Proyecto convertidor DC-DC Flyback

Leví Gadiel Sojos González
Departamento de Ingeniería Eléctrica
Universidad de Concepción
Concepción, Chile
levsojos2021@udec.cl

Sofía Lorena Villegas Silva
Departamento de Ingeniería Eléctrica
Universidad de Concepción
Concepción, Chile
svillegas2022@udec.cl

Nicolás Alfredo Gajardo Saavedra
Departamento de Ingeniería Eléctrica
Universidad de Concepción
Concepción, Chile
nigajardo2021@udec.cl

***Resumen*—El presente informe detalla el proceso de diseño, cálculo e implementación de un convertidor de voltaje DC-DC aislado, basado en la topología Flyback, operando en modo CCM. El objetivo principal del proyecto es obtener una tensión de salida regulada a partir de una alimentación de entrada. La metodología de diseño se aborda mediante una estructura de tres niveles, donde posteriormente, se muestran los resultados experimentales obtenidos a partir del prototipo desarrollado en éstos. Además, se presenta un diseño final con las correcciones y mejoras realizadas. Como punto final, se introduce el proyecto 2 de monitoreo a través de un ESP32.**

***Index Terms*—PIC18, microcontroladores, aplicaciones industriales, programación de microcontroladores.**

I. Introducción

En el contexto actual de la electrónica de potencia, la demanda de fuentes de alimentación eficientes, compactas y fiables es cada vez mayor. A diferencia de las fuentes lineales tradicionales, las fuentes conmutadas (SMPS) ofrecen una densidad de potencia superior y una mayor eficiencia energética. Para el desarrollo de este proyecto, se ha seleccionado la topología **Flyback**. Esta configuración es ampliamente utilizada en aplicaciones de baja y media potencia debido a su simplicidad estructural, bajo coste de componentes y, fundamentalmente, por su capacidad de proporcionar aislamiento galvánico entre la entrada y la salida.

II. Objetivos

- **Diseñar e implementar** un convertidor DC-DC Flyback aislado para modo CCM.
- **Desarrollar** un sistema de control digital mediante PIC18F4550 para la generación de PWM y una interfaz gráfica de usuario (GUI) para la gestión de parámetros.
- **Validar** el sistema completo contrastando resultados de simulación frente a mediciones experimentales.

III. Metodología

El desarrollo del proyecto se dividió en 3 niveles. Los cuales se detallan a continuación.

A. Nivel 1: Diseño e implementación prototipo convertidor Flyback

Este nivel constituye la base física del sistema. El objetivo fue materializar la topología Flyback en un circuito impreso funcional y seguro. Lo primero a considerar son las especificaciones dadas por el usuario, ya que a partir de ellas se obtendrán los demás valores.

TABLE I
Consideraciones de diseño

Parámetro	Símbolo	Valor / Rango	Unidad
Voltaje de Entrada	V_{in}	11-14	$[V]$
Voltaje de Salida	V_{out}	40-50	$[V]$
Potencia de salida	P_{out}	10	$[W]$
Frecuencia de Conmutación	f_{sw}	60	$[kHz]$

1) Cálculos de diseño:

- Función de transferencia
  Para operación en CCM, la ecuación de diseño más importante del convertidor flyback, es la que relaciona el ciclo de trabajo con los voltaje de entrada y salida. Esta ecuación se presenta a continuación:

$$\frac{Vout}{Vin} = \frac{D}{1-D} \cdot \frac{N2}{N1} \tag{1}$$

  Esta función de transferencia nos permite conocer el comportamiento del circuito dependiendo de una razón de vueltas y un ciclo de trabajo determinados. En el gráfico que se muestra a continuación, se ve el efecto que tiene cambiar la relación de vueltas de los devanados. En este caso, se mantiene una razón de 15 vueltas para el primario, mientras que para el secundario se prueban distintos valores.

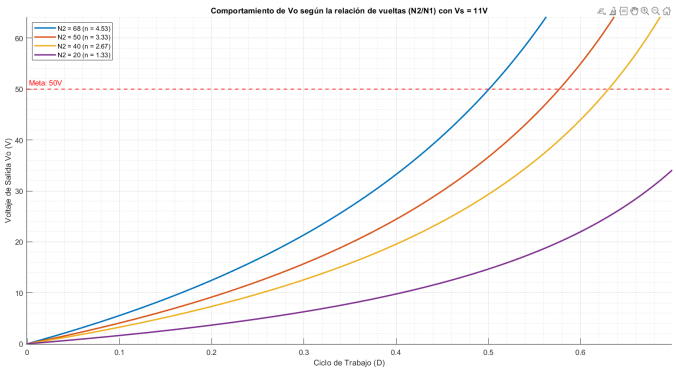


Fig. 1. Curva característica del sistema

  Se ve que el valor que permite obtener el voltaje deseado (50 V) con un menor ciclo de trabajo (50%) es N2=68.
- Corrientes

Conocer los valores de corriente tanto del devanado primario como secundario, nos permitirá elegir un calibre adecuado para los cables. Lo primero es calcular la corriente media en la rama magnetizante:

$$ILm = \frac{Vin \cdot D}{(1-D)^2 \cdot R} \cdot (\frac{N2}{N1})^2 \tag{2}$$

Luego, para calcular la corriente máxima y mínima se utiliza la expresión:

$$ILm_{max,min} = ILm \pm \frac{\Delta ILm}{2} \tag{3}$$

El valor de ΔILm se escoge como un porcentaje de la corriente media.

$$\Delta ILm = 0.45 \cdot ILm \tag{4}$$

Luego, podemos asumir que la corriente en la bobina primaria es equivalente a la corriente máxima de la rama magnetizante. A partir de ello, se tiene la expresión de la corriente en el devanado secundario:

$$I2 = ILm_{max} \cdot \frac{N2}{N1} \tag{5}$$

- Inductancias
  Al igual que con la corriente, se puede asumir que la indutancia del primario es equivalente a la de magnetización. Entonces, las expresiones de inductancia para primario y secundario son:

$$Lm = Lp = \frac{Vout \cdot (1-D)}{\Delta ILm \cdot f} \cdot \frac{N1}{N2} \tag{6}$$

$$Ls = Lm \cdot (\frac{N2}{N1})^2 \tag{7}$$

- Voltajes de rizado y capacitancia
  La fórmula que relaciona el voltaje de rizado con la capacitancia es la siguiente:

$$\frac{\Delta Vo}{Vo} = \frac{D}{RCf} \tag{8}$$

- Voltajes MOSFET y diodo rectificador
  Para la selección de estos componentes, lo más importante es calcular el voltaje que deben soportar. En el caso del MOSFET, se tiene que la ecuación para su voltaje drain-source, está dada por:

$$VDS_{max} = Vin + \frac{Ns}{Np}(Vout + VD) \tag{9}$$

Para el diodo rectificador, se tiene una ecuación similar:

$$VD_{max} = Vout + \frac{Ns}{Np}Vin \tag{10}$$

- Red snubber
  Para proteger el interruptor MOSFET de los picos de voltaje generados por la inductancia de dispersión (L_{lk}) durante el apagado, se diseñó una red de amortiguamiento (snubber) tipo RCD. Se estimó que la energía almacenada en L_{lk} debe ser disipada completamente por la resistencia del snubber en cada ciclo de conmutación.La potencia que debe disipar la resistencia R_{sn} se determina a partir de la energía de la inductancia de fugas y la frecuencia de operación:

$$P_{Rsn} = \frac{1}{2} \cdot L_{lk} \cdot I_{pk}^2 \cdot f_{sw} \tag{11}$$

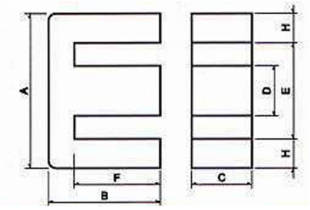
Donde I_{pk} es la corriente pico del primario y L_{lk} se estimó como un porcentaje de la inductancia magnetizante total. A partir de esta potencia y definiendo un voltaje de fijación (V_{clamp}) que asegure que el voltaje total (V_{DS}) no supere el límite del MOSFET, se calcula la resistencia:

$$R_{sn} = \frac{V_{clamp}^2}{P_{Rsn}} \tag{12}$$

Finalmente, el capacitor C_{sn} se dimensiona para mantener el voltaje de fijación constante, minimizando el rizado de voltaje (ΔV_{clamp}) durante la conmutación:

$$C_{sn} = \frac{V_{clamp}}{\Delta V_{clamp} \cdot R_{sn} \cdot f_{sw}} \tag{13}$$

- Transformador
  Se decidió utilizar un núcleo de ferrita como requisito para operar adecuadamente en la frecuencia de 60 [khz]. Para el cual se tienen las siguientes dimensiones:



A	B	C	D	E	F	H
3.4 cm	1.4 cm	0.9 cm	0.9 cm	2.5 cm	1.0 cm	0.4 cm

Fig. 2.

Dado que se dispone de un núcleo con área efectiva $A_c = 81mm^2$, el diseño se centró en determinar el número de vueltas y el entrehierro (air gap) necesarios para obtener la inductancia de magnetización requerida ($L_m \approx 126\mu H$) y que el transformador opere como un inductor acoplado, es decir, tiene una fase de almacenamiento y una fase de liberación de energía.
La longitud del entrehierro se calcula como:

$$lg = \frac{\mu_o \cdot \mu_r Np^2 \cdot Ac \cdot 10^{-3}}{Lp} \tag{14}$$

Donde μ_o y μ_r representan la permeabilidad del vacío y permeabilidad relativa, respectivamente.
Para garantizar el factor de relleno ($K_f < 1$) y facilitar el bobinado manual, se ajustó el número de vueltas a

$N_p = 15$, compensando la inductancia mediante un ajuste en el gap. La selección de los conductores se basó en las corrientes máximas a soportar por primario (ILm_{max}) y secundario I2. Los parámetros constructivos finales se resumen en la Tabla con los valores finales de diseño:

TABLE II
PARÁMETROS FINALES DE DISEÑO Y COMPONENTES

Símbolo	Parámetro	Valor
f_{sw}	Frecuencia de conmutación	60 kHz
V_{in}	Voltaje de entrada	11 V
V_{out}	Voltaje de salida	50 V
D	Ciclo de trabajo	50%
A_c	Área efectiva del núcleo	$81mm^2$
N_p	Vueltas del primario	15
N_s	Vueltas del secundario	68
n	Relación de vueltas	0.22
L_m	Inductancia magnetizante (L_p)	$126\mu H$
L_s	Inductancia secundaria	$2.31mH$
ΔILm	Inductancia de rizado	$0.814mH$
l_g	Longitud del entrehierro	$0.18mm$
$I_{L,max}$	Corriente máx. primario (Bobina)	$2.22A$
I_2	Corriente secundario	$0.49A$
C_{out}	Capacitancia de salida	1.67 μF
L_{lk}	Inductancia de fugas (Est.)	$2.54\mu H$
I_{pk}	Corriente pico (Para diseño Snubber)	$2.27A$
V_{clamp}	Voltaje de fijación (ΔV)	$23V$
P_{Rsn}	Potencia disipada calc.	$0.39W$
R_{sn}	Resistencia Snubber	$1.3k\Omega$
C_{sn}	Capacitor Snubber	$5.4nF$
$V_{DS,max}$	Voltaje máx. MOSFET	240 V
$V_{D,max}$	Voltaje máx. Diodo	150 V

2) Simulación: Para corroborar los valores de diseño calculados, se realizó una simulación en el software LTSpice, el cual funciona mediante el motor SPICE, especial para electrónica de potencia y convertidores DC-DC.

