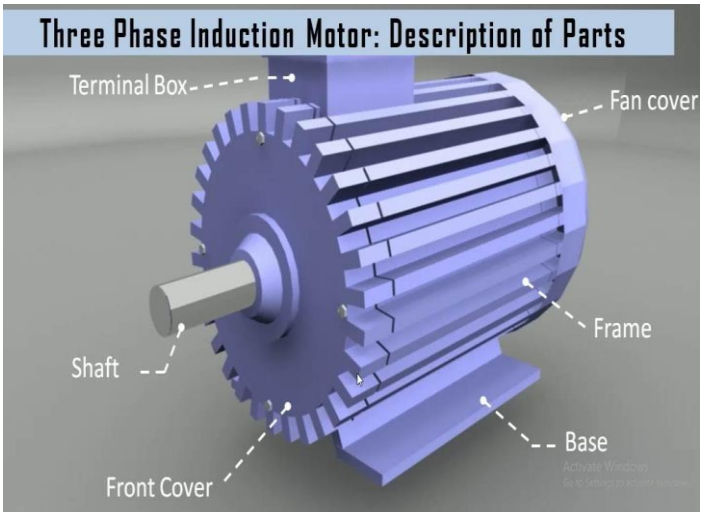


Speed Control of 3 Phase Asynchronous Motor using PID Controller

VIBHUSHA K.K.L
180661B
Final Report
Electrical Engineering
University of Moratuwa

TABLE OF CONTENTS

Abstract

This report is an experimental approach of control methods of an induction motor by VSI control(variable frequency and voltage drive) by using a PID controller. The modelling is done by using basic mathematical principles and by using state space representation. The induction motor is modelled by d-q axis dynamic modelling and also to derive torque speed characteristics we used sinusoidal based modelling approach.

The 3-phase variable frequency drive is used to control the speed of the motor by using a PID controller to change the frequency due to error value between the desired output and the real time output.

The components I used to make a hardware simulation is

1.Arduino uno board
2.Mosfet driver
3.IR sensor module
4.ISO138 oscilloscope

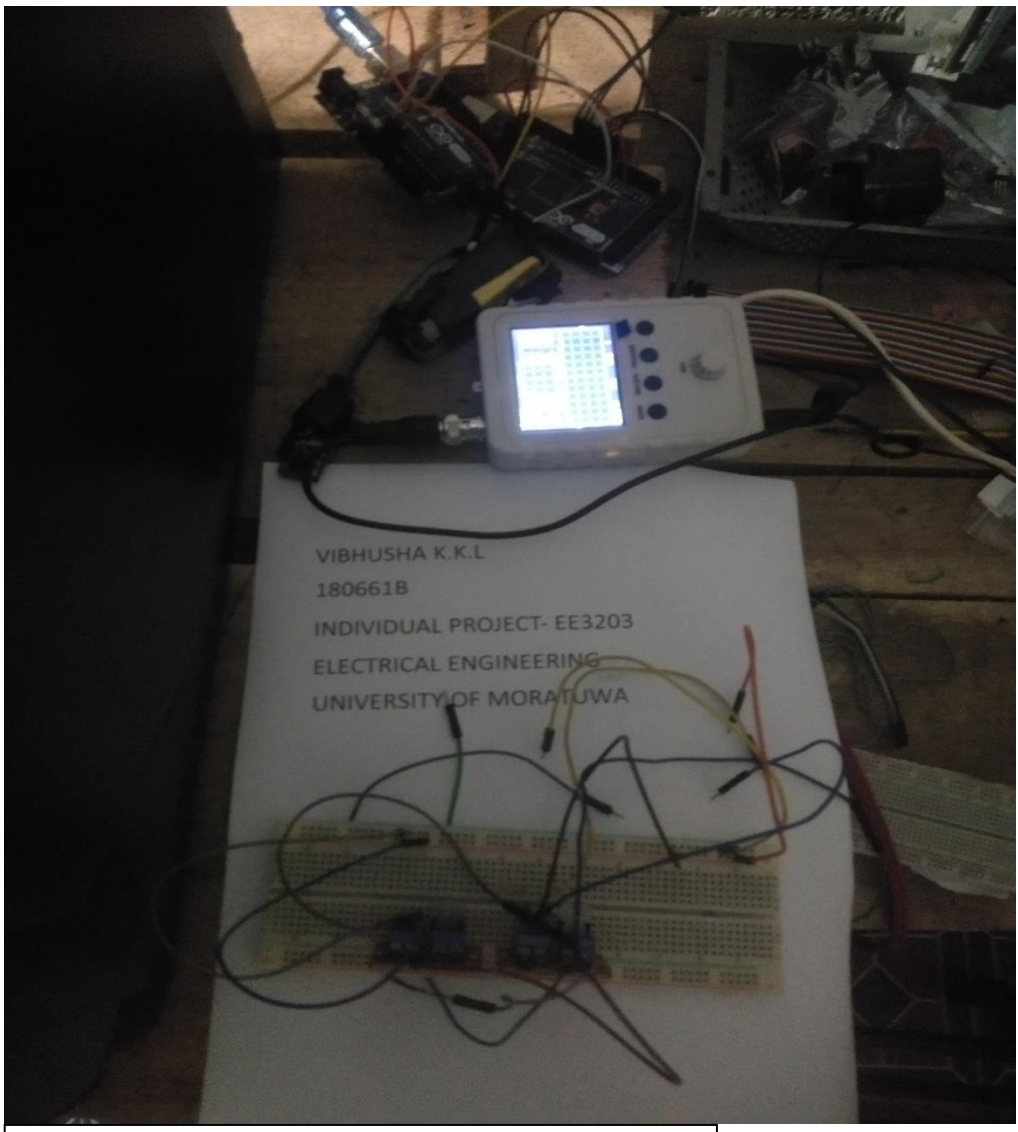


Fig: practical Implementation Equipment

INTRODUCTION

For many industrial applications the induction motor is used due to following reasons.

- Simple in construction.
- Economical
- less maintenance is required.
- Highly efficient.
- Low maintenance cost.
- Self starting(unlike reluctance motors).
- Robust.

But due to phenomenon like back EMF losses and other causes the controlling of the system is somewhat nonlinear.

- ➨ So it is very difficult to control speed using mathematical approximations alone. So we have to create a complex system to control the system.
- ➨ In this project I used scalar V/F control method.
- ➨ Also to generate the appropriate voltage and frequency levels I used a PID controller.

A 3 phase Asynchronous motor has a stator and a rotor.

Stator of 3 Phase Asynchronous motor
The **stator** of this motor is made up of numbers of slots to construct a 3 phase winding circuit which is connected to an inverter circuit controlled by MOSFETS or IGBTs.

Rotor of 3 Phase Asynchronous motor
The **rotor** contains of a cylindrical laminated core with parallel slots with conductors. The slots are not 100% parallel to the axis of the shaft a little skewed because this reduces magnetic humming noise and can avoid vibrations of the motor.

