15.01.2020

MIDDLE EAST TECHNICAL UNIVERSITY
ELECTRICAL and ELECTRONICS ENGINEERING DEPARTMENT

EE463 STATIC POWER CONVERSION I

Fall 2019

HARDWARE PROJECT REPORT

Team Name: Dynamic Power

Team Members:	**Canberk Duman**	**2030450**
	Hamza Solak	**2263762**
	Musa Yeli	**2263846**

To be submitted to: Assist. Prof. Ozan KEYSAN

Table of Contents

1. INTRODUCTION

The aim of this project is to design and implement a controlled three-phase rectifier which enables to drive a DC motor which is loaded with the generator (a kettle). In order to meet requirements, three different topologies which are 3-phase thyristor rectifier, 1-phase thyristor rectifier and diode rectifier plus buck converter were discussed and investigated.

In this report, the selection of topology, design and implementation steps of the project are presented with reasoning, computer simulations and test results.

2. DESIGN DECISIONS

2.1. Topology Selection

The topologies were discussed with respect to three main parameters which are simplicity, performance and offerings (four-quadrant drive). All three topologies were discussed and compared with each other according to these parameters.

3- Phase Thyristor Rectifier:

Three-phase topology is very common in power system and high power applications. It offers more DC like output and also less AC components (harmonics) in the load side as compared to single-phase rectifiers. Therefore, filtering is simpler than single-phase to reach steady DC output and less ripple. In this topology, all six thyristor should be fired properly and synchronized to achieve desired voltage level and steady DC output. The gate driver and controller design of this topology is very problematic. Although we want to prefer it for four-quadrant drive at the beginning of the design process, the possible mistakes and complexity of it changed our direction.

Single-Phase Thyristor Rectifier:

As mentioned above, the single-phase topology does not offer any advantages over three-phase in our case. The same complexity and possible failures due to thyristor driver are also valid for this topology. Although single-phase preference seems cheaper and more compact than three-phase topologies, diode rectifiers are not that big or expensive than thyristors.

3 Phase Diode rectifier Buck Converter:

This topology is mainly composed of a buck converter (switch) and 3-phase diode rectifier. Diode bridge rectifier converts 3-phase AC to DC voltage across a capacitor legs and the buck converter (switch) manage the amplitude of DC voltage at output via controlled gate driver and a freewheeling diode. The main advantage of this topology is the ability which can be driven by a single and simple gate driver. It refers to simplicity and compactness for the product. Filtering process could be problematic according to switching frequency, but the motor itself is a low pass filter and there is no need for huge inductors.

In the light of these knowledge and facts, 3-phase diode rectifier plus buck converter is the final decision for the topology. Although 3-phase thyristor was the first choice to enable four-quadrant operation, the topology decision has changed due to the complexity of gate drivers and isolation problem between low-side and high-side. Therefore, both 3-phase diode rectifier and buck converter is chosen for compactness, less ripple at the load side and simpler controller and gate driver.

2.2. Gate Driver

There are two options for gate driver which are using a microcontroller or a 555 timer package. In order to drive the switch(IGBT) and implement a PID controller, an arduino nano was used. The arduino also helps to access soft-start ability which is very important to prevent impulsive high current drawn due to sudden changes in duty cycle. Two digital pin of arduino are connected to two buttons which refer to increase and decrease in duty cycle. The controlled PWM output of arduino is managed via these buttons and PID controller. The code of gate driver and controller can be found in Appendix B and Appendix C.

An isolation circuit is also necessary between the arduino and the gate of IGBT(switch). An optocoupler (TLP250) was used as gate driver of IGBT, can be seen in Figure 1. The optocoupler was chosen according to isolation voltage level, 2500V for TLP250.

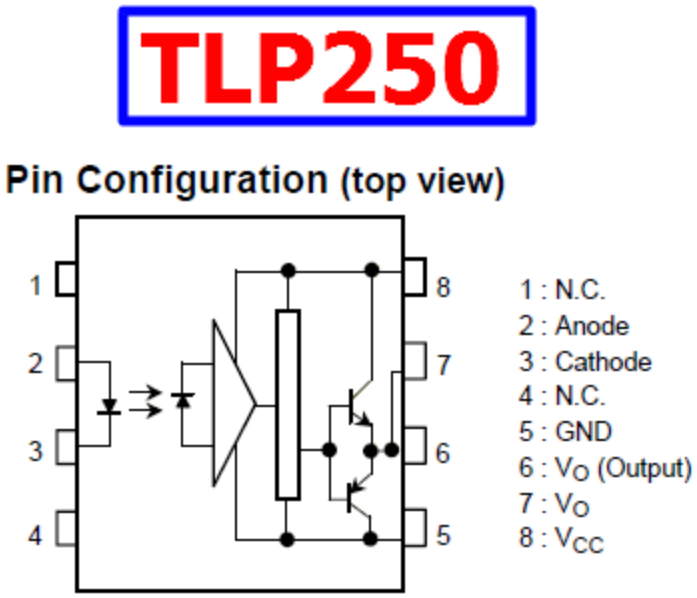

Figure 1. Optocoupler as the gate driver

The PID controller was also designed for constant speed operation under varying load, however implementation of it could not be successful. A hall effect sensor was used for unity feedback measurement, it works with magnets and flux phenomenons. The output of hall sensor is connected to a digital pin of the arduino nano and it generates ‘high’ output for each cycle. A simple counter code calculates the speed of shaft by counting these ‘pulses’.

3. COMPUTER SIMULATIONS

After choosing the topology of the driver, we started to determine the proper equipment by making simulation.Output voltage, dc motor inrush current, output power motor rating voltage are effect the device selection.We measured the voltage of full bridge rectifier for choosing three-phase diode rectifier, dc-link capacitor and igbt. We tried to adjust output voltage at 180 Volts which is limited by project description.

We simulate our topology in Matlab Simulink. We arranged the parameters of the simulation with respect to the datasheet of the device. We simulate the duty cycle 0.25, 0.5, 0.75, 1 and we observe the input and output characteristic of the circuit. Also we observe boundary condition of continuous conduction mode. The circuit schematic is shown in Figure 2.

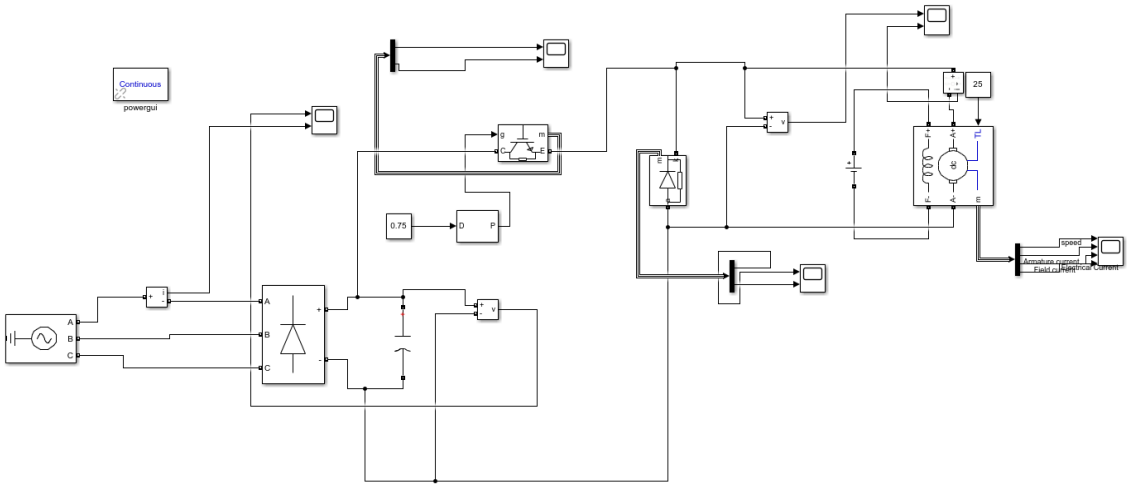

Figure 2. The circuit schematic of converter in Simulink

Firstly, we adjust the duty cycle 1 and observe armature current, dc link capacitor voltage, and input current and diode voltage. Results are shown in figure 3 and figure 4.

