Final Report

DC-DC Boost Converter (5V to 13V)

05/14/2017

EE 458 Design of Power System Components

Professor: Wajdi Aghnatios

Sadaf Sadat Riazi (012163760)
Abdulaziz Aljaber (010616656)

Abstract

The DC-DC boost converter is made possible by generating a square wave pulse with a certain frequency and duty cycle. A DC-DC boost converter has two main jobs. First, it boosts up the DC voltage to a more suitable voltage that is steady and smooth. Second, it steps down the current instantaneously. The smooth and steady output voltage and current are obtained by using the perfect values of our inductor and capacitor that were calculated using the proper formulas. The N-channel MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) acts as a high frequency switch in order to boost up the output voltage. The high frequency signal that is required to drive the MSOFET can be obtained through Arduino UNO, 555 Timer or a Switching regulator IC such as LM2577T.

Table of Contents

List of Figures

List of Tables

Introduction

Basically, a DC-DC Boost converter is a power system component that converts a low voltage input to a higher voltage output. It is used in many different industries to run devices that use low input voltage but need to be run in a high voltage. Boost converters are usually used in hybrid electric vehicles (HEV) and lighting systems. As an example, the NHW20 model Toyota Prius HEV uses a 500 V motor. Without a boost converter, the Prius would need nearly 417 cells to power the motor. However, a Prius actually uses only 168 cells and boosts the battery voltage from 202 V to 500 V. The main or ideal components of the circuit are a switch (usually MOSFET), an inductor, a capacitor, and a resistor, however, more components can be used to improve the circuit or get an adjusted output. In this report, we will be introducing the design of a user friendly 5 Volts to 13 Volts boost converter that is used to run a 13 Volts motor.

Theory and Design Analysis

Figure 1.1 shows the boost converter circuit when the frequency of the square wave that is applied to the MOSFET gate at its high period. At this time, the MOSFET conduct and the current flows between the positive and negative supply terminals passing thought the inductor. The inductor within the circuit is used to store the energy in its magnetic field. There is almost entirely no current flowing through the remainder of the circuit mainly because the other parts of the circuit have much higher impedance that the path directly thought the heavily conduction MOSFET.

The next figure 1.2 shows the current going through the circuit when the frequency of the square wave is at its low period. At this frequency, the MOSFET is turned off and this rapid drop in the current causes the inductor to produce a back e.m.f in the opposite polarity of the voltage across inductor. Therefore, the two voltages of the input voltage and the back e.m.f voltage across the inductor are applied to the circuit, which are in series with each other. Since there is no current going through the MOSFET, the high voltage of ($V_{IN} + V_L$) will flow directly to the diode within the circuit. The current going through the diode charges up the capacitor to ($V_{IN} + V_L$) and supplies the load within the circuit.

Figure 1.3 shows the circuit during the high period of the switching square wave cycle after the initial start up. Every time the MOSFET conducts, the diode is turned of because the cathode of the diode is more positive than its anode, due to the charge on capacitor. Although the diode is isolated from the circuit, the load continues to have the output voltage of ($V_{IN} + V_L$) from the charge of the capacitor. The capacitor is recharged each time the MOSFET switches off. Therefore the load maintains an almost steady output voltage.

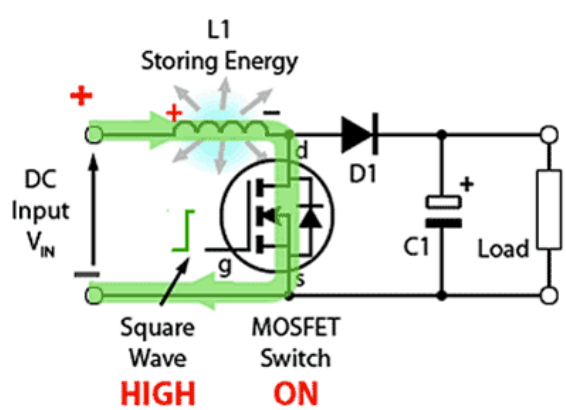

Figure 1.1: Current Path at MOSFET ON (Initial Start Up)

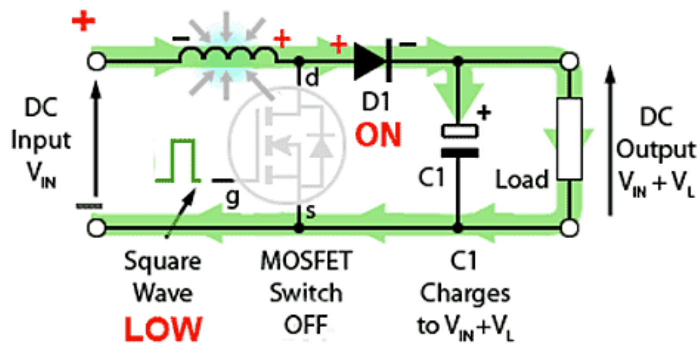

Figure 1.2: Current Path at MOSFET OFF

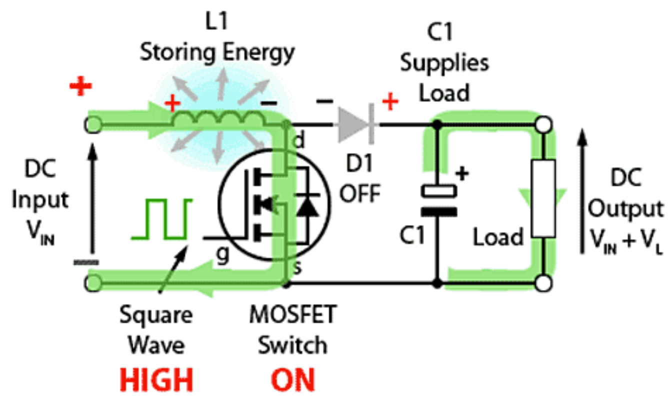

Figure 1.3: Current Path at MOSFET ON

<u>Apparatus and Equipment List</u>

Quantity	Component Name	Manufacturer/Model Number
1	Digital Oscilloscope	DS1102E/ Agilent 2000X
1	Digital Multi-Meter	INNOVA 3320
1	DC Power Supply	DP832

Table 1.1: Equipment List

Experimental Procedure

Initially, a basic DC-DC boost converter circuit was built with the use of the RLC circuit along with a diode and an N-channel MOSFET to control the ON and OFF switching of the circuit. Having this type of circuit required us to generate a square wave pulse that would help us to produce a certain frequency and duty cycle that would activate our MOSFET. This square wave pulse would help the N-Channel MOSFET to switches ON and OFF rapidly and make the circuit work. We tried many different ways to generate this type of pulse at 50,000 Hertz and the duty cycle of 58 percent. The 0.58 duty ratio was obtained through our calculations used for a DC-DC boost converter circuit with an input voltage of 5 Volts and an output voltage of 12 Volts.

At first, we tried using a 555-timer circuit to generate this type of square wave pulse for our MOSFET. Throughout our researches and calculations, we figured that a capacitor with the value of 100 nF and a resistor with the value of 50 Ohms and another resistor with the value of 120 Ohms has to be connected in parallel to each other and the correct pins of the 555 timer in order to produce a frequency of 50,000 Hertz and a duty ratio of 0.58 accordingly. Although, we were not successful in producing 0.58-duty ratio through our 555-timer circuit but we were able to produce the 50K Hertz frequency. Due to the losses by the components used in this circuit, we were able to only produce a 0.40 duty ratio, which was not enough for boosting up the voltage. Through our researches we also found out an alternative way of generating this type of pulse with an Arduino UNO. With the use of Arduino UNO board, we were able to create a PWM needed to generate the pulse we wanted but our circuit still didn't work.

Throughout our researches we found out that both 555-timer and the Arduino Uno board generate the pulse needed for the boost converter circuit at a very low current that was not enough to activate our N-channel MOSFET. We also found out that we need to build an amplifier or a gate driver circuit with the use of a NPN BJT transistor to boost up the current coming from either the 555-timer or the Arduino UNO in order to activate the MOSFET within our boost converter circuit to run our circuit.