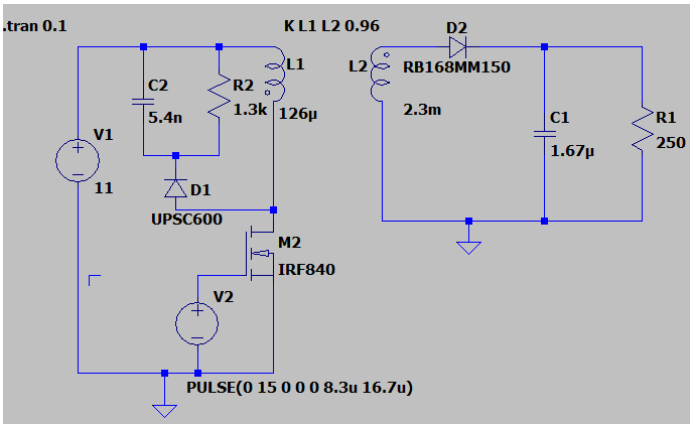


Fig. 3. Circuito LTSpice

Se verificó que el VDS del MOSFET estuviera de los rangos establecidos, es decir, que el circuito de snubber cumpliera su función:

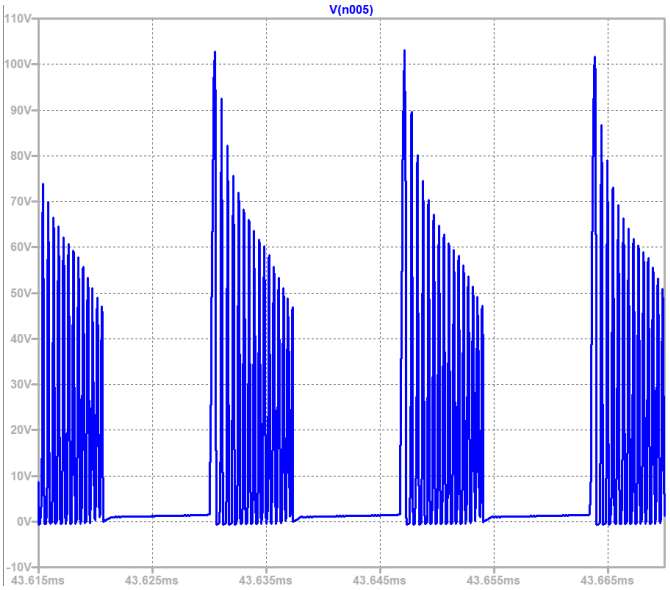


Fig. 4. Voltaje VDS

Se graficó las corrientes en devanado primario y secundario, para de esa manera validar el diseño del transformador y verificar que se está trabajando en CCM.

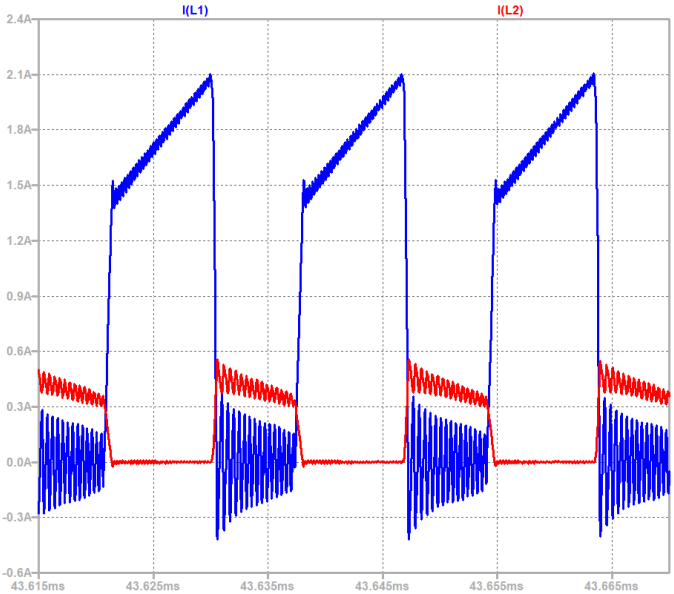


Fig. 5. Corrientes en el transformador

Se analizó la forma de onda del voltaje de salida, la cual corresponde a una señal triangular. Durante la fase en que el interruptor primario está encendido (ON), el diodo de salida bloquea el paso y el capacitor se ve obligado a descargarse linealmente para alimentar la carga, creando una

rampa descendente. Cuando el interruptor se apaga (OFF), la energía magnética del transformador se libera y recarga el capacitor rápidamente, creando la rampa ascendente. Este ciclo constante de carga y descarga dibuja la señal de diente de sierra o triángulo sobre el voltaje DC.

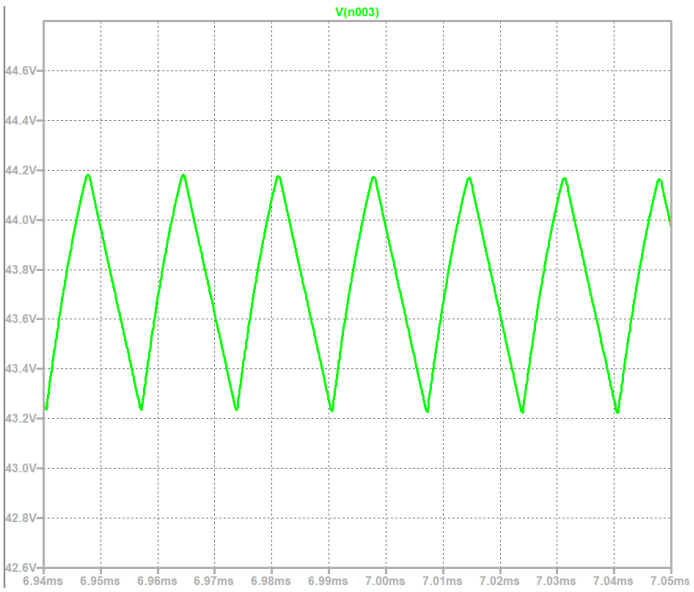


Fig. 6. Voltaje de salida

3) Selección de componentes: A partir de los cálculos teóricos y los resultados obtenidos en simulación, se realizó una selección de componentes, la cual se detalla en la tabla al final de esta subsección, considerando el modelo, valor y costo de cada componente y/o material.

- MOSFET
  Uno de los componentes más importantes a seleccionar es el MOSFET, que es el elemento conmutador, el cual nos permitirá almacenar y descargar la energía dentro del circuito. Para su selección, se debe considerar el voltaje que este debe soportar, y que además, debe ser capaz de trabajar a altas frecuencias. El MOSFET seleccionado corresponde al IRFP250, el cual posee una alta velocidad de conmutación.
- Driver
  En este caso, se seleccionó el driver HCPL-3120 ya que proporciona la corriente de pico necesaria (2.5A) para cargar y descargar rápidamente la capacitancia de compuerta (Gate) del MOSFET, minimizando las pérdidas por conmutación. Adicionalmente, al ser optoacoplado, provee aislamiento galvánico, protegiendo al circuito de control (bajo voltaje) ante fallos en la etapa de potencia (alto voltaje).
- Diodo de salida
  El diodo de salida se seleccionó de acuerdo a la simulación y cálculos realizados, donde lo importante es el voltaje a soportar. Además, debe ser un diodo rápido, ya que de esta manera se tiene un menor tiempo de recuperación inversa t_rr, lo cual permite trabajar adecuadamente a la frecuencia de 60 Khz. El diodo seleccionado es el MUR1060CT.
- Diodo de snubber
  Al igual que para el diodo de salida, debe ser capaz de soportar un voltaje específico, que en la simulación vemos que llega hasta 180 V. Además, también se seleccionó un diodo rápido, en este caso, el U810.

Algunos de los valores de los componentes difieren de los calculados teóricamente debido a disponibilidad y pruebas experimentales. A continuación se muestra la lista de componentes final:

Componente	**Valor / Detalle**	**Modelo/Tipo**	**Costo**
Driver Opto.	$I_{pk} = 2.5A$	HCPL-3120	$0
MOSFET N-Ch	$V_{GS} = 20V, I_D = 33A$	IRFP250	$2000
Diodo Rápido	$600V, 10A$	MUR1060CT	$2000
Diodo Rápido	$600V, 8A$	U810	$2000
Núcleo Transf.	$A_c = 81mm^2$	Ferrita (Ej. E32)	-
Bobina Primaria	15 Vueltas	**Alambre AWG 18**	-
Bobina Secundaria	68 Vueltas	**Alambre AWG 23**	-
Cap. Snubber	$4.7nF$	Cerámico	$200
Cap. Driver	$100nF$	Cerámico	$200
Cap. Salida	$2.2\mu F$	Electrolítico	$200
2 Cap. Salida	$1.1\mu F$	Electrolítico	$200
R. Carga	$250\Omega, 10W$	Reóstato	$0
R. Snubber	470Ω	Cerámica	$100
R. Input (LED)	270Ω	-	$100
R. Gate (R_g)	6Ω	-	$100
Borneras	2 pines, 5.08mm	KF301-2P	$500
Varios	*(Cables, Disipador, Flux, Estaño)*		$8900
Total Estimado			**$17000**

Fig. 7. Lista de componentes

4) Diseño PCB: El software seleccionado para realizar el diseño corresponde a KiCad. Lo primero a realizar es el esquemático, en el cual se hacen todas las conexiones del circuito, especificando adecuadamente las tierras. Este último punto es muy importante, ya que por ejemplo, la etapa de control y de potencia deben estar aisladas la una de la otra. Cada componente dentro del esquemático posee una "huella", la cual indica el modelo y dimensiones.

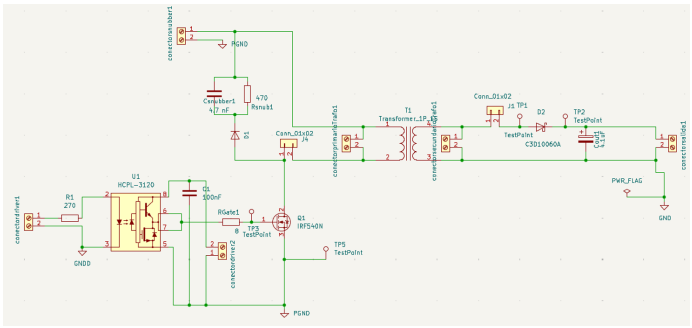


Fig. 8. Esquemático del circuito

A continuación, se presenta el diseño de la PCB, donde las conexiones entre los componentes se realizon mediante pistas de cobre. También se trabajó con "islas", las cuales nos permiten tener una tierra para potencia, control y salida, además de que aseguran una mayor disipación térmica y manejo de altas corrientes.