- PWM Duty Cycle = 1

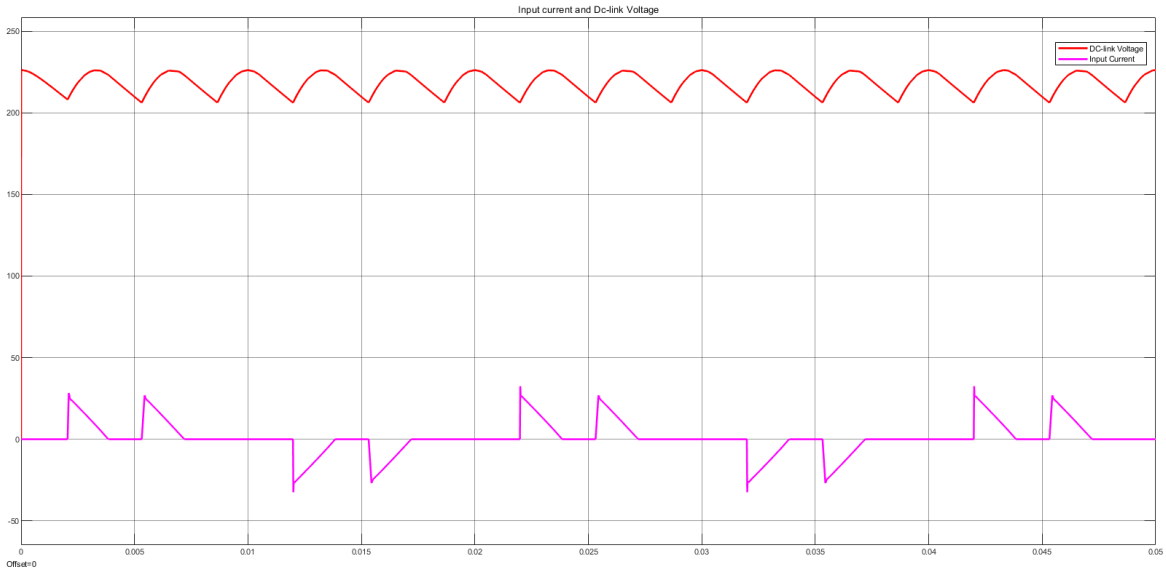

Figure 3. DC-link Voltage and Input Current for Duty Cycle 1

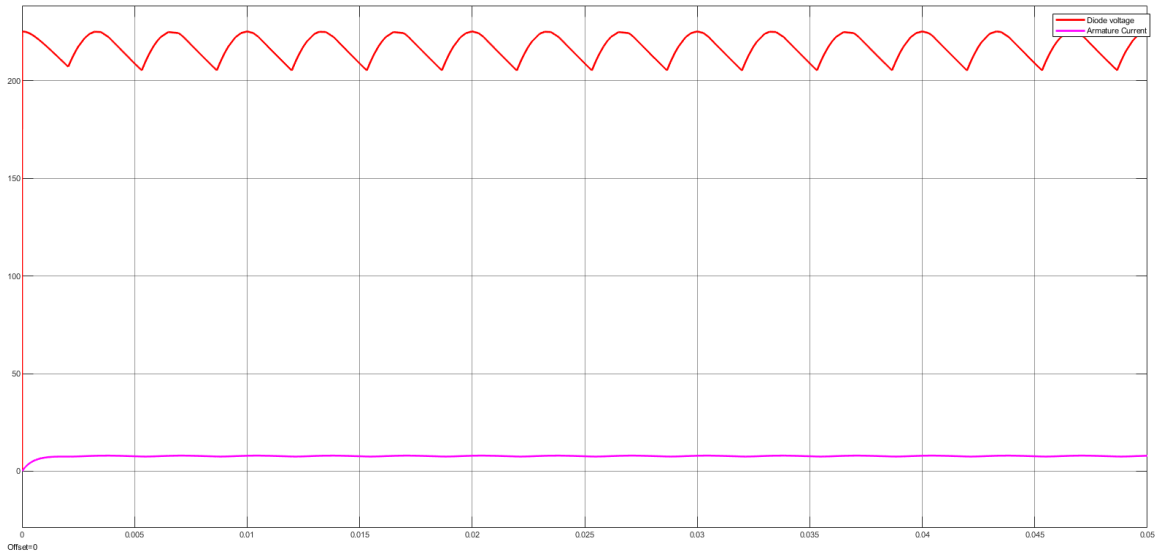

Figure 4. Diode Voltage and Armature Current for Duty Cycle 1

We do not work in duty cycle 1 our maximum operating duty cycle is 0.75 duty cycle. We set duty cycle 0.75 and observe armature current, dc link capacitor voltage, and input current and diode voltage. Results are shown in Figure 5 and Figure 6.

- PWM Duty Cycle = 0.75

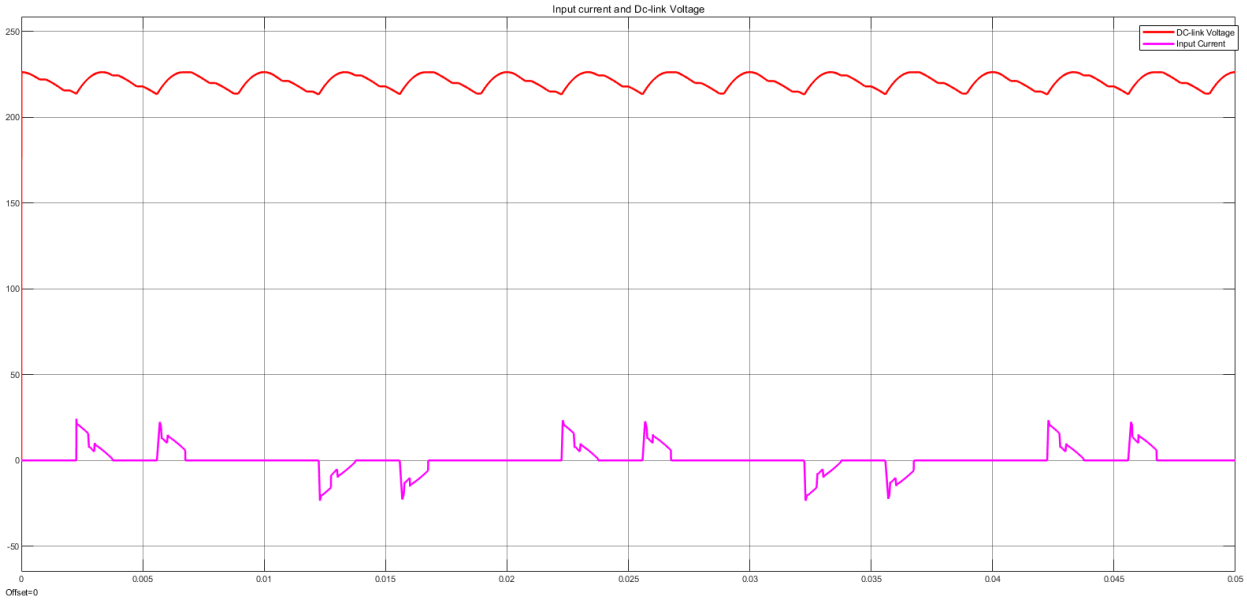

Figure 5. DC-link Voltage and Input Current for Duty Cycle 0.75

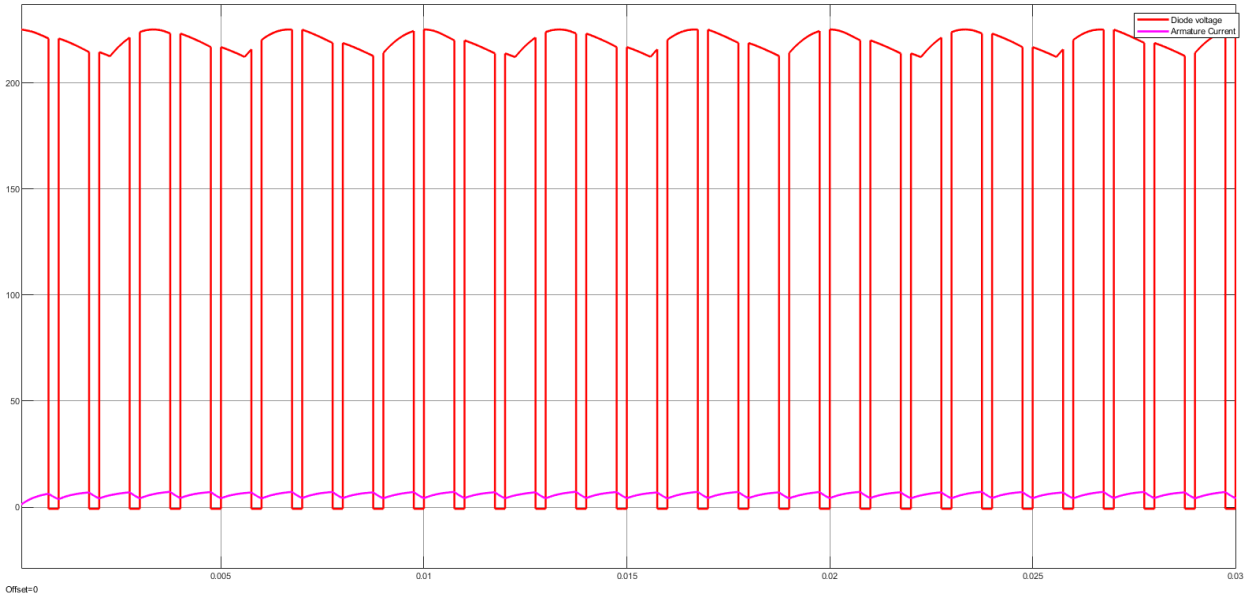

Figure 6. Diode Voltage and Armature Current for Duty Cycle 0.75

Another case is duty cycle is equal to 0.5. We observe armature current, dc link capacitor voltage, and input current and diode voltage. Simulation results are shown in Figure 7 and Figure 8.