Although none of the experimental procedures mentioned above gave us the result we wanted, we did not stop our researches and continued our hard work in order to find an alternative solution to make our DC-DC boost converter to work. Our purpose was to build a DC-DC boost converter that anyone can use at their homes without the need of any equipment such as the pulse generator from the electrical and electronic laboratories. Therefore we did not stop trying and continued our researches, until we found a switching regulator IC switch (LM2577T) that had a built in PWM that would give us the frequency and the duty ratio that we were looking for to make our boost converter circuit work. With the use of LM2577T, we were successful in making a DC-DC boost converter that would step up the voltage from 5 Volts input voltage to 13 Volts output voltage, which would be good enough to run a 12 Volts motor or charge a 12 Volts battery.

Mathlab Simulink and Graphs for Basic Boost Converter

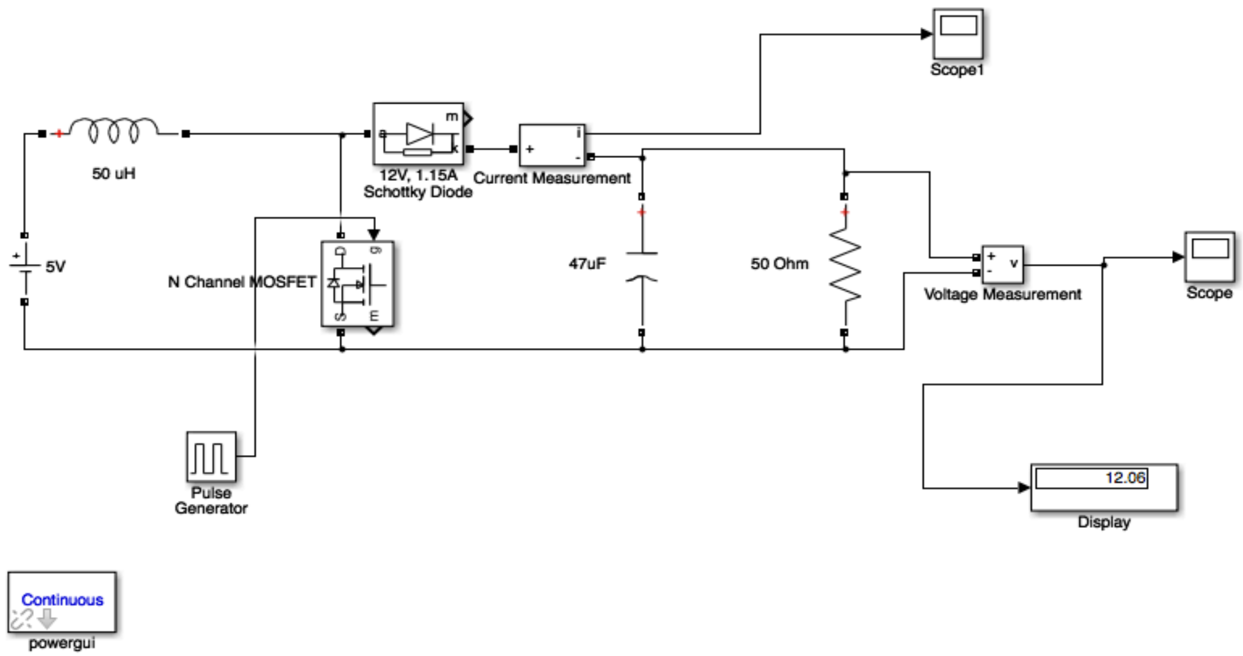

Figure 1.4: Schematic for Basic DC-DC Boost converter (5V to 12V)

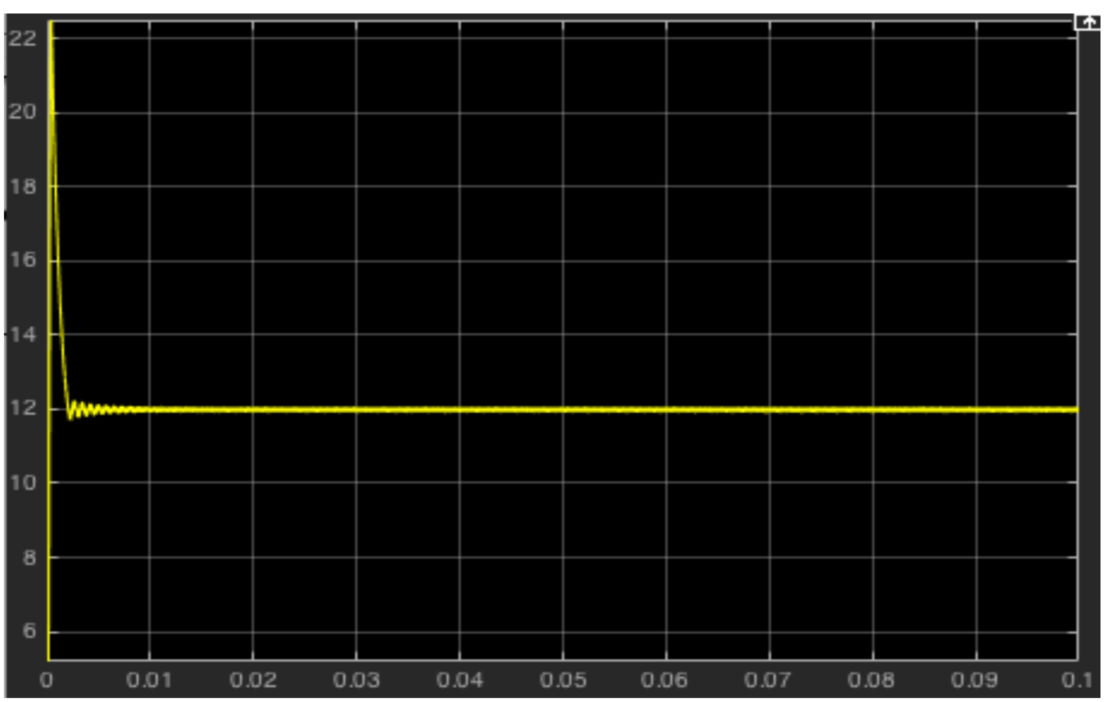

Figure 1.5: Output DC Voltage Graph for the Matlab Simulink (12 Volts)