MOTIVATIONS

The 3 phase Asynchronous motor is relatively low priced motor with a simple construction architecture. But when constant speed applications are needed due to simplicity in the field the PMSM or DC motors are used. This motor is highly efficient due to absence of brushes and other friction losses. If we optimize the speed control of the motor we can use this motor for almost any application.

This project also motivated me to learn concepts of induction motor ,PWM techniques like SPWM, SVPWM controlling systems like PID control ,fuzzy logic control and applications of neural networks to projects. Also it helped me to understand pros and cons of each controlling method.

OBJECTIVES AND SCOPE

Objective of this system is to develop a highly sophisticated and accurate motor speed controller for an induction motor by using closed loop control methods.

1.Scalar V/F control Method

2.Field Oriented (FOC) control (Not fully implemented)

Why is that the Induction motor needs a sophisticated controlling mechanism?

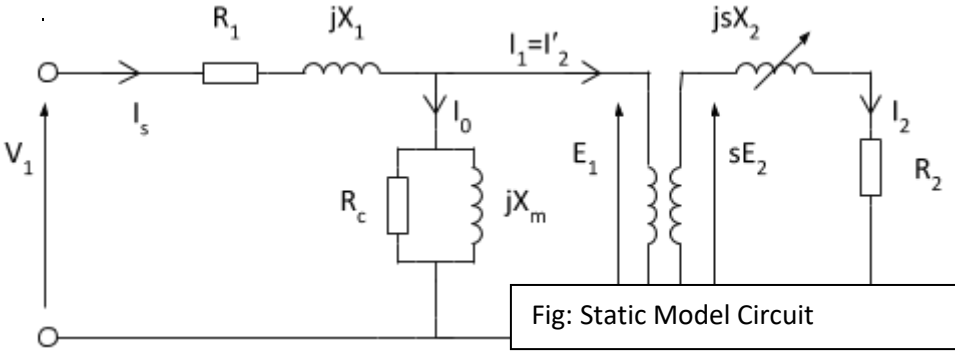


Fig: Static Model Circuit

Torque-Speed/Torque-Slip Characteristics derivation of a 3 phase asynchronous motor.

$$\tau \propto \phi |I_2| \cos\theta_2$$

$$\phi \propto E_1$$

$$E_2 \propto E_1$$

$$\phi \propto E_2$$

Applying equations to the right side of the equivalent circuit,

$$\cos\theta_2 = \frac{R_2}{|Z_2|} = \frac{R_2}{\sqrt{(R_2)^2 + (sX_2)^2}}$$

$$I_2 = \frac{sE_2}{Z_2}$$

$$|I_2| = \frac{sE_2}{\sqrt{(R_2)^2 + (sX_2)^2}}$$

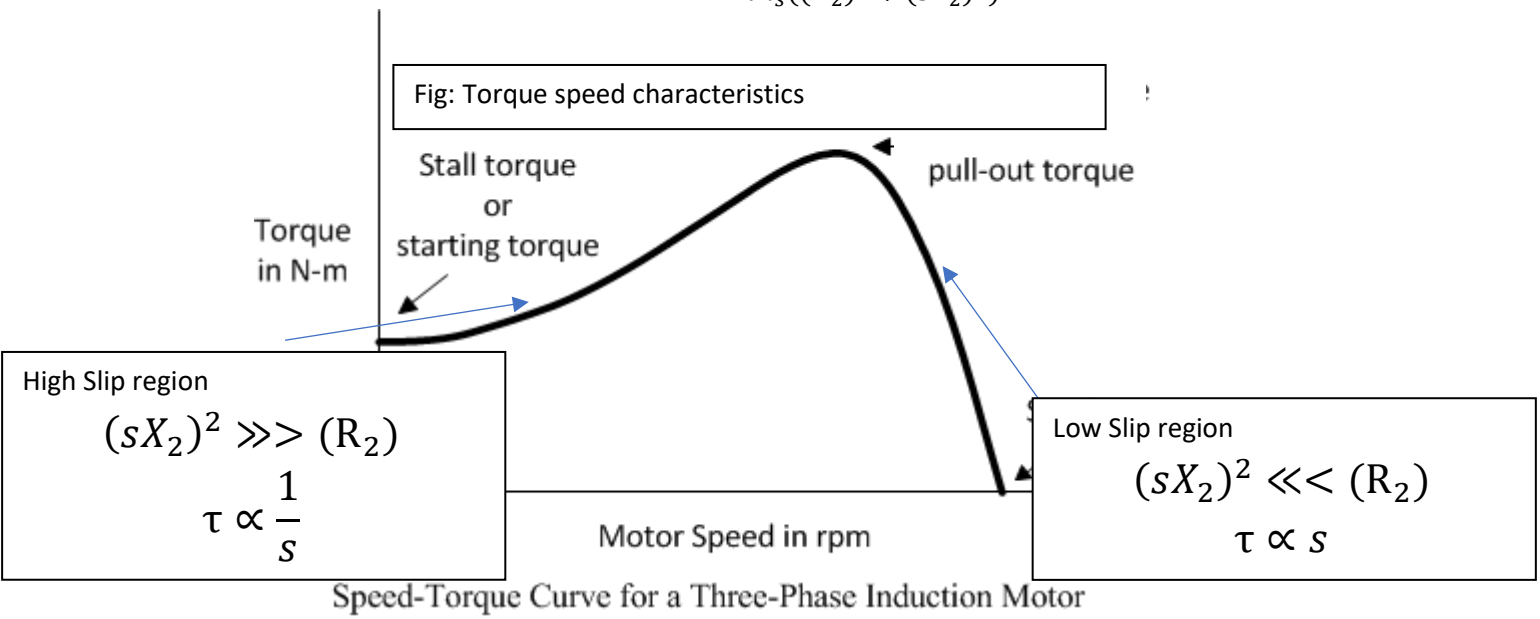
$$\tau \propto \frac{sE_2}{(R_2)^2 + (sX_2)^2}$$

proportionality constant can be derived by writing another torque equation for losses in the circuit.

$$K = \frac{3}{2\pi n_s}$$

$$\tau = \frac{3sE_2}{2\pi n_s((R_2)^2 + (sX_2)^2)}$$

Fig: Torque speed characteristics



High Slip region

$$(sX_2)^2 \gg (R_2)$$

$$\tau \propto \frac{1}{s}$$

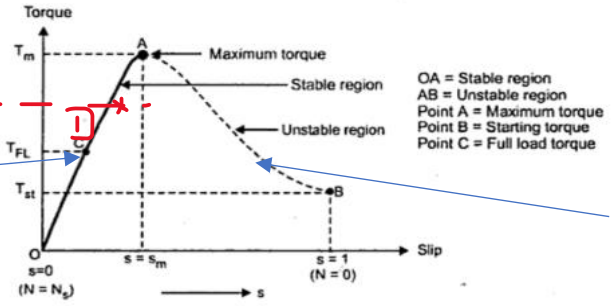
Low Slip region

$$(sX_2)^2 \ll (R_2)$$

$$\tau \propto s$$

Speed-Torque Curve for a Three-Phase Induction Motor

Stability of Torque and speed



$\tau > \tau_{load}$ => deceleration speed will be decreased.