- PWM Duty Cycle = 0.5

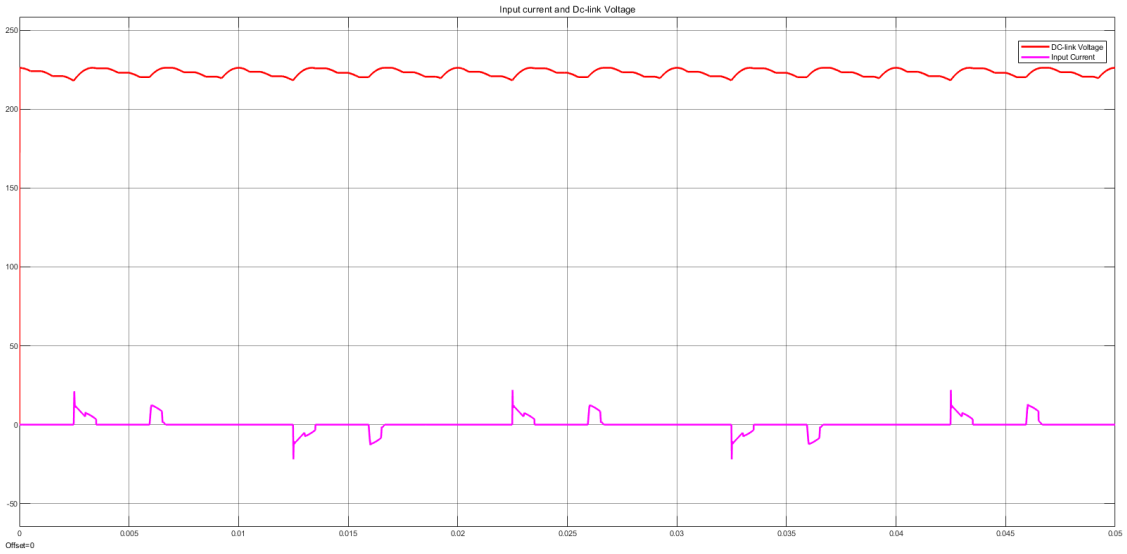

Figure 7. DC-link Voltage and Input Current for Duty Cycle 0.5

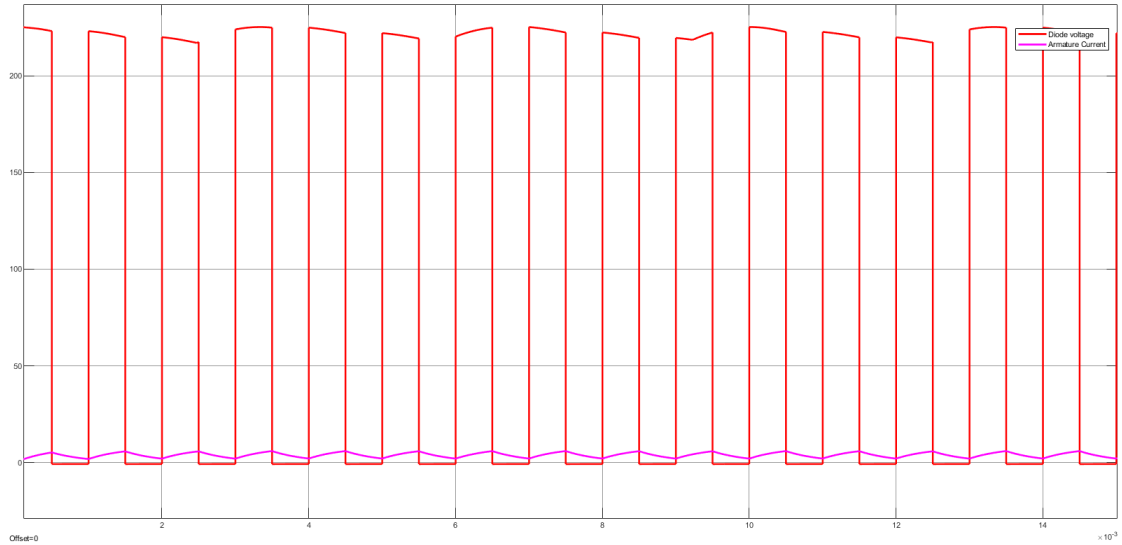

Figure 8. Diode Voltage and Armature Current for Duty Cycle 0.5

We give 0.25 duty cycle to the igbt and observe armature current, dc link capacitor voltage, and input current and diode voltage. Simulation results are shown in Figure 9 and Figure 10.

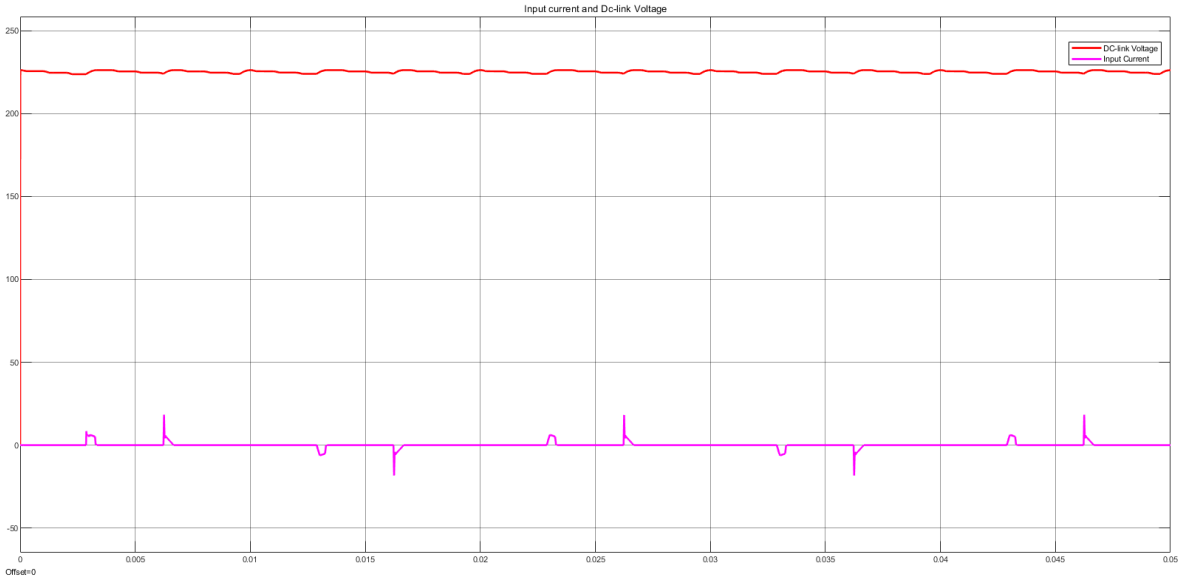

Figure 9. DC-link Voltage and Input Current for Duty Cycle 0.25

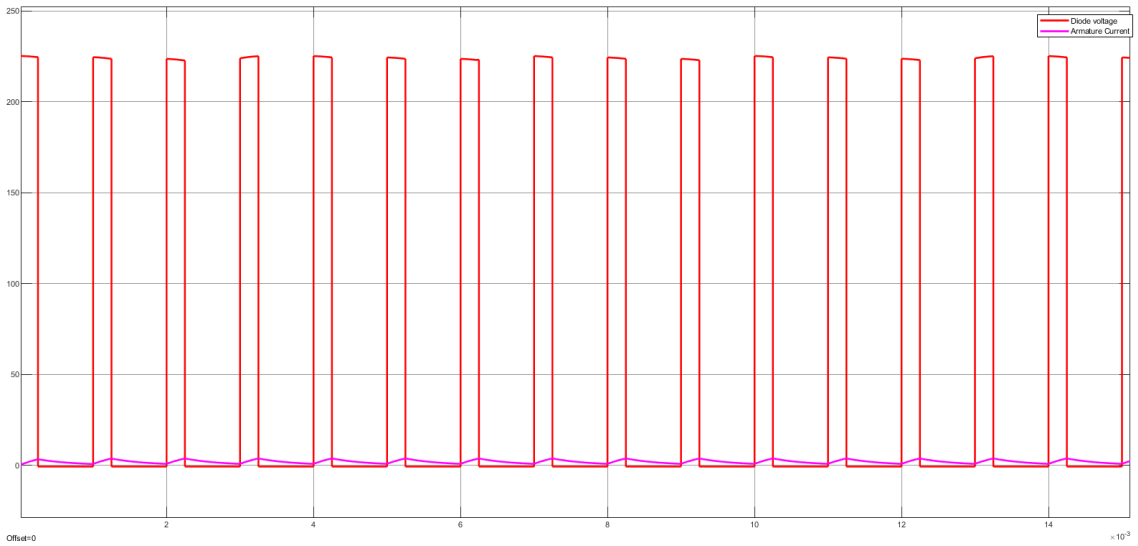

Figure 10. Diode Voltage and Armature Current for Duty Cycle 0.25

Finally we simulate boundary conditions between continuous conduction mode and discontinuous conduction mode. Nearly 0.1 duty cycle current is edge of continuous conduction mode. Simulation results are shown in Figure 11 and Figure 12.