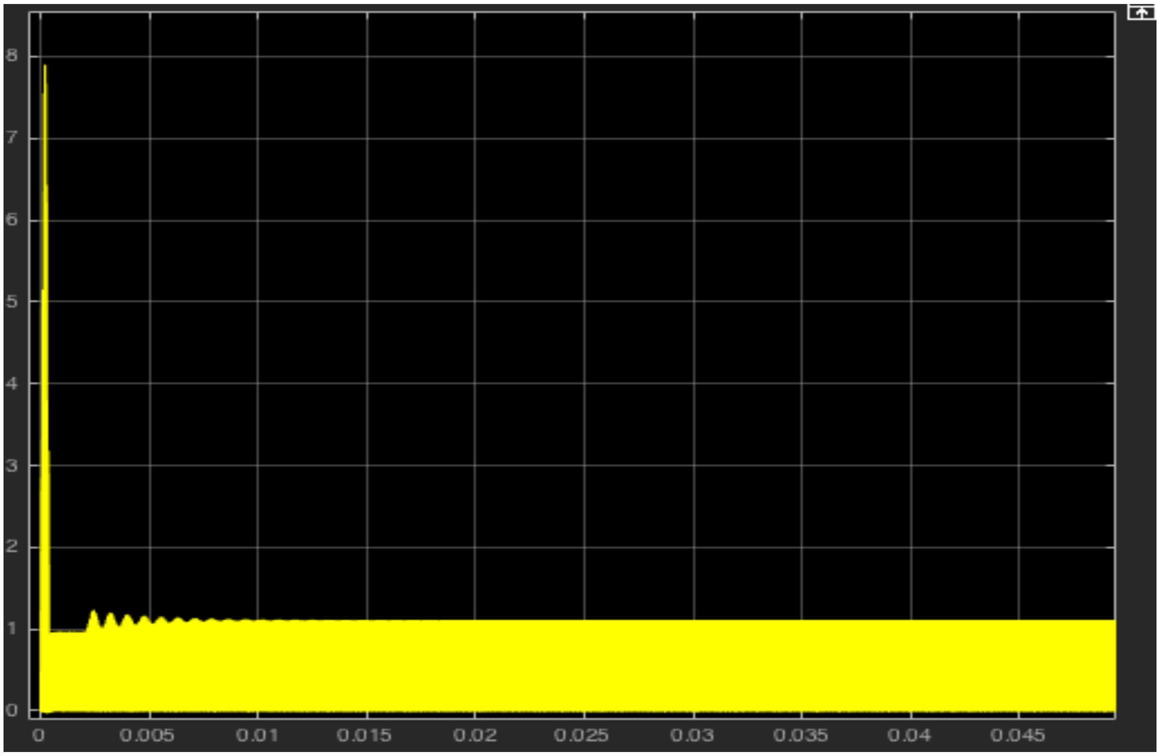

Figure 1.6: Output Current Graph for the Matlab Simulink

Mathematical Model for Boost Converter

$$Input\ Voltage\ (Vin)\ =\ 5\ Volts$$

$$Output\ Voltage\ (Vout)\ =\ 12\ Volts$$

$$Switching\ Frequency\ (fs) =\ 50{,}000\ Hertz$$

$$Resistor\ (R) =\ 50\ Ohms$$

$$Duty\ Ratio\ (D)\ =\ 1 - \frac{Vin}{Vout} = 1 - \frac{5}{12} = 0.58 = 58\%$$

$$I_L(ave) = \frac{Vin^2}{R * (1-D)^2} = \frac{5^2}{50 * (1-0.58)^2} = 2.83\ Amps$$

$$Capacitance\ (C) = \frac{D}{R * \left(\frac{\Delta Vo}{Vo}\right) * fs} = \frac{0.58}{50 * (0.0005) * 50K} = 464\ uF \approx 470uF$$

$$L_{min} = \frac{D(1-D)^2 * R}{2 * fs} = \frac{0.58(1-0.58)^2 * 50}{2 * 50{,}000} = 51.16\ uH\ \approx 47\ uH$$

$$Maximum\ Current\ (I_{\max}) = I_L(ave) + \left(\frac{(Vin * D)}{2 * L * fs}\right) = 2.83 + \frac{5 * 0.58}{2 * 51.6u * 50K} = 3.39\ Amps$$

$$Minimum\ Current\ (I_{min}) = \ I_L(ave) - \frac{1}{2}\ \Delta i_L = 2.269\ Amps$$

<u>Components Used for Basic Boost Converter</u>

Quantity	Component Name	Calculated Values	Measured Values	Manufacturer/Model Name	Type
1	Inductor	47 uH		Philmore	RF CHOKE
1	Capacitor	470 uF	-	Jack Conway	Polarized Electrolytic
1	MOSFET	-	-	FQP30N06L	Logic N-Channel
1	Heatsink	16.2 C/W		287-1ABE	-
1	Resistor	50 Ohms	49.4 Ohms	NTE	¼W 2% Carbon Film
1	Diode	100V	-	1N4148	Schottky Diode
1	Breadboard	-	-	Michael Josh	-

Table 1.2: Basic Boost Converter Components

N-Channel MOSFET Characteristics

MOSFET (Metal Oxide Semiconductor Field Effect Transistor) is a type of field-effect switch transistors that is used in many of today's applications. It consists of a Gate (G), Drain (D), and a Source (S). The switch has three modes of operations that work depending on its intended application, which are: Cutoff, Saturation, and Linear region shown in figure 1.7. The MOSFET is very important in electronics. They are used significantly in many applications, especially as Amplifiers in analog circuit and Electronic Switches in digital circuit.

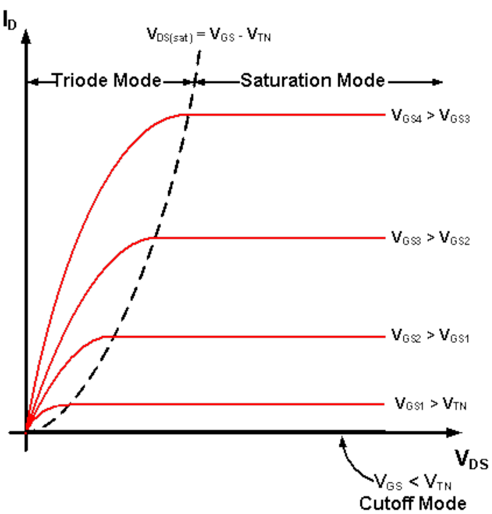

Figure 1.7: Three Operational Modes of MOSFET

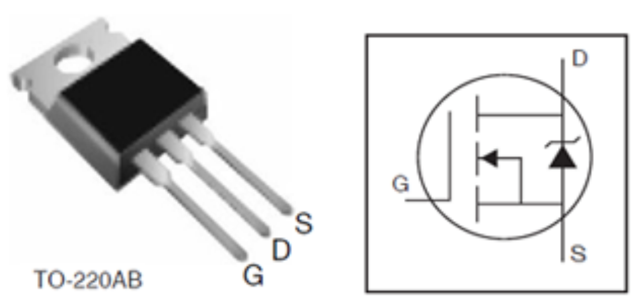

Figure 1.8: N-Channel MOSFET Pin-out "FQP30N06L"

Pulse Width Modulation (PWM)

A Pulse Width Modulated (PWM) signal is a method for generating an analog signal using a digital source. The duty cycle and the frequency are the two main components that define the behavior of PWM. The frequency of the PWM shows how fast the PWM signal completes a cycle. For example, 50,000-Hertz signal would complete 50,000 cycles per second. The frequency also controls how fast the signal is switching between the high state and the low state. However, the duty cycle within the PWM controls the amount of time the signal is in the high or the ON state as a percentage of the total time it takes to complete one cycle. PWM can be used for different purposes, such as driving signal in a DC-DC Boost converter.

