UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO Y PROGRAMACIÓN DEL CONTROL ACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSP F28027 DE TEXAS INSTRUMENTS

Trabajo Final de Grado

Autor:

Chen Zhu, Oscar Jia Xing

Tutor:

Orts Grau, Salvador

Titulación:

Grado en Ingeniería Electrónica Industrial y Automática

Curso Académico 2020/2021

Agradecimientos

A mi tutor, Salvador Orts Grau, por su ayuda y apoyo, incluso en verano.

A mi familia por estar allí cuando lo necesito.

A mis amigos y compañeros por animarme.

A mi pareja por apoyarme y motivarme en todo momento.

Resumen

El presente proyecto propone el desarrollo e implementación del control modo corriente digital en un convertidor Buck mediante el microcontrolador TMS320F28027FPTT de Texas Instruments. El programa se desarrollará para la placa de desarrollo "LAUNCHXL-F28027F" y el convertidor reductor "BOOSTXL-BUCKCONV", del mismo fabricante. El objetivo es obtener los compensadores del lazo de tensión y corriente a partir del método factor K. Una vez realizada el diseño teórico, se simulará el funcionamiento del sistema a través del programa Matlab, concretamente Simulink, con las librerías Simscape. Luego de verificar correctamente el control, se implementará los parámetros calculados y comprobados dentro del DSC. Además, se desarrollará una interfaz gráfica por puerto serie bajo "QT C++", de forma que permita configurar parámetros de control y realizar un seguimiento de las tensiones de salida del Buck.

Palabras clave: modo corriente media, DSP, QT, control digital, interfaz, DSC, C2000, Buck

Abstract

This Project proposes the design and implementation of digital controlled average current mode Buck converter using the TMS320F28027FPTT microcontroller from Texas Instruments. The program will be developed for the "LAUNCHXL-F28027F" development board and the "BOOSTXL-BUCKCONV" buck converter, from the same manufacturer. Both the current and voltage loop are designed using the K factor method. Once the theoretical design has been carried out, the control will be simulated through the program, Matlab, specifically Simulink, with the Simscape libraries. After successfully verified the control operation, the parameters will be implemented within the DSC. In addition, an interface will be developed under QT C++, allowing the control of the main control parameters and monitor the ouput voltage of the converter.

Keywords: average current mode, DSP, QT, digital control, interface, DSC, C2000, Buck

Contenido

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO Y PROGRAMACIÓN DEL CONTROL ACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSP F28027 DE TEXAS INSTRUMENTS

TRABAJO FINAL DE GRADO

DOCUMENTO I: MEMORIA TÉCNICA

Autor:

Chen Zhu, Oscar Jia Xing

Tutor:

Orts Grau, Salvador

Titulación:

Grado en Ingeniería Electrónica Industrial y Automática

Curso Académico 2020/2021

Índice

Lista de tablas

Lista de figuras

1. OBJETO DEL PROYECTO

El presente proyecto plantea el desarrollo y la programación del control modo corriente media de un convertidor buck mediante un DSP de Texas Instruments (TI). Se pretende emplear la placa de desarrollo "LAUNCHXL-F28027F" y el convertidor reductor "BOOSTXL-BUCKCONV", ambos del fabricante TI.

El alcance del proyecto será el siguiente:

- Estudio de la plataforma hardware a emplear.

- Diseño del control ACC.

- Simulación del control mediante Matlab Simulink ®

- Desarrollo del software e implementación sobre el DSP para el control modo tensión y ACC del convertidor Buck.

- Implementación sobre el hardware disponible y realización de los ensayos experimentales.

- Desarrollo de una interfaz de usuario para PC y su comunicación vía serie con el DSP para la monitorización del sistema, y que además permita la configuración de los parámetros del control implementado.

2. ANTECEDENTES

El objetivo es combinar los conocimientos adquiridos en dos de las asignaturas de la mención de electrónica de la titulación: Sistemas Electrónicos Industriales (SEI) y Sistemas Digitales Aplicados (SDA). Se pretende emplear uno de los DSPs estudiados en SDA para implementar el control modo corriente media estudiado en SEI sobre un convertidor Buck y llevar a la práctica la teoría estudiada. Además, se pretende expandir la base desarrollada en QT impartida en la asignatura de Sistemas Informáticos Industriales (SII) implementando una interfaz del sistema a través del puerto serie. Como resultado del trabajo se obtendrá una práctica experimental para la asignatura de SDA de gran valor didáctico que reforzará los conocimientos adquiridos en las asignaturas antes mencionada junto con la de Electrónica de Potencia.

3. INTRODUCCIÓN

Las fuentes de alimentación conmutadas son una de las formas más sencillas de modificar las tensiones continuas. El convertidor Buck o convertidor reductor, es un convertidor de potencia, sin aislamiento galvánico con una eficiencia superior al 95% y regulable. Es un método para reducir el voltaje y presenta una eficiencia mucho mayor que los puentes resistivos o transformadores. El funcionamiento de este sistema se divide en dos estados:

· $(0 \leq t \leq \delta T)$ Q: ON $\quad i_i = i_L$ **(conducción)**

· $(\delta T \leq t \leq T)$ Q: OFF, D: ON $\quad i_A = i_L$ ***(no conducción)***

*siendo δ, el ciclo de trabajo del interruptor Q.

Observando el modelo del convertidor Buck (ver figura 1), en el primer estado, cuando el tiempo está entre 0 y δ, el interruptor Q se mantiene cerrado permitiendo el paso de la corriente hacia el convertidor. Aquí la corriente de entrada coincide con la corriente por la bobina. A continuación, al cambiar al segundo estado, cuando el tiempo se sitúa entre δ y el periodo, el

interruptor se abre, cortando el paso de la energía y la corriente almacenada en la bobina se descarga por el diodo, igualando la corriente por la bobina con la del diodo.

Comparando las expresiones planteadas anteriormente con el modelo, se puede deducir que la tensión de salida depende del ciclo de trabajo del interruptor a partir de la siguiente ecuación:

$$V_o = V_{AB(AV)} = \frac{1}{T}\int_0^T v_{AB}\, dt = \frac{1}{T} \cdot \delta \cdot \mathrm{T} \cdot V_i = \delta V_i$$

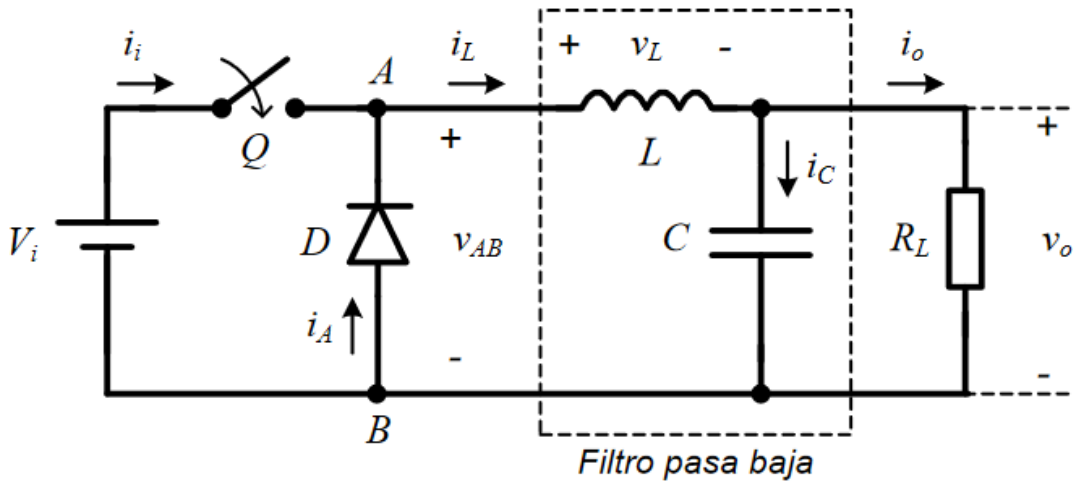


Figura 1. Circuito de un convertidor Buck [1].

El tipo de control del convertidor a implementar será el control ACC, que presenta dos lazos de realimentación: corriente y tensión. Esto permite obtener estabilidad tanto en la tensión de salida como en la corriente por la carga, y su vez, una respuesta rápida.

Para este proyecto, se empleará el método del factor K de Venables. Es una técnica que permite simplificar el diseño de los lazos de control y determinar los componentes electrónicos necesarios para conseguir dichos resultados. Esta herramienta matemática se basa en el diseño de un amplificador de retroalimentación utilizando unas pocas ecuaciones algebraicas para obtener una frecuencia de cruce y un margen de fase (razonable) deseado.

La implementación del control se realizará de manera digital mediante el uso de un microcontrolador de TI, que forma parte de la familia C2000. El fabricante ofrece distintos modelos con un rango variado de potencia, memoria y funciones. El modelo seleccionado es el **TMS320F28027F**, uno de los modelos más simples de la compañía, pero suficiente para los requisitos del proyecto. El MCU permitirá controlar algunos parámetros del control del convertidor. Por ello, una interfaz gráfica simplificaría el manejo de estas variables y conseguir un seguimiento del funcionamiento de esta fuente conmutada.

4. DESCRIPCIÓN DEL HARDWARE A EMPLEAR. LIMITACIONES Y CONDICIONANTES

4.1. Características y limitaciones del convertidor Buck

El modelo de convertidor reductor utilizado es el "BOOSTXL-BUCKCONV" diseñado por "Texas Instruments" (ver figura 2). El modelo del Buck está específicamente preparado para la placa de desarrollo "LAUNCHXL-F280049C", pero también es funcional con el modelo propuesto "**LAUNCHXL-F28027F**".

El convertidor DC-DC necesita una alimentación fija de 9 V tanto para la etapa de potencia como la etapa de control. La corriente máxima por la bobina es de 2 A. La carga es una resistencia variable con un valor entre 1.579 Ω y 7.5 Ω mediante la conmutación de un MOSFET.

Existe un problema con la lectura de la corriente. Cuando el ciclo de trabajo del interruptor supera el 50%, la corriente obtenida es de valor cercano al 0.

4.2. Características y limitaciones del DSP

La placa de desarrollo empleada es el "LAUNCHXL-F28027" que forma parte de la serie F2802x de Texas Instruments (ver figura 3). Integra el depurador "XDS100v2" con conexión USB/UART. La placa incluye 4 LEDs y un botón programable. Además, posee también un botón para el reinicio del sistema. Las conexiones físicas se centran 2 conectores de 2x20 pines.

El microcontrolador integrado es el TMS320F28027F y presenta una CPU de 32 bits trabajando a una velocidad de 60 MHz. La resolución del conversor analógico digital es de 12 bits con un rango máximo de 3.3 V. Este valor determinará la tensión máxima de salida del convertidor a un valor de 6.7 V. La potencia del microcontrolador puede limitar la velocidad de ejecución del código.

Figura 2. Convertidor Buck "BOOSTXL-BUCKCONV" [2].

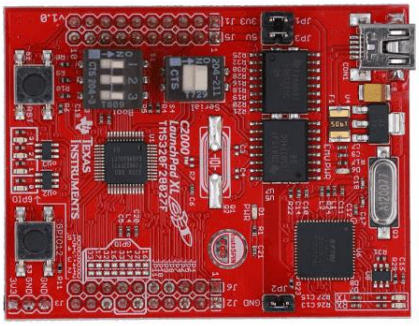


Figura 3. Placa de desarrollo "LAUNCHXL-F28027F" [3].

4.3. Condicionantes del sistema

Este control del convertidor DC-DC se ha diseñado para transformar una tensión de entrada de 9 V a una tensión de salida entre 0.9 V y 4.05 V. Para el control corriente media se limitará el ciclo de trabajo del PWM entre un mínimo de 5 % y un máximo de 45 %. Además, disminuyendo el ciclo de trabajo a valores menores a un 20 % para el transistor de carga, el sistema entra en conducción discontinua.

5. SOLUCIONES ALTERNATIVAS PARA EL DISEÑO DEL SISTEMA DE CONTROL DEL CONVERTIDOR. ELECCIÓN Y JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA.

5.1. Control Modo Tensión

El método de control más básico de un convertidor es el control modo tensión. La característica más destacada de este método es que emplea una retroalimentación de la tensión para su control. Se compara la tensión de salida deseada con la real y se modifica el ciclo de trabajo del interruptor para controlar la tensión en la bobina. El PWM aplicado se genera a partir de la comparación del error de voltaje con una rampa de valor constante. La limitación de corriente debe de realizarse aparte.

Características:

- Un lazo de tensión: más fácil de diseñar y de analizar.

- Facilidad de sensar la tensión.

- Respuesta lenta ante variaciones de la tensión de entrada

5.2. Control Modo Corriente

El control modo corriente es un sistema con dos lazos de realimentación. Una interna de corriente (rápido) y otra externa de tensión. Como en el caso anterior, en el lazo exterior se compara la tensión de salida con la tensión de referencia para obtener el error. A continuación, en el lazo interior, se compara la corriente por la bobina con el error del lazo de tensión.

5.2.1. Modo Corriente Pico

En este modo, durante el comparador finaliza la conducción del interruptor cuando la corriente instantánea alcanza el valor deseado. Este valor vendrá determinado por el lazo de tensión. Normalmente la rampa de la corriente es bastante pequeña por lo que es susceptible a ruidos, especialmente cuando la tensión de entrada es pequeña.

Con este método, el sistema es inestable cuando el ciclo de trabajo excede el 50%, formando oscilaciones subharmónicas. Se suele añadir una rampa externa sumando la corriente medida antes del comparador que elimina esta inestabilidad.

5.2.2. Modo Corriente Media

A diferencia del modo corriente pico, que posee poca ganancia en el lazo de corriente, con este modo se soluciona este problema, al introducir un integrador comparador de alta ganancia en el lazo de corriente. La salida, que es el error de corriente amplificado se vuelve a comparar con una rampa que generará el ciclo del trabajo del interruptor.

4.3. Elección y justificación de la solución adoptada

El control elegido es el modo corriente media (ACCM). El control modo corriente presenta mejores prestaciones en velocidad frente al control modo tensión. Además, al implementarse digitalmente no presenta inconvenientes respecto al incremento de precios en el apartado de control. Asimismo, el control ACCM posee mayor resistencia frente al ruido y no está limitada por la rampa externa, que limita el ciclo de trabajo en el cruce por cero.

6.DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADAPTADA

6.1. Convertidor Buck

6.1.1. Modelo del conmutador PWM

La etapa de potencia de un convertidor Buck puede funcionar en modo conducción continua (**CCM**) o discontinua de la corriente por la bobina. La conducción continua se caracteriza por la corriente conduciendo continuamente durante todo el ciclo de trabajo. Mientras tanto, en la conducción discontinua, la corriente por la bobina alcanza valores nulos en una parte del ciclo de trabajo. El siguiente desarrollo corresponde con el convertidor funcionando en CCM.

El modelo del conmutador PWM es un método para el modelado de convertidores basados en PWM. La idea principal es la sustitución de los interruptores por sus modelos promediados en el tiempo. El interruptor se sustituye por el modelo mostrado en la figura 4.

Para obtener el circuito lineal y simplificar las ecuaciones, se hallará el circuito equivalente entre los tres terminales. La figura 5 muestra la relación entre los terminales del modelo:

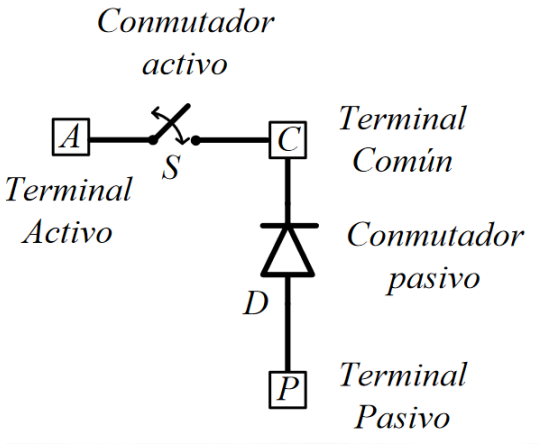


Figura 4. Modelo del conmutador PWM [4].

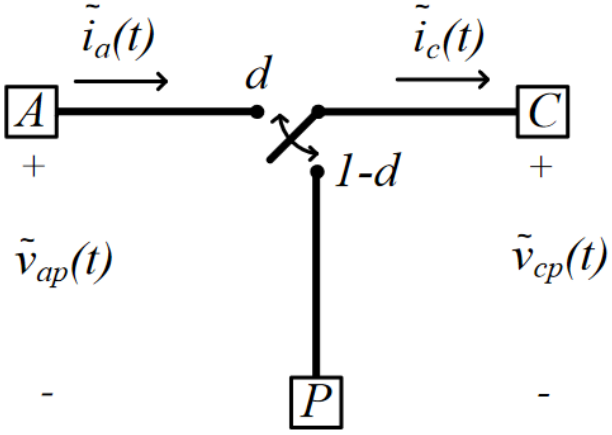


Figura 5. Modelo del conmutador PWM con la relación entre los terminales [4].

A partir de las corrientes y tensiones instantáneas se deduce las siguientes expresiones:

$$i_a(t) = \begin{cases} i_c(t), & 0 < t < dT \\ 0, & dT < t < T \end{cases}$$

$$v_{ap}(t) = \begin{cases} i_c(t), & 0 < t < dT \\ 0, & dT < t < T \end{cases}$$

Promediando estos valores en un ciclo de conmutación se obtiene que:

$$i_a = d \cdot i_c$$

$$v_{cp} = d \cdot v_{ap}$$

La forma propuesta anteriormente es un modelo no lineal de gran señal. Al aplicarle una perturbación a esa señal y linealizarlo, se obtendrá la forma linealizada en el punto de operación. La idea principal es asumir un punto operativa y luego añadir pequeñas variaciones a ese punto de trabajo. Se asumirá un ciclo de trabajo fijo D (la mayúscula indica cantidades DC) y, a continuación, añadirle una pequeña variación $\hat{d}$.

$$i_a = I_a + \hat{\imath}_a$$

$$i_c = I_c + \hat{\imath}_c$$

$$v_{ap} = V_{ap} + \hat{v}_{ap}$$

$$v_{cp} = V_{cp} + \hat{v}_{cp}$$

$$d = D + \hat{d}$$

*el símbolo ^ indica pequeñas perturbaciones en AC y las mayúsculas, la señal en DC.

Ahora, se elimina el producto entre términos de pequeña señal porque las variaciones AC son pequeñas y su producto resulta despreciable. Las funciones resultantes son las siguientes:

$$i_a = I_a + \hat{\imath}_a = D\,I_c + D\,\hat{\imath}_c + \hat{d}\,I_c + \hat{d}\,\hat{\imath}_c \approx D\,I_c + D\,\hat{\imath}_c + \hat{d}\,I_c$$

$$v_{cp} = V_{cp} + \hat{v}_{cp} = D\,V_{ap} + D\,\hat{v}_{ap} + \hat{d}\,V_{ap} + \hat{d}\,\hat{v}_{ap} \approx D\,V_{ap} + D\,\hat{v}_{ap} + \hat{d}\,V_{ap}$$

Separando los términos DC y AC, se obtienen las siguientes ecuaciones:

- **Ecuaciones DC**

$$I_a = D \cdot I_c$$

$$V_{cp} = D \cdot V_{ap}$$

- **Ecuaciones de pequeña señal**

$$\hat{\imath}_a = \hat{d}\,I_c + D\,\hat{\imath}_c$$

$$\hat{v}_{cp} = \hat{d}\,V_{ap} + D\,\hat{v}_{ap}$$

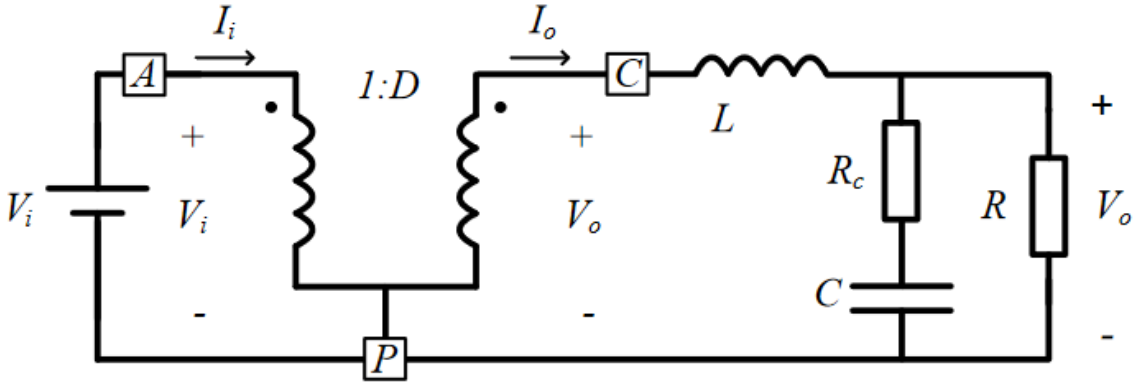


Figura 6. Circuito equivalente DC de un Buck en CCM [4].

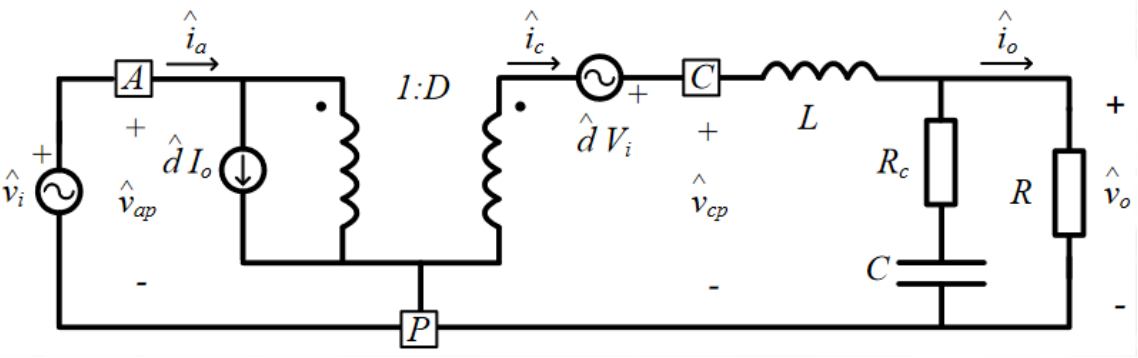


Figura 7. Circuito equivalente de pequeña señal de un Buck en CCM [4].

6.1.1. Características del convertidor Buck

En este caso se trata de un convertidor Buck síncrono donde se sustituye el diodo por otro interruptor FET. Esta nueva topología ofrece una menor resistencia y ausencia de caída de tensión durante la conducción. Sin embargo, aumenta los costes frente al asíncrono y requiere de un "deadband" entre los cambios de estado de los dos interruptores.

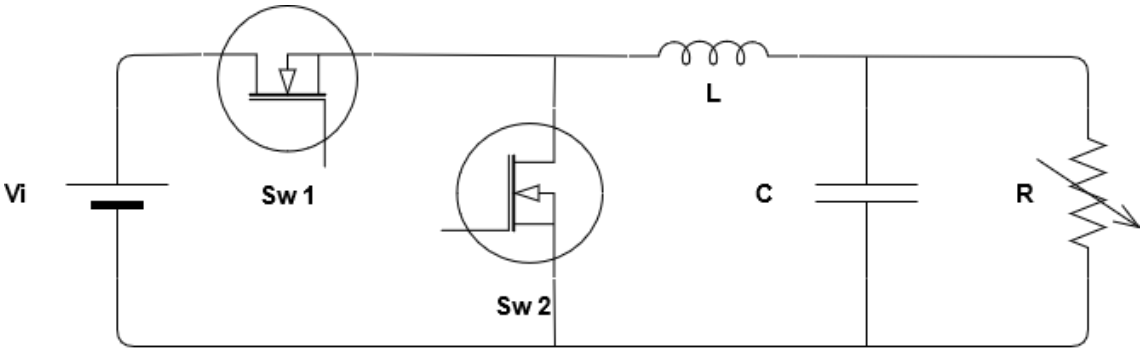


Figura 8. Circuito equivalente del convertidor "BOOSTXL-BUCKCONV".

El sistema trabajará en conducción continua "CCM" a una frecuencia de conmutación de **200 kHz**. Los valores de los componentes del circuito se detallan a continuación:

L	4,8 µH
C	330 µF
R	2 Ω
Vi	9 V

Figura 9. Valores de los componentes del convertidor "BOOSTXL-BUCKCONV"

Como la **resistencia** es variable, se ha fijado a un valor de **2 Ω**. Para los cálculos se tomarán estos datos.

6.1.2. Sensor de corriente

La lectura de la corriente de la bobina (figura 10) se realiza mediante una resistencia shunt (R4) y un amplificador restador (U2). Se obtiene la tensión en la resistencia y empleando la ley de Ohm se puede obtener la corriente que circula por ella.

La ganancia del amplificador vendrá determinada por la resistencia de la realimentación negativa del operacional y las resistencias de las entradas inversora y no inversora. Teniendo en cuenta que las resistencias de las dos puertas de entrada son iguales y aplicando la ley de Ohm, la ganancia del sensor de corriente Ri es:

$$Ri = \frac{R11}{R8 \cdot R4} = 487.8\ V/V$$

La tensión máxima "V_{max}" que es capaz leer el ADC del circuito de sensado se obtiene a partir de la ganancia del amplificador y la tensión máxima de entrada del ADC (3.3 V) será:

$$V_{max} = \frac{R11}{\text{Tensión máxima de entrada del ADC} \cdot R8} = 4.435\ V$$

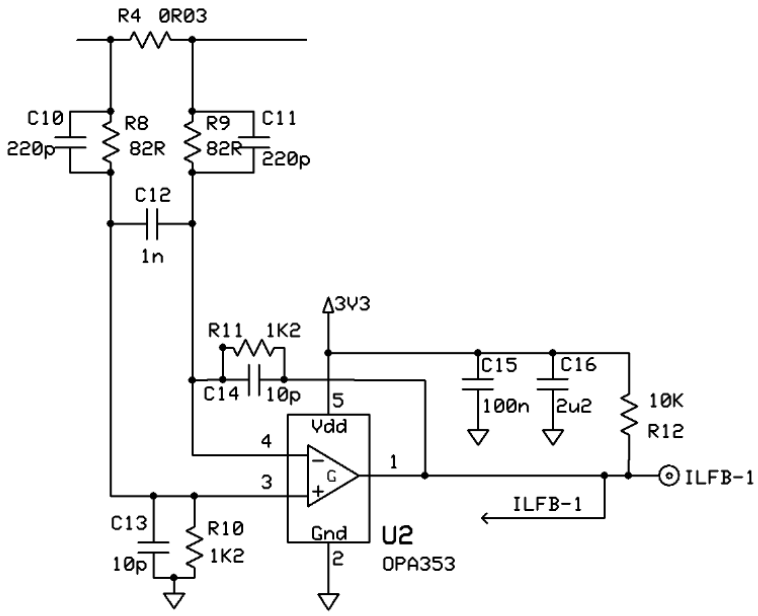


Figura 10. Circuito de sensado de la corriente por la bobina.

6.1.3. Sensor de tensión de salida

La ganancia del sensor de tensión de salida "β" viene dada por el divisor de tensión formador por dos resistencias en paralelo (R5 y R7) y la resistencia R6. Al trabajar en corriente continua el condensador se desprecia.

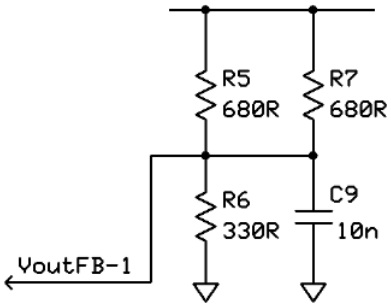


Figura 11. Divisor de tensión a la salida del convertidor.

La ganancia β será:

$$\beta = \frac{R6}{(R6 + \frac{R5 \cdot R7}{R5 + R7})} = 0.4925\ V/V$$

Dado que la lectura de la tensión de salida "VoutFB-1" del puente resistivo viene limitada por la tensión máxima de entrada del ADC (3.3 V), se calculará la tensión máxima de salida del Buck "$Vout_{max}$", para comprobar su límite:

$$\mathrm{Vout}_{max} = \frac{Tensión\ máxima\ de\ entrada\ del\ ADC}{\beta} = 6.7\ V$$

Como se puede observar, el control del Buck deberá de limitar la tensión de salida a 6.7 V. En caso de exceder el valor máximo desestabilizaría el control del convertidor. El DSP no podría determinar esas tensiones debido a las características propias del ADC.

6.1.4. Sensor de tensión de entrada

La ganancia de la tensión de entrada "γ" antes del ADC viene determinada por el puente resistivo formado por R1 y R2. Su ecuación es la siguiente:

$$\gamma = \frac{R2}{R1 + R2} = 0.248\ V/V$$

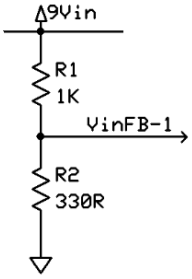


Figura 12. Circuito de la ganancia de la tensión de entrada para su lectura.

6.1.5. Resistencia de carga variable

La resistencia variable está formada por dos resistencias de potencia, PR1 y PR2. Esta última está conectada con un MOSFET como se indica en la siguiente figura.

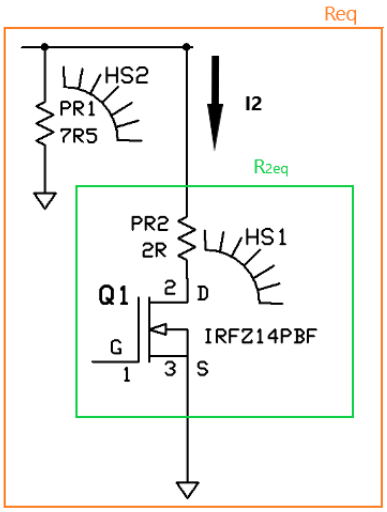


Figura 13. Circuito de la resistencia de carga variable del convertidor "BOOSTXL-BUCKCONV".

Al variar el ciclo de trabajo de la señal PWM que alimenta la puerta del transistor, se modifica el valor de la resistencia equivalente "**R2eq**" entre PR2 y Q1.

$$R_{2eq} = \frac{V_o}{I_{2(AV)}}$$

$$I_{2(AV)} = \frac{1}{T}\int_0^{\delta T} \frac{V_o}{R_{2eq}}\,dt = \frac{\delta V_o}{2}$$

$$R_{2eq} = \frac{2}{\delta}$$

*siendo **δ** el ciclo de trabajo, **I2**, la corriente que circula por la resistencia y **Vo**, la tensión en la carga.

