Conversor AC/DC Trifásico Controlado

Sergio Esteban Jaiquel Barón
u1802912@unimilitar.edu.co
Profesor: Luis Francisco Niño Sierra

I. Introducción

Existen cargas tanto resistivas como RL que necesitan un voltaje y una corriente DC, pero utilizando fuentes AC preferiblemente trifásicas debido a que a diferencia de las monofásicas se efectúa en tres fases, asegurando condiciones necesarias para un mejor suministro de energía con esfuerzos y potencia instantánea constantes y sin sufrir tantas vibraciones; para ellos se utilizan rectificadores trifásicos que pueden ser controlados o no, eso dependerá de los dispositivos usados en el circuito, para rectificadores controlados se utiliza el SCR y para no controlados se utiliza un diodo normal.

Se realiza un circuito rectificador trifásico controlado de media onda con el fin de mantener una corriente constante sobre una carga inductiva, para ello se hicieron cálculos, simulaciones.

II. Desarrollo de la práctica

Se buscan distintos modelos de Conversores AD/DC trifásicos como guía para su implementación, se encuentra que el diseño más óptimo implementable para esta práctica vendría a ser el rectificador trifásico de media onda.

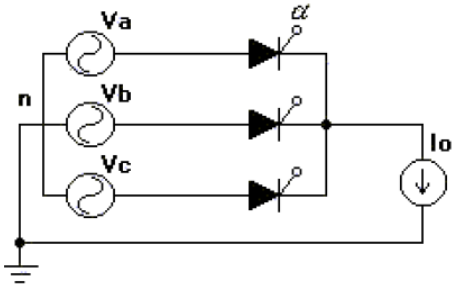

Figura 1. Esquemático de diseño rectificador

Una vez elegido el modelo de conversor que se desea implementar, se plantea la forma en la que se diseñará el sistema de control para el disparo de los tiristores. Se decide utilizar una red de optocuplas conectados cada uno a cada fase, que serán capaces de entregar una señal lógica inversa respecto al voltaje entregado por la fase medida. En otras palabras, siempre y cuando la fase entregue un voltaje positivo, la optocupla entregará un cero lógico y viceversa.

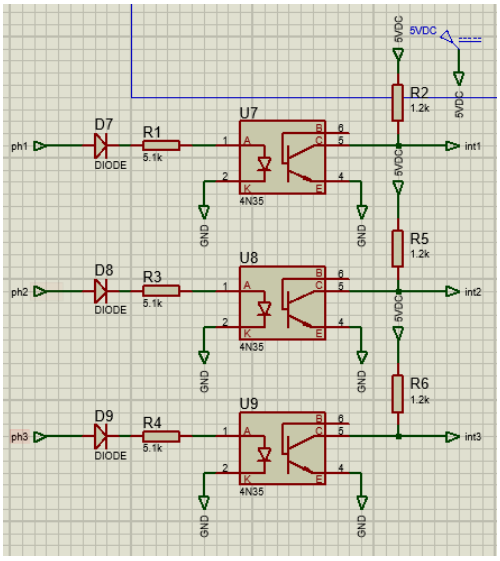

Figura 2. Esquemático de diseño de control rectificador

Las señales lógicas son entregadas a un microcontrolador Arduino, el cual al captar las interrupciones por flanco de bajada provenientes de las optocuplas, emite señales lógicas que activan el disparo de los tiristores en un ángulo determinado.

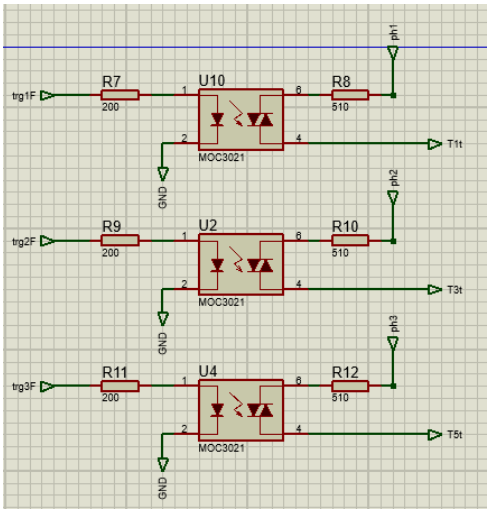

Figura 3. Esquemático de diseño de control rectificador

Se utiliza entonces una red de Optoacopladores uno conectado a cada fase, los cuales permiten el paso de corriente en su fase de potencia, cada vez que reciben un voltaje lógico positivo en su fase de control. Este

procedimiento se usa para realizar el disparo sobre los tiristores, donde se utiliza la misma fase que los alimenta para realizar el disparo garantizando así la corriente que necesitan para recibirlo apropiadamente.

Por último se procede a conectar la carga a la fase DC del circuito, como se ve en la siguente figura. Se realizan las mediciones de corriente, voltaje y potencia sobre esta carga.