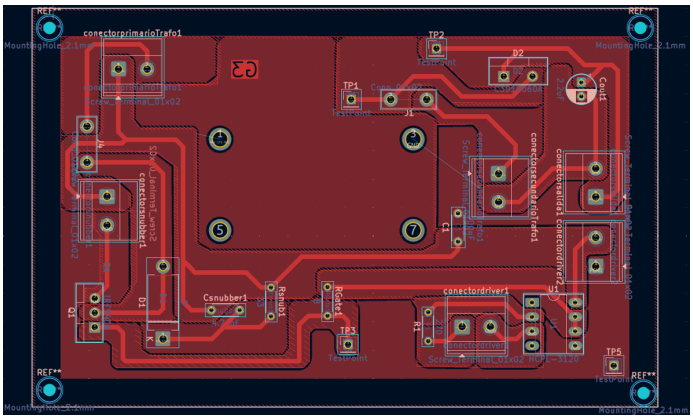


Fig. 9. Vista PCB

Para la huella del transformador, se tomó como referencia el modelo 3D de la base. Con esto, se determinó el diámetro de los orificios, así como la separación entre éstos. A continuación se muestra la vista 3D del circuito completo:

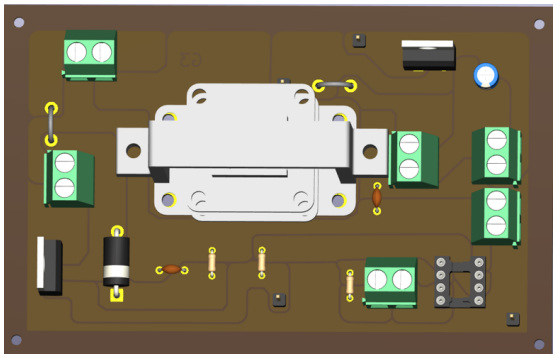


Fig. 10. Vista modelo 3D

Además de los componentes necesarios para el funcionamiento del circuito, se añadieron orificios para luego añadir los puntos de medición tanto de voltaje como de corriente. Específicamente, se colocaron dos puntos de medición de corriente, para primario y secundario, y dos puntos de medición de voltaje, para el VGS del MOSFET y para el voltaje en el diodo de salida.

B. Nivel 2: Generación de pulso PWM utilizando un microcontrolador

Tras la validación del hardware de potencia, se implementó la etapa de control digital utilizando el microcontrolador PIC18F4550 para generar la señal de Modulación por Ancho de Pulso (PWM). Esta señal gobierna los tiempos de conmutación (conducción y corte) del interruptor de potencia, permitiendo la regulación precisa del voltaje y la potencia de salida, cumpliendo así con el objetivo principal del diseño.

1) Características del microcontrolador: El PIC18F4550 cuenta con una arquitectura Harvard de 8 bits, lo que permite el acceso simultáneo a la memoria de programa y datos a través de buses separados, optimizando la velocidad de ejecución de instrucciones. Para aplicaciones de electrónica de potencia, destaca su módulo CCP (Capture/Compare/PWM), capaz de generar señales PWM con una resolución de hasta 10 bits.Adicionalmente, el dispositivo integra un oscilador interno con PLL (Phase Locked Loop), permitiendo elevar la frecuencia de trabajo hasta 48 MHz. Esta alta velocidad de reloj es fundamental para lograr una resolución temporal fina en una señal PWM de 60 kHz, facilitando ajustes precisos en el ciclo de trabajo (D) y asegurando la estabilidad del voltaje de salida.

2) Configuración de periféricos: El funcionamiento del sistema se basa en la coordinación de múltiples módulos de hardware integrados (periféricos). Los módulos configurados para este proyecto son: el convertidor Analógico-Digital (ADC) de 10 bits, el Timer2 y el módulo CCP. Asimismo, se configuró el sistema de oscilación para establecer la base de tiempo necesaria. El objetivo del control es variar el ciclo de trabajo de la onda cuadrada en función de una referencia externa. Esta referencia es un voltaje analógico regulado por un potenciómetro, el cual determina proporcionalmente la duración del pulso en alto. La relación matemática que define el ciclo de trabajo digital se expresa como:

$$D_{PWM} = \frac{ADC_{val} \cdot PWM_{max}}{2^n - 1} \tag{15}$$

Donde ADC_{val} es la lectura digital del potenciómetro, PWM_{max} es el valor máximo del periodo configurado y n es la resolución del ADC (10 bits, por lo que el denominador es 1023). A continuación, se detalla la función de cada periférico:

- ADC (Convertidor Analógico-Digital)
  Su función es discretizar la señal de voltaje analógica proveniente del potenciómetro para que la CPU pueda procesarla. Este módulo utiliza aproximaciones sucesivas (SAR) para convertir el nivel de tensión en un valor

digital de 10 bits (0 a 1023). A continuación, se muestran las líneas de código más relevantes que definen el funcionamiento de este módulo.

```
// 1. Configuracion (Setup)
ADCON1bits.PCFG = 0b1110; // AN0 Analogico
TRISAbits.TRISA0 = 1;// RA0 Entrada
ADCON2bits.ADCS = 0b110; // Reloj Fosc/64
ADCON2bits.ADFM = 1;// Justif. Derecha

// 2. Lectura (Bucle)
ADCON0bits.GO = 1;// Iniciar conv.
while(ADCON0bits.GO);// Esperar fin
// Unir los 10 bits (MSB + LSB)
adc_valor = (ADRESH<<8)|ADRESL;
```

Listing 1. Configuración del ADC

- Oscilador y PLL
  Proporcionan la señal de reloj del sistema. Se seleccionó el modo de oscilador primario utilizando un cristal externo de 20 MHz. El PLL toma esta frecuencia y la multiplica para entregar una frecuencia de trabajo de 48 MHz (F_{osc}), necesaria para la velocidad de procesamiento requerida. A continuación, se muestran las líneas de código más relevantes para estos módulos.

```
// --- BITS DE CONFIGURACION (FUSES)
#pragma config FOSC = HSPLL_HS// Osc. HS + PLL
#pragma config PLLDIV = 5 // Entrada 20MHz/5
#pragma config CPUDIV = OSC1_PLL2 // SysClk 48MHz
#pragma config USBDIV = 2 // Clock USB
```

Listing 2. Configuración del Oscilador y PLL

- Timer2
  Este temporizador es esencial para la generación de PWM, ya que establece la base de tiempo que define la frecuencia de la señal de salida. Trabaja en conjunto con el registro PR2 para determinar el periodo de la onda.

```
// Configurar Frecuencia PWM (~60kHz)
// Fosc=48MHz, Prescaler=1
PR2 = 197; // Periodo
T2CONbits.T2CKPS = 0; // Prescaler 1:1
T2CONbits.TMR2ON = 1; // Encender Timer2
```

Listing 3. Configuración del Timer2 (Frecuencia)

- Módulo CCP (Modo PWM)
  Configurado en modo PWM, este bloque compara constantemente el valor del Timer2 con el valor del ciclo de trabajo deseado (registros CCPR1L y bits de control). Cuando coinciden, el hardware conmuta automáticamente el estado del pin de salida, generando el pulso sin intervención continua de la CPU.

```
// 1. Configuracion (Setup)
TRISCbits.TRISC2 = 0;// Pin RC2 Salida
CCP1CONbits.CCP1M = 0xC; // Modo PWM (0b1100)

// 2. Control del Ciclo de Trabajo (Bucle)
// Se cargan los 10 bits en dos registros:
CCPR1L = (duty_pwm >> 2);// 8 bits altos
CCP1CONbits.DC1B = (duty_pwm & 0x03); // 2 bits bajos
```

Listing 4. Configuración y control del módulo CCP

3) Simulación: Para validar el correcto funcionamiento del circuito, se realizó una simulación donde se verifica que el código realizado en MPLAB genera adecuadamente la señal PWM que va al driver y luego al MOSFET del convertidor.

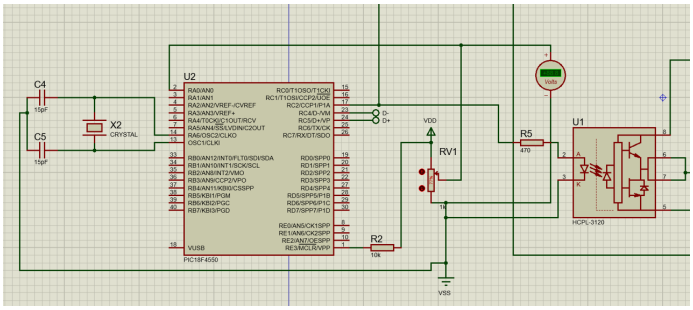


Fig. 11. Simulación nivel 2

De la imagen se puede observar que los pines utilizados son los mismos que se configuraron dentro del código. Para la salida de las señal PWM se utiliza el pin RC2, mientras que para la entrada del voltaje, el cual está dado por el potenciómetro, se utiliza el pin AN0. Se conecta además, una resistencia de 10k para así evitar que el microcontrolador se reinicie. También, para limitar la corriente, se incluye una resistencia de 470 Ω. Para la etapa 3, se utilizán también los pines RC6 y RC7 que permitirán la comunicación vía TX/RX (transmisor/receptor).

C. Nivel 3: Interfaz Gráfica de Usuario

Para dotar al sistema de capacidad de supervisión y control, se desarrolló una aplicación de software en un ordenador, el cual permite la interacción humano-máquina (HMI). Esta aplicación fue desarrollada utilizando el lenguaje **Python**, aprovechando las librerías *PySide6* para la interfaz gráfica y *PyQtGraph* para la visualización de señales en tiempo real.

1) Descripción de la Interfaz y Componentes: La interfaz gráfica ha sido diseñada con un enfoque modular, dividiendo la ventana principal en dos áreas funcionales: visualización (izquierda) y control (derecha). Los componentes principales son:

- **Controles Deslizantes (Sliders):** Son los elementos primarios de interacción. Permiten al usuario ajustar los parámetros críticos del convertidor de forma intuitiva:
  - *Frecuencia:* Rango ajustable de 50 kHz a 70 kHz.
  - *Ciclo de Trabajo (Duty Cycle):* Rango ajustable de 0% a 90%.

  Estos controles no envían datos crudos continuamente; implementan una lógica de "limpieza de eventos" para transmitir cambios solo cuando son significativos y estables, evitando saturar el canal de comunicación.
- **Monitores Gráficos:** Se utilizan dos paneles gráficos de alta velocidad (*PyQtGraph*) que simulan un osciloscopio digital:
  - *Monitor PWM:* Visualiza la forma de onda de control generada, permitiendo verificar visualmente el ancho de pulso antes de aplicarlo.
  - *Monitor de Voltaje:* Grafica la tendencia del voltaje de salida en el tiempo. Si el sensor Wi-Fi está conectado, muestra datos reales; de lo contrario, muestra una predicción basada en el modelo matemático.

- **Panel de Conexión Híbrida:** Permite gestionar independientemente el puerto de control (USB-Serial hacia el PIC) y el puerto de telemetría (Socket Wi-Fi hacia el ESP32), facilitando pruebas modulares.
- **Indicador Numérico:** Un display digital de gran tamaño que muestra el voltaje de salida actual (medido o estimado) para una lectura rápida por parte del operador.

2) Protocolos de comunicación: El microcontrolador PIC18F4550 cuenta con módulos de comunicación avanzados, incluyendo un puerto USB 2.0 de alta velocidad y un módulo EUSART (*Enhanced Universal Synchronous Asynchronous Receiver Transmitter*). Si bien la interfaz USB permite tasas de transferencia superiores, para este proyecto se optó por utilizar el módulo **EUSART** configurado en modo asíncrono (**UART**). Esta decisión se fundamenta en la simplicidad de la implementación del firmware, la robustez del protocolo y la facilidad de integración con el PC mediante convertidores USB-TTL estándar, evitando la complejidad de desarrollar una pila USB completa.

