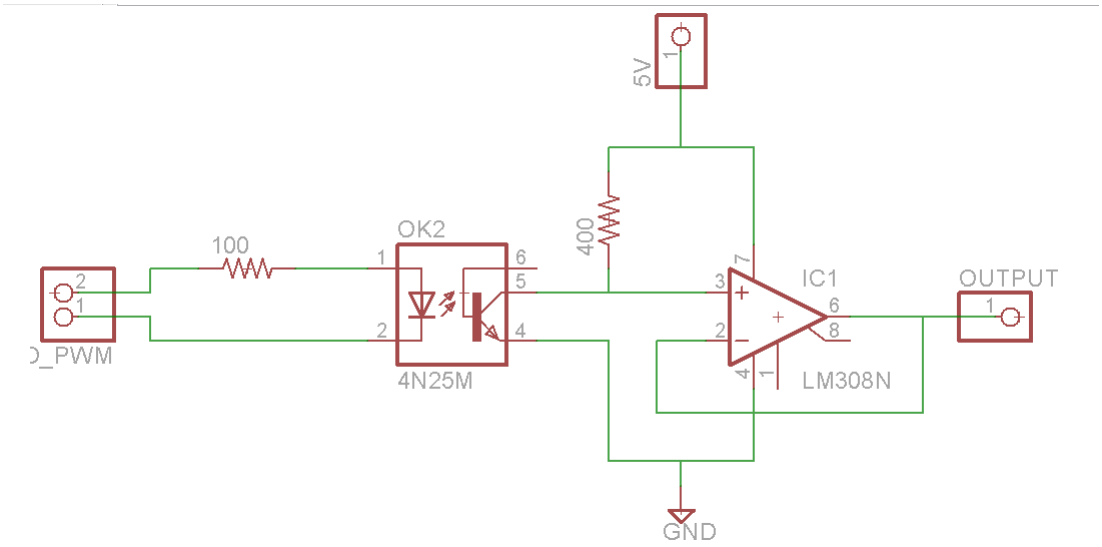

Figure 4.4: Light Dimmer Isolation Circuit

4.1.2.2 PWM signals

The arduino's pwm ouputs has a default frequency of 500Hz. To change the duty cycle of the pwm output the analogwrite function is used. A sample arduino program for changing the duty cycle is shown in Fig. 4.5. The analogwrite function of the arduino is very limited (brightness = 0 to 255 only). It can only produce 256 different pwm duty ratios. Thus the close to continuous load variation is impossible to implement with arduino's default settings. To achieve a near continuous variation of the load, the internal registers corresponding to the pwm pins of arduino must re-programmed. Fig. 4.6 shows a sample code for editing the frequency and range of pwm pins of the arduino micro-controller. The frequency and duty ratio range of the pwm signal is edited through ICR4. As seen on the code the 6,7,8 pwm pins are set to operate at 16MHz÷16000 = 1kHz, and can be varied in from 0-16000.

```
int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    //
//how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
  analogWrite(led, brightness);
}
```

Figure 4.5: Snippet of code for controlling arduino's pwm output

```
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

const int range = 16000;
void setup() {

// this set-up affects the range and frequency of 6 7 and 8 pwm pins of Arduino Mega
sbi(TCCR4B, CS40);        // clear bits for register n where n is from OCRnA or OCRnB
cbi(TCCR4B, CS41);
cbi(TCCR4B, CS42);
sbi(TCCR4B, WGM43);
sbi(TCCR4B, WGM42);
sbi(TCCR4A, WGM41);
cbi(TCCR4A, WGM40);

ICR4= range;               // frequency = 16MHz/ ICR4 for 6, 7 and 8..... and duty cycle from 0 - ICR4....
pinMode(6,OUTPUT);         // Load 1
analogWrite(6,1);          // for intializing the PWM
OCR4A =3600;               // can be varied from 0-16000
```

Figure 4.6: Snippet of Revised Arduino code for increasing the range of the analogWrite function

4.1.2.3 Mosfet Selection

The real load per phase consists of 4 100W light bulbs paralleled at 230V supply thus the maximum current that each mosfet must handle should be 400W/230 = 1.74 A. However considering the high harmonic current produced by the switching mosfets, one must provide a leeway on considering the appropriate MOSFET for the light dimmer. The main parameters we considered in selecting the MOSFET are the VDS which must 230V, maximum current of the load, and availability in the local electronics store. The MOSFET we used is IRFP460 which has a maximum Vds of 500 V and maximum current of 13A at 100 degree Celsius.

Though IRFP460 MOSFETs can withstand the voltage and current requirements of our 4 100W light bulbs, we separated the circuit for each pair of 100W light bulbs as shown in Fig.**??**

because of temperature limit of the MOSFETs. The IRFP460 heats too much even it is installed with locally available heatsinks so its mandatory for us to separate each pair of light bulbs.

4.2 Designing the Static Var Controller

As shown in our RRW the Static Var Control is used for correcting the voltage of power systems by injecting or absorbing an amount of reactive power in the system. In our case we will use it to model reactive power consuming/creating load. The lagging load can modeled by switching out the capacitors connected in the load module and adjusting the voltage seen by the inductor. While the leading load can be modeled by varying the voltage across the inductor with the capacitor still intact to the load module.

4.2.1 Software Simulation

Like in the light dimmer the software simulation of the static var controller is implemented in the simulator PSIM.

4.2.2 Hardware Creation

The proposed design of the Static Var controller as shown in Fig.2.12.

4.2.2.1 Logic Signals

The creation of the logic signals consists of 2 parts. First is the creation of the 60Hz pulse generator as shown in Fig.4.7 . The output of 60Hz pulse generator is a square wave from 0 to 5V that is synchronized with the mains ac supply. The output of the 60 Hz pulse generator together with the pwm signal of the arduino is then fed to the logic circuits to produce the 4 necessary signals in driving our inductive load. Fig. 4.8 shows the eagle schematic of the logic circuit. This consists of commercially available logic circuits 74LS04n(inverter) and 74LS32n(OR)

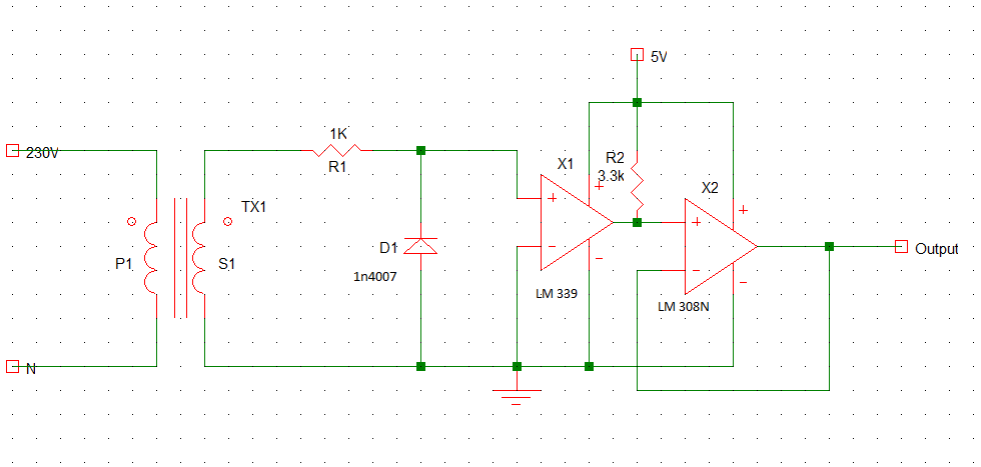

Figure 4.7: 60 Hz Pulse Generator

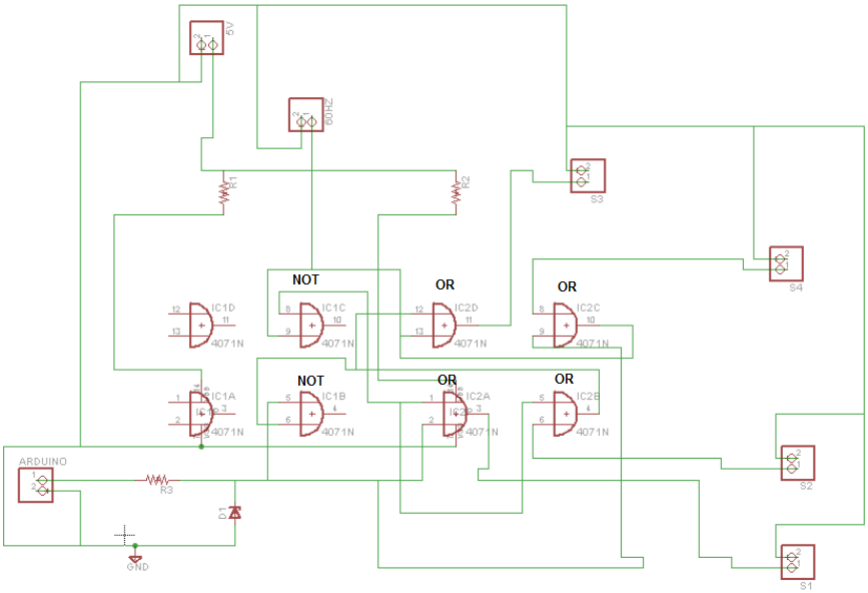

Figure 4.8: Logic Circuit

4.2.2.2 IGBT gate driver

Unlike with the light dimmer we try to implement the gate driver of the igbt of the static var controller with higher frequency(i.e 10kHz). Higher frequency switching is necessary for faster dynamic response and minimize the charge stored in the inductor.