$\tau > \tau_{load}$ => acceleration speed will be increased.

In this we can see for the same torque due to speed change one of two things can happen. If slip is higher than S_0 the torque will be lesser and lesser and speed will increase with that. If slip is lower than S_0 the speed will be lower and lower. So there are two regions depending on the speed of the motor. **A simple control can't be implemented because of this.**

More characteristics of the induction motor

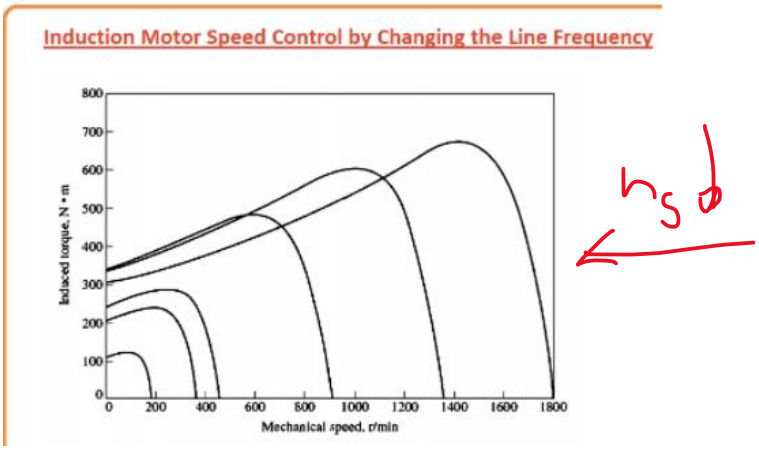


Fig: Torque-speed characteristics with synchronous speed

We can change the torque of an induction motor by changing the synchronous speed(in another words-

Line frequency).

$$n_s = \frac{120\,f}{P}$$

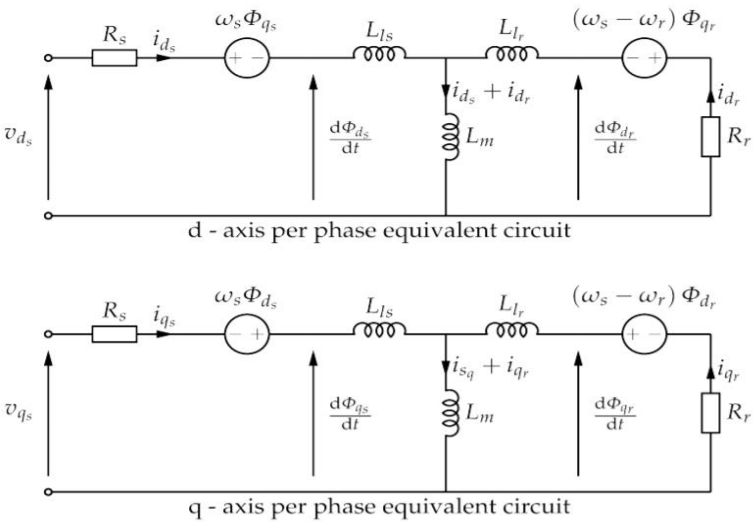
f-frequency of the supplied 3 phase sinusoidal input.

By changing f we can change torque of the motor.

Literature Review

MATHEMATICAL MODELLING OF THIS PROJECT

Mathematical modelling of induction motor using d-q axis model.



The following equations can be written referring this model.

Fig: Dynamic Model Circuit

$$v_{ds} - i_{ds}R_s - \omega_s\phi_{qs} - L_{1s}\frac{di_{ds}}{dt} - L_m\frac{d(i_{ds}+i_{dr})}{dt} = 0$$

$$v_{ds} - i_{ds}R_s - \omega_s\phi_{qs} - L_{1s}\frac{di_{ds}}{dt} - L_m\frac{d(i_{ds}+i_{dr})}{dt} = 0$$

$$\frac{d\phi_{ds}}{dt} = v_{ds} - i_{ds}R_s - \omega_s\phi_{qs}$$

$$\frac{d\phi_{qs}}{dt} = v_{ds} - i_{qs}R_s + \omega_s\phi_{ds}$$

$$\frac{d\phi_{dr}}{dt} = -i_{dr}R_r - (\omega_s - \omega_r)\phi_{qr}$$

$$\frac{d\phi_{qr}}{dt} = -i_{qr}R_r + (\omega_s - \omega_r)\phi_{dr}$$

$$\phi_{sq} = (i_{qs} + i_{qr})L_m + i_{qs}L_{1s}$$

$$\phi_{sd} = (i_{ds} + i_{dr})L_m + i_{ds}L_{1s}$$

$$\phi_{rq} = (i_{qs} + i_{qr})L_m + i_{qr}L_{1r}$$

$$\phi_{rd} = \left(i_{ds} + i_{dr)}L_m\right) + i_{qs}L_{1s}$$

$$i_{qs} = \frac{\phi_{qs} - L_m i_{qr}}{L_m + L_{1s}}$$

$$i_{ds} = \frac{\phi_{ds} - L_m i_{dr}}{L_m + L_{1s}}$$

$$T_e = \frac{3P}{4}\left((i_{qs}(i_{ds} + i_{dr})L_m\right) + i_{ds}(i_{qs} + i_{qr})L_m))$$

$$\frac{d\omega_r}{dt} = \frac{T_e - D\omega_r - T_L}{J}$$

SVPWM modelling

Rotating magnetic field

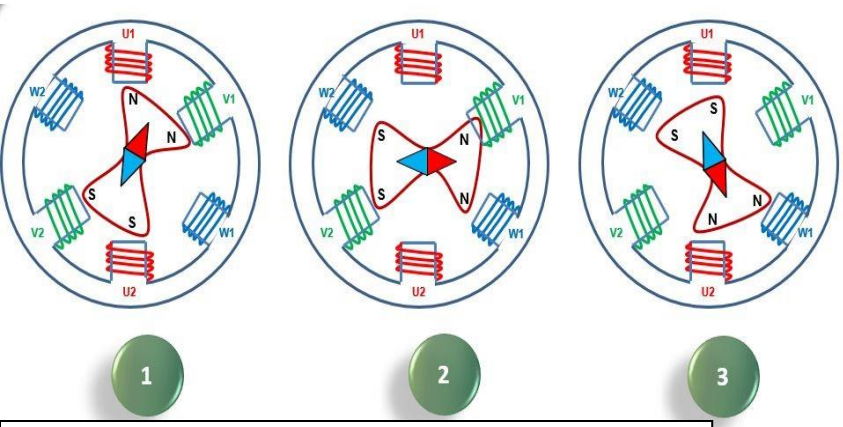


Fig: Rotating magnetic field

A rotating magnetic field can be generated by placing 3 pole pairs of magnetic coils 120 degrees different along a circle and by applying 120 degrees phase shifted sinusoidal waveforms to those pole pairs differently.