Asimismo, R2eq está situado en paralelo con la resistencia PR1 y, por tanto, influye en la resistencia equivalente "Req" formada por estas dos resistencias.

$$R_{eq} = \frac{1}{\frac{1}{PR1} + \frac{1}{R_{2eq}}} = \frac{15}{2 + 7.5\delta}$$

Como el ciclo de trabajo se sitúa entre un 0% y 100%, el **rango teórico de R**eq se hallará entre un valor de **1.579 Ω y 7.5 Ω**.

Para la implementación en el DSP será necesario obtener el ciclo de trabajo a partir de la resistencia equivalente deseado dentro del rango disponible:

$$\delta = \frac{2}{R_{eq}} - 0.2666$$

6.1.6. Modulador PWM

El modulador PWM se basa en la comparación de una señal de referencia proveniente de un compensador y una señal triangular. El resultado es un tren de pulsos de ancho variable que se utilizan para la conmutación del convertidor. La ganancia del PWM **Fm** viene determinada por la tensión de la señal triangular **Vm** en la siguiente ecuación:

$$Fm = \frac{1}{V_m} = \frac{1}{3\,V} = 0.3333\frac{1}{V}$$

6.2. Control Modo Corriente Media

En este apartado se detallará los cálculos teóricos del control modo corriente media. El esquema general de este control se encuentra en la siguiente figura:

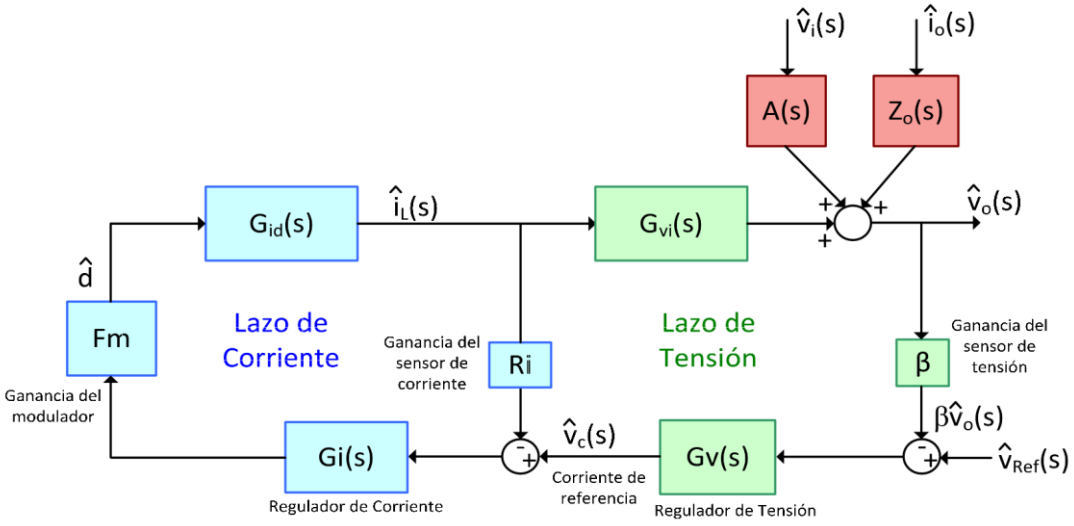


Figura 14. Diagrama de bloques del esquema de control [5].

6.2.1. Método Factor K

Para empezar con el diseño se selecciona primero la frecuencia en la que la ganancia de lazo cerrado sea de valor unitario. La frecuencia de cruce normalmente se elige con el mayor valor posible ya que significa una respuesta transitoria más rápida. A continuación, se determina el margen de fase en la frecuencia de cruce. Un margen de fase 90º significa que el sistema es muy estable. Cuando el MF es de 60º el circuito presenta una buena relación entre estabilidad y velocidad de respuesta. Con 30º o menos, el sistema es muy susceptible a las variaciones externas.

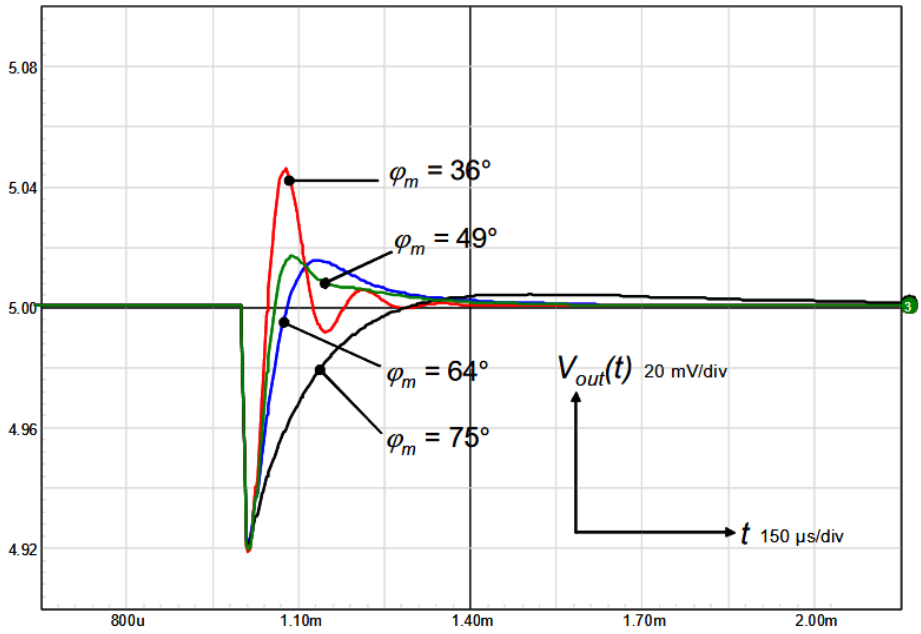


Figura 15. Sobreoscilación respecto al margen de fase [6].

Posteriormente se calcula la ganancia y aumento de fase **AUFA** (si es necesario) en la frecuencia de cruce:

$$AUFA = MF - \arg(G_{id}(j\omega_c)) - 90º$$

Dependiendo del aumento de fase necesario para cumplir los objetivos planteados, se elegirá el tipo de compensador o amplificador:

- **Compensador Tipo 1**. Cuando no es necesario un aumento de fase. Su fórmula es:

$$A_v(j\omega) = \frac{\omega_{p0c}}{j\omega}$$

$$|Ti(j\omega) = |Gid \cdot Ri \cdot Fm \cdot Ai| = 1$$

$$\omega_{p0c} = \frac{\omega_c}{Ri \cdot Fm \cdot |Gid(j\omega c)|}$$

- **Compensador Tipo 2**. Cuando AUFA < 90º. Su fórmula es:

$$A_v(j\omega) = \frac{\omega_{p0c}}{j\omega} \cdot \frac{1 + \frac{j\omega}{\omega_{zc}}}{1 + \frac{j\omega}{\omega_{pc}}}$$

$$K = \tan\left(\frac{AUFA}{2} + 45º\right)$$

$$\omega_{zc} = \frac{\omega_c}{K}$$

$$\omega_{pc} = K \cdot \omega_c$$

- **Compensador Tipo 3**. Si AUFA < 180º. Su fórmula es:

$$A_i(j\omega) = \frac{\omega_{p0c}}{j\omega} \cdot \frac{\left(1 + \frac{j\omega}{\omega_{zc}}\right)^2}{\left(1 + \frac{j\omega}{\omega_{pc}}\right)^2}$$

$$K = \left(\tan\left(\frac{AUFA}{2} + 45º\right)\right)^2$$

$$\omega_{zc} = \frac{\omega_c}{\sqrt{K}}$$

$$\omega_{pc} = \sqrt{K} \cdot \omega_c$$

6.2.2. Función de transferencia del ciclo de trabajo a corriente por la bobina

Para diseñar el lazo de corriente, primero hay que obtener la función de transferencia del ciclo de trabajo a corriente por la bobina a partir del circuito equivalente del convertidor:

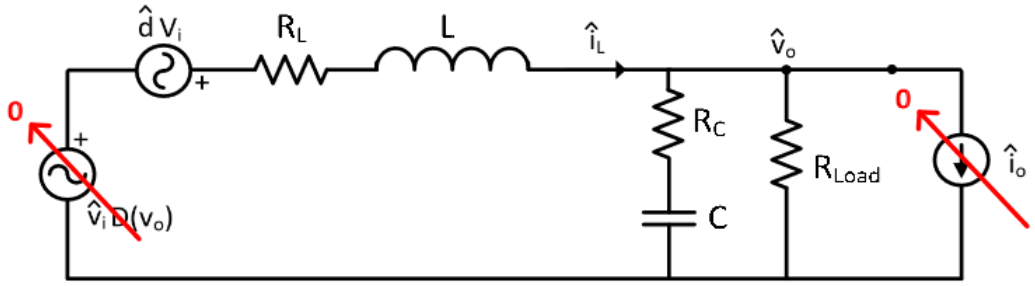


Figura 16. Circuito equivalente de un Buck en modo conducción continua [5].

Tomando la tensión de entrada y la corriente de salida como nulo se obtiene la siguiente expresión:

$$G_{id}(s) = \left.\frac{\hat{\imath}_L(s)}{\hat{d}(s)}\right|_{\hat{v}i=\hat{\imath}o=0}$$

Simplificando el sistema y despreciando el valor resistivo de la bobina, se obtendría este circuito:

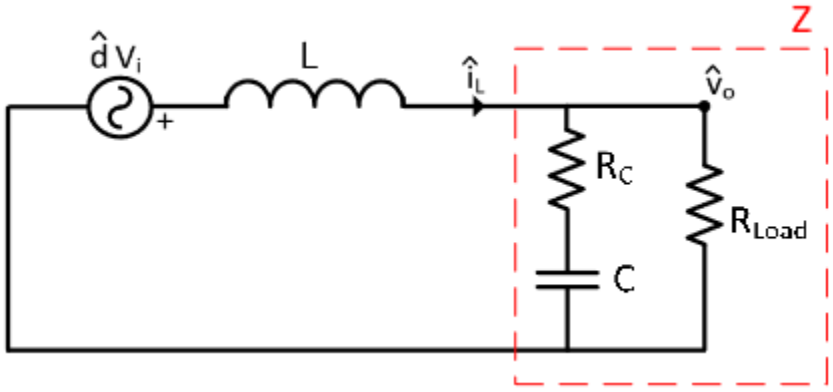


Figura 17. Circuito simplificado de un Buck en modo conducción continua [5].

A partir de este modelo simplificado se obtendrías las siguientes ecuaciones:

$$Z(\omega j) = \frac{R_{Load}(R_c C\omega j + 1)}{(R_{Load} + R_c)C\omega j + 1}$$

$$\hat{\imath}_L = \frac{\hat{d}V_i}{L\omega j + Z(\omega j)} = \frac{\hat{d}V_i}{L \cdot s + Z(s)}$$

$$G_{id}(s) = \frac{\hat{\imath}_L}{\hat{d}} = V_i \frac{1}{L \cdot s + Z(s)}$$

Despreciando la resistencia de la bobina, se obtiene la siguiente función de transferencia, del ciclo de trabajo a corriente por la bobina:

$$G_{id}(s) = \frac{Vi}{R_{Load}} \cdot \frac{1 + s \cdot (R_{Load} + R_c) \cdot C}{S^2 LC \frac{R_{Load} + R_c}{R_{Load}} + s \cdot \left(\frac{L}{R_{Load}} + R_c C\right) + 1}$$

Considerando que Rload >> Rc se obtiene la versión simplificada:

$$G_{id}(s) = \frac{\hat{\imath}_L(s)}{\hat{d}(s)}\bigg|_{\hat{v}i=\hat{\imath}o=0} = \frac{Vi}{R_{Load}} \cdot \frac{1 + s \cdot R_{Load} C}{S^2 LC + s \cdot \left(\frac{L}{R_{Load}} + R_c C\right) + 1}$$

A continuación, se detallan las ecuaciones de la situación del polo "ω_n" y cero "ω_z" y el factor de calidad "Q":

$$\omega_n = \frac{1}{\sqrt{LC\left(\frac{R_{Load} + R_c}{R_{Load}}\right)}}$$

$$\omega_z = \frac{1}{(R_{Load} + R_c) \cdot C}$$

$$f_n = \frac{\omega_n}{2\pi}$$

$$f_z = \frac{\omega_z}{2\pi}$$

$$Q = \frac{1}{\omega_n \left(\frac{L}{R_{Load}} + R_c C\right)}$$

$$\xi = \frac{1}{2Q}$$

*siendo fn y fz, las frecuencias de polo y cero en Hz, respectivamente; RLoad, resistencia de carga; Rc, la ESR del condensador.

A partir de los valores actuales del convertidor de este proyecto, se obtienen estos resultados:

fn	3.64 kHz
fz	199,95 Hz
Q	6,87

Tabla 1. Tabla con las frecuencias del polo y cero, y el factor de calidad.

En la respuesta en frecuencia del filtro, mostrada en la figura 18, se puede comprobar que **fn** coincide con la frecuencia de resonancia.

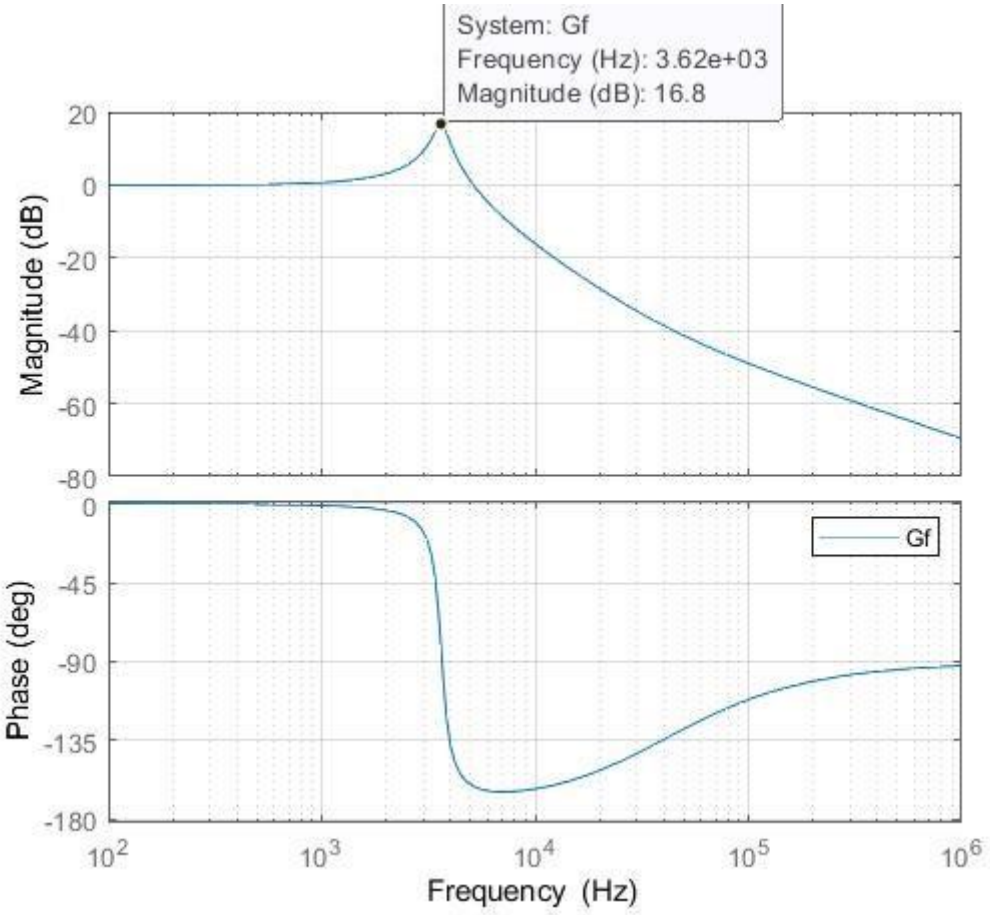


Figura 18. Diagrama de bode del filtro del convertidor "BOOSTXL-BUCKCONV"

6.2.3. Elección de frecuencia de cruce y margen de fase

Para una correcta implementación del regulador, la frecuencia de cruce de corriente **fci** debe de situarse entre dos veces la frecuencia natural **fn** y una quinta parte de la frecuencia de conmutación **fsw.**

$$2f_n < f_{ci} < f_{sw}/5$$

$$7.978\ kHz < f_{ci} < 40kHz$$

$$\boldsymbol{f_{ci} = 5\ kHz}$$

La frecuencia de cruce se ha situado por debajo de las especificaciones. Esto es debido a que, cumpliendo los criterios mencionados anteriormente, el sistema no es estable durante la

implementación en el DSP. Con fci obtenida, se seleccionará la frecuencia de cruce del lazo de tensión fcv, que deberá de ser menor a la mitad de la frecuencia de cruce del lazo de corriente:

$$f_{cv} < f_{ci}/2$$

$$f_{cv} < 2.5\ kHz$$

$$\boldsymbol{f_{cv} = 370\ Hz}$$

A continuación, se elegirá el margen de fase que ofrezca las mejores características en relación con las sobreoscilaciones. El margen de fase (MF), tanto para el lazo de corriente como de tensión será de:

$$MF\ =\ 70º$$

6.2.4. Diseño de lazo de corriente

Los objetivos planteados para el lazo de corriente son los siguientes:

Objetivos	
Fci	5 kHz
MF	70º

Tabla 2. Objetivos de diseño del lazo de corriente.

Para empezar, se obtendrá la fase actual a la frecuencia de cruce:

$$\arg\left(G_{id}(\omega_{ci}\right) = -79.57\ º$$

*siendo wci la frecuencia de cruce del lazo de corriente en rad/s

Y conociendo la fase objetivo:

$$Fase\ objetivo = -180º + MF = -110\ º$$

Se comprueba que es necesario un aumento de fase:

$$AUFA = MF - \arg(G_{id}(j\omega_{ci})) - 90º = 59.57\ º$$

Entonces, como AUFA es menor que 90 º el compensador necesario será de **tipo 2**. Con lo cual, el factor K, y la situación del polo y cero, respectivamente será:

$$K = \tan\left(\frac{AUFA}{2} + 45º\right) = 3.6772$$

$$\omega_{pi} = K \cdot \omega_{ci} = 115.52\ krad/s$$

$$\omega_{zi} = \frac{\omega_{ci}}{K} = 8.54\ krad/s$$

Y la ganancia del regulador será:

$$K_i = \frac{\omega_{ci}}{|Gid(\omega_{ci})| \cdot Ri \cdot Fm} \cdot \frac{1}{K} = 470.53$$

Finalmente se forma el regulador de corriente:

$$Gi(s) = \frac{5.436 \cdot 10^7 s + 4.644 \cdot 10^{11}}{8543 s^2 + 9.87 \cdot 10^8 s}$$

El diagrama de bode del regulador se representa en la figura 19, mientras que en la figura 20 se puede comprobar que el margen de fase del nuevo lazo de corriente coincide con los objetivos planteados.

Para poder implementar el regulador en el DSP, la función de transferencia se debe de discretizar. En este caso la transformación se realizará por el método "**Tustin**" a una frecuencia de muestreo de 50 kHz (comprobado previamente su funcionamiento en simulación). Este es el método recomendado por Matlab para compensadores.

$$\mathrm{Z_i(z)} = \frac{0.03204\, z^2 \,+\, 0.005044\, z \,-\, 0.027}{z^2 \,-\, 0.928\, z \,-\, 0.07203}$$

*siendo Zi el compensador de corriente en el plano Z.

Una vez discretizada, la función debe pasar a ecuación en diferencias:

$$\mathrm{y(n)} = 0.928\, \mathrm{y[n-1]} + 0.07203\, \mathrm{y[n-1]} + 0.03204\, \mathrm{u[n]} + 0.005044\, \mathrm{u[n-1]} - 0.027\, \mathrm{u[n-2]}$$

*siendo y, u, la salida y entrada del regulador, respectivamente.

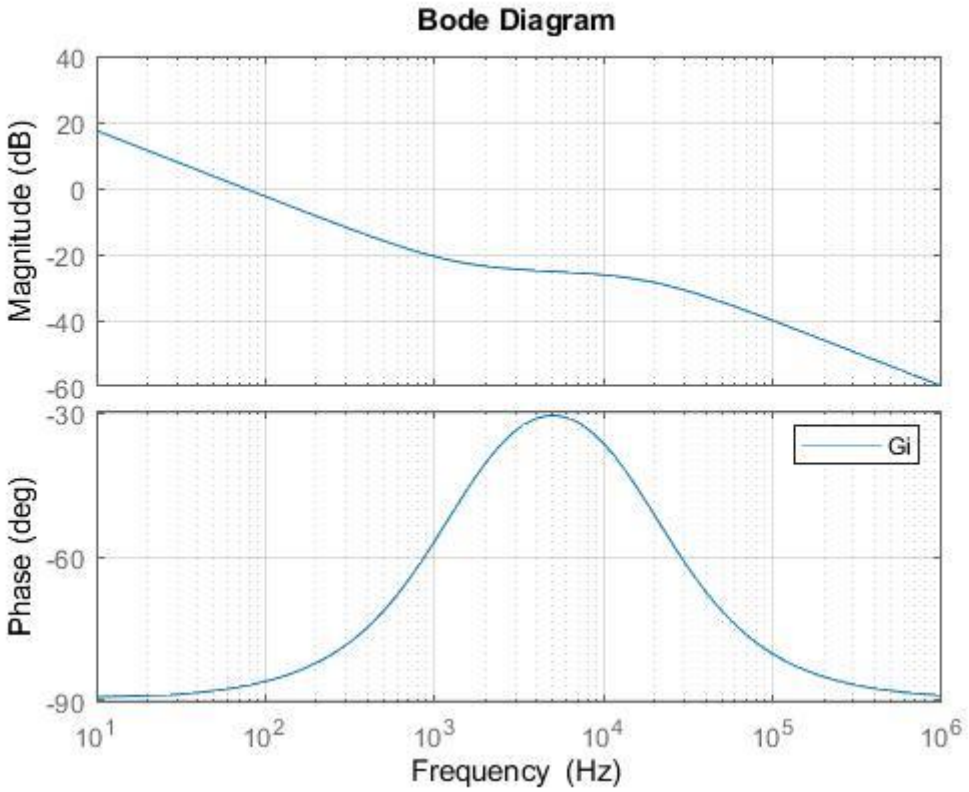


Figura 19. Respuesta en frecuencia del regulador de corriente.

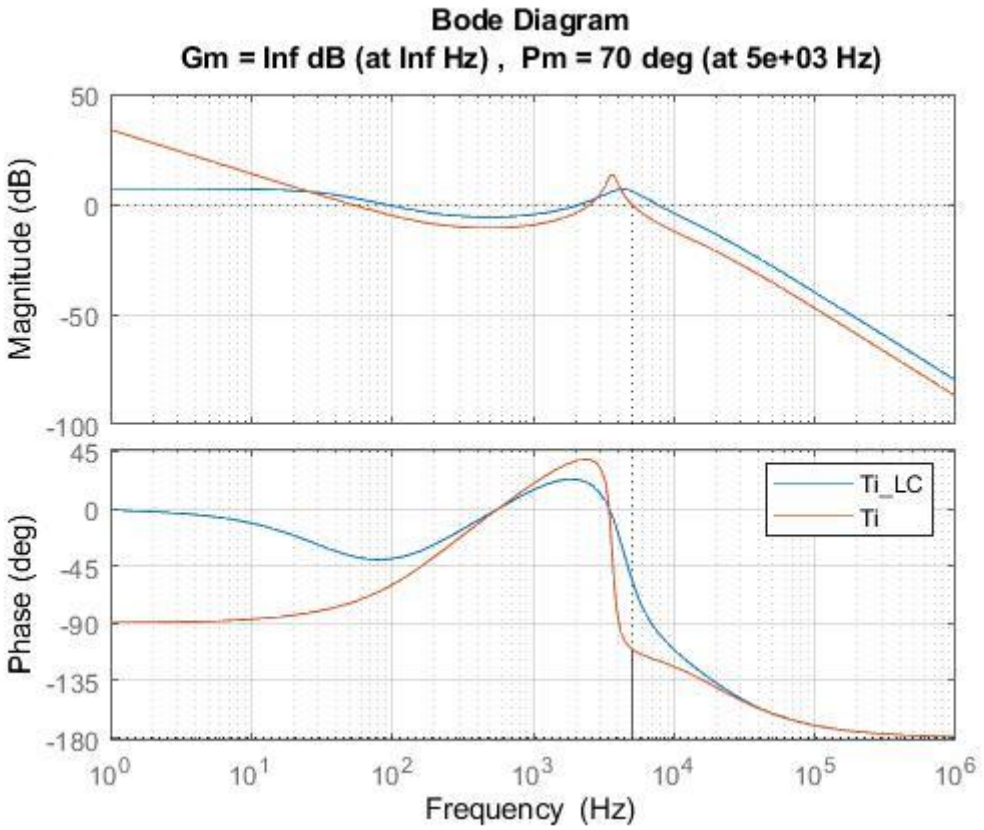


Figura 20. Ganancia del lazo de corriente con el regulador "Ti" y su respuesta en lazo cerrado "Ti_LC"

5.2.5. Función de transferencia de la tensión de salida a corriente de referencia

Antes iniciar con el lazo de tensión, es necesario reducir el diagrama de bloques del lazo de corriente "Ti_LC" a partir de la figura 14. El resultado se indica en la siguiente figura con su expresión correspondiente.

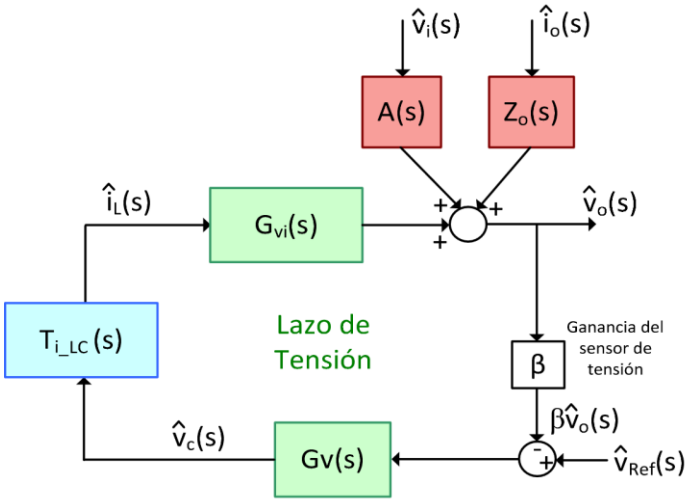


Figura 21. Diagramas de bloque reducido del esquema de control [5].

$$T_{i\_LC}(s) = \frac{\hat{\imath}_L(s)}{\hat{\imath}_{L\_ref}(s)} = \frac{F_m \cdot G_{id}(s) \cdot Gi(s)}{1 + R_i \cdot F_m \cdot G_{id}(s) \cdot Gi(s)} = \frac{1}{R_i} \cdot \frac{T_i(s)}{1 + T_i(s)}$$

Ahora se obtendrá la función de transferencia de la tensión de salida a la corriente por la bobina "Gvi":

$$G_{vi}(s) = \frac{\hat{v}_o(s)}{\hat{\imath}_L(s)}\Big|_{\hat{v}i=\hat{\imath}o=0} = \frac{R_{Load}(1 + s \cdot R_c C)}{1 + s \cdot (R_{Load} + R_{c)C}}$$

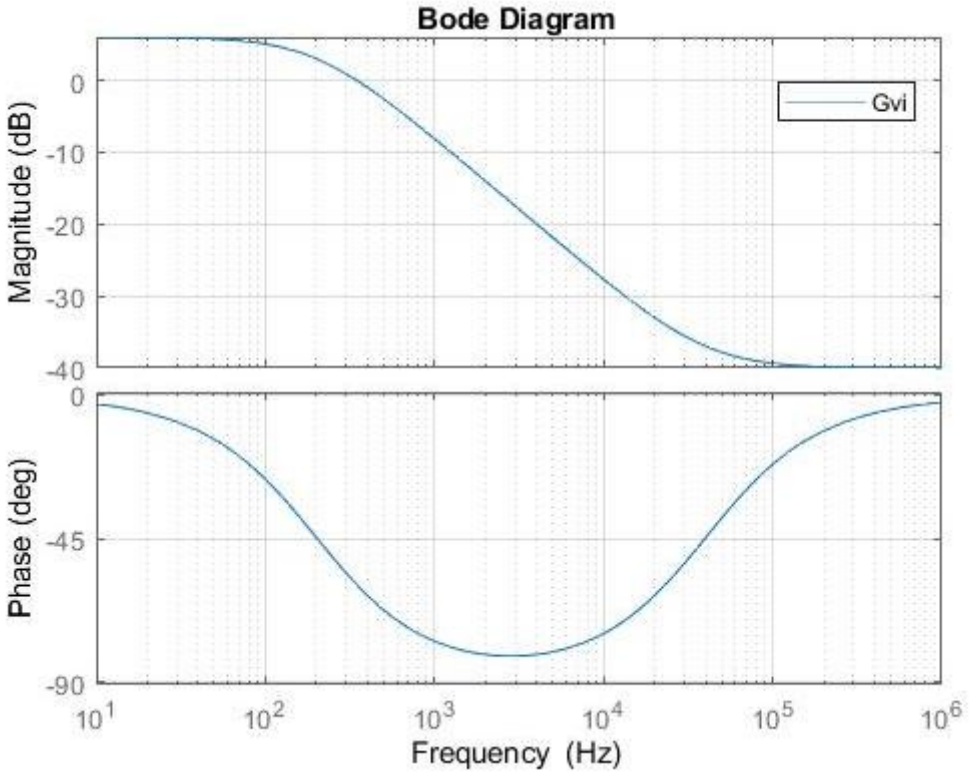


Figura 22. Respuesta en frecuencia de la expresión de la tensión de salida a la corriente por la bobina

A continuación, se calculará la función de transferencia de la tensión de salida a corriente de referencia "Gvc" (representada en frecuencia en la figura 23):

$$G_{vc}(s) = \frac{\hat{v}_o(s)}{\hat{v}_c(s)}\Big|_{\hat{v}i=\hat{\imath}o=0} = T_{i_{LC}}(s) \cdot G_{vi}(s) = \frac{1}{R_i} \cdot \frac{T_i(s)}{1 + T_i(s)} \cdot \frac{R_{Load}(1 + s \cdot R_c C)}{1 + s \cdot (R_{Load} + R_{c)C}}$$

La ganancia del lazo de tensión "Tv(s)" partirá de la siguiente ecuación:

$$T_v(s) = \beta \cdot G_{vc}(s) \cdot G_v(s)$$

*siendo Gv el compensador del lazo de tensión.