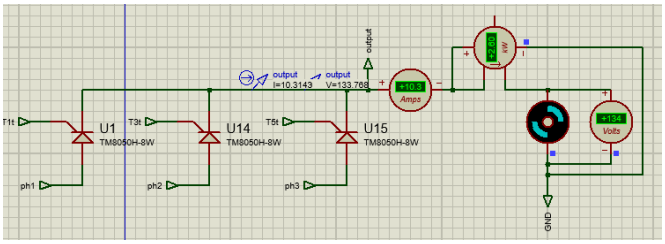

Figura 4. Esquemático de rectificador implementado

Realizando esta simulación se obtiene la respuesta del sistema medida desde el osciloscopio del simulador Proteus.

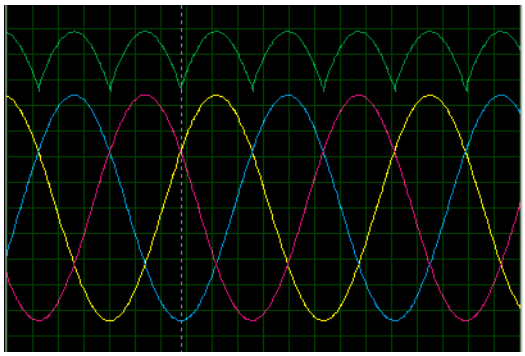

Figura 5. Respuesta de salida del rectificador

Así mismo se toman las gráficas de corriente y voltaje utilizando la herramienta de "Analogue Analisis" del simulador Proteus.

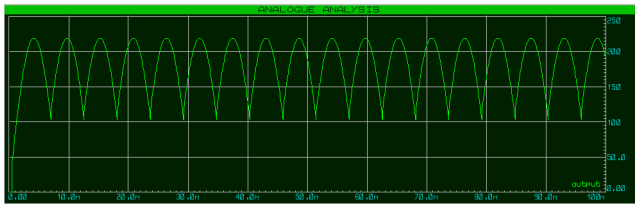

Figura 6. Gráfica de voltaje de salida del rectificador

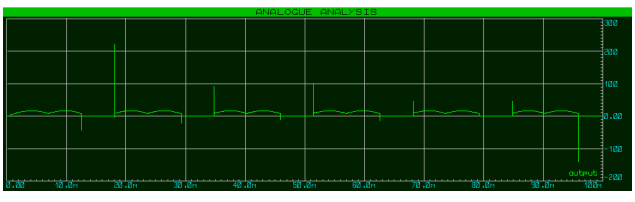

Figura 7. Gráfica de corriente de salida del rectificador

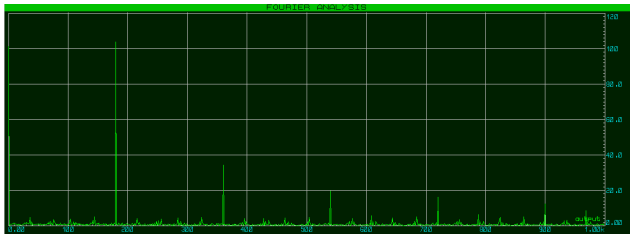

Figura 8. Gráfica los armónicos de salida del rectificador

Se puede observar en estas gráficas que tanto la corriente como el voltaje se comportan de la forma en que se espera según el texto guía entregado por el docente en teoría.

Se puede observar entonces, como en cada uno de los ciclos de cada fase, el disparo de los tiristores solo permite el flujo de voltaje en el punto más alto de cada uno de los ciclos de cada fase. Dando como resultado una señal dc con un un risado relativamente alto, y con una frecuencia armónica de 3 veces la frecuencia de la fuente, en este caso al ser la fuente de 60 hz, sus armónicos poseen una frecuencia de 180 Hz.

Se tiene en este tipo de rectificadores que la tensión de salida promedio puede expresarse como

$$V_{out} = V_{phase} * 0{,}93 = 204{,}6V$$

Siendo $V_{LL} = \sqrt{3}$ Veces el voltaje de fase.

De la misma forma, el voltaje RMS de la señal se expresa como:

$$V_{RMS} = V_{out} * 1{,}6554 = 210{,}26V$$

Teniendo en cuenta entonces la parte resistiva de la carga, se procede a realizar los cálculos de potencia:

$$P_{DC} = V_{out} * I_{out} = 3{,}488KW$$

$$P_{AC} = V_{RMS} * I_{RMS} = 3{,}684KW$$

$$\eta = \frac{P_{DC}}{P_{AC}} = \frac{3{,}488}{3{,}684} = 94{,}6\,\%$$

Finalmente de forma teórica se alaniza el factor de potencia del circuito, teniendo en cuenta que todos estos cálculos son realizados para un álguno de disparo $\alpha = 0$.

$$FP = \frac{45}{50} = 90\,\%$$

III. Conclusiones

- Se logró manejar el voltaje de una fase con un control de poca potencia (menos de 20 mA) con el uso correcto de un tiristor activado con un microcontrolador.

- Se lograron realizar tres circuitos de disparo para manejar cada fase independientemente de manera correcta con un error mínimo de sincronizada con la señal de la toma trifásica, este valor no era afectado

en cada ciclo ya que para cada periodo se sincronizaba nuevamente.

- Aunque el ejercicio es meramente de simulación, se realizó el diseño cumpliendo con estándares de protección contra corto circuito para la implementación real del montaje.

- Debe garantizarse la corriente de disparo mínima sobre la puerta del tiristor en el momento adecuado para garantizar su correcto funcionamiento, aunque exista una señal en el momento apropiado alimentando esta entrada, si no posee la corriente suficiente, el circuito no se comportara de la manera deseada.

IV. Referencias

[1] Aapuntes tomados en clase

[2] Rashid M. (Pearson) Power Electronics Devices, Circuits, and Applications. Capítulo 10. Rectificadores controlados.

V. Anexos

```
volatile int t1F = 0, t2F = 0, t3F = 0, t1R = 0, t2R = 0, t3R = 0;
int phase_shift = 0, voltaje = 0, angulo = 0;
int phase1F_in = 2,  phase1R_in = 18, phase1R_out = 7, phase1F_out = 11;
int phase2F_in = 3,  phase2R_in = 20, phase2R_out = 6, phase2F_out = 10;
int phase3F_in = 19, phase3R_in = 21, phase3R_out = 5, phase3F_out = 9;

void setup() {

  cli();//stop interrupts

    //set timer1 interrupt at 12KHz
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
  // set compare match register for 1hz increments
  OCR1A = 5335;// = (16*10^6) / (1*3*10^3) - 1 (must be <65536)
  // turn on CTC mode on timer1
  TCCR1B |= (1 << WGM12);
  // Set CS10 bit for psc = 1
  TCCR1B |= (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  sei();//allow interrupts
```

```
sei();//allow interrupts

 //set falling edge interrupt
pinMode(phase1F_out,OUTPUT); // Pin 11 digital output
pinMode(phase2F_out,OUTPUT); // Pin 10 digital output
pinMode(phase3F_out,OUTPUT); // Pin 9 digital output

pinMode(phase1R_out,OUTPUT); // Pin 7 digital output
pinMode(phase2R_out,OUTPUT); // Pin 6 digital output
pinMode(phase3R_out,OUTPUT); // Pin 5 digital output

pinMode(phase1F_in, INPUT);  // Pin 2 digital input
attachInterrupt(digitalPinToInterrupt(phase1F_in), Falling_Edge1, FALLING);  //goes into Falling_Edge interrupt when falling edge is detected on pin 2
pinMode(phase1R_in, INPUT);  // pin 18 digital input
attachInterrupt(digitalPinToInterrupt(phase1R_in), Rising_Edge1, RISING);    //goes into Rising_Edge interrupt when rising edge is detected on pin 18

pinMode(phase2F_in, INPUT);  // Pin 3 digital input
attachInterrupt(digitalPinToInterrupt(phase2F_in), Falling_Edge2, FALLING);  //goes into Falling_Edge interrupt when falling edge is detected on pin 3
pinMode(phase2R_in, INPUT);  // pin 20 digital input
attachInterrupt(digitalPinToInterrupt(phase2R_in), Rising_Edge2, RISING);    //goes into Rising_Edge interrupt when rising edge is detected on pin 20

pinMode(phase3F_in, INPUT);  // Pin 19 digital input
attachInterrupt(digitalPinToInterrupt(phase3F_in), Falling_Edge3, FALLING);  //goes into Falling_Edge interrupt when falling edge is detected on pin 19
pinMode(phase3R_in, INPUT);  // pin 21 digital input
attachInterrupt(digitalPinToInterrupt(phase3R_in), Rising_Edge3, RISING);    //goes into Rising_Edge interrupt when rising edge is detected on pin 21
```

```
ISR(TIMER1_COMPA_vect){
   //interrupt commands here
   t1F++; // Counter phase 1 - 1 half period
   t1R++; // Counter phase 1 - 2 half period
   t2F++; // Counter phase 2 - 1 half period
   t2R++; // Counter phase 2 - 2 half period
   t3F++; // Counter phase 3 - 1 half period
   t3R++; // Counter phase 4 - 2 half period
```

```
void loop() {
  // Falling edge interrupts logic result

  phase_shift = angulo*0.2;

  if ( t1F > phase_shift && t1F < phase_shift+4)
  {digitalWrite(phase1F_out,HIGH);}
  else { digitalWrite(phase1F_out,LOW); }

  if ( t2F > phase_shift && t2F < phase_shift+4)
  {digitalWrite(phase2F_out,HIGH);}
  else { digitalWrite(phase2F_out,LOW); }

  if ( t3F > phase_shift && t3F < phase_shift+4)
  {digitalWrite(phase3F_out,HIGH);}
  else { digitalWrite(phase3F_out,LOW); }

  //Rising edge interrupts logic result

   if ( t1R > phase_shift && t1R < phase_shift+4)
  {digitalWrite(phase1R_out,HIGH);}
  else { digitalWrite(phase1R_out,LOW); }

  if ( t2R > phase_shift && t2R < phase_shift+4)
  {digitalWrite(phase2R_out,HIGH);}
  else { digitalWrite(phase2R_out,LOW); }

  if ( t3R > phase_shift && t3R < phase_shift+4)
  {digitalWrite(phase3R_out,HIGH);}
  else { digitalWrite(phase3R_out,LOW); }

}//end void loop
```