La comunicación se estableció a una velocidad de **9600 baudios**, con una configuración de trama de 8 bits de datos, sin paridad y 1 bit de parada (8N1).

Para el intercambio de datos, se diseñó un protocolo de capa de aplicación basado en caracteres ASCII, lo que facilita la depuración y asegura la integridad de los comandos. El protocolo define dos tipos de tramas de control enviadas desde la interfaz hacia el PIC:

- **Control de Frecuencia:** Trama iniciada por el carácter 'F', seguida del valor numérico en Hertz (ej. `F60000.0`).
- **Control de Ciclo de Trabajo:** Trama iniciada por el carácter 'D', seguida del porcentaje deseado (ej. `D50.5`).

IV. Resultados Experimentales

Para la validación de los tres niveles, es necesario el uso de algunos instrumentos y materiales que se solicitaron en pañol, estos se detallan a continuación:

Cant.	Instrumento	Función
2	Fuentes DC	Alimentación
1	Osciloscopio	Ver V_{DS}, V_{GS}, rizado
1	Gen. Ondas	PWM de control
1	Multímetro	Voltaje y continuidad
1	Inductómetro	Caracterización (L_m)
1	Pinza Amp.	Corrientes devanados
1	Reóstato	Carga variable
	Cables y Conexión	
Var.	Caimán-Caimán	Potencia
Var.	Caimán-Banana	Fuentes
Var.	Banana-Banana	Instrumentación
Var.	Jumpers	Protoboard

TABLE III
Lista de Materiales

Para la validación final del prototipo, se realizaron tablas comparativas entre lo calculado, simulado y práctico. Se tiene una tabla para el caso de voltaje mínimo de entrada y mínimo de salida, además de otra para el caso de voltaje máximo de entrada y máximo de salida.

Para comprobar el diseño realizado en el nivel 1, se observaron las formas de onda mediante el osciloscopio, donde se capturó la forma de onda de las corrientes en los devanados, voltaje de salida y voltaje VGS del MOSFET.



Fig. 12. Voltaje de salida caso 11-40V



Fig. 13. Voltaje de salida caso 14-50V

De las imágenes se puede observar que tenemos una forma de onda triangular, lo cual encaja con las simulaciones realizadas, ya que, el voltaje de salida es el resultado de tomar un voltaje alterno y rectificarlo mediante un diodo y un capacitor, produciendo así una onda que pretende ser continua, pero tiene una cierta variación o "rizado" entre un mínimo y un máximo.

Para el caso de las corrientes, su importancia radica en que nos permitirán saber si realmente estamos trabajando en CCM.



Fig. 14. Corriente I1 caso 11-40V

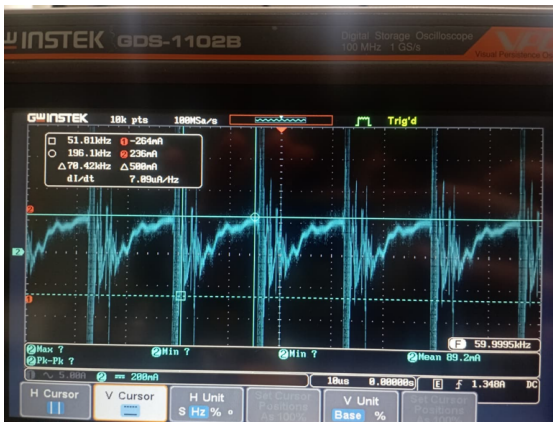


Fig. 15. Corriente I2 caso 11-40V



Fig. 16. Corriente I2 caso 11-40V

Caso 1: 14V-50V			
Mediciones	**Valor teórico**	**Valor simulado**	**Valor práctico**
Duty cycle	50%	50%	75%
Voltaje de entrada	14 V	14 V	14 V
Rizado voltaje de salida	1 V	490 mV	640 mV
Voltaje min de salida	49 V	54.89 V	49.66 V
Voltaje máx de salida	51 V	55.38 V	50.94 V
Voltaje promedio salida	50 V	55.14 V	50.3 V
Corriente de entrada	0.909 V	1.1 A A	1.2 A
Corriente de salida	0.2 V	0.220 A	0.2012 A
Rizado corriente bobina primaria	0.814 A	1.128 A	2.04 A
Corriente min bobina primaria	1.409 A	1.496 A	679 mA
Corriente máx bobina primaria	2.227 A	2.624 A	2.72 A
Corriente máx secundario	491 mA	755 mA	460 mA
Capacitancia de salida	2.2 uF	4.2 uF	4.2 uF
Vueltas primario	26.948	x	15
Vueltas secundario	122.12	x	68
Relación de vueltas	0.22	0.22	0.22
Inductancia primario	126 uF	100 uH	100 uH
Inductancia secundario	2.321 mF	1.509 mH	1.509 mH
Voltaje VGS MOSFET	18 V	18 V	18.2 V
Corriente gate MOSFET	2.143 A	2.082 A	2.17 A
Resistencia de carga	250 ohm	250 ohm	250.6 ohm
Potencia entrada	10	15.4 W	16.8 W
Potencia salida	10	12.23 W	10.12 W
Eficiencia	100%	79.4%	60.2%

TABLE IV
COMPARACIÓN DE VALORES PARA EL RANGO 14V-50V

Caso 1: 11V-40V			
Mediciones	**Valor teórico**	**Valor simulado**	**Valor práctico**
Duty cycle	50%	50%	75%
Voltaje de entrada	11 V	11 V	11 V
Rizado voltaje de salida	1 V	380 mV	580 mV
Voltaje min de salida	39 V	43.02 V	38.662 V
Voltaje máx de salida	41 V	43.40 V	39.838 V
Voltaje promedio salida	40 V	43.21 V	39.25 V
Corriente de entrada	0.909 A	0.99 A	1.11 A
Corriente de salida	0.25 A	0.174 A	0.157 A
Rizado corriente bobina primaria	0.814 A	0.9 A	1.48 A
Corriente min bobina primaria	1.409 A	1.2 A	740 mA
Corriente máx bobina primaria	2.227 A	2.1 A	2.22 A
Corriente máx secundario	765 mA	600 mA	500 mA
Capacitancia de salida	2.2 uF	4.2 uF	4.2 uF
Vueltas primario	26.948	x	15
Vueltas secundario	122.12	x	68
Relación de vueltas	0.22	0.22	0.22
Inductancia primario	126 uF	100 uH	100 uH
Inductancia secundario	2.321 mF	1.509 mH	1.509 mH
Voltaje VGS MOSFET	18 V	15 V	18.2 V
Corriente gate MOSFET	2.143 A	2.082 A	2.17 A
Resistencia de carga	250 ohm	250 ohm	250.6 ohm
Potencia entrada	10	10,89 W	12.21 W
Potencia salida	10	7.52 W	6.16 W
Eficiencia	100%	69%	50%

TABLE V
COMPARACIÓN DE VALORES PARA EL RANGO 11V-40V

De las imágenes para el primer caso, se puede observar que, aunque el diseño se realizó para operar en CCM, las formas de onda muestran que se está operando en DCM. En un funcionamiento ideal en CCM, la corriente debería conmutar inmediatamente entre la bajada y la siguiente subida, sin tiempos muertos. Sin embargo, en las capturas experimentales se identifica claramente un intervalo de corriente nula (t_{dead}) al finalizar la descarga de la bobina. Durante este intervalo, la corriente permanece en cero y se observan oscilaciones amortiguadas. En síntesis, se tiene que en CCM la energía en el núcleo nunca llega completamente a 0, mientras que en DCM sí. La razón por la que se tiene esta discrepancia, puede deberse a que la inductancia práctica es menor a la teórica.

Como último punto, se muestra a continuación el voltaje VGS en el MOSFET, el cual coincide con el voltaje que permite la conmutación,en el rango recomendado por datasheet y que corresponde a 18 V. Cabe destacar, que para esta etapa, la generación de la señal PWM se realizó mediante un generador de funciones, en el cual podemos variar tanto la frecuencia como el ciclo de trabajo.

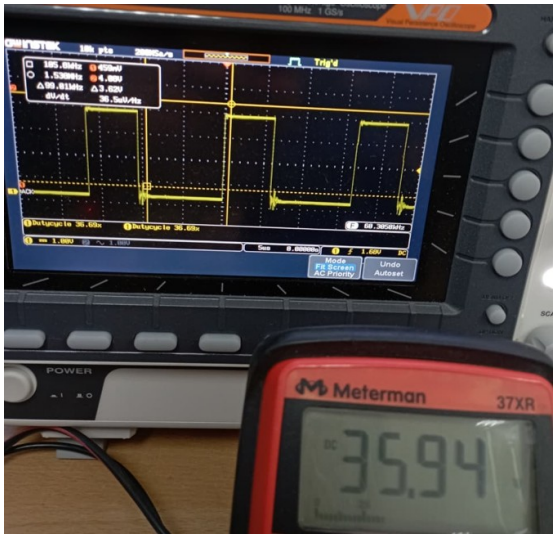


Fig. 18. Señal PWM mediante osciloscopio

Para la validación de los niveles 2 y 3, se observaron las formas de onda mediante el osciloscopio. Esta vez, la señal es generada por el microcontrolador y los valores de frecuencia y duty se modifican a través de la interfaz. Para el caso del nivel 2, se verificó que el microcontrolador entregase el valor de la señal PWM de amplitud 5V, la cual, al pasar por el driver, se convierte en una señal de 18 V, que sería el voltaje VGS para conmutar el MOSFET.



Fig. 17. Interfaz gráfica

La interfaz gráfica cuenta con diferentes elementos que permiten al usuario interactuar con el circuito. Se muestran las gráficas de la señal PWM y del voltaje de salida estimado. Los sliders permiten variar frecuencia y ciclo de trabajo y además se muestra un recuadro con el valor de voltaje promedio a la salida.

En las imágenes anteriores se puede verificar tanto el nivel 2 y 3, ya que tanto el microcontrolador como la interfaz logran comunicarse de tal forma que se puede controlar los parámetros de frecuencia y ciclo de trabajo. En la interfaz se asignó una frecuencia de 60 Khz y un duty de 37% aproximadamente. Esto se ve reflejado en el osciloscopio, donde también se muestra una frecuencia de aproximadamente 60 Khz y en el multímetro, donde se obtiene un voltaje de aproximadamente 36 V.

V. Diseño final

A. Estudio de componentes

Para el diseño final se buscó reducir el espacio utilizado, es decir, que la PCB sea lo más pequeña posible sin que el circuito pierda su eficiencia. Para lograr esto, se consideró la mayoría de los componentes del tipo SMD.

Los componentes SMD (Surface Mounting Device) son dispositivos electrónicos diminutos que se sueldan directamente sobre la superficie de una placa de circuito impreso (PCB) en lugar de atravesar agujeros, lo que permite circuitos más compactos y eficientes.

Además de el tipo de componente, debemos tener en cuenta que en el prototipo, utilizamos tres fuentes de voltaje, una para la alimentación del convertidor, otra para alimentación exclusiva del driver, y una para el microncontrolador. Para evitar el uso de tantas fuentes, se diseñó un circuito con reguladores de tensión.

B. Selección de componentes

A continuación se muestra una tabla resumen con los componentes seleccionados, la cual se detalla más en el anexo.