Y finalmente simplificando el lazo tensión, se consigue la función de transferencia del control lazo cerrado de la tensión de salida:

$$T_{v\_LC}(s) = \frac{\hat{v}_o(s)}{\hat{v}_{Ref}(s)}\Big|_{\hat{v}i=\hat{\imath}o=0}$$

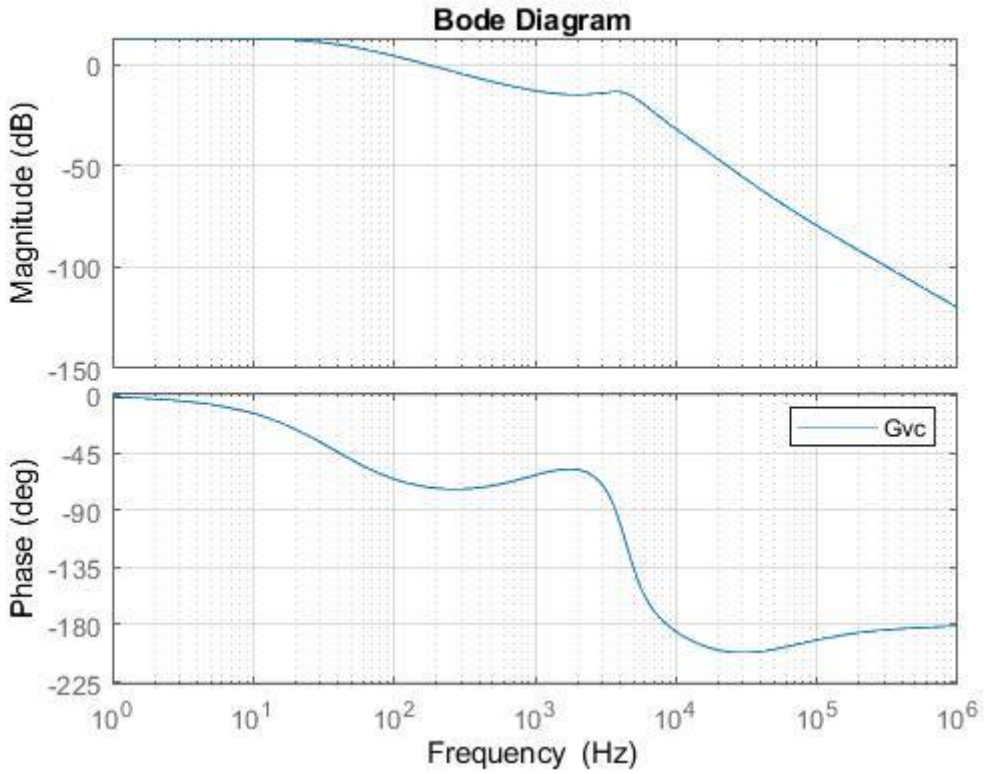


Figura 23. Respuesta en frecuencia de la expresión de la tensión de salida a la corriente de referencia.

6.2.6. Diseño del lazo de tensión

Los objetivos planteados para el lazo de tensión son los siguientes:

Objetivos	
Fcv	370 Hz
MF	70º

Tabla 3. Objetivos de diseño del lazo de tensión.

Para empezar, se obtendrá la fase actual a la frecuencia de cruce:

$$\arg\left(G_{vd}(\omega_{cv}\right) = -72.83\ º$$

*siendo **wcv** la frecuencia de cruce del lazo de tensión en rad/s

Y conociendo la fase objetivo:

$$Fase\ objetivo = -180º + MF = -110\ º$$

Se comprueba que es necesario un aumento de fase:

$$AUFA = MF - \arg(G_{vd}(j\omega_{cv})) - 90º = 52.83º$$

Entonces, como AUFA es menor que 90 º el compensador necesario será de **tipo 2**. Con lo cual, el factor K será:

$$K = \tan\left(\frac{AUFA}{2} + 45º\right) = 2.9743$$

Una vez calculado K se obtienen las frecuencias del polo y del cero del regulador:

$$\omega_{pv} = K \cdot \omega_{cv} = 6.91\ krad/s$$

$$\omega_{zv} = \frac{\omega_{cv}}{K} = 781.63\ rad/s$$

Y la ganancia del regulador Kv que modifica la ganancia lazo para conseguir 0 dB a la frecuencia de cruce:

$$K_v = \frac{\omega_{ci}}{|Gid(\omega_{ci})| \cdot Ri \cdot Fm} \cdot \frac{1}{K} = 3209$$

Finalmente se forma el regulador de tensión, cuya función de transferencia se muestra a continuación:

$$Gv = \frac{2.219 \cdot 10^7 s + 1.735 \cdot 10^{10}}{781.6 \cdot s^2 + 5.405 \cdot 10^6 s}$$

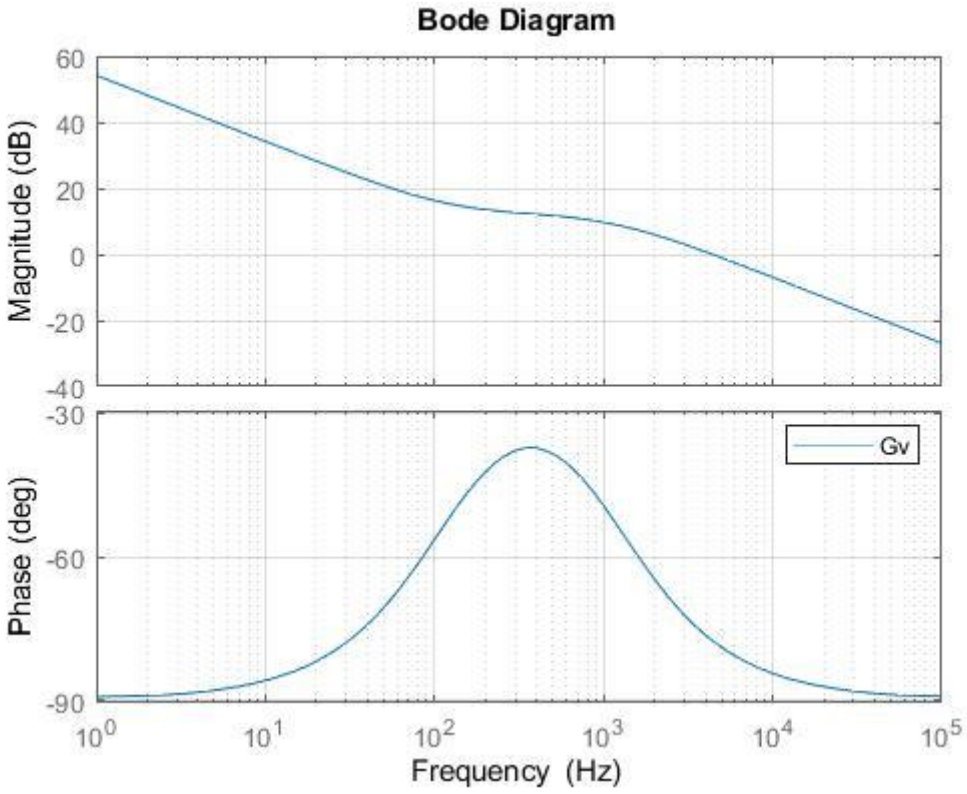


Figura 24. Respuesta en frecuencia del regulador de tensión.

En la siguiente figura se puede comprobar que el margen de fase del lazo de tensión y su posición coinciden con los objetivos planteados.

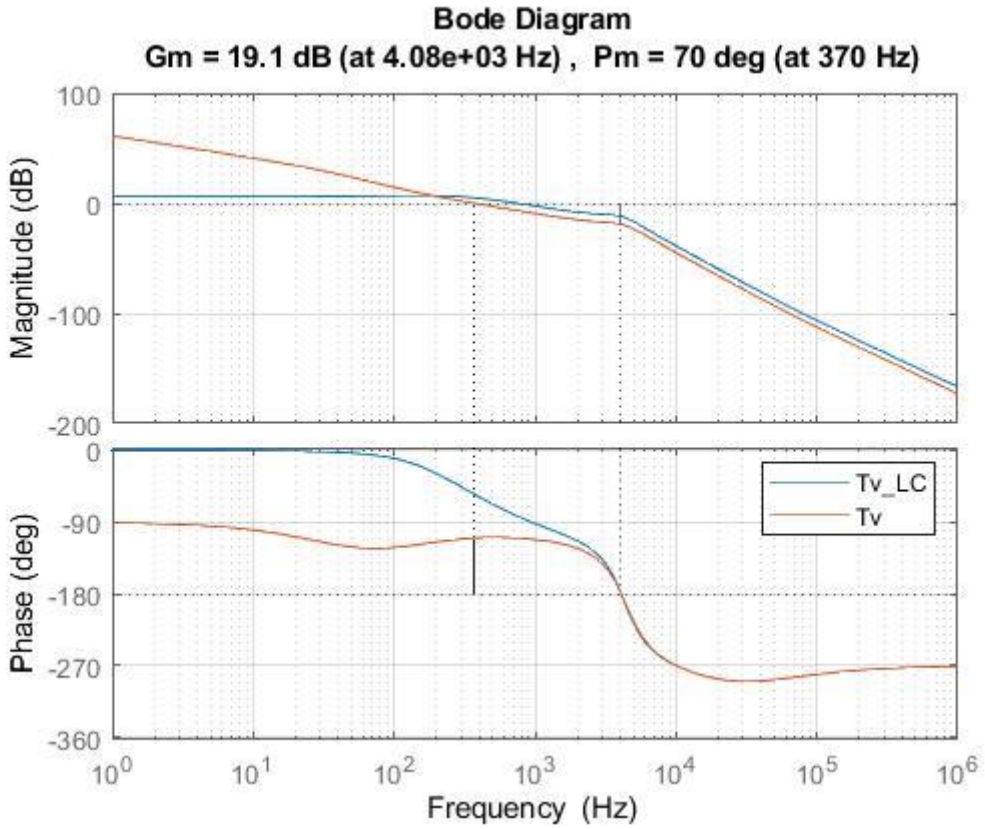


Figura 25. Ganancia del lazo de tensión con el regulador "Tv" y su respuesta en lazo cerrado "Tv_LC"

Como en el lazo de tensión, se procederá a discretizar la función de transferencia del compensador por el método “Tustin”, pero a una frecuencia de muestreo de 25 kHz:

$$Z_v(z) = \frac{0.5066\,z^2 + 0.0156\,z - 0.491}{z^2 - 1.757\,z + 0.757}$$

*siendo Zi el compensador de corriente en el plano Z.

Y la expresión del compensador en ecuación en diferencias:

$$\mathrm{y(n)} = 1.757\,\mathrm{y[n-1]} - 0.757\,\mathrm{y[n-1]} + 0.5066\,\mathrm{u[n]} + 0.0156\,\mathrm{u[n-1]} - 0.491\,\mathrm{u[n-2]}$$

*siendo y, u, la salida y entrada del regulador, respectivamente.

6.3. Simulación del sistema en Matlab

6.3.1. Circuito en Simulink

La simulación se ha diseñado bajo el entorno de programación visual “Simulink”, con el programa Matlab. Para el modelado de los componentes electrónicos se ha utilizado la librería “SimScape”, específicamente “**Simscape Electrical**”. Para simplificar la simulación del circuito, se ha optado por utilizar el esquema de un Buck Asíncrono. Asimismo, se incluye el formato de resistencia variable del convertidor “BOOSTXL-BUCKCONV” en la simulación para observar mejor los cambios de corriente por la bobina. En la figura 26 se muestra el circuito del Buck simulado.

El circuito de control se compone de: un arranque suave, un lazo de corriente, un lazo de tensión y un generador PWM. El lazo de tensión se forma a partir de un restador para la señal de referencia y la tensión de salida, y la ecuación discretizada del compensador de tensión. El lazo de corriente también se compone de un restador, comparando la corriente de referencia generada por el lazo de tensión y la corriente por la bobina. A continuación, la señal circula por el compensador de corriente, donde la acción de control resultante se restringe con un limitador. Finalmente, la señal PWM se crea a partir de la comparación de la señal de control con una señal triangular. En la figura 27, se muestra el circuito de control simulado.

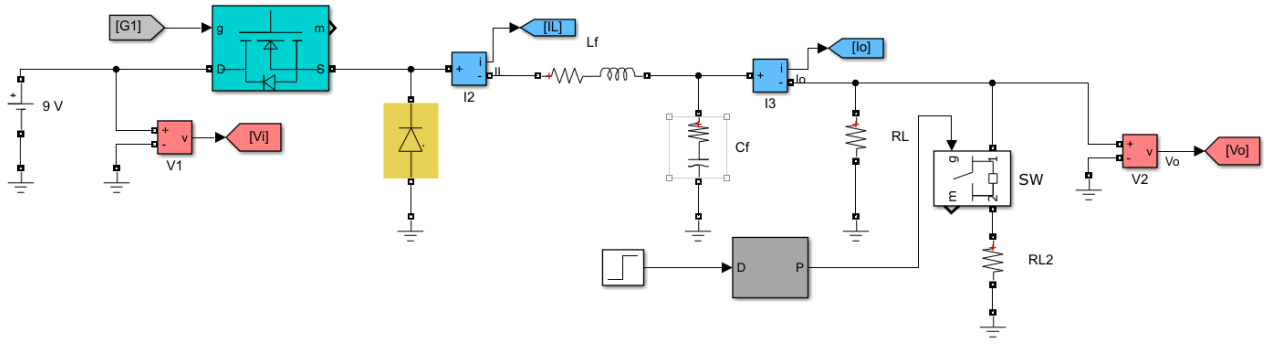


Figura 26. Circuito de potencia en Simulink.

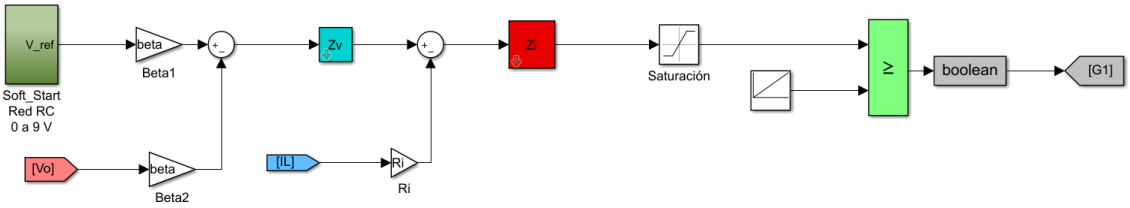


Figura 27. Circuito de control en Simulink.

6.3.2. Resultados de Simulink

Los resultados de la simulación (figura 28) muestran una rápida respuesta del lazo de control. El seguimiento de la tensión de salida "**Vo**" hacia la tensión de referencia de 3 V se realiza en torno a los 0.025 segundos, sin sobreoscilación gracias al alto valor de los márgenes de fase marcados durante el diseño de los compensadores. En la corriente de salida "**Io**" presenta un rizado de 0.75 A aproximadamente. Esto es debido a la conmutación de la resistencia de carga "**RL2**", que simula una resistencia variable. Al sustituir esta resistencia por un valor fijo, eliminando el transistor, Io presenta un rizado mínimo. En una aplicación real del convertidor, las cargas son fijas y la figura 29 representaría una aproximación fiel a la realidad.

El control diseñado presenta grandes limitaciones debido al convertidor. La poca tensión de entrada (9 V) induce al sistema a entrar fácilmente en conducción discontinua. En la figura 30, empleando un 60 % de ciclo de trabajo en el transistor de carga, el reductor entra casi en límite entre DCM y CCM.

En la figura 31, se puede observar que la corriente por la bobina "**IL**" alcanza valores nulos. El convertidor no tiene potencia suficiente para mantener la conducción continua del sistema. Con cargas resistivas mayores o iguales al equivalente del 60% en el ciclo de trabajo que controla el transistor en serie con RL2, el sistema entra en modo conducción discontinua.

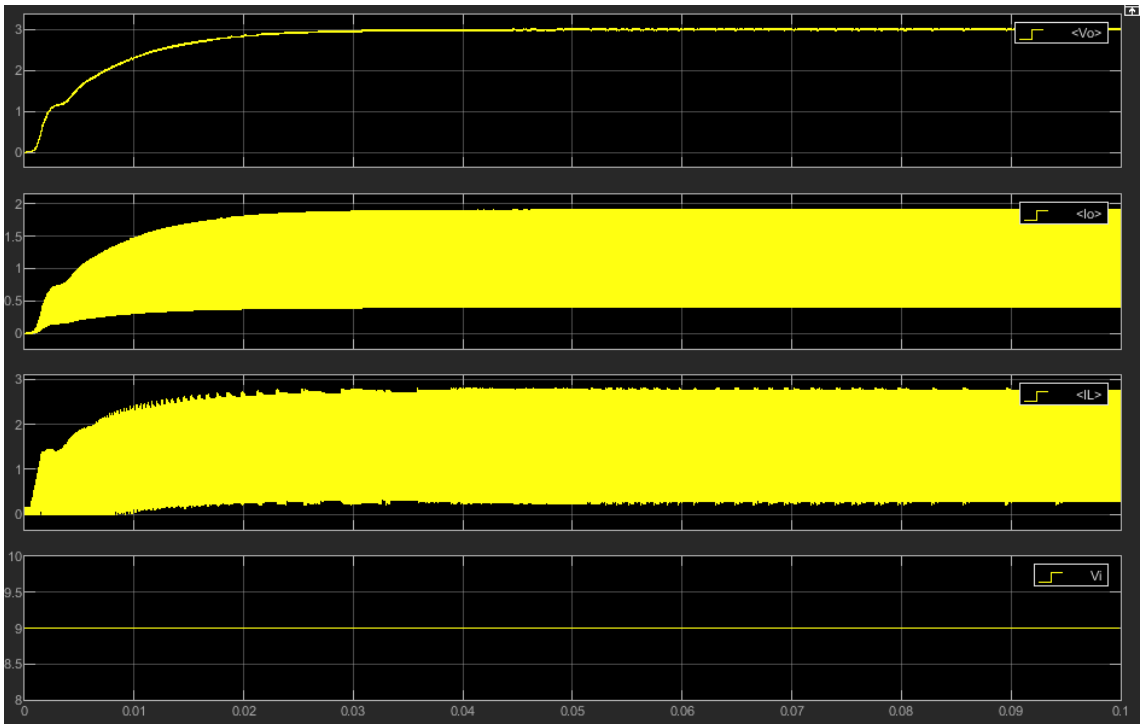


Figura 28. Medidas de la simulación del Buck con un ciclo de trabajo de 75% en el transistor de carga.

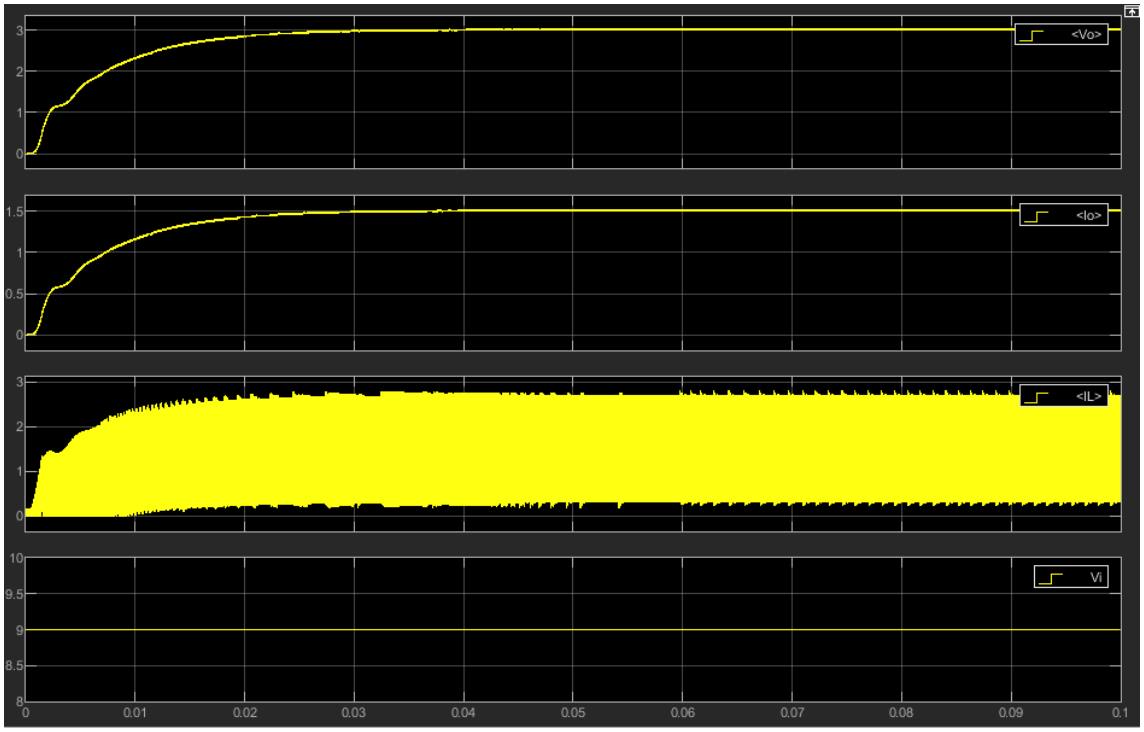


Figura 29. Medidas del convertidor Buck con una resistencia de carga fija de 2 Ω.

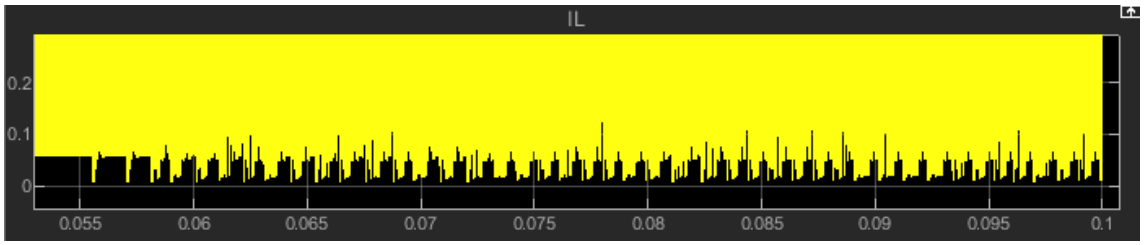


Figura 30. Límite a 0 de la corriente por la bobina en la simulación del Buck con un ciclo de trabajo de 60% en el transistor de carga con 3 V de referencia.

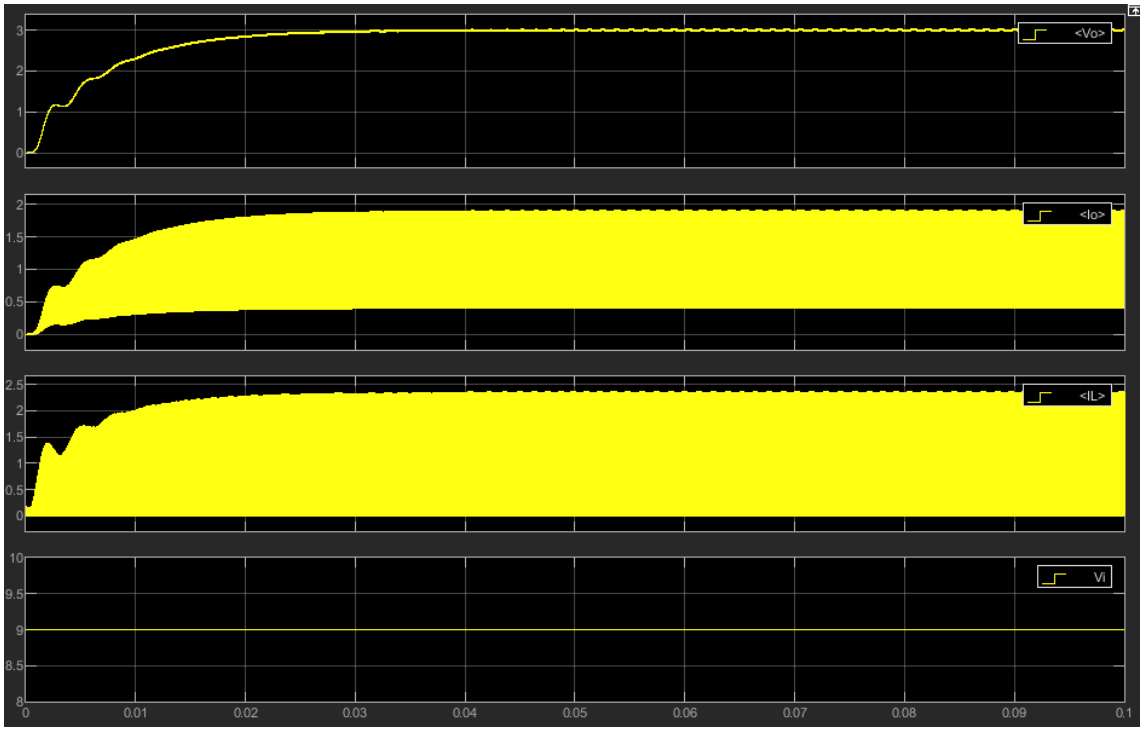


Figura 31.Medidas de la simulación del Buck con un ciclo de trabajo de 75% en el transistor de carga.

En la siguiente figura se muestra la corriente por la bobina duplicando la frecuencia de conmutación del interruptor. El control mejora para cargas mayores, pero las pérdidas por conmutación también se duplican. El convertidor al ser de poca potencia, esta energía transformada en calor supone una disminución considerable en la eficiencia del convertidor por lo que se descarta esta opción. Durante las pruebas, las mejoras se limitan hasta el 30 % en el transistor de carga. A mayores cargas, el sistema entra en DCM.

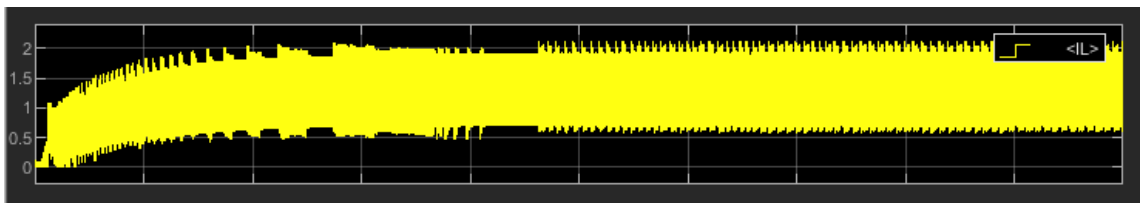


Figura 32. Corriente por la bobina con la frecuencia de conmutación a 400 kHz y un ciclo de trabajo de 60 % en el transistor de carga.

La figura 33 muestra la respuesta del sistema diseñado frente a un cambio de carga. La corriente de salida presenta alteraciones mínimas, por lo que el control funciona de manera adecuada. El restablecimiento de la tensión de salida se realiza en 1ms, que indica un rápido seguimiento del

control. La variación de tensión es de aproximadamente de 100 mV, menos del 5 % de la tensión de referencia marcada.

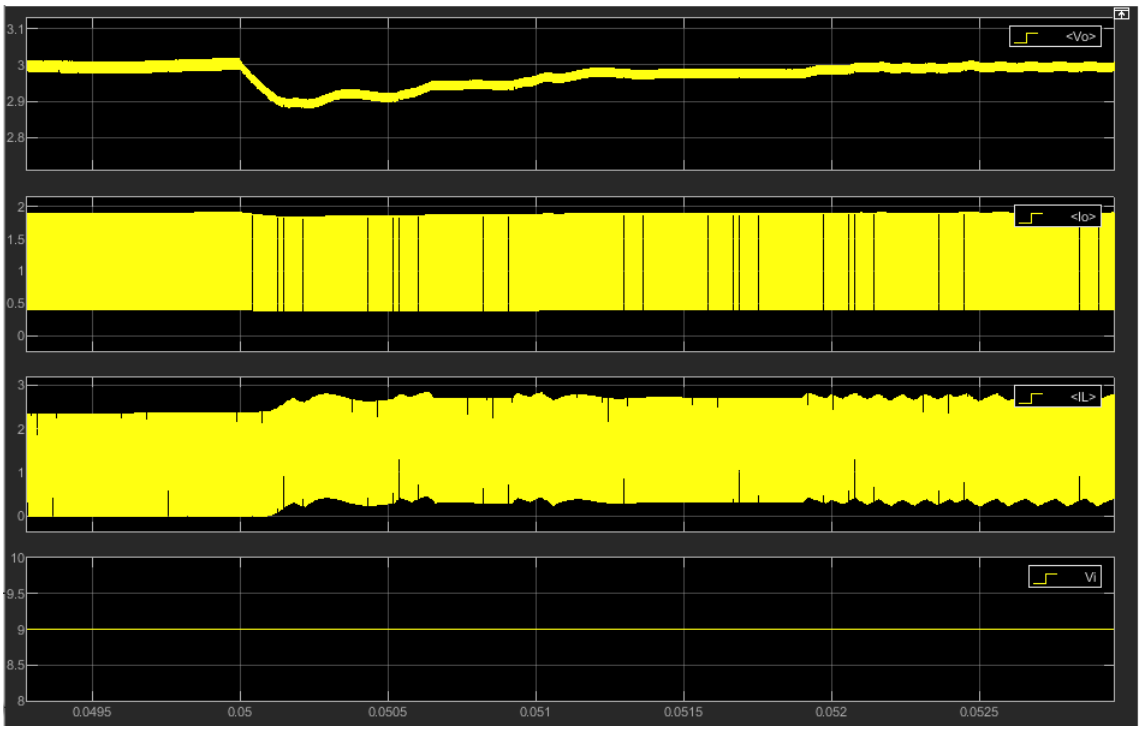


Figura 33. Respuesta del control frente a un cambio de la resistencia de carga.

Cabe destacar, que el ciclo de trabajo del transistor está limitado a un rango entre un 5% y 75% (figura 34). Eliminando estas limitaciones, la respuesta podría ser más rápida y precisa. Pero en convertidores comerciales se limitan el ciclo de trabajo para mantener la durabilidad de los componentes, además de favorecer el cumplimiento de las garantías. El proyecto se ha realizado con la intención de aproximar los resultados a las fuentes conmutadas del mercado.