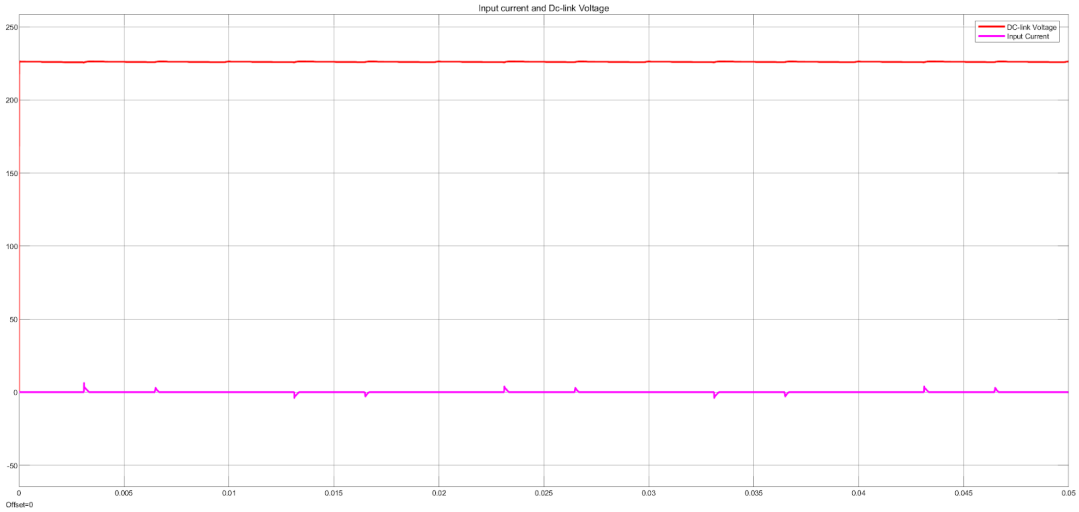

Figure 11. DC-link Voltage and Input Current for Edge of Continuous Conduction Mode

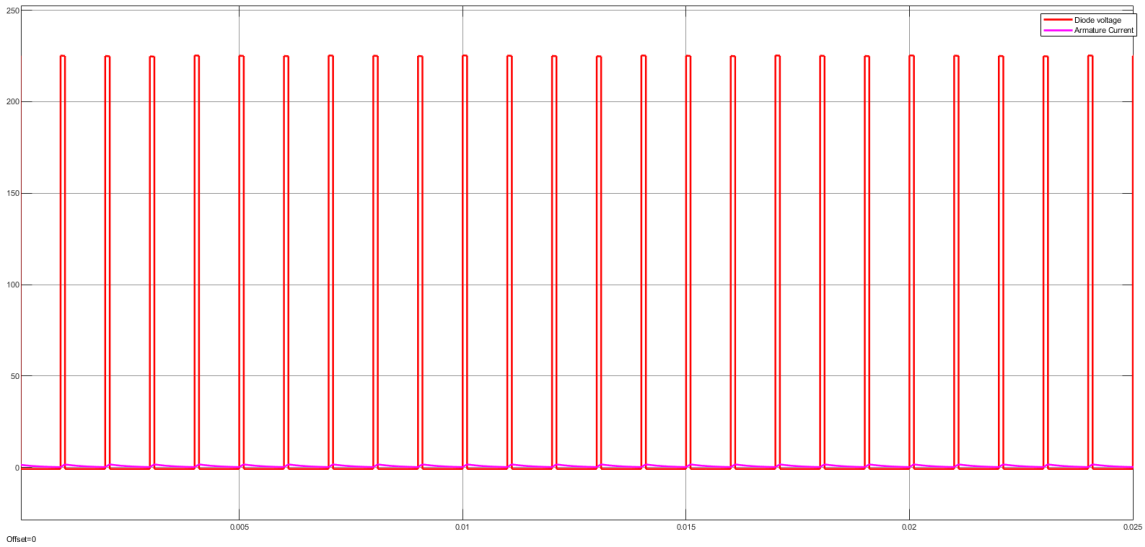

Figure 12. Diode Voltage and Armature Current for Edge of Continuous Conduction Mode

4. COMPONENT SELECTION

Equipment Selection

In this section, the main components selections of the project will be discussed in detail. All components will be told in terms of basic parameters and selected properties.

Although the components has chosen with respect to simulation results, higher voltage/current rating semiconductor devices were used due to lack of time and stock problems of local suppliers. The conservative design approach also affected the component selection process.

Three-Phase Diode Rectifier:

As Dynamic Power team, in order to integrate our project in an easy way and not to get integration troubles, we have decided to use a compact three-phase diode rectifier instead of separate six diodes. The SBR3512 Bridge Diode Rectifier[1] was chosen to use as AC to DC converter in our project. The rectifier can be seen in Figure 13. The basic parameters are listed below for the three-phase diode rectifier;

- Repetitive Peak Reverse Voltage: 1200V
- Average Rectified Output Current: 35A
- Junction Temperature: -55-155 C
- Max. forward surge current: 400A
- Forward voltage drop: 1.2V
- Thermal resistance (junction to case plus case to heatsink): 1.35K/W

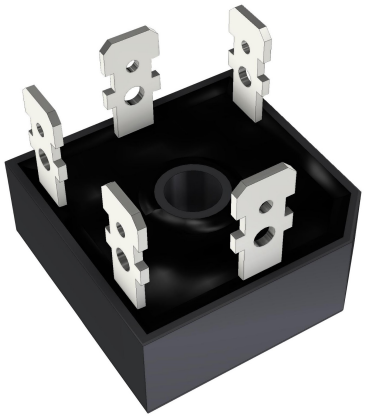

Figure 13. SBR 3512 Bridge Diode Rectifier

IGBT:

In order to switch our system IXGH 24N60C4D1 High-Gain IGBT[2] is used. The main reason of selection this component is that IGBT is more reliable in terms of rush current while comparing with MOSFET. On the other side, the IGBT is a little more expensive than MOSFET. The switching frequency of IGBT is selected 1 kHz by considering all devices heat capacitances and switching losses such as free wheeling diode loss. The IXGH 24N60C4D1 High-Gain IGBT is shown in Figure 14 and important parameters are shown in Figure 15.

Figure 14. IXGH 24N60C4D1 High-Gain IGBT

Symbol	Test Conditions	Maximum Ratings	
V_{CES}	T_J = 25°C to 150°C	600	V
V_{CGR}	T_J = 25°C to 150°C, R_{GE} = 1MΩ	600	V
V_{GES}	Continuous	±20	V
V_{GEM}	Transient	±30	V
I_{C25}	T_C = 25°C	56	A
I_{C110}	T_C = 110°C	24	A
I_{F110}	T_C = 110°C	18	A
I_{CM}	T_C = 25°C, 1ms	130	A
SSOA (RBSOA)	V_{GE} = 15V, T_{VJ} = 125°C, R_G = 10Ω Clamped Inductive Load	I_{CM} = 48 @ ≤ V_{CES}	A
P_C	T_C = 25°C	190	W
T_J		-55 ... +150	°C
T_{JM}		150	°C
T_{stg}		-55 ... +150	°C
T_L	Maximum Lead Temperature for Soldering	300	°C
T_{SOLD}	1.6 mm (0.062in.) from Case for 10s	260	°C
M_d	Mounting Torque	1.13/10	Nm/lb.in.
Weight		6	g

Figure 15.: Ratings of IGBT

Freewheeling diode:

The DSEI30-12A Fast Recovery Epitaxial Diode (FRED)[3] is used as the freewheeling diode in the Buck converter in Figure 16.