It is observed that we can write a space vector which is proportional to the magnetic field and with the same angle.

$$V_R(t) = \frac{2}{3}(V_{AN}(t) - V_{BN}(t)e^{j\frac{2\pi}{3}} - V_{CN}(t)e^{j\frac{4\pi}{3}})$$

For the system represented below we can write following equations.

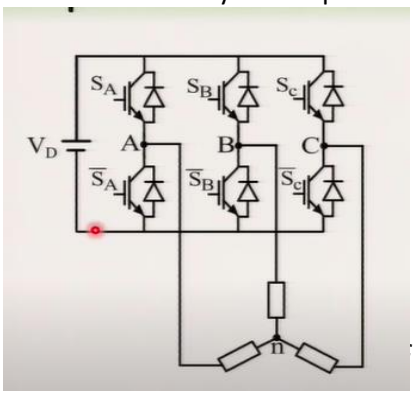


Fig: switches connected to the inverter

$$V_{AN}(t) = \frac{2}{3}V_{AO}(t) - \frac{1}{3}V_{BO}(t) - \frac{2}{3}V_{C0}(t)$$

$$V_{BN}(t) = \frac{2}{3}V_{BO}(t) - \frac{1}{3}V_{CO}(t) - \frac{1}{3}V_{C0}(t)$$

$$V_{CN}(t) = \frac{2}{3}V_{CO}(t) - \frac{1}{3}V_{AO}(t) - \frac{1}{3}V_{B0}(t)$$

$= 0$

$$consider\ V_{A0} = V_D, V_{B0} = 0, V_{C0} = 0$$

$$V_{AN} = \frac{2}{3}V_D$$

$$V_{BN} = -\frac{1}{3}V_D$$

$$V_{CN} = -\frac{1}{3}V_D$$

$$V_R(t) = \frac{2}{3}V_D e^{j0}$$

This process can be replicated to other switching states.

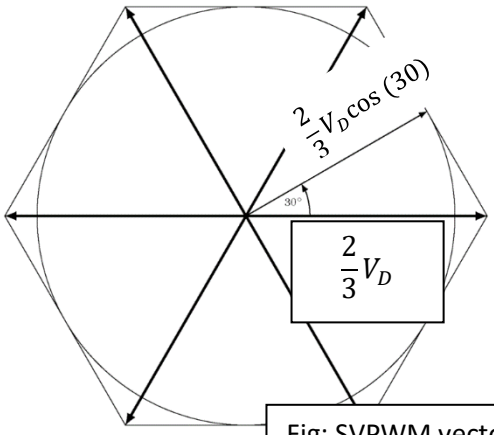
Then we have space vector values as below.

Space Vector	Switching states	Resultant Space Vector	
V_0	000	$V_0 = 0$	Zero Vector
V_1	010	$V_1 = \frac{2}{3}V_D e^{j0}$	

V_2	110	$V_2 = \frac{2}{3}V_D e^{j\pi/3}$	Active Vector
V_3	010	$V_3 = \frac{2}{3}V_D e^{j2\pi/3}$	
V_4	011	$V_4 = \frac{2}{3}V_D e^{j3\pi/3}$	
V_5	001	$V_5 = \frac{2}{3}V_D e^{j4\pi/3}$	
V_6	101	$V_6 = \frac{2}{3}V_D e^{j5\pi/3}$	
V_7	111	$V_7 = \frac{2}{3}V_D e^{j7\pi/3}$	Zero Vector

Fig: Switching Sequence table

We can represent those vectors as in diagram below.



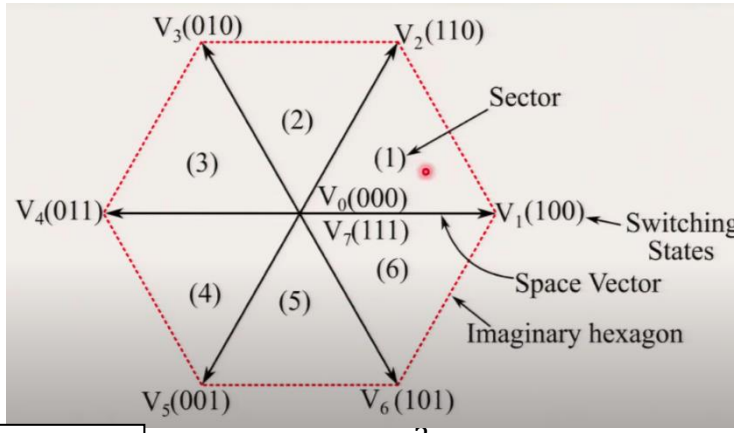


Fig: SVPWM vector

$$V_{R(\max)} = \frac{2}{3}V_D Cos(30) = 0.577V_D$$

Consider Vectors V_1 , V_2 and an intermediate vector.

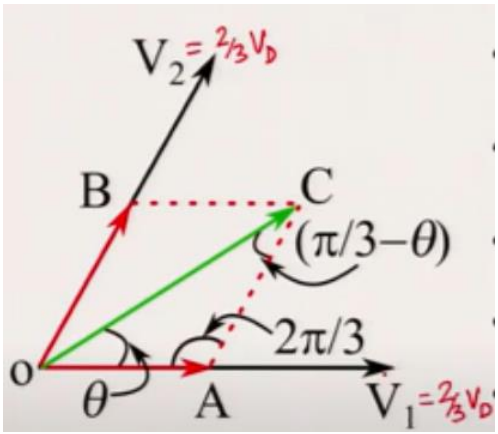


Fig: Vector between two switching states

Using sine law to triangle OAC,

$$\frac{OA}{\sin(\frac{\pi}{3}-\theta)} = \frac{OB}{\sin(\theta)} = \frac{OC}{\sin(\frac{2\pi}{3})}$$

$$\frac{V_1T_1}{\sin(\frac{\pi}{3}-\theta)} = \frac{V_2T_2}{\sin(\theta)} = \frac{V_RT_s}{\sin(\frac{2\pi}{3})}$$

$$T_1 = \sin\left(\frac{\pi}{3}-\theta\right)\frac{V_R}{V_1}\frac{2}{\sqrt{3}}T_s = \sin\left(\frac{\pi}{3}-\theta\right)\frac{V_R}{V_D}\sqrt{3}T_s$$

$$T_2 = \sin(\theta)\frac{V_R}{V_1}\frac{2}{\sqrt{3}}T_s = \sin(\theta)\frac{V_R}{V_D}\sqrt{3}T_s$$

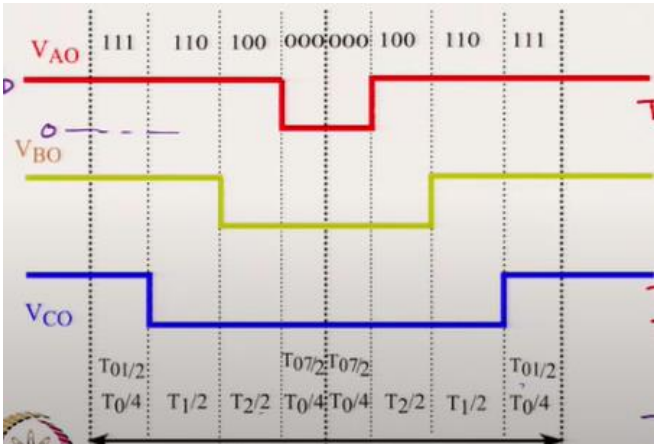
$$T_0 = T_s - T_1 - T_2$$

The space vectors which define the boundary are switched between certain dura of time to produce a resultant space vector of that section.