Our first implentation IGBT gate driver is shown in Fig. 4.9

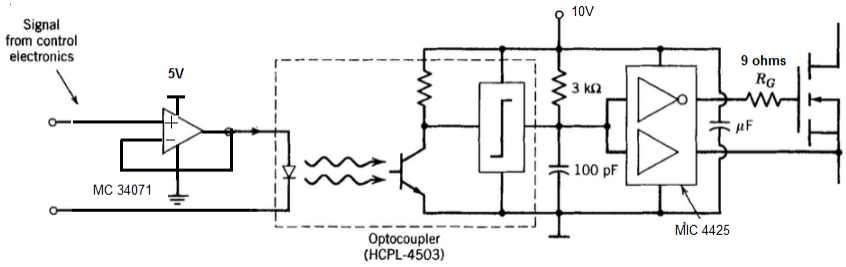

Figure 4.9: IGBT Gate Driver

4.2.2.3 Selection of inductive load

There are available inductive loads in the laboratory and these are 25mH and 16.7mH with 22AWG. Since we proposed an inductive load that can consume a maximum of 400 Var. We need to reduce the voltage of the mains supply to a lower value. There are available control transformers in laboratory that have turns ratio of 115 to 346. Computing the input voltage we have $230 * \frac{115}{346} = 76.4451V$ at the secondary winding of the control transformer. Having a one 25mH and one 16.7mH inductor. The total inductve load will be 41.7mH. The maximum Var consumption given the value of the inductive load and the input voltage will be $= \frac{76.4451^2}{2\pi * 60 * 41.7m} = 371.7337$ Var

4.3 Designing The Power Meter.

The design of the power meter is shown in Fig. 4.10

An IC called CS5464 is needed in the creation of Power Meter. The CS5464 is a power measurement integrated circuit which requires a sample of the voltage and current that is less than 250mV p-p. Potential transformer and current transformer are needed to sense the analog currents and voltages of the load. After that, it is connected to the voltage and current pins of the circuit where voltage division takes place before going to the pins of the CS5464. After the circuit is formed, it will be integrated to the Arduino. A program is created to read and display the the processor registers of the CS5464. Each register of the CS5464 contains different parameters like voltage, current, real power and reactive power. The whole arduino code is shown in the appendix.

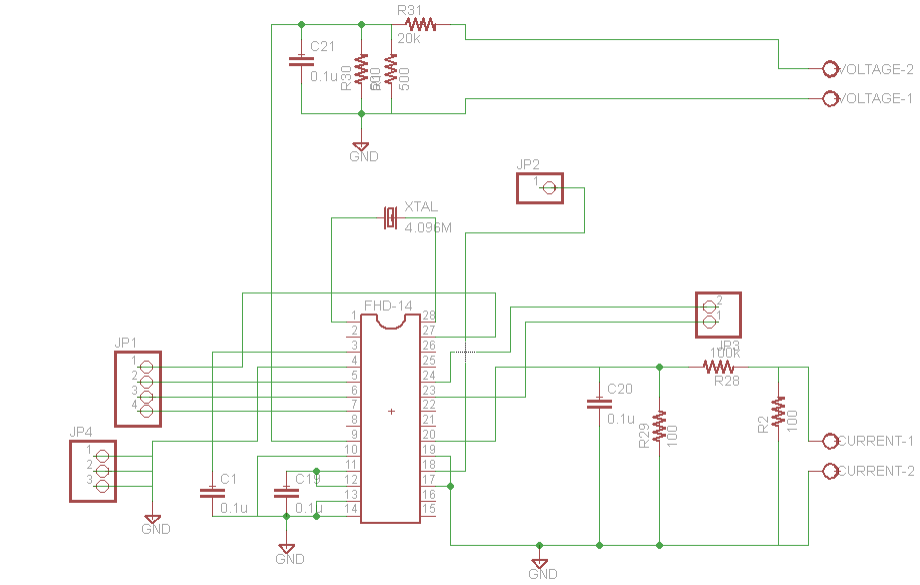

Figure 4.10: Schematics of Power Meter

4.4 Designing the Input and Ouput units of the Load Simulator Module

To be able to make our load simulator module user-friendly, we willl design the input and output units of our module. Fig.4.11 summarizes the input and output characteristics of the module we propose. As seen in the figure the control units for input and output are separated so that both the input and output functions are capable of functioning at the same time.

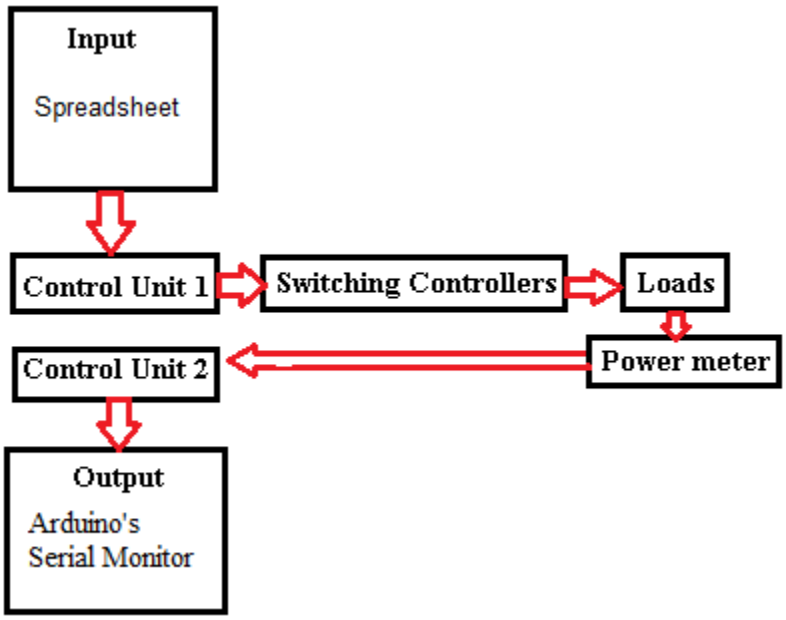

Figure 4.11: Input and Output characterization

4.4.1 SD Card Module

The SD Card module is compatible with the gizduino x of E-Gizmo. So proper placement of the SPI pins of the SD Card Module and the Arduino Mega must be done.

For SPI communication 4 connections must be wired between the SD card module and the Arduino Mega. The four connections are summarized below.

PINS for SPI connection		
SPI Signal	SD Car Module	MEGA
SCLK(clock)	13	52
MISO(data out)	12	50
MOSI(data in	11	51
SS(Slave Select)	10	53

Figure 4.12: SPI connections of the SD Card module and ARduino Mega

Old arduino Megas are by default incompatible with the SD Card Module so a Software SPI must be programmed. The Software SPI is a function for setting the SPI signals of arduino Mega. Shown in the appendix is the complete arduino code for communicating the the SD Card module and the Arduino Mega.

The Input of the load simulator is gathered from the SD Card connected in the SD Card Module. The datas are parsed from a .csv file named Input.csv. and passed to the Arduino Mega. Then the arduino mega will provide necessary pwm signals to the mosfet and igbt drivers.

Chapter 5

Results and Analysis

5.1 Light dimmer

5.1.1 Software Simulation

The Software simulation and gating signals of the light dimmer are already done. In the simulation, real power load can be varied from 0 to 400W as shown in the images.