Figure 16. DSEI30-12A Fast Recovery Epitaxial Diode (FRED)

The switching characteristics, current/voltage ratings are considered while selection of diode especially reverse breakdown voltage and switching frequency are based. The basic parameters is shown in Figure 17.

Symbol	Test Conditions		Maximum Ratings	
I_{FRMS}	$T_{VJ} = T_{VJM}$		70	A
I_{FAVM} ①	T_C = 85°C; rectangular, d = 0.5		26	A
I_{FRM}	t_P < 10 µs; rep. rating, pulse width limited by T_{VJM}		375	A
I_{FSM}	T_{VJ} = 45°C;	t = 10 ms (50 Hz), sine	200	A
		t = 8.3 ms (60 Hz), sine	210	A
	T_{VJ} = 150°C;	t = 10 ms (50 Hz), sine	185	A
		t = 8.3 ms (60 Hz), sine	195	A
I^2t	T_{VJ} = 45°C	t = 10 ms (50 Hz), sine	200	A^2s
		t = 8.3 ms (60 Hz), sine	180	A^2s
	T_{VJ} = 150°C;	t = 10 ms (50 Hz), sine	170	A^2s
		t = 8.3 ms (60 Hz), sine	160	A^2s
T_{VJ}			-40...+150	°C
T_{VJM}			150	°C
T_{stg}			-40...+150	°C
P_{tot}	T_C = 25°C		138	W
M_d	Mounting torque with screw M3		0.45/4	Nm/lb.in.
	Mounting torque with screw M3.5		0.55/5	Nm/lb.in.
Weight			6	g

Figure 17. Ratings of free-wheeling diode

DC Link Capacitor:

A smoothing capacitor which has value of 470uF that can resist 400V continuous voltage is used for the project.

Figure 18. 470 uF Capacitor

Optocoupler:

The TLP250 is used as an isolator between Arduino and gate driver circuit. This component is explained in detail in gate driver part of report.

Arduino:

As a microcontroller, Arduino Nano is used to generate Pulse Width Modulation signal. The main purpose of using Arduıno is to get feedback from motor rpm and generate automotive PWM for constant speed. But due to the some technıcal problems we could not get feedback from motor. This component is also explained in Gate Driver section in detail.

Heatsink + Fan:

In order to prevent damages from overheating and decrease power dissipation we have used three different heatsink for three-phase diode rectifier, IGBT and freewheeling diode. Theoretical power loss calculations were done according to datasheets of semiconductor devices and switching frequency. While choosing heatsink dimension we have considered heatsink calculation parameters which are shown in Figure 20. After experimental results, an active cooling with a DC fan was added to deal with high temperature.

Figure 19. Heatsink

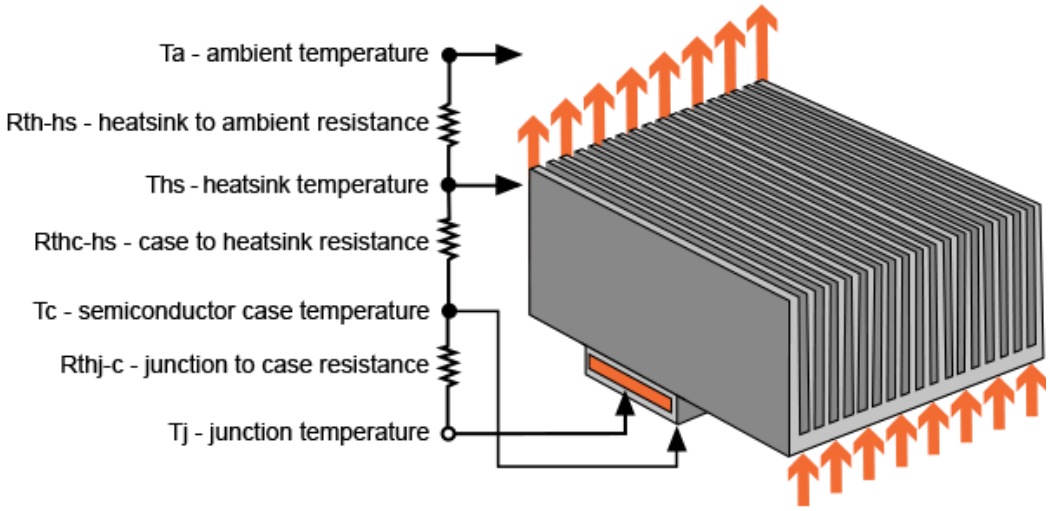

Figure 20. Heatsink Calculation Parameters

Three-Phase Heatsink Calculation

We model heat transfer as a circuit. We take data from data sheet and calculate the minimum heatsink. Three-phase diode rectifier operating temperature are between -55 C to 155. For maximum operating condition we choose 110 C. Maximum current is 10A on the full load and power loss is 30W in that condition. Our model circuit is shown in Figure 21.

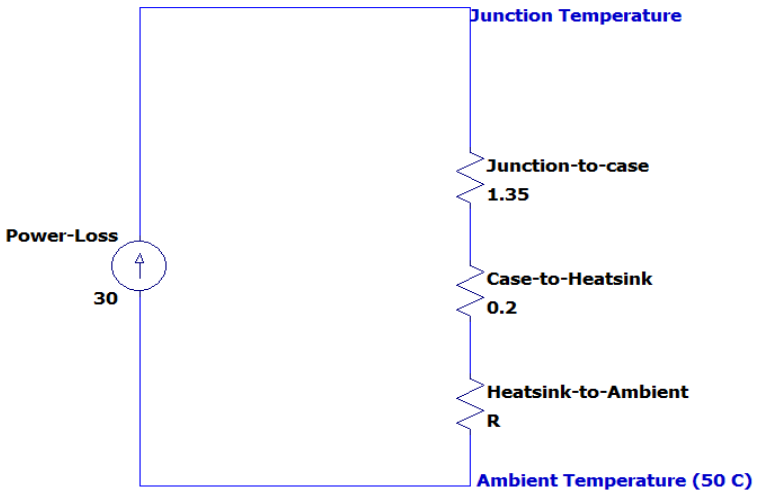

Figure 21. Three-phase Diode rectifier Heat Transfer circuit

$$Req = \frac{Junction\ Temperature - Ambient\ Temperature}{Power - Loss}$$

$$Req = \frac{110-50}{30} = 2$$

So heatsink to ambient thermal resistance equal to

2-1.55=0.45

Igbt Heatsink Calculation

Igbt junction temperature increase very quickly because there is switching loss and when duty cycle increase most of power flow through the igbt. In the 2kW operation Igbt maximum power loss is 90W. This valu taken from datasheet of Igbt. Junction to case thermal resistance of Igbt is 0.342 C/W and case to heatsink thermal resistance is 0.21 C/W. Our heat transfer model is shown in Figure 22.

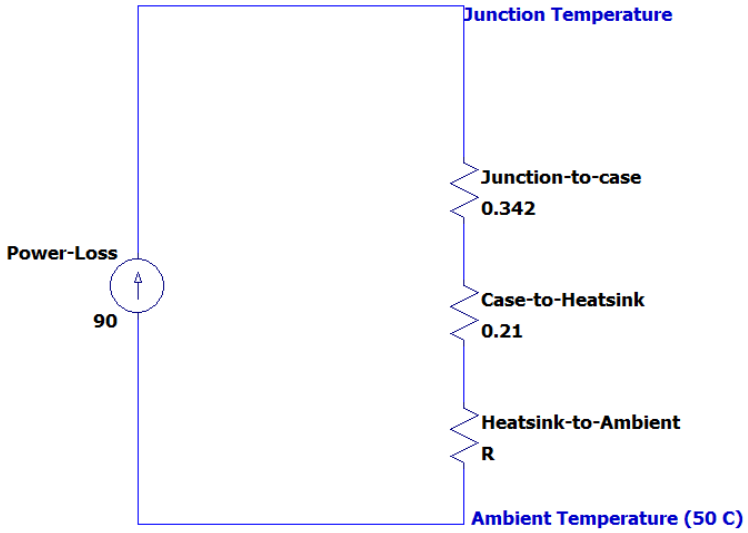

Figure 22. Igbt Heat Transfer Circuit

$$Req = \frac{Junction\ Temperature - Ambient\ Temperature}{Power - Loss}$$

$$Req = \frac{120-50}{90} = 0.77$$

So heatsink to ambient thermal resistance equal to

0.77-0.552=0.218

Diode Heatsink Calculation

Diode junction temperature increase slowly because current flow through the Igbti. In 2kW load operation most of the current flow through the igbt diode current is so small. Equivalent heat transfer circuit is shown in Figure 23.