TABLE VI
Resumen de Componentes

Componente	Identificador	Cant.
Transformador	WE-FB_EP13	1
Mosfet	Q1	1
Driver	U3	1
Microcontrolador	U1	1
Crystal	Crystal1	1
Entrada USB	J1	1
Borneras	J2, J3	2
Diodo Salida	D2	1
Diodo Snubber	D1	1
Resistencia Snubber	RSnubber	1
Capacitor Snubber	CSnubber	1
Capacitor Salida 1	Cs11, Cs12	2
Capacitor Salida 2	Cs21, Cs22	2
Regulador Voltaje	U2	1
R Gate Mosfet	RGateMosfet1	1
Cap. Driver/Reg	Cdriver1/3, Creg2	3
Capacitor 2 Driver	Cdriver2	1
Capacitor 4 Driver	Cdriver4	1
Capacitor 1 Reg.	Creg1	1
Capacitor 3 Reg.	Creg3	1
Capacitor Crystal	Ccrystal1/2	2
Capacitor USB	Cusb	1
Cap. Entrada Driver	CinDriver	1
Inductor	L2	1
Resistencia 1 Reg.	R1	1
Resistencia 2 Reg.	R2	1
Rin Driver	Rin1	1

C. Diseño PCB

Para el diseño del circuito, se consideraron dos partes escenciales, la etapa de diseño y la etapa de control. A

continuación se detallan los elementos más importantes de cada una:

- **U1 (PIC18F45K50):** Es el cerebro del sistema. Genera la señal PWM para el MOSFET y gestiona la comunicación USB.
- **J1 (USB) & Cusb:** J1 es el puerto de datos/energía. Cusb (0.47 μF) es crítico para estabilizar el regulador interno de 3.3V; sin él, el USB no funciona.
- **Oscilador (Crystal1):** Reloj externo de alta precisión. Es obligatorio para cumplir con los tiempos estrictos del protocolo USB.
- **ICSP1:** Puerto de programación. La marca 'X' indica que no se montará (DNP) en la producción final.
- **U2 (TPS560200):** Regulador Buck auxiliar. Reduce el alto voltaje de entrada a un nivel seguro y estable para alimentar el driver.
- **U3 (UCC5350MCD):** Driver de Gate aislado. Amplifica la señal del microcontrolador para activar el MOSFET y protege la lógica de control.
- **Q1 (MOSFET de Potencia):** Interruptor principal. Carga energía en el transformador al cerrarse y la libera al abrirse.
- **Transformador (WE-FB_EP13):** Almacena la energía magnética y provee aislamiento galvánico (seguridad) entre entrada y salida.
- **Red Snubber (RCD):** Circuito de protección que absorbe los picos de voltaje destructivos generados por la inductancia de fuga.
- **Salida (D2 y Condensadores):** D2 rectifica la corriente y el banco de condensadores filtra el rizado (ripple) para entregar un voltaje limpio en J3.

A continuación, se muestra el esquemático realizado, es decir, las conexiones entre cada uno de los componentes:

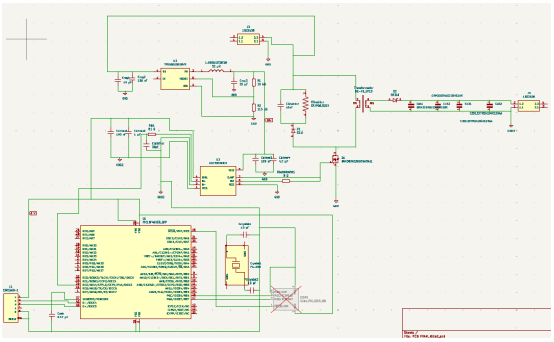


Fig. 19. Esquemático

Luego de realizar el esquemático, se organizaron los componentes de la PCB. El diseño de la tarjeta de circuito impreso se ha desarrollado bajo los siguientes parámetros técnicos:

- **Tecnología de Fabricación:** PCB rígida de doble capa (*Top* y *Bottom*) sobre sustrato FR-4 estándar de 1.6 mm de espesor.
- **Estrategia de Enrutamiento:** Se empleó un trazado con ángulos de 45° para minimizar reflexiones de señal y evitar trampas de ácido. El diseño prioriza la separación física entre la etapa lógica (microcontrolador) y la etapa de potencia.
- **Dimensionamiento de Pistas:** Se utilizaron anchos de pista diferenciados según la corriente de operación:
  - *Señales de control:* Ancho nominal de ≈ 0.25 mm para garantizar la densidad de interconexión.
  - *Líneas de potencia:* Anchos superiores a 0.8 mm en los lazos de corriente del driver y MOSFET para reducir la impedancia y caída de tensión.
- **Integridad de Señal:** Incorporación de condensadores de desacoplo en proximidad inmediata a los pines de alimentación de los circuitos integrados y segregación de planos de masa (*GND*) para el aislamiento de ruido entre dominios de voltaje.
- **Montaje:** Predominancia de tecnología de montaje superficial (SMD) para componentes activos y pasivos, reservando la tecnología THT únicamente para conectores mecánicos robustos.

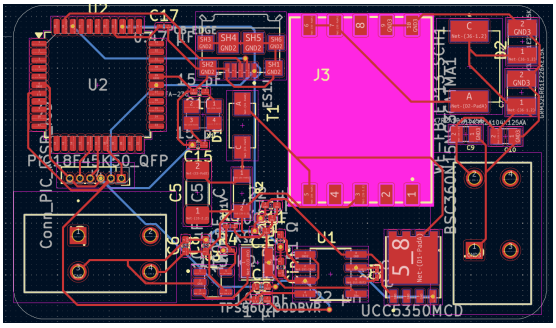


Fig. 20. Diseño PCB

Por último, se muestra el modelo 3D del diseño, el cual posee unas dimensiones de 5x3 cm aproximadamente

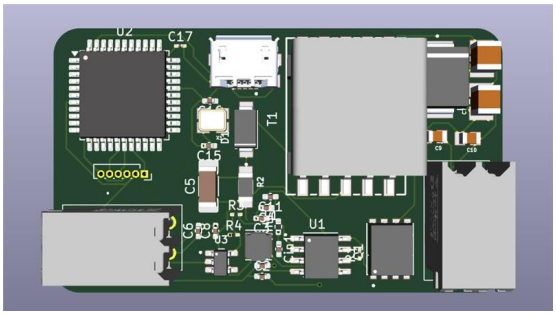


Fig. 21. Modelo 3D diseño final

VI. Proyecto 2

Como evolución del proyecto 1, se implementó una etapa de monitoreo remoto para la adquisición de datos en tiempo real del convertidor Flyback. El objetivo de este subsistema es medir el voltaje de salida real, transmitirlo de forma inalámbrica y visualizarlo en la interfaz de usuario existente.

A. Arquitectura del Sistema de Sensado

El sistema de telemetría se basa en una arquitectura cliente-servidor sobre el protocolo TCP/IP, utilizando el microcontrolador **ESP32** como nodo sensor inteligente.

- Nodo Sensor (ESP32): Actúa como un servidor TCP que lee continuamente el voltaje analógico, lo procesa digitalmente y espera conexiones entrantes.
- **Cliente de Datos (PC):** La interfaz Python, mediante la librería `socket`, se conecta a la dirección IP del ESP32 a través de la red Wi-Fi local para recibir el flujo de datos.

B. Acondicionamiento de Señal (Hardware)

Dado que el voltaje de salida del convertidor Flyback puede alcanzar hasta 60V, y el convertidor analógico-digital (ADC) del ESP32 tiene un rango máximo de 3.3V, fue necesario diseñar un circuito de acondicionamiento de señal para proteger el microcontrolador y escalar la medición.

Se implementó un **divisor de tensión resistivo** conectado en paralelo a la carga (reóstato). El diseño utiliza una resistencia de alta impedancia en la rama superior para minimizar el consumo de corriente del sensor y evitar afectar la carga principal.

$$V_{ADC} = V_{Flyback} \cdot \frac{R_2}{R_1 + R_2} \tag{16}$$

Valores seleccionados:

- $R_1 = 22\,k\Omega$ (Rama superior, conectada a V_{out})
- $R_2 = 1\,k\Omega$ (Rama inferior, conectada a GND)

Con esta configuración, para una entrada máxima de 60V, el voltaje en el pin del ADC es:

$$V_{ADC(max)} = 60V \cdot \frac{1000}{23000} \approx 2.61V$$

Este valor se encuentra dentro del margen de seguridad del ESP32 (0V–3.3V).

C. Implementación del Firmware (ESP32)

El firmware del nodo sensor fue desarrollado en C++ (Arduino Framework) y realiza las siguientes funciones críticas:

1) Conectividad Wi-Fi: El ESP32 se conecta a la red local y levanta un servidor en el puerto 8080.
2) Filtrado Digital: Para mitigar el ruido eléctrico inherente a la conmutación de alta frecuencia del Flyback, se implementó una técnica de sobremuestreo (*oversampling*), promediando 50 lecturas consecutivas del ADC antes de emitir un valor válido.
3) Transmisión de Datos: Los valores procesados se envían al cliente Python utilizando un protocolo de texto simple (`V:xx.xx`), lo que permite una fácil decodificación en el lado de la interfaz.

D. Integración con la Interfaz HMI

La interfaz gráfica original fue refactorizada para operar como un sistema híbrido. Ahora incorpora un gestor de conexiones dual:

- Canal de Control (Serial/USB): Mantiene la comunicación bidireccional con el PIC18F4550 para el envío de comandos PWM.
- Canal de Telemetría (Wi-Fi/Socket): Gestiona la recepción asíncrona de datos desde el ESP32.

VII. Conclusiones

El desarrollo e implementación del convertidor DC-DC Flyback aislado ha permitido validar experimentalmente los conceptos teóricos de la electrónica de potencia y el control digital. A partir de los resultados obtenidos, se destacan las siguientes conclusiones:

- Validación de la Topología y Regulación: Se logró implementar exitosamente un convertidor capaz de elevar un voltaje de entrada de 11-14 V a un rango de salida regulado de 40-50 V. Las pruebas experimentales demostraron la capacidad del circuito para sostener el voltaje deseado, obteniendo una eficiencia energética práctica cercana al 60% en condiciones de carga.
- Efectividad del Control Digital: La etapa de control basada en el microcontrolador PIC18F4550 demostró ser robusta y precisa. La configuración de los módulos CCP y Timer2 permitió generar una señal PWM estable de 60 kHz con resolución de 10 bits, la cual, acoplada mediante el driver HCPL-3120, garantizó la correcta conmutación del MOSFET dentro de los rangos de voltaje ($V_{GS} \approx 18V$) y corriente requeridos.
- Interfaz HMI y Comunicación: La implementación de la interfaz gráfica en Python (PySide6) validó exitosamente la arquitectura de control. El sistema fue capaz de gestionar el envío de comandos de control (ciclo de trabajo y frecuencia) vía comunicación Serial hacia el PIC.
- Optimización del Diseño (SMD): Finalmente, el análisis del prototipo llevó a la propuesta de un diseño final basado en componentes de montaje superficial (SMD). Esta evolución del diseño permitiría reducir drásticamente las dimensiones físicas del circuito (aprox. 5x3 cm) y minimizar las inductancias parásitas asociadas a las pistas largas y componentes through-hole, mejorando potencialmente la eficiencia y la respuesta ante el ruido electromagnético.

References

[1] Malik, N. R., *Circuitos electrónicos: análisis, simulación y diseño*, Prentice Hall, 1996.