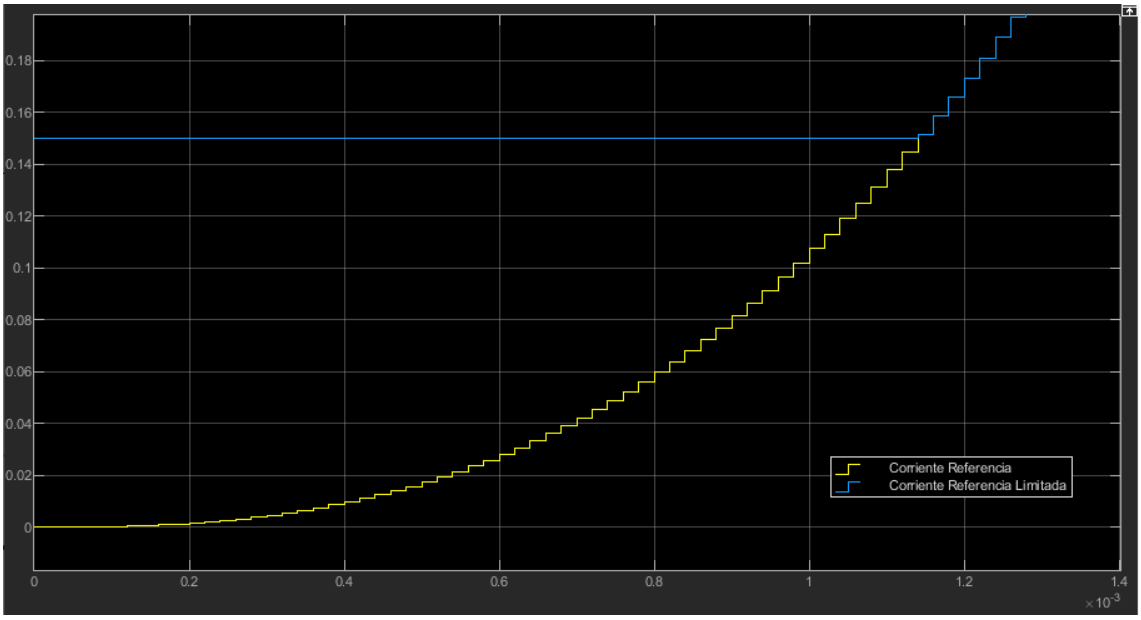


Figura 34. Señal de la corriente de referencia respecto la corriente de referencia limitada.

7. Implementación del control en el DSP

7.1. Características del programa a desarrollar

En primer lugar, se van a especificar las características básicas que debe tener el software a implementar en el DSP:

- Selección de modo de control: lazo abierto, modo tensión y modo corriente media.

- Capacidad de implementar reguladores de tipo I, II y III.

- Posibilidad de modificar la tensión de referencia, resistencia de carga, frecuencia de conmutación y las constantes de los compensadores.

- Seguimiento de la tensión y corriente del convertidor

- Funciones de los compensadores en RAM para disminuir la latencia

- Comprobación de las tramas en las comunicaciones por serie.

7.2. Conexiones entre el DSP y el convertidor Buck

La disposición de los pines tanto del DSP como del convertidor son compatibles y permite la conexión directa entre los dos dispositivos. Para aplicar el control, primero será necesario conocer la función de los pines del Buck (figura 35) que corresponden con los del microcontrolador (figura 36 y 37), para determinar las configuraciones pertinentes.

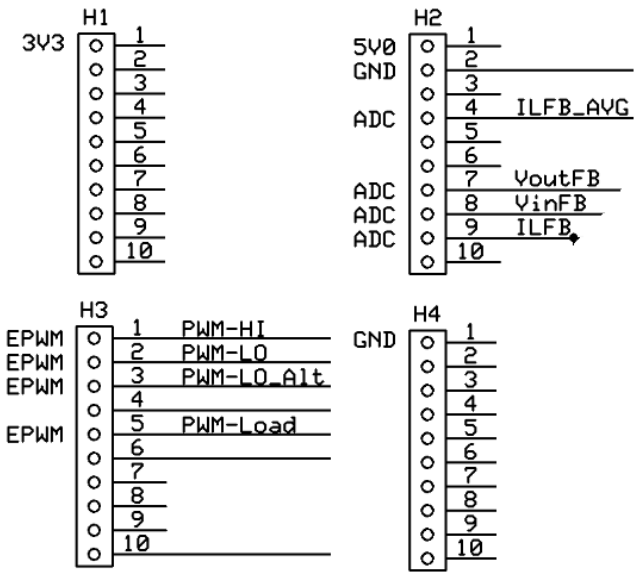


Figura 35. Disposición de los pines del convertidor "BOOSTXL-BUCKCONV"

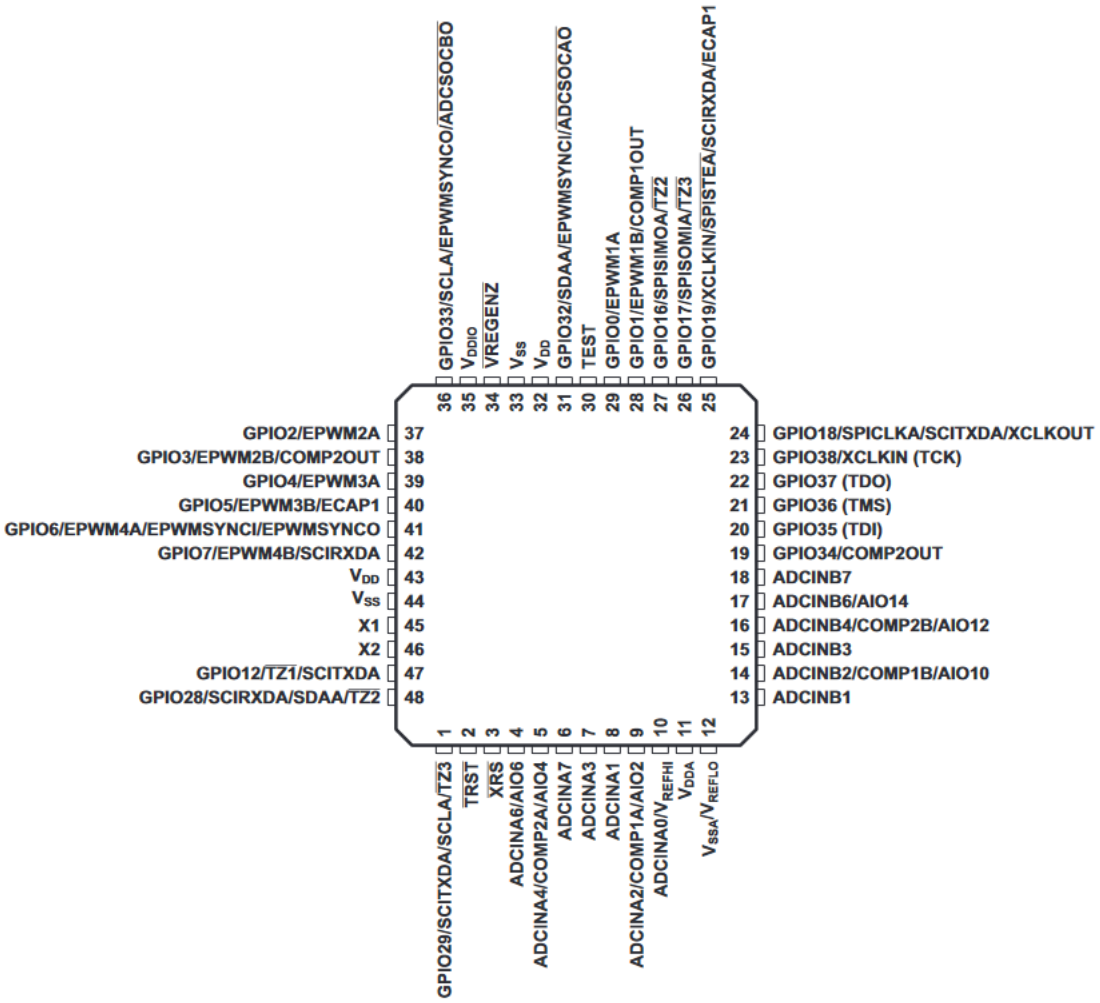


Figura 36. Diagrama de los pines del microcontrolador TMS320F28027F [7].

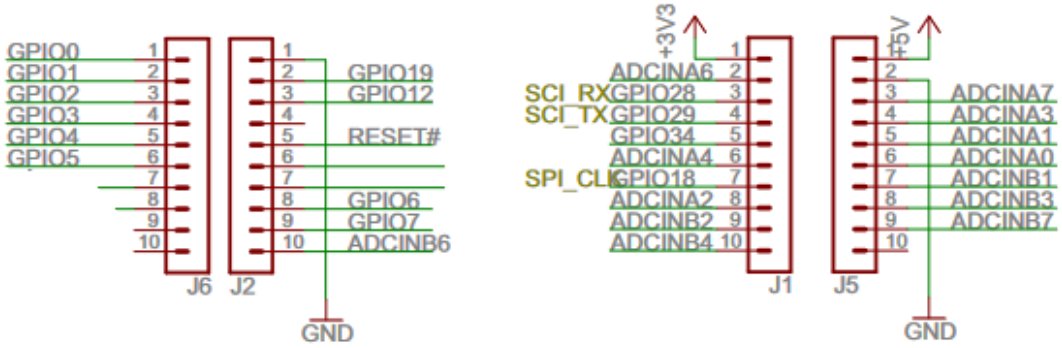


Figura 37. Diagrama de los pines de la placa de desarrollo LAUNCHXL-F28027F.

Pin	Columna Buck	Columna LaunchPad	Función
7	H2	J5	ADCINB1
8	H2	J5	ADCINB3
9	H2	J5	ADCINB5
1	H3	J6	ePWM1A
2	H3	J6	ePWM1B
5	H3	J6	ePWM3A

Tabla 4. Equivalencias de los pines entre el convertidor y la placa de desarrollo, y su función.

Como se puede observar en la tabla anterior, determinar la disposición de los terminales de la placa de desarrollo permite configurar el módulo del microcontrolador y asociarlo al pin físico correspondiente.

7.3. Configuraciones

7.3.1. Entradas y salidas de propósito general

Los registros GPXMUX del multiplexor de los GPIO (General Purpose Input/Ouput) se utilizan para seleccionar la función de un pin físico compartido. En este proyecto se requiere el uso del ePWM1A, ePWM 1B, ePWM3A. Por ello, se configurará el registro según la figura siguiente para habilitar las funciones pertinentes.

GPAMUX1 Register Bits	Default at Reset Primary I/O Function (GPAMUX1 bits = 00)	Peripheral Selection 1 (GPAMUX1 bits = 01)	Peripheral Selection 2 (GPAMUX1 bits = 10)	Peripheral Selection 3 (GPAMUX1 bits = 11)
1-0	GPIO0	EPWM1A (O)	Reserved(1)	Reserved(1)
3-2	GPIO1	EPWM1B (O)	Reserved	COMP1OUT (O)
5-4	GPIO2	EPWM2A (O)	Reserved	Reserved(1)
7-6	GPIO3	EPWM2B (O)	Reserved	COMP2OUT (O)
9-8	GPIO4	EPWM3A (O)	Reserved	Reserved(1)
11-10	GPIO5	EPWM3B (O)	Reserved	ECAP1 (I/O)
13-12	GPIO6	EPWM4A (O)	EPWMSYNCI (I)	EPWMSYNCO (O)
15-14	GPIO7	EPWM4B (O)	SCIRXDA (I)	Reserved
17-16	Reserved	Reserved	Reserved	Reserved
19-18	Reserved	Reserved	Reserved	Reserved
21-20	Reserved	Reserved	Reserved	Reserved
23-22	Reserved	Reserved	Reserved	Reserved
25-24	GPIO12	$\overline{TZ1}$ (I)	SCITXDA (O)	Reserved
27-26	Reserved	Reserved	Reserved	Reserved
29-28	Reserved	Reserved	Reserved	Reserved
31-30	Reserved	Reserved	Reserved	Reserved

Figura 38. Registro GPAMUX con las funciones a configurar [3].

7.3.2. PWM

La modulación por ancho de pulsos se configurará a 200 kHz, tanto para el ePWM1 como para el ePWM3, siguiendo los ejemplos presentados por el fabricante. La frecuencia del PWM "$\mathbf{f_{PWM}}$" está controlado por un registro de tiempo TBPRD y un modo de contador del tiempo. En este caso se utilizará el modo de contador "Up-Count Mode", que ofrece más resolución frente al "Up-Down Mode" y más sencillez frente al "Down-Count Mode". El conteo se realiza desde cero y aumenta hasta llegar al valor del registro determinado (TBPRD).

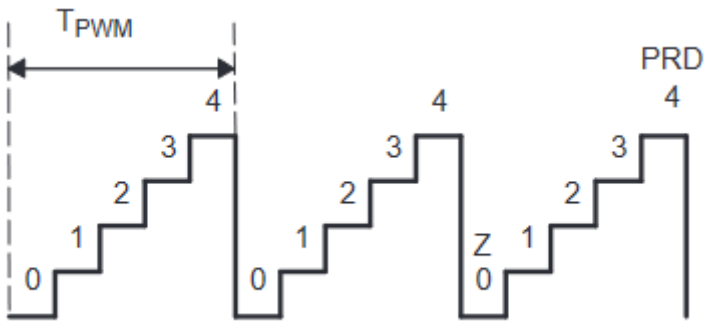


Figura 39. Up-Count Mode [3].

$$TBPRD = \frac{f_{TBCLK}}{f_{PWM}} - 1 = 299$$

*siendo f_{TBCLK} la frecuencia del reloj del microcontrolador, que es de 60 MHz

El control del periodo se realiza a través del submódulo “Action Qualifier”. Existe diversos bits del registro AQCTLA/AQCTLB que comparan el conteo a un determinado valor para generar una acción en la señal PWM. Como se observa en la siguiente figura, el dispositivo se ha configurado de manera que cuando el valor del registro TBPRD disminuye hasta cero, fija a modo alto el ePWMxA. En cambio, cuando aumenta hasta un valor determinado del registro CMPA, fija la señal de vuelta a modo bajo. Este registro controla el ciclo de trabajo “δ “a través de las cuentas del registro TBPRD.

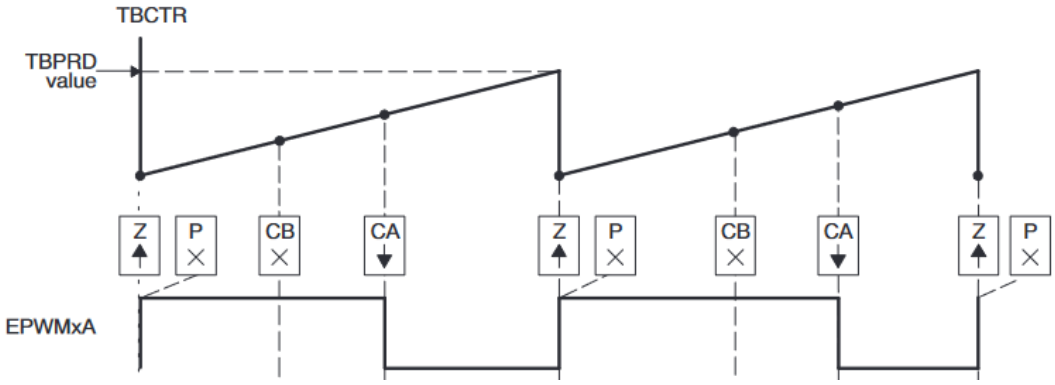


Figura 40. Configuración del PWM [3].

$$CMPA = \delta \cdot TBPRD$$

Las señales de control de los transistores del Buck síncrono corresponden con el EPWM1A y EPWM1B. Como pertenecen al mismo módulo PWM, permite emplear el submódulo “Dead-Band Generator” y generar una señal complementaria para el MOSFET que actúa como diodo. Además, este módulo, posibilita configurar un “dead-band” entre las dos señales para evitar la conducción simultánea de los transistores.

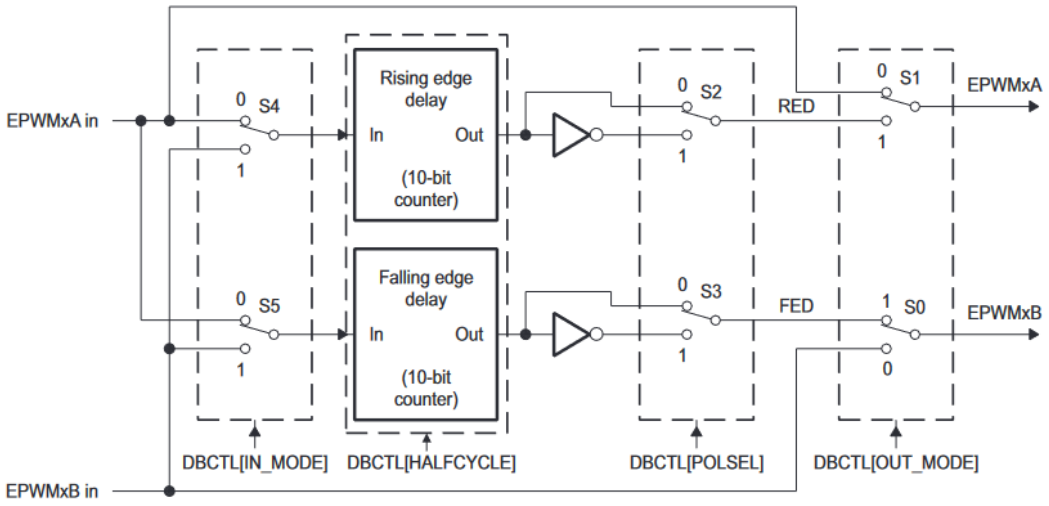


Figura 41. Opciones para la configuración del submódulo "Dead-Band Generator" [3].

Acorde a las opciones disponibles de la figura anterior, se configurará el submódulo de la siguiente manera:

S0	S1	S2	S3	S4	S5
1	1	0	1	0	0

Tabla 5. Configuracción del submódulo "Dead-Band Generator".

La señal resultante sería un “Rising Edge Delayed” para el ePWM1A y un “Active High Complementary” para el ePWM1A.

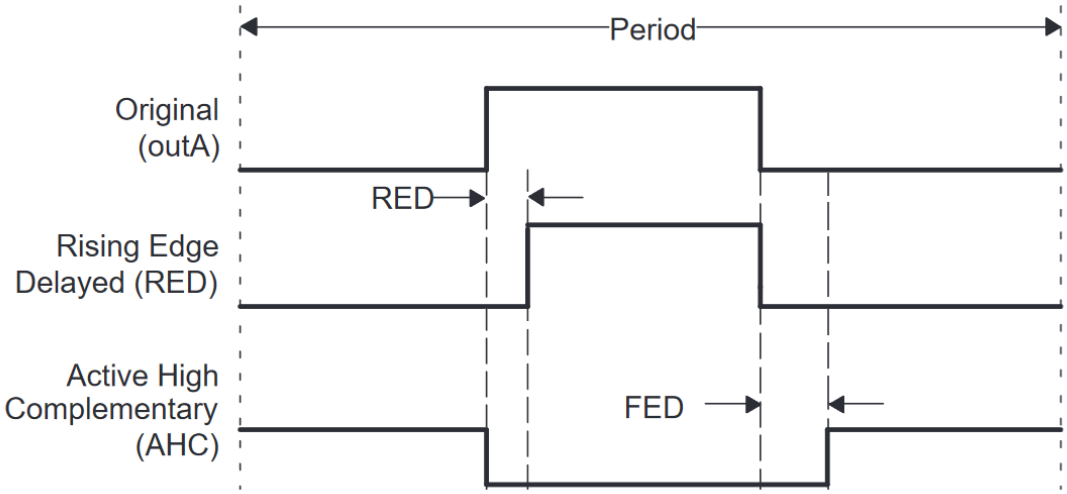


Figura 42. Formas de onda con el "dead-band" aplicado [3].

Para fijar el tiempo de desplazamiento RED y FED, se debe de fijar mediante los registros **DBRED** y **DFED**. Con el registro “**HALFCYCLE**” activado, permite aumentar la resolución al doblar la velocidad del reloj. Las expresiones quedarían de la siguiente manera:

$$RED = DBRED\ x\ T_{TBLCK}/2$$

$$FED = DBFED\ x\ T_{TBLCK}/2$$

*siendo T_{TBLCK} el periodo del reloj

La resolución del “deadband” viene correlacionado con el periodo del reloj “T_{TBLCK}”:

$$Resolución = \frac{T_{TBLCK}}{2} = 8.333\ ns$$

En la figura 43 y 44, se muestran las señales ePWM1A y ePWM1B (**high input** y **low input**, respectivamente) que alimenta la puerta de los MOSFETS. Se puede apreciar que las señales se cruzan y que la señal “high input” es más rápida. Esto indica que en dos puntos de cada ciclo del PWM, los dos transistores conducen y, por tanto, el sistema entrará en cortocircuito. Para solucionarlo se deberá de añadir un “deadband” con un tiempo correspondiente al toque por cero de la señal en conducción. A partir de las siguientes imágenes se puede obtener el tiempo necesario de desplazamiento. Aplicando este tiempo a las expresiones que controlan el desplazamiento, se configurará el microcontrolador de la siguiente manera:

$$DBRED = \frac{16.8\ ns}{T_{TBLCK}/2} = 2.01 \approx 2$$

$$DBFED = \frac{58\ ns}{T_{TBLCK}/2} = 6.96 \approx 7$$

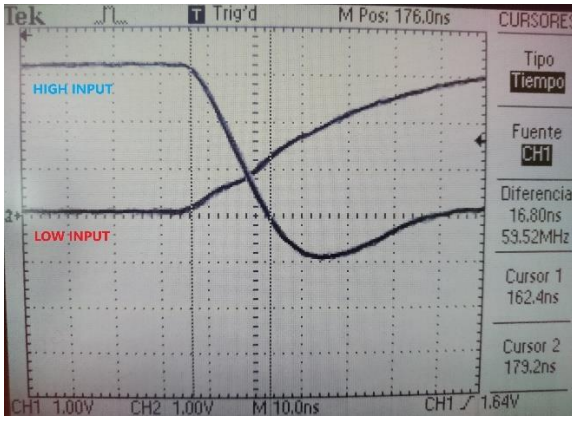


Figura 43. Señales ePWM1A y ePWM1B entrando en modo no conducción.

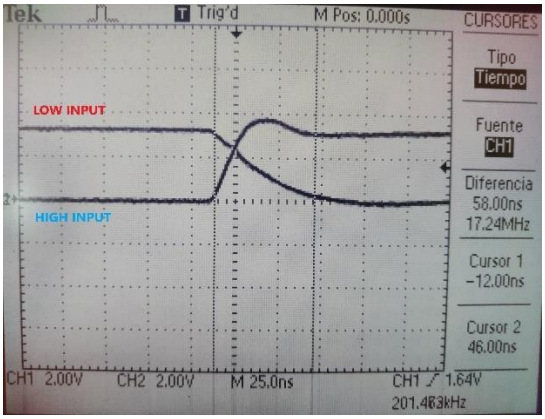


Figura 44. Señales ePWM1A y ePWM1B entrando en modo conducción

En la figura 45 y 46, se muestra las señales con el “deadband” aplicado. Las señales mostradas se cruzan cuando los valores de las tensiones se acercan a 0. Esto significa que no existe cortocircuito, es decir, que los dos transistores no conducen al mismo tiempo. Además, según la hoja de características del driver “LM5109B” incorporado, la tensión umbral para que la salida del driver sea modo bajo, la entrada no debe de superar los **800 mV,** como se marca en las figuras mencionadas anteriormente. Con un margen mínimo aproximado de **750 mV** es suficiente para compensar también el retraso que existe entre la salida high y low del controlador, que es de **2 ns** (ver hoja de características).

Asimismo, la señal “high input” presenta una **sobreoscilación** debida a la frecuencia de muestro fijada a 50 kHz, equivalente a 20 ns. Como la respuesta se realiza a esa velocidad, la reacción del control también está limitada a esa frecuencia.

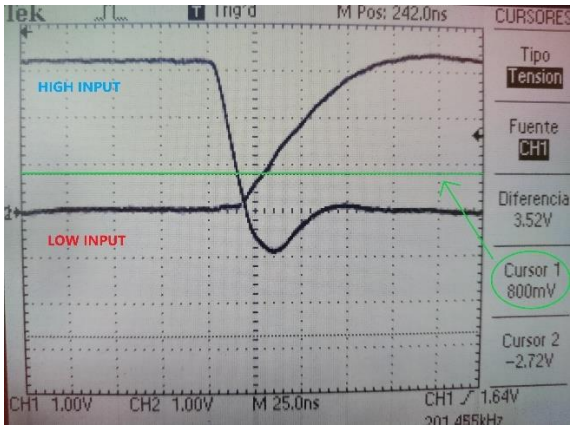


Figura 45. Señales ePWM1A y ePWM1B entrando en modo no conducción con "deaband".

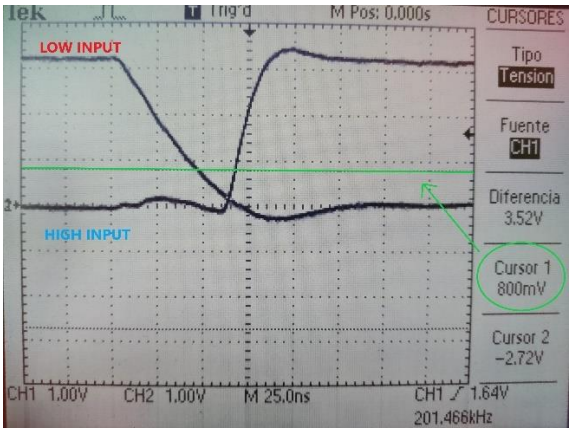


Figura 46. Señales ePWM1A y ePWM1B entrando en modo conducción con "deaband".

7.3.2. ADC

Las lecturas de los valores de tensión y corriente se realizarán a través del conversor analógico digital "ADC" a una frecuencia de 50 kHz. La conversión se inicia mediante el "Timer 0", que actúa también como contador. Al terminar la conversión, activa una interrupción para ejecutar las funciones necesarias.

El principio de funcionamiento del ADC está basado en SOCs (Start of Conversion). Este término indica las configuraciones para un canal: el origen que activa el inicio de la conversión, el canal a convertir del multiplexor y el tiempo del "Sample & Hold". Para este proyecto, el ADC iniciará con el "Timer 0", con un tiempo de adquisición de 10 ciclos de "Sample & Hold" y con los canales seleccionados como se indica en la siguiente figura:

SOC	Canal	ADC
0	9	ADCINB1
1	11	ADCINB3
2	13	ADCINB5

Tabla 6. Tabla con el canal por cada SOC y su ADC correspondiente.

El valor del SOC indica la prioridad, siendo 0 el más urgente. Al terminar la conversión del SOC2 activará una interrupción para ejecutar los compensadores del control modo corriente media.

Ahora se procederá a comprobar la velocidad del ADC, para asegurar que para la frecuencia de 50 kHz no se solape, siendo este la frecuencia de muestreo más rápida de entre los dos compensadores. Teniendo en cuenta que el ADC necesita 13 ciclos de reloj y el "Sample & Hold" es de 10 ciclos, el tiempo necesario será:

$$Tiempo = \frac{1}{60\ MHz} \cdot (13 + 10) = 383.33\ ns$$

Observando el resultado, se comprueba que, con una frecuencia de muestreo de 50 kHz, que equivale a 20 µS, es suficiente tiempo para procesar las lecturas.

7.3.3. Comunicación en serie

Las comunicaciones por el puerto serie emplean el formato NRZ (non-return-to-zero). En este caso dentro de este formato se ha configurado la transmisión de datos con: un bit de parada, sin bit de paridad y 8 bits de datos.

7.4. Funcionamiento del programa

El sistema (figura 47) comienza con la inicialización de las variables y funciones. Se carga las funciones más cruciales dentro de la memoria RAM como pueden ser la de los compensadores e interrupciones para aumentar la velocidad de acceso. A continuación, se inicializa los registros para el control de la memoria FLASH. Como el programa excede el límite del tamaño de la RAM, se debe de resituar gran parte del código en la FLASH. Aquí se incluye las configuraciones de los módulos necesarios para el control. Después, se inicializa los registros del sistema, es decir, la inicialización del sistema y se deshabilita las interrupciones por si existiese un programa cargado previamente. Además, se configura e inicializa todos los módulos necesarios: SCI, EPWM, ADC, Timer 0 (se reinicia el contador también) e interrupciones. Finalmente entra en un bucle infinito.

La función del bucle infinito (figura 48) envía cada segundo las lecturas de corriente por la bobina y de tensión tanto de entrada como de salida.

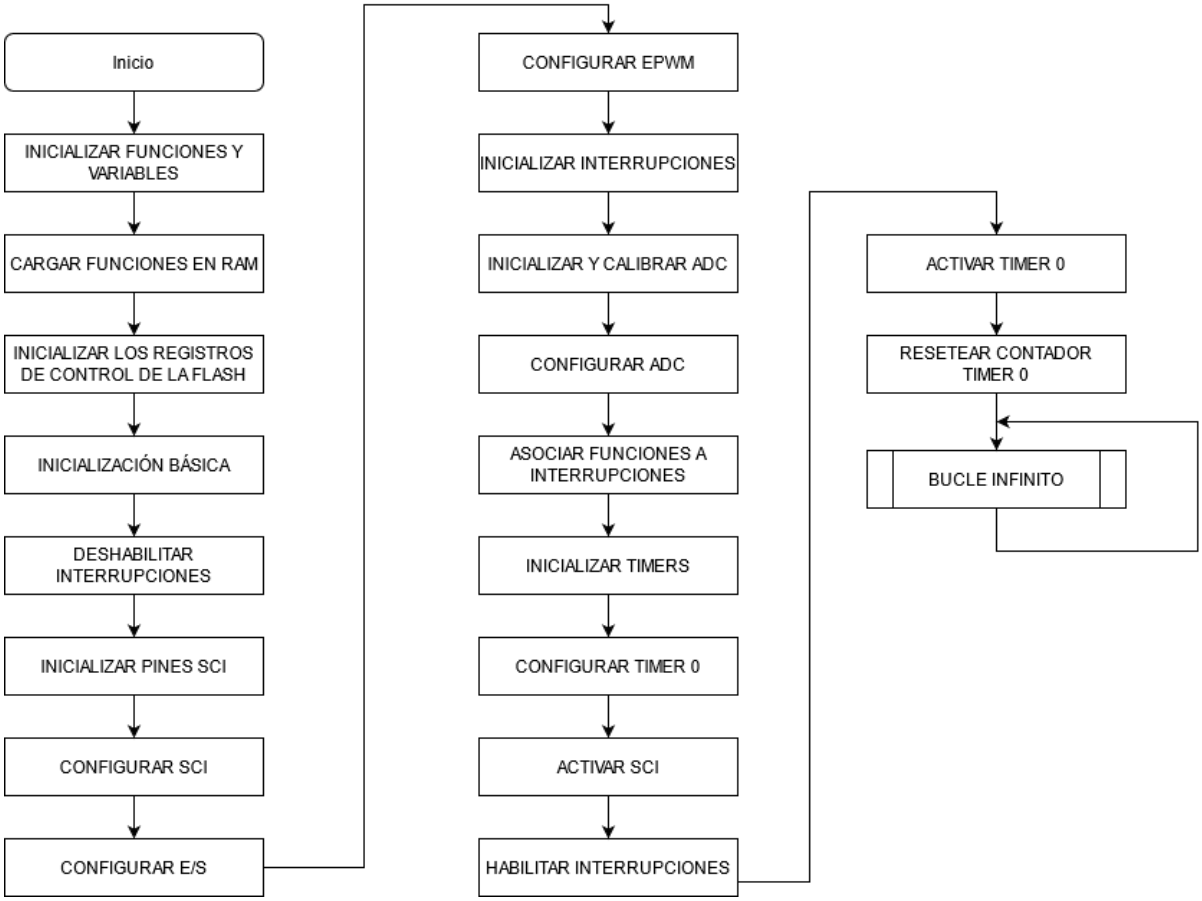


Figura 47. Flujograma del programa principal.