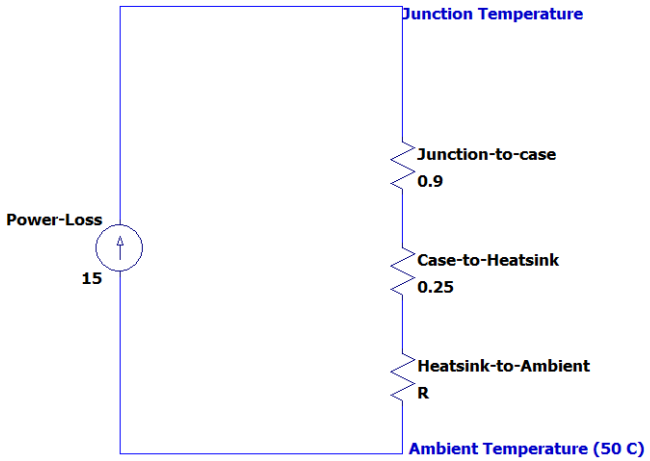

Figure 23. Diode Heat Transfer Circuit

$$Req = \frac{Junction\ Temperature - Ambient\ Temperature}{Power - Loss}$$

$$Req = \frac{110-50}{15} = 4$$

So heatsink to ambient thermal resistance equal to 4-1.15=2.85

Cost Analysis

MAIN COMPONENTS	COST(TL)
3-Phase diode rectifier module	55
IGBT	35
Freewheeling diode	18
Capacitor	25
Arduino	30
Optocoupler	13
Heat Sink(x3)	3x8
5V DC FAN	15
PCB	20
Plastic box	25

Table 1. Cost Analysis Table

In additiın to above components, we have used resistors, ceramic capacitors, battery, connectors, sockets, thermal paste and cables.

5. TEST RESULTS

We have tested all submodules after buying them. We control equipment working. Firstly, we tested three-phase full bridge rectifier and dc link capacitor and buck converter in R-L load. We give three-phase input and observe output in the oscilloscope. We also tested arduino and optocoupler in switching operation. Test results are shown in Figure 24.

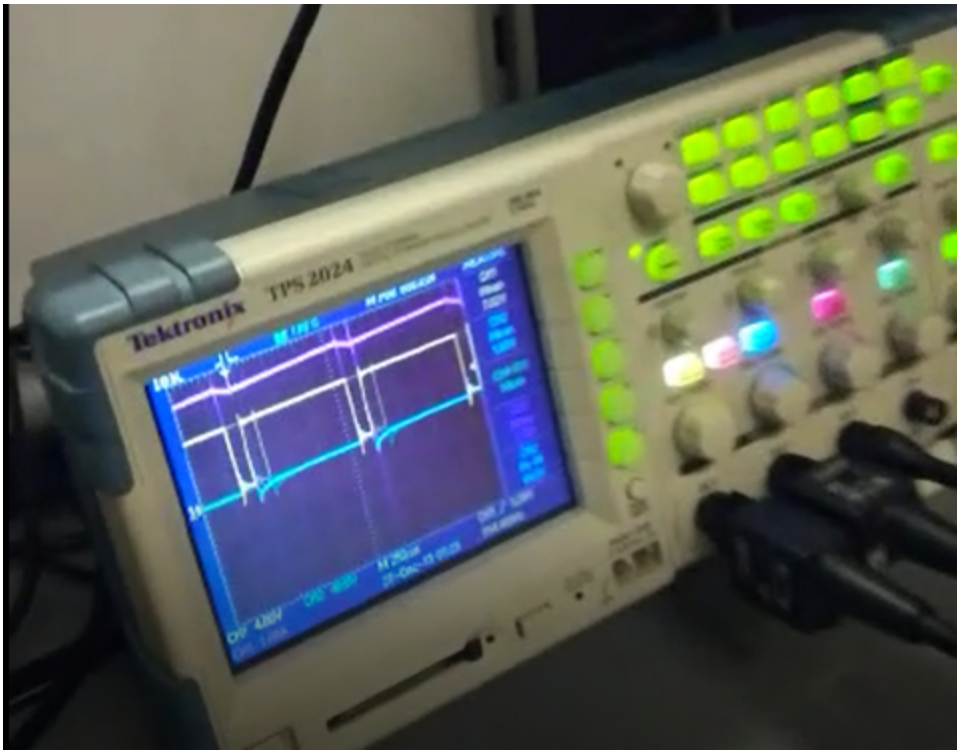

Figure 24. PWM and Output voltage of buck converter in R-L load

We also change the gate signal of the IGBT generated by the Arduino and observe output voltage of buck converter in R-L load. Results of the test is shown in Figure 25.

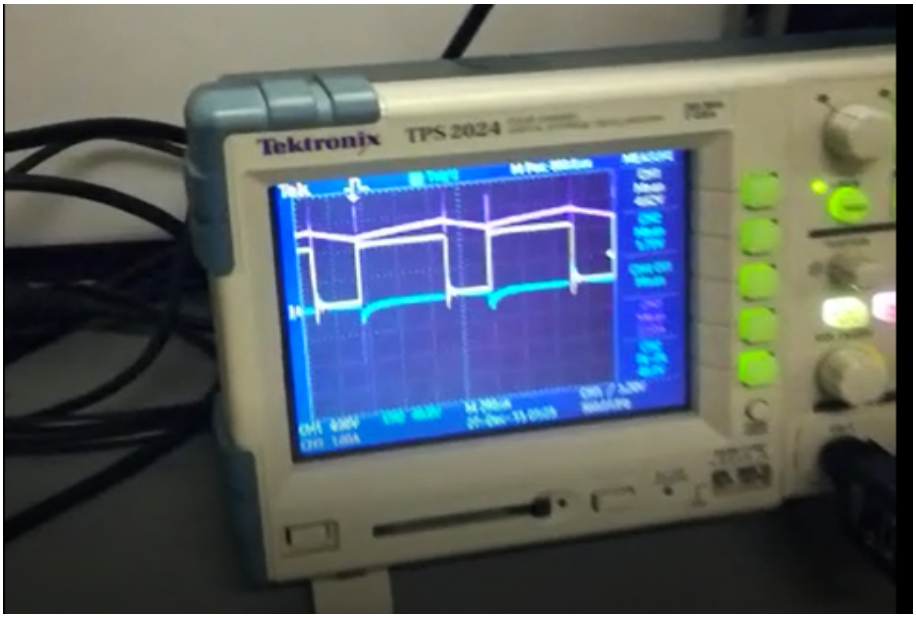

Figure 25. Output of buck converter and optocoupler output in R-L load

Finally, we were ready to test our circuit with motor. Results are same as R-L load but there is inrush current at the starting. But we use button and arduino for soft starting when you push button PWM increases slowly so inrush current can not damage the our circuit. Test results are expected. Firstly we drive motor with no-load. We observe current PWM. Test results are shown in Figure 26 .

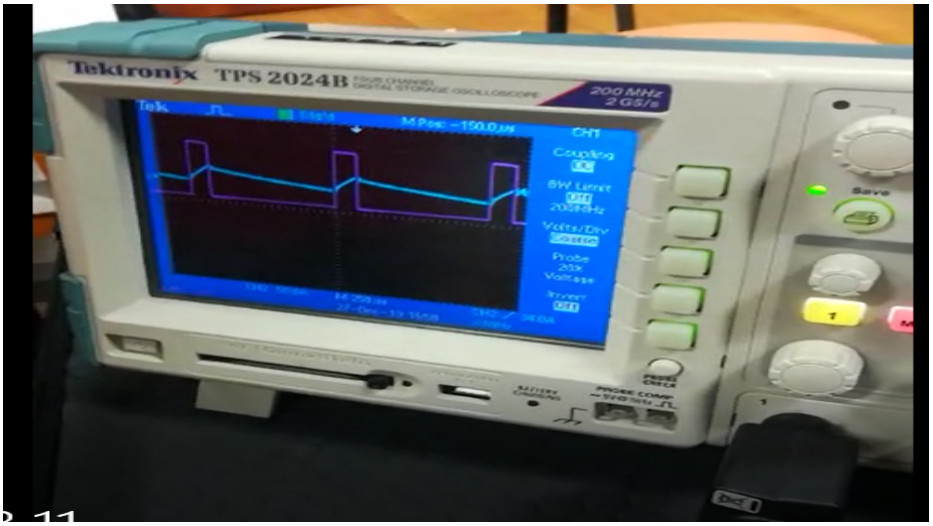

Figure 26. PWM and motor current.

We also observe boundary between continuous conduction mode and discontinuous conduction mode. Edge of continuous conduction mode test results is shown in Figure 27.