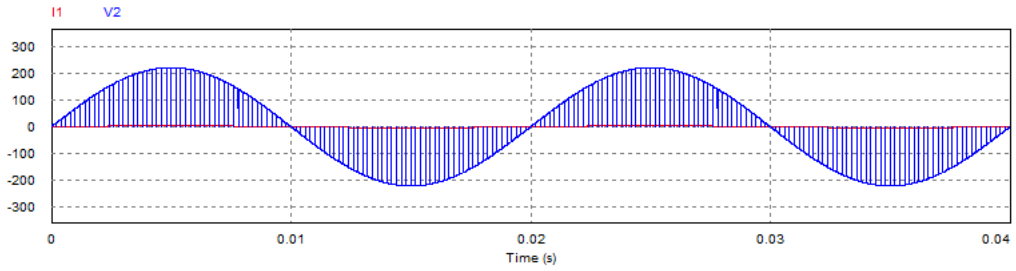

Figure 5.1: Waveform at large duty cycle

Real Power	
Time From	2.7777000e-007
Time To	3.9999991e-002
I1 vs. V2	4.0221285e+002

Figure 5.2: Real Power of the waveform above

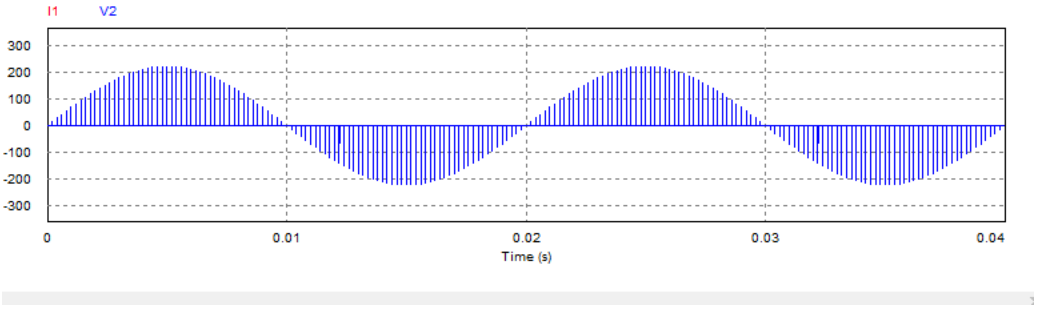

Figure 5.3: Waveform at min duty cycle

Real Power	
Time From	2.7777000e-007
Time To	3.9999991e-002
I1 vs. V2	1.1203698e+000

Figure 5.4: Real Power of the waveform above

5.1.2 Hardware simulation

The output voltage of the light bulbs displayed in the oscilloscope are shown below and the graphs shows that it vary according to the duty cycle of the arduino pwm signal.

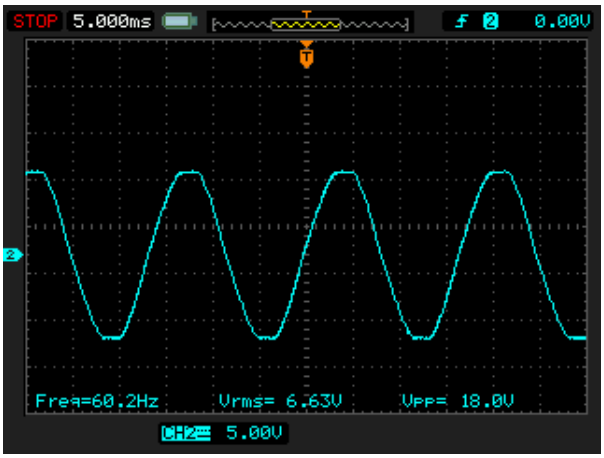

Figure 5.5: Waveform at 1 duty cycle

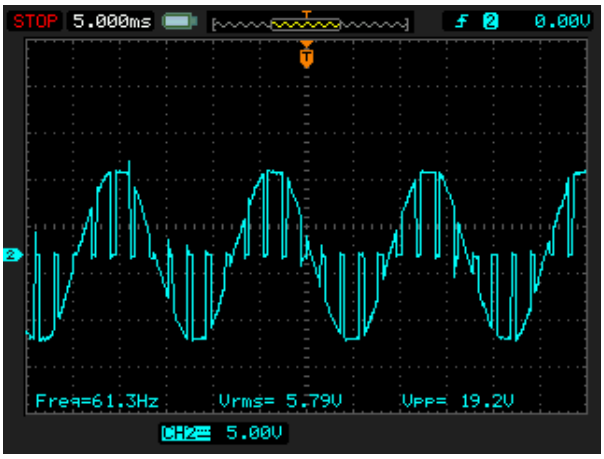

Figure 5.6: Waveform at .75 duty cycle

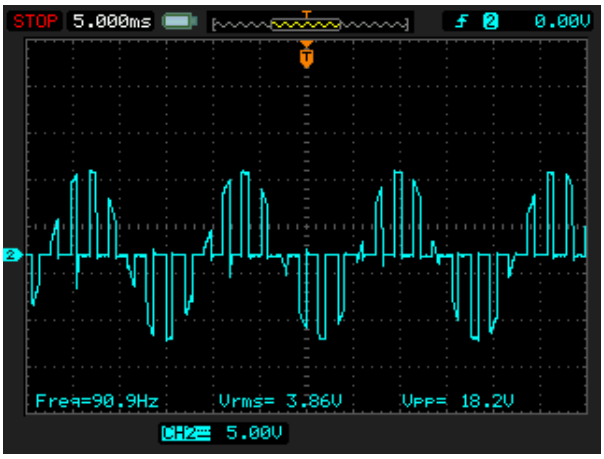

Figure 5.7: Waveform at .25 duty cycle

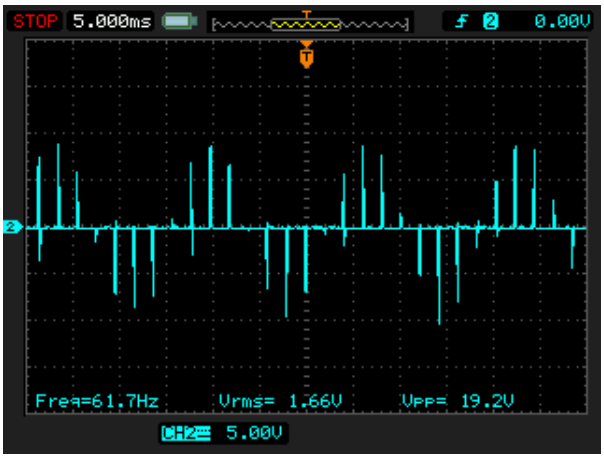

Figure 5.8: Waveform at .05 duty cycle

The light dimmer again is tested at higher frequency(i.e. 1kHz) and 4 light bulbs are dimmed in a single phase supply. The corresponding duty cycle(maximum is 15990) of the PWM for each power value is measured and mapped. The duty cycle is then fed to the microcontroller and allow the light dimmer to consume the power required. The tabulation of data are shown in Fig. **??**. The pwm value and the real power are linearized as shown in Fig. **??**.

Power(W)	value	linearized values	Duty Cycle
25	15350	16202.6	0.959615
50	14800	15070.2	0.925231
75	13932	13937.8	0.870968
100	13000	12805.4	0.812703
125	12000	11673	0.750188
150	11000	10540.6	0.687672
175	9900	9408.2	0.618905
200	8500	8275.8	0.531383
225	7300	7143.4	0.456364
250	6100	6011	0.381345
275	5000	4878.6	0.312578
300	3600	3746.2	0.225056
325	2400	2613.8	0.150038
350	900	1481.4	0.056264

Figure 5.9: PWM values vs the real power reading of the Wattmeter

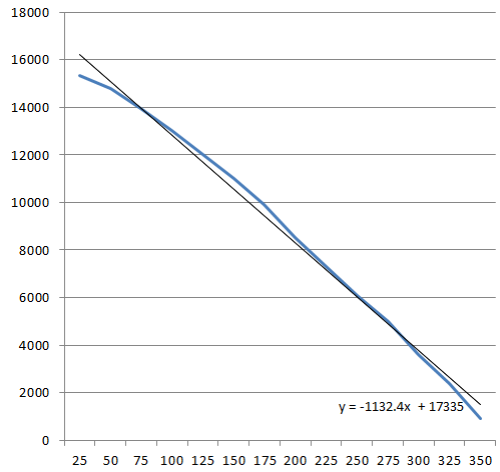

Figure 5.10: Graph of the mapped values

5.2 Static Var Controller

5.2.1 Software Simulation

The schematic diagram of the Static Var compensator in PowerSim Silmulator is shown in Fig.5.11. The gate signals and output voltage of the inductive load are shown in Figs. 5.12 and 5.13 respectively.