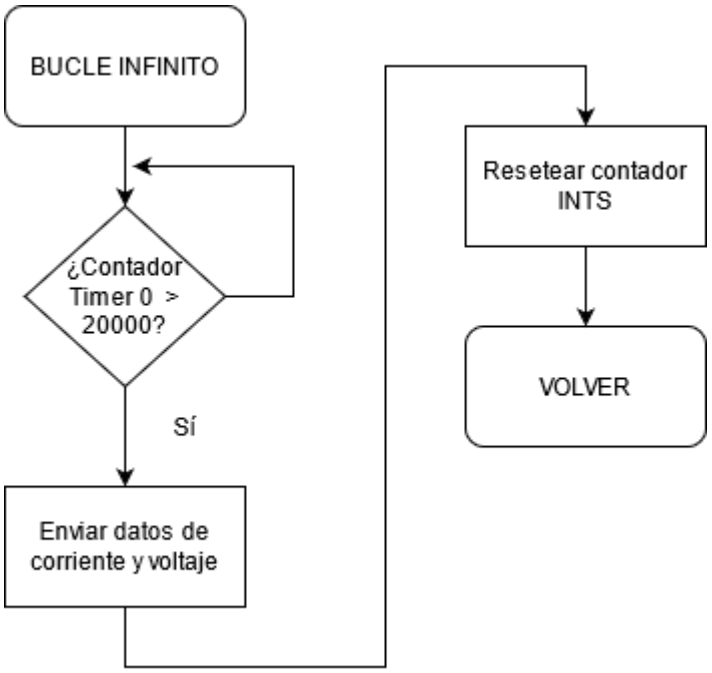


Figura 48. Flujograma del bucle infinito.

La interrupción del timer 0 (figura 49) se conforma con un contador y su reinicio.

La interrupción del ADC (figura 50) comienza cuando se han terminado las lecturas de corriente y tensión. Los datos se guardan en variables y se suma uno al contador de medias. La variable de la corriente es un vector de 10 posiciones donde el contador de medias indica la posición a guardar. A continuación, se realiza la operación de la media y dependiendo del control seleccionado realizará una acción:

- Lazo abierto: modifica simplemente el ciclo de trabajo.
- Modo tensión: realiza la función del regulador del control modo tensión.
- Modo corriente media: se implementa el regulador de corriente calculado anteriormente, y cada 2 interrupciones, también realiza la función del regulador de tensión. Al alcanzar el límite el contador de medias vuelve a su valor inicial de 0.

La función termina con el reset de la propia interrupción.

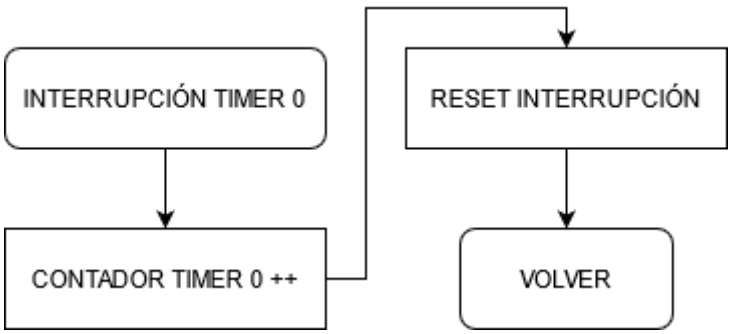


Figura 49. Interrupción del Timer 0.

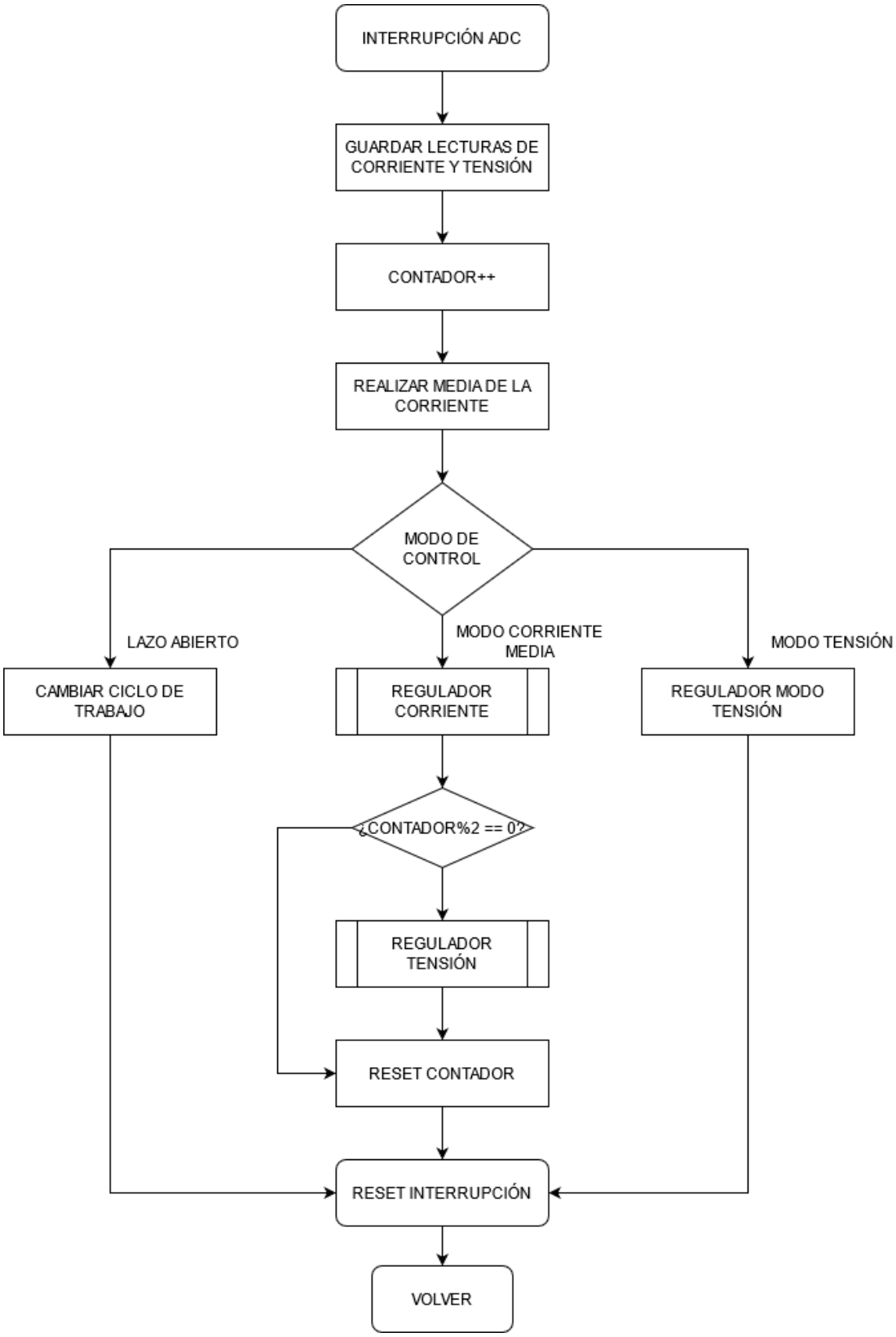


Figura 50. Flujograma de la interrupción del ADC

La función del regulador del corriente consiste en implementar el compensador de corriente dentro del programa. Inicia con el guardado de datos de corriente comparadas y acciones de control anteriores. Después se compara la corriente de referencia con la corriente por la bobina. Se aplica la ecuación en diferencias del compensador calculado y transforma la acción de control a ciclo de trabajo. Hay que tener en cuenta, que en el microcontrolador el PWM está integrado y se controla a través del ciclo de trabajo. El resultado obtenido se limita hasta un 45% porque existe un problema con la lectura de las corrientes. Cuando el transistor supera un ciclo de trabajo del 50%, los datos de corriente oscilan a valores cercanos al 0. Finalmente se aplica el ciclo de trabajo obtenido.

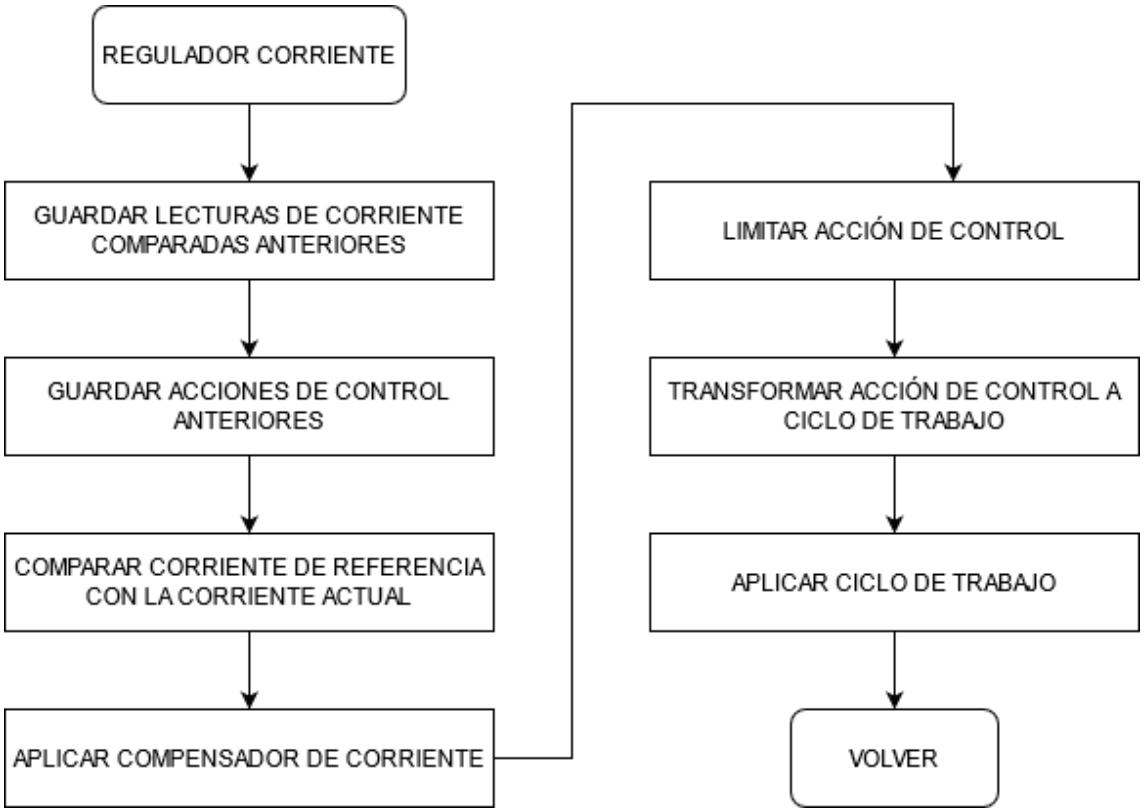


Figura 51. Flujograma de la función del regulador de corriente.

La función del regulador de tensión se basa en la implementación del regulador de tensión calculado para el control modo corriente. Comienza con el guardado de las lecturas de tensión y corrientes de referencia anteriores para la implementación posterior en el compensador. A continuación, se compara la tensión de referencia con la tensión actual de salida. Más tarde se aplica el compensador de tensión mediante la ecuación en diferencias calculada en los apartados anteriores.

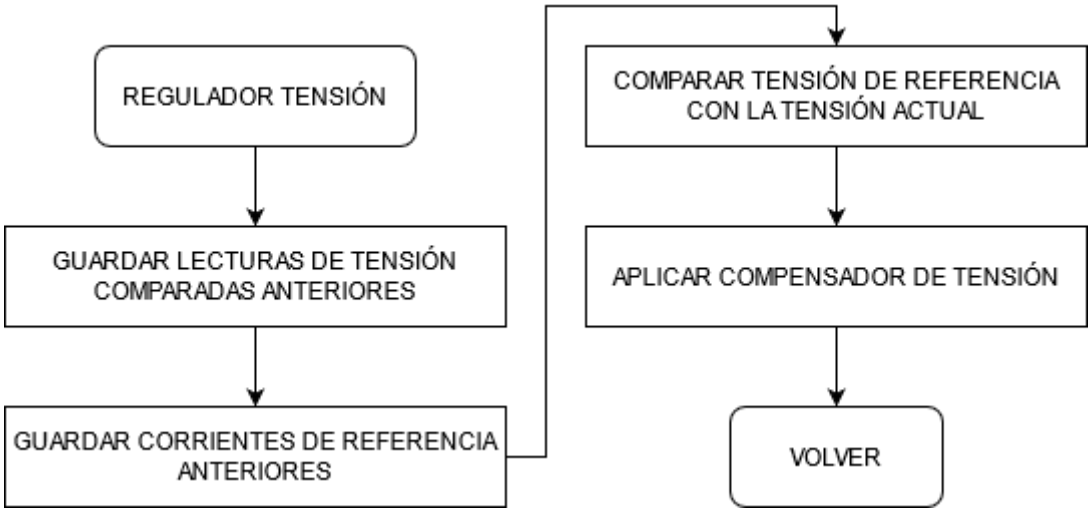


Figura 52. Flujograma de la función del regulador de tensión.

La interrupción del SCI inicia a partir de la recepción de 4 bytes por el puerto serie. Estos datos se guardan y comienza la comprobación de la trama. Si es correcta, comprueba si es una trama numérica, en caso contrario, es una trama de control. Las tramas de control indican a partir del parámetro, el tipo de dato que se va a recibir y modificar, y luego el parámetro, cuál exactamente. En la posterior interrupción se obtendrá una trama numérica, posiblemente incompleta debido a que las tramas son de 4 bytes. Cuando se terminan de recibir todas las cifras se modifica el dato indicado por la trama de control anterior.

En la implementación del código se ha realizado una máquina de estados para cada variable. Los comandos se asocian a un valor numérico de una cifra que se multiplica por 10, y a continuación, se suman el valor del parámetro. Este resultado indica el número de estado, es decir, indica el dato a modificar.

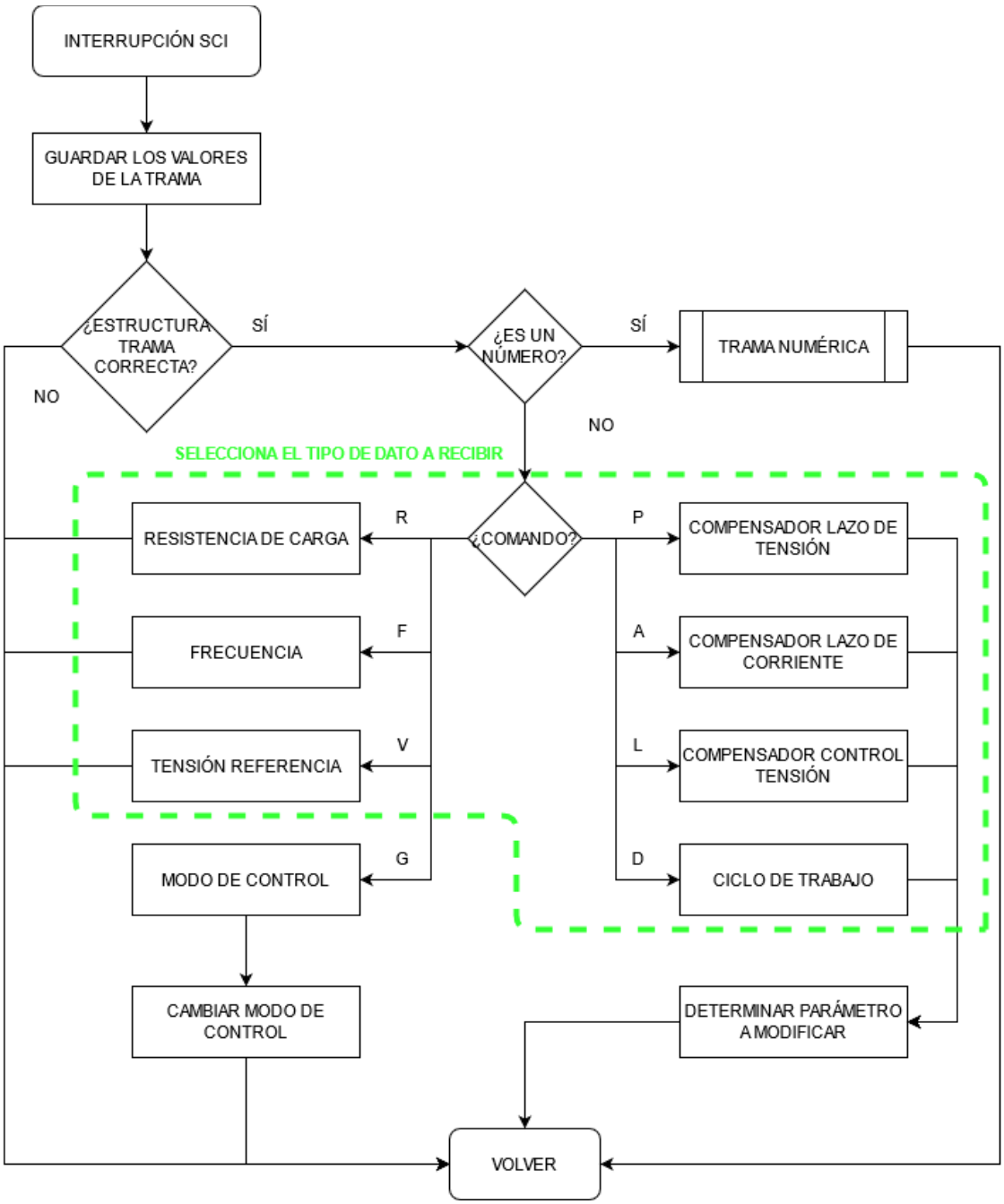


Figura 53. Flujograma de la función de interrupción del SCI.

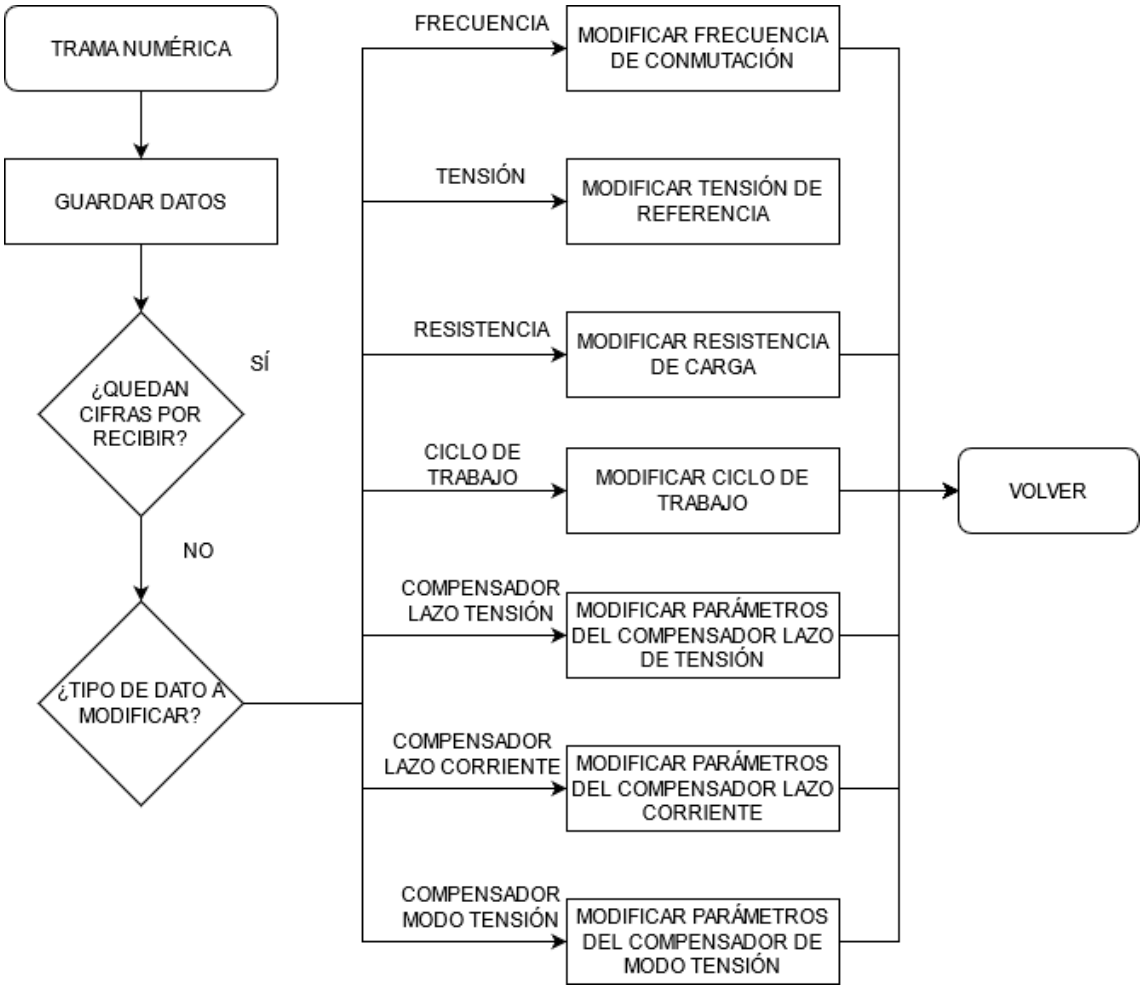


Figura 54. Flujograma de la función de la trama numérica.

8. INTERFAZ DE USUARIO Y PROTOCOLO DE COMUNICACIÓN

Se ha desarrollado una interfaz de usuario para posibilitar la configuración de los diferentes parámetros y modos de funcionamiento de forma sencilla. El programa de la interfaz ha sido desarrollado en **QT C++** y permite la modificación de los siguientes parámetros: frecuencia de conmutación de los MOSFETs, resistencia de carga, tensión de referencia, parámetros de los compensandores y ciclo de trabajo (solo en lazo abierto).

El protocolo de comunicación con el DSC es el que se detalla a continuación.

8.1. Protocolo de comunicación

8.1.1. Tramas de Interfaz a DSC

La comunicación de la interfaz al DSC se realiza a través de tramas, con una longitud fija de 4 bytes en un formato común. Primero el sistema recibe una trama de control con una disposición que se muestra a continuación:

Trama de control			
STX	Comando	Parámetro	ETX

Tabla 7. Formato de la trama de control.

- **Byte 0**: Carácter de comienzo de trama STX (código 0x02)
- **Byte 1**: Identificador del comando
- **Byte 2**: Parámetro del comando
- **Byte 3**: Carácter de fin de trama ETX (código 0x03)

Los campos Comando y Parámetro indican la variable a modificar o cambio de funcionamiento del sistema. El byte de Comando indica el grupo de variables/control y el byte de Parámetro especifica la variable/control en cuestión.

Estado	Comando	Parámetros	Variable	Función
1x	P	0	Vky[n-1]	Parámetros del compensador del *lazo de tensión* del control **modo Corriente Media**
		1	Vky[n-2]	
		2	Vky[n-3]	
		3	Vku[n]	
		4	Vku[n-1]	
		5	Vku[n-2]	
		6	Vku[n-3]	
2x	A	0	Aky[n-1]	Parámetros del compensador del *lazo de corriente* del control **modo Corriente Media**
		1	Aky[n-2]	
		2	Aky[n-3]	
		3	Aku[n]	
		4	Aku[n-1]	
		5	Aku[n-2]	
		6	Aku[n-3]	
3x	L	0	Vm_ky[n-1]	Parámetros del compensador del control **modo Tensión**
		1	Vm_ky[n-2]	
		2	Vm_ky[n-3]	
		3	Vm_ku[n]	
		4	Vm_ku[n-1]	
		5	Vm_ku[n-2]	
		6	Vm_ku[n-3]	
4x	V	0	Vref	Tensión de referencia
5x	R	0	Rload	Resistencia de carga
6x	F	0	Frequency	Frecuencia conmutación
7x	D	0	Duty	Ciclo de trabajo
8x	C	0	controlMode	Lazo Abierto
		1		Modo Tensión
		2		Modo Corriente Media

Tabla 8. Tabla con la combinación de comandos y parámetros, con sus respectivas variables y funciones relaciones, y el estado.

La x indica el valor del parámetro. El sistema funciona con una máquina de estados, donde cada estado indica la función a modificar. Luego de seleccionar la variable a modificar, el sistema recibe el valor en tramas numéricas como se indica en la tabla 9.

Nº	Trama numérica			
1	STX	Num	Num/ETX	Num/ETX
2	Num/ETX	Num/ETX	Num/ETX	Num/ETX
3	Num/ETX	Num/ETX	Num/ETX	ETX

Tabla 9. Formato de las tramas numéricas.

El microcontrolador puede llegar a recibir hasta un máximo de 3 tramas numéricas por cada variable. Como se indica en la figura anterior:

- **Byte 0**: Carácter de comienzo de trama STX (código 0x02)
- **Byte 1**: Cifra de un número
- **Byte 2-10**: Pueden ser una cifra o el carácter de final de trama (código 0x03)
- **Byte 11**: Carácter de final de trama (código 0x03) si se utiliza la trama nº 3

Dentro de las cifras se incluye también el punto de separación entre enteros y decimales. En caso de no completar una trama por falta de cifras, se rellenan con el carácter de final de trama.

8.1.2. Tramas de DSC a Interfaz

Como en el apartado anterior, la comunicación se realiza por tramas. En este caso, la interfaz al no tener limitación en el buffer, las tramas serán de longitud variable, en dos tamaños; de 4 bytes para la trama de control y de 6 bytes para la trama numérica. A continuación, se muestra la disposición de las tramas:

Trama de control			
STX	Comando	Parámetro	ETX

Tabla 10. Formato de la trama de control de DSC a interfaz.

- **Byte 0**: Carácter de comienzo de trama STX (código 0x02)
- **Byte 1**: Identificador del comando
- **Byte 2**: Parámetro del comando
- **Byte 3**: Carácter de fin de trama ETX (código 0x03)

Trama numérica					
STX	Num	Num	Num	Num	ETX

Tabla 11. Formato de la trama numérica de DSC a interfaz.

- **Byte 0**: Carácter de comienzo de trama STX (código 0x02)
- **Byte 1-4**: Cifra de un número adquirido en el formato del ADC del DSC.
- **Byte 5**: Carácter de fin de trama ETX (código 0x03)

8.2. Descripción detallada de la interfaz de usuario

La interfaz diseñada (figura 55) se divide en varias partes dependiendo de su función: barra de herramientas, parámetros del Buck, lecturas y control.

La barra de herramientas (figura 56) se divide en 3 iconos. De izquierda a derecha se compone de: conexión con el DSC, desconexión con el microcontrolador y configuración de los parámetros del puerto serie. Al seleccionar este último icono, se abrirá una ventana (figura 57) que permitirá al configurar el puerto serie según la estructura seleccionado en el microcontrolador. Los parámetros modificables son los siguientes: puerto, baudios, número de bits de información, paridad y el número de bits de parada. Una terminado la selección, al aplicar estos datos, se

guarda la configuración permitiendo cerrar la ventana. A continuación, al hacer click en el icono de conexión, el programa se conectará con el DSP y desbloqueará el icono de desconexión mientras que el primer icono se deshabilitará. En caso de error, se mostrará una ventana indicando el error.

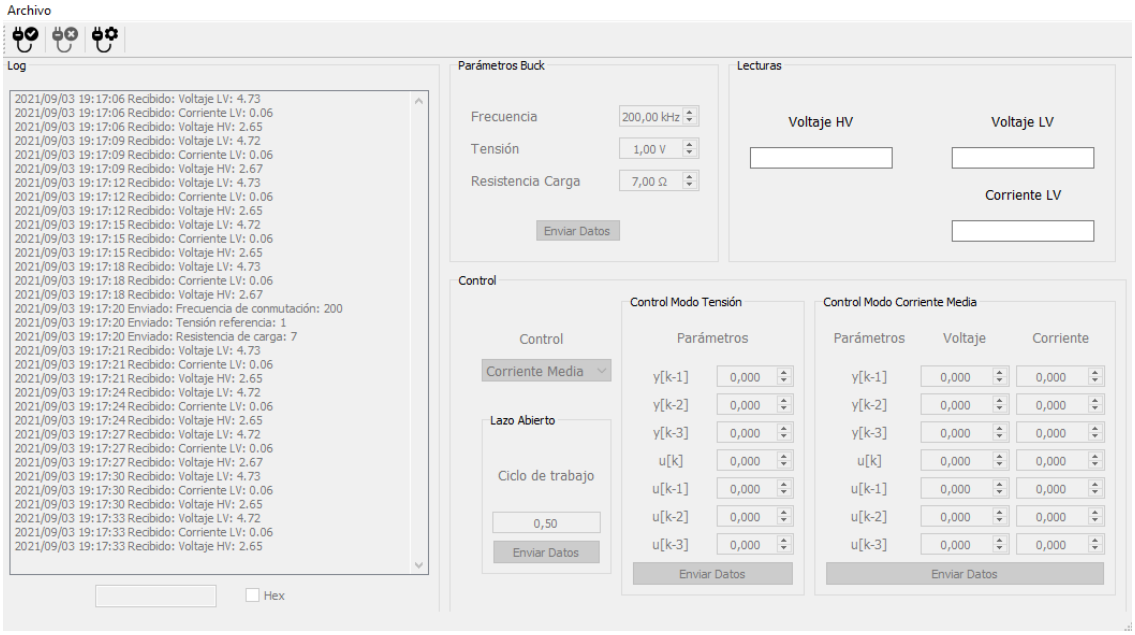


Figura 55. Interfaz gráfica diseñada.

Figura 56. Barra de herramientas de la interfaz.

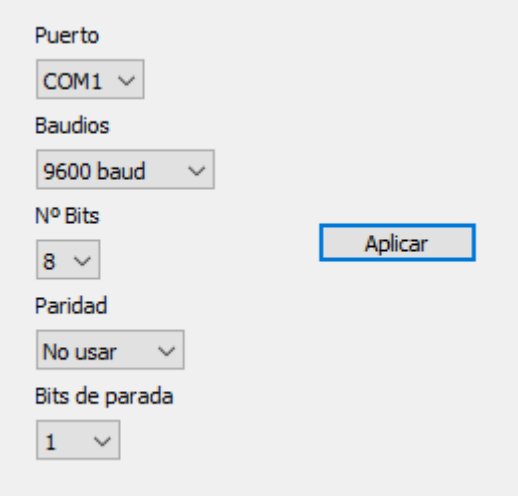


Figura 57. Ventana de configuración del puerto serie.

