Proyecto N°1: Convertidor SEPIC con transistor IGBT, pulso obtenido mediante microcontrolador PIC18F4550 que interactúa con interfaz gráfica en Matlab
(enero 2021).

Jocelyn Matus y Rafael Burgos, Ingeniería Civil Electrónica, Asignatura Sistemas Electrónicos, Código, Departamento de Ingeniería Eléctrica, Universidad de Concepción, Chile. Profesor Lautaro Salazar, Ayudante Felipe Pineda y Esteban Badilla.

Abstract – **En este informe se va a estudiar, investigar y crear un prototipo en PCB de un circuito de potencia Single-ended primary-inductor converter (SEPIC), el cual transforma un voltaje directo (DC) de cierta magnitud a otra magnitud. Además, se va a trabajar con el microcontrolador PIC18F4550 para poder crear la señal de control de Modulación por Ancho de Pulso (PWM), el cual va a ser interactuado por el usuario a través de una interfaz gráfica.**

INTRODUCCIÓN

Los circuitos de potencia DC/DC son circuitos que, desde una fuente DC, aumenta o disminuye el voltaje recibido. Hay muchos tipos de topología, pero, en nuestro caso, se va a realizar el circuito con topología SEPIC, el cual, su característica principal, es que en la salida puede ser mayor o menor que el voltaje en la entrada. Además, la señal de control para poder realizar esto, es a través del microcontrolador PIC18F4550, el cual controla el switch.

TEORÍA

Circuito de potencia SEPIC:

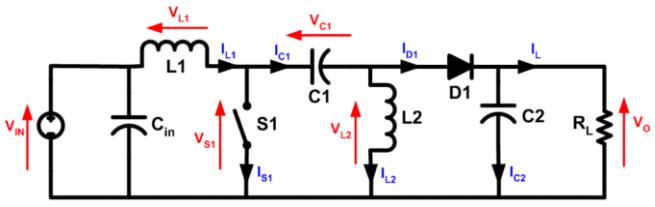


Figura 1. Circuito SEPIC con inductores separados.

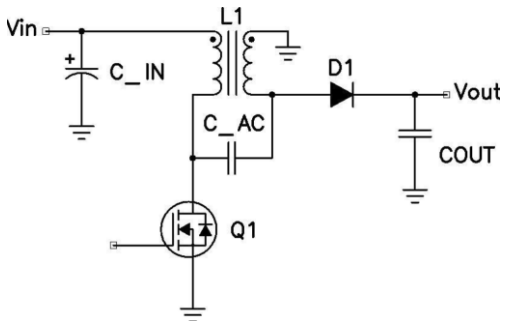


Figura 2. Circuito SEPIC con inductor acoplado.

El circuito de potencia SEPIC es un circuito que, desde una fuente de alimentación en la entrada constante, un voltaje DC, se puede obtener en la salida un voltaje mayor o menor a este, con la misma polaridad que en la entrada, en comparación con otros circuitos que no ocurre esto, como el buck-boost.

En nuestro proyecto se va a trabajar en modo de corriente continua; es decir, siempre se va a estar entregando corriente a la carga, y siempre va a haber corriente circulando en los inductores.

La operación del circuito es el siguiente:

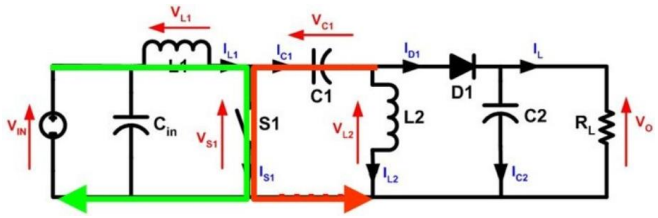


Figura 3. Operación SEPIC switch cerrado.

Cuando el switch S1 está cerrado, la corriente del inductor L1 se incrementa y la corriente del inductor L2 se hace más negativa. La energía que se usa para aumentar la corriente del inductor L1 proviene de la fuente de entrada. Ya que el switch S1 está cerrado, y el voltaje instantáneo del inductor L1 es aproximadamente el voltaje de entrada, Vin, el voltaje del inductor L2 es aproximadamente -Vc1, el negativo del voltaje de la capacitancia C1. Entonces, por esto, el diodo D1 está abierto, ya que el ánodo tiene un mayor voltaje que el cátodo, y el capacitor C1 entrega la energía para aumentar la magnitud de la corriente del inductor L2, y, con esto, la energía que se guarda en L2.

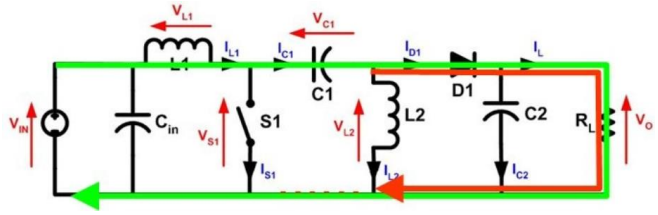


Figura 4. Operación SEPIC switch abierto.

Cuando el switch está abierto, la corriente del capacitor C1 se transforma en la misma que la de la corriente de L1, ya que la corriente de los inductores no puede cambiar instantáneamente. La corriente del inductor L2 va a continuar en la misma dirección. Con estas dos corrientes en el mismo sentido, la potencia se va directamente a la carga. Y, mientras pasa esto, el capacitor C1 se esta cargando por la corriente de L1, y la capacitancia C2 por la corriente de L1 y L2, la cual, la última, va a carga L2 durante el próximo ciclo [1].

Ecuaciones:

Las ecuaciones de diseño fueron obtenidas del paper de la compañía Texas Instruments, "Designing a SEPIC Converter" [2].

Duty:

Para poder calcular el duty, se tiene la siguiente ecuación:

$$D = \frac{V_{out} + V_D}{V_{in} + V_{out} + V_D}$$

Donde Vout es el voltaje de salida, Vd es el voltaje del diodo D1, y Vin el voltaje de entrada.

El duty máximo que se puede aplicar al circuito es el siguiente:

$$D_{max} = \frac{V_{out} + V_D}{V_{in(\min)} + V_{out} + V_D}$$

Con Vin(min) el voltaje mínimo de entrada que se va a aplicar.

Selección de inductor:

Primero, se tiene que saber el ripple que va a tener de corriente los inductores:

$$\triangle I_L = I_{in} \cdot \%(Ripple) = I_{out} \cdot \frac{V_{out}}{V_{in(min)}} \cdot \%(Ripple)$$

Entonces, el valor de los inductores (no acoplados) es el siguiente:

$$L_1 = L_2 = L = \frac{V_{in(min)}}{\triangle I_L \cdot f_{switch}} \cdot D_{max}$$

Con fswitch la frecuencia de conmutación.

Ahora, los valores de corriente peak de los inductores son los siguientes:

$$I_{L1_{peak}} = I_{out} \cdot \frac{V_{out} + V_D}{V_{in(min)}} \cdot \left(1 + \frac{\%(Ripple)}{2}\right)$$

$$I_{L2_{peak}} = I_{out} \cdot \left(1 + \frac{\%(Ripple)}{2}\right)$$

Si los inductores L1 y L2 están acoplados, entonces se tiene que el valor de la inductancia debe ser el siguiente:

$$L = \frac{V_{in(min)}}{2 \cdot \triangle I_L \cdot f_{switch}} \cdot D_{max}$$

Selección Switch:

En lo que se refiere a elegir un switch, nosotros elegimos un IGBT, por requerimiento de proyecto, y , por esto, las ecuaciones de este paper no están adecuadas a nuestra situación. Aun así, la corriente que debería aguantar el transistor no cambia, ni tampoco la corriente peak y RMS que pasa por esta.

La corriente peak que pasa por el transistor es la siguiente:

$$I_{Q1_{peak}} = I_{L1_{peak}} + I_{L2_{peak}}$$

La corriente RMS que pasa por esta dado por:

$$I_{Q1_{RMS}} = I_{out} \cdot \sqrt{\frac{\left(V_{out} + V_{in(min)} + V_D\right) \cdot (V_{out} + V_D)}{V_{in(min)}^2}}$$

Selección diodo D1:

En la selección del diodo D1 se debe considerar la corriente peak que debe soportar y el voltaje de reversa. En un SEPIC, la corriente peak es la misma que tiene que aguantar el transistor Q1, y el voltaje de reversa que debe aguantar es el siguiente:

$$V_{reverse_{D1}} = V_{in(max)} + V_{out(max)}$$

Para minimizar la perdida de eficiencia, los diodos Schottky son los recomendados, por la rapidez de conmutación que tienen.

Selección de capacitor C1:

La selección de este capacitor depende de la corriente RMS, la cual es la siguiente:

$$I_{C1_{RMS}} = I_{out} \cdot \sqrt{\frac{V_{out} + V_D}{V_{in(min)}}}$$

El capacitor, ya que la polaridad cambia de un ciclo a otro, tiene que ser uno que no este polarizado, como un capacitor cerámico o de tantalio.

El voltaje que debe aguantar debe ser mayor que el voltaje máximo de entrada.