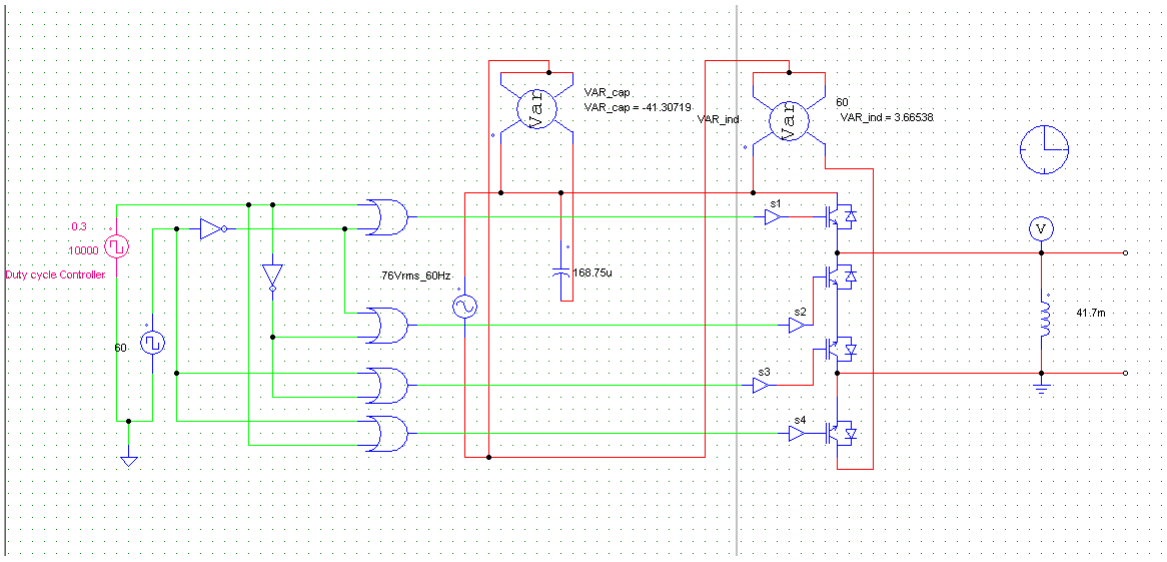

Figure 5.11: Schematic Diagram of Static Var Controller

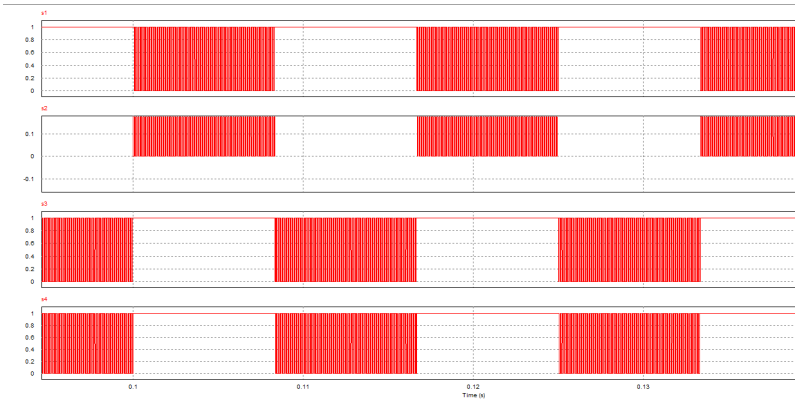

Figure 5.12: Gate Signals

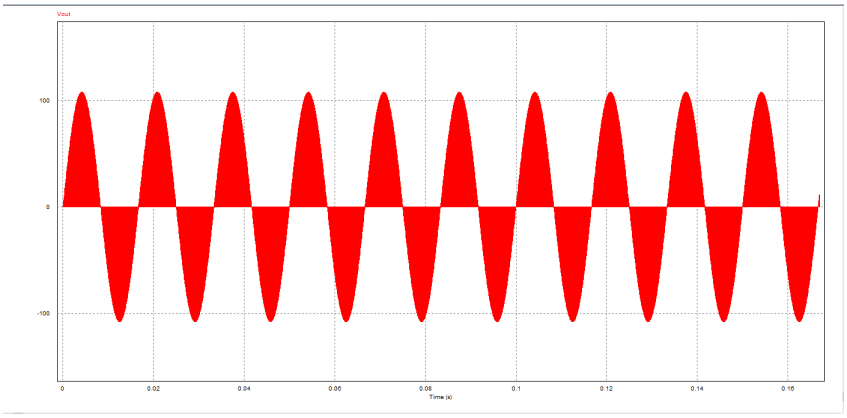

Figure 5.13: Inductive Load Waveform

5.2.2 Hardware Creation

The experimental output voltage of the pulse generator and the logic circuit are shown in Figs. **??**,**??**, and **??**.

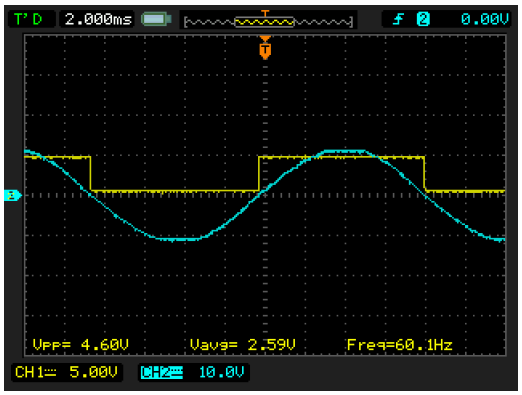

Figure 5.14: Pulse Generator Output waveform vs 230V Sinuoidal voltage

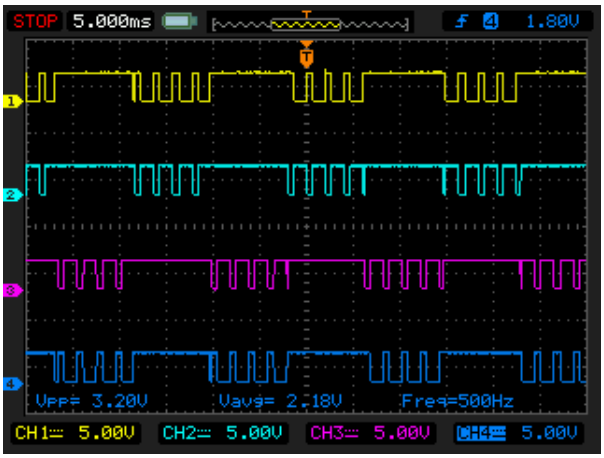

Figure 5.15: Logic circuit signals at Low frequency

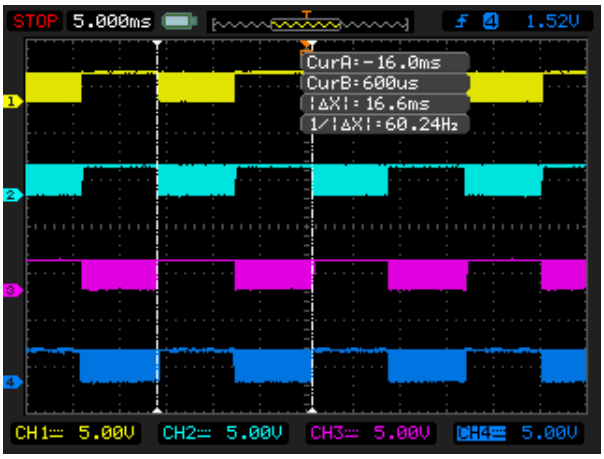

Figure 5.16: Logic circuit signals at high frequency

Currently, the said implementation of Static Var controller is not working and causes the destruction of IGBTs. It might be because of a high current passing through the base of the IGBT driver so the MOSFET driver IC is replaced by a high slew rate buffer Op-amp(i.e. MC34071). However, the IGBTs are still destroyed the by second implementation. Most likely, the primary reason for the destruction of IGBTs is having the same 10V supply for the 4 drivers of the Static Var controller Isolation/pulse transformers must be installed to completely isolate control signals of four gate drivers. To be certain, there must be further study on the gate drivers to avoid the destruction of expensive IGBTs.

5.3 Power Meter

The power meter created uses the CS5464 IC as said in the methodology.The current and potential transformers available in the PSSL are used to lower the current and voltage, respectively. The Arduino mega 2560 has only 1 set of Serial Peripheral pins, only 1 phase can be measured at a time. To measure the other phase, a rotary switch is used to change the current that is being sampled. Only current, voltage and power are extracted from the CS5464 since the other parameters can be calculated form that. The value extracted from the CS5464 is calibrated in the arduino microcontroller. Since the slope formed using the output of CS5464 is not linear, a piecewise function is implemented using if-else statement to make the the power meter more accurate.

5.4 Input/Output Module

The Arduino microcontroller can read a .csv file from the a Secure Digital Card. It can parse the texts and converts it into an integer. The resulting integers are stored in an array that can be accessed by the arduino to set the time interval, real and reactive power of each phase.

The serial monitor of the arduino will display the current, voltage, real power, reactive power, apparent power and power factor. The display will update their values every second.

5.5 Integration

The single phase light dimmer is now integrated with the power meter and input/output unit. To input a particular load profile, power values are loaded in the corresponding phase and time interval. The power meter will show RMS current, RMS voltage, active, reactive, apparent power and power factor in the LCD display. The result of a single phase light dimmer is shown below.

		Current cs				
Theoretical	Actual	1st	2nd	3rd	average	percent error
0.10869565	0.54	0.54	0.55	0.56	0.55	1.851851852
0.2173913	0.725	0.71	0.72	0.71	0.713333333	1.609195402
0.32608696	0.832	0.83	0.83	0.83	0.83	0.240384615
0.43478261	0.96	0.96	0.96	0.97	0.963333333	0.347222222
0.54347826	1.056	1.06	1.06	1.06	1.06	0.378787879
0.65217391	1.14	1.14	1.15	1.15	1.146666667	0.584795322
0.76086957	1.216	1.25	1.25	1.26	1.253333333	3.070175439
0.86956522	1.3	1.31	1.3	1.31	1.306666667	0.512820513
0.97826087	1.356	1.36	1.36	1.36	1.36	0.294985251
1.08695652	1.411	1.41	1.41	1.42	1.413333333	0.165367352
1.19565217	1.45	1.43	1.44	1.43	1.433333333	1.149425287
1.30434783	1.512	1.51	1.5	1.52	1.51	0.132275132
1.41304348	1.54	1.56	1.54	1.53	1.543333333	0.216450216
1.52173913	1.59	1.6	1.61	1.59	1.6	0.628930818
1.63043478	1.63	1.63	1.64	1.65	1.64	0.613496933

Figure 5.17: Current measured by CS5464 and Multimeter

		Power cs				
Theoretical	actual	1st	2nd	3rd	average	percent error
25	25	25	43	32	33.33333	0.333333333
50	50	46	48	49	47.66667	0.046666667
75	75	81	82	75	79.33333	0.057777778
100	100	97	107	87	97	0.03
125	125	124	124	128	125.3333	0.002666667
150	150	154	155	151	153.3333	0.022222222
175	175	175	186	173	178	0.017142857
200	200	198	197	194	196.3333	0.018333333
225	225	222	219	221	220.6667	0.019259259
250	250	246	250	246	247.3333	0.010666667
275	275	271	277	271	273	0.007272727
300	300	294	296	301	297	0.01
325	325	323	324	326	324.3333	0.002051282
350	350	347	350	353	350	0
375	375	375	379	378	377.3333	0.006222222

Figure 5.18: Power measured by CS5464 and wattmeter

Theoretical	Actual	percent error
0.108695652	0.54	396.8
0.217391304	0.725	233.5
0.326086957	0.832	155.1466667
0.434782609	0.96	120.8
0.543478261	1.056	94.304
0.652173913	1.14	74.8
0.760869565	1.216	59.81714286
0.869565217	1.3	49.5
0.97826087	1.356	38.61333333
1.086956522	1.411	29.812
1.195652174	1.45	21.27272727
1.304347826	1.512	15.92
1.413043478	1.54	8.984615385
1.52173913	1.59	4.485714286
1.630434783	1.63	0.026666667

Figure 5.19: Theoretical current vs actual current

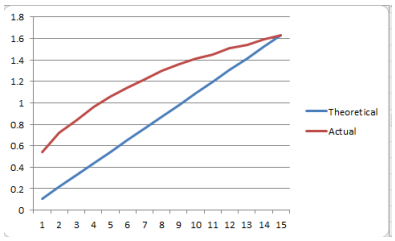

Figure 5.20: Graph of the current

The current measured using the multimeter and CS5464 are found to be close enough but the theoretical values are far from both. As the current increases, the error between the actual and the theoretical current decreases. At current is equal 1.63 A, the theoretical and actual values converge.

The non-linearity of the actual current is brought about by the harmonics produced by the PWM of the light dimmer. At higher duty cycle where the power is low, the harmonics produced are too high which results to higher current relative to the original. Using the current, voltage and real power, the apparent and reactive power can be computed manually. It turns out that the harmonics causes a reactive power to exist in the system.

The Panel Design for the whole load module are shown in the appendix.

Chapter 6

Conclusion and Recommendation

6.1 Conclusion

In the PSSL, the Programmable Variable Load Module the first attempt in creating a better load module. A huge part of the project is succesfully achieved with the unreal parts deemed unfinishable. Since load in a real power system is actually changing overtime, the ability to simulate time-varying load is certainly a big leap from the existing load module in the PSSL. This will closen the gap between the power system simulation and real power system. Due to that, this load module will be very useful for training and educating individuals on the operations of the power system.

6.2 Recommendation

The light dimmer produced too much harmonics when working with a higher duty cycle. Sometimes, this harmonics affect the operations of the microcontroller and even the personal computer. It is suggested to make a filter to get rid of the harmonics. It is also better to work on the lower duty cycle, where the harmonics are low. That way, it is ensured that the harmonics would be too small to be considered.

Once in a while a trash data is produced by the CS5464. A mechanism where the corrupted data is identified and rejected is recommended to solve this problem.

Due to the power meter and the input/output unit, the Programmable Variable Load Module is certainly useful alone. Load profile can be simulated and parameters are displayed. But, if it is integrated with other modules , its effectiveness increases drastically. This load module together with generator and transmission module, is sufficient enough to model a complete real

power system.

Creation of multiple load module is also a good idea. With these, multiple feeder systems or a whole distribution system can be created. A fault simulator can also be integrated to the load module. This feature can show what will happen to the each phase of the load when different faults occur just like in real power system.

Lastly, this load module can be used in a number of experiments. Since it is on the demand side of the power system, the generator modules can be tested if it can cope up with the varying load demands.