Una vez realizada la conexión, se desbloqueará todos los controles de la interfaz (figura 58). El apartado de control permite seleccionar diversos controles del convertidor (lazo abierto, modo tensión y modo corriente media) y modificar los datos de los parámetros de control del microcontrolador. La interfaz impedirá modificar variables no relacionadas con el tipo de control seleccionado.

Figura 58. Apartado de control de la interfaz.

Además, en el apartado de "Parámetros Buck" (figura 59), se puede modificar el funcionamiento del convertidor como puede ser: la frecuencia de conmutación, la tensión de referencia y la resistencia de carga.

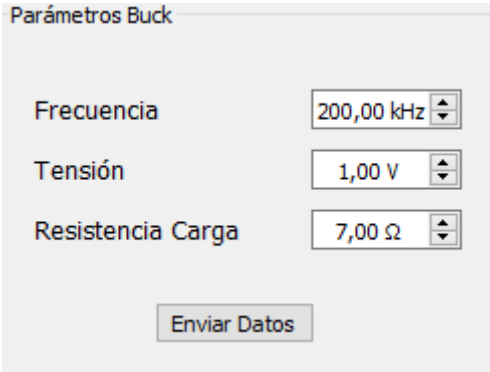


Figura 59. Apartado de parámetros del Buck en la interfaz.

Al conectarse la interfaz con el DSC, el programa comenzará a recibir datos cada segundo, sobre las lecturas de tensión de entrada y salida, y corriente por la bobina del convertidor, mostrándolas en el apartado de lecturas (figura 60). A su vez, el "logger" o registrador de datos (figura 61), realizará un seguimiento de toda la información recibida o enviada, y guardándola en un archivo "txt". La barra del menú, situado en el extremo superior, permite seleccionar donde el registro de datos del logger.

Como extra, debajo del logger se ha presentado un espacio, con la capacidad de enviar comandos tanto en decimal como en hexadecimal. Esto posibilita depurar con mayor eficacia el microcontrolador.

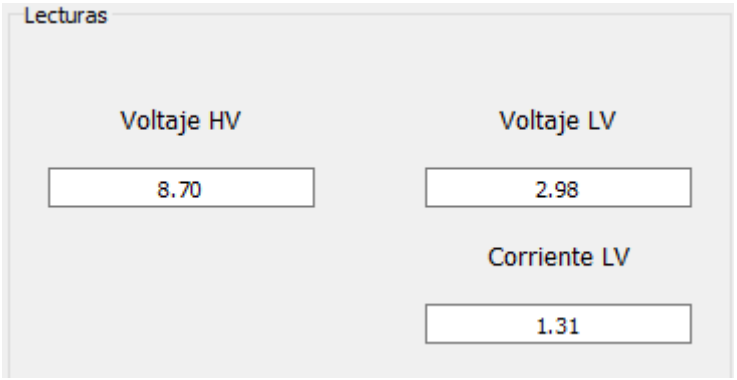


Figura 60. Apartado de lecturas del Buck en la interfaz.

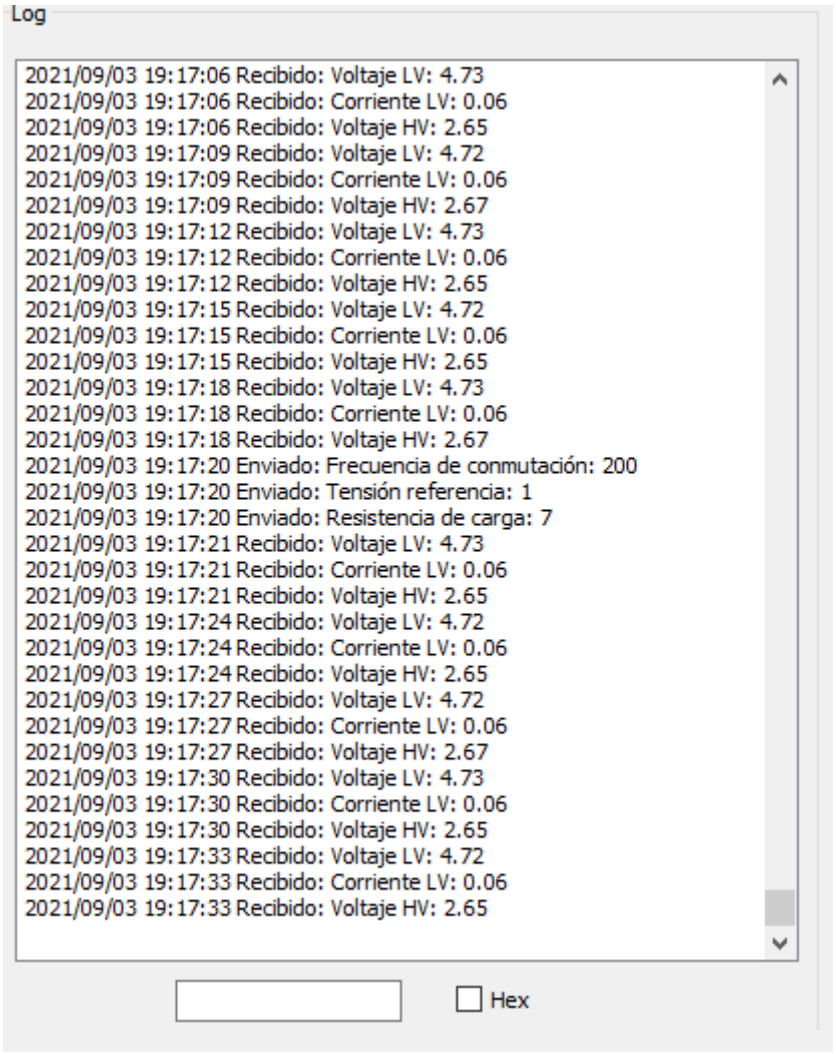


Figura 61. Registro del funcionamiento de la interfaz.

9. ENSAYOS EXPERIMENTALES

Los valores de los compensadores del control modo corriente se han implementado dentro del DSP TMS320F28027FPTT operando a 60 MHz. La tensión de salida del convertidor Buck fijando una referencia de 2 V se muestra en la figura 62. La señal presenta perturbaciones de ruido durante la adquisición de datos. Empleando los cursores se ha determinado aproximadamente el rizado de la tensión, con un valor de 70 mV de amplitud, es decir, un 3.5%.

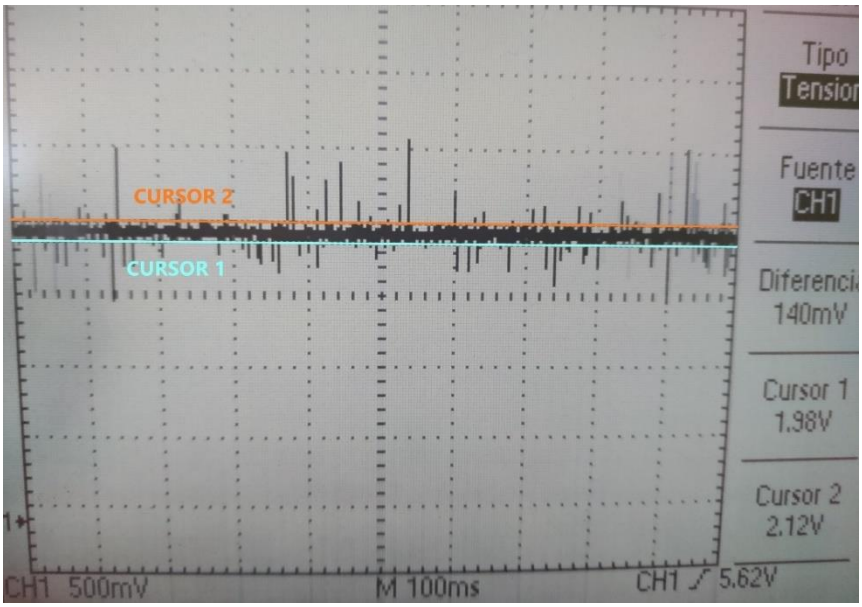


Figura 62. Tensión de salida del convertidor con una referencia de 2 V y transistor de carga a 50 % de duty.

La tensión de salida frente a un cambio de tensión de referencia (de 1 V a 2 V) se muestra en la figura 63. La respuesta del control es rápida, con un tiempo de establecimiento de aproximadamente 24 ms. La sobreoscilación que presenta la señal es de 200 mV o un 10 %. Una diferencia notable respecto al MF marcado durante los cálculos.

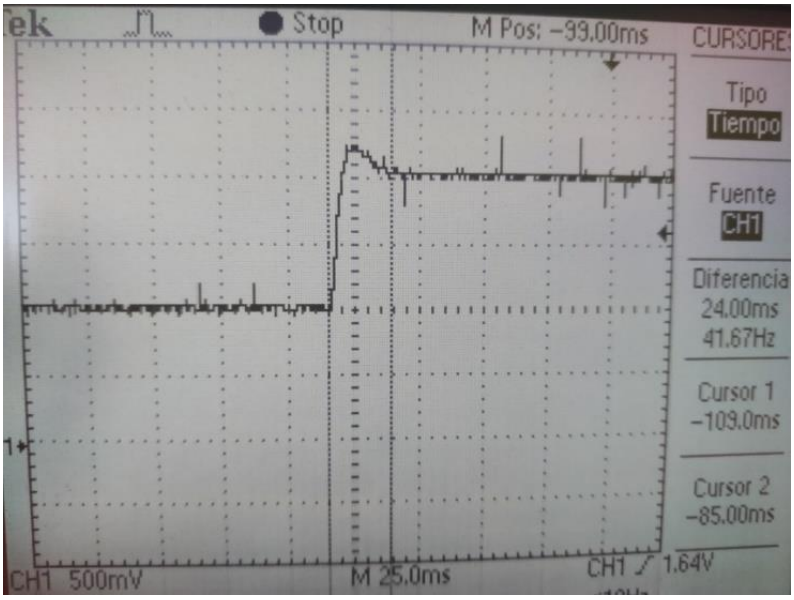


Figura 63. Respuesta de la tensión de salida frente a un cambio de la tensión de referencia (1 V a 2 V) y transistor de carga a 50 % de duty. Los cursores marcan el tiempo de establecimiento.

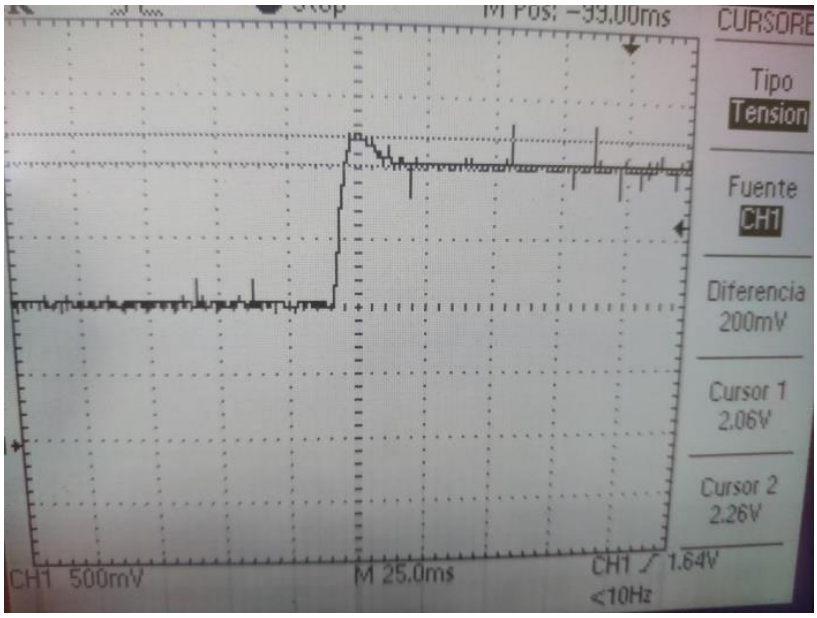


Figura 64. Respuesta de la tensión de salida frente a un cambio de la tensión de referencia (1 V a 2 V) y transistor de carga a 50 % de duty. Los cursores marcan la sobreoscilación.

En la figura 65, se muestra la corriente por la bobina con una tensión de referencia de 3 V como en la simulación. Aplicando un ciclo de trabajo del 50 % en el transistor de carga, el sistema se mantiene en conducción continua. El resultado real es mejor que el planteado en Matlab. Existe diversos factores que afectan a esta corriente como: el uso de un convertidor síncrono que evita la tensión de conducción del diodo, y/o los valores reales de los componentes.

La lectura de la corriente se realiza a través de un amplificador restador. Para obtener la corriente real se deberá de dividir la tensión entre la ganancia del circuito de captura y transformar esa tensión a corriente a través de la ley de Ohm (resistencia shunt = 0.03 Ω). Realizando los cálculos, la **corriente media de la bobina** es de aproximadamente 1.34 A, coincidiendo con los datos de la simulación.

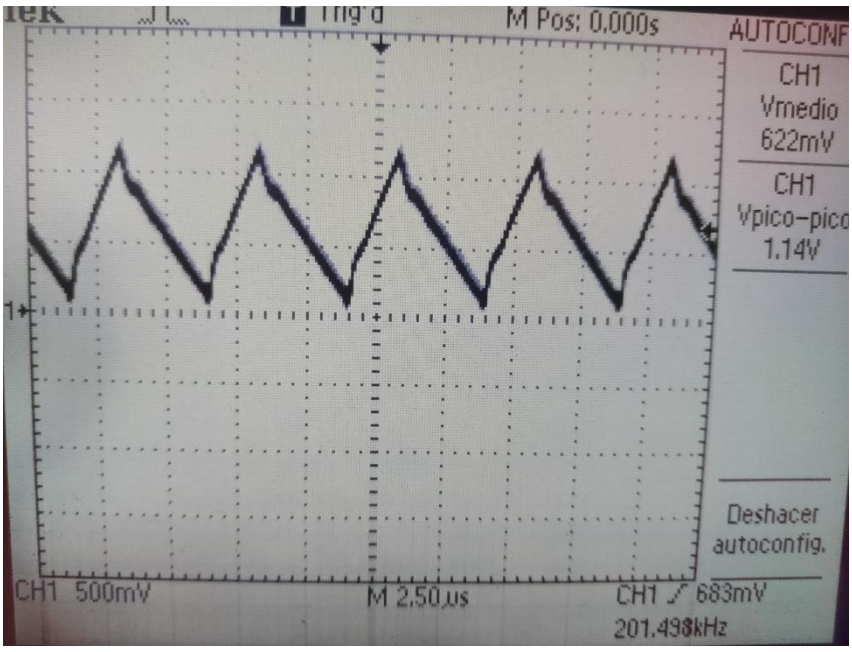


Figura 65. Corriente por la bobina del convertidor con una referencia de 3 V y un ciclo de trabajo en el transistor de carga del 50 %.

La tensión de entrada obtenida a través de la placa de desarrollo no es correcta. Tiene una desviación de aproximadamente 0.25 V. Esto puede ser causado por la variación del valor resistivo proporcionado en los planos.

Figura 66. Tensión de entrada medida con el DSP.

Comprobando el tiempo de procesamiento del control (figura 67) y teniendo en cuenta la velocidad del microcontrolador, el control no se realiza a la frecuencia de muestreo marcada. La frecuencia mínima de los reguladores "f_{min_real}"del sistema viene dada por la siguiente ecuación:

$$f_{min\_real} = \frac{1}{nº\ ciclos\ \cdot\ T_{BCLK}} = \frac{1}{9214 \cdot 16.667\ ns} = 6511\ Hz$$

Figura 67. Número de ciclos de reloj máximo de la interrupción del control.

El control propuesto no es el óptimo. Los reguladores funcionan a una frecuencia mucho menor a la propuesta (50 kHz y 25 kHz). Se debería de volver a discretizar el control a esta frecuencia. En caso de no ser estable, se optaría por bajar la frecuencia de cruce de los compensadores. No ha habido tiempo suficiente para implementar el nuevo control.

10. CONCLUSIONES

Durante la realización del proyecto se ha comprobado el funcionamiento del control modo corriente mediante el empleo del método factor k de Venables. Este método proporciona una mayor sencillez durante el diseño de los compensadores, aunque la determinación de las frecuencias de cruce no ha sido óptima. Al implementar otras frecuencias, en la simulación en Matlab, la corriente por la bobina alcanzaba valores nulos, por lo que el sistema entraría en conducción discontinua. Además, la implementación del compensador en el DSP ha sido difícil. Existe una gran variabilidad entre los resultados simulados y los reales. Modificando las frecuencias de cruce y la frecuencia de muestreo podía dar a lugar al fallo del control en el microcontrolador.

Los resultados finales obtenidos son satisfactorios. La respuesta de la tensión de salida frente a cambios de referencia en 24 ms indica un rápido control del sistema diseñado. La sobreoscilación presentada durante los cambios de tensión de referencia son bastante más altas que las previstas teóricamente. Aunque hay que destacar que la velocidad de los reguladores no es correcta y, por tanto, existe margen de mejora al fijar de manera más adecuada los compensadores y las frecuencias de muestreo.

La limitación del sensor de corriente al exceder el 50% del ciclo de trabajo del interruptor ha supuesto una gran limitación en cuanto al rango de tensiones de salida disponibles. Asimismo, sustituyendo el convertidor por otro de mayor potencia y/o tensión de entrada hubiese supuesto menores dificultades para la fijación de frecuencias de cruce y mantenimiento del sistema en conducción continua.

La interfaz por puerto serie se ha diseñado con el objetivo de facilitar las pruebas y modificación del control de un convertidor reductor. El programa del microcontrolador diseñado con el cambio de las ganancias de tensión y corriente posibilitaría su implementación en otros convertidores y a su vez, empleando la misma interfaz.

En conclusión, el sistema diseñado es eficiente pero limitado por factores físicos del convertidor y del DSP. Empleando otro Buck supondría una mejora del control general, tanto a nivel de diseño como en resultados.

11. REFERENCIAS

[1] “Fuentes de alimentación conmutadas”, apuntes de clase de Electrónica de Potencia. Departamento de Ingeniería Electrónica, Universidad Politécnica de Valencia, Primavera 2021.

[2] “Reference Designs: TIDM-DC-DC-BUCK C2000™ Digital Power BoosterPack™”, Texas Instruments.

[3] “TMS320F2802x, TMS320F2802xx Piccolo Technical Reference Manual”, Texas Instruments.

[4] “Modelización de etapas de potencia de convertidores DC/DC”, apuntes de clase de SEI. Departamento de Ingeniería Electrónica, Universidad Politécnica de Valencia, Primavera 2021.

[5] “Control de Inversores Monofásicos”, apuntes de clase de SEI. Departamento de Ingeniería Electrónica, Universidad Politécnica de Valencia, Verano 2021.

[6] “The Link Between The Phase Margin And The Converter Transient Response”, Christophe Basso.

[7] “Datasheet TMS320F2802x Microcontrollers”, Texas Instruments.

12. BIBLIOGRAFÍA

Mammano, R. (1999). *Switching Power Supply Topology.* Texas Instruments.

Microchip. (2021, 4 25). *Microchip Developer Help*. Retrieved from Microchip Developer Help: https://microchipdeveloper.com/pwr3201:buck-converter-average-current-mode

Murad, I. K. (2019). Efficiency of Synchronous and Asynchronous Buck-Converter at Low Output Current. *Journal of University of Babylon for Engineering Sciences, 27*(2), 13.

Rogers, E. (1999). *Understanding Buck Power Stages in Switchmode Power.* Texas Instruments.

Venable, H. D. (2021, 8 6). *Venable Instruments.* Retrieved from Venable Instruments: https://www.venableinstruments.com/hubfs/The%20K%20Factor%20a%20New%20Mathematical%20Tool%20for%20Stability%20Analysis.pdf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO Y PROGRAMACIÓN DEL CONTROL ACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSP F28027 DE TEXAS INSTRUMENTS

TRABAJO FINAL DE GRADO

DOCUMENTO II: PLANOS

Autor:

Chen Zhu, Oscar Jia Xing

Tutor:

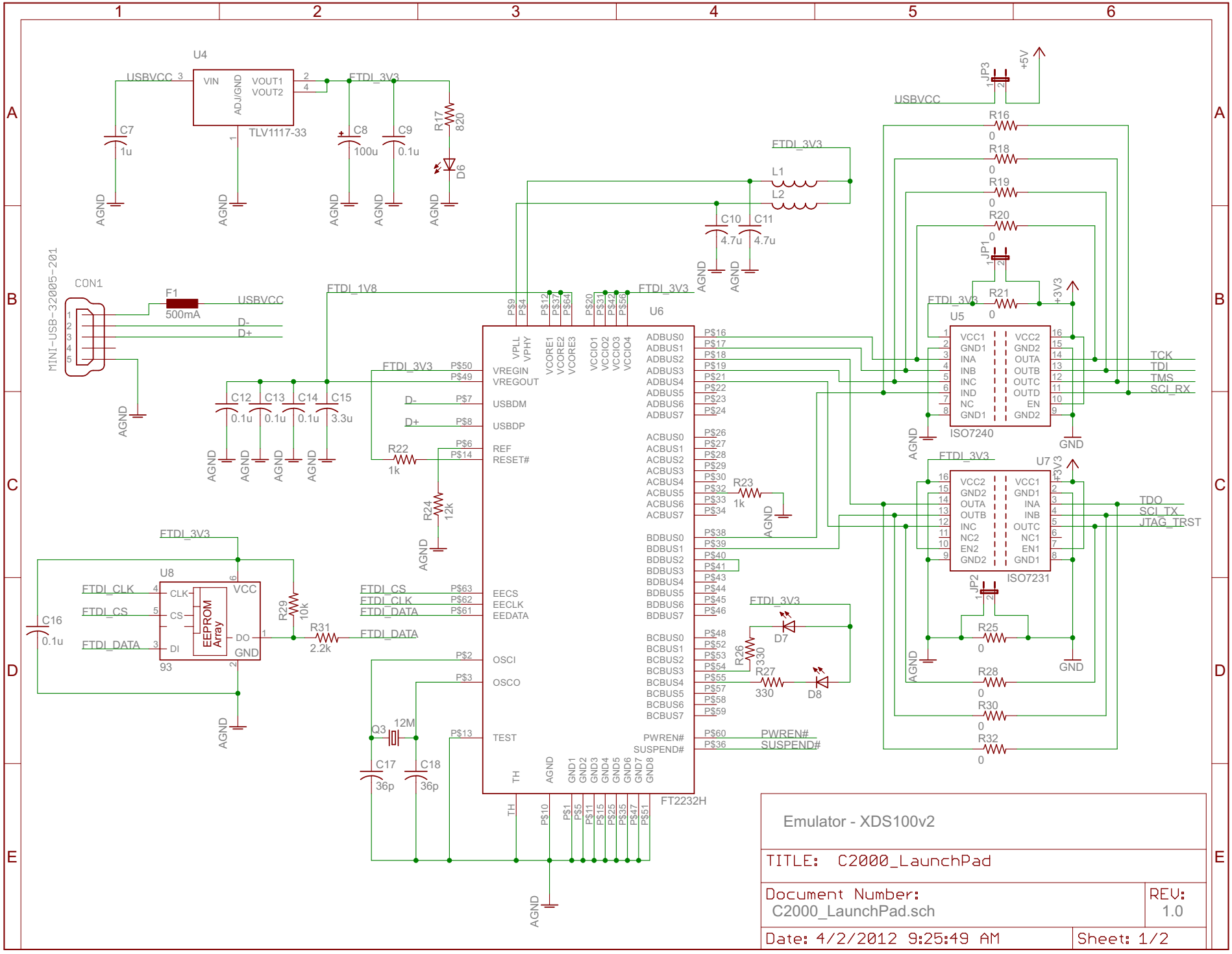
Orts Grau, Salvador

Titulación:

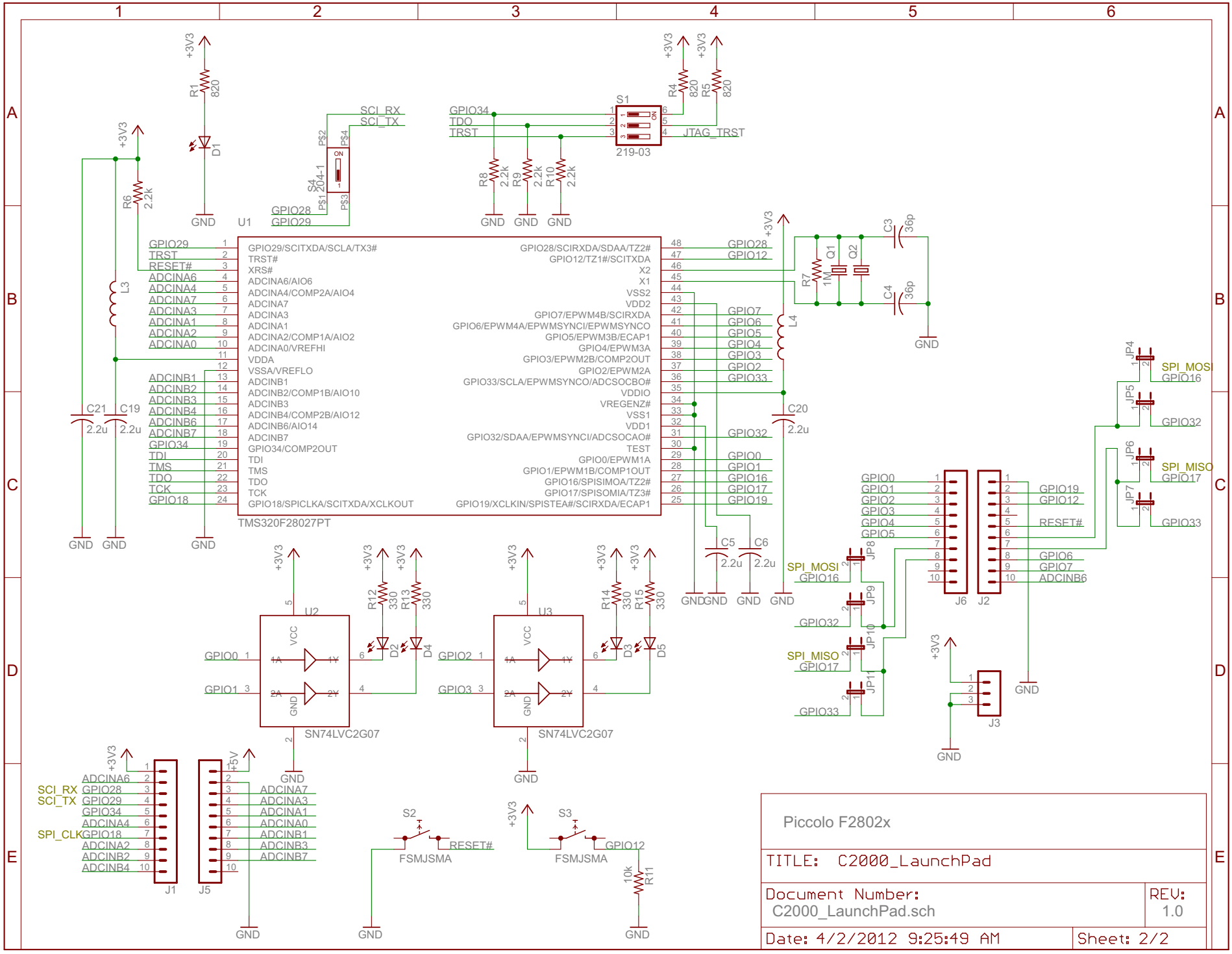
Grado en Ingeniería Electrónica Industrial y Automática