To avoid harmonic distortion we use both zero vectors in switching process.

By reference [3] research paper I adapted Harmonic reduction method to the switching pattern.

$$T_{01} = T_{07} = \frac{T_0}{2}$$



Consider a vector in sector 1. -The switching sequence.

$$V_7 = 1\,1\,1 = \frac{T_0}{4}$$
$$V_2 = 1\,1\,0 = \frac{T_1}{2}$$
$$V_1 = 1\,0\,0 = \frac{T_2}{2}$$
$$V_0 = 0\,0\,0 = \frac{T_0}{4}$$
$$V_0 = 0\,0\,0 = \frac{T_0}{4}$$
$$V_2 = 1\,0\,0 = \frac{T_2}{2}$$
$$V_1 = 1\,1\,0 = \frac{T_1}{2}$$
$$V_7 = 1\,1\,1 = \frac{T_0}{4}$$

PID controller

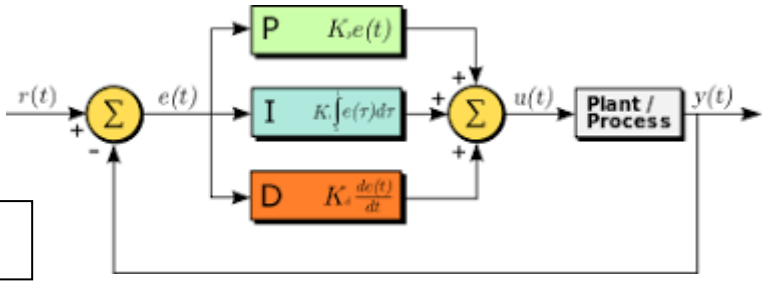


Fig: PID block diagram

PID(proportional integral Derivative) is a control-loop mechanism which is commonly used control applications .A PID controller calculates the error value as the difference between a measured value and desired value and applies a correction command value on Proportional, Integral, Derivative control(denoted *P*, *I*, and *D* respectively). Because of that it is called a PID controller.

$$\mathrm{e(t)} = \mathrm{r(t)} - \mathrm{y(t)}$$
$$u(t) = K_p e(t) + K_i \int e(t)dt + K_e \frac{de(t)}{dt}$$

Tuning a PID controller using Ziegler -Nichols Method

$$u(t) = K_p(e(t) + \frac{1}{T_i}\int e(t)dt + T_e \frac{de(t)}{dt})$$
$$K_i = \frac{K_p}{T_i}$$
$$K_d = K_p T_d$$

Procedure of Ziegler-Nichols Method

1)Start with small $K_p\ value\ \&\ K_i = 0\ K_d = 0$

2)Increase K_P until the system achieves neutral stability.

3)Record the critical/Ultimate gain $K_u =$ K_p at neutral stability.

4)Record the critical period of oscillation T_u

5)Look up the table and Calculate K_p , K_i , K_d values according to the table and the scenario you want to implement.

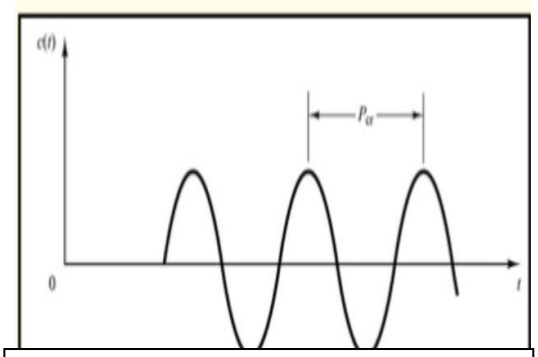


Fig: sinusoidal model response

Advantages of PID controller

Very easy to understand and no expertise is required.

No model or a simulation is needed to implement

Control Type	K_P	T_i	T_d	$K_I = K_P/T_i$	$K_D = T_d K_P$
PID (classic)	$0.6\ K_U$	$T_U/2$	$T_U/8$	$1.2K_U/T_U$	$0.075\ K_U T_U$
P	$0.5\ K_U$	-	-	-	-
PI	$0.45\ K_U$	$T_U/1.2$	-	$0.54K_U/T_U$	-
PD	$0.8\ K_U$	-	$T_U/8$	-	$0.1K_U T_U$
Pessen Integration	$0.7\ K_U$	$2T_U/5$	$3\ T_U/20$	$1.75K_U/T_U$	$0.105\ K_U T_U$
Some Overshoot	$K_U/3$	$T_U/2$	$T_U/3$	$(2/3)K_U/T_U$	$(1/9)K_U/T_U$
No Overshoot	$0.2\ K_U$	$T_U/2$	$T_U/3$	$(2/5)K_U/T_U$	$(1/15)K_U/T_U$

Fig: PID values for different applications

Disadvantages

Best fit for relatively stable systems

VSI control method of the induction motor

Voltage Source Inverter Control of asynchronous Motor allows a variable frequency AC voltage supply to be obtained from a dc supply source. Also It can reduce the supply voltage to the motor by changing switching patterns of the MOSFET or IGBT. In here we use SVPWM technique to control the speed of the motor. The magnitude of the Space Vector is proportional to the supply Voltage and rotating speed of the Space Vector is proportional to the synchronous speed of the motor. By changing those two variables we can control the speed of the motor.