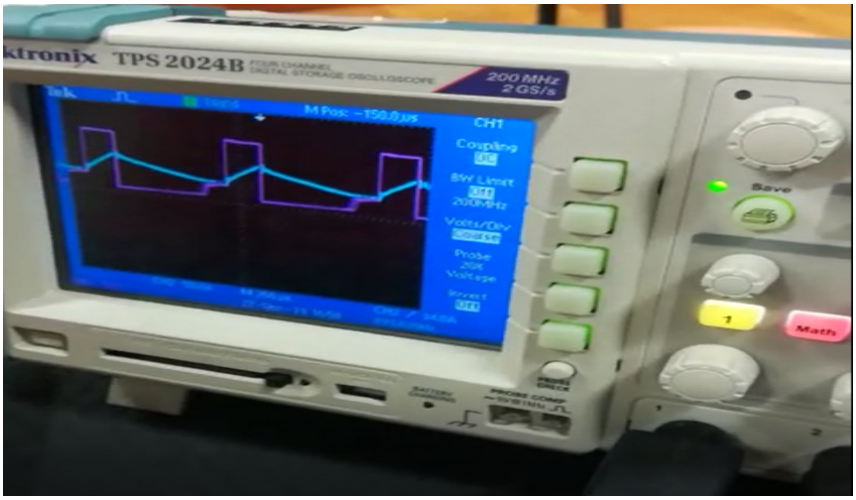

Figure 27. PWM and motor current at the edge of continuous conduction mode.

Dc motor coupled with generator so we can load the motor with kettle. For industrial bonus our circuit must supply 2kW power to the load. For testing industrial design we set output voltage to the 180V with increasing PWM. After that we load the kettle. First observation is .

speed of dc motor decreases due to the load. Also we observe that motor current is also increases. Test results shown in Figure 28.

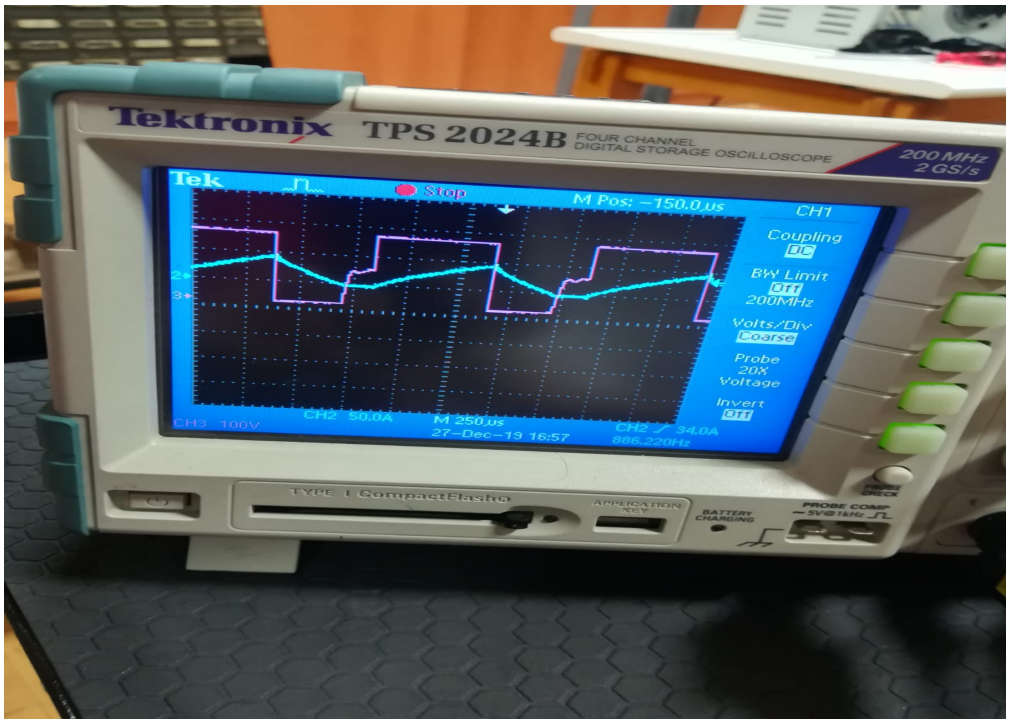

Figure 28. PWM and motor current on the load.

In the test we also observed the power current and voltage we took photos of measurements. We supply 2kW power to the load and supply this load five minute. Current is nearly 10A and voltage is 190 V rms. Test results are shown in Figure 29.

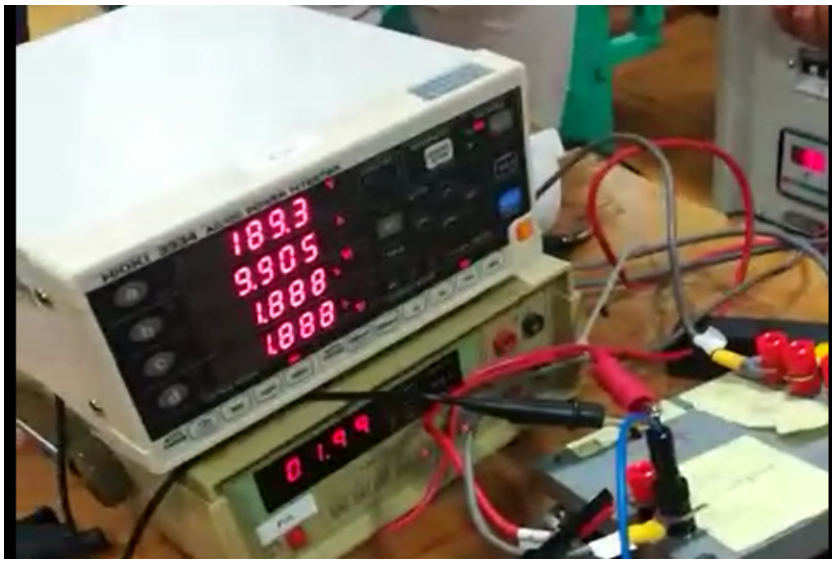

Figure 29. Motor voltage current and power

According to experimental results, the efficiency of the system is 95-96 percent which is similar to our theoretical results.

We also measure the heat of the devices because every device has operating temperature so we can not pass this operation temperature. Highest temperature is igb because there is switching loss and on the full load we work %80 duty cycle so most of the current pass through the igbt. In the test we measure the igbt temperature it's temperature increase very quickly. Test results shown in figure 30. As we see operating temperature is 95 C under 2kW load.

Figure 30. Temperature of Igbt in 2kW Load Operation

6. CONCLUSION

The purpose of this project is to design and implement a DC motor driver which is able to operate under 2kW load at least two minutes. To meet these requirements, 3-phase diode rectifier plus buck converter topology is selected due to simpler and more robust behaviour. The product can be divided into three main parts which are AC to DC conversion, step-down (buck) converter and controller.

In design process of the project, different simulation tools such as Simulink, Proteus and Matlab were used to identify the fundamental properties of the product. Component selections and topology decisions were guided by the results of these simulations. These results are also used to optimize the performance of the product without burning anything. Moreover, KiCad was also used to design the PCB layout of circuit to achieve compact and industrial design.

In the laboratory, we have faced different types of difficulties and Murphy's Law. The most challenging ones are the controller unit and wrong pcb layout. The tight schedule also pushed us to our limits. In spite of these difficulties, nothing did not blow up and we caught the demo calendar.

In conclusion, we are able to practice our theoretical knowledge by working on project. We learned the converter topologies, the importance of isolation, manufacturability and time management. As a result, this hardware project encouraged us to experience the practical application of our theoretical knowledge .

All these processes are recorded and accessible in below link .