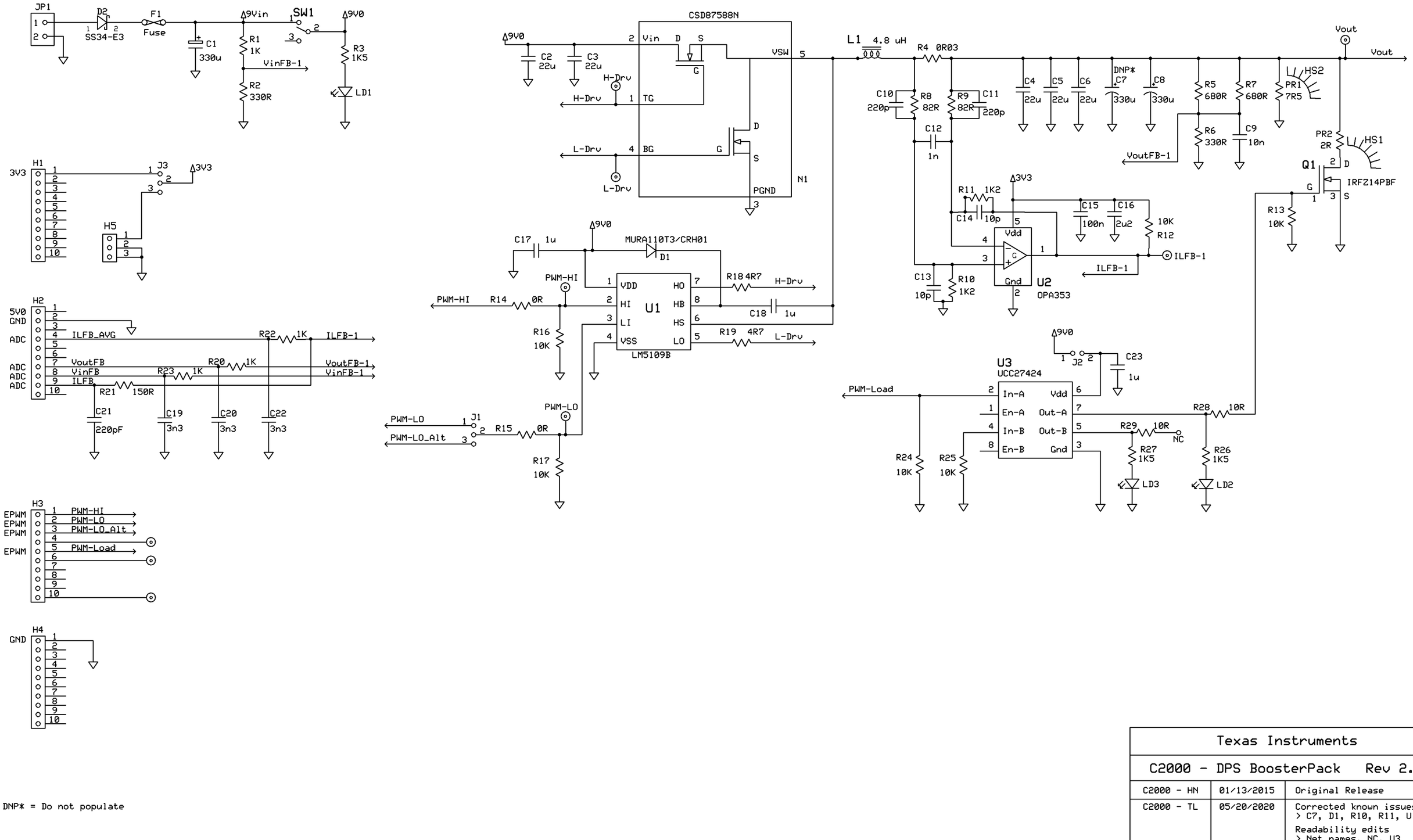
Curso Académico 2020/2021

Índice de planos



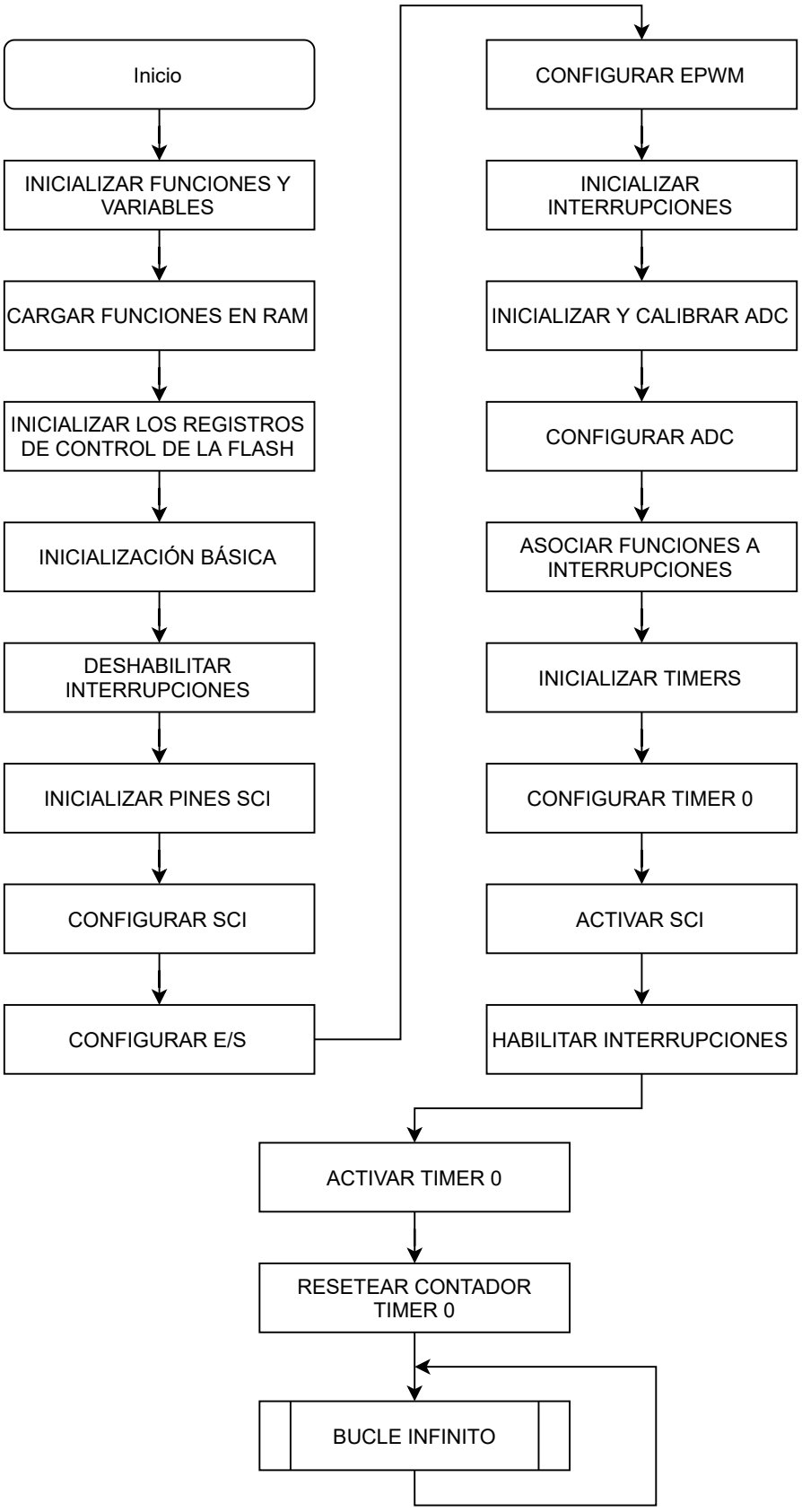
4/2/2012 9:27:15 AM f=0.89 C:\controlsuite_gitorious_dev\development_kits\C2000_LaunchPad\C2000_LaunchPad.sch (Sheet: 1/2)


Piccolo F2802x
TITLE: C2000_LaunchPad
Document Number: C2000_LaunchPad.sch
REV: 1.0
Date: 4/2/2012 9:25:49 AM
Sheet: 2/2
TMS320F28027PT
SN74LVC2G07
SN74LVC2G07
219-03
FSMJSMA
FSMJSMA



DNP* = Do not populate

Texas Instruments		
C2000 - DPS BoosterPack Rev 2.1		
C2000 - HN	01/13/2015	Original Release
C2000 - TL	05/20/2020	Corrected known issues > C7, D1, R10, R11, U1-1 Readability edits > Net names, NC, U3



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**

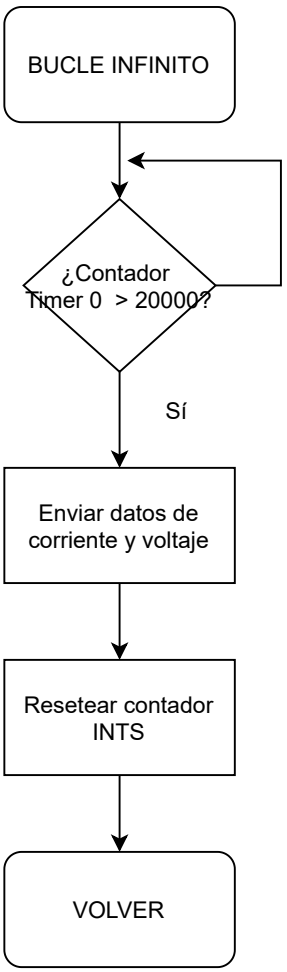
Fecha: 07/09/21

Escala
E/S

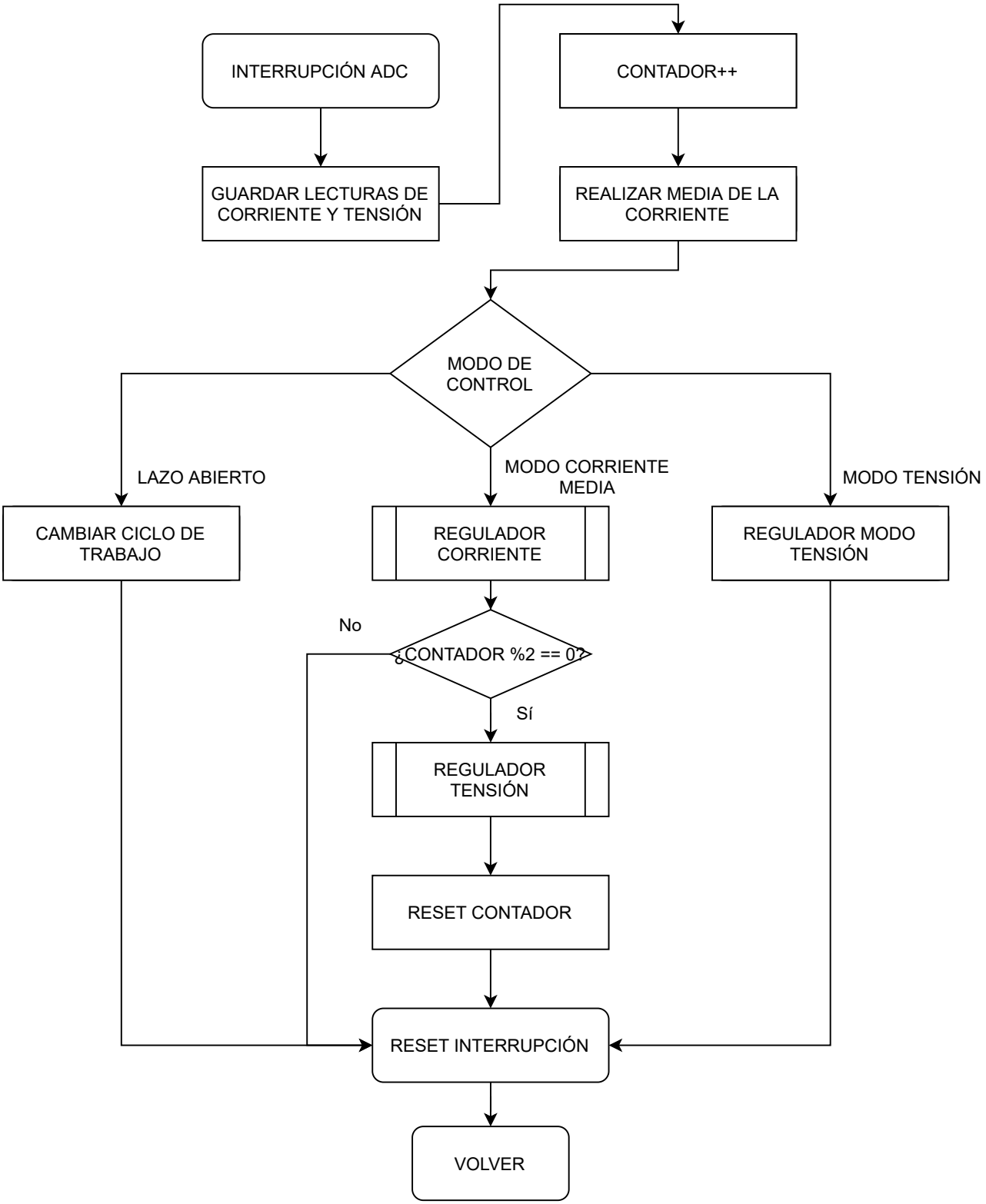
Autor: Oscar Jia Xing
Chen Zhu

Flujograma
Programa principal del DSC

Plano Nº
01



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**		Fecha: 07/09/21 Escala E/S
Autor: Oscar Jia Xing Chen Zhu	Flujograma Función bucle infinito	Plano Nº 02



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**

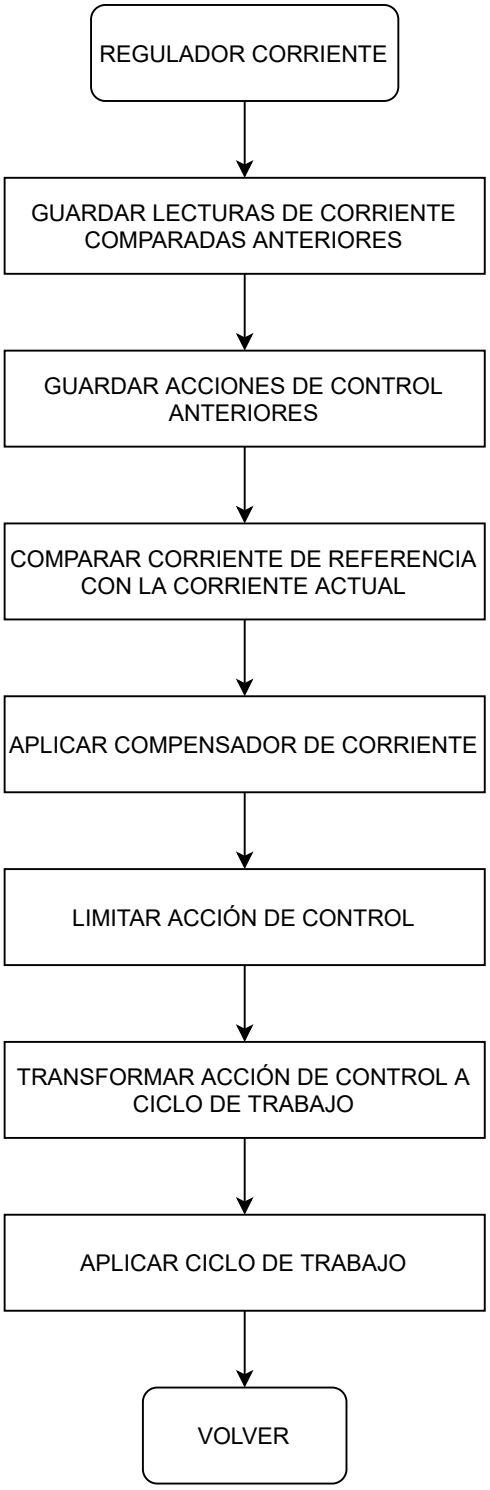
Fecha: 07/09/21

Escala
E/S

Autor: Oscar Jia Xing
Chen Zhu

Flujograma
Interrupción ADC

Plano Nº
03



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**

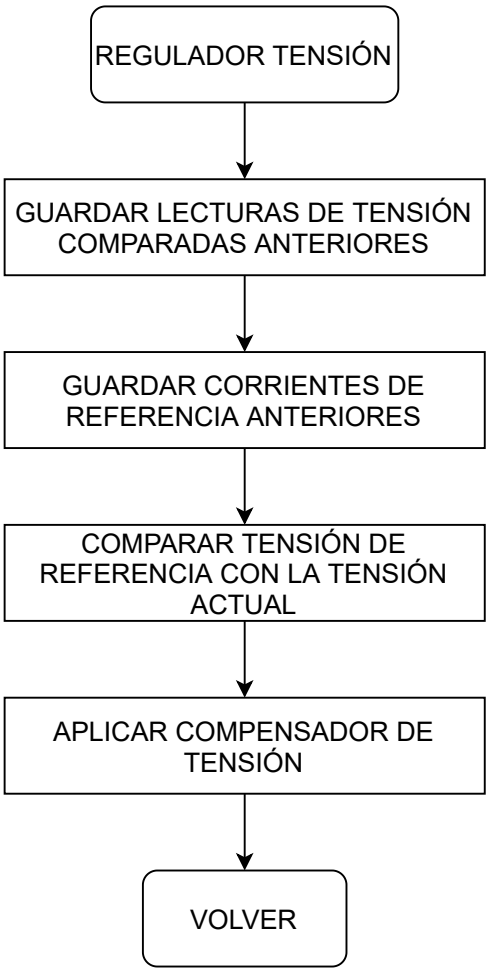
Fecha: 07/09/21

Escala
E/S

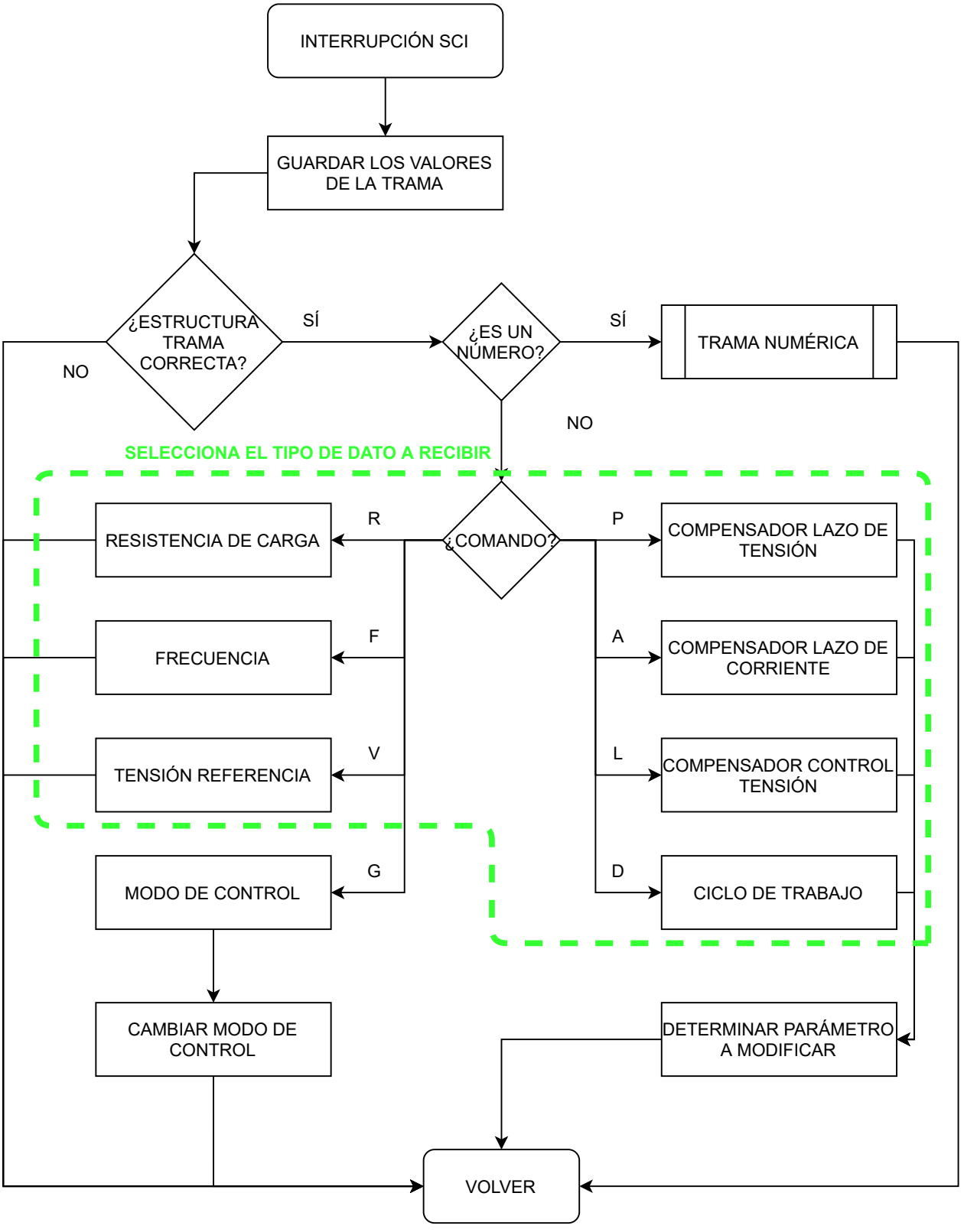
Autor: Oscar Jia Xing
Chen Zhu

Flujograma
Función regulador de corriente

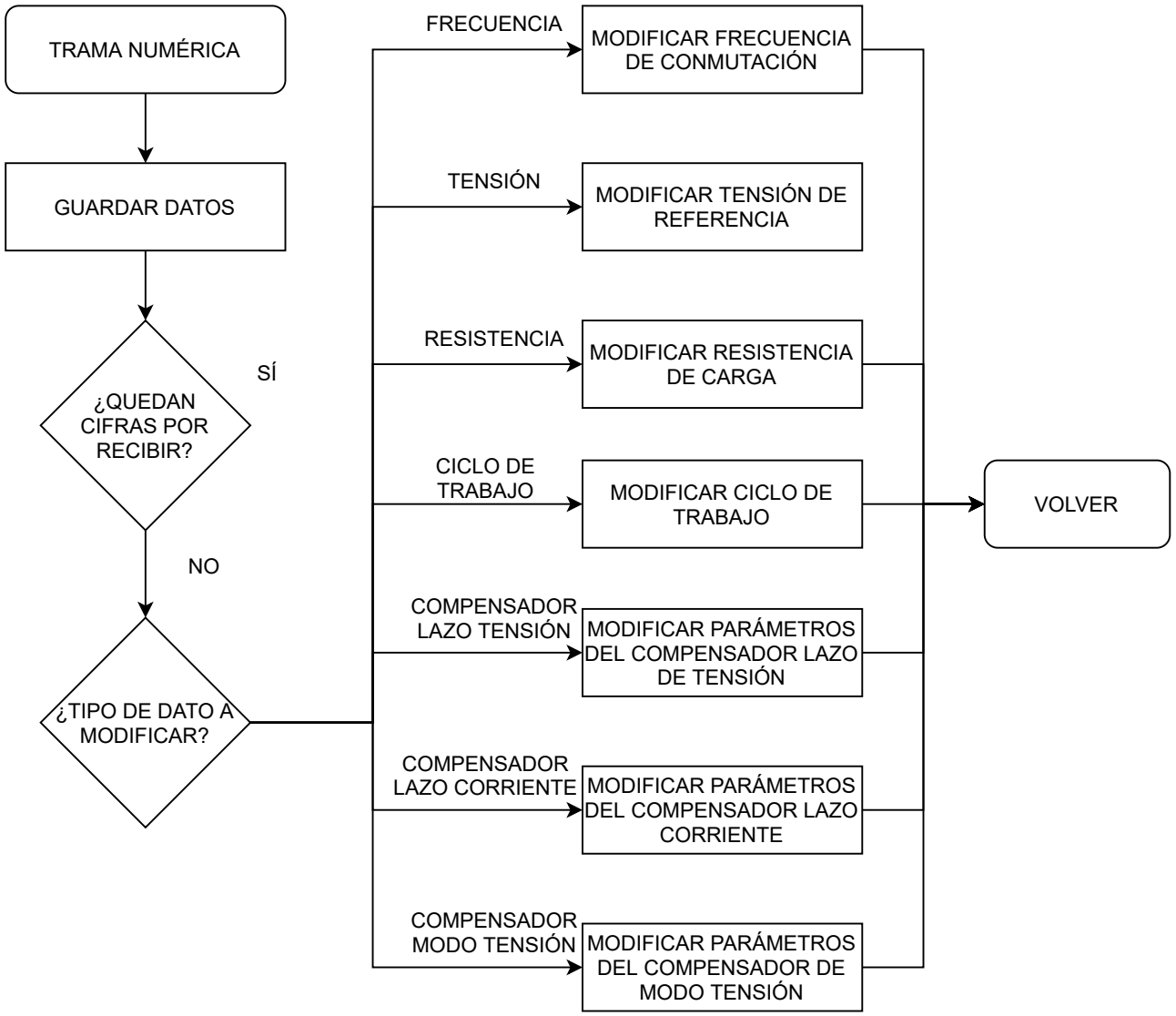
Plano Nº
04



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**		Fecha: 07/09/21 Escala E/S
Autor: Oscar Jia Xing Chen Zhu	Flujograma Función regulador de tensión	Plano Nº 05



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**		Fecha: 07/09/21 Escala E/S
Autor: Oscar Jia Xing Chen Zhu	Flujograma Interrupción SCI	Plano Nº 06



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**

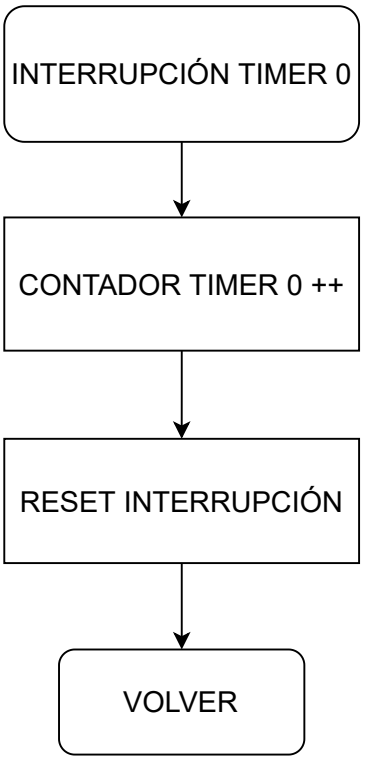
Fecha: 07/09/21

Escala
E/S

Autor: Oscar Jia Xing Chen Zhu

Flujograma
Función Trama Numérica

Plano Nº
07



PROYECTO: **DESARROLLO Y PROGRAMACIÓN DEL CONTROLACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSPF28027 DE TEXAS INSTRUMENTS**

Fecha: 07/09/21

Escala
E/S

Autor: Oscar Jia Xing
Chen Zhu

Flujograma
Interrupción Timer 0

Plano Nº
08

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO Y PROGRAMACIÓN DEL CONTROL ACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSP F28027 DE TEXAS INSTRUMENTS

TRABAJO FINAL DE GRADO

DOCUMENTO III: PRESUPUESTO

Autor:

Chen Zhu, Oscar Jia Xing

Tutor:

Orts Grau, Salvador

Titulación:

Grado en Ingeniería Electrónica Industrial y Automática

Curso Académico 2020/2021

PRESUPUESTO

A continuación, se muestra el precio descompuesto de la mano de obra empleado para el diseño del proyecto.

MANO DE OBRA			
CONCEPTO	PRECIO UNITARIO [€/h]	CANTIDAD [h]	PRECIO [€]
Diseño del control ACC	12	80	960
Simulación del control ACC	12	50	600
Implementación del control sobre el DSP	12	100	1200
Diseño de interfaz	12	70	840
		TOTAL MO	3600

También, se ha incluido el coste de la licencia del software y materiales utilizados.

LICENCIA SOFTWARE			
CONCEPTO	PRECIO ANUAL [€/año]	CANTIDAD [meses]	PRECIO [€]
Matlab R2021a	800	1	66,67
Simulink	1200	1	100,00
Simscape	800	1	66,67
Simscape Electrical	1200	1	100,00
System Identification Toolbox	460	1	38,33
Microsoft 365	69	1	5,75
QT	3948	1	329,00
		TOTAL SOFTWARE	706,42

MATERIALES			
CONCEPTO	PRECIO UNITARIO [€]	CANTIDAD[h]	PRECIO [€]
LAUNCHXL-F28027F	19,59	1	19,59
BOOSTXL-BUCKCONV	66,46	1	66,46
		TOTAL MATERIAL	86,05

Más adelante se detalla el resumen del presupuesto, incluyendo los gastos generales, IVA y beneficio industrial.

TIPO DE COSTE	**IMPORTE [€]**
Mano de obra	3600,00
Licencia software	706,42
Materiales	86,05
Presupuesto de ejecución material	**4392,47**
13% de gastos generales	571,02
6% de beneficio industrial	263,55
Suma	**5227,04**
21% IVA	1097,68
TOTAL PRESUPUESTO	**6324,71**

El presupuesto general asciende a la expresad cantidad de SEIS MIL TRESCIENTOS VEINTICUATRO CON SETENTAIOCHO EUROS.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO Y PROGRAMACIÓN DEL CONTROL ACC DE UN CONVERTIDOR BUCK MEDIANTE CONTROL DIGITAL IMPLEMENTADO SOBRE EL DSP F28027 DE TEXAS INSTRUMENTS

TRABAJO FINAL DE GRADO

DOCUMENTO IV: ANEXOS

Autor:

Chen Zhu, Oscar Jia Xing

Tutor:

Orts Grau, Salvador

Titulación:

Grado en Ingeniería Electrónica Industrial y Automática

Curso Académico 2020/2021

Índice

1. Código LAUNCHXL-F28027

Archivo “main.c”

```
/**********************************************************************
 *      Desarrollo y programación del control ACC de un convertidor Buck
 *                  mediante control digital implementado sobre el
 *                       DSP F28027 de Texas Instruments
 *
 *                                  main.c
 *
 *                      Autor: Oscar Jia Xing Chen Zhu
 **********************************************************************/

#include "DSP28x_Project.h"
#include "setup.h"
#include "stdlib.h"
#include "stdio.h"
#include "math.h"
#include "f2802x_epwm_defines.h"
#include "f2802x_examples.h"

// Declaracion de funciones
void Sci_Envia_Caracter(int a);

float check_number(void);
bool check_command(void);

void mef_sci(void);
bool check_sci();
void send_data(uint16_t comando, uint16_t parametro, float data);

void regulador_tension(void);
void regulador_corriente(void);
void regulador_modoTension(void);

// Interrupciones
interrupt void scia_rx_isr(void);
interrupt void cpu_timer0_isr(void);
interrupt void adc1_isr(void);

// Funciones guardadas en Flash y funcionando en RAM
#pragma CODE_SECTION(adc1_isr, "ramfuncs");
#pragma CODE_SECTION(cpu_timer0_isr, "ramfuncs");
#pragma CODE_SECTION(regulador_tension, "ramfuncs");
#pragma CODE_SECTION(regulador_corriente, "ramfuncs");
#pragma CODE_SECTION(regulador_modoTension, "ramfuncs");

// Variables externas necesarias para iniciar funciones en RAM
extern uint16_t RamfuncsLoadStart;
extern uint16_t RamfuncsLoadSize;
extern uint16_t RamfuncsRunStart;
```

Archivo “main.c” (continuación)

```
// Parámetros compensador control modo tensión
params param_voltageMode = {.y = {0, 0, 0, 0}, .u = { 0, 0, 0, 0}, .ky = {-
0.5005, 0.9376, 0.5628}, .ku = {3.184, -1.218, -2.88, 1.521}};

// Parámetros compensador control modo corriente
params param_currentMode0 = {.y = { 0, 0, 0, 0}, .u = { 0, 0, 0, 0}, .ky =
{0.928, 0.07203, 0}, .ku = {0.03204, 0.005044, -0.027, 0}};
params param_currentMode1 = {.y = { 0, 0, 0, 0}, .u = { 0, 0, 0, 0}, .ky =
{1.757, -0.757, 0}, .ku = {0.5066, 0.0156, -0.491, 0}};


// Función principal
int main(void)
{
    // Carga las funciones en la ram
    memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (size_t)&RamfuncsLoadSize);

    // Mejora el rendimiento de las funciones alojadas en la memoria FLash
    InitFlash();

    // Inicializacion basica
    InitSysCtrl(); // DSP2802x_SysCtrl.c

    // Deshabilita interrupciones
    DINT;

    // Inicializa E/S (solo pines SCI-A)
    InitSciaGpio();

    // Configuración SCI
    Setup_Sci();

    // Configuración E/S (LEDs)
    Setup_Gpio();

    // Configuración ePWM
    Setup_ePWM();

    // Inicializa interrupciones
    InitPieCtrl();  // DSP2833x_PieCtrl.c
    InitPieVectTable();

    // Inicializa ADC
    InitAdc();
    AdcOffsetSelfCal();

    // Configuración ADC
    config_ADC();
```