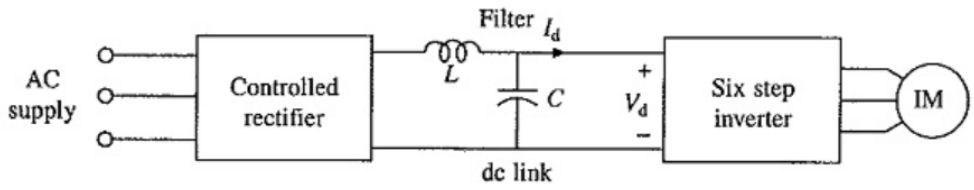
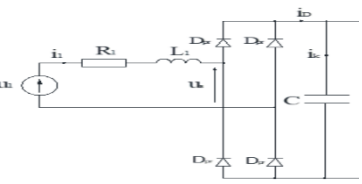


Fig: Rectifier and Inverter



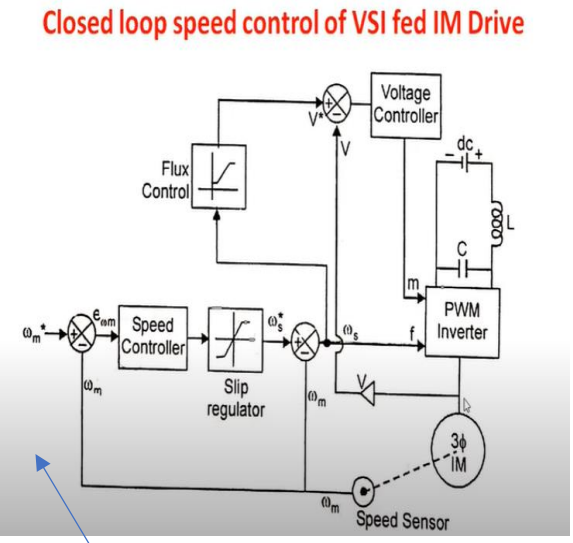
First We take AC supply and Convert it to the DC Volage Source using rectifier Circuit.

Then We use MOSFETS(or IGBTs) to convert DC voltage to 3 phase Voltage by switching techniques. (Fig 1)

By using SVPWM we create a rotating magnetic field which drives the induction motor.

<u>How Do we control the speed If the motor is nonlinear??</u>

<u>V/F control Method</u>



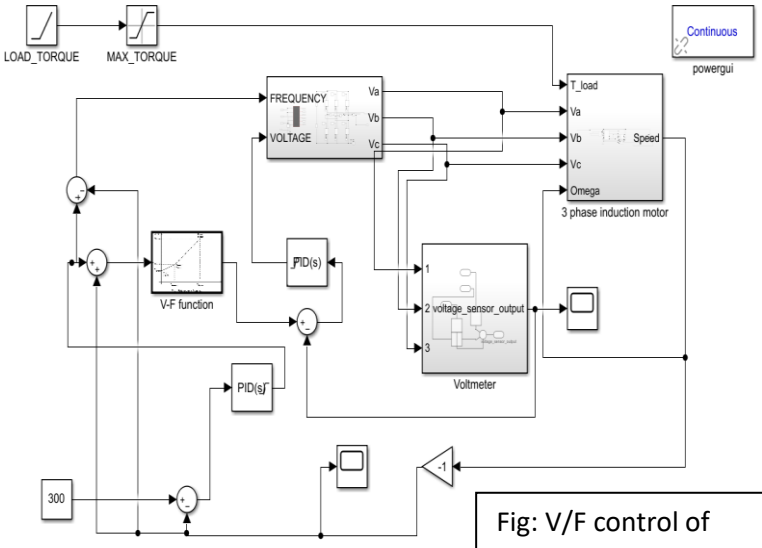


Fig: V/F control of induction motor

This ω_m value is what our reference(desired) speed. We use a speed sensor to measure current speed and feed error to the PID controller. (mechanical speed - speed controller output speed) is fed to SVPWM generator frequency input and the Flux control V & F map function input. The function generates a reference value and (reference value - voltage sensor value) is fed to another PID controller and that generated voltage level is fed to SVPWM magnitude input.

From reference [1],

Maximum input frequency to the MOSFETS are when producing SVPWM is 6KHz.

Modulation index is 0.9 ➔

Modulating frequency*0.9 <Frequency deviation (This fact is in the simulation as a saturation point).

Components that I used for the practical Implementation

From reference [2] research paper I concluded that MOSFET switching is optimal for my task.

IRF 540 MOSFET drive module

Bootstrapping of a MOSFET

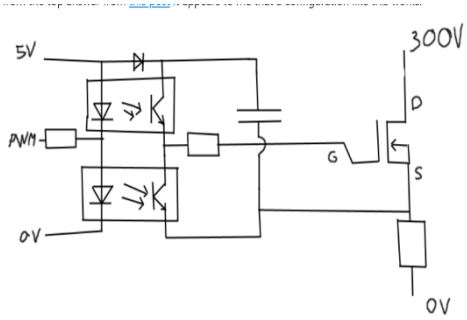


Fig: Bootstrapping

The in area of the of MOSFET circuits, bootstrapping is essential to mean pulling up operating point of a MOSFET above the power supply voltage. If the MOSFET is not bootstrapped when MOSFET is turned on $V_{DS} \approx 0$ and $V_{GS} < 0$ because of that MOSFET turns off always. To mend this issue the circuit below is used. The bootstrapping in driver module is built in implemented.

Intent of practical implementation-Demonstration of PID controller to the change of SVPWM input

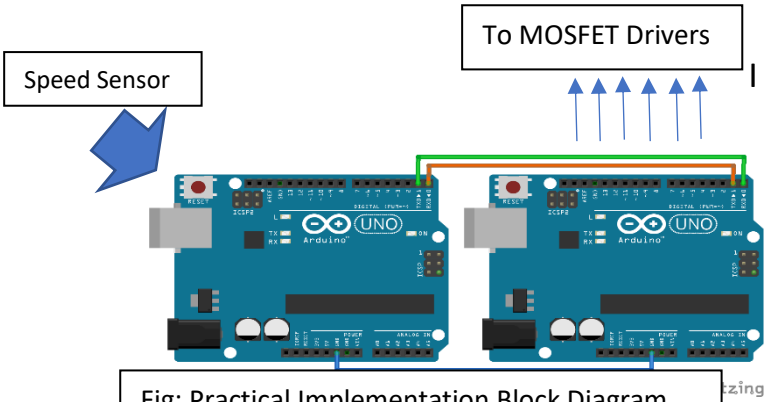


Fig: Practical Implementation Block Diagram

I used two Arduino boards because the tachometer sensing value has to be calculated using one Arduino board(1 parallel thread). The other board is used to generate SVPWM input(digital signals) to the MOSFET.

Arduino can't handle two complex codes simultaneously due to low calculation power. Parallel threading is not an option we have because of this. Either we can use a Raspberry PI or other higher capability board or we can use two Arduino boards for two simultaneous calculations with a serial connection between them.

CONCLUSIONS AND FUTURE WORK

Conclusions

This project's main goal is to build a low cost highly efficient 3 phase induction motor controller driver. Also to understand where I can optimize the circuit and the algorithm of the driver. I haven't connected this to a load with electrical loading phenomenon . Certainly there would be loading effect to this driver . The Sinusoidal wave would be effected due to this loading . To continue to optimized this I have to implement a practical application to this project.

Recommendation in Future Work

From what I read in research Paper reference [4] Fuzzy Logic control based system can be more accurate in some controlling situations where the non-linearities are higher. Also artificial neural networks can be used to control the induction motor. By designing the layers and by using back propagation algorithms to tune the neural weights of the network. I was looking into neural networks and since time has limited I couldn't complete the model of the system.