Appendix A

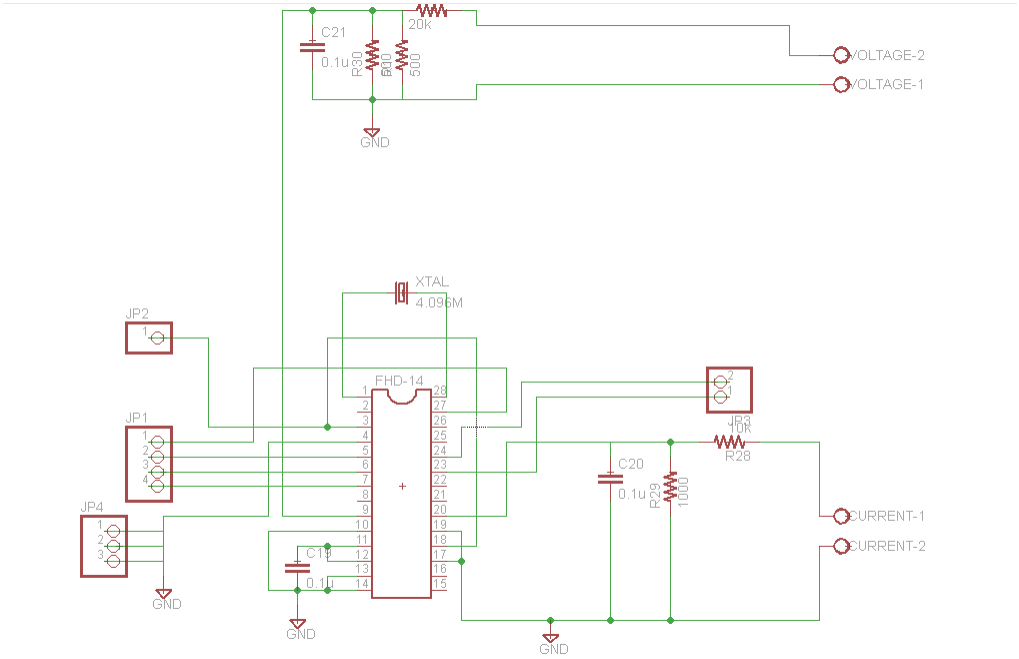

Figure 6.1: Design of the Power Meter

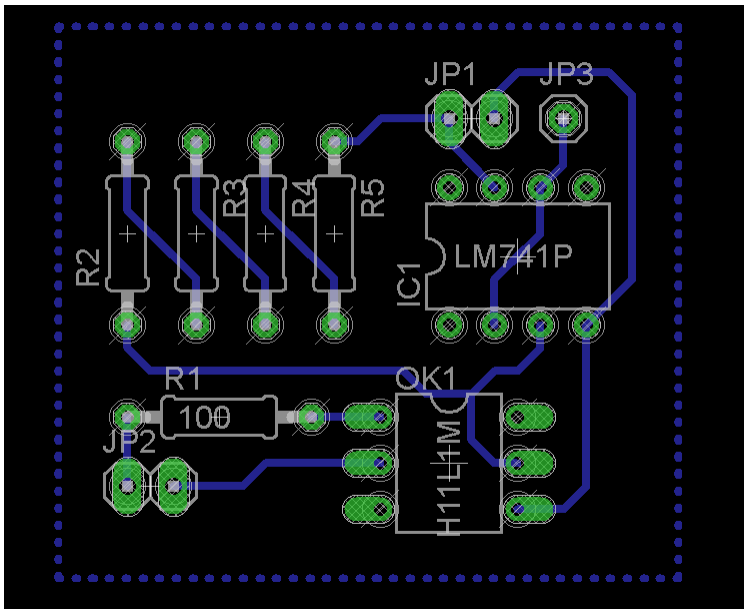

Figure 6.2: Board layout of the isolation circuit

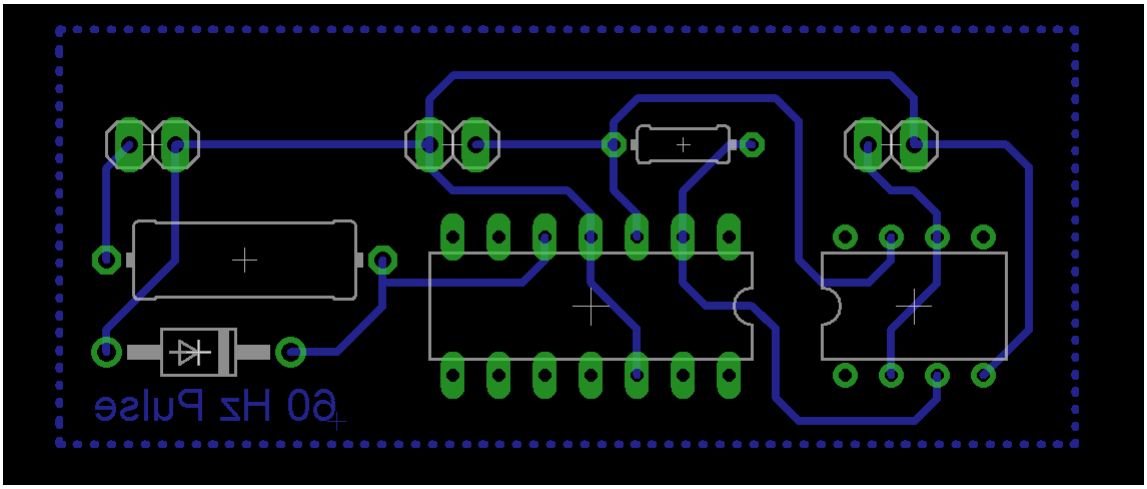

Figure 6.3: Board layout of the 60 Hz Pulse Generator

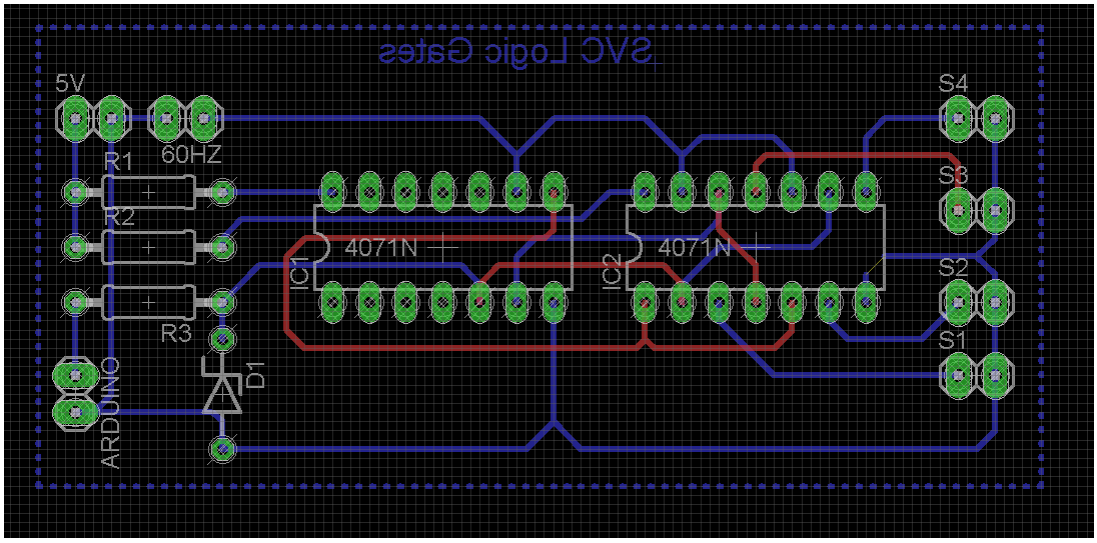

Figure 6.4: Board layout of the Logic Circuit

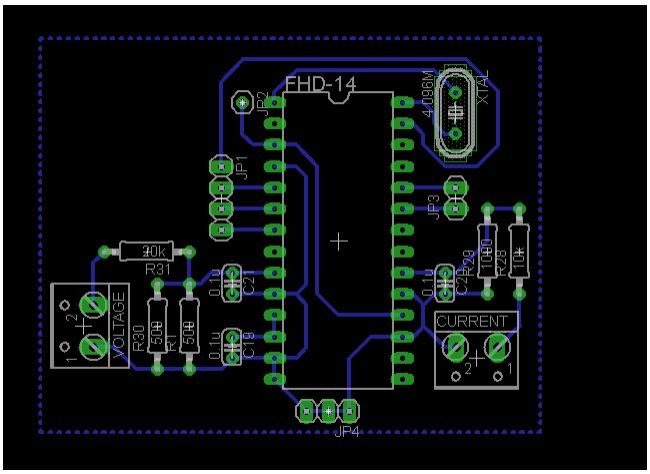

Figure 6.5: Board Layout of the Power meter

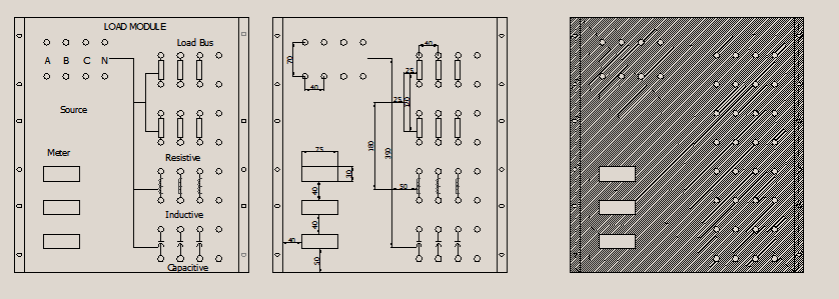

Figure 6.6: Panel design of the Load module

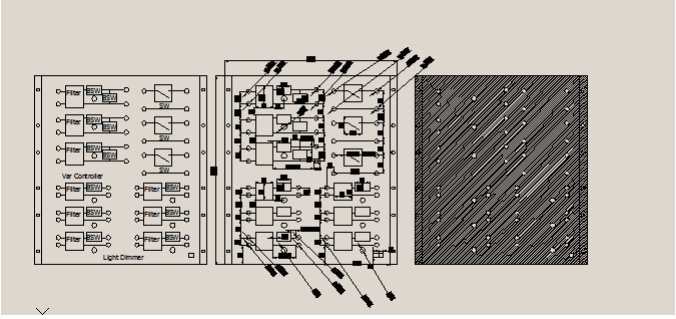

Figure 6.7: Panel design of the Drivers of Static Var Contoller and Light Dimmer

Appendix B

Listing 6.1: Light dimmer control

```
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif
//Arduino PWM - Mega 2560 pins, registers and changing the frequency and range
//Arduino Pin     Register
//2                OCR3B
//3                OCR3C
//4                OCR4C  need clarifications or OCR0n--- > Do not Use
//5                OCR3A
//6                OCR4A
//7                OCR4B
//8                OCR4C
//9                OCR2B
//10               OCR2A
//11               OCR1A
//12               OCR1B
//13               OCR0A  -- > Do not Use
//44               OCR5C
//45               OCR5B
//46               OCR5A



const int range = 15996; //for light bulb use 1kHz or ICR4 = 16000

void setup() {
  // put your setup code here, to run once:
// for 6 7 and 8
sbi(TCCR4B, CS40);         // clear bits for register n where n is from OCRnA or OCRnB
cbi(TCCR4B, CS41);
cbi(TCCR4B, CS42);
sbi(TCCR4B, WGM43);
sbi(TCCR4B, WGM42);
sbi(TCCR4A, WGM41);
cbi(TCCR4A, WGM40);



ICR4= range;                  // frequency = 16MHz/ ICR4 for 7 and 8..... and duty cycle from 0 - ICR4....
pinMode(6,OUTPUT);           // Load 1
analogWrite(6,1);           // for intializing the PWM
OCR4A =3600;
```

```
pinMode(7,OUTPUT);          // Load 2
analogWrite(7,1);          // for intializing the PWM
OCR4B = 0;
/*
pinMode(8,OUTPUT);
analogWrite(8,1);          // for intializing the PWM
OCR4C = 0;
*/
}

void loop() {
  // put your main code here, to run repeatedly:

OCR4A += fade;
//OCR4B += fade;
if(OCR4A == 0 || OCR4A == range){
fade = -fade;
}
*/
/*
if(OCR4B == 0 || OCR4B == range){
fade = -fade;
}
*/
/*
OCR4C += fade;
if(OCR4C == 0 || OCR4C == range){
fade = -fade;
}
*/
delay(20);
}
```