Graphs of PWM signals at Different Duty Cycles

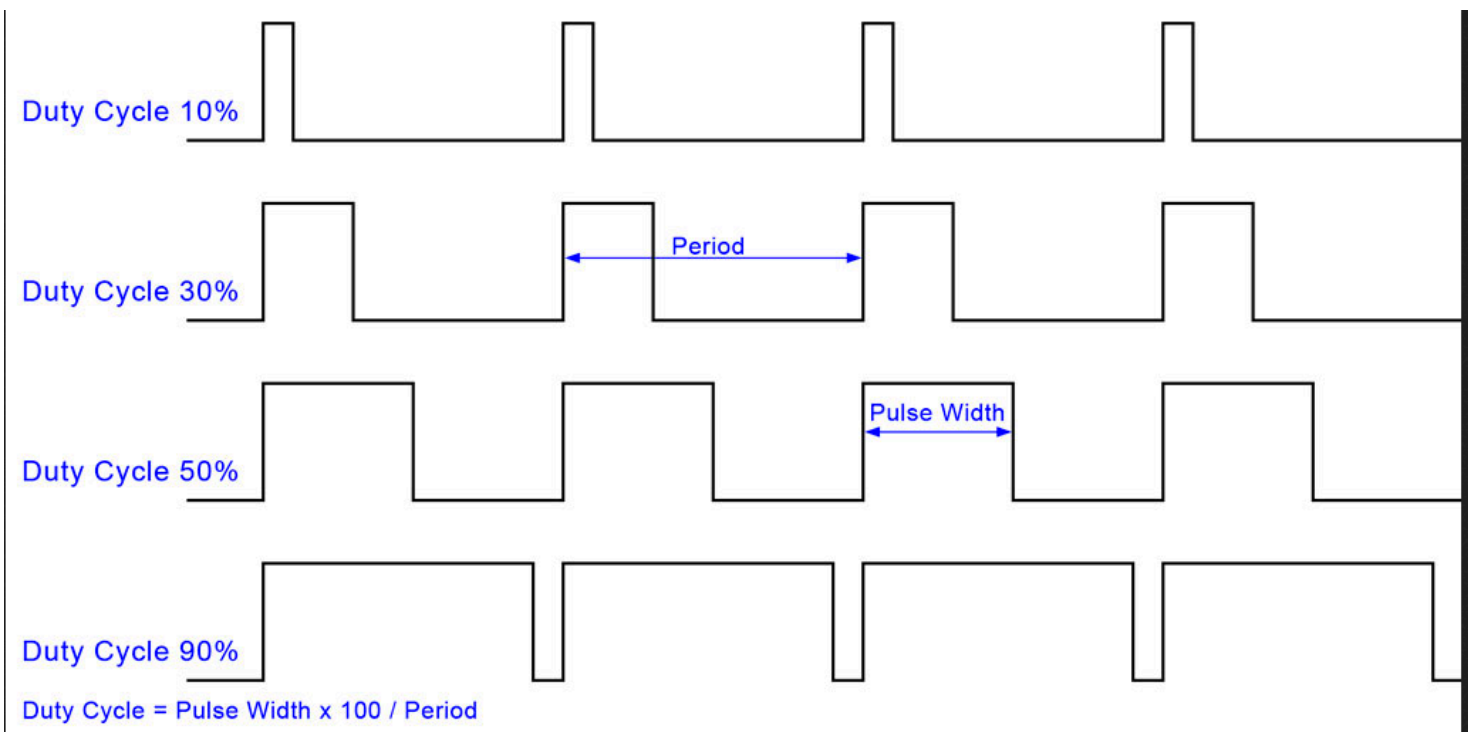

Figure 1.9: PWM signals at 10%, 330%, 50% and 90% Duty Cycles

555 Timer Circuit Analyses

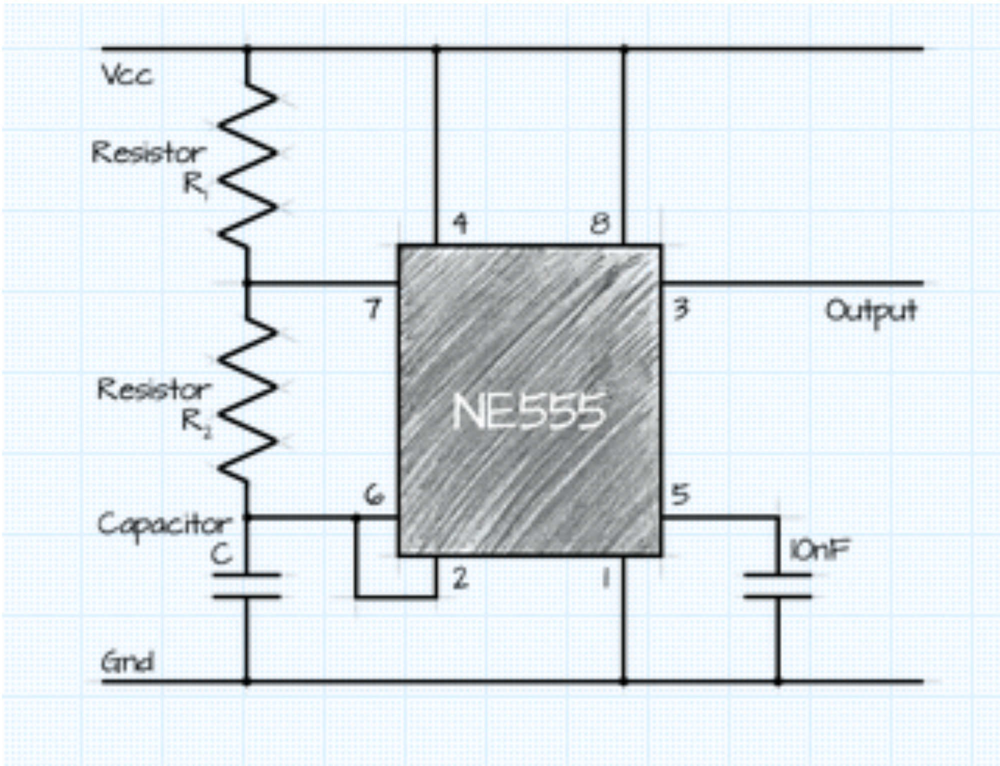

Figure 1.10: Schematic for 555 Timer Circuit

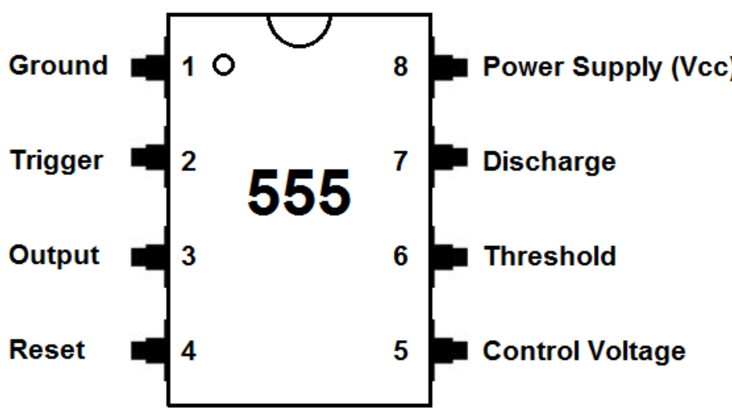

Figure 1.11: 555 Timer Chip Pin-out

The 555 Timer is an IC chip that its main purpose is to produce a pulse signal. It can be used in many applications that involve time control. In our case, we are required to produce a pulse signal with a specific duty cycle and frequency to turn on our MOSFET switch in the converter. 555 Timer also operates in three modes: Bistable, Monostable, and Astable. Bistable mode operates as a 'flip-flop', which is used to include bounce-free latched switches. Monostable mode operates as a one-time pulse generator depending on the connections of the circuit and its wanted output (i.e. PWM operation). Astable mode, which is used in this project, is a continuous non-stop pulse generator. It is the most commonly used mode because of its simplicity and demand. Astable mode is used in applications where the circuit requires non-stop pulses in order to get the right output. 555Timers can be used for both AC & DC converters.

Throughout our researches, the Values Obtained for the Resistor 1 and Resistor 2 and the Capacitor used to produce 50,000-Hertz frequency and 0.58 Duty Ratio is:

Quantity	Component Name	Calculated Values	Measured Values	Manufacturer/Model Name	Type
1	Capacitor 1	100 nF	-	NTE	Ceramic
1	Capacitor 2	10 nF	-	NTE	Ceramic
1	Resistor 1	50 Ohms	49.1 Ohms	NTE	¼W 2% Carbon Film
1	Resistor 2.1	10 Ohms	9.93 Ohms	NTE	¼W 2% Carbon Film
1	Resistor 2.2	10 Ohms	9.87 Ohms	NTE	¼W 2% Carbon Film
1	Resistor 2.3	100 Ohms	98.2 Ohms	NTE	¼W 2% Carbon Film

Table 1.3: Components Used in 555 Timer Circuit

We connected the three resistors of 2.1, 2.2 and 2.3 in series to each other in order to get the value of 120 Ohms for the resistor 2 since there was no 120 Ohms resistor manufactured.