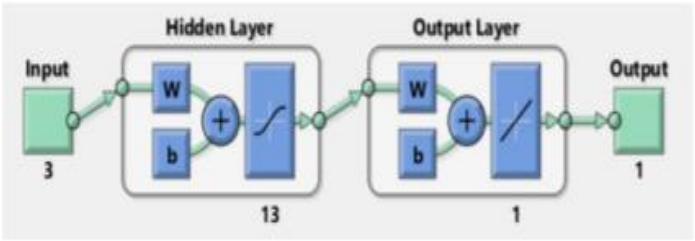


Fig: Neural network

FOC control(Field Oriented Control)

FOC control is mostly used for controlling PMSM motors .But it can also be adapted to asynchronous motors.

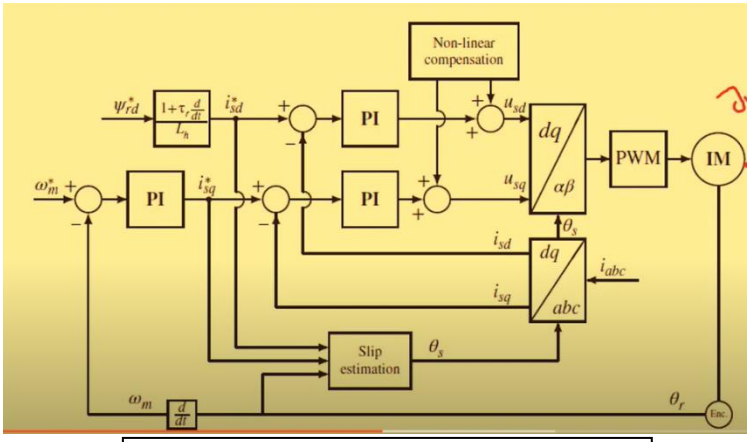


Fig: FOC control block Diagram

(Part of the control is explained in the video). I couldn't work out math part(Park transformation, Clarke transformation,dq0 frame) of this controller In time since because of this I didn't do a simulation on this part.

References

[1]Peña, Juan Moreano, and Edilberto Vásquez Díaz. "Implementation of V/f scalar control for speed regulation of a three-phase induction motor." 2016 IEEE ANDESCON. IEEE, 2016.

[2] Shukla, Niraj Kumar, and Rajeev Srivastava. "Performance Evaluation of Three Phase Induction Motor Using MOSFET & IGBT Based Voltage Source Inverter." *International Research Journal of Engineering and Technology (IRJET)* 4.06 (2017): 2395-0056.

[3] Baitade, A. T., and S. S. Chopade. "Harmonic reduction, power factor improvement and speed detection for 3-phase induction motor drive system." *2016 Conference on Advances in Signal Processing (CASP)*. IEEE, 2016.

[4] Ali, Ahmed Jadaan, Ziyad Farej, and Nashwan Sultan. "Performance evaluation of a hybrid fuzzy logic controller based on genetic algorithm for three phase induction motor drive." International Journal of Power Electronics and Drive Systems 10.1 (2019): 117.

[5] Azman, M. A. H., et al. "A comparative study of fuzzy logic controller and artificial neural network in speed control of separately excited DC motor." *2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE, 2017.

Appendix

Code for board 1

```
#include <PID_v_1.h>  // pid library
double SetPoint ;
double Input;  //IR sensor
double Output ; //LED
//praameters
double Kp=0, Ki=10, Kd=0;

 char str[4];

PID myPID(&Input, &Output, &SetPoint, Kp, Ki, Kd, DIRECT);
const int dataIN = 2; //sensor inputs

unsigned long prev_millis; // define time
unsigned long dura; // time difference difference
unsigned long refresh; // To store refresh time

int rpm; // RPM value

boolean currentState;
boolean prev_state;

void setup()
{
   Serial.begin(9600);
  SetPoint = 75;
  myPID.SetMode(AUTOMATIC);
  myPID.SetTunings(Kp, Ki, Kd);
  pinMode(dataIN,INPUT);
  prev_millis = 0;
  prev_state = LOW;

}

void loop()
{
 // RPM Measurement
  currentstate = digitalRead(dataIN); // Read IR sensor state
 if( prev_state != currentstate) // If there is change in input
   {
    if( currentstate == HIGH ) // If input only changes from LOW to HIGH
      {
       dura = ( micros() - prev_millis ); // Time difference between revolution in microsecond
```