Listing 6.2: Power meter program

```
// For CS5464 chip from Cirrus
// Read the peak current on channel I2
// Check for zero value coming back from CS5464, if so, then reset the device

#include <SPI.h>

// Pin configurations
// set pin 12 as the slave select for the digital pot:
const int slaveSelectPin = 53;

int i;
unsigned int check;

// set pin 11 as the Reset pin for the CS5464:
const int resetPin = 3;



// Create a data type for the 4 Byte data and command packet
union FourByte {
 struct {
   unsigned long value: 24; //24bit register values go in here
   byte command: 8; //8bit command goes in here.
 };
 byte bit8[4]; //this is just used for efficient conversion of the above into 4 bytes.
};

// Initialize SPI and pins
void setup() {
```

```
 // reset the CS5464 device


 pinMode(resetPin, OUTPUT);
 delay(100);
 digitalWrite(resetPin, LOW);
 delay(100);
 digitalWrite(resetPin, HIGH);
 delay(100);

 SPI.begin();
 SPI.setBitOrder(MSBFIRST);
 SPI.setDataMode(SPI_MODE3); //I believe it to be Mode3
 SPI.setClockDivider(SPI_CLOCK_DIV16);
 pinMode(slaveSelectPin, OUTPUT);
 digitalWrite(slaveSelectPin, HIGH);

  Serial.begin(9600);
 delay(2000);  // Pause allowing time to open the serial monitor
 Serial.println("setup...");

 //Perform software reset:
 SPI_writeCommand(0x80);
 delay(2000);
 unsigned long status;
 do {
   status = SPI_read(0b00011110); //read the status register  0x1E
   //status = SPI_read(0x1E); //read the status register  0x1E
   //status &= (1UL << 23);
  // status &= (0x00000001 << 23);
   Serial.print("\nstatus: ");
   Serial.println(status);
   Serial.print("\n");
   delay(2000);
   //status &= (1UL << 23);
 } while (!status);

  // SPI_writeCommand(0xE0); //Set single conversion mode
 SPI_writeCommand(0xE8); //Set continuous conversion mode

 // Print the configuration register, to confirm communication with the CS5464
 check = SPI_read(0x00); //read the config register.
 Serial.print("Config = ");
 Serial.println(check, HEX);

  /*
   //example of writing a register
   union FourByte data;
   data.command = 0b01000000; //Write to config register
   data.value = 1; //This is the default value from datasheet, just using it as an example.
   SPI_writeRegister(data);
   */
}

// Main loop
void loop() {
 delay(1000 );
 //example of reading data

  unsigned long  voltage;
  unsigned long current;
  unsigned long power;
  unsigned long real;
  unsigned long reactive;
  unsigned long power_factor;
```

```
  unsigned long peak;
  float volt;
  float cur;
  float app;
  float reala;
  float reactivea;
  float pf;

  current = SPI_read(0b00001010); //read Register 5 Current Channel 1
  voltage = SPI_read(0b00001100); //read Register 6  voltage Channel 1

  real = SPI_read(0b00001000); //read Register  4‘
  peak = SPI_read(0b00100100); //read Register  4‘


  //current = current >> 5;
  voltage = voltage >> 15;
  //real = real >> 10;


 if((real>=16769800)&&(real<=16772300)){
  reala=0;
}
else if((real>=16772300)&&(real<=16775300)){//25-50
   reala=(float)((real)-16770000)/95;
}
else if((real>=16775300)&&(real<=16777200)){//75
   reala=(float)((real)-16775300)/22.4;
}
else if(real<=400){
  reala=75;
}
else if(real<=2268){
  reala= (float)(real/21);//75
}
else if(real<=4370){
  reala=20+(float)(real/34);//100
}
else if(real<=6470){
  reala=15+(float)(real/44);//125
}
else if(real<=8800){
  reala=7+(float)(real/50);//175-200
}
else if(real<=9700){
  reala=(float)(real/50)-10;//175-200
}
else if(real<=12000){
  reala=(float)(real/60)+2;//200-225
}

else if(real<=17010){
  reala=5+(float)(real/65);//225-250
}
else if(real<=19300){
  reala=(float)(real/63)-7;//250-275
}
else if(real<=21100){//275-300
  reala=(float)(real/68);
}
else if(real<=22700){//300-325
  reala=(float)(real/69);
}
else{//325---
  reala=(float)(real/70)-7;
```

```
}
if(current<= 202700)
{
cur=0;
}
else if( (current>=202700)&&(current<=213000)&&(reala<=250)){//0-.54
  cur = (float)((current)-200000)/16000.0;
}
else if( (current>=212700)&&(current<=240000)&&(reala<=250)){//.54-.725
  cur = (float)((current)-200000)/22500.00;
}
else if( (current>=240000)&&(current<=266000)&&(reala<=250)){//.725-.832
 cur = (float)((current)-200000)/51000.00;
}
else if( (current>=266000)&&(current<=290000)&&(reala<=250)){//.832-.96
  cur = (float)((current)-200000)/74000.00;
}
else if( (current>=290000)&&(current<=309000)&&(reala<=250)){//.96-1.056
 cur = (float)((current)-200000)/91000.00;
}
else if( (current>=309000)&&(current<=320000)&&(reala<=250)){//1.056-1.14
 cur = (float)((current)-200000)/101000.00;
}
else if( (current>=320000)&&(current<=326000)&&(reala<=250)){//1.14-1.216
 cur = (float)((current)-200000)/103800.00;
}
else if( (current>=326000)&&(current<=332500)&&(reala<=250)){//1.216-1.3
 cur = (float)((current)-200000)/102000.00;
}
else if( (current>=332500)&&(current<=337000)&&(reala<=235)){//1.3-1.356
 cur = (float)((current)-200000)/103500.00;
}
else if( (current>=331100)&&(reala<260)){//1.356-1.411
 cur = (float)((current)-200000)/96000.0;
 // cur=6;
}
else if( (current>=322000)&&(reala<285)){//1.411-1.45
 cur = (float)((current)-200000)/90700.0;
  //cur=5;
}
else if( (current>=315000)&&(reala<310)){//1.45-1.512
  cur = (float)((current)-200000)/81700;
   //cur=4;
}
else if( (current>=308000)&&(reala<335)){//1.512-1.54
cur = (float)((current)-200000)/75000.0;
// cur=3;
}
else if( (current>=304000)&&(reala<360)){//1.54-1.59
 cur = (float)((current)-200000)/70000.0;
 // cur=2;
}
else if( (current>=312000)&&(reala>360)){//1.59-1.63
 cur = (float)((current)-200000)/72000.0;
// cur=1;
}
else{
cur=1.64;
}

if(voltage<=40){
volt=0;
}
else{
  volt = (float)(voltage)*1.4;
```

```
}

app=volt*cur;
reactivea=sqrt((pow(app,2)-pow(reala,2)));
pf=reala/app;
//if(pf){
//   pf=0;}
/*   reactivea=(float)(reactive);
    app= (float)(power);
  pf=(float)(power_factor);*/
 Serial.print("\nCurrent = ");
// Serial.print(voltage, HEX);
 Serial.print ("   ");
 Serial.println(current);
 Serial.println(cur);

 /* Serial.print("\nPeak current = ");
// Serial.print(voltage, HEX);
 Serial.print ("   ");
 Serial.println(peak);
// Serial.println(cur);*/

 Serial.print("Voltage = ");
 Serial.print ("   ");
 Serial.println(voltage);
 Serial.println(volt);

 Serial.print("Real Power = ");
 Serial.print ("   ");
 Serial.println(real);
 Serial.println(reala);

 Serial.print("Apparent Power = ");
 Serial.print ("   ");
 Serial.println(app);

  Serial.print("Reactive Power = ");
 Serial.print ("   ");
 Serial.println(reactivea);\

  Serial.print("Power Factor= ");
 Serial.print ("   ");
 Serial.println(pf);

  /*Serial.print("Apparent Power = ");
 Serial.print ("   ");
 Serial.println(power);
 Serial.println(app);*/

 /*Serial.print("Reactive Power = ");
 Serial.print ("   ");
 Serial.println(reactive);
 Serial.println(reactivea);

  Serial.print("Power Factor= ");
 Serial.print ("   ");
 Serial.println(power_factor);
 Serial.println(pf);*/


 // check if the device has accidentally reset
 // Found that noise from switching AC loads can cause the CS5464 to lock-up.
//   need to improve power or signal filtering, but this is a patch for initial testing.
   if (voltage == 0) { // If we get a 0 reading, then reconfigure the device
   SPI_writeCommand(0xE8); //Set continuous conversion mode
```

```
   delay(100);
   unsigned long check = SPI_read(0x00); //read the config register.
   Serial.print("Config = ");
   Serial.println(check, HEX);
 }
}

// Send a Command to the CS5464 - see the data sheet for commands
void SPI_writeCommand(byte command) {
 digitalWrite(slaveSelectPin, LOW); //SS goes low to mark start of transmission
 union FourByte data = {0xFEFEFE, command}; //generate the data to be sent, i.e. your command plus the Sync bytes.
 Serial.print("SPI_writeCommand: ");
 for (char i = 3; i >= 0; i--) {
   SPI.transfer(data.bit8[i]); //transfer all 4 bytes of data - command first, then Big Endian transfer of the 24bit value.
   Serial.print(data.bit8[i], HEX);
 }
 Serial.println();
 digitalWrite(slaveSelectPin, HIGH);
}

// Read a register from the CS5464 - just supply a command byte (see the datasheet)
unsigned long SPI_read(byte command) {
 digitalWrite(slaveSelectPin, LOW); //SS goes low to mark start of transmission
 union FourByte data = {0xFEFEFE, command}; //generate the data to be sent, i.e. your command plus the Sync bytes.
 // Serial.print("SPI_Read
 unsigned long val = 0;
 unsigned char v;
 for (char i = 3; i >= 0; i--) {
//for (int i = 0; i <4; i++) {
   //Serial.print("\nsend\n");
   //Serial.print(data.bit8[i], HEX);
   v = SPI.transfer(data.bit8[i]);
   //data.bit8[i] = SPI.transfer(data.bit8[i]); //send the data whilst reading in the result
      //Serial.print("\nrecieve\n");
      //Serial.print(v, HEX);
      val = val << 8;
      val = val +v ;

      //Serial.print(data.bit8[i], HEX);
}

//Serial.print("\nrecieve all\n");
//Serial.print(val, HEX);
//Serial.print("\nlabas\n");
 //  Serial.println();
 digitalWrite(slaveSelectPin, HIGH); //SS goes high to mark end of transmission
 return val;
 //return data.value; //return the 24bit value recieved.
}
```

Listing 6.3: Input/Output Program

```
// An example of the SdFatSoftSpi template class.
// This example is for an Adafruit Data Logging Shield on a Mega.
// Software SPI is required on Mega since this shield connects to pins 10-13.
// This example will also run on an Uno and other boards using software SPI.
//
#include <LiquidCrystal.h>
#include <SPI.h>
#include <SdFat.h>

#define MAXDIGITS 6;

int num_entries =0;
```

```
int values[3];
SdFile myFile;
int skip =0;
int okay =0;        // equal to 1 if the last entry of the user is used
LiquidCrystal lcd(27, 26, 25, 24, 23, 22);

#if SD_SPI_CONFIGURATION >= 3  // Must be set in SdFat/SdFatConfig.h
//
// Pin numbers in templates must be constants.
const uint8_t SOFT_MISO_PIN = 50;
const uint8_t SOFT_MOSI_PIN = 51;
const uint8_t SOFT_SCK_PIN  = 52;
//
// Chip select may be constant or RAM variable.
const uint8_t SD_CHIP_SELECT_PIN = 53;

// SdFat software SPI template
SdFatSoftSpi<SOFT_MISO_PIN, SOFT_MOSI_PIN, SOFT_SCK_PIN> sd;

void setup() {

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
//  Serial.begin(9600);
//  while (!Serial) {}  // Wait for Leonardo

  moving_print("Initializing SD Card.....");
  if (!sd.begin(SD_CHIP_SELECT_PIN)) {

    moving_print("Can't connect to SD card");
    return;
  }

  moving_print("SD Card Connected");
  // Open the file for reading:
  if (!myFile.open("Input.csv", O_READ)) {
    sd.errorHalt("opening Input.csv for read failed");
  }
  moving_print("Input.csv is readable");
  moving_print("Simulating.......");

  // read from the file until there's nothing else in it://
//  int data;
  /*
  while ((data = myFile.read()) >= 0) {
    Serial.write(data);
  }

  // close the file:
  myFile.close();
  */

}
//------------------------------------------------------------------------------
void loop()
{
/*
  while(myFile.available() && okay ==0)
  {
    lcd.clear();
    lcd.print("sec ||kW  ||kVar");
//  lcd.print("    ||    ||     ");
    unsigned long start = millis();
    unsigned long duration = acquire() - start;
//    Serial.println(duration);
```

```
    lcd.setCursor(0,1);
    lcd.print(values[0]);
    lcd.setCursor(4,1);
    lcd.print("||");
    lcd.print(values[1]);
    lcd.setCursor(9,1);
    lcd.print(" ||");
    lcd.print(values[2]);
    delay(10);
  }
  myFile.close();

  if(okay ==0){
    lcd.clear();
    moving_print("The number of entries is: ");
    lcd.print(num_entries);
  }
  okay =1;
  */
}

/************************ Used for Acessing the values of Input.csv in the SD Card************************************/
unsigned long acquire(void)
{
  char strValue[6];
  char myChar;
  int j =0;
  int index =0;

  while (myFile.available()) {
      myChar= myFile.read();
      if(isDigit(myChar)){
        strValue[index++] = myChar;
      }else if(myChar =='\n' && skip==0){   // for first encounter of new line and to discard it
        skip++;
      }else{
        if(skip >0 && myChar == ','){
          strValue[index] = '\0';
          values[j] = atoi(strValue);
          index = 0;
          j++;
        }else{
          if(skip>0 && myChar== '\n'){
            strValue[index] = '\0';
            values[j] = atoi(strValue);
            index = 0;
            j =0;
            num_entries++;
            return (millis());
          }
        }
      } // end of else
    } // end of while loop
}


/******* Moves display to the left at 200 ms  the text appears two times********/
void moving_print(String param1)
{
 int i=0;
 int j=0;
 int maxlen = 16;
 int skipper =0;

 while( i <= param1.length())
```

```
 {
  lcd.setCursor(0,0);
  while( j< maxlen){
    if(j>=param1.length()-1){
         if((j - (param1.length()-1)) <= 1 ){
           lcd.print(" ");
         }else{
           lcd.print(param1[j-(param1.length())-1]);
         }
    }else{
      lcd.print(param1[j]);
    }
    j++;
  }
  delay(300);
  i++;
  j= i;
  maxlen++;
 }
 return;
}

#else  // SD_SPI_CONFIGURATION >= 3
#error SD_SPI_CONFIGURATION must be set to 3 in SdFat/SdFatConfig.h
#endif  //SD_SPI_CONFIGURATION >= 3
```

User manual

1. Create a .csv file with a file name of Input.csv
2. Input the time, real and reactive power in the following manner shown below.
3. Save te Input.csv file in an SD card.
4. Insert the SD Card to the SD card module
5. Connect the bidirectional switch to the load and the three phase power supply.
6. Connect the power supply of the arduino and other auxillary power supplies.
7. Run the simulation by turning on the 3 phase 230V power supply.

time-A(s)	Real-A(W)	Reactive-A(Var)	Real-B(W)	Reactive-B(Var)	Real-C(W)	Reactive-C(Var)
10	84	34	0	45	84	34
5	65	45	43	100	65	45
6	77	30	46	78	77	30
77	87	34	87	12	87	34
25	65	78	44	22	65	78
65	0	45	77	30	0	45
65	43	100	87	34	43	100
123	46	78	65	78	46	78

Figure 6.8: Sample

Bibliography

[1] A. M. Al-Kandari A. K. Al-Othman, Nabil A. Ahmed and H. K. Ebraheem. AC chopper voltage controller-fed single phase inductor employing symmetrical PWM control technique. *Electrical Power System research*, 55, july 2000.

[2] A. M. Al-Kandari A. K. Al-Othman, Nabil A. Ahmed and H. K. Ebraheem. Selective Harmonic Elimination of PWM AC/AC Voltage Controller Using Hybrid RGAPS Approach. *International Journal of Electrical, Robotics, Electronics and Communications Engineering*, 1, 2007.

[3] Jong-Hyun Kim ; Jee-Hoon Jung ; Myung-Hyo Ryu ; Ju-Won Baek. A simple dimmer using a MOSFET for AC driven lamp. In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, pages 2872 – 2876, November 2011.

[4] V.K. Mehta ; M.H. Ayalani ; Y.B. Bhavsar. Pulse width modulated single phase AC chopper for passive R-L load. In *2013 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–6, 2013.

[5] N. Burany. Safe Control of Four-Quadrant Switches. In *IEEE IAS Conf., California*, pages 1190–1194, 1989.

[6] R.D DelMundo. *EEE 103 Introduction to Power Systems Notes 1.* upeeei.

[7] Murat Bilgic Deniz Yildirim. PWM AC Chopper of Single Phase Induction Motor Variable-Speed Fan Application. In *IEEE Trans. Annual Industrial Electronics conference*, pages 1337 – 1342, November 2008.

[8] D.Hart. *Power Electronics.* McGraw-Hill, 2011.

[9] Mohammed K. Edan. Effect of Unbalance Voltage on the Operation and Performance of a Three Phase Distribution Transformers. *Journal of Babylon University, Engineering Sciences*, 21, 2013.

[10] Pw. Wheeler ; D.A. Grant. A low loss matrix converter for AC variable-speed drives. In *Fifth European Conference on Power Electronics and Applications*, pages 27–32, September 1993.

[11] Yasuda Y. ; Yokoyama A. ; Tada Y. ; Okamoto H. Study on Low-Cost and Easy-Use Real-Time, Digital Power System Simulator in MATLAB/SIMULINK Environment. In *International Conference on Power System Technology, 2000. Proceedings. PowerCon 2000.*, pages 921 – 926, December 2000.

[12] Guan-Chyun Hsieh and Jung-Chien Li. Design and implementation of an AC active load simulator circuit. *IEEE Transactions on Aerospace and Electronic Systems*, 29:157 – 165, January 1993.

[13] L. Hua Jin ; Goos, G. ; Lopes. An efficient switched-reactor-based static VAr compensatort. *IEEE Transactions on Industry Applications*, 30:998 – 1005, 1994.

[14] B. Nelson J. Banks, J. Carson and D. Nicol. *Discrete-Event System Simulation.* Prentice Hall, 2001.

[15] M. SARMA J. GLOVER and T. J. OVERBYE. *Power System Analysis and Design -Fifth edition.* Cengage Learning, 2012.

[16] Marco Matteini. *Control Techniques for Matrix Converter Adjustable Speed Drives.* PhD thesis, Department of Electrical Engineering, University of Bologna, 2001.