Archivo “main.c” (continuación)

```
// Asociación de funciones a interrupciones
    EALLOW;
    PieVectTable.TINT0 = &cpu_timer0_isr;   // registra ISR de Timer-0
    PieVectTable.SCIRXINTA = &scia_rx_isr;  // registra ISR de RX SCI
    PieVectTable.rsvd10_1 = &adc1_isr;      // registra ISR de ADCINT1
    EDIS;

    // Inicializa timers
    InitCpuTimers();
    ConfigCpuTimer(&CpuTimer0,60,20);   // configura Timer-0 a 100 us


    // Activa interrupciones
    PieCtrlRegs.PIEIER1.bit.INTx7 = 1;   // Timer 0
    PieCtrlRegs.PIEIER9.bit.INTx1 = 1;   // SCIRXINTA
    PieCtrlRegs.PIEIER10.bit.INTx1 = 1;  // ADCINT1

    IER |= M_INT1;
    IER |= M_INT9;
    IER |= M_INT10;

    // Habilita interrupciones
    EINT;
    ERTM;

    // Pone Timer-0 en marcha
    CpuTimer0Regs.TCR.bit.TSS = 0;

    // Reinicia contador del Tiomer 0
    CpuTimer0.InterruptCount = 0;


    // Bucle infinito
    while(1)
    {
        // Espera 1 s
        while(CpuTimer0.InterruptCount <= 50000){}

        // Reset contador
        CpuTimer0.InterruptCount = 0;

        // Envía datos de voltaje y corriente a la interfaz cada 1s
            send_data('V','0', voltajeIn);
            send_data('I','0', corrienteF);
            send_data('V','1', voltajeOut);
    }
}
```

Archivo “main.c” (continuación)

```
// Envia un caracter por el SCI
void Sci_Envia_Caracter(int a)
{
    while (SciaRegs.SCIFFTX.bit.TXFFST != 0) {}
    SciaRegs.SCITXBUF = a;
}

// ISR del Timer 0 | 20 kHz
interrupt void cpu_timer0_isr(void)
{
    // Contador Timer 0 +1
    CpuTimer0.InterruptCount++;

    // Reconoce la interrupcion
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}

// Interrupción ADCINT1 | Timer
interrupt void adc1_isr(void)
{
    voltage[0] = AdcResult.ADCRESULT0;    // Tension_LV
    voltage[1] = AdcResult.ADCRESULT1;    // Tension_HV

    current[0] = AdcResult.ADCRESULT2;    // Corriente_LV
    current[1] = AdcResult.ADCRESULT3;    // Corriente_LV Filtrada

    // Lectura de datos reales para seguimiento en CSS
    corriente = current[0] * 3.3 / 4096 * 82 /1200 / 0.03;
    corrienteF = current[1] * 3.3 / 4096 * 82 /1200 / 0.03;
    voltajeIn = voltage[1] * 3.3 / 4096 * (1000 + 330) / 330;
    voltajeOut = voltage[0] * 3.3 / 4096 * (330 + 340) / 330;


    // Lazo Abierto
    if(controlMode == 0){

        EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD * duty;

        // Control Modo Tensión
    } else if(controlMode == 1){
        regulador_modoTension();

        //Control Modo Corriente Media
    } else if(controlMode == 2){

        // Regulador del lazo de corriente
        regulador_corriente();
        contador25kHz++;
```

Archivo “main.c” (continuación)

```
// Regulador del lazo de tensión
        if(contador25kHz >= 2){
            regulador_tension();
            contador25kHz = 0;
        }
    }

    AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;      //Clear ADCINT1 flag reinitialize
for next SOC
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP10;   // Acknowledge interrupt to PIE
}


// ISR de recepcion del SCI
interrupt void scia_rx_isr(void)
{
    int i;

    // Lee y almacena la trama recibida
    for(i = 0; i < 4; i++)
        receivedData[i] = SciaRegs.SCIRXBUF.all;

    if(trama_num == 1)
        check_sci();
    else {
        mef_sci();
    }


    // Borra flags y reconoce interrupcion
    SciaRegs.SCIFFRX.bit.RXFFOVRCLR = 1;  // borra flag de overflow
    SciaRegs.SCIFFRX.bit.RXFFINTCLR = 1;  // borra flag de interrupcion
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;  // ack PIE
}

// Comprueba el comando de la trama | Devuelve 0 si es erróneo
bool check_command(void)
{
    switch(receivedData[1])
    {
        case 'P':   estado = 1; break;
        case 'A':   estado = 2; break;
        case 'L':   estado = 3; break;
        case 'V':   estado = 4; break;
        case 'R':   estado = 5; break;
        case 'F':   estado = 6; break;
        case 'D':   estado = 7; break;
        case 'C':   estado = 8; break;
```

Archivo “main.c” (continuación)

```
        default:
            estado = 0;
            return 1;
    }
    return 0;
}

// Comprueba el parámetro de la trama y cambia de estado
void check_param(void)
{
    // Devuelve al estado inicial si el valor del parámetro no está dentro de
las especificaciones
    if(receivedData[2] < '0' || receivedData[2] > '9' )
    {
        estado = 0;
        return;
    }

    // Actualiza el estado del MEF
    estado = estado * 10 + receivedData[2] - 48;

    // Comrpueba si es un cambio de modo de control
    if(estado >= 80 && estado < 83)
    {
        mef_sci();
        return;
    }
    trama_num = 1;

}

// Máquina de estados que selecciona las direcciones de los datos a guardar
void mef_sci(void)
{
    switch(estado)
    {
        // Comprueba si la estructura de la trama es correcta
        case 0: if(receivedData[0] == 0x02 || receivedData[3] == 0x03)
        {
            if(!check_command()) check_param();
        }
        break;

        // Parámetro Lazo Tensión Modo Corriente Media
        case 10: param_currentMode1.ky[0] = check_number();  break;
        case 11: param_currentMode1.ky[1] = check_number();  break;
        case 12: param_currentMode1.ky[2] = check_number();  break;
        case 13: param_currentMode1.ku[0] = check_number();  break;
        case 14: param_currentMode1.ku[1] = check_number();  break;
        case 15: param_currentMode1.ku[2] = check_number();  break;
        case 16: param_currentMode1.ku[3] = check_number();  break;
```

Archivo “main.c” (continuación)

```
// Parámetro Lazo Corriente Modo Corriente Media
        case 20: param_currentMode0.ky[0] = check_number();  break;
        case 21: param_currentMode0.ky[1] = check_number();  break;
        case 22: param_currentMode0.ky[2] = check_number();  break;
        case 23: param_currentMode0.ku[0] = check_number();  break;
        case 24: param_currentMode0.ku[1] = check_number();  break;
        case 25: param_currentMode0.ku[2] = check_number();  break;
        case 26: param_currentMode0.ku[3] = check_number();  break;

        // Parámetros Modo Tensión
        case 30: param_voltageMode.ky[0] = check_number();  break;
        case 31: param_voltageMode.ky[1] = check_number();  break;
        case 32: param_voltageMode.ky[2] = check_number();  break;
        case 33: param_voltageMode.ku[0] = check_number();  break;
        case 34: param_voltageMode.ku[1] = check_number();  break;
        case 35: param_voltageMode.ku[2] = check_number();  break;
        case 36: param_voltageMode.ku[3] = check_number();  break;

        // Tensión de referencia
        case 40: vref = check_number(); break;

        // Resistencia de carga
        case 50: rload = check_number();
                 duty2 = 2/rload - 0.26666;
                 EPwm3Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD * duty2;    break;

        // Frecuencia de conmutación del interruptor del Buck
        case 60: frequency = check_number() * 1000;
        EPwm1Regs.TBPRD = 60000000 / frequency - 1;; break;

        // Cambia el ciclo de trabajo (solo en lazo abierto)
        case 70: duty = check_number();
                 EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD * duty; break;

        // Modo Control Lazo Abierto
        case 80: controlMode = 0;
                 estado = 0;            break;

        // Modo Control Modo Tension
        case 81: controlMode = 1;
                 estado = 0;            break;

        // Modo Control Modo Corriente Media
        case 82: controlMode = 2;
                 estado = 0;            break;

        default:
            estado = 0;
            break;
    }
}
```

Archivo “main.c” (continuación)

```
// Transforma los datos guardados por check_sci en una variable float
float check_number(void)
{
    float number;
    int i = 0;

    // Transforma char a float
    number = (float) atof(savedData);

    // Reincia a 0 los datos números guardados
    while(i < numberpos)
    {
        savedData[i]= 0;
        i++;
    }

    stx_checked = 0;
    trama_num = 0;
    estado = 0;
    return number;
}

// Comprueba y guarda el número recibido por el puerto serie || Devuelve 1 si la trama no es correcta
bool check_sci(void)
{
    int i = 0;
    if(!stx_checked)
    {
        if(receivedData[0] != 0x02)     // Comprueba el primer valor de la trama
        {
            estado = 0;
            return 1;
        }
        else{
            stx_checked = 1;            // Indica que está comprobado la trama
            i++;                        // Salta al siguiente valor, el primero es el inicio de la trama
            numberpos = 0;
        }
    }

    for(i; i < 4; i++)                  // Guarda los números hasta llegar a 0x03 (Fin de número)
    {
        if(receivedData[i] == 0x03)
        {
            stx_checked = 0;
            mef_sci();
            return 0;
        }
```

Archivo “main.c” (continuación)

```
        savedData[numberpos] = receivedData[i];
        numberpos++;
    }

    return 0;
}

// Función que envía los datos por SCI a partir de un comando, parámetro y el
valor a enviar
void send_data(Uint16 comando, Uint16 parametro, float data)
{
    int i = 0;

    // Datos con el tipo de dato que se va a enviar
    trama_RX[0] = 0x02;
    trama_RX[1] = comando;
    trama_RX[2] = parametro;
    trama_RX[3] = 0x03;

    for(i = 0; i < 4; i++)
    {
        Sci_Envia_Caracter(trama_RX[i]);
    }

    //Envía el inicio de la trama numérica
    Sci_Envia_Caracter(0x02);

    // Transforma float a cadena de char
    snprintf(buffer, sizeof(buffer), "%f", data);

    // Envía los datos del ADC
    for(i = 0; i < 4; i++)
    {
        Sci_Envia_Caracter(buffer[i]);
    }

    // Envía carácter de fin de trama numérica
    Sci_Envia_Caracter(0x03);
}


// Función del regulador de corriente del control ACC
void regulador_corriente(void)
{
    // 10kHz Lazo de corriente
    param_currentMode0.y[2] = param_currentMode0.y[1];
    param_currentMode0.y[1] = param_currentMode0.y[0];

    param_currentMode0.u[2] = param_currentMode0.u[1];
    param_currentMode0.u[1] = param_currentMode0.u[0];
```

Archivo “main.c” (continuación)

```
  // Comparación con referencia del regulador de corriente
    param_currentMode0.u[0] = param_currentMode1.y[0] - (current[0] * 3.3 / 4096
/ 0.03);

    // Regulador de corriente
    param_currentMode0.y[0] = param_currentMode0.ky[0] * param_currentMode0.y[1]
+ param_currentMode0.ky[1] * param_currentMode0.y[2] + param_currentMode0.ku[0]
* param_currentMode0.u[0] + param_currentMode0.ku[1] * param_currentMode0.u[1] +
param_currentMode0.ku[2] * param_currentMode0.u[2];

    // Ciclo de trabajo limitado
    duty = constrain(param_currentMode0.y[0] / 3, 0.1, 0.45);

    // Aplicación del ciclo de trabajo
    EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD * duty;   // Initial duty 50%
}

// Función del regulador de tensión del control ACC
void regulador_tension(void)
{
    // 25kHz Lazo de tensión
    // Guarda las variables de entrada y salida necesarias
    param_currentMode1.y[2] = param_currentMode1.y[1];
    param_currentMode1.y[1] = param_currentMode1.y[0];

    param_currentMode1.u[2] = param_currentMode1.u[1];
    param_currentMode1.u[1] = param_currentMode1.u[0];

    // Comparación con referencia del regulador de corriente
    param_currentMode1.u[0] = (vref * 33 / 67) - (voltage[0] * 3.3 / 4096);

    // Regulador de corriente
    param_currentMode1.y[0] = param_currentMode1.ky[0] * param_currentMode1.y[1]
+ param_currentMode1.ky[1] * param_currentMode1.y[2] + param_currentMode1.ku[0]
* param_currentMode1.u[0] + param_currentMode1.ku[1] * param_currentMode1.u[1] +
param_currentMode1.ku[2] * param_currentMode1.u[2];
}

// Función del regulador del control modo Tensión
void regulador_modoTension(void)
{
    // Guarda las variables de entrada y salida necesarias
    param_voltageMode.y[3] = param_voltageMode.y[2];
    param_voltageMode.y[2] = param_voltageMode.y[1];
    param_voltageMode.y[1] = param_voltageMode.y[0];

    param_voltageMode.u[3] = param_voltageMode.u[2];
    param_voltageMode.u[2] = param_voltageMode.u[1];
    param_voltageMode.u[1] = param_voltageMode.u[0];
```

Archivo “main.c” (continuación)

```
// Comparación con referencia
    param_voltageMode.u[0] = (vref * 33 / 67) - (voltage[0] * 3.3 / 4096);

    // Regulador de tensión
    param_voltageMode.y[0] = param_voltageMode.ky[0] * param_voltageMode.y[1] +
param_voltageMode.ky[1] * param_voltageMode.y[2] + param_voltageMode.ky[2] *
param_voltageMode.y[3] + param_voltageMode.ku[0] * param_voltageMode.u[0] +
param_voltageMode.ku[1]* param_voltageMode.u[1] + param_voltageMode.ku[2] *
param_voltageMode.u[2] + param_voltageMode.ku[3] * param_voltageMode.u[3];

    // Limitación del ciclo de trabajo
    duty = constrain(param_voltageMode.y[0]/3, 0.1, 0.9);

    // Aplicación del ciclo de trabajo
    EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD * duty;   // Initial duty 50%
}
```

Archivo “setup.h”

```
/*********************************************************************
 *      Desarrollo y programación del control ACC de un convertidor Buck
 *                  mediante control digital implementado sobre el
 *                       DSP F28027 de Texas Instruments
 *
 *                                  setup.h
 *
 *                       Autor: Oscar Jia Xing Chen Zhu
 ********************************************************************/

#ifndef SETUP_H_
#define SETUP_H_

// Declaración de funciones
void Setup_Gpio(void);
void Setup_ePWM(void);
void config_ADC(void);
void Setup_Sci(void);

#endif /* SETUP_H_ */
```

Archivo “setup.c”

```
/**********************************************************************
 *      Desarrollo y programación del control ACC de un convertidor Buck
 *                  mediante control digital implementado sobre el
 *                       DSP F28027 de Texas Instruments
 *
 *                                  setup.c
 *
 *                      Autor: Oscar Jia Xing Chen Zhu
 **********************************************************************/

#include "DSP28x_Project.h"
#include "setup.h"

// Inicializa E/S
void Setup_Gpio(void)
{
    EALLOW;
    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1; // ePWM1A
    GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1; // ePWM1B
    GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1; // ePWM3A

    GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;  // GPIO2 --> salida (LED 2 en LaunchPad)
    GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;  // GPIO3 --> salida (LED 3 en LaunchPad)
    EDIS;

    GpioDataRegs.GPASET.bit.GPIO2 = 1;  // Apaga LED 2
    GpioDataRegs.GPASET.bit.GPIO3 = 1;  // Apaga LED 3

}

// Configuración ePWM1
void Setup_ePWM(void)
{

    // Configura ePWM1
    EPwm1Regs.TBCTL.bit.CLKDIV =  0;                // CLKDIV = 1
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;              // HSPCLKDIV = 1
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;      // Modo Up
    EPwm1Regs.TBPRD = 299;                          // 200kHz

    // Action Qualifier
    EPwm1Regs.AQCTLA.bit.ZRO= AQ_SET;
    EPwm1Regs.AQCTLA.bit.CAU= AQ_CLEAR;

    // Ciclo de trabjao
    EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD * 0.5;     // Initial duty 50%
```

Archivo “setup.c” (continuación)

```
    // Deaband
    EPwm1Regs.DBCTL.bit.HALFCYCLE = true;
    EPwm1Regs.DBCTL.bit.IN_MODE = 0;
    EPwm1Regs.DBCTL.bit.POLSEL = 2;
    EPwm1Regs.DBCTL.bit.OUT_MODE = 3;

    EPwm1Regs.DBRED = 7;
    EPwm1Regs.DBFED = 2;

    EPwm1Regs.TZCTL.all =  0;

    // Configuración ePWM3 -> Transistor de carga
    EPwm3Regs.TBCTL.bit.CLKDIV =  0;    // CLKDIV = 1
    EPwm3Regs.TBCTL.bit.HSPCLKDIV = 0;  // HSPCLKDIV = 1
    EPwm3Regs.TBCTL.bit.CTRMODE = 0;    // modo up
    EPwm3Regs.TBPRD = 299;              // 200kHz

    // Action Qualifier
    EPwm3Regs.AQCTLA.bit.ZRO= AQ_SET;
    EPwm3Regs.AQCTLA.bit.CAU= AQ_CLEAR;
    EPwm3Regs.CMPA.half.CMPA = EPwm3Regs.TBPRD * 0.5;     // Initial duty 0%

    EPwm3Regs.TZCTL.all =  0;
}

// Configuración ADCINA: 0, 1, 3, 7
void config_ADC(void)
{
    EALLOW;
    AdcRegs.ADCCTL1.bit.INTPULSEPOS = 1;        // ADCINT trips after AdcResults
latch

    AdcRegs.INTSEL1N2.bit.INT1E = 1;            // Enabled ADCINT1
    AdcRegs.INTSEL1N2.bit.INT1CONT = 0;         // Disable ADCINT1 Continuous
mode

    AdcRegs.INTSEL1N2.bit.INT1SEL = 3;          // Setup EOC2 to trigger ADCINT1
to fire

    AdcRegs.ADCSOC0CTL.bit.CHSEL = 9;           // Set SOC0 channel select to
ADCINB1
    AdcRegs.ADCSOC1CTL.bit.CHSEL = 11;          // Set SOC1 channel select to
ADCINB3
    AdcRegs.ADCSOC2CTL.bit.CHSEL = 13;          // Set SOC2 channel select to
ADCINB5
    AdcRegs.ADCSOC2CTL.bit.CHSEL = 3;           // Set SOC3 channel select to
ADCINA3
```

Archivo “setup.c” (continuación)

```
    AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 1;          // SOC0 Trigger Timer 0
    AdcRegs.ADCSOC1CTL.bit.TRIGSEL = 1;          // SOC1 Trigger Timer 0
    AdcRegs.ADCSOC2CTL.bit.TRIGSEL = 1;          // SOC2 Trigger Timer 0
    AdcRegs.ADCSOC3CTL.bit.TRIGSEL = 1;          // SOC3 Trigger Timer 0

    AdcRegs.ADCSOC0CTL.bit.ACQPS = 9;            // Set SOC0 S/H Window to 10 ADC
Clock Cycles, (9 ACQPS plus 1)
    AdcRegs.ADCSOC1CTL.bit.ACQPS = 9;            // Set SOC1 S/H Window to 10 ADC
Clock Cycles, (9 ACQPS plus 1)
    AdcRegs.ADCSOC2CTL.bit.ACQPS = 9;            // Set SOC2 S/H Window to 10 ADC
Clock Cycles, (9 ACQPS plus 1)
    AdcRegs.ADCSOC2CTL.bit.ACQPS = 9;            // Set SOC3 S/H Window to 10 ADC
Clock Cycles, (9 ACQPS plus 1)

    AdcRegs.SOCPRICTL.bit.SOCPRIORITY = 5;
    EDIS;
}

// Configuración SCI
void Setup_Sci(void)
{
    // Configura el SCI
    SciaRegs.SCICCR.all = 0x0007;   // 1 bit stop bit, loopback off
                                    // sin paridad, 8 bits
                                    // modo async, sin protocolo
    SciaRegs.SCICTL1.all = 0x0003;  // activa TX, RX, SCICLK interno,
                                    // desactiva RXERR, SLEEP, TXWAKE
    SciaRegs.SCICTL2.all = 0x0003;  // habilita ints RXRDY y TXRDY

    //            LSPCLK
    //  BRR = -------------- - 1
    //         8 x BAUDRATE

        SciaRegs.SCIHBAUD = 0x0000;  // 9600 baud @LSPCLK = 15MHz (60 MHz
SYSCLK)
        SciaRegs.SCILBAUD = 0x00C2;

    // Configura cola e interrupciones
    SciaRegs.SCIFFTX.bit.SCIFFENA = 1;  // habilita la cola
    SciaRegs.SCIFFRX.bit.RXFFIENA = 1;  // habilita la interrupcion de recepcion
    SciaRegs.SCIFFRX.bit.RXFFIL = 4;    // interrumpe cuando haya 4 caracteres
en cola

    // Saca el SCI del reset
    SciaRegs.SCICTL1.all = 0x0023;
}
```

2. Código Interfaz

Archivo "mainwindow.cpp"

```
/*****************************************************************************
 *      Desarrollo y programación del control ACC de un convertidor Buck
 *                  mediante control digital implementado sobre el
 *                       DSP F28027 de Texas Instruments
 *
 *                       Autor: Oscar Jia Xing Chen Zhu
 *****************************************************************************/

#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "sci.h"

#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow),
    settings_dialog(new Sci(this))
{
    ui->setupUi(this);

    // Deshabilita los menús de control al iniciar el programa
    ui->actionDisconnect->setEnabled(false);
    ui->loggroupBox->setEnabled(false);
    ui->groupBox_Control->setEnabled(false);

    // Configura el logger con los datos guardados
    logger = new Logger;
    ui->plainTextEdit->clear();
    ui->plainTextEdit->appendPlainText(logger->readData());

    // Conexiones entre QWidgets y funciones
    connect(ui->actionSerial_Settings, &QAction::triggered, this,
&MainWindow::openDialog);
    connect(ui->actionConnect, &QAction::triggered, this,
&MainWindow::openSerialPort);
    connect(ui->actionDisconnect, &QAction::triggered, this,
&MainWindow::closeSerialPort);
    connect(ui->actionSave_As, &QAction::triggered, this,
&MainWindow::fileDirectory);
    connect(ui->controModeBox,
QOverload<int>::of(&QComboBox::currentIndexChanged), this, [=](int
index){changeControlMode(index);});
    connect(ui->sendDuty, &QPushButton::clicked, this,
&MainWindow::changeDutyCycle);
    connect(ui->sendParamACC, &QPushButton::clicked, this,
&MainWindow::changeACCParams);
    connect(ui->sendParamVM, &QPushButton::clicked, this,
&MainWindow::changeVMParams);
    connect(ui->pushButton, &QPushButton::clicked, this,
&MainWindow::enviarParametros);
```

Archivo “mainwindow.cpp” (continuación)

```
connect(ui->lineEdit, &QLineEdit::returnPressed, this,
&MainWindow::enviardatos);
    connect(settings_dialog->getSerial_port(), &QSerialPort::readyRead,
this, &MainWindow::recibirdatos);
    connect(ui->HexCheckBox, SIGNAL(stateChanged(int)), this,
SLOT(hexChecked(int)));
}

MainWindow::~MainWindow()
{
    delete settings_dialog;
    delete ui;
}

// Abre el puerto serie según la configuración
void MainWindow::openSerialPort()
{
    if(settings_dialog->openPort())
    {
        ui->actionConnect->setEnabled(false);
        ui->actionDisconnect->setEnabled(true);
        ui->actionSerial_Settings->setEnabled(false);
        ui->loggroupBox->setEnabled(true);
        ui->groupBox_Control->setEnabled(true);
        ui->groupBox_BuckParams->setEnabled(true);
    } else {
        // Mensaje de error si no abre el puerto serie
        QMessageBox::critical(this, tr("Error"), settings_dialog-
>getSerial_port()->errorString());
    }

}

// Cierra el puerto serie abierto
void MainWindow::closeSerialPort()
{
    settings_dialog->closePort();

    ui->actionConnect->setEnabled(true);
    ui->actionDisconnect->setEnabled(false);
    ui->actionSerial_Settings->setEnabled(true);
    ui->loggroupBox->setEnabled(false);
    ui->groupBox_Control->setEnabled(false);
    ui->groupBox_BuckParams->setEnabled(false);
}
```

Archivo “mainwindow.cpp” (continuación)

```
// Envía datos por el puerto serie y lo escribe en el cuadro ****Por
modificar****
void MainWindow::enviardatos()
{
    QString text;

    if(ui->lineEdit->hasAcceptableInput())
    {
        settings_dialog->getSerial_port()->write(QByteArray::fromHex(ui-
>lineEdit->text().toLatin1()));
        text = logger->writeData(ui->lineEdit->text(), Logger::sendMode);
        ui->plainTextEdit->insertPlainText(text);
    } else {
        QMessageBox::critical(this, tr("Error"), "Carácteres introducidos
incorrectos");
    }
    ui->lineEdit->clear();
}

// Lee datos del puerto serie y lo escribe en el cuadro ****Por
modificar****
void MainWindow::recibirdatos()
{
    QString text;

    buffer += settings_dialog->getSerial_port()->readAll();

    if(buffer.size() < 10)   return;

    // Comprueba STX y ETX de la primera trama de 4 bytes
    if((buffer.at(0) == 0x02) && buffer.at(3) == 0x03)
    {
        // Comprueba parámetro de la primera trama de 4 bytes
        if(buffer.at(2) == '0' || buffer.at(2) == '1')
        {
            // Compruema comando de la primera trama de 4 bytes
            switch (buffer.at(1))
            {
            case 'V': estado = 10;  break;
            case 'I': estado = 20;  break;
            default:
                break;
            }

            estado += buffer.at(2) - '0';
            buffer.remove(0,4);
        }
    }

    qDebug() << "Buffer Size: " << buffer.size();
```

Archivo “mainwindow.cpp” (continuación)

```
// Comprueba que la longitud de la trama coincide con 6 bytes y que
existe STX y ETX
    if((buffer.size() >= 6) && (buffer.at(0) == 0x02) && (buffer.at(5) ==
0x03))
    {
        text.clear();
        for(int i = 1; i < 5; i++)
        {
            text.append(buffer.at(i));
        }

        // Asigna el tipo de dato recibido por el DSC
        switch (estado)
        {
            case 10:    ui->voltage_HVEdit->setText(text);
                        text.prepend( "Voltaje HV: "); break;
            case 11:    ui->voltage_LVEdit->setText(text);
                        text.prepend( "Voltaje LV: "); break;
            case 20:    ui->current_LVEdit->setText(text);
                        text.prepend( "Corriente LV: "); break;

            default:    estado = 0;
            break;
        }

        text = logger->writeData(text, Logger::receiveMode);
        ui->plainTextEdit->insertPlainText(text);
        buffer.remove(0,6);
        estado = 0;
        qDebug() << "Datos recibidos: " + text;
    }

    // Repite la función hasta que se termine los datos en el buffer
    while(buffer.size() >= 10)
    {
        recibirdatos();
    }
}

// Selecciona el directorio del archivo que registra el logger
void MainWindow::fileDirectory()
{
    logger->saveAs();
}

// Envía los parámetros del Buck (Frecuencia, Tensión Referencia y
Resistencia de carga variable)
void MainWindow::enviarParametros()
{
    QString text;
```

Archivo “mainwindow.cpp” (continuación)

```
    settings_dialog->sendNumber('F', '0', ui->frequencySwitchBox-
>value());
    settings_dialog->sendNumber('V', '0', ui->vRefBox->value());
    settings_dialog->sendNumber('R', '0', ui->rLoadBox->value());

    text = logger->writeData("Frecuencia de conmutación: " +
QString::number(ui->frequencySwitchBox->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("Tensión referencia: " + QString::number(ui-
>vRefBox->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("Resistencia de carga: " +
QString::number(ui->rLoadBox->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);

    QMessageBox::information(this, tr("Parámetros Buck"), "Información
enviada");
    qDebug() << "Datos enviados";

}

// Obliga al usuario a utilizar hexadecimal al enviar comandos a mano
void MainWindow::hexChecked(int state)
{
    if(state == 2){
        ui->lineEdit->setInputMask("HH HH HH HH;");
    }else{
        ui->lineEdit->setInputMask("");
    }
}

// Abre el menú de configuración del puerto serie
void MainWindow::openDialog(void)
{
    settings_dialog->show();
    settings_dialog->fillListPorts();

}

// Cambia el tipo de control del DSC (ACM o Lazo Abierto)
void MainWindow::changeControlMode(int mode)
{
    // Lazo abierto
    if(mode == 0) {
        ui->groupBox_NC->setEnabled(true);
        ui->groupBox_VM->setEnabled(false);
        ui->groupBox_ACC->setEnabled(false);
        settings_dialog->sendControl('C', '0');
        ui->plainTextEdit->insertPlainText(logger->writeData("Modo Lazo
Abierto", Logger::sendMode));
        qDebug() << "Lazo Abierto";
    }
```

Archivo “mainwindow.cpp” (continuación)

```
else if(mode == 1){
        ui->groupBox_NC->setEnabled(false);
        ui->groupBox_VM->setEnabled(true);
        ui->groupBox_ACC->setEnabled(false);
        settings_dialog->sendControl('C', '1');
        ui->plainTextEdit->insertPlainText(logger->writeData("Modo
Tensión", Logger::sendMode));
        qDebug() << "Modo Tensión";
    }

    // ACC
    else if(mode == 2){
        ui->groupBox_NC->setEnabled(false);
        ui->groupBox_VM->setEnabled(false);
        ui->groupBox_ACC->setEnabled(true);
        settings_dialog->sendControl('C', '2');
        ui->plainTextEdit->insertPlainText(logger->writeData("Modo
Corriente Media", Logger::sendMode));
        qDebug() << "Modo Corriente Media";
    }

}

// Envía los datos modificados del ciclo de trabajo (Lazo Abierto Solo)
void MainWindow::changeDutyCycle(void)
{
    QString text;

    settings_dialog->sendNumber('D', '0', ui->dutyBox->value());
    text = logger->writeData("Ciclo de trabajo: " + QString::number(ui-
>dutyBox->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    ui->dutyBox->clearFocus();
}

// Envías los parametros modificados de los reguladores de tensión y
corriente (ACM solo)
void MainWindow::changeACCParams(void)
{
    QString text;

    settings_dialog->sendNumber('A', '0', ui->yCurrentBox0->value());
    settings_dialog->sendNumber('A', '1', ui->yCurrentBox1->value());
    settings_dialog->sendNumber('A', '2', ui->yCurrentBox2->value());

    settings_dialog->sendNumber('A', '3', ui->uCurrentBox0->value());
    settings_dialog->sendNumber('A', '4', ui->uCurrentBox1->value());
    settings_dialog->sendNumber('A', '5', ui->uCurrentBox2->value());
    settings_dialog->sendNumber('A', '6', ui->uCurrentBox3->value());

    settings_dialog->sendNumber('P', '0', ui->yVoltageBox0->value());
    settings_dialog->sendNumber('P', '1', ui->yVoltageBox1->value());
    settings_dialog->sendNumber('P', '2', ui->yVoltageBox2->value());
```

Archivo “mainwindow.cpp” (continuación)

```
    settings_dialog->sendNumber('P', '3', ui->uVoltageBox0->value());
    settings_dialog->sendNumber('P', '4', ui->uVoltageBