1.5 kW PWM Bipolar Inverter

By

Michael Newbry and Percy Vigo

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2009

TABLE OF CONTENTS

List of Tables and Figures

Acknowledgments

Mike:

I would like to thank Professor Taufik for taking on the role of my senior project advisor, teaching me power electronics, and nudging us along when we got stuck. I would also like to thank Jonathan Paolucci for his suggestions here and there. Software was slightly frustrating and would have taken 10 times as long without the help of Tom Hickcok. I would like to thank my partner Percy Vigo for sticking with me and keeping the team optimistic when success seemed grim. Lastly, none of this would have been possible without my parent's support and investment in my college career.

Percy:

I would like to give thanks to Professor Taufik for sparking my interest in power electronics and for being a great advisor throughout the course of this senior project. I would also like to give thanks to Jonathan Paolucci for his help with some aspects of the project. I would like to thank my partner Mike Newbry for being extra picky to make sure that we make an outstanding project. Finally, I would like to thank my family for their support throughout my entire college career.

Abstract

This report focuses on the design, construction, and testing of a PWM inverter capable of converting 1.5 kW. The purpose for the inverter is to convert DC solar energy to power an air conditioner. The inverter uses a bipolar control scheme which comes from a PIC microcontroller. This report contains the advantages and disadvantages of all the PWM control schemes. Furthermore, the testing shows bipolar inverter characteristics and a sample use of filters to obtain sinusoidal waveforms. The design of the circuit makes the inverter easy and safe to use and test.

I. Introduction

Power electronics plays a crucial role for integrating present day renewable energy sources into our everyday lives. Renewable energy sources, such as solar and wind, get converted into a direct current (DC) via photovoltaic cells and wind turbines. However, the energy we receive at our homes comes from the utility companies as an alternating current (AC). So in order to use the renewable energies stated above, one needs a circuit called an inverter which converts DC into AC. Our senior project consists of the design and assembly of an inverter which will take the DC power from solar panels and produce the AC power needed for a household air conditioning unit. Air conditioner units consume a lot of power and are one of the main causes for brown outs during summer time. People run their air conditioners mostly around noon until sun down. This causes the utility companies to supply more power during these hours. Since solar panels convert the sun's energy to electrical energy during the same time as people run their air conditioners, our inverter design could help balance the load for the utility companies which would result in less brown outs.

As far as this senior project goes, we will implement a pulse width modulated (PWM) inverter to convert a fixed DC voltage into a 120 Vrms 60 Hz AC voltage which will mimic a sine wave. The inverter uses insulated gate bipolar transistor (IGBT) switches which can handle the large amount of current and voltage that air conditioners require. As a senior project our inverter is subject to certain design restrictions that we could improve in an industrial environment. The first and main cut back will be an open loop system as opposed to a closed loop system. This means we will not have a feedback circuit at the output which will keep our inverters frequency and voltage at the exact values. However, we will use the PIC16F88 microcontroller to digitally control the circuit which is more reliable and resilient to change than analog components. The frequency will be pretty close to 60 Hz, but any change in the input DC voltage will result in an increase in the RMS output voltage. A solution to this problem could be solved by designing the solar panels to produce a larger DC voltage than required and sense the output voltage using the microcontroller to compute the RMS voltage and adjust the amplitude modulation accordingly.

II. Background

Power inverters are circuits used to convert Direct Current (DC) to Alternating Current (AC). The input of the inverter may come from a DC source or from rectified AC input. There are two main categories for switch mode voltage source inverters: square wave and pulse width modulated (PWM). The basic configuration for both circuits looks like Figure 1, but the difference comes from how each switch gets turned on. Square wave inverters are the simplest to implement. The simplicity of the square wave inverter comes along with the disadvantage of harmonics close to the fundamental frequency. PWM inverters function by comparing a sinusoidal control signal at the desired output frequency with a triangular carrier signal at switching the frequency. The harmonics of PWM inverters are located at multiples of the carrier signal frequency which is typically in the kHz range. This simply means the output waveform of PWM appears more sinusoidal than a square wave inverter. Also, higher frequency harmonics are easier to filter than harmonics near the fundamental frequency.

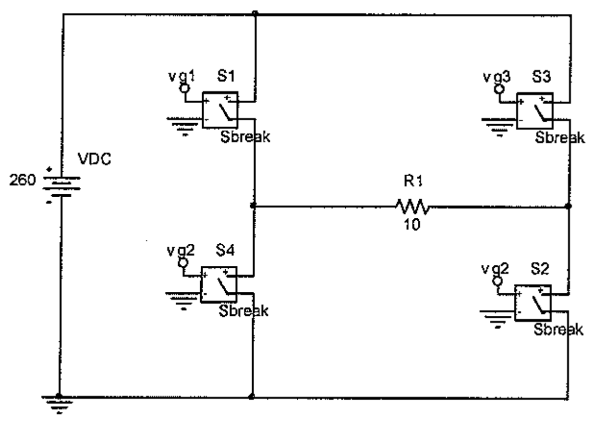


Figure 1: Basic Inverter Circuit with Ideal Switches

There are two different types of PWM inverters: bipolar and unipolar. Looking at Figure 1, the output voltage occurs across R1. In a bipolar PWM inverter, the output switches between +Vdc and –Vdc. The switches s1 and s2 turn on, and switches s3 and s4 turn off, when the amplitude of the sinusoidal control signal is greater than the amplitude of the triangle carrier signal. For a unipolar PWM inverter, the output can switch from +Vdc, 0, and –Vdc. There are 2 different ways to implement a unipolar control circuit. Type 1 has two sinusoidal control signals that are 180 degrees out of phase which both get compared to the same carrier signal. The switches turn on for the following cases:

S1: Vsin > Vtri
S2: -Vsin < Vtri
S3: -Vsin > Vtri
S4: Vsin < Vtri

An easier approach which we call type 2, has gates 1 and 4 turn on the same as above, but turns on S2 when Vsin > 0, and S3 when Vsin < 0. However, type 2 brings back the low frequency harmonics.

In order to decide which circuit to implement, we ran PSpice simulations for bipolar, and type 1 and 2 unipolar circuits. We found that unipolar type 1 has the best THD and would be the easiest to filter since the harmonics occur at multiples of 2*f_c (carrier frequency). However, PIC microcontrollers are not fast enough to implement the switch control signals. Since, low frequency harmonics are harder to filter; we will implement a bipolar PWM inverter instead of the unipolar type 2. Bipolar does have the advantage of the output RMS voltage being the closest to the input DC voltage. Figure 2 shows the output voltage as a function of the input voltage for each PWM inverter.

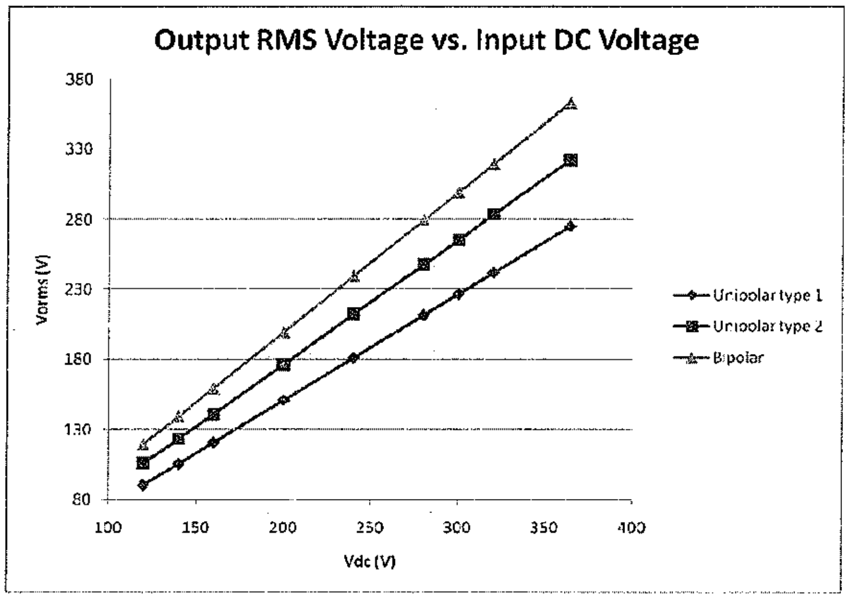


Figure 2: Output RMS Voltage for Different Input Voltages for Each PWM Inverter

Figure 2 shows the RMS output voltage of PWM inverters has a linear relationship with the DC voltage. PWM inverters have the added advantage that the output RMS voltage also shares a linear relationship with the amplitude modulation. The amplitude modulation is defined as the amplitude of the control signal over the carrier signal. When designing the final circuit, we can vary both the input DC and the amplitude modulation to obtain the correct output voltage. Since solar panels come rated at different voltages for the same amount of power, the amplitude modulation could be adjusted to produce the correct RMS for different systems.

III. Requirements

The requirements for our inverter go as follows:

- Output Voltage Equal to 120 Vrms
- Fixed Output Frequency of 60 Hz
- Capable of Supplying 1.5 kW
- Maximum Load Current of 12 A
- Input DC Voltage Between 120 V-130 V
- Efficiency at Full Load Current Greater than 90%
- The Circuit Should Fit in a 7" x 5" x 3" Project Enclosure
- Through Hole Soldering of the Integrated Circuits
- Digitally Controlled

The output voltage and current requirements come from standard 12,000 BTU window air conditioners. Most 12,000 BTU window air conditioners run at 120 Vrms, but have different efficiencies so the current range runs from 6 A-12 A. Since we are not sure about the voltage drop across our IGBT's the input voltage to the inverter could range from 120 V-130 V. A DC buck converter from the solar panels to the inverter could be used to maintain the DC voltage required to produce 120 Vrms since the DC voltage shares a linear relationship with the output RMS voltage. This means the solar array will need to connect the panels in series to obtain a DC voltage greater than 120V. In order for the inverter to transfer energy efficiently, we imposed the 90% minimum efficiency limit.

IV. Design

The design of our PWM inverter breaks up into software and hardware. The software produces the control signals from the PIC16F88 microcontroller, and the hardware manipulates those signals to drive the IGBT switches. The hardware also contains the power unit for the microcontroller and other ICs.

a. Software

We chose to use a PIC microcontroller because they are easy to learn and the software to program them with is free online. The internet has a few compilers to choose from, but we chose to use mikroC because it comes with example code of how to use various PIC features. MicroC functions as follows: the user chooses their PIC from a list of multiple PIC's the software supports, choose the desired clock frequency and a few start up features, write the program in C programming language, then microC compiles the C code to an assembly language .hex file. MicroC has a built in code checker which identifies any errors in the code. A programmer is used to get the .hex file onto the microcontroller. We did a search for a programmer on the site we purchased our PIC from and came up with the PGM-0008 serial port programmer. This cost less than the other programmers on the site, but it comes with the drawback of requiring a computer to have a serial port. Serial ports are becoming less popular for desktops and do not exist on most laptop, not to mention we had to search multiple stores to find a serial to serial cable to interface between a desktop and the programmer. However, the serial port programmer was easy to use. It uses a program called icprog where a user selects his/her PIC, opens up the .hex file, and then hits the program button. Moving the PIC between the programmer and the breadboard to test the code was the most taxing part of the programming process. One needs to remove the PIC carefully because it has a snug fit to the programmer and the pins can bend or break. We lost an output pin during this process, but luckily we did not need it.

From a recommendation from a friend who had experience with PIC microcontrollers, we decided to use the PIC16F88. Consulting the pin diagram shown in Figure 3, the PIC has 16 inputs/outputs that function as either general IOs, a PWM port, a capture/compare port, a 10 bit analog to digital converter input, a USART, as well as a few other functions. The other 2 pins are the power and ground pins. The PIC operates on voltages from 2.0 V to 5.5 V so we chose 5 V. The PIC16F88 has 7168 bytes of program memory which was more than enough to store our code. The microcontroller has an internal oscillator with programmable frequencies up to 8 MHz, but it can work on frequencies up to 20 MHz with an external oscillator source. The microcontroller has three timer modules that function in similar manners. The oscillator goes through a user determined clock prescaler and postscaler to slow down the clock rate, then it increments a register until it either matches a user selected value or overflows. At that point it will generate an interrupt if they have been enabled. This process became important when deciding how to obtain a PWM waveform.

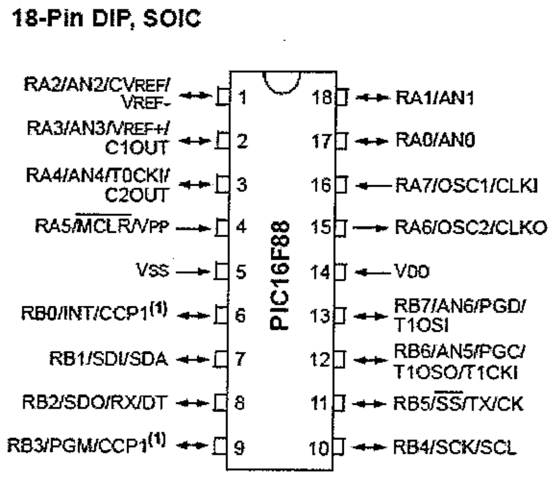


Figure 3: Pin Diagram for the PIC16F88 [8]

We thought of two methods to generate the PWM signals we required. The first method inovlved generating a sinusoidal wave and a triangle wave and comparing the two all in software. Figure 4 below shows a PSpice waveforms of this scheme. When the sine wave is greater than the triangle wave, output pin 1 would be set high while output pin 2 set low, then when the sine wave is less than the triangle wave, it would clear pin 1 while setting pin 2.

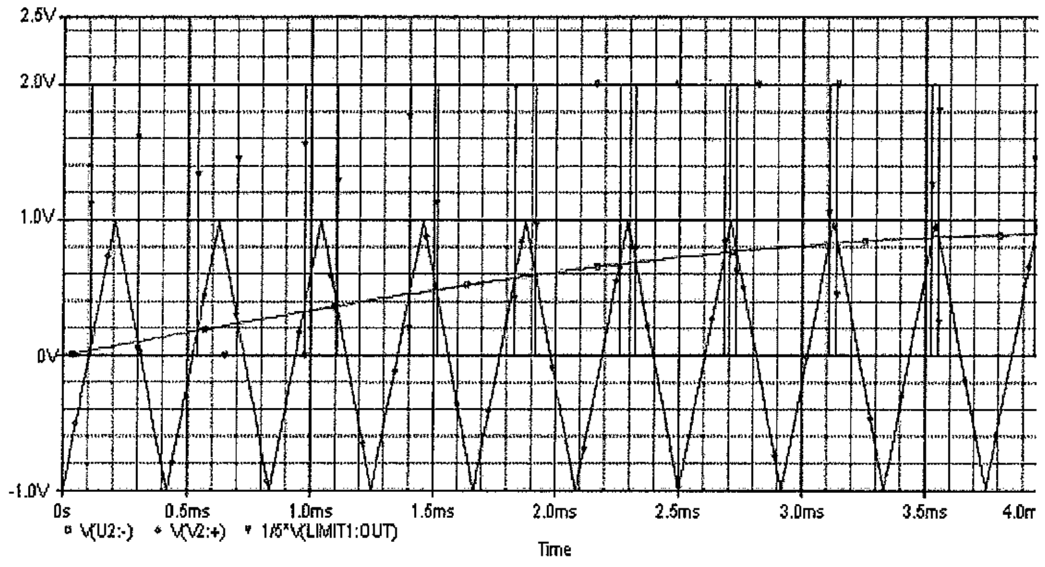


Figure 4: PSpice Graph of Generating the PWM Control Signals

We knew we wanted a 2.4 kHz switching frequency which is the frequency of the triangle wave. The PWM pattern would then repeat every 60 Hz which is the sine wave's frequency. To generate the triangle wave we increment a variable by a set amount at a specific time that is set by one of the timer module's interrupt until it reaches the maximum value, then we decrement the variable by the same amount and time until it reaches the minimum when the pattern repeats. The sine wave is generated by a look up table in which the magnitude of the values stored in an array correspond with the amplitude of the sine wave at a specific time. Following this approach, a negative sine wave look up table could be implemented and compared to the same triangle wave to obtain the other 2 control signals required for a unipolar inverter.

The problem with this approach came down to the oscillator speed. We needed 10 samples in order to obtain an adequate triangle wave which means we would have to generate interrupts every 24 kHz. Since the microcontroller executes instructions at 8 MHz, we would only have time to execute 333 instructions between interrupts which was not enough instructions to generate the triangle wave and compare it to a look up table. This came as a shock at first because the C code for this method had less than 50 lines, but the assembly language requires many instructions to execute basic C functions like "if" statements.

The second method which we use in our final circuit used the PWM functions of the PIC1688. The actual code is shown in appendix C. This simpler method only used three of the pins of the microcontroller – Vss, Vdd, and the CCP1 pin. Regarding Figure 4, all the PWM wave does is increase the duty cycle as the sine wave approaches a maximum and decrease as the sine wave approaches a minimum. The PWM period equals the switching frequency. Consulting the datasheet, the PWM period = (PR2+1)*4*Tosc*(TMR2 prescale value) [8]. With an 8 MHz clock, a prescale value of 4, and PR2 set to 207, we obtain a period of 2403.9 Hz which is the closest we can get to 2.4 kHz. At 2.4 kHz, the sine wave lookup table pattern needs to repeat 40 values in order to mimic a sine wave of 60 Hz (2.4 kHz/40 = 60 Hz). Since we are implementing bipolar signals, we chose our duty cycles to equal 50% at t=0, 95% at t=T/4, and 5% at t=3T/4. The equation for the duty cycle is (CCPR1L:CCP1CON<5:4>)*Tosc*TMR2 prescale [8]. We have to load a value of 416 for a 50% duty cycle, 790 for 95%, and 42 for 5%. Using microsoft excel we found the 40 values to load using the equation 416+374*SIN(2*pi()*n/40) where n is an integer between 0 and 40. Loading this value is more complicated because the register involved has 10 bits, the 8 most significant bits in register CCPR1L and the 2 least significant bits in CCP1CON<5:4> . A general trend for loading CCPR1L mimics the 10bit equation so we used microsoft excel again with the equation 104+93*SIN(2*pi()*n/40). Then we had to make adjustments entry per entry to adjust the 2 LSBs to obtain the correct 10 bit entry for our desired duty cycle. With the sine-table look ups figured out, all the software has to do is load the current index which represents the duty cycle every 2.4 kHz then increment to the next index. Once the index reaches the end at 40, it returns to the start of the array to begin the mock sinewave over again.

b. Hardware

Insulated Gate Bipolar Transistor (IGBT) Modules

The first and most important items for the inverter circuit are the IGBT modules. The inverter circuit requires 4 IGBTs configured as shown in Figure 5. Remembering that IGBTs Z1 and Z2 turn on at the same time as Z3 and Z4 turn off, this leaves +Vdc across the switches when they are off. This means the switches have to work on voltages greater than Vdc. The next sizing requirment is the amount of current that flows through the switches which we specified as 12 A above. Since this is a higher power device heat sinking becomes an issue for the IGBTs. When searching for IGBTs, we had the choice of buying discrete components or half-bridge modules. Since the modules came in boxes that functioned as heat sinks we decided to get these. The modules are much more expensive, but lucky for us, our faculty advisor Professor Taufik had some extra modules that he let us use.

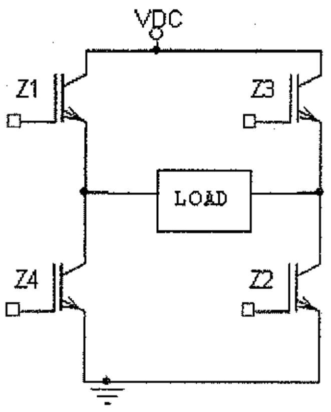


Figure 5: PSpice Schematic of an Inverter with IGBTs

The ID2260A2 Dual IGBT Modules are rated at 600V and 25 A which passes our requirements of 160 V and 12 A [4]. The module with its circuit configuration is shown below in Figure 6. The screws contain the C1, E2, and C2/E1 connections which get connected to Vdc, ground, and to the load respectively. The pins on the right hand side contact the gates and the emitters of both IGBTs so that is where the control signals get routed. According to the data sheet, the gate-emitter threshold voltage ranges from 3 V to 6 V and the maximum gate-emitter voltage equals 20 V [4].

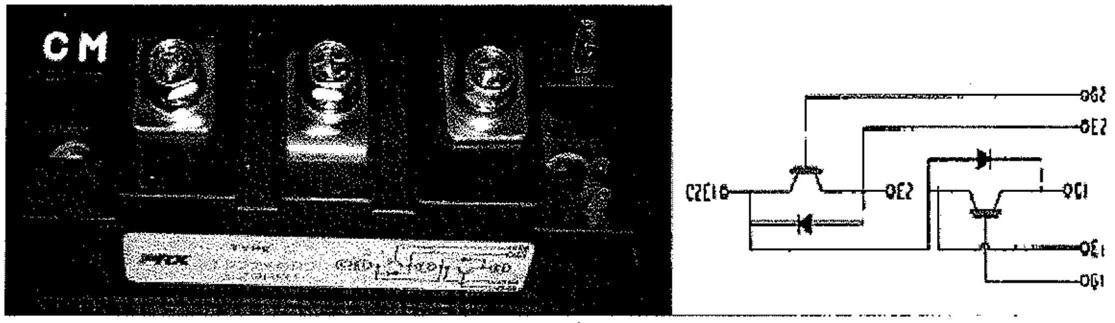


Figure 6: IGBT Module with the Corresponding Circuit Diagram

The diode that connects between the emitters and collectors of each IGBT is a fast recovery free wheel diode. This provides protection to the IGBTs for inductive loads in case of the current spikes. The diodes reverse recovery time is 150 ns, and it can handle currents up to 25 A. As far as heat dissipation goes, the IGBTs have a maximum thermal resistance of 0.8 ^{O}C/W, the diodes have 2 ^{O}C/W, and each half module has 0.15 ^{O}C/W [4]. The datasheet doesn't specify the on resistance of the IGBTs, but at full load of 12 A and an over estimate for the on resistance of 0.5 Ω would leave a power dissipation for the module of 2* (12A)^2 * 0.5Ω *0.15 ^{O}C/W = 21.6^{O}C or 71^{O}F.

<u>IGBT Drivers</u>

Drivers take in the logic signals and convert them to an ample voltage to turn on the IGBTs. Drivers also isolate the control signal circuits from the high voltage IGBTs. There are various IGBT drivers, but since an inverter has a high side switch, we found a high-voltage high and low side driver. The voltage rating, logic input voltage, and the drivers supply voltage are important when selecting a proper driver.

At first we attempted to use some spare IR2110 high and low side drivers to drive our IGBTs, but we realised this driver required a 9.5 V logic input and a 10 V supply. This was problematic because our microcontroller outputs a 5 V signal. After many attempts to level shift the signal to 10 V using CMOS inverters and OP AMPs we decided to get a high and low side driver chip that accepts smaller logic inputs. We came up with the L6388E high voltage high and low side driver. This accepts logic inputs from 3.3 V, 5 V, up to 15 V; it has a high voltage rating of 600 V, and requires greater than 10 V for the supply voltage. The L6388E has interlocking circuitry that prevents the high and low voltage outputs from turning on at the same time as well as 220 ns-420 ns of dead time which is the amount of time after one output turns off from the time the other can turn on [7]. All these features come in a simple 8 pin DIP with a typical configuration shown below in Figure 7.

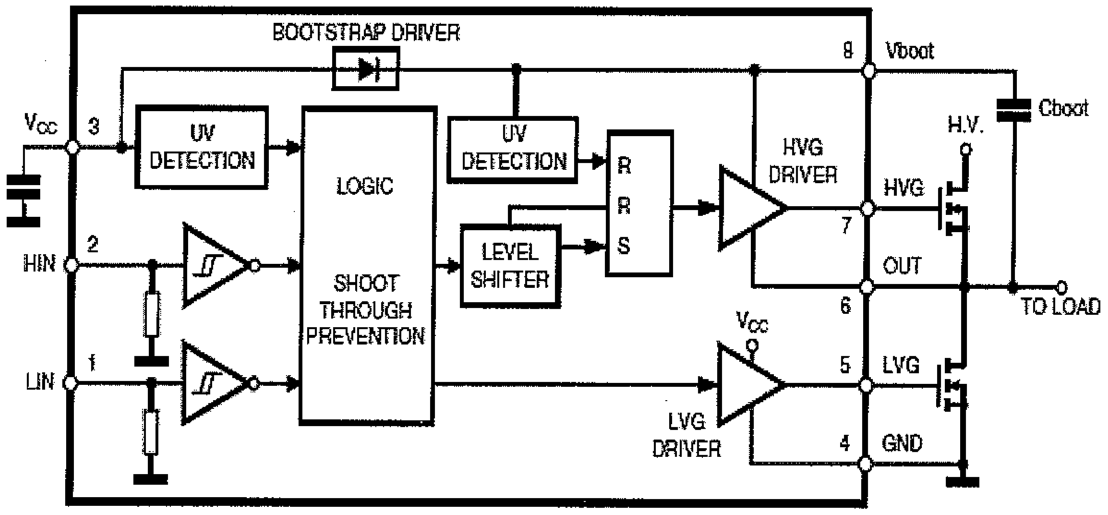


Figure 7: H-Bridge Configuration for the L6388E High and Low Side Driver [7]

This driver does not require that much external circuitry due to the internal bootstrap diode. The Cboot capacitor gets charged to +Vcc when the low side IGBT conducts, then when the high side IGBT conducts, the HVG driver can send a signal that is +Vcc greater than the emitter voltage which will keep the high side conducting. The value of Cboot needs to be much larger than Cext which is defined as Qgate/Vgate which approximately equals 8 nF (80 nC/10 V). We chose a 1 μF capactior for our bootstrap capacitor which is much larger than required but carried over from our uses of the IR2110 drivers. The datasheet did not contain any information on the power supply capacitor so we stuck in a 0.1 μF capacitor.

<u>NAND Gates</u>

The NAND gates serve three useful purposes for our circuit. It provides the compliment of our PWM signal, it provides more dead time, and it can supply more current to the drivers than the microcontroller. We used the CD4011BE QUAD 2-Input NAND gate which is shown below in Figure 8. A unique feature of this chip is it can run off of 5 V, 10 V, and 15 V rails. We initially purchased this chip as an additional attempt to step up our 5 V signals to 10 V when we were using the IR2110 drivers, but ended up just supplying this chip with 5 V when we got the L6388E drivers. Each NAND gate has a propagation delay between 60 ns and 250 ns and can source up to 1 mA [1]. The IGBTs have a maximum rise time of 700 ns so with this addition to the 220 ns-420ns of the drivers keeps the IGBTs from turning on at the same time. We used three of the four NAND gates in our design – two compliment the microcontroller signal twice returning the same signal but allowing the NAND to supply the current to the drivers, and the third produces the compliment of the microcontroller signal which drives the other 2 IGBTs.

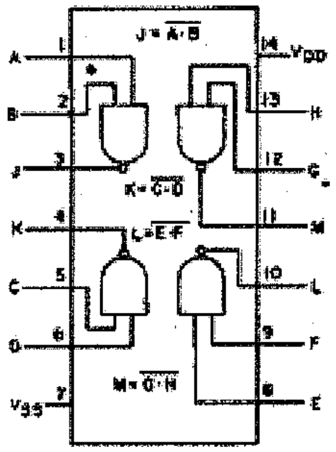


Figure 8: Schematic of the CD4011BE Quad 2-Input NAND Gate [1]

Power Circuit

The power circuit needed two voltage levels – 5 V for the microcontroller and NAND gate, and >10 V for the driver supply voltage. We used a linear voltage regulator circuit combined with a 5 V voltage regulator as below in Figure 9. The resistor that ties the collector of the BJT to the base keeps the BJT operating in the forward region. The 1N759A 12 V zener diode keeps the base of the transistor at 12 V as long as Vdc is greater than 18 V. The resistor value was chosen to allow enough current for the zener diode to conduct when testing at 20 V while at the same time not exceeding the ¼ W power dissipation of the resistor at the nominal input voltage of 120 V. In fact, it dissipates 0.23 W ((120-12)^2/51000) at the nominal input voltage. This pushes the resistor limit, but when we used 100 kΩ resistor the zener diode did not turn on all the way. The FJP554 is a high voltage BJT rated at a collector to emitter voltage of 400 V [3]. The base to emitter turn on voltage is slightly less than a volt which sets the driver supply voltage from 11-11.3 V. This discrete BJT tends to get warm when operating at 120 V since the voltage drop across it is 109 V and the ICs it powers consume around 15 mA-25 mA so it dissipates 1.6-2.7 W. To help with the heat dissipation we mounted a heat sink to the BJT. The last component of the power unit is the 7805A 5 V voltage regulator. The voltage regulator maintains an output voltage of 5 V for input voltages greater than 5 V and less than 35 V. The datasheet specifies a maximum line regulation of 50 mV for input voltages up to 12 V, and a load regulation of 50 mV for currents up to 750 mA which is significantly more than the microcontroller and NAND gates require [5].

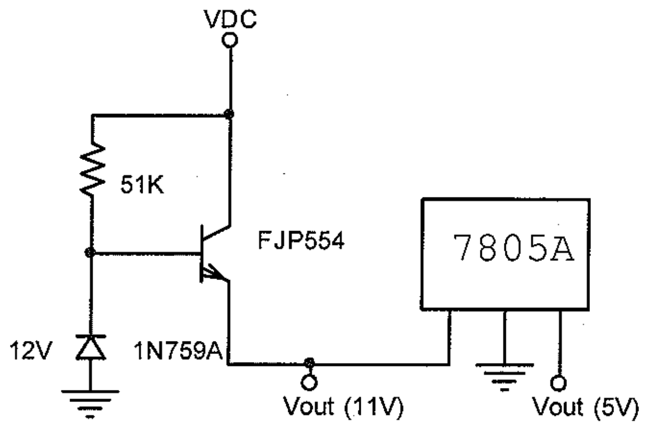


Figure 9: 11 V and 5 V Power Supply Circuit

V. Development and Construction

We first constructed our circuit using a bread board so that we could test it before we soldered everything together. This preliminary testing was quite tedius because we had to use a lot of long wires to make connections. When connecting the entire circuit, we needed six banana to spade cables for the load, Vdc, and ground connections, eight banana to grabber cables with alligator clips on the ends for the gate and emitter connections, two more banana to grabber cables to power the bread board, and lastly scope probes to check the signals. On the breadboard itself, there were wires running all over the place routing signals between ICs.This initial chaos lasted a few weeks while we tried fervently to get the IR2110 drivers to work with our 5 V signals. Once we obtained the L6388E drivers we spent two days testing them with the same breadboard set up until we were sure the circuit was working properly.

We soldered the components to a grid style PC board in the same order that we had them lined out on the breadboard. We used IC sockets for each of our ICs incase we needed to replace one or we needed to update the software on the PIC. Also, these IC sockets keep the IC from overheating during the soldering process. The bulk of the soldering involved routing 22 gauge wires from IC to IC. When connecting the ICs, we kept most the wires pressed down against the board and their lengths as small as possible. Once we got more and more connections we had to arch some of the wires so when mounting the PC board, we bought ¾ " PCB standoffs to lift it up so the wires don't touch the bottom of the case. Appendix B shows the layout of the overall circuit, and Figure 10 shows a picture taken of the completed circuit.

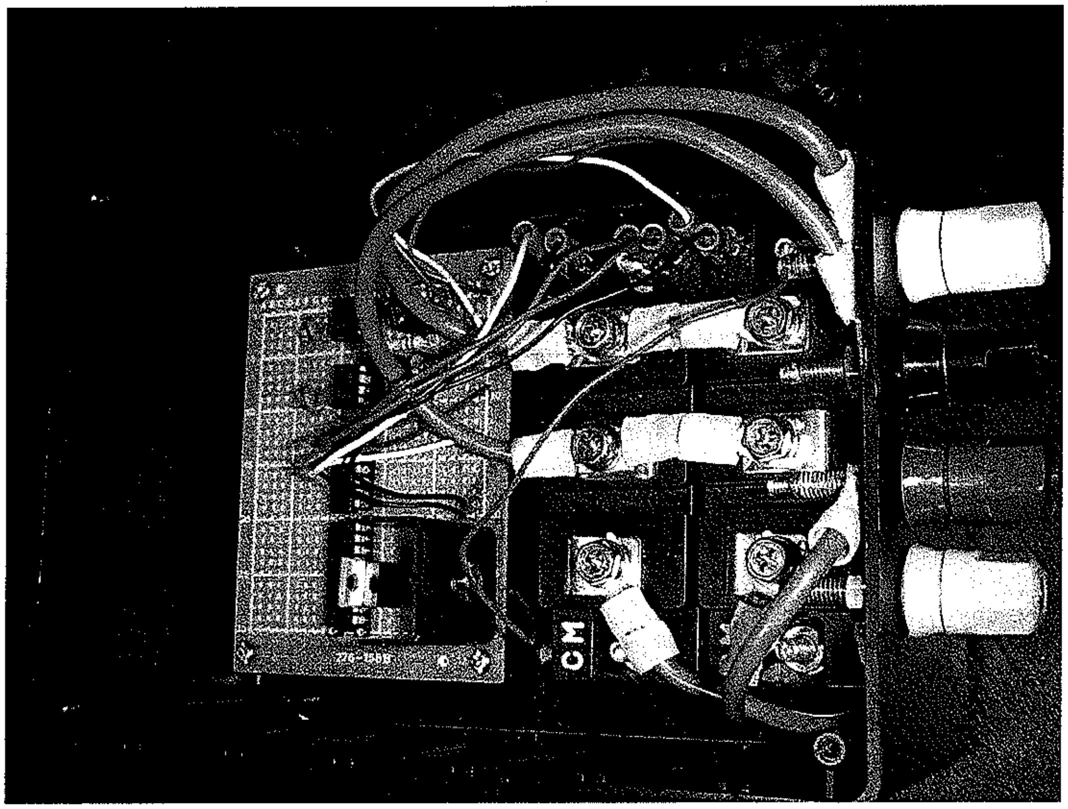


Figure 10: 1.5kW Bipolar PWM Complete Circuit Photograph

Figure 10 shows the top view of our project enclosure which we purchased from radioshack. We layed out the PC board, the IGBT modules, and the five way binding posts and marked the locations for the mounting screws. The holes were drilled for the various screw sizes, and afterwards everything was mounted in place with nuts. Looking at Figure 10, there are two sizes of wire used in the project. The thin wires are 22 gauge wires that run from the drivers to the gates and emitters.

The thicker wires are 12 gauge wires that run from the input and output terminals and between the IGBTs. The gauge of the wire determines how much current can flow through it so the control wires can be much smaller than the wires attached to the load and inputs. There is one stray 22 gauge wire that connects the input terminal to the linear voltage regulator, but since the ICs only consume mA of current, it is the correct size. Each end of the 12 gauge wire strips end with spade connections which get screwed into place to the metal contacts of the IGBT modules. To make the spade connections we stripped about a half inch of the wire insulation on each end, twisted the copper wires, and crimped them into place. The crimping took a lot of force to get the wire to stay in place. The 22 gauge wire got one end soldered to the PC board and the other end crimped to the female end of quick disconnects. These disconnects make contact with IGBT modules by sliding over the pins that correspond to the gate and emitter contacts. The disconnects weren't made for that purpose, but they fit snuggly to the pins which makes a good contact.

The five way binding posts are titled so because they have five places to make a connection. On the inside of the box, the spade end of the 12 gauge wire gets held in place between two nuts. The 2nd type of connection on the inside is solder at the ends of the screws which we utilized for the 22 gauge linear voltage regulator connection. The outside can connect with banana cables, spade cables, and as well as grabbers since there is a hole in which one can stick the end of a wire in. After everything was complete as shown in Figure 10, all we needed was 4 banana to banana cables to hook up the inverter.

VI. Test Results

Our initial testing went hand in hand when working with the microcontroller. We would edit some code, swap it from the programmer to a breadboard, power it up, and observe the waveform on an oscilloscope. After a few hundred repetitions of this we obtained the waveform seen in Figure 11. This shows the PWM waveform declining from the maximum duty cycle on its approach to the minimum. We set the cursors to a width of 60 Hz and ensured that forty pulses occurred between them. Each pulse measured near 2.4 kHz, and measuring from the rising edge of the first pulse to the rising edge of the 41st, we found the duty cycle pattern repeated every 59.8 Hz. This frequency in our PWM scheme equals the frequency of the sine wave.

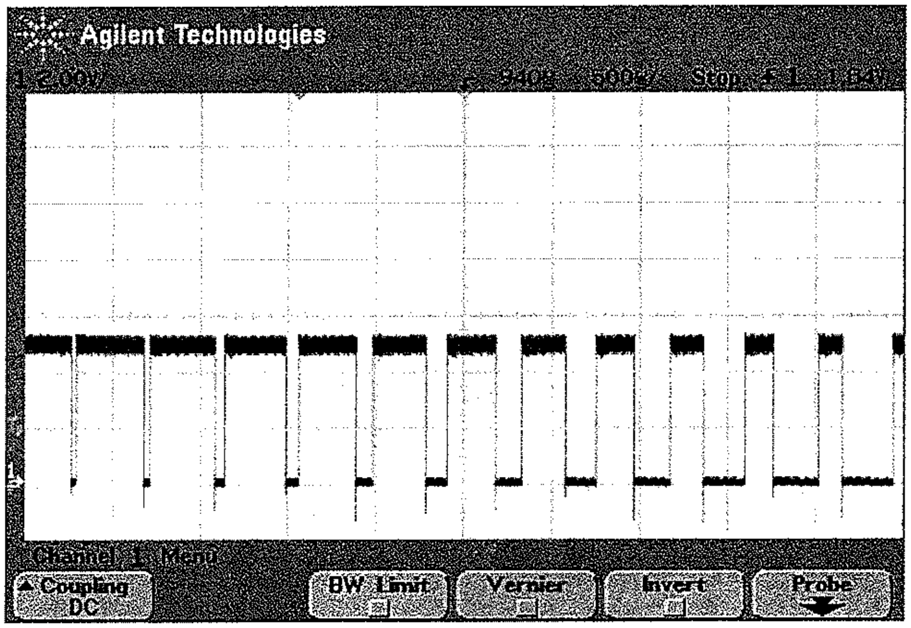


Figure 11: Oscilloscope Capture of the PIC16F88 PWM Waveform

With the PWM waveform working, we had to verify the other control signal coming from the NAND. Figure 12 shows the two control signals. Both signals have an amplitude of 5 V and the same PWM shapes. The top waveform is the NAND gate and the bottom waveform is the microcontroller. The NAND gate's output goes high when the microcontroller's output goes low which shows the NAND gate performs as expected.

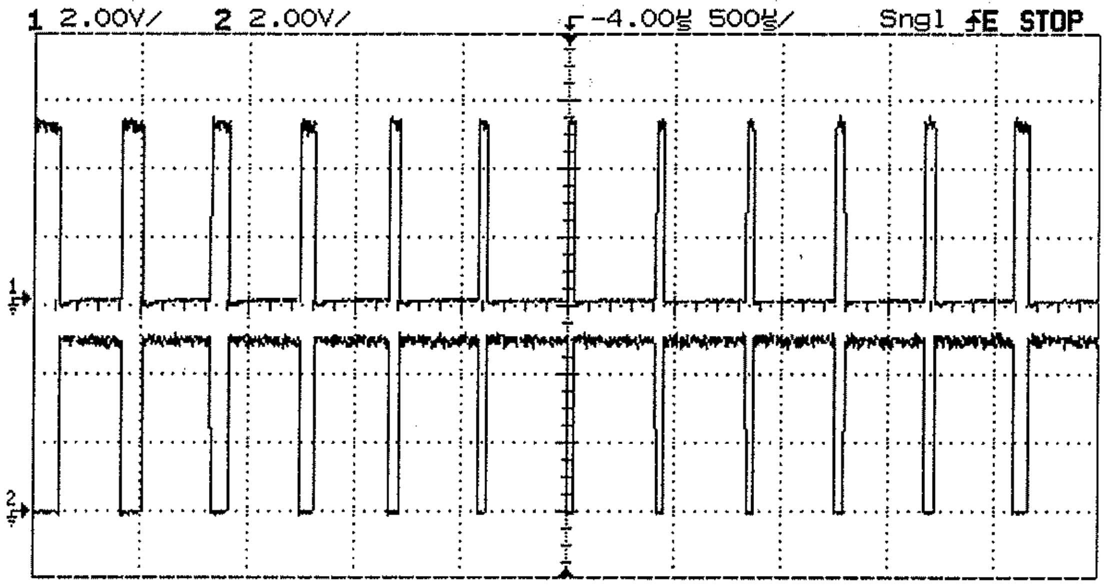


Figure 12: Oscilloscope Capture of the Control Signals: Channel 1:PIC, Channel 2: NAND

With the control signals working properly, we connected the drivers and powered up the inverter using a 20 V DC power supply at the input and a decade power resistor set at over 3 kΩ across the output terminals. Due to the size of the resistor in the linear voltage regulator circuit, the zener diode does not clamp the base of the BJT to 12 V until the input DC voltage equals 16 V. At this point, 11 V appears at the emitter of the BJT which gives the drivers enough voltage to operate. Figure 13 shows an oscilloscope capture of the bipolar output with a DC input of 20 V. Figure 13 shows the bipolar inverter working as evident from the voltage swing of

+20 V to -20 V and the duty cycle varying according to the control signals. This capture shows the voltage just before the crest of the sine wave to just before the trough of the sine wave.

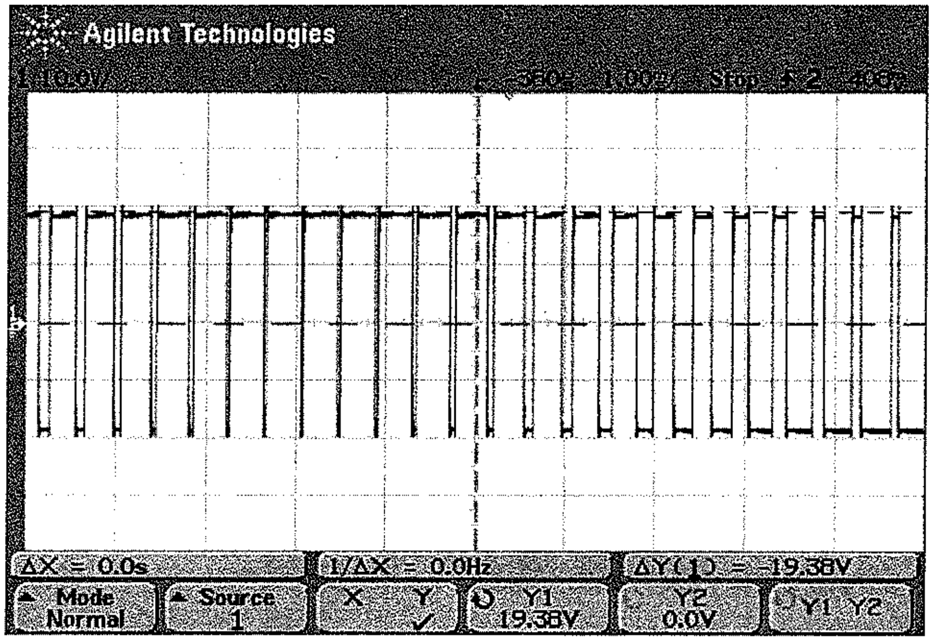


Figure 13: Oscilloscope Capture of Bipolar Inverter at 20V DC

Using a different oscilloscope and the same 20 V DC input, we used the math function to perform a finite forier transform (FFT) analysis of the output waveform. Figure 14 shows the FFT with cursors marking the 2nd and 3rd peaks. These peaks occur at 2.4 kHz and 4.8 kHz which are integer multiples of the switching frequency as expected. The first peak occurs at the fundamental frequency of 60 Hz and has a larger magnitude than the other peaks. The higher the frequency, the smaller the peaks become. This is again consistent with the operation of bipolar PWM inverters.

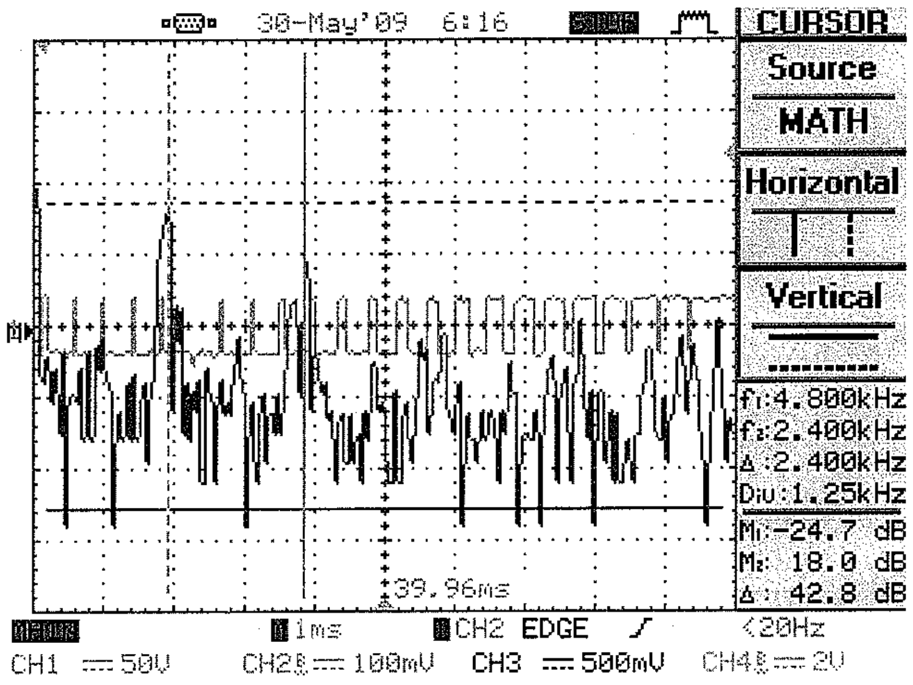


Figure 14: Oscilloscope Capture of the FFT for the Output

With the operation of the inverter confirmed, it was time to test in the high power realm. We moved to the electric machinery room that has two 125 V nominal DC generators. We connected the DC input to a 120 Ω, 3 A potentiometer so we could vary the input while recording the RMS output. We used digital multimeters to measure the voltages while we used a Fluke Scopemeter to observe the output wave function. With a large resistive load to emulate an open circuit, we adjusted the input voltage in intervals of 20 V and the data is shown in Table I. Since the output is either +Vdc or –Vdc, the square of the bipolar waveform looks the same as the square of the DC voltage so the RMS voltage equals the DC voltage as shown in

Table I. The table shows that the higher the voltage, the closer the output RMS voltage gets to the magnitude of the DC input voltage.

Table I: Output RMS Voltage at Different DC Inputs

Vin (V)	Vorms (V)
40	35.7
60	58.8
80	79.5
100	99.6
120	120

Now at the full DC voltage, we removed the potentiometer, and added multimeters to measure the DC input current and the output RMS current. Using the decade power resistor we decreased the size of the resistance while recording the multimeter data into Microsoft Excel. With this data in Excel, we determined the input power, output power, and the efficiency at various load values which is shown in Table II. An efficiency plot is shown in Figure 15. The efficiency at low load equals 72.5%, which is mostly due to the control circuitry requiring around 15 mA while the output current equals only 37 mA. As the output current increases the efficiency improves to a point then decreases due to $I^2 * R_{IGBT}$ losses.

Table II: Inverter Data for Determining Efficiency

Vin (V)	Iin (A)	Vorms (V)	RL (Ω)	Iorms (A)	Pin (W)	Pout (W)	Efficiency (%)
120	0.051	119.9	3200	0.037	6.12	4.44	72.49
116.8	1.14	114	100	1.14	133.15	129.96	97.60
120.9	1.44	118.6	80	1.44	174.10	165.89	95.29
121.5	1.61	119.2	70.5	1.603	195.62	181.16	92.61