[2] Hart, D. W., *Electrónica de Potencia*, Prentice Hall, 2001.

[3] Giménez Pérez, A., "Diseño y simulación de un convertidor Flyback para un sistema fotovoltaico", Trabajo de Fin de Grado, Universitat Politècnica de València, Escuela Técnica Superior de Ingeniería del Diseño, Valencia, 2016.

[4] Panacek, J., "Creating a Primary-Side Regulation Flyback Converter Using a Conventional Boost Controller", Texas Instruments Power Supply Design Seminar, SEM2600, Topic 6, Documento SLUP416, 2024.

[5] Hudson, T. & Ametller, M., “How to Design a Flyback Converter in Seven Steps”, Monolithic Power Systems (MPS), Application Note.
[6] Basso, C., “The Flyback Converter with Leakage Inductance Part I – Hardware Description”, ON Semiconductor.

APPENDIX A
CÓDIGO FUENTE MICROCONTROLADOR NIVEL 2

```
/*
 * Archivo principal para simulaci n de Flyback (Modo USB)
 * Lee un voltaje en AN0 y ajusta el Duty Cycle del PWM en CCP1.
 *
 * *** CONFIGURACI N PARA USB NATIVO ***
 * Frecuencia del Cristal Externo: 20 MHz
 * Frecuencia del PLL: 96 MHz
 * Frecuencia del CPU (Fosc): 48 MHz
 * Frecuencia del USB: 48 MHz
 * Frecuencia del PWM: ~60.6 kHz (recalculada)
 */

#include <xc.h>

// --- Definici n del Reloj ---
#define _XTAL_FREQ 48000000

// --- BITS DE CONFIGURACI N (FUSES)
#pragma config FOSC = HSPLL_HS    // Oscilador High-Speed (HS) con PLL habilitado
#pragma config PLLDIV = 5         // Prescaler del PLL: divide (20 MHz / 5) = 4 MHz para el PLL
#pragma config CPUDIV = OSC1_PLL2 // Postscaler del CPU: (96 MHz PLL / 2) = 48 MHz para CPU
#pragma config USBDIV = 2         // Postscaler del USB: (96 MHz PLL / 2) = 48 MHz para USB

// --- Fuses ---
#pragma config WDT = OFF          // Watchdog Timer desactivado
#pragma config MCLRE = OFF        // Pin MCLR desactivado
#pragma config LVP = OFF          // Low Voltage Programming desactivado
#pragma config PBADEN = OFF       // Pines PORTB<4:0> como digitales
#pragma config CCP2MX = ON        // CCP2 se multiplexa a RC1 (default)

void main(void) {

    unsigned int adc_valor;         // Variable de 16 bits para guardar la lectura (es de 10
        bits)
    unsigned int duty_cycle_pwm;    // Variable para calcular el duty cycle (10 bits)

    // 2. --- Configurar ADC (AN0) ---
    ADCON1bits.PCFG = 0b1110; // AN0 Anal gico, el resto Digital
    TRISAbits.TRISA0 = 1;     // Poner pin AN0 (RA0) como entrada

    // ( RECALCULADO !) Reloj del ADC (Tad)
    // Fosc = 48 MHz. Necesitamos un Tad > 0.7us.
    // 48 MHz / 32 = 1.5 MHz (Tad = 0.66us)
    // 48 MHz / 64 = 0.75 MHz (Tad = 1.33us -> m s seguro)
    // Usemos Fosc/64 = 0b110
    ADCON2bits.ADCS = 0b110; // Reloj ADC = Fosc/64
    ADCON2bits.ACQT = 0b010; // Tiempo de adquisici n 4 TAD
    ADCON2bits.ADFM = 1;     // Justificado a la derecha

    ADCON0bits.CHS = 0b0000; // Seleccionar Canal AN0 por defecto
    ADCON0bits.ADON = 1;     // Encender el m dulo ADC

    // 3. --- Configurar PWM (CCP1) ---
    TRISCbits.TRISC2 = 0;     // Poner pin CCP1 (RC2) como salida
    CCP1CONbits.CCP1M = 0b1100; // Configurar CCP1 en modo PWM

    // 4. --- Configurar Timer2 (Base de tiempo del PWM) ---
    // ( RECALCULADO !)
    // F_timer_clock = Fosc / 4 = 48 MHz / 4 = 12 MHz
    // Queremos F_pwm = ~60.6 kHz
    // F_pwm = F_timer_clock / ( (PR2 + 1) * Prescaler )
    // 60600 = 12,000,000 / ( (PR2 + 1) * 1 )
    // (PR2 + 1) = 12,000,000 / 60600 = 198.01
    // PR2 = 197
```

```
    //
    // Frecuencia real: 12,000,000 / (198 * 1) = 60,606 Hz ( Perfecto !)
    PR2 = 197;                // ( NUEVO  VALOR!) Valor del Periodo (PR2=197)
    T2CONbits.T2CKPS = 0b00;  // Prescaler = 1 (Sin cambios)
    T2CONbits.TMR2ON = 1;     // Encender el Timer2

    while(1) {
        // --- Aqu  va el c digo de lectura del ADC ---
        __delay_us(5); // Este delay ahora es preciso a 48 MHz
        ADCON0bits.GO_nDONE = 1;
        while(ADCON0bits.GO_nDONE);
        adc_valor = (ADRESH << 8) | ADRESL;

        //El valor m ximo del Duty Cycle ahora es (PR2+1)*4
        // (197 + 1) * 4 = 198 * 4 = 792
        duty_cycle_pwm = (unsigned long)adc_valor * 792 / 1023;

        CCPR1L = (duty_cycle_pwm >> 2);
        CCP1CONbits.DC1B = (duty_cycle_pwm & 0x03);
    }
}
}
```

APPENDIX B
CÓDIGO FUENTE MICROCONTROLADOR NIVEL 3

```
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// --- 1. CONFIGURACI N DE FUSES (PIC18F4550 a 48MHz) ---
// Cristal Externo de 20MHz -> PLL (96MHz) -> CPU (48MHz)
#pragma config PLLDIV = 5         // (20 MHz / 5) = 4 MHz entrada al PLL
#pragma config CPUDIV = OSC1_PLL2 // (96 MHz / 2) = 48 MHz reloj del sistema
#pragma config USBDIV = 2         // Clock USB a 48 MHz
#pragma config FOSC = HSPLL_HS    // Oscilador HS con PLL habilitado
#pragma config FCMEN = OFF        // Fail-Safe Clock Monitor deshabilitado
#pragma config IESO = OFF         // Oscilador Switchover deshabilitado
#pragma config PWRT = OFF         // Power-up Timer deshabilitado
#pragma config BOR = OFF          // Brown-out Reset deshabilitado
#pragma config WDT = OFF          // Watchdog Timer deshabilitado
#pragma config MCLRE = ON         // MCLR pin habilitado
#pragma config LVP = OFF          // Low Voltage Programming deshabilitado
#pragma config PBADEN = OFF       // PortB como digital I/O en reset

// --- 2. DEFINICI N DE FRECUENCIA ---
#define _XTAL_FREQ 48000000 // 48 MHz

// --- CONSTANTES ---
#define RX_BUF_SIZE 20

// --- VARIABLES GLOBALES ---
char rx_buffer[RX_BUF_SIZE];
volatile unsigned char rx_index = 0;
volatile unsigned char cmd_received = 0;

// Valores iniciales seguros
float current_freq = 60000.0; // 60 kHz para Flyback
float current_duty = 0.0;     // Empezar apagado por seguridad

// --- PROTOTIPOS ---
void setup(void);
```

```
void update_pwm(float freq, float duty);
void process_command(void);

// --- INTERRUPCI N DE ALTA PRIORIDAD ---
void __interrupt() isr(void) {
    // Verificar error de Overrun (buffer lleno)
    if (RCSTAbits.OERR) {
        RCSTAbits.CREN = 0; // Reiniciar m dulo receptor
        RCSTAbits.CREN = 1;
    }

    // Interrupci n por recepci n de datos UART
    if (PIR1bits.RCIF) {
        char c = RCREG;
        // Detectar fin de comando (Enter o Salto de l nea)
        if (c == '\n' || c == '\r') {
            rx_buffer[rx_index] = '\0'; // Terminar string
            cmd_received = 1;           // Bandera para el main
            rx_index = 0;               // Resetear  ndice
        }
        else if (rx_index < (RX_BUF_SIZE - 2)) {
            rx_buffer[rx_index++] = c;
        }
    }
}

// --- MAIN LOOP ---
void main(void) {
    setup();

    // Bucle infinito esperando  rdenes  de Python
    while (1) {
        if (cmd_received) {
            process_command();
            cmd_received = 0;
        }
    }
}

// --- CONFIGURACI N DE HARDWARE ---
void setup(void) {
    // 1. Puertos
    ADCON1 = 0x0F;        // Todos los pines digitales
    TRISCbits.TRISC2 = 0; // RC2 como Salida (Pin del PWM / CCP1)

    // 2. UART: Configuraci n para 9600 Baudios @ 48MHz
    // F rmula: Baud = Fosc / (4 * (SPBRG + 1)) -> BRGH=1, BRG16=1
    // SPBRG = (48000000 / (4 * 9600)) - 1 = 1249
    TRISCbits.TRISC6 = 0; // TX Salida
    TRISCbits.TRISC7 = 1; // RX Entrada

    TXSTAbits.BRGH = 1;    // Alta velocidad
    BAUDCONbits.BRG16 = 1; // 16-bit Baud Rate Generator

    // Cargar valor 1249 (0x04E1) en registros SPBRG
    SPBRG = 0xE1;  // Byte bajo
    SPBRGH = 0x04; // Byte alto

    TXSTAbits.SYNC = 0;    // Modo As ncrono
    RCSTAbits.SPEN = 1;    // Habilitar puerto serie
    TXSTAbits.TXEN = 1;    // Habilitar transmisi n
    RCSTAbits.CREN = 1;    // Habilitar recepci n continua

    // 3. Interrupciones
    PIE1bits.RCIE = 1;     // Habilitar int. recepci n UART
    INTCONbits.PEIE = 1;   // Habilitar int. perif ricos
    INTCONbits.GIE = 1;    // Habilitar int. globales
```

```
105
106     // 4. Iniciar PWM apagado
107     update_pwm(current_freq, current_duty);
108     T2CONbits.TMR2ON = 1;  // Encender Timer 2
109 }
110
111 // --- L GICA PWM INTELIGENTE (Auto-Prescaler) ---
112 void update_pwm(float freq, float duty) {
113     if (freq < 1000.0) freq = 1000.0; // Protecci n  m nima
114
115     unsigned char prescaler_bits = 0; // 00 = 1:1
116     unsigned char prescaler_val = 1;
117
118     // --- CALCULO AUTOM TICO DE PRESCALER ---
119     // Intentamos encajar la frecuencia en el Timer2 (8 bits)
120     // Probamos 1:1, si desborda (>255), subimos a 1:4, luego a 1:16.
121
122     // Intento 1: Prescaler 1:1
123     float pr2_calc = (48000000.0 / (4.0 * freq * 1.0)) - 1.0;
124
125     if (pr2_calc > 255.0) {
126         // Intento 2: Prescaler 1:4
127         prescaler_val = 4;
128         prescaler_bits = 0b01;
129         pr2_calc = (48000000.0 / (4.0 * freq * 4.0)) - 1.0;
130     }
131
132     if (pr2_calc > 255.0) {
133         // Intento 3: Prescaler 1:16
134         prescaler_val = 16;
135         prescaler_bits = 0b10; // 1x = 1:16
136         pr2_calc = (48000000.0 / (4.0 * freq * 16.0)) - 1.0;
137     }
138
139     // Limitar PR2 a 255 (8 bits)
140     if (pr2_calc > 255.0) pr2_calc = 255.0;
141     if (pr2_calc < 0.0) pr2_calc = 0.0;
142
143     unsigned char pr2_reg = (unsigned char)pr2_calc;
144     PR2 = pr2_reg;
145
146     // Aplicar Prescaler sin apagar el Timer (Masking seguro)
147     // T2CKPS son los bits 0 y 1 de T2CON.
148     T2CONbits.T2CKPS = prescaler_bits;
149
150     // --- C LCULO DE DUTY CYCLE ---
151     // Ciclo de trabajo basado en los ticks totales disponibles
152     float max_ticks = (float)((pr2_reg + 1) * 4);
153     unsigned int duty_ticks = (unsigned int)((max_ticks * duty) / 100.0);
154
155     CCPR1L = duty_ticks >> 2;          // 8 bits altos
156     CCP1CONbits.DC1B = duty_ticks & 0x03; // 2 bits bajos
157
158     CCP1CONbits.CCP1M = 0b1100; // Asegurar modo PWM
159 }
160
161 // --- PROCESAMIENTO DE COMANDOS UART ---
162 void process_command(void) {
163     // Formato esperado: "F20000" o "D50"
164     if (rx_buffer[0] == 'F' || rx_buffer[0] == 'f') {
165         current_freq = atof(&rx_buffer[1]);
166         update_pwm(current_freq, current_duty);
167     }
168     else if (rx_buffer[0] == 'D' || rx_buffer[0] == 'd') {
169         current_duty = atof(&rx_buffer[1]);
170         // L mite de seguridad: nunca llegar al 100% en un Flyback
171         if (current_duty > 95.0) current_duty = 95.0;
```

```
        update_pwm(current_freq, current_duty);
    }
}
```