Code for board 2

```
#include <math.h>
char str[4];
int sectorPos = 1;  //current sector.

float freq_sth = 10000.00; // Switching freqency 5000Hz
float freq = 50.0; // Variable fundamental freqency 50
float sample_time = (freq_sth/freq); // sample_time(per cycle) = switching_time/fundamental_output_freq.
float base_angle = (360.0/sample_time); //Base angle will be used to increment angle per cycle.

float m = 0.8;     // modulation index
int vdc = 100;     // rectified voltage
int vref = (vdc)*(m);  // Ref voltage

float tz = ((1/freq_sth))/pow(10,-6);    //All switching time
float timer_1 = 0, timer_2 = 0, timer_0 = 0; // Switching times

float angle = 0;
float angle_rad = 0;

float timer_div = 0;     // Constant in time calculation equations.
double theta_1 = 0, theta_2 = 0; //
int M_SWITCH_1 = 2, M_SWITCH_2 = 3, M_SWITCH_3 = 4, M_SWITCH_4 = 5, M_SWITCH_5 = 6, M_SWITCH_6 = 7; // Arduino digital pins
int sensor_value = 0;

void setup() {

 Serial.begin(9600);
                // Arduino digital pins
 pinMode(M_SWITCH_1, OUTPUT);  // High U PIN DIGITAL PIN 2
 pinMode(M_SWITCH_2, OUTPUT);  // High V PIN DIGITAL PIN 4
 pinMode(M_SWITCH_3, OUTPUT);     // High W PIN DIGITAL PIN 6
 pinMode(M_SWITCH_4, OUTPUT);  // Low U PIN DIGITAL PIN 3
 pinMode(M_SWITCH_5, OUTPUT);  // Low V PIN DIGITAL PIN 5
```

```
      rpm = (60000000/dura); // rpm = (1/ time millis)*1000*1000*60;
      prev_millis = micros(); // store time for nect revolution calculation
     }
  }
 prev_state = currentstate; // store this scan (prev scan) data for next scan


 if( ( millis()-refresh ) >= 100 )
   {
 Input = map(rpm, 0, 1024, 0, 255);  // photo senor is set on analog pin 5
 //PID calculation
 myPID.Compute();
 //Write the output as calculated by the PID function
 analogWrite(3,Output);
  itoa(rpm, str, 10);
  Serial.write(str, 4); //Write the serial data
  Serial.end();
 Serial.begin(9600);
 delay(1000);
   }

}
```

```
 pinMode(M_SWITCH_6, OUTPUT);     // Low W PIN DIGITAL PIN 7  digitalWrite(M_SWITCH_1, LOW);
 digitalWrite(M_SWITCH_2, LOW);
 digitalWrite(M_SWITCH_3, LOW);
 digitalWrite(M_SWITCH_4, LOW);
 digitalWrite(M_SWITCH_5, LOW);
 digitalWrite(M_SWITCH_6, LOW);

}

void loop() {
int i = 0;
        if (Serial.available()) {

         delay(100); //allows all serial sent to be received together
          while (Serial.available() && i < 4) {
           str[i++] = Serial.read();

          }
          Serial.flush();


        }
 sensor_value = int(str);
 sensor_value = (sensor_value/30);
 Serial.write(str);

 if (sensor_value<3) {
    sensor_value = 3;
   }

 base_angle = sensor_value;

 //  Serial.println(base_angle);
 //  delay(100);
 // base_angle = (360.0/sensor_value);
 // Serial.println(base_angle);
 // delay(300);
 // */

 sector_tracker();
 time_cal();
 mosfet_switch();
 angle_incr();
 // delay(10);
}

void sector_tracker()
{
 if(angle >= 60) //When angle is >= 60 that means we have covered one 1 sector.
 {
    if(sectorPos != 7) // this condition is here to make sure we are never in sector 7 (i.e wer
    {
    angle = 0;
    sectorPos = (sectorPos+1);
    }
    if(sectorPos == 7) // if after expression 'sectorPos = sectorPos + 1' in above case,we get 7, we need to set everything to initial point since no sector 7 exists.
    { //reason for putting this condition after the above condition is because there was a case when sectorPos = 7 after we update the sectorPos.
      angle = 0;
      sectorPos = 1;
    }
 }
}

void time_cal() {
   timer_div = (2/sqrt(3) * tz * m);
```

```
    angle_rad =  angle * (M_PI/180);     //converting angle->degrees to angle->radians. Need radian term for sin.
    theta_1 = sin((M_PI/3.0)-angle_rad);
    theta_2 = sin(angle_rad);
    timer_1 = timer_div * theta_1;
    timer_2 = timer_div * theta_2;
    timer_0 = (tz - (timer_1 + timer_2))/2.0;
     //all time in (us)
    timer_1 = round(timer_1);
    timer_2 = round(timer_2);
    timer_0 = round(timer_0);

/*
  Serial.print("Sector: ");
  Serial.println(sectorPos);
  Serial.print("Timer 1:");
  Serial.println(timer_1, 2);
  Serial.print("Timer 2:");
  Serial.println(timer_2, 2);
  Serial.print("Timer 0:");
  Serial.println(timer_0, 2);
  delay(300);
*/
}

void angle_incr() {
    angle = angle + base_angle;
}

void mosfet_switch() {
  if (sectorPos == 1)
  {
    //v_0-v_1-v_2-v_7-v_2-v_1-v_0
    //-----
    //v_0:
    digitalWrite(M_SWITCH_1,LOW);
    digitalWrite(M_SWITCH_3,LOW);
    digitalWrite(M_SWITCH_5,LOW);
    digitalWrite(M_SWITCH_2,HIGH);
    digitalWrite(M_SWITCH_4,HIGH);
    digitalWrite(M_SWITCH_6,HIGH);
    delayMicroseconds(timer_0);
    //v_1:
    digitalWrite(M_SWITCH_1,HIGH);
    digitalWrite(M_SWITCH_3,LOW);
    digitalWrite(M_SWITCH_5,LOW);
    digitalWrite(M_SWITCH_2,LOW);
    digitalWrite(M_SWITCH_4,HIGH);
    digitalWrite(M_SWITCH_6,HIGH);
    delayMicroseconds(timer_1);
    //v_2:
    digitalWrite(M_SWITCH_1,HIGH);
    digitalWrite(M_SWITCH_3,HIGH);
    digitalWrite(M_SWITCH_5,LOW);
    digitalWrite(M_SWITCH_2,LOW);
    digitalWrite(M_SWITCH_4,LOW);
    digitalWrite(M_SWITCH_6,HIGH);
    delayMicroseconds(timer_2);
    //v_7:
    digitalWrite(M_SWITCH_1,HIGH);
    digitalWrite(M_SWITCH_3,HIGH);
    digitalWrite(M_SWITCH_5,HIGH);
    digitalWrite(M_SWITCH_2,LOW);
    digitalWrite(M_SWITCH_4,LOW);
    digitalWrite(M_SWITCH_6,LOW);
    delayMicroseconds(timer_0);
    //v_2:
    digitalWrite(M_SWITCH_1,HIGH);
    digitalWrite(M_SWITCH_3,HIGH);
    digitalWrite(M_SWITCH_5,LOW);
    digitalWrite(M_SWITCH_2,LOW);
    digitalWrite(M_SWITCH_4,LOW);
    digitalWrite(M_SWITCH_6,HIGH);
    delayMicroseconds(timer_2);
```

```
  //v_1:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_1);
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
 }
 if (sectorPos == 2)
 {
  //v_0-v_3-v_2-v_7-v_2-v_3-v_0
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
  //v_3:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_2);
  //v_2:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_1);
  //v_7:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_0);
  //v_2:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_1);
  //v_3:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_2);
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
```

```
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);

 }
 if (sectorPos == 3)
 {
  //v_0-v_3-v_4-v_7-v_4-v_3-v_0
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
  //v_3:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_1);
  //v_4:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_2);
  //v_7:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_0);
  //v_4:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_2);
  //v_3:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_1);
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
 }
 if (sectorPos == 4)
 {
  //v_0-v_5-v_4-v_7-v_4-v_5-v_0
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
```

```
  delayMicroseconds(timer_0);
  //v_5:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_2);
  //v_4:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_1);
  //v_7:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_0);
  //v_4:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_1);
  //v_5:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_2);
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
 }
 if (sectorPos == 5)
 {
  //v_0-v_5-v_6-v_7-v_6-v_5-v_0
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
  //v_5:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_1);
  //v_6:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
```

```
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_2);
  //v_7:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_0);
  //v_6:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_2);
  //v_5:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_1);
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
 }
 if (sectorPos == 6)
 {
  //v_0-v_1-v_6-v_7-v_6-v_1-v_0
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
  //v_1:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_2);
  //v_6:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_1);
  //v_7:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,HIGH);
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,LOW);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_0);
  //v_6:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,LOW);
```

```
  digitalWrite(M_SWITCH_5,HIGH);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,LOW);
  delayMicroseconds(timer_1);
  //v_1:
  digitalWrite(M_SWITCH_1,HIGH);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,LOW);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_2);
  //v_0:
  digitalWrite(M_SWITCH_1,LOW);
  digitalWrite(M_SWITCH_3,LOW);
  digitalWrite(M_SWITCH_5,LOW);
  digitalWrite(M_SWITCH_2,HIGH);
  digitalWrite(M_SWITCH_4,HIGH);
  digitalWrite(M_SWITCH_6,HIGH);
  delayMicroseconds(timer_0);
 }
}
```