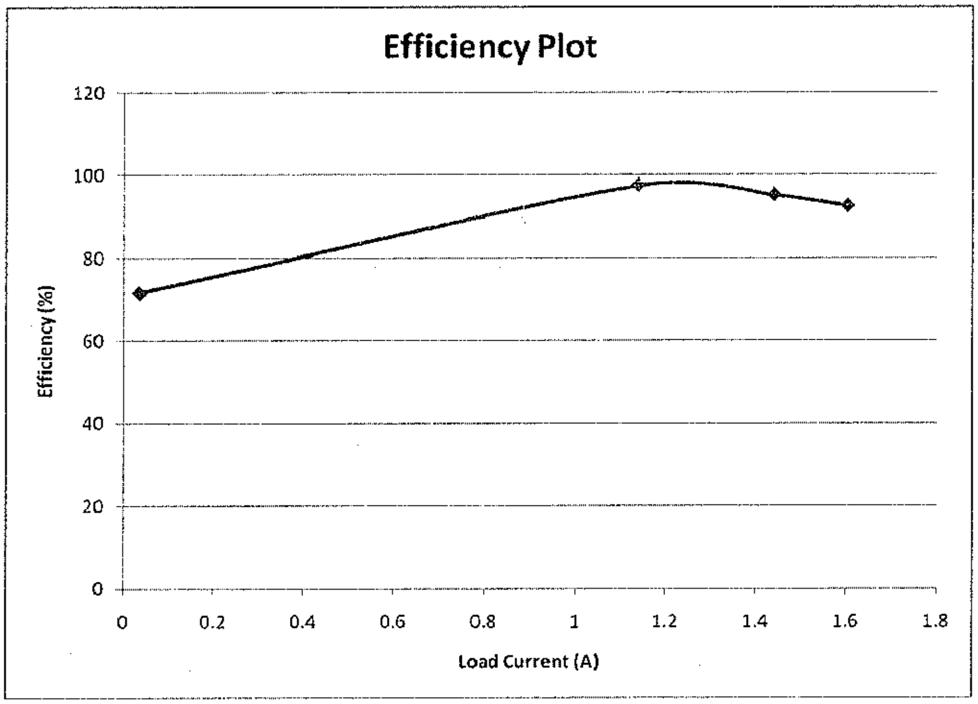


Figure 15: Efficiency Plot for Inverter

During this process we ran into problems with resistor current ratings. The power decade box had a current rating of 1.5 A so we had to use the bench resistors which have a current rating of 3.5 A. Our inverter circuit requires an open circuit output before high current can flow so we had to connect the decade resistor box set at 2 kΩ, then connect the bench resistors in parrallel. The bench resistors consist of 6-35 Ω resistors so we connected two in series to obtain a load impedance of 70.5 Ω . We were not able to obtain data at loads larger than this since the amount of current flowing through the load was too near the resistors current ratings (I = 120 V/35 Ω = 3.43 A). Though at 181 W, our circuit was not heating up and the waveform looked good on the fluke scopemeter suggesting we could increase higher if we had higher rated resistors.

Preparing for a motor, we connected an RL load consisting of the decade resistance box and a 34.44 mH inductor. While monitoring the waveform across the resistor at 120 V DC, we decreased the resistance slowly until 79.8 Ω which we observed the sinusoidal waveform shown below in Figure 16. The sinusoidal waveform has a period of 16.8 ms which is shown on the cursors of the Fluke meter. The frequency of the wave equals 59.52 Hz and the magnitude is near 100 V. The output RMS voltage and current equals 65 V and 0.81 A respectively, for an I^2*R dissipation of the resistor equal to 52.36W. This shows that adding a low pass filter to the inverter will eliminate the high frequency harmonics and produce a nice sinusoidal waveform.

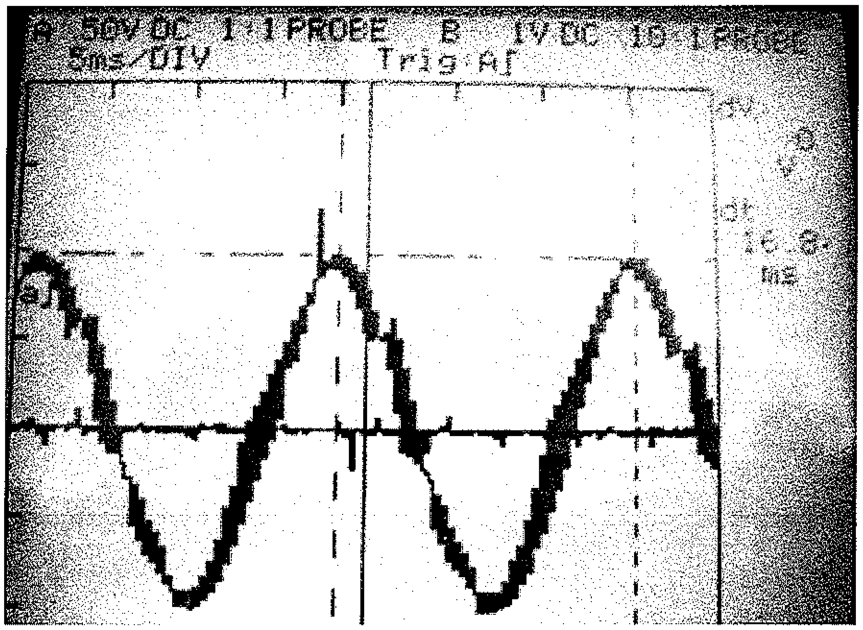


Figure 16: Filtered Inverter Output with L=34.44 mH and R=79.8 Ω

When connecting our circuit to a single phase motor we flipped on the DC voltage and heard a pop. Doing some testing we learned that the IGBTs were blown. The motors are known to have an initial current spike up to 30 A which exceeds the rating of our IGBTs which could have caused the destruction. Though, we are not entirely sure what the pop came from, our microcontroller, high voltage BJT, and voltage regulator were fried.

VII. Conclusion

We constructed an inverter using components capable of handling high voltages and high currents. We encased the circuitry in a project box that implemented five way binding posts making the circuit easy to test and safer for the user. Testing of the inverter showed we obtained the correct PWM waveform up to output RMS voltages of 120 V. We tested the inverter up to 1.6 A, but were limited by the test resistor current ratings. If we had high current resistors, the 25 A ratings of the IGBT's should allow us to test up to our required load of 12 A. The multimeters we used had current ratings of 10 A so we would need different multimeters that could handle the high current as well.

The bipolar waveform had a frequency of 59.8 Hz which is near the requirement of 60 Hz. The slight reduction in the frequency is due to the resolution in the timer variables that generate the period for the PWM port on the microcontroller. Adjusting the variable up and down near the calculated value of 2.4 kHz and measuring the period on oscilloscope we could only come within 8 Hz of our mark. FFT plots showed the the relative magnitude and frequencies of the harmonic contents. By adding an RL filter, we turned our bipolar waveform into a sinusoidal waveform with a 59.5 Hz waveform. Additional lowering of the resistance would improve this waveforms clarity, and we would be able to mark the cursors more correctly to obtain closer to the 59.8 Hz. We stopped at the point we did because the amount of current made a loud ringing in the inductor which lacked a current rating display and we did not want to break school equipment.

Since solar panels can be connected in series to boost the voltage or in parallel to increase the current either a boost converter or buck converter could be used between the panels and the inverter. I found 200 W panels that produce 20 V DC so to supply 1.5 kW a system would need eight panels. If all these were connected in series, a closed loop buck converter designed for 160 V to 120 V could supply the voltage to the inverter. During our testing, we showed the bipolar RMS output nearlly equalled the DC input. Since the voltage at normal wall sockets can vary between 115 Vrms and 120 Vrms, the fact that some voltage drops across our IGBTs is not a big deal.

There were two major downfalls of our circuit. The first was it requires the voltage supplied at no-load before high current can pass through. This has to do with the set-up time of the microcontroller and drivers. The second dissapointment was its performance with the motor load which was meant to mimic an air conditioner. Though the cause of the whole circuit destruction was unclear, higher current ratings on the IGBTs or lower surge current on the air conditioner could solve this problem. In any case, we should have implemented fuses in our circuit to protect all the hardware. The circuit needs two fuses, one rated at 25 A to protect the IGBTs, and the other rated at ¼ A to protect the control circuitry. Though the IGBTs and the high voltage BJT can support more than three times the voltage requirement, some sort of voltage protection would make the circuit more complete.

VIII. Bibliography

[1]. "CMOS NAND Gates: CD4011B, CD4012B, CD4023B Types." Texas Instruments. 11 March 2005. PDF Online < http://focus.ti.com/lit/ds/symlink/cd4011b.pdf>

[2]. Dr. Taufik. Introduction to Power Electronics 6th revision. San Luis Obispo, CA: Taufik, 2008.

[3]. "FJP5554 High Voltage Fast Switching Transistor." Fairchild Semiconductor. Oct. 2008. PDF Online < http://www.fairchildsemi.com/ds/FJ%2FFJP5554.pdf>

[4]. "ID2260A2 Dual IGBTMOD™ Power Module." Powerex, INC. PDF Online < http://www.ic-on-line.cn/iol/viewpdf/id2260a2_643364.htm>

[5]. "KA78XX/KA78XXA 3-Terminal 1A Positive Voltage Regulator." Fairchild Semiconductor. 1 June 2001. PDF Online <http://www.datasheetcatalog.org/datasheets/228/390068_DS.pdf>

[6]. Kernighan Brian and Ritchie, Denis. The C Programming Language 2nd Edition. Englewood Cliffs: Prentice Hall P T R, 1988.