APPENDIX C
CÓDIGO INTERFAZ NIVEL 3

```

import sys
import numpy as np
import serial
import serial.tools.list_ports
import time
from PySide6.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout,
    QHBoxLayout, QLabel, QSlider,
    QPushButton, QTextEdit, QComboBox, QMessageBox
)
from PySide6.QtCore import Qt, QTimer
import pyqtgraph as pg

# Configuraci n Inicial (Arranca en Oscuro)
pg.setConfigOption('background', '#0F172A')
pg.setConfigOption('foreground', '#E2E8F0')
pg.setConfigOption('antialias', True)

class FlybackInterface(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("PIC18F4550 Flyback Controller - Final")
        self.resize(1000, 600)

        # Estado del tema
        self.is_dark_mode = True

        # --- PALETA MODO OSCURO (Modern Navy) ---
        self.dark_theme = {
            'bg_main': '#0F172A',       # Fondo Principal
            'bg_panel': '#1E293B',      # Paneles/Cajas
            'text_main': '#F1F5F9',     # Texto Principal
            'text_sec': '#94A3B8',      # Texto Secundario
            'accent': '#38BDF8',        # Azul Cian (Highlight)
            'border': '#334155',        # Bordes
            'btn_bg': '#334155',        # Botones fondo
            'btn_hover': '#475569',
            'graph_bg': '#0F172A',      # Fondo Gr fico
            'graph_axis': '#E2E8F0',    # Color de n meros/ejes
            'console_bg': '#1E293B'
        }

        # --- PALETA MODO CLARO (Professional Grey - NO BLANCO PURO) ---
        self.light_theme = {
            'bg_main': '#E2E8F0',       # Gris medio (Slate 200)
            'bg_panel': '#F1F5F9',      # Gris muy claro (Slate 100)
            'text_main': '#0F172A',     # Azul muy oscuro (Casi negro)
            'text_sec': '#475569',      # Gris oscuro
            'accent': '#0284C7',        # Azul fuerte
            'border': '#CBD5E1',        # Bordes grises
            'btn_bg': '#CBD5E1',        # Botones fondo
            'btn_hover': '#94A3B8',
            'graph_bg': '#E2E8F0',      # Fondo Gr fico claro
            'graph_axis': '#0F172A',    #  Ejes  oscuros para que se vean!
            'console_bg': '#F8FAFC'
```

```
        }

        # Variables de L gica
        self.ser = None
        self.is_connected = False
        self.duty_cycle = 50.0
        self.freq = 60000.0

        self.last_sent_duty = -1.0
        self.last_sent_freq = -1.0
        self.last_send_time = 0
        self.send_interval = 0.05

        self.sim_voltage = 0.0
        self.volt_history = np.zeros(200)

        # Inicializar UI
        self.init_ui()
        self.refresh_ports()

        # Aplicar tema oscuro por defecto
        self.apply_theme(self.dark_theme)

        self.timer = QTimer()
        self.timer.timeout.connect(self.system_loop)

    def init_ui(self):
        main_widget = QWidget()
        self.setCentralWidget(main_widget)
        layout = QHBoxLayout(main_widget)

        # --- IZQUIERDA: GR FICAS ---
        graphs_layout = QVBoxLayout()

        header_layout = QHBoxLayout()
        header_layout.addStretch()
        self.btn_theme = QPushButton("       Cambiar Tema")
        self.btn_theme.setFixedWidth(130)
        self.btn_theme.clicked.connect(self.toggle_theme)
        header_layout.addWidget(self.btn_theme)
        graphs_layout.addLayout(header_layout)

        # Gr fico PWM
        self.pwm_plot = pg.PlotWidget(title="Se al PWM (Duty Cycle)")
        self.pwm_plot.setYRange(-0.5, 5.5)

        self.pwm_curve = self.pwm_plot.plot(pen=pg.mkPen('#38BDF8', width=3))
        graphs_layout.addWidget(self.pwm_plot)

        # Gr fica Voltaje
        self.volt_plot = pg.PlotWidget(title="Voltaje de Salida (Estimado)")
        self.volt_plot.setYRange(0, 65)
        self.volt_curve = self.volt_plot.plot(pen=pg.mkPen('#22C55E', width=3))
        # Relleno
        self.volt_fill = pg.FillBetweenItem(self.volt_curve, self.volt_plot.plot(pen=None),
            brush=pg.mkBrush(34, 197, 94, 50))
        self.volt_plot.addItem(self.volt_fill)
        graphs_layout.addWidget(self.volt_plot)

        # Consola
        self.console = QTextEdit()
        self.console.setReadOnly(True)
        self.console.setMinimumHeight(200)
        self.console.setMaximumHeight(300)
        graphs_layout.addWidget(self.console)

        layout.addLayout(graphs_layout, stretch=2)
```

```
123
124         # --- DERECHA: CONTROLES ---
125         self.controls_panel = QWidget() # Guardamos ref para estilo
126         self.controls_panel.setFixedWidth(340)
127         controls_layout = QVBoxLayout(self.controls_panel)
128         controls_layout.setSpacing(20)
129         controls_layout.setContentsMargins(20, 20, 20, 20)
130
131         # Conexi n
132         self.lbl_conn = QLabel("Configuraci n Serial")
133         self.lbl_conn.setStyleSheet("font-weight: bold; font-size: 11pt;")
134         controls_layout.addWidget(self.lbl_conn)
135
136         self.port_combo = QComboBox()
137         self.btn_refresh = QPushButton("Actualizar Puertos")
138         self.btn_refresh.clicked.connect(self.refresh_ports)
139
140         self.btn_connect = QPushButton("CONECTAR")
141         self.btn_connect.setFixedHeight(40)
142         self.btn_connect.clicked.connect(self.toggle_connect)
143
144         controls_layout.addWidget(self.port_combo)
145         controls_layout.addWidget(self.btn_refresh)
146         controls_layout.addWidget(self.btn_connect)
147
148         # Separador
149         self.line_sep = QLabel(); self.line_sep.setFixedHeight(2)
150         controls_layout.addWidget(self.line_sep)
151
152         # Frecuencia
153         self.lbl_freq_title = QLabel("Frecuencia")
154         self.lbl_freq = QLabel("60.0 kHz")
155         self.lbl_freq.setStyleSheet("font-size: 16pt; font-weight: bold;")
156         self.lbl_freq.setAlignment(Qt.AlignmentFlag.AlignRight)
157
158         freq_row = QHBoxLayout()
159         freq_row.addWidget(self.lbl_freq_title)
160         freq_row.addWidget(self.lbl_freq)
161         controls_layout.addLayout(freq_row)
162
163         self.slider_freq = QSlider(Qt.Orientation.Horizontal)
164         self.slider_freq.setRange(500000, 700000)
165         self.slider_freq.setValue(600000)
166         self.slider_freq.valueChanged.connect(self.update_freq_ui)
167         controls_layout.addWidget(self.slider_freq)
168
169         # Duty Cycle
170         self.lbl_duty_title = QLabel("Duty Cycle")
171         self.lbl_duty = QLabel("50.0 %")
172         self.lbl_duty.setStyleSheet("font-size: 16pt; font-weight: bold;")
173         self.lbl_duty.setAlignment(Qt.AlignmentFlag.AlignRight)
174
175         duty_row = QHBoxLayout()
176         duty_row.addWidget(self.lbl_duty_title)
177         duty_row.addWidget(self.lbl_duty)
178         controls_layout.addLayout(duty_row)
179
180         self.slider_duty = QSlider(Qt.Orientation.Horizontal)
181         self.slider_duty.setRange(0, 900)
182         self.slider_duty.setValue(500)
183         self.slider_duty.valueChanged.connect(self.update_duty_ui)
184         controls_layout.addWidget(self.slider_duty)
185
186         # Display Voltaje
187         controls_layout.addSpacing(20)
188         self.lbl_v_title = QLabel("Voltaje Estimado")
189         self.lbl_v_title.setAlignment(Qt.AlignmentFlag.AlignCenter)
```

```
        controls_layout.addWidget(self.lbl_v_title)

        self.lbl_voltage_big = QLabel("0.00 V")
        self.lbl_voltage_big.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.lbl_voltage_big.setStyleSheet("font-size: 42pt; font-weight: 800; color: #38BDF8;
            border: 2px solid #334155; border-radius: 12px; padding: 10px;")
        controls_layout.addWidget(self.lbl_voltage_big)

        controls_layout.addStretch()
        layout.addWidget(self.controls_panel)

    def toggle_theme(self):
        self.is_dark_mode = not self.is_dark_mode
        if self.is_dark_mode:
            self.btn_theme.setText("       Modo Claro")
            self.apply_theme(self.dark_theme)
        else:
            self.btn_theme.setText("     Modo Oscuro")
            self.apply_theme(self.light_theme)

    def apply_theme(self, t):
        # 1. CSS GLOBAL
        self.setStyleSheet(f"""
            QMainWindow {{ background-color: {t['bg_main']}; color: {t['text_main']}; }}
            QLabel {{ color: {t['text_main']}; }}

            /* Panel Derecho */
            QWidget {{ font-family: 'Segoe UI', sans-serif; }}

            /* Scrollbar */
            QScrollBar:vertical {{ border: none; background: {t['bg_main']}; width: 10px; }}
            QScrollBar::handle:vertical {{ background: {t['border']}; border-radius: 5px; min-
                height: 20px; }}
            QScrollBar::handle:vertical:hover {{ background: {t['btn_hover']}; }}
            QScrollBar::add-line:vertical, QScrollBar::sub-line:vertical {{ height: 0px; }}
            QScrollBar::add-page:vertical, QScrollBar::sub-page:vertical {{ background: none;
                }}

            /* Combobox */
            QComboBox {{
                background-color: {t['bg_panel']}; color: {t['text_main']};
                border: 1px solid {t['border']}; border-radius: 5px; padding: 8px; font-size:
                    11pt;
            }}
              QComboBox::drop-down {{
                subcontrol-origin: padding; subcontrol-position: top right; width: 20px; border
                    -left-width: 0px;
            }}
            QComboBox QAbstractItemView {{
                background-color: {t['bg_panel']}; color: {t['text_main']}; selection-
                    background-color: {t['accent']};
            }}

            /* Botones Comunes */
            QPushButton {{
                background-color: {t['btn_bg']}; color: {t['text_main']};
                border: 1px solid {t['border']}; border-radius: 5px; padding: 8px; font-weight:
                     bold;
            }}
            QPushButton:hover {{ background-color: {t['btn_hover']}; }}

            /* Consola */
            QTextEdit {{
                background-color: {t['console_bg']}; color: {t['text_main']};
                border: 1px solid {t['border']}; border-radius: 5px;
                font-family: 'Consolas', monospace; font-size: 10pt;
            }}
```

```
250
251             /* Sliders */
252             QSlider::groove:horizontal {{
253                 border: 1px solid {t['border']}; height: 6px; background: {t['btn_bg']}; margin
                        : 2px 0; border-radius: 3px;
254             }}
255             QSlider::handle:horizontal {{
256                 background: {t['accent']}; border: 1px solid {t['accent']};
257                 width: 16px; height: 16px; margin: -6px 0; border-radius: 8px;
258             }}
259         """)
260
261         self.controls_panel.setStyleSheet(f".QWidget {{ background-color: {t['bg_panel']};
                    border-left: 1px solid {t['border']}; }}")
262         self.line_sep.setStyleSheet(f"border-bottom: 2px solid {t['border']};")
263
264         if not self.is_connected:
265             self.btn_connect.setStyleSheet("background-color: #22C55E; color: white; border:
                    none; font-weight: bold; border-radius: 5px;")
266         else:
267             self.btn_connect.setStyleSheet("background-color: #EF4444; color: white; border:
                    none; font-weight: bold; border-radius: 5px;")
268
269         self.lbl_voltage_big.setStyleSheet(f"""
270             font-size: 42pt; font-weight: 800;
271             color: {t['accent']};
272             border: 2px solid {t['border']};
273             background-color: {t['bg_main']};
274             border-radius: 12px; padding: 10px;
275         """)
276
277         for plot in [self.pwm_plot, self.volt_plot]:
278             plot.setBackground(t['graph_bg'])
279             plot.getAxis('bottom').setPen(t['graph_axis'])
280             plot.getAxis('bottom').setTextPen(t['graph_axis'])
281             plot.getAxis('left').setPen(t['graph_axis'])
282             plot.getAxis('left').setTextPen(t['graph_axis'])
283
284             plot.getPlotItem().titleLabel.item.setHtml(f'<span style="color: {t["text_main"]};
                        font-size: 12pt; font-weight: bold;">{plot.plotItem.titleLabel.text}</span>')
285
286         if self.is_dark_mode:
287             self.pwm_curve.setPen(pg.mkPen('#38BDF8', width=3))
288             self.volt_curve.setPen(pg.mkPen('#22C55E', width=3))
289         else:
290
291             self.pwm_curve.setPen(pg.mkPen('#0284C7', width=3)) # Azul fuerte
292             self.volt_curve.setPen(pg.mkPen('#16A34A', width=3)) # Verde fuerte
293
294     def refresh_ports(self):
295         self.port_combo.clear()
296         ports = serial.tools.list_ports.comports()
297         for p in ports:
298             self.port_combo.addItem(p.device)
299
300     def toggle_connect(self):
301         if not self.is_connected:
302             try:
303                 port = self.port_combo.currentText()
304                 self.ser = serial.Serial(port, 9600, timeout=0.1)
305                 self.is_connected = True
306                 self.btn_connect.setText("DESCONECTAR")
307                 self.btn_connect.setStyleSheet("background-color: #EF4444; color: white; border
                        : none; font-weight: bold; border-radius: 5px;")
308                 self.log("Connected to " + port)
309                 self.send_freq_manual()
310                 self.send_duty_manual()
```

```
                self.timer.start(30)
            except Exception as e:
                QMessageBox.critical(self, "Error", str(e))
        else:
            self.ser.close()
            self.is_connected = False
            self.btn_connect.setText("CONECTAR")
            self.btn_connect.setStyleSheet("background-color: #22C55E; color: white; border:
                none; font-weight: bold; border-radius: 5px;")
            self.console.clear()
            self.log("Disconnected")

            self.pwm_curve.setData([], [])
            self.volt_curve.setData([], [])
            self.sim_voltage = 0.0
            self.lbl_voltage_big.setText("0.00 V")
            self.volt_history = np.zeros(200)
            self.timer.stop()

    def update_freq_ui(self, val):
        self.freq = val / 10.0
        freq_khz = self.freq / 1000.0
        self.lbl_freq.setText(f"{freq_khz:.1f} kHz")

    def update_duty_ui(self, val):
        self.duty_cycle = val / 10.0
        self.lbl_duty.setText(f"{self.duty_cycle:.1f} %")

    def send_freq_manual(self):
        if self.is_connected:
            try: self.ser.write(f"F{self.freq:.1f}\n".encode())
            except: pass

    def send_duty_manual(self):
        if self.is_connected:
            try: self.ser.write(f"D{self.duty_cycle:.1f}\n".encode())
            except: pass

    def system_loop(self):
        current_time = time.time()
        if self.is_connected and (current_time - self.last_send_time > self.send_interval):
            if abs(self.freq - self.last_sent_freq) > 0.05:
                cmd = f"F{self.freq:.1f}\n"
                try:
                    self.ser.write(cmd.encode())
                    self.log(f"TX: {cmd.strip()}")
                    self.last_sent_freq = self.freq
                    self.last_send_time = current_time
                except: pass

            elif abs(self.duty_cycle - self.last_sent_duty) > 0.05:
                cmd = f"D{self.duty_cycle:.1f}\n"
                try:
                    self.ser.write(cmd.encode())
                    self.log(f"TX: {cmd.strip()}")
                    self.last_sent_duty = self.duty_cycle
                    self.last_send_time = current_time
                except: pass

        num_cycles = 5
        x_pwm = []
        y_pwm = []

        if self.duty_cycle < 0.1:
            x_pwm = [0, num_cycles * 100]
            y_pwm = [0, 0]
        else:
```

```
            for i in range(num_cycles):
                offset = i * 100
                x_pwm.extend([offset, offset, offset + self.duty_cycle, offset + self.
                    duty_cycle])
                y_pwm.extend([0, 5, 5, 0])
            x_pwm.append(num_cycles * 100)
            y_pwm.append(0)

        self.pwm_curve.setData(x_pwm, y_pwm)

        vin = 11.0
        n_ratio = 45 / 11
        d_val = min(self.duty_cycle / 100.0, 0.95)
        gain = d_val / (1.0 - d_val)
        ideal_voltage = vin * n_ratio * gain
        saturation_factor = 1.0 / (1.0 + (gain * 0.6))
        target = ideal_voltage * saturation_factor
        self.sim_voltage += (target - self.sim_voltage) * 0.1
        self.lbl_voltage_big.setText(f"{self.sim_voltage:.2f} V")

        noise = np.random.uniform(-0.8, 0.8)

        voltage_plot_val = self.sim_voltage  + noise

        # --- ACTUALIZAR DATOS DE LA GR FICA ---
        self.volt_history = np.roll(self.volt_history, -1)
        self.volt_history[-1] = voltage_plot_val

        self.volt_curve.setData(self.volt_history)

    def log(self, msg):
        if "TX:" in msg:
            msg = msg.replace("TX:", '<span style="color:#A855F7"><b>[TX]</b></span>')
        elif "Connected" in msg:
            msg = f'<span style="color:#22C55E"><b>[OK]</b></span> {msg}'
        elif "Disconnected" in msg:
            msg = f'<span style="color:#EF4444"><b>[INFO]</b></span> {msg}'
        self.console.append(msg)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = FlybackInterface()
    window.show()
    sys.exit(app.exec())
```