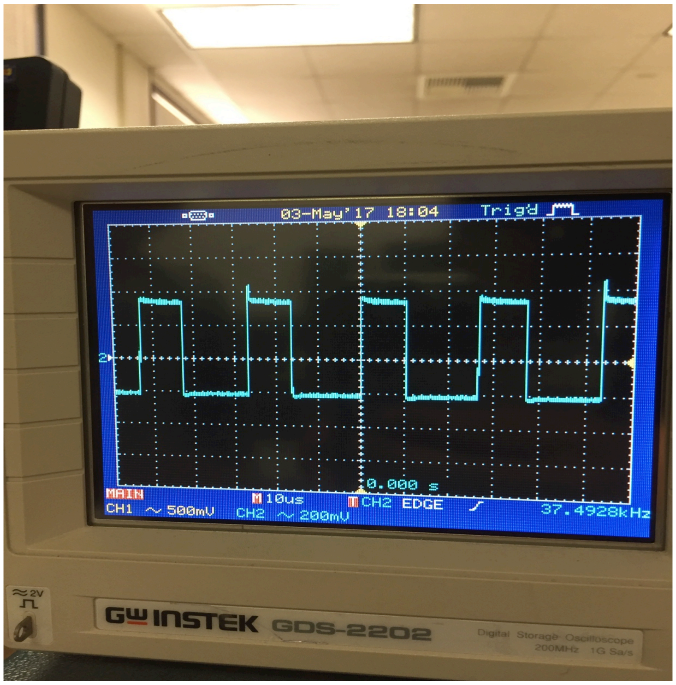

Figure 1.12: Output Square Wave Graph for 555-Timer

Arduino Uno PWM Code:

The switching frequency of 50,000 Hertz was selected for our DC-DC Boost converter. The highest frequency that can be obtained from the Arduino Uno without making any changes to the internal timer on the ATMEGA chip is supposed to be 62500 Hertz. Our selected frequency (50KHz) was reasonable enough to generate a PWM using Arduino Uno. The duty ratio of the boost converter is changed depending on the value of our input and output voltages. With the values of 5 Volts input voltage and 12 volts output voltage, the desired duty ratio to boost our input voltage was selected to be 0.58 or 58%. The attached "PWM.h" file helped us to obtain the desired frequency by taking care of the registers inside the ATMEGA chip. Figure 1.11 shows the Arduino code that controls the frequency and the duty ratio of the PWM. The duty cycle can be changed depending on the desired output voltage. A call to pwmWrite is on a scale of 0 to 255, in which pwmWrite(255) sets a 100% duty cycle. In order to get a 58% duty cycle, that is the 58% of 255, a duty value of 147.9 (353/2.38) was set to the PWM pin.

Boost_Conv

```
#include "PWM.h"
int PWM = 4;  // PWM is connected to digital pin 4
in val = 0;   // variable to store the read value
void setup()
{
 InitTimers();
 SetPinFrequency (PWM, 50000);  // sets the frequency of PWM to 5000 Hz
    pinMode(PWM, OUTPUT);       //sets the pin as otuput
}

void loop()
{
  if (Val<353)        // Limits the value analogPin to values higher than 353
  {
    Val = 353;
  }
  if (val>770)        // Limits the value analogPin to values lower than 770
  {
    val = 770;
  }
  pwmWrite(PWM, 353/2.39);  // analogRead values go from 0 to 1023, Duty Cycle 58%
}
  // put your main code here, to run repeatedly:

}
```

Figure 1.13: Main PWM Code for Boost Converter Using Arduino Uno

pwm.h

```
/*
Copyright (c) 2012 Sam Knight
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*/

/*
This library is built to support two of the AVR Architecture 'groups' that Arduino uses
a) ATmega48/88/168/328,
b) ATmega640/1280/1281/2560/2561
*/

#ifndef PWM_H_
#define PWM_H_

#include "avr/pgmspace.h"
```

```
#include "avr/pgmspace.h"
#include "math.h"

#if defined(__AVR_ATmega640__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega1281__) || defined(__AVR_ATmega2560__) || defined(__AVR_ATmega2561__)
  #include "utility/ATimerDefs.h"
#elif defined(__AVR_ATmega48__) || defined(__AVR_ATmega88__) || defined(__AVR_ATmega88P__) || defined(__AVR_ATmega168__) || defined(__AVR_ATmega168P__) || defined(__AVR_ATmega328__
  #include "utility/BTimerDefs.h"
#endif


#if defined(__AVR_ATmega640__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega1281__) || defined(__AVR_ATmega2560__) || defined(__AVR_ATmega2561__)
// 16 bit timers
extern uint32_t GetFrequency_16(const int16_t timerOffset);
extern bool   SetFrequency_16(const int16_t timerOffset, uint32_t f);
extern uint16_t GetPrescaler_16(const int16_t timerOffset);
extern void   SetPrescaler_16(const int16_t timerOffset, prescaler psc);
extern void   SetTop_16(const int16_t timerOffset, uint16_t top);
extern uint16_t GetTop_16(const int16_t timerOffset);
extern void   Initialize_16(const int16_t timerOffset);
extern float  GetResolution_16(const int16_t timerOffset);

// 8 bit timers
extern uint32_t GetFrequency_8(const int16_t timerOffset);
extern bool   SetFrequency_8(const int16_t timerOffset, uint32_t f);
extern uint16_t GetPrescaler_8(const int16_t timerOffset);
extern void   SetPrescaler_8(const int16_t timerOffset, prescaler psc);
extern void   SetPrescalerAlt_8(const int16_t timerOffset, prescaler_alt psc);
extern void   SetTop_8(const int16_t timerOffset, uint8_t top);
extern uint8_t  GetTop_8(const int16_t timerOffset);
extern void   Initialize_8(const int16_t timerOffset);
```

```
extern void   Initialize_8(const int16_t timerOffset);
extern float  GetResolution_8(const int16_t timerOffset);

#endif

#if defined(__AVR_ATmega48__) || defined(__AVR_ATmega88__) || defined(__AVR_ATmega88P__) || defined(__AVR_ATmega168__) || defined(_

// 16 bit timers
extern uint32_t GetFrequency_16();
extern bool   SetFrequency_16(uint32_t f);
extern uint16_t GetPrescaler_16();
extern void   SetPrescaler_16(prescaler psc);
extern void   SetTop_16(uint16_t top);
extern uint16_t GetTop_16();
extern void   Initialize_16();
extern float  GetResolution_16();

// 8 bit timers
extern uint32_t GetFrequency_8(const int16_t timerOffset);
extern bool   SetFrequency_8(const int16_t timerOffset, uint32_t f);
extern uint16_t GetPrescaler_8(const int16_t timerOffset);
extern void   SetPrescaler_8(const int16_t timerOffset, prescaler psc);
extern void   SetPrescalerAlt_8(const int16_t timerOffset, prescaler_alt psc);
extern void   SetTop_8(const int16_t timerOffset, uint8_t top);
extern uint8_t  GetTop_8(const int16_t timerOffset);
extern void   Initialize_8(const int16_t timerOffset);
extern float  GetResolution_8(const int16_t timerOffset);

#endif
```

```
//common functions

extern void  InitTimers();
extern void  InitTimersSafe();                //doesn't init timers responsible for time keeping functions
extern void  pwmWrite(uint8_t pin, uint8_t val);
extern void  pwmWriteHR(uint8_t pin, uint16_t val);        //accepts a 16 bit value and maps it down to the timer for maximum resolution
extern bool  SetPinFrequency(int8_t pin, uint32_t frequency);
extern bool  SetPinFrequencySafe(int8_t pin, uint32_t frequency);  //does not set timers responsible for time keeping functions
extern float GetPinResolution(uint8_t pin);           //gets the PWM resolution of a pin in base 2, 0 is returned if the pin is not connected to a timer

#endif /* PWM_H_ */
```

Figure 1.14: The "PWM.h" Included File

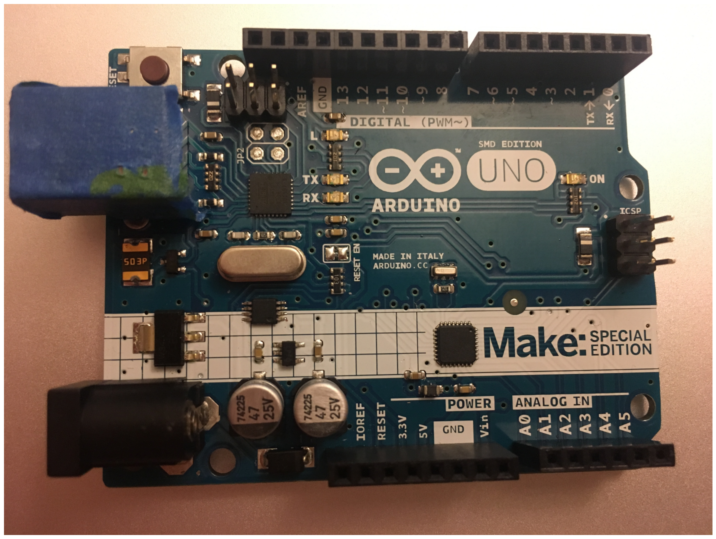

Figure 1.15: Arduino Uno Board

Amplifier or Gate Driver Circuit Analysis

Figure 1.14 shows the amplifier or the gate driver circuit that was used in order to boost the current that was generated from either the 555-timer circuit or the Arduino Uno. This is because the PWM current that goes from the output of 555-timer circuit or the Arduino Uno is too low to activate the N-channel MOSFET in the boost converter circuit.