[7]. "L6388E High-Voltage high and low side driver." STMicroelectronics. Oct. 2007. PDF Online. < http://www.st.com/stonline/products/literature/ds/13991/l6388e.pdf>

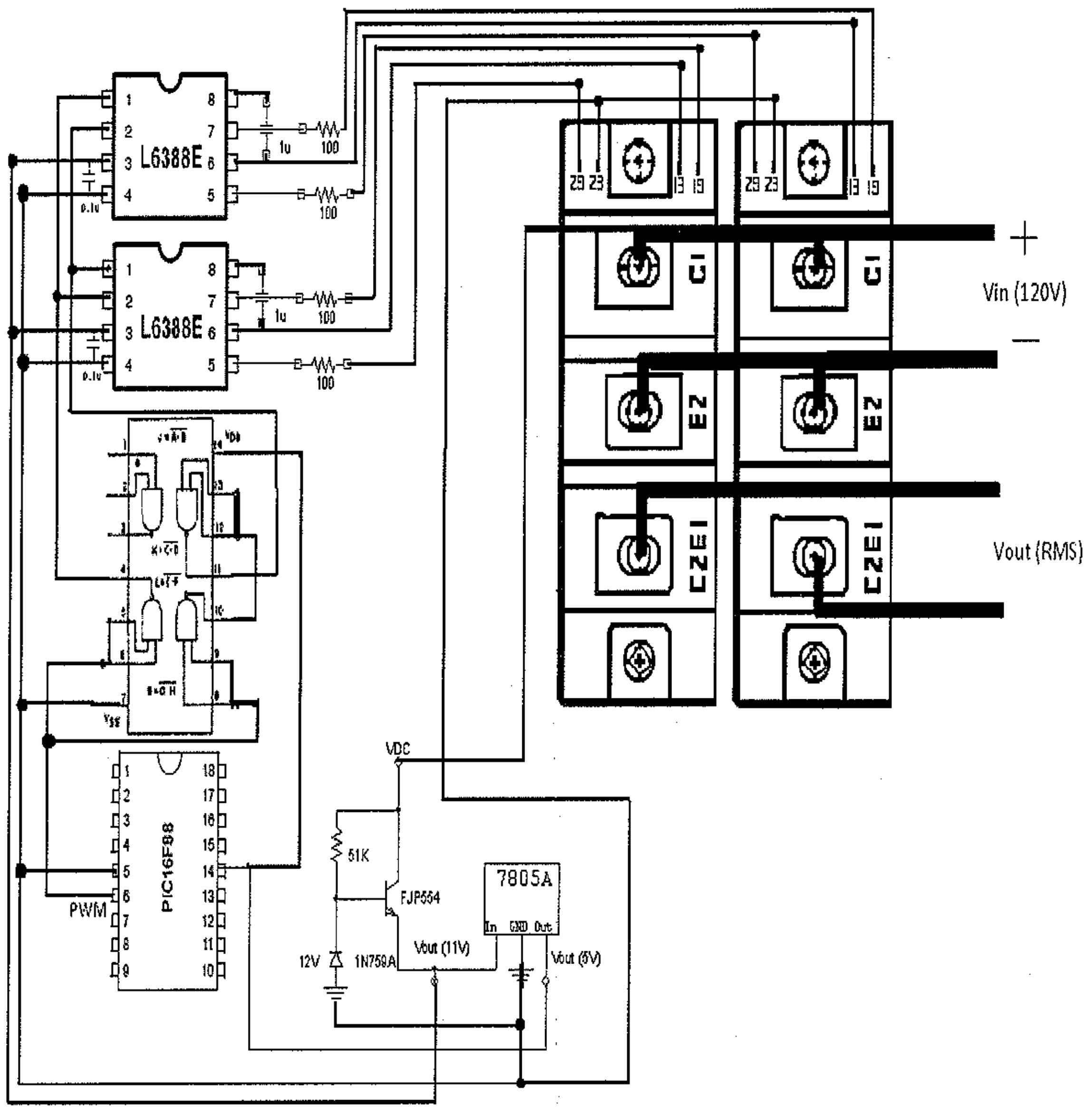
[8]. "PIC16F87/88 Data Sheet." Microchip Technology Inc. 28 Aug. 2003. PDF Online. < http://web.media.mit.edu/~jackylee/mas742/PIC16F88.pdf>

[9]. Rashid, Muhammad H. Power Electronics Circuits, Devices, and Applications 3rd Edition. New Jersey: Pearson Education, Inc, 2004.

Appendix A: Parts List and Cost

Item	Unit Price ($)	Quantity	Item Total ($)
PIC16F88 Microcontroller	4.75	1	4.75
Serial Port Programmer	13.95	1	13.95
1N4742 12V Zener Diode (2)	1.51	1	1.51
Project Box 7x5x3	5.99	1	5.99
Female Quick Disconnect (8)	2.19	1	2.19
Spade Connectors (6)	1.99	2	3.98
12 Gauge Red Wire (20ft)	4.99	1	4.99
PCB Standoff (4)	1.79	1	1.79
CD4011BE Quad 2-Input NAND	0.4	1	0.4
L6388E High-low Side Drivers	2.54	2	5.08
FJP554 Hi-VLTG BJT	0.61	1	0.61
PC Board	4.49	1	4.49
5V voltage Regulator	0	1	0
IC Sockets	0	4	0
100Ohm Resistor	0	4	0
51KOhm Resistor	0	1	0
50V 1uF Elec. Cap	0	2	0
100V 0.1uF Cap	0	2	0
25V 0.1uF Cap	0	4	0
ID2260A2 IGBT Module	0	2	0
5 Way Binding Posts	0	4	0
22 Gauge Wire	0	1	0
Total			49.73

Appendix B: IC Location and Hardware Configuration



Appendix C: C Code for PIC16F88 Microcontroller

```
/*
 This program initializes the PIC16F88 for PWM mode running at 2.4KHz.
 Everytime an interrupt from TIMER 2 occurs, the duty cycle is changed
 according to the PWM_Table which mimics a sinusoidal waveform

 by Mike Newbry
 edited 5/31/2009
*/

#define sbi(a, b) ((a) |= 1 << (b))       //sets bit B in variable A
#define cbi(a, b) ((a) &= ~(1 << (b)))   //clears bit B in variable A
#define tbi(a, b) ((a) ^= 1 << (b))       //toggles bit B in variable A


const int pwm_tableL[41] ={ 104,118,133,146,159,170,179,187,193,196,197,
                            196,193,187,179,170,159,146,133,118,104,
                            89 ,75 ,61 , 49, 38, 28, 20, 15, 11, 10,
                            11, 15, 20, 28 , 38, 49, 61, 75, 89,104} ;

const int pwm_tableH[41] ={ 0,3,0,2,0,0,3,1,0,1,2,
                           1,0,1,3,0,0,2,0,3,0,
                           1,0,2,0,0,1,3,0,3,2,
                           3,0,3,1,0,0,2,0,1,0 } ;



unsigned load;
int duty_cycle;

void main() {

 ANSEL = 0;              //set as digital I/O
 TRISA = 0;             // Configure PORTA as output
 OSCCON = 0b01110000;     // b6..4 = 111 = 8MHz
 PORTA = 0;              // Initialize PORTA
 INTCON = 0xC0;            // Enable GIE, PEIE

 PIE1 = 0x02;            // Enable TMR2IE
 PR2 = 207;             // sets period to 2.404kHz
 CCPR1L = 103;
```

```
  TRISB = 0xFE;            // Port B Bit 0 set to output
  T2CON = 0b00000101;      // Prescaler = 4
  CCP1CON = 0b00001100;    // Set for PWM Mode

  while (1) {

   if (load==1)  {
     duty_cycle++;
     if (duty_cycle >= 40) {
       duty_cycle = 0;
     }
     CCPR1L = pwm_tableL[duty_cycle];        //load 8 MSB's
     if (pwm_tableH[duty_cycle] == 0) {      //load 2 MSB's algorithm
       cbi(CCP1CON, 5);
       cbi(CCP1CON, 4);
     } else if (pwm_tableH[duty_cycle] == 1) {
       cbi(CCP1CON, 5);
       sbi(CCP1CON, 4);
     } else if (pwm_tableH[duty_cycle] == 2) {
       sbi(CCP1CON, 5);
       cbi(CCP1CON, 4);
     } else if (pwm_tableH[duty_cycle] == 3) {
       sbi(CCP1CON, 5);
       sbi(CCP1CON, 4);
     }
     load = 0;
   }
  }
}

void interrupt() {
 if (PIR1 & 0x02) {     // see if TMR2 matched PR2
    TMR2 = 0;
    load = 1;
    PIR1 = 0x00;        // clear TMR2 interrupt flag
 }
}
```