APPENDIX D
LISTA DE COMPONENTES DETALLADA

TABLE VII
LISTA DE COMPONENTES DETALLADA

Comp.	**ID**	**Cant.**	**Modelo**	**Especificaciones**	**Fabr.**	**Dist.**	**Valor**	**Link**	**DS**	**Total**
Transformador	WE-FB_EP13	1	750311624	EP13; Ui=10-40V; Uo=15V/1A	Wurth	Mouser	9.68	Compra	DS	9.68
Mosfet	Q1	1	BSC360N15	150V / 36 mΩ / OptiMOS 3	Infineon	Digikey	2.39	Compra	DS	2.39
Driver	U3	1	UCC5350-Q1	10A Single-Channel Iso	TI	TI	2.67	Compra	DS	2.67
Microcontrolador	U1	1	PIC18F45K50	32KB Flash, 2KB RAM	Microchip	Digikey	3.56	Compra	DS	3.56
Crystal	Crystal1	1	FA-238	24.576 MHz ±50ppm	Epson	WinSrc	0.97	Compra	DS	0.97
Entrada USB	J1	1	1981568-1	MICRO USB, TYPE B	TE Conn.	Digikey	2.03	Compra	DS	2.03
Borneras	J2, J3	2	1823638	SPT-THR 1,5/ 2-H-3,5	Phoenix	Phoenix	0.5	Compra	DS	1.00
Diodo Salida	D2	1	SS310	3A, 100V, Schottky	Taiwan	Digikey	0.6	Compra	DS	0.60
Diodo Snubber	D1	1	ES1J	Ultra-fast, Vr≥400V, If≥1A	Diotec	Digikey	0.43	Compra	DS	0.43
R Snubber	RSnubber	1	ERJP08J221V	220 Ω 5% 2/3W 1206	Panas.	Digikey	0.15	Compra	DS	0.15
Cap. Snubber	CSnubber	1	C2012X7R...	0.1 μF 100V X7R	TDK	Digikey	0.16	Compra	DS	0.16
Cap. Salida 1	Cs11,Cs12	2	GRM32ER6...	22μF X5R	Murata	Digikey	1.89	Compra	DS	3.78
Cap. Salida 2	Cs21,Cs22	2	C2012X7R...	0.1 μF 100V X7R	TDK	Digikey	0.16	Compra	DS	0.32
Reg. Voltaje	U2	1	TPS560200	17V In, 500mA Step-Down	TI	Digikey	0.69	Compra	DS	0.69
R Gate	RGate1	1	RC0201...	8 Ω 1% 1/20W 0201	Yageo	Digikey	0.1	Compra	DS	0.10
Cap. Varios	Cdrv/Reg	3	GRM035...	100 nF MLCC	Murata	Digikey	0.2	Compra	DS	0.60
Inductor	L2	1	NR3015...	22 μH 2.1A	Taiyo	Digikey	0.21	Compra	DS	0.21