In our circuit, we chose the value of 560 Ohms between 500 to 1.2K (5V-12V) Ohms resistance. We also chose our R2 to be 10 times the value of our R3. We chose 2N222 BJT transistor in order to get the pulse from the Arduino or the Timer and transfer the current to the MOSFET of the boost converter after boosting the current.

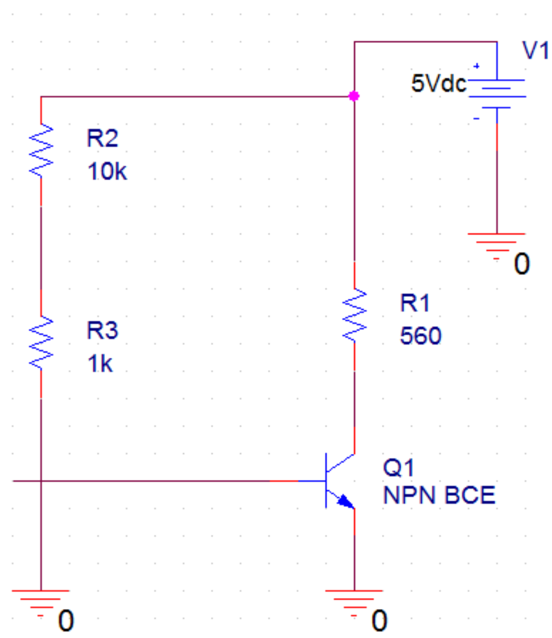

Figure 1.16: Schematic of the Gate Driver

Components Used for Gate Driver Circuit

Quantity	Component Name	Calculated Values	Measured Values	Manufacturer/Model Name	Type
1	Resistor 1	10K Ohms	9.99K Ohms	NTE	¼W 2% Carbon Film
1	Resistor 2	1K Ohms	995 Ohms	NTE	¼W 2% Carbon Film
1	Resistor 3	560 Ohms	556 Ohms	NTE	¼W 2% Carbon Film
1	Transistor	30V	-	Farnell 2N2222	NPN BJT

Table 1.4: List of Gate Driver Parts

Pspice Schematic and Graph of Boost Converter Using LM2577T

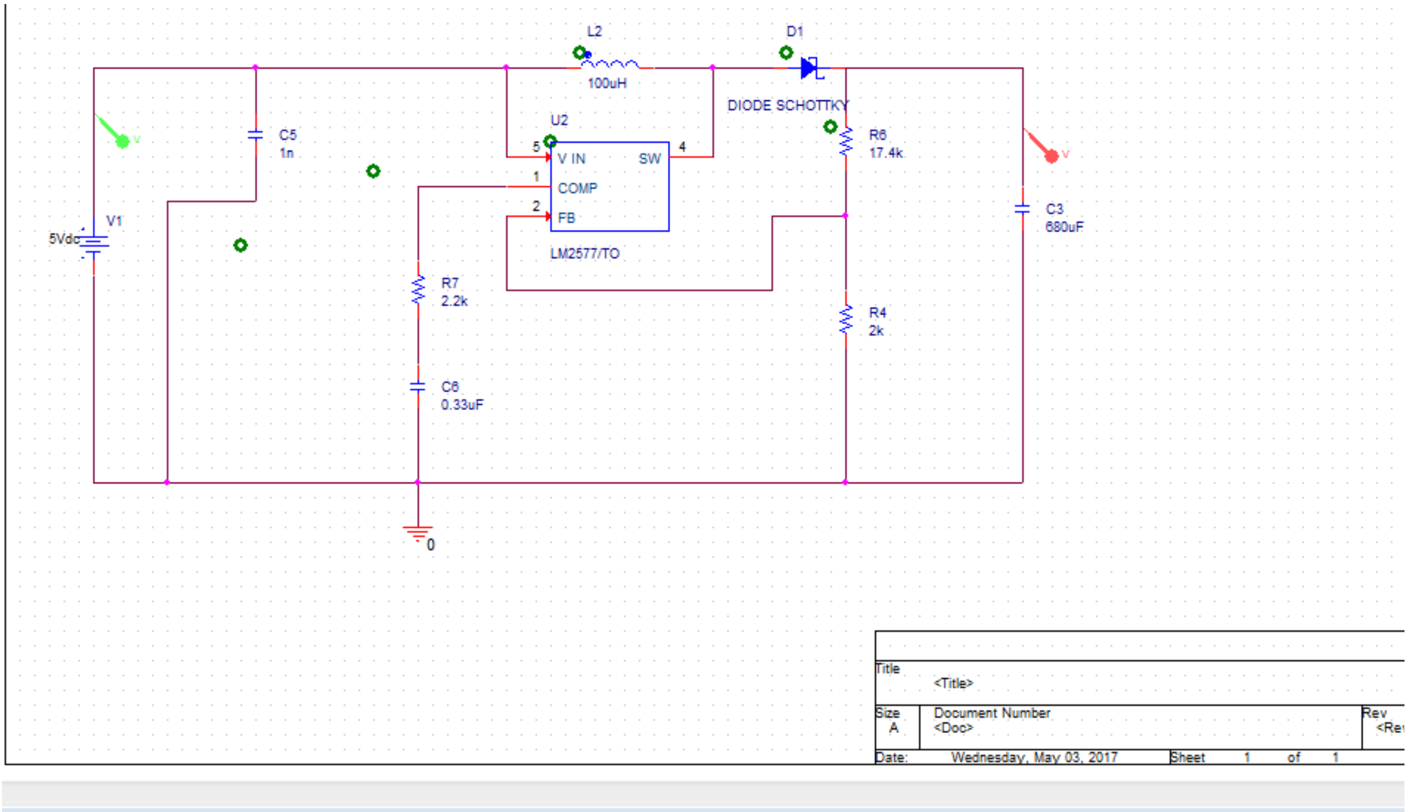

Figure 1.17: Pspice Schematic of Boost Converter Using LM2577T

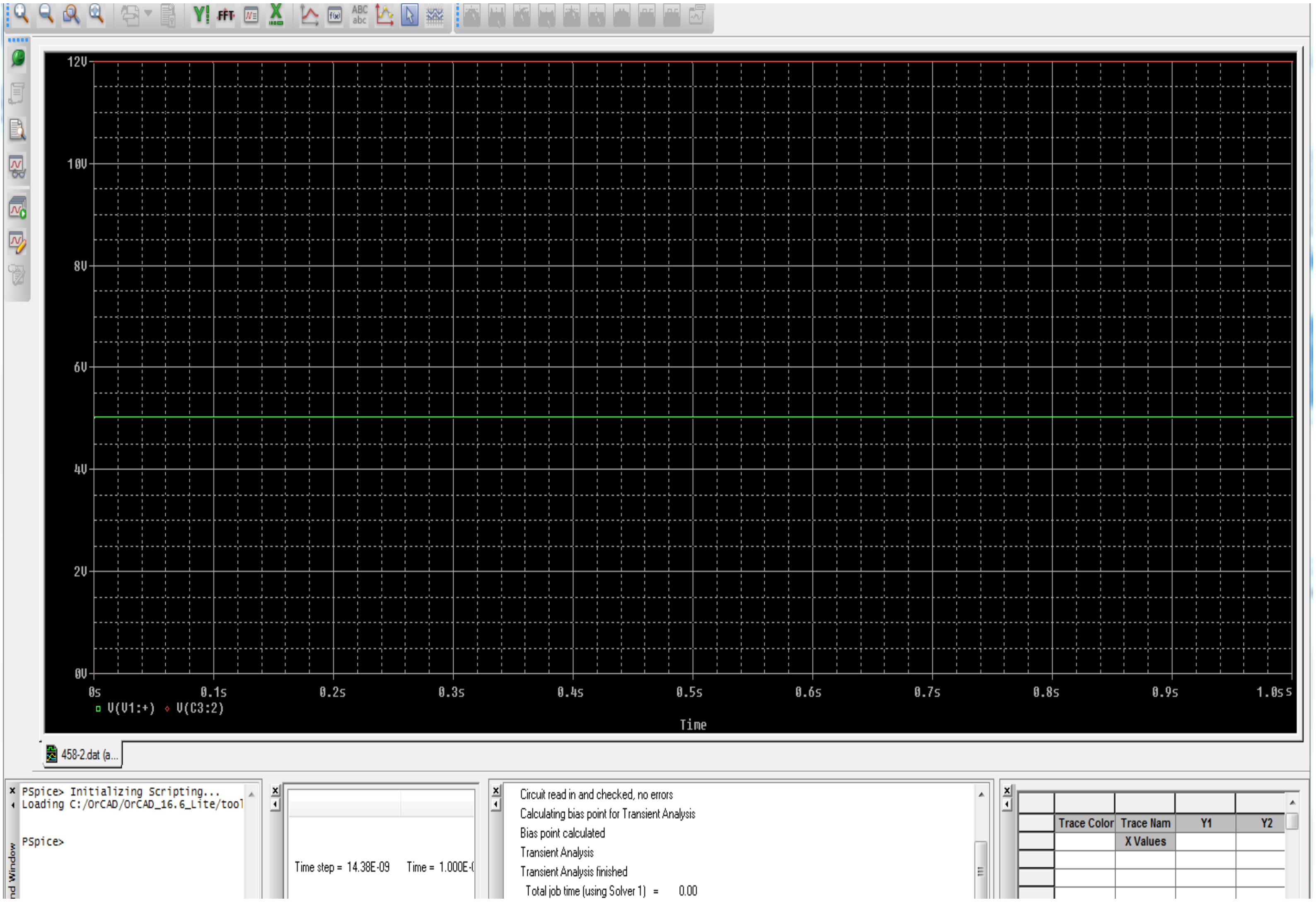

Figure 1.18: Pspice Input (Green) and Output (Red) Graph of Boost Convert Using LM2577T

Components Used in Boost Converter Using LM2577T

Quantity	Component Name	Calculated Values	Measured Values	Manufacturer/Model Name	Type
1	Regulator IC	-	-	LM2577T	Switching
1	Inductor	100 uH	-	Philmore	RF CHOKE
1	Capacitor	680 uF	-	Jack Conway	Polarized Electrolytic
1	Capacitor	0.33 uF	-	NTE	Ceramic

1	Capacitor	0.1 uF	-	NTE	Ceramic
1	Resistor 1	18K Ohms	17.9K Ohms	NTE	¼W 2% Carbon Film
1	Resistor 2	2K Ohms	1.97K Ohms	NTE	¼W 2% Carbon Film
1	Resistor 3	2.2K Ohms	2.19K Ohms	NTE	¼W 2% Carbon Film
1	Diode	400 V	-	Motorola1N4004	Schottky Diode
1	DC Motor	12V	-	Sparkfun	Gear Motor (140 RPM)
4	Batteries	1.2 V (Each)	-	NUN1100-AAC	AA
1	AA Battery Holder	-	-	-	4 Cell
1	Breadboard	-	-	Michael Josh	-

Table 1.5: Components Used in Boost Converter Using LM2577T

LM2577T Switching Regulator IC

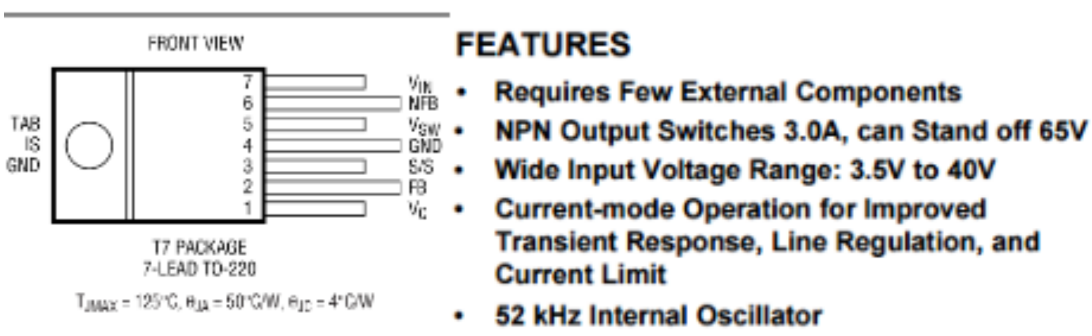

Figure 1.19: LM2577T Features and Pin-out

The LM2577 is a Step-up voltage regulator that is mainly designed for boost converters. It has a built in switch and an internal 52kHz fixed frequency that requires no external oscillator. Using this IC will help with designing the boost converter. It has an NPN output switch of 3A, and can stand off 65V output. Input voltage can range from 3.5V to 40V. However, the chip is available in three different outputs versions: 12V, 15V, and adjustable.

Circuit Using 555 Timer

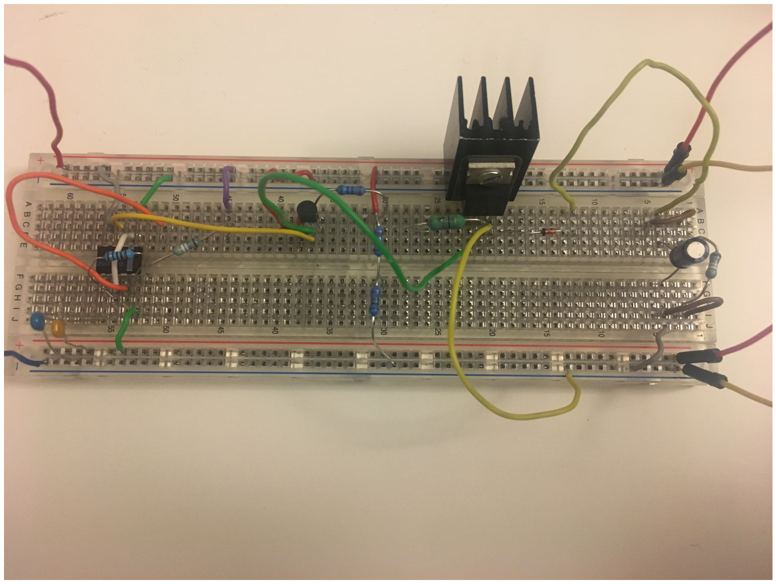

Figure 1.20: Boost Converter Circuit Using 555 Timer

Circuit Using Arduino Uno

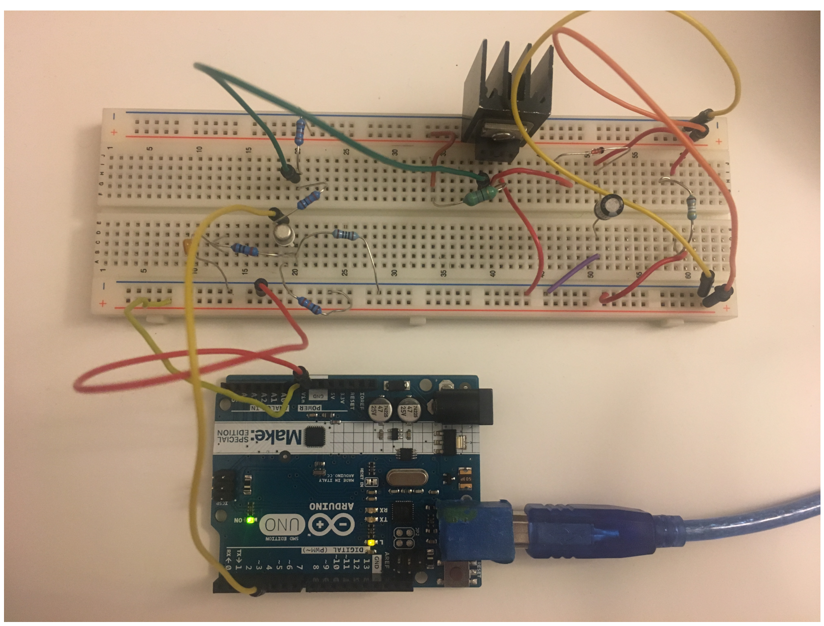

Figure 1.21: Boost Converter Circuit Using Arduino Uno

Circuit Using LM2577T (Final)

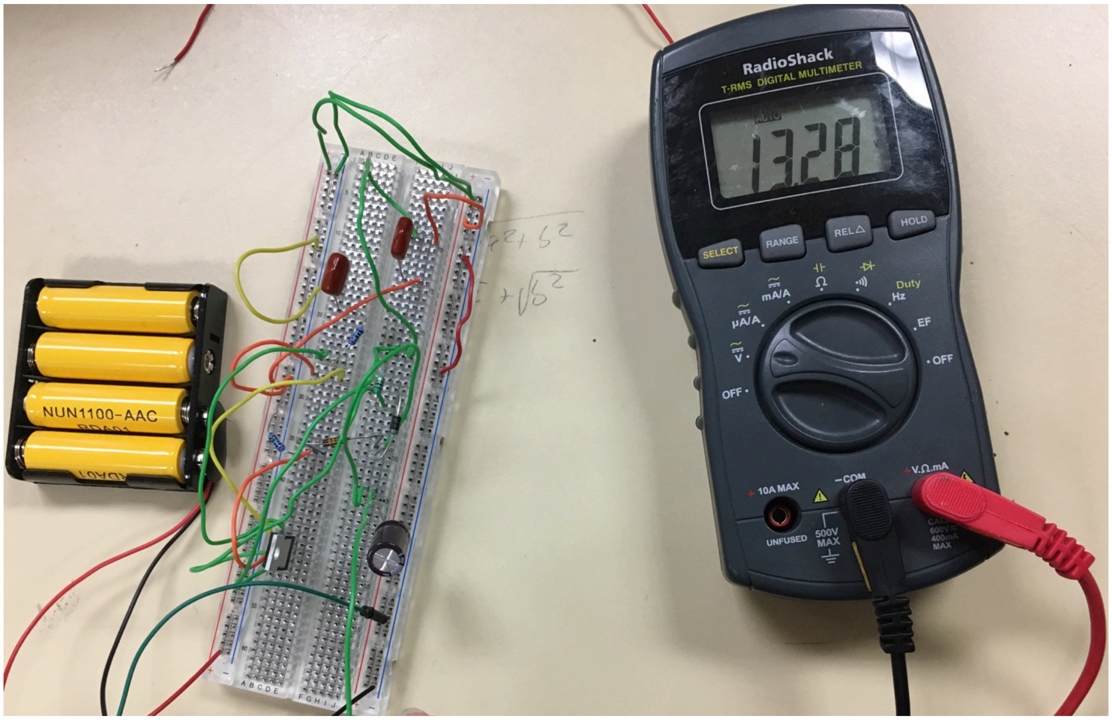

Figure 1.22: Boost Converter Circuit Using LM2577T (Multi-meter)

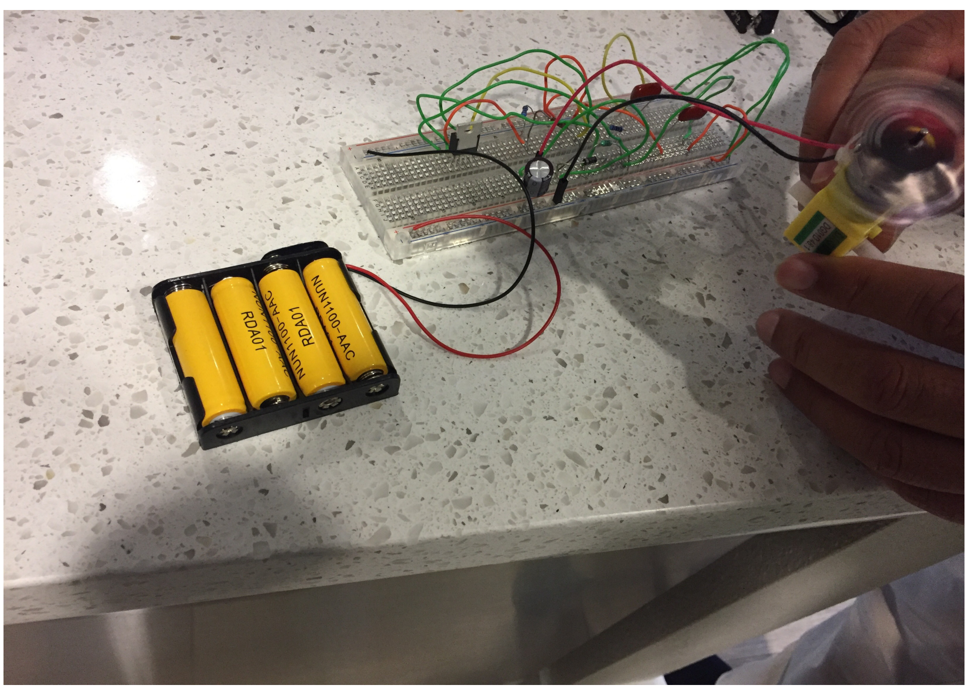

Figure 1.23: Boost Converter Circuit Running 12 Volts Motor with 5 Volts Input

Conclusion and Recommendations

In conclusion, we were able to finish up the project completely. When we first chose our project we thought that it might be easy, but it turned out to be quite the opposite. Building a boost converter requires many aspects in electrical engineering that involves circuit design, circuit analysis, and control. The basic boost converter that we built using Simulink taught us the basics of building a boost converter. Of course, building the circuit in real life is very different than simulating it on software. Finding or purchasing the components was fairly simple, we either bought it online or from a nearby store. But, connecting the circuit and getting the desired output wasn't easy. Our main issue was with the signal. We tried using different pulse generators such as Arduino and 555-Timer IC, but the current wasn't high enough to turn the MOSFET switch ON. Then we used a gate driver to increase the current, but still the output voltage was lower than the input. And finally, we used the LM2577 that has an internal oscillator and that solved our problem. Getting the desired duty cycle and frequency was our main challenge in this project and after many tries we finally solved it. Generally the course wasn't very intensive, and it allowed the students to work freely for their project to discover their own challenges and learn the new skills and experiences. Throughout the semester, my lab partner and I gained many skills and experiences in the circuit design, the circuit analysis, and the control of power system components.

References

1. “Boost Converter” <https://en.wikipedia.org/wiki/Boost_converter> Edited 6 April 2017.
2. “Boost Converter Operation” <http://www.learnabout-electronics.org/PSU/psu32.php> Revised 17 February 2017.
3. Mohan, N., Undeland, T. M., & Robbins, W. P. (n.d.). “Power Electronics: Converters, Applications and Design” (2nd ed.).
4. “Embedded PWM” <http://www.embedded.com/electronics-blogs/beginner-s-corner/4023833/Introduction-to-Pulse-Width-Modulation>.
5. “ATmega168A Pulse Width Modulation – PWM” <https://protostack.com.au/2011/06/atmega168a-pulse-width-modulation-pwm/>.
6. “555 Timer Pin-out” <http://www.learningaboutelectronics.com/Articles/555-timer-pinout.php>.
7. “555 Astable Circuit Calculator” <http://www.ohmslawcalculator.com/555-astable-calculator>.
8. “FQP30N06L N-Channel MOSFET Pin-out” <https://i0.wp.com/www.g7smy.co.uk/gallery/var/albums/blog/irf540n_chip.png?resize=350%2C154>
9. “Arduino analogWrite()” <https://www.arduino.cc/en/Reference/AnalogWrite>.
10. “LM2577T Datasheet” <http://www.ti.com/lit/ds/symlink/lm2577.pdf>.
11. “555 Timer IC” <https://en.wikipedia.org/wiki/555_timer_IC>.
12. “How an n-Channel MOSFET Works” <http://macao.communications.museum/eng/exhibition/secondfloor/MoreInfo/2_10_4_HowFETWorks.html>.
13. “MOSFET” <https://en.wikipedia.org/wiki/MOSFET#Operation>.
14. “LM555 Timer Datasheet” <http://www.ti.com/lit/ds/symlink/lm555.pdf>.