https://www.youtube.com/watch?v=q_Ks-do5HX4&featureyoutu.be

APPENDIX A: Printed Circuit Board (PCB) Layout

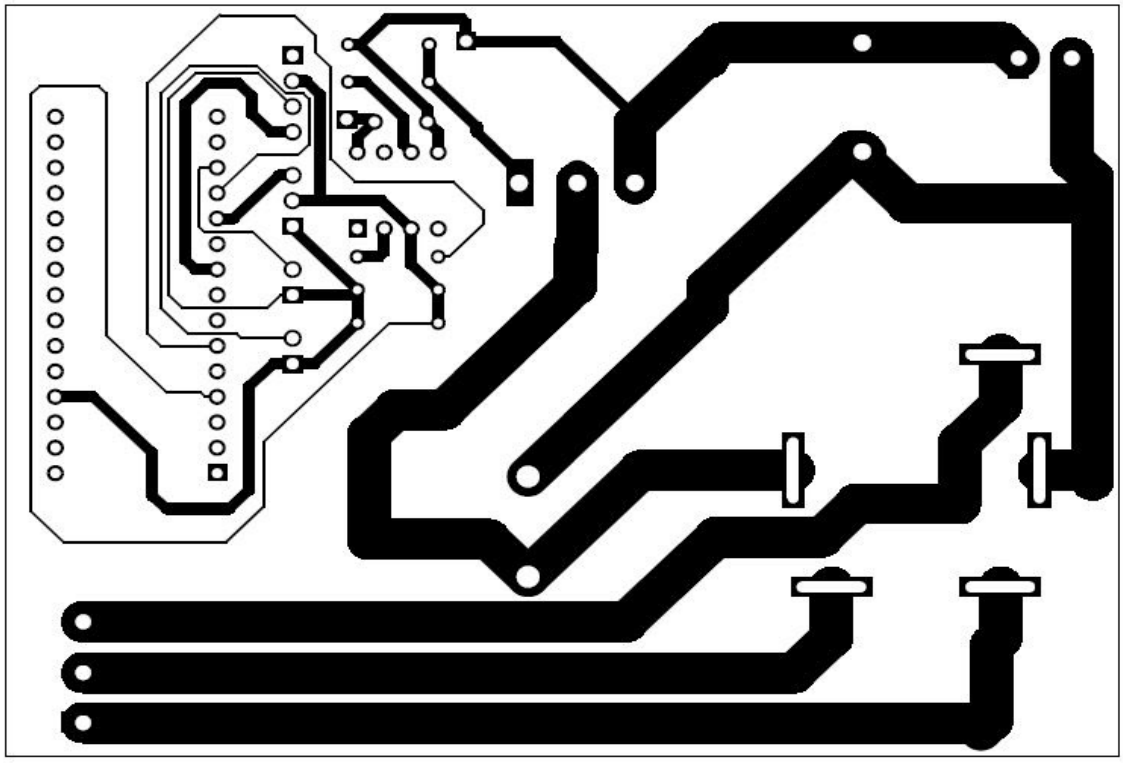

Figure 31. PCB Layout

APPENDIX B: Source Code of Arduino Gate Driver and Feedback Loop

```
// EE 463 Motor Driver Pwm and Feedback Arduino Code
const int speedUp = 12;     // the number of the pushbutton pin
const int speedDown = 11;     // the number of the pushbutton pin
const int optopin =  3;      // the number of the optopin pin
const int maxspeed = 160;
const int minspeed = 0;
int hall_pin = 0;
unsigned long t = 0;
unsigned long lastmillis = 0;
float rpm = 0;
float a = 0;
float b = 0;
int desired_rpm = 0;
int duty = 0;
int c = 0;
float rpm1 = 0;
float rpm2 = 0;
float rpm3 = 0;
float rpm4 = 0;
float rpm5 = 0;
volatile float rpm_overall = 0;
// set number of hall trips for rpm_val_val reading (higher improves accuracy)
float hall_thresh = 100.0;
int d = 0;
// variables will change:spd
volatile int rpmcount = 0;
unsigned long t_prev = 0;
unsigned long t_diff = 0;
float e = 0;
int ti = 0;
int spd = minspeed;
int interval = 1;
int deger;
void setup() {
```

```
void setup() {
  // initialize the opto pin as an output:
  pinMode(optopin, OUTPUT);
  // initialize the pushbutton pins as an input:
  pinMode(speedUp, INPUT);
  pinMode(speedDown, INPUT);
  attachInterrupt(hall_pin, control, FALLING);
  Serial.begin(9600);
  TCCR2B = (TCCR2B & 0b11111000) | 0x03; //980.39 [Hz]
}


void loop()
{

  // button code
  if (digitalRead(speedUp) == HIGH && spd < maxspeed)
  {
    spd = spd + interval;
    d = 0;
    //if the spd Up button is pushed, add one degree of spd to the current level
    Serial.println(spd); //for debugging purposes
    //Serial.println(rpm_overall);
  }
  if (digitalRead(speedDown) == HIGH && spd > 0)
  {
    spd = spd - interval;
    d = 0;
    //if the spd Down button is pushed, subtract one degree of spd from the current level
    Serial.println(spd); //for debugging purposes
    //Serial.println(rpm_overall);
  }
  delay(100);
  analogWrite(optopin, map(spd, 0, maxspeed, 0, 225));
  //this code maps the max spd constant to the max LED spd
```

```
  // closed loop feeed back code

  if (rpm2 > 300) {
    a = 1;
    Serial.println("a=1");
  }
  if (a == 1) {
    while (rpm2 > 400  && digitalRead(speedDown) == LOW && a == 1) {
      spd = spd - interval;
      Serial.println("azalt");
      Serial.println("rpm:  ");
      Serial.println(rpm2);
      rpm2 = rpm1();


      Serial.println(rpm2);
    }
    while (rpm2 < 300 && digitalRead(speedDown) == LOW && a == 1) {

      spd = spd + interval;
      Serial.println("artır");
      Serial.println("rpm:  ");

      Serial.println(rpm2);
      rpm2 = rpm1();
    }
  }
}
```

```
// rpm reading code
void control () {
  t_prev = t;
  t = millis();
  t_diff = t - t_prev;
  b = (float) t_diff;
  a = 1000 / b;
  rpm = 60 * a;
  Serial.println(rpm);
  if (rpm > 0 && rpm < 1500) {

    rpm5 = rpm4;
    rpm4 = rpm3;
    rpm3 = rpm2;
    rpm2 = rpm1;
    rpm1 = rpm;



    rpm_overall = (rpm1 + rpm2 + rpm3 + rpm4 + rpm5) / 5;
    Serial.println(rpm_overall);
  }

}
```

APPENDIX C: Source Code of Arduino Gate Driver and Feedback Loop(PID)

463_controller

```
#define Motor 9          //  assign ninth PWM output pin as Motor
#define hall_sensor 0         //  assign analog input 0 pin as hall_sensor

// PID parameters of Motor will be decided later
double kp_h = 970;       // Motor Kp
double ki_h = 0.12;       // Motor Ki
double kd_h = 0.03;       // Motor Kd


unsigned long currentTime, previousTime = 0;
double elapsedTime;
double error;
double lastError = 0;        // at first we set last error to zero
double output;
double cumError, rateError;


float Temp;
float Speed;  // Current Speed
float Sset=30;  // Set Point of Speed
void setup(){

  pinMode(hall_sensor,INPUT);        // assign hall_sensor pin as input pin


  pinMode(Motor,OUTPUT);           // assign Motor pin as output pin

  Serial.begin(9600);

}


float getSpeed() {
  Serial.println("**********hall_sensor********");
  float Temp = log(10000.0*((1024.0/analogRead(hall_sensor)-1)));
```

463_controller

```
  Serial.begin(9600);

}


float getSpeed() {
  Serial.println("**********hall_sensor********");
  float Temp = log(10000.0*((1024.0/analogRead(hall_sensor)-1)));
  //         =log(10000.0/(1024.0/RawADC-1)) // for pull-up configuration
  Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
  Temp = Temp - 273.15;            // Convert Kelvin to Celcius
  Serial.print("Temp C: ");
  Serial.println(Temp); //send the data to the computer
  return Temp;
}



void loop()
{

  float Speed = getSpeed();   // Speed read
 Serial.print(Speed);
 Serial.print(",");
Serial.println(Sset);

  if(Sset - Speed>0){
            output = computePID_Motor(Speed);
            if(output>1023)
            output=1023;      // Since the PWM output can be up to 1023, we have set that point as the highest value we can give from the outpoint.

            analogWrite(Motor, output);
  }else{
            output = computePID_cooler(Speed);
            output=-output;
```

463_controller

```
            if(output>1023)
            output=1023;      // Since the PWM output can be up to 1023, we have set that point as the highest value we can give from the outpoint.

            analogWrite(Motor, output);
  }else{
            output = computePID_cooler(Speed);
            output=-output;
            if(output>1023)
            output=1023;   //  Since the PWM output can be up to 1023, we have set that point as the highest value we can give from the outpoint.
            output= map(output,0,1023,80,1023);
            analogWrite(Motor, 0);

  }


  delay(1);


}

double computePID_Motor(double Tnow){
        currentTime = millis();                //get current time
        elapsedTime = (double)(currentTime - previousTime);        //compute time elapsed from previous computation

        error = Sset - Tnow;                                 // determine error

        cumError += error * elapsedTime;               // compute integral
        rateError = (error - lastError)/elapsedTime;   // compute derivative
        double out = kp_h*error + ki_h*cumError + kd_h*rateError;               //PID output

        lastError = error;                                  //remember current error
        previousTime = currentTime;                         //remember current time

        return out;                                         //have function return the PID output
}
```

REFERENCES

[1]http://www.farnell.com/datasheets/2864130.pdf?_ga=2.226296671.494831948.1579023259-1719140146.1572778914

[2] https://pdf1.alldatasheet.com/datasheet-pdf/view/415136/IXYS/IXGH24N60C4D1.html

[3]https://pdf1.alldatasheet.com/datasheet-pdf/view/235059/IXYS/DSEI30-12A.html

[4]