El ripple de voltaje que tiene que aguantar el capacitor es el siguiente (asumiendo que no tiene RMS):

$$\triangle V_{C1} = \frac{I_{out} \cdot D_{max}}{C1 \cdot f_{sw}}$$

Selección de capacitor de salida, C2:

En un convertidor SEPIC, cuando el transistor Q1 esta prendido, el inductor se esta cargando y la corriente de salida es proporcionada por el capacitor de salida. Como resultado, el capacitor de salida tiene grandes ripples de corriente. Por esto, la corriente RMS que tiene que aguantar es el siguiente:

$$I_{C2_{RMS}} = I_{out} \cdot \sqrt{\frac{V_{out} + V_D}{V_{in(min)}}}$$

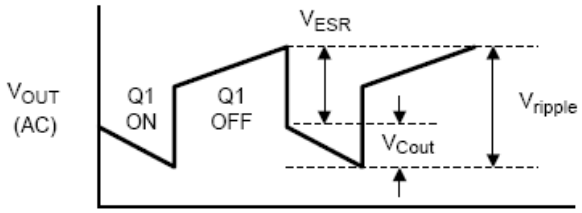


Figura 5. Ripple Voltaje de Salida.

Los valores de ESR y de capacitancia que debe tener el capacitor C2, para obtener cierto ripple de voltaje en la salida, son los siguientes:

$$ESR \leq \frac{V_{Ripple} \cdot 0.5}{I_{L1_{peak}} + I_{L2_{peak}}}$$

$$C_{out} \geq \frac{I_{out} \cdot D}{V_{Ripple} \cdot 0.5 \cdot f_{switch}}$$

El capacitor puede ser cerámico o de tantalio, o, si fuera necesario, por el alto valor de capacitancia que se podría tener, sería más fácil obtener unos capacitores electrolíticos.

Selección capacitancia de entrada Cin:

El uso de un capacitor de entrada en este circuito es para prevenir interacciones de impedancia con la fuente de voltaje de entrada.

El valor que debería tener tiene que ser mayor a 10 [uF], y la corriente RMS que debe soportar es la siguiente:

$$I_{Cin_{RMS}} = \frac{\triangle I_L}{\sqrt{2}}$$

SEÑAL PWM Y SU CONFIGURACIÓN CON MICROCONTROLADOR

La manera de controlar la conmutación del transistor Q1, que en este caso es un IGBT, es a través de un método llamado PWM, Pulse Width Modulation (modulación por ancho de pulsos), el cual se modifica el ciclo de trabajo de una señal periódica a cierta frecuencia.

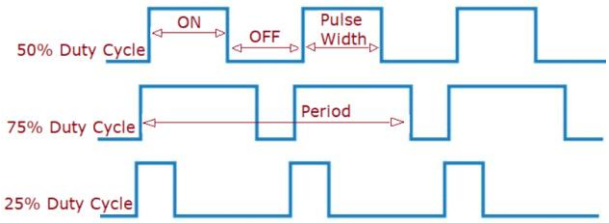


Figura 6. Señal PWM.

Esta señal es entregada al gate del IGBT, el cual se abre y se cierra, y, según el duty y la frecuencia de conmutación, en nuestro caso, se puede modificar el voltaje de salida que entrega el circuito.

Para poder realizar esto, vamos a utilizar el microcontrolador PIC18F4550, el cual, configurando sus registros, podemos programarlo para que, a través de una interfaz gráfica, podamos interactuar con este , a través de una comunicación serial, y manejar la señal PWM que se entrega al gate del transistor IGBT.

El microcontrolador PIC18F4550 es un circuito integrado programable, el cual cuenta con 40 pines para distintas función I/O, de los cuales, vamos a utilizar los siguientes:

-PIN 17 (RC2), salida señal PWM

-PIN 12, GND

-PIN 26 (RC7), RX

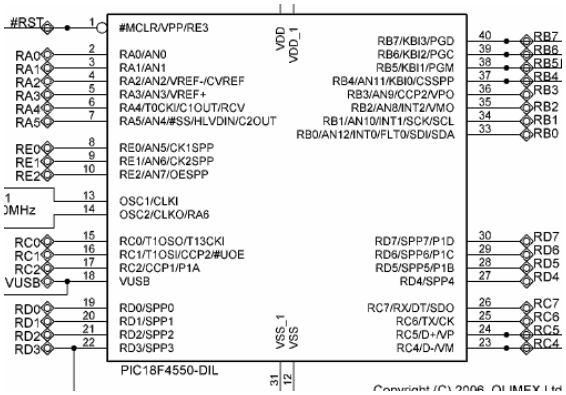


Figura 7. Pines Microcontrolador PIC18F4550.

En el anexo se encuentra el código desarrollado en la aplicación MPLAB, el cual es un programa para la generación de código de microcontroladores.

INTERFAZ GRAFICA

La interfaz grafica funciona para poder tener una interacción entre el usuario y el microcontrolador, el cual, en este caso, se va a utilizar para cambiar el ciclo de trabajo y la frecuencia de la señal PWM.

Los datos van a ser enviados a través de un conversor USB-TTL, a través de comunicación serial.

Se utilizará, como programa para programar la interfaz gráfica, la aplicación de Matlab AppDesigner, la cual esta diseñada para crear interfaces graficas. El código de este se encuentra en el Anexo, y la interfaz grafica se muestra a continuación:

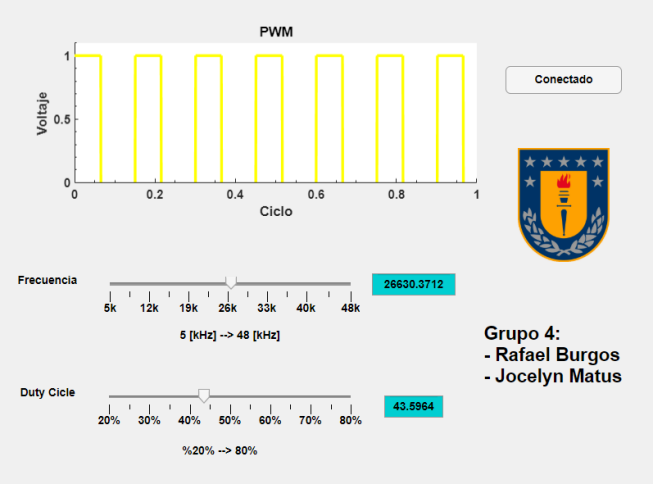


Figura 8. Interfaz Gráfica.

La conexión que se realiza, para poder realizar la comunicación exitosamente, es la siguiente:

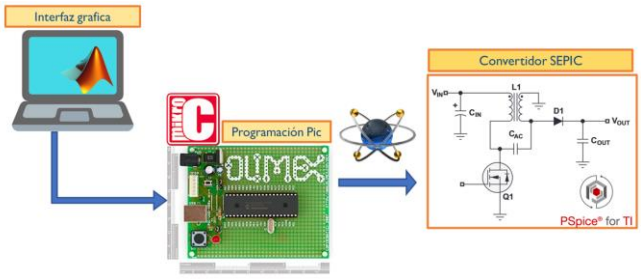


Figura 9. Esquema de comunicación.

SELECCIÓN COMPONENTES

Antes de poder seleccionar los componentes, se tiene que buscar una componente más del circuito: un optoacoplador.

La función del optoacoplador es la de aislar eléctricamente el circuito de potencia con la señal de control, en este caso, con el microcontrolador.

Esto se realiza a través de la siguiente manera:

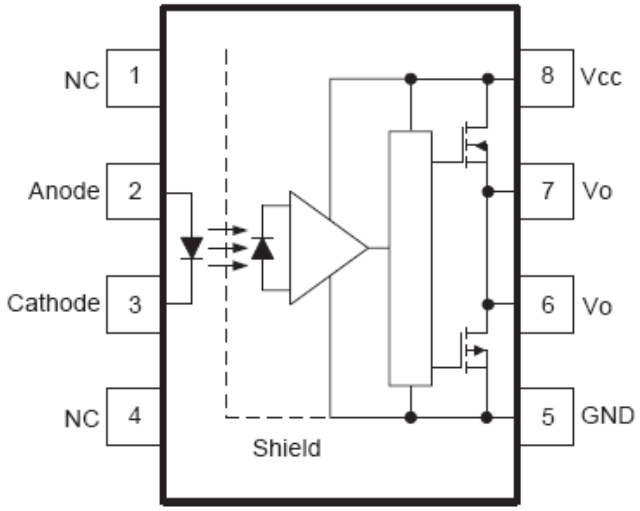


Figura 10. Pines Optoacoplador.

La señal de control entra al pin 2, mientras que el pin 3 esta conectado a la tierra del microcontrolador. La señal activa la emisión de luz por parte del led del lazo izquierdo, lo cual activa al led del lado derecho, lo cual hace que se active el optoacoplador y salga una señal con mayor ganancia en la salida. Hay que tener en consideración que, en el pin 8, hay una fuente de voltaje que alimenta el optoacoplador, el cual debe estar conectada a una tierra distinta a la del microcontrolador. Esto es para aislar, totalmente, la señal de control con la señal que entrega el optoacoplador.

En nuestro caso, se eligió el optoacoplador LTV-3120, ya que fue creado para este tipo de aplicaciones, al tener una conmutación rápida y entregar la corriente y voltaje suficiente para activar el IGBT.

Ahora, para poder saber que componentes hay que elegir, se tiene que realizar los cálculos de las ecuaciones que vimos anteriormente. Para esto, se tiene los siguientes requerimientos:

Potencia	5 [W]
Voltaje entrada	12 [V]
Voltaje salida	5 [V]
Ripple voltaje	0.5% a 1%
Ripple corriente	7% a 10%
Carga	Resistiva
Frecuencia conmutación	20 [kHz]

A partir de esto, se hacen algunos cálculos, con algunos valores impuestos por nosotros:

$$R = 10\ [\Omega]$$

$$I_{out} = \sqrt{\frac{P}{R}} = \sqrt{\frac{5\ [W]}{10\ [\Omega]}} = 0.707\ [A]$$

Entonces, con estos valores, y con los valores impuestos, se procede a calcular las ecuaciones:

Calculo de Duty Cicle

$$Duty := \frac{Vout + V\_D}{Vin + Vout + V\_D} = 0.323$$

$$Duty\_max := \frac{Vout + V\_D}{Vin\_min + Vout + V\_D} = 0.323$$

$$Duty\_min := \frac{Vout + V\_D}{Vin\_max + Vout + V\_D} = 0.323$$

Figura 11. Calculo Duty Cicle.

$$Ripple\_I\_inductor := I\_out \cdot \frac{Vout}{Vin\_min} \cdot 0.10 = 0.347$$

$$I\_peak\_inductor1 := I\_out \cdot \frac{Vout + V\_D}{Vin\_min} \cdot \left(1 + \frac{0.1}{2}\right) = 4.165$$

$$I\_peak\_inductor2 := I\_out \cdot \left(1 + \frac{0.1}{2}\right) = 0.83$$

Figura 12. Calculo Corriente Inductor.

Ahora, antes de seguir, se tiene que decidir si se va a ocupar inductores no acoplados, inductores separados, o un inductor acoplado. En nuestro caso, para no trabajar manualmente con los inductores, como enrollarlo y buscar materiales, elegimos usar el inductor acoplado.

Además de esto, hay ciertos beneficios de usarlo, como de que es más compacto en el circuito y reduce la fuga de inductancia que puede haber con dos inductores separados.

Entonces, por esto, se elige una inductancia acoplada.

$$L\_inductancia\_coupled\_2013 := \frac{Vin\_min \cdot Duty\_max}{2 \cdot Ripple\_I\_inductor \cdot f\_sw} = 6.853 \times 10^{-5}$$

Figura 13. Calculo Valor Inductancia.

Para el diodo, se tiene lo siguiente:

- Seleccion de diodo

$$Reverse\_voltage\_diode := Vin\_max + Vout\_max = 17$$

Figura 14. Calculo Voltaje Reversa Diodo.

Y, tiene que soportar esta corriente peak:

$$I\_peak\_Q1 := I\_peak\_inductor1 + I\_peak\_inductor2 = 4.995$$

Figura 15. Calculo Corriente Peak Diodo.

Además, se debe tener en cuenta que tiene que ser un diodo Schottky.

Para la selección del transistor, este tiene que soportar la misma corriente peak que la del diodo D1, y tiene que soportar, también, esta corriente RMS:

$$I\_RMS\_Q1 := I\_out \cdot \sqrt{\frac{(Vout + Vin\_min + V\_D) \cdot (Vout + V\_D)}{Vin\_min^2}} = 4.344$$

Figura 16. Calculo Corriente RMS Q1.

Para la selección del capacitor de al medio, C1, tiene que soportar esta corriente RMS:

$$I\_RMS\_Cp := I\_out \cdot \left(\sqrt{\frac{Vout + V\_D}{Vin\_min}}\right) = 1.771$$

Figura 17. Calculo Corriente RMS Capacitor de al medio.

Además, el capacitor debe aguantar más que el voltaje de entrada, 12 [V], y, en nuestro caso, elegimos uno que sea de valor 10 [uF].

Con este valor, se tiene que siguiente valor de ripple:

$$Ripple\_V\_Cp := \frac{I\_out \cdot (Duty\_max)}{Cp \cdot f\_sw} = 3.296$$

Figura 18. Calculo Ripple Voltaje Capacitor de al medio.

Además, se tiene que considerar no tiene que ser un capacitor polarizado.

Para el capacitor de salida, tiene que aguantar esta corriente RMS:

$$I\_Cout\_rms := I\_out \cdot \sqrt{\frac{Vout + V\_D}{Vin\_min}} = 1.771$$

Figura 19. Calculo Corriente RMS Capacitor de Salida.

Tiene que poder aguantar el voltaje de salida, y tener un RMS menor o igual al siguiente valor:

$$ESR\_Cout := \frac{Ripple\_V \cdot 0.5}{I\_peak\_inductor1 + I\_peak\_inductor2} = 5.005 \times 10^{-3}$$

Figura 20. Calculo ESR Capacitor de salida.

Y un valor mayor o igual de capacitancia de este valor:

$$Cout := \frac{I\_out \cdot Duty}{Ripple\_V \cdot 0.5 \cdot f\_sw} = 5.104 \times 10^{-5}$$

Figura 21. Calculo Valor Capacitancia de Salida.

Para el capacitor de salida, debe tener un valor mayor a 20 [uF], y debe soportar esta corriente RMS:

$$I\_RMS\_Cin := \frac{Ripple\_I\_inductor}{\sqrt{12}} = 0.1$$

Figura 22. Calculo Corriente RMS Capacitor de entrada.

Entonces, con estos valores, se eligieron los siguientes componentes (los datasheets se encontrarán en Anexo, con links hacia estos):

Inductor acoplado: SRF0703-680M.

Figura 23. Inductor acoplado.

Diodo Schottky: 1N5822.

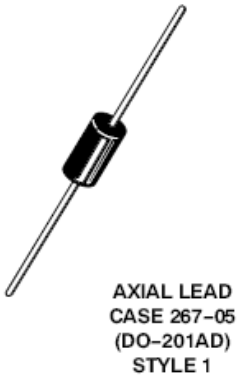


Figura 24. Diodo Schottky.

IGBT: FGH60N60SFD.

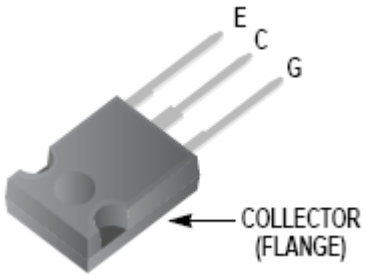


Figura 25. Transistor IGBT.

Capacitor de al medio, C1:

Un capacitor cerámico de 10 [uF] con un ERS de 0.008 [Ω], y de un valor de voltaje de 25 [V].

Capacitancia de salida, C2:

Un capacitor cerámico, idealmente, si no, dos electrolíticos, de valor 75 [uF], y un ESR de 0.014 [Ω], y valor de voltaje de 25 [V].

Capacitor de entrada:

Un capacitor de 20 [uF], que aguante 25[V].

Calculo potencia:

Al ya tener los componentes ya seleccionados, se puede calcular la potencia que consumen estos, ya que los valores para calcular la potencia ya están definidos en los datasheets de los componentes. En nuestro caso, vamos a calcular la potencia para el IGBT y el diodo Schottky.

IGBT:

Se tiene el siguiente calculo:

Perdida de conduccion $Vce := 1.25$

$rc\_grafico := \frac{1.25}{15} = 0.083$ $Ic\_avg := 0.703$ $Ic\_RMS := 5.243$

$Pcondu := Vce \cdot Ic\_avg + rc\_grafico \cdot Ic\_RMS^2 = 3.17$

Perdida de conmutacion:

$E\_total := (2 + 3) \cdot 10^{-3} = 5 \times 10^{-3}$ $f\_sw := 20000$

$Pconmu := E\_total \cdot f\_sw = 100$

Perdida total:

$P\_total\_IGBT := Pcondu + Pconmu = 103.17$

Figura 26. Calculo Potencia IGBT.

Se puede ver que, en la conmutación, es donde se consume más potencia, mientras que en la perdida de conducción apenas consume un 3% de la potencia total consumida.

También, la potencia total esta dentro del rango para la disipación de potencia máxima, por lo que debería trabajar correctamente.

Diodo Schottky:

Se tiene los siguientes cálculos:

Datos

$Vf := 0.4$ $If := 2$ $T := \frac{1}{f\_sw} = 5 \times 10^{-5}$ $ton := 0.315 \cdot T = 1.575 \times 10^{-5}$

$Vr := 8$ $Ir := 0.4 \cdot 10^{-3}$

Potencia on:

$Pon := Vf \cdot If \cdot \left(\frac{ton}{T}\right) = 0.252$

Potencia off:

$Poff := Vr \cdot Ir \cdot \left(\frac{ton}{T}\right) = 1.008 \times 10^{-3}$

Potencia total

$Potencia\_diodo := Pon + Poff = 0.253$

Figura 27. Calculo Potencia Diodo Schottky.

La potencia consumida por el diodo es muy baja en comparación con el IGBT. En este caso, los cálculos de potencia, cuando hace el switch, no se consideraron, ya que, según el datasheet, son demasiado bajas para ser consideradas.

Simulación Circuito

En la realización del circuito, se realizó unas modificaciones, más que nada para poder simular con los componentes que se van a trabajar en la vida real, ya que no se pudieron conseguir los componentes que queríamos.

Además, ya que no se pudo encontrar un modelo para el inductor acoplado, se uso un transformador modificado, con los valores más parecidos posibles al del inductor acoplado.

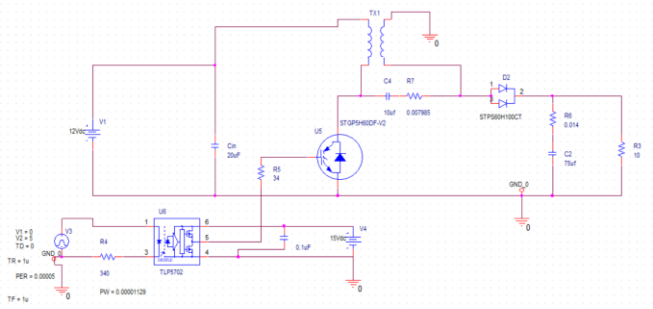


Figura 28. Esquemático Circuito PSPICE.

Entonces, con este esquemático, se va a simular el circuito en el programa Orcad Allegro

Voltaje de salida:

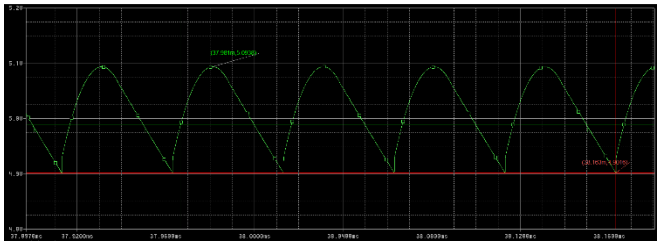


Figura 29. Voltaje de Salida.

Corriente de salida:

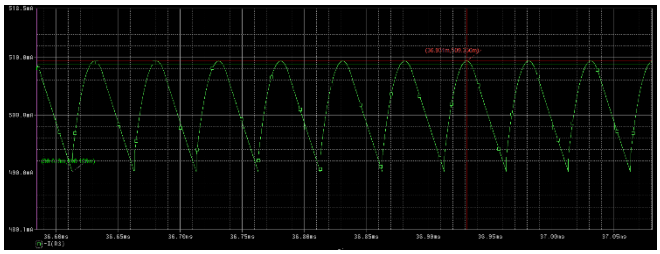


Figura 30. Corriente de Salida.

Se puede notar de que el ripple del voltaje es de aproximadamente 1.8%, el cual fue calculado gráficamente, el cual es cercano al valor impuesto en el proyecto. Esto se puede explicar por la selección de componentes que se realizó, y que estaba a nuestro alcance.

En el ripple de corriente, se calculo un ripple de 3.4%, el cual, aun con los componentes que elegimos, cumple con el requerimiento del proyecto.

También se nota que estos son los valores que se obtuvieron como máximos y mínimo de cada señal:

	Mínimo	Máximo
Voltaje	5.09 [V]	4.90 [V]
Corriente	509.3 [mA]	490.1 [mA]

Corriente de entrada:

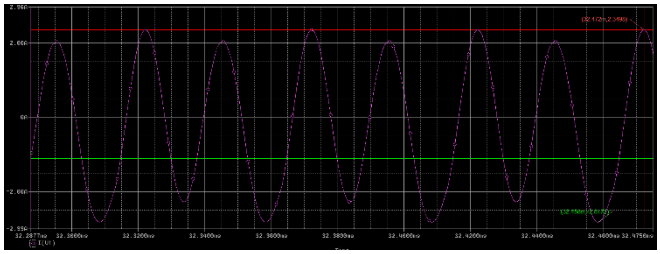


Figura 31. Corriente de Entrada.

La corriente de entrada es aceptable, teniendo un peak de 2.3 [A]. Es una señal bi-periodica.

Corriente inductor:

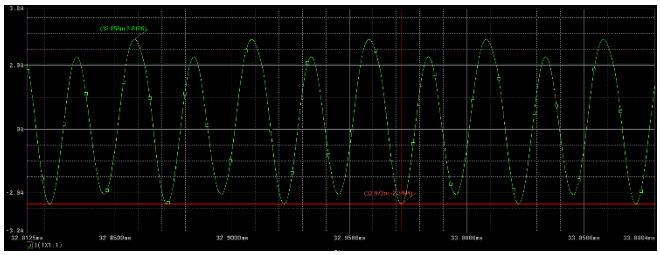


Figura 32. Corriente del Inductor.

Se nota que es la misma que la corriente de entrada.

Voltaje de capacitor de al medio, C1:

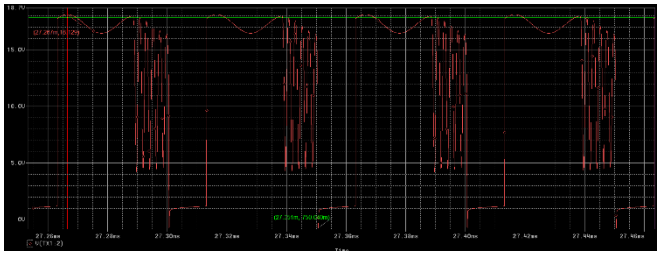


Figura 32. Voltaje Capacitor de al medio C1.

Se parece extraña a lo que debería realmente salir, pero esto se puede explicar por el uso del transformador como equivalente del inductor acoplado. Lamentablemente, es lo más cercano que se pudo llegar en la realidad, pero sirve como guía para los voltajes máximos y mínimos que tiene que aguantar el capacitor C1 en la realidad, que, en este caso, son con un peak de 18 [V], el cual esta dentro de lo que puede aguantar el capacitor.

Corriente colector del IGBT, Q1:

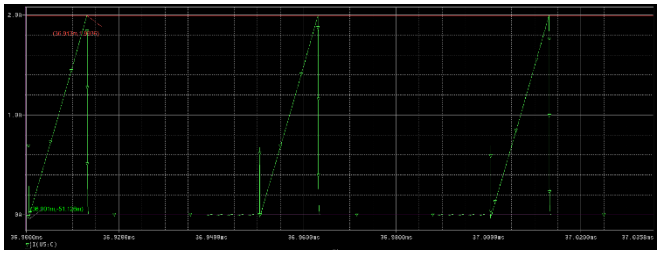


Figura 33. Corriente Colector IGBT.

Se puede ver cuando se apaga y prende el IGBT, con las formas de corriente que se pueden ver. El peak de esta corriente es de aproximadamente 2[A], lo cual no es tan cercano a lo que se calculo teóricamente, pero, aun así, es aceptable.

Corriente capacitor de salida, C2:

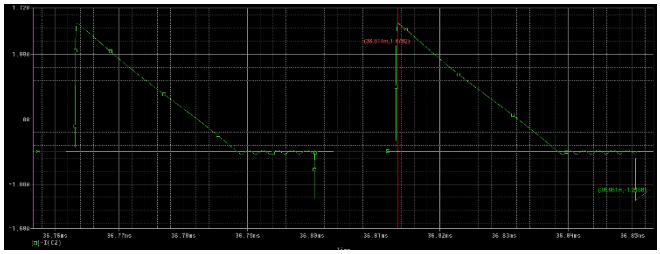


Figura 34. Corriente Capacitor de Salida, C2.

Se puede ver que es una señal triangular, casi exponencial decadente, que es cuando se descarga el condensador. El valor RMS de este es de 0.808 [A], el cual esta alejado del valor que nosotros habíamos calculado, pero, cuando se calculó, se consideró un voltaje mínimo de entrada, en cual, en la simulación, se considero un voltaje de entrada de 12 [V]. Es por esto por lo que entrega ese valor RMS.

Voltaje señal de control:

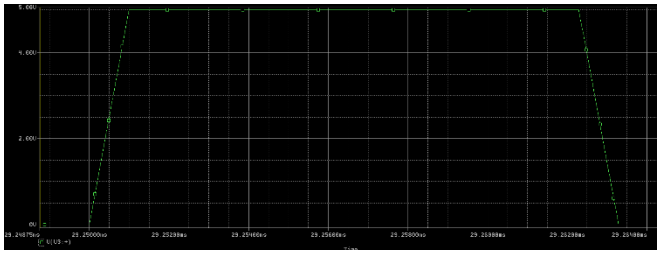


Figura 35. Voltaje Señal de Control.

Aquí se puede ver la señal de control, una señal cuadrada, la cual tiene una subida y bajada de 1 [us], con un valor de amplitud de 5 [V].

Corriente señal de control:

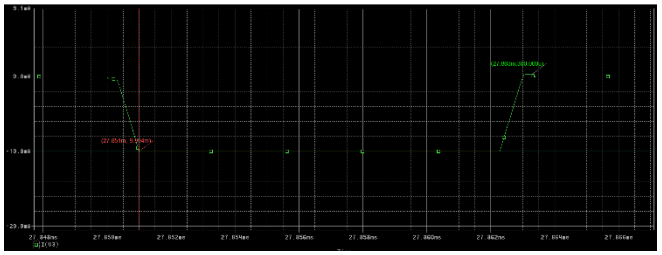


Figura 36. Corriente Señal de Control.

Aquí se puede ver la corriente que se usa para la activación del optoacoplador, la cual llega a ser 10 [mA].

Voltaje IGBT gate:

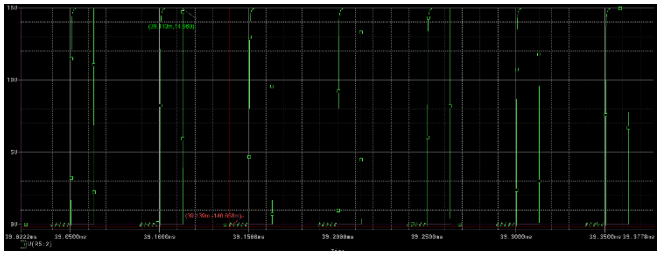


Figura 37. Voltaje IGBT Gate.

Aquí se encuentra el voltaje que se le aplica al gate del IGBT, el cual es una señal cuadrada que llega hasta los 15[V].

Corriente IGBT gate:

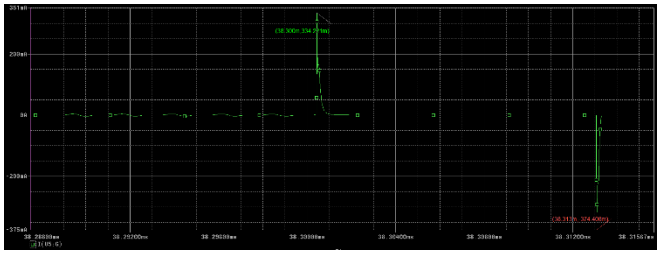


Figura 38. Corriente IGBT Gate.

Aquí se puede ver los peak de corriente de carga y descarga del capacitor parasito del gate del IGBT, el cual llega a valores de magnitud de 370 [mA]. Se puede ver que se alcanza a descargar el IGBT, por lo que hay

no hay problema en que se le pueda aplicar una señal con frecuencia de 20 [kHz].

Ahora, ya que no se pudo encontrar el componente del inductor acoplado como modelo SPICE, pero si se encontró el archivo de 4 puertos de parámetro S, se realizó una conversión de esos parámetros a un modelo SPICE. El circuito modificado es el siguiente:

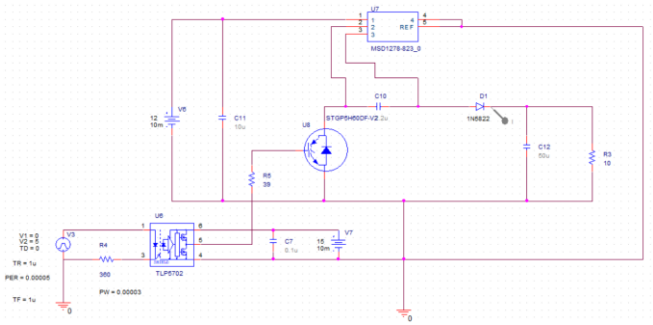


Figura 39. Esquemático circuito actualizado PSPICE.

Entonces, el voltaje de salida y de corriente son los siguientes:

Voltaje de salida:

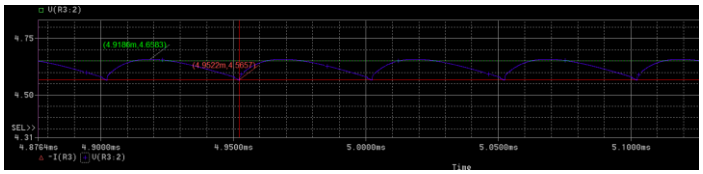


Figura 40. Voltaje de Salida.

Corriente de salida:

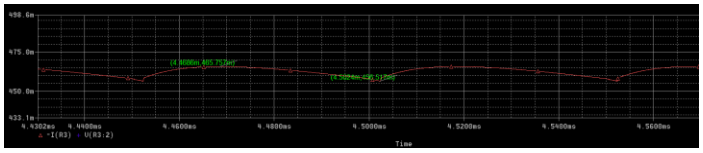


Figura 41. Corriente de Salida.

Los valores de ripple que se obtuvieron fueron los siguientes:

Voltaje: 2.008%

Corriente: 2.004%

Además, se tienen los siguientes gráficos de algunas de las señales restantes:

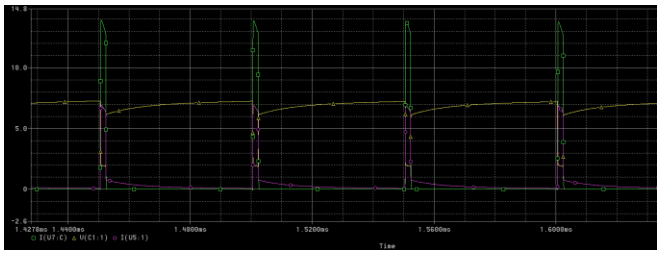


Figura 42. Gráfico Corrientes Inductor Acoplado y IGBT Colector, Voltaje Capacitor Medio.

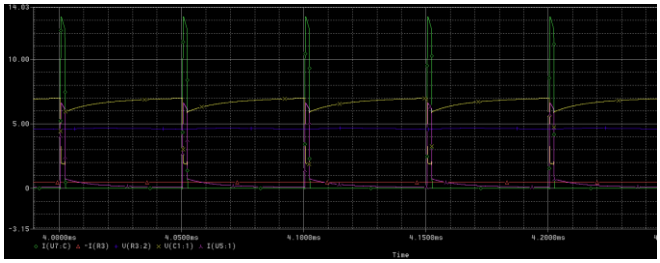


Figura 43. Gráfico anterior, con voltaje y corriente de salida agregados.

Son distintas a las señales que se obtuvieron con el transformador equivalente, siendo mucho más suaves y cercanas a la realidad de como deben ser las señales de un circuito SEPIC.

PCB

La realización de la PCB en simulación 3D fue realizado a través del programa KiCad, el cual es un programa de libre acceso, que se usa para el diseño de PCB en acido.

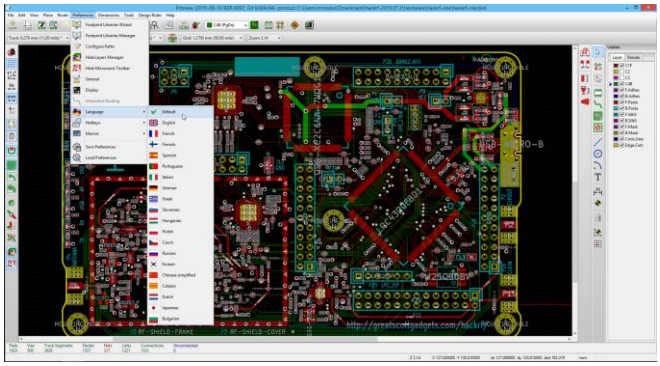


Figura 44. Programa KiCad.

El esquemático que se realizo es el siguiente:

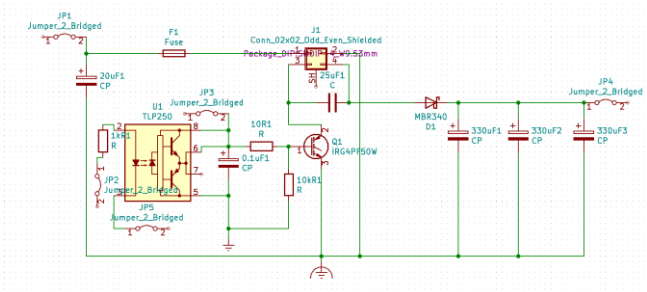


Figura 45. Esquemático Circuito PCB.

Y estas serían las pistas que estarían creadas en la placa de cobre:

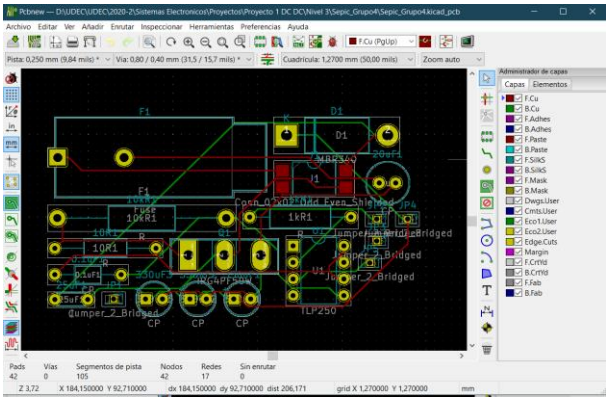


Figura 46. Circuito con placas de cobre y caminos.

Y ahora se ve de manera grafica como quedaría, con un modelo 3D:

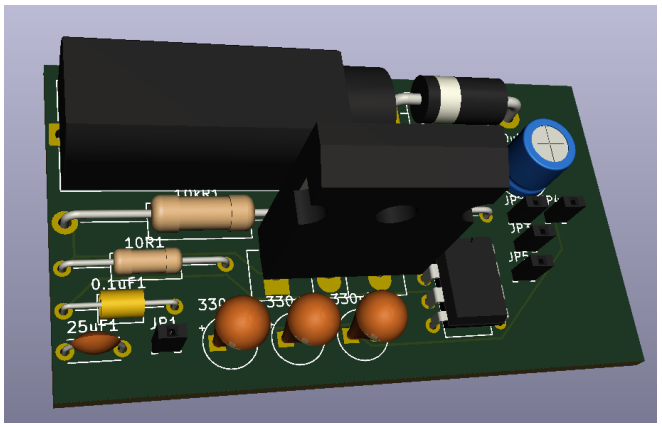


Figura 47. Placa PCB, vista superior.

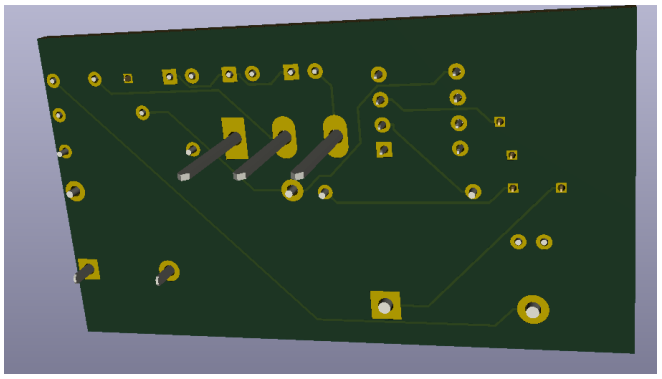


Figura 48. Placa PCB, vista inferior.

CONCLUSIÓN

La realización del proyecto fue ardua por la complejidad del circuito, además del estudio e investigación de la señal de control y su implementación con la interfaz gráfica. De manera general, los proyectos de esta índole son difíciles pero emocionantes de realizar, por el manejo técnico que la persona tiene que tener y la motivación que se debe tener para poder terminarlo.

Dentro de todo, fue una experiencia gratificante.

REFERENCIAS

[1] *SEPIC Equations and Component Ratings*, Maxim Integrated Products. [Online]. Disponible en: https://pdfserv.maximintegrated.com/en/an/AN1051.pdf

[2] D. Zhang, *Designing a SEPIC converter*, Texas Instruments, AN-1484, 2006. [Online]. Disponible en: https://www.ti.com/lit/an/snva168e/snva168e.pdf

ANEXO

Anexo A: Datasheets.

Inductor acoplado: https://www.coilcraft.com/getmedia/9bb1bfe8-4110-44c7-9e7e-54ca2b271b64/msd1278.pdf

Optoacoplador: https://media.digikey.com/pdf/data%20sheets/lite-on%20pdfs/ltv-3120_series_rev1.pdf

Diodo Schottky: https://www.st.com/resource/en/datasheet/1n5822.pdf

IGBT: https://belchip.by/sitedocs/17500.pdf

Anexo B: Código MicroPIC 18F4550 en MPLAB.

```
#include <xc.h> //librerias
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include "config.h"

// CONFIG1L
#pragma config PLLDIV = 5       //
#pragma config CPUDIV = OSC1_PLL2//
#pragma config USBDIV = 2       //

// CONFIG1H
#pragma config FOSC = HSPLL_HS  //
#pragma config FCMEN = OFF      //
#pragma config IESO = OFF       //

// CONFIG2L
#pragma config PWRT = OFF       //
#pragma config BOR = ON         //
#pragma config BORV = 3         //
#pragma config VREGEN = ON      //

// CONFIG2H
#pragma config WDT = ON         //
#pragma config WDTPS = 32768    //

// CONFIG3H
#pragma config CCP2MX = ON      //
#pragma config PBADEN = OFF     //
#pragma config LPT1OSC = ON     //
#pragma config MCLRE = ON       //
```

Figura 49. Bits de configuración.

```

// CONFIG4L
#pragma config STVREN = ON
#pragma config LVP = ON
#pragma config ICPRT = OFF
#pragma config XINST = OFF

// CONFIG5L
#pragma config CP0 = OFF
#pragma config CP1 = OFF
#pragma config CP2 = OFF
#pragma config CP3 = OFF

// CONFIG5H
#pragma config CPB = OFF
#pragma config CPD = OFF

// CONFIG6L
#pragma config WRT0 = OFF
#pragma config WRT1 = OFF
#pragma config WRT2 = OFF
#pragma config WRT3 = OFF

// CONFIG6H
#pragma config WRTC = OFF
#pragma config WRTB = OFF
#pragma config WRTD = OFF
```

Figura 50. Bits de configuración.

```
#pragma config WRTD = OFF

// CONFIG7L
#pragma config EBTR0 = OFF
#pragma config EBTR1 = OFF
#pragma config EBTR2 = OFF
#pragma config EBTR3 = OFF

// CONFIG7H
#pragma config EBTRB = OFF
```

Figura 51. Bits de configuración.

```
void Init_PWM() //inicia el modulo PWM
{
    PR2 = 74;           // frecuencia
    CCPR1L = 43;        // duty cycle
    TRISCbits.RC2 = 0;          // pin RC2 como salida
    CCP1CONbits.CCP1M = 0b1100; // modo PWM
    T2CONbits.TMR2ON = 1;       // enable timer 2
    T2CONbits.T2CKPS = 0b01;    //  prescale = 4

}
```

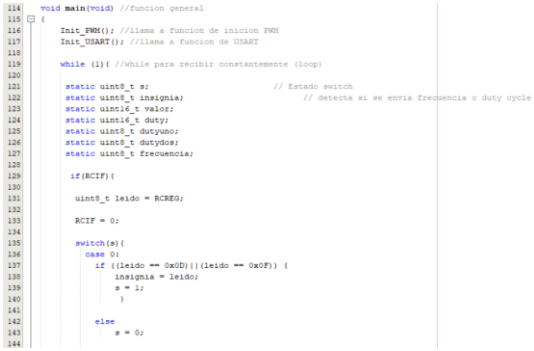
Figura 52. Función Init_PWM().

```
void Init_USART() //Se inicia el modulo USART
{
    //Baud rate
    SPBRG = 25; // 77 -> 9600; 25 -> 115200
    BAUDCON= 0;
    // salida y entrada
    TRISCbits.RC6 = 0; // Tx
    TRISCbits.RC7 = 1; // Rx

    // setup PIC
    TXSTAbits.BRGH = 1; // 0 low speed , 1 high speed
    TXSTAbits.SYNC = 0; // 0 asincrono, 1 sincrono
    RCSTAbits.SPEN = 1; // habilitar Tx y Rx
    RCIE = 1;           // interrupcion serial

    // activar transmision y recepcion
    TXSTAbits.TX9 =  0;      // 8 bits
    TXSTAbits.TXEN = 1;     // activar transmision
    RCSTAbits.RX9 =  0;      // 8 bits
    RCSTAbits.CREN = 1;     // activar recepcion
    RCSTAbits.SREN = 0;
    RCSTAbits.ADDEN =0;
    RCSTAbits.FERR = 0;
    RCSTAbits.OERR = 0;
    RCSTAbits.RX9D = 0;
    TXSTAbits.SENDB =1;
    TXSTAbits.TRMT = 1;
    TXSTAbits.TX9D = 0;
    TXSTAbits.CSRC = 0;
    TXSTAbits.TRMT = 1;
}
```

Figura 53. Init_USART().



```
void main(void) //funcion general
{
    Init_PWM(); //llama a funcion de inicion PWM
    Init_USART(); //llama a funcion de USART

    while (1){ //while para recibir constantemente (loop)

     static uint8_t s;                          // Estado switch
     static uint8_t insignia;                       // detecta si se envia frecuencia o duty cycle
     static uint16_t valor;
     static uint16_t duty;
     static uint8_t dutyuno;
     static uint8_t dutydos;
     static uint8_t frecuencia;

      if(RCIF){

       uint8_t leido = RCREG;

       RCIF = 0;

       switch(s){
         case 0:
           if ((leido == 0x0D)||(leido == 0x0F)) {
               insignia = leido;
               s = 1;
                }

           else
               s = 0;
```

Figura 54. Función principal.

```
            break;

          case 1:
            valor = leido;
            s = 2;

            break;

          case 2:
            if(insignia == 0x0D){
              valor |= ((uint16_t)leido << 8);
              duty = valor;
              dutyuno = (uint8_t)(valor>>2);
              dutydos = (uint8_t)valor;
              CCPR1L = dutyuno;
              CCP1CONbits.DC1B = dutydos;
                  }

            else if (insignia == 0x0F) {
              valor |= ((uint16_t)leido << 8);
              frecuencia = (uint8_t)valor;
              PR2 = frecuencia;
                  }
            s = 0;

            break;

          default:
          s = 0;
            }
```

Figura 55. Función principal.

Anexo C: Código de interfaz gráfica.

```
classdef app1_modificado < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure  matlab.ui.Figure
        UIAxes    matlab.ui.control.UIAxes
        slider    matlab.ui.control.Slider
        text6     matlab.ui.control.Label
        sliderd   matlab.ui.control.Slider
        editd     matlab.ui.control.EditField
        edit      matlab.ui.control.EditField
        text7     matlab.ui.control.Label
        onoff     matlab.ui.control.Button
        Image     matlab.ui.control.Image
        Grupo4RafaelBurgosJocelynMatusLabel  matlab.ui.control.Label
        texto     matlab.ui.control.Label
        textod    matlab.ui.control.Label
    end


    properties (Access = private)   %% Función comunicación PIC

        DutyCycle; % Property
        Frequ;     % Property
        SerPort;   % Property

        Fosc = 48*10^(6);         % Freq PIC
        Tosc = 1/(48*10^(6));     % Tmpo Oscil.
        PS_tmr2 = 4;              % Prescaler

        CCPRxL;
```

Figura 56. Código inicio interfaz gráfica.

```
        %Variables de bits fwrite Freq
        F = hex2dec('0F');        % Primer fwrite Freq
        PR2;
        null = 0;                 % Tercer fwrite Freq (8 bits de 0s)

        %Variables de bits fwrite Duty
        D = hex2dec('0D');        % Primer fwrite Duty
        ccpfw2;                   % Segundo fwrite Duty
        ccpfw3;                   % Tercer fwrite Duty

        Conectado;

        Vout;                     % Vout teórico
    end


    methods (Access = private)     %% Inicializa funciones



        function DutyCalc(app)       % Duty a enviar

            % 0x0D se envia en fwrite1 duty (ver funcion Serial)

            % Calculo CCPRxL
            app.CCPRxL = 4*(app.PR2+1)*((app.DutyCycle)/100); % Valor DutyCycle a enviar
```

Figura 57.

```
            % Declarar uint16(*) con valor de CCPRxL // Variable 'nueva' a operar (2 bytes)
            ccpduty = uint16(app.CCPRxL);

            %% 2do fwrite_ducy (envía 1 byte de los 2 bytes)
            % A uint16 >>8 (hacia los LSB), enviar esos 8bits LSB
            ccp2MSB = bitsrl(ccpduty, 8); % 8MSB >>8

            app.ccpfw2 = ccp2MSB;
            % ccpfw2 se envia en fwrite3 duty

            %% 3er fwrite_ducy (meter variable 2 bytes originales, enviar 1byte)
            % Con &&0xFF, tomar los LSB, guardarlo en 8bits, enviar esos LSB

            ccplsb = uint16(255);   % 255 = 00000 0000 1111 1111 en binario // Máscara a aplicar

            app.ccpfw3 = bitand(ccpduty, ccplsb);      %Recoge los LSB a enviar (multiplica binarios ccplsb
            % ccpfw3 se envia en fwrite2 duty

            if app.Conectado

                FWrites(app);

            end

        end
```

Figura 58.

```
        function FreqCalc(app)       % Calculo PR2,    Frecuencia a enviar

            % 0x0F se envia en fwrite1 freq (ver funcion Serial)

            Tpwm = 1/(app.Frequ);     % Tmpo PWM

            % Calculo PR2
            app.PR2 = (Tpwm/(4*(app.Tosc)*(app.PS_tmr2)))-1;
            % pr2 a enviar en fwrite2 freq

            % byte de 0s se envía en fwrite3 freq (ver funcion Serial)

            DutyCalc(app);    % Con PR2 calculado llamada a ejecutar funcion Dut


        end
```

Figura 59.

```
        function FWrites(app)

            % Valores properties
            %Freq
            fF = app.F;
            fPR2 = app.PR2;
            fnull = app.null;
            %Duty
            fD = app.D;
            fccpfw2 = app.ccpfw2;
            fccpfw3 = app.ccpfw3;
            serpic = serial('COM3');     % Nombre asignado al PIC
            set(serpic,'BaudRate',115200);
            fopen(serpic);

            %% FWRITES DE FREQ
                %% 1er fwrite_freq (1byte)
                % 0x0F
                fwrite(serpic, fF, 'uint8');   %1er fwrite

                %% 2do fwrite_freq (1byte)
                % PR2 (1byte)
                fwrite(serpic, fPR2, 'uint8');          % LISTO

                %% 3er fwrite_freq (1byte)
                % byte de 0s
                fwrite(serpic, fnull, 'uint8');             %LISTO

            %% FWRITES DE DUTY
                %% 1er fwrite_ducy (1byte)
                % 0x0D
                fwrite(serpic, fD, 'uint8');
```

Figura 60.

```
                %% 2do fwrite_ducy (1byte)
                % Envía los 8 LSB del valor del uint16 de DutyCycle
                fwrite(serpic, fccpfw3, 'uint8');

                %% 3er fwrite_ducy (1byte)
                % Envía los 8 MSB >>8 (MSBs hacia los LSB)
                fwrite(serpic, fccpfw2, 'uint8');

            fclose(serpic);

        end
    end  %% Termina de declarar funciones
```

Figura 61.

```
    % Callbacks that handle component events
    methods (Access = private)

        % Value changed function: edit
        function editValueChanged(app, event)
            frecuencia= str2double(app.edit.Value);
            duty= app.sliderd.Value;
            app.slider.Value= frecuencia;
            app.editd.Value=num2str(duty);
            largo=0:0.001:1;

             if (frecuencia>=5000)&&(frecuencia<=1.114285714285714e+04)
                q=square(2*pi*frecuencia*largo/500, duty);

            elseif (frecuencia>1.114285714285714e+04)&&(frecuencia<=1.728571428571429e+04)
                q=square(2*pi*frecuencia*largo/1000, duty);

            elseif (frecuencia>1.728571428571429e+04)&&(frecuencia<=2.342857142857143e+04)
                q=square(2*pi*frecuencia*largo/2000,duty);

            elseif (frecuencia>2.342857142857143e+04)&&(frecuencia<=2.957142857142857e+04)
                q=square(2*pi*frecuencia*largo/4000,duty);

            elseif (frecuencia>2.957142857142857e+04)&&(frecuencia<=3.571428571428572e+04)
                q=square(2*pi*frecuencia*largo/6000,duty);

            elseif (frecuencia>3.571428571428572e+04)&&(frecuencia<=4.185714285714286e+04)
                q=square(2*pi*frecuencia*largo/8000,duty);

            elseif (frecuencia>4.185714285714286e+04)&&(frecuencia<=48000)
                q=square(2*pi*frecuencia*largo/10000,duty);
```

Figura 62.

```
 end

if ((frecuencia<=48000) && (frecuencia>=5000))

    plot(app.UIAxes,largo,q,'y','linewidth',2);


    app.texto.Text = '5 [kHz] --> 48 [kHz]';

    app.Frequ= frecuencia;
    app.DutyCycle= duty;
    FreqCalc(app)


else
    app.texto.Text = 'Por favor, ingrese un valor entre 5 [kHz] y 48 [kHz]';

end
```

Figura 63.

```
        end

        % Value changed function: slider
        function sliderValueChanged(app, event)
            frecuencia= app.slider.Value;
            duty= app.sliderd.Value;
            app.edit.Value= num2str(frecuencia);
            app.editd.Value=num2str(duty);
            largo=0:0.001:1;

             if (frecuencia>=5000)&&(frecuencia<=1.114285714285714e+04)
                q=square(2*pi*frecuencia*largo/500, duty);

            elseif (frecuencia>1.114285714285714e+04)&&(frecuencia<=1.728571428571429e+04)
                q=square(2*pi*frecuencia*largo/1000, duty);

            elseif (frecuencia>1.728571428571429e+04)&&(frecuencia<=2.342857142857143e+04)
                q=square(2*pi*frecuencia*largo/2000,duty);

            elseif (frecuencia>2.342857142857143e+04)&&(frecuencia<=2.957142857142857e+04)
                q=square(2*pi*frecuencia*largo/4000,duty);

            elseif (frecuencia>2.957142857142857e+04)&&(frecuencia<=3.571428571428572e+04)
                q=square(2*pi*frecuencia*largo/6000,duty);

            elseif (frecuencia>3.571428571428572e+04)&&(frecuencia<=4.185714285714286e+04)
                q=square(2*pi*frecuencia*largo/8000,duty);

            elseif (frecuencia>4.185714285714286e+04)&&(frecuencia<=48000)
                q=square(2*pi*frecuencia*largo/10000,duty);

             end
```

Figura 64.

```
            if ((frecuencia<=48000) && (frecuencia>=5000))

                plot(app.UIAxes,largo,q,'y','linewidth',2);


                app.texto.Text = '5 [kHz] --> 48 [kHz]';

                app.Frequ= frecuencia;
                app.DutyCycle= duty;
                FreqCalc(app)

            else
                app.texto.Text = 'Por favor, ingrese un valor entre 5 [kHz] y 48 [kHz]';

            end
        end
```

Figura 65.

```
        % Value changed function: sliderd
        function sliderdValueChanged(app, event)
            frecuencia= app.slider.Value;
            duty= app.sliderd.Value;
            app.edit.Value= num2str(frecuencia);
            app.editd.Value=num2str(duty);
            largo=0:0.001:1;

             if (frecuencia>=5000)&&(frecuencia<=1.114285714285714e+04)
                q=square(2*pi*frecuencia*largo/500, duty);

            elseif (frecuencia>1.114285714285714e+04)&&(frecuencia<=1.728571428571429e+04)
                q=square(2*pi*frecuencia*largo/1000, duty);

            elseif (frecuencia>1.728571428571429e+04)&&(frecuencia<=2.342857142857143e+04)
                q=square(2*pi*frecuencia*largo/2000,duty);

            elseif (frecuencia>2.342857142857143e+04)&&(frecuencia<=2.957142857142857e+04)
                q=square(2*pi*frecuencia*largo/4000,duty);

            elseif (frecuencia>2.957142857142857e+04)&&(frecuencia<=3.571428571428572e+04)
                q=square(2*pi*frecuencia*largo/6000,duty);

            elseif (frecuencia>3.571428571428572e+04)&&(frecuencia<=4.185714285714286e+04)
                q=square(2*pi*frecuencia*largo/8000,duty);

            elseif (frecuencia>4.185714285714286e+04)&&(frecuencia<=48000)
                q=square(2*pi*frecuencia*largo/10000,duty);

             end
```

Figura 66.

```
            if ((duty<=80) && (duty>=20))

                plot(app.UIAxes,largo,q,'y','linewidth',2);


                app.textod.Text = '%20% --> 80%';

                app.Frequ= frecuencia;
                app.DutyCycle= duty;
                FreqCalc(app)
            else
                app.textod.Text = 'Por favor, ingrese un valor entre 20% y 80%';

            end
```

Figura 67.

```
        end

        % Value changed function: editd
        function editdValueChanged(app, event)
            frecuencia= app.slider.Value;
            duty= str2double(app.editd.Value);
            app.edit.Value= num2str(frecuencia);
            app.sliderd.Value=duty;
            largo=0:0.001:1;

             if (frecuencia>=5000)&&(frecuencia<=1.114285714285714e+04)
                q=square(2*pi*frecuencia*largo/500, duty);

            elseif (frecuencia>1.114285714285714e+04)&&(frecuencia<=1.728571428571429e+04)
                q=square(2*pi*frecuencia*largo/1000, duty);

            elseif (frecuencia>1.728571428571429e+04)&&(frecuencia<=2.342857142857143e+04)
                q=square(2*pi*frecuencia*largo/2000,duty);

            elseif (frecuencia>2.342857142857143e+04)&&(frecuencia<=2.957142857142857e+04)
                q=square(2*pi*frecuencia*largo/4000,duty);

            elseif (frecuencia>2.957142857142857e+04)&&(frecuencia<=3.571428571428572e+04)
                q=square(2*pi*frecuencia*largo/6000,duty);

            elseif (frecuencia>3.571428571428572e+04)&&(frecuencia<=4.185714285714286e+04)
                q=square(2*pi*frecuencia*largo/8000,duty);

            elseif (frecuencia>4.185714285714286e+04)&&(frecuencia<=48000)
                q=square(2*pi*frecuencia*largo/10000,duty);

             end
```

Figura 68.

```
            if ((duty<=80) && (duty>=20))

                plot(app.UIAxes,largo,q,'y','linewidth',2);


                app.textod.Text = '%20% --> 80%';

                app.Frequ= frecuencia;
                app.DutyCycle= duty;
                FreqCalc(app)
            end
        end
```

Figura 69.

```
    % Button pushed function: onoff
    function onoffButtonPushed(app, event)
        info = strcmp(app.onoff.Text,'Conectado');%string compare para poder comparar las palabras y as
        if    info== 1
            app.onoff.Text= 'Desconectado';
            app.Conectado= 0;


        elseif info == 0
            app.onoff.Text= 'Conectado';
            app.Conectado=1;
        end
    end
end
```

Figura 70.

```
    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'PWM')
            xlabel(app.UIAxes, 'Ciclo')
            ylabel(app.UIAxes, 'Voltaje')
            app.UIAxes.PlotBoxAspectRatio = [2.88343558282209 1 1];
            app.UIAxes.FontWeight = 'bold';
            app.UIAxes.YLim = [0 1.1];
            app.UIAxes.XMinorTick = 'on';
            app.UIAxes.YMinorTick = 'on';
            app.UIAxes.Position = [35 266 441 186];

            % Create slider
            app.slider = uislider(app.UIFigure);
            app.slider.Limits = [5000 48000];
            app.slider.MajorTicks = [5000 12000 19000 26000 33000 40000 48000];
            app.slider.MajorTickLabels = {'5k', '12k', '19k', '26k', '33k', '40k', '48k'};
            app.slider.ValueChangedFcn = createCallbackFcn(app, @sliderValueChanged, true);
            app.slider.MinorTicks = [8500 15500 22500 29500 36500 44000];
            app.slider.Tag = 'slider';
            app.slider.FontSize = 11;
            app.slider.FontWeight = 'bold';
```

Figura 71.

```
            app.slider.FontWeight = 'bold';
            app.slider.Position = [110 198 235 3];
            app.slider.Value = 20000;

            % Create text6
            app.text6 = uilabel(app.UIFigure);
            app.text6.Tag = 'text6';
            app.text6.HorizontalAlignment = 'center';
            app.text6.VerticalAlignment = 'top';
            app.text6.FontSize = 11;
            app.text6.FontWeight = 'bold';
            app.text6.Position = [10 188 78 22];
            app.text6.Text = 'Frecuencia';

            % Create sliderd
            app.sliderd = uislider(app.UIFigure);
            app.sliderd.Limits = [20 80];
            app.sliderd.MajorTicks = [20 30 40 50 60 70 80];
            app.sliderd.MajorTickLabels = {'20%', '30%', '40%', '50%', '60%', '70%', '80%'};
            app.sliderd.ValueChangedFcn = createCallbackFcn(app, @sliderdValueChanged, true);
            app.sliderd.MinorTicks = [25 35 45 55 65 75];
            app.sliderd.Tag = 'sliderd';
            app.sliderd.FontSize = 11;
            app.sliderd.FontWeight = 'bold';
            app.sliderd.Position = [109 88 236 3];
            app.sliderd.Value = 60;

            % Create editd
            app.editd = uieditfield(app.UIFigure, 'text');
            app.editd.ValueChangedFcn = createCallbackFcn(app, @editdValueChanged, true);
            app.editd.Tag = 'editd';
            app.editd.HorizontalAlignment = 'center';
            app.editd.FontSize = 11;
```

Figura 72.

```
            app.editd.FontWeight = 'bold';
            app.editd.BackgroundColor = [0 0.8078 0.8196];
            app.editd.Position = [378 69 57 22];
            app.editd.Value = '60';

            % Create edit
            app.edit = uieditfield(app.UIFigure, 'text');
            app.edit.ValueChangedFcn = createCallbackFcn(app, @editValueChanged, true);
            app.edit.Tag = 'edit';
            app.edit.HorizontalAlignment = 'center';
            app.edit.FontSize = 11;
            app.edit.FontWeight = 'bold';
            app.edit.BackgroundColor = [0 0.8078 0.8196];
            app.edit.Position = [366 188 81 22];
            app.edit.Value = '20000';

            % Create text7
            app.text7 = uilabel(app.UIFigure);
            app.text7.Tag = 'text7';
            app.text7.HorizontalAlignment = 'center';
            app.text7.VerticalAlignment = 'top';
            app.text7.FontSize = 11;
            app.text7.FontWeight = 'bold';
            app.text7.Position = [9 78 80 22];
            app.text7.Text = 'Duty Cicle';

            % Create onoff
            app.onoff = uibutton(app.UIFigure, 'push');
            app.onoff.ButtonPushedFcn = createCallbackFcn(app, @onoffButtonPushed, true);
            app.onoff.Tag = 'onoff';
            app.onoff.FontSize = 11;
            app.onoff.FontWeight = 'bold';
```

Figura 73.

```
            app.onoff.FontWeight = 'bold';
            app.onoff.Position = [496 385 114 27];
            app.onoff.Text = 'Apagado';

            % Create Image
            app.Image = uiimage(app.UIFigure);
            app.Image.Position = [501 221 104 111];
            app.Image.ImageSource = 'Escudo_udec.gif';

            % Create Grupo4RafaelBurgosJocelynMatusLabel
            app.Grupo4RafaelBurgosJocelynMatusLabel = uilabel(app.UIFigure);
            app.Grupo4RafaelBurgosJocelynMatusLabel.FontSize = 18;
            app.Grupo4RafaelBurgosJocelynMatusLabel.FontWeight = 'bold';
            app.Grupo4RafaelBurgosJocelynMatusLabel.Position = [475 88 208 83];
            app.Grupo4RafaelBurgosJocelynMatusLabel.Text = {'Grupo 4: '; '- Rafael Burgos'; '- Jocelyn Ma

            % Create texto
            app.texto = uilabel(app.UIFigure);
            app.texto.Tag = 'texto';
            app.texto.HorizontalAlignment = 'center';
            app.texto.VerticalAlignment = 'top';
            app.texto.FontSize = 11;
            app.texto.FontWeight = 'bold';
            app.texto.Position = [159 124 136 32];
            app.texto.Text = '5 [kHz] --> 48 [kHz]';

            % Create textod
            app.textod = uilabel(app.UIFigure);
            app.textod.Tag = 'textod';
            app.textod.HorizontalAlignment = 'center';
            app.textod.VerticalAlignment = 'top';
            app.textod.FontSize = 11;
            app.textod.FontWeight = 'bold';
```

Figura 74.

```
            app.textod.Position = [146 11 143 32];
            app.textod.Text = '%20% --> 80%';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = app1_modificado

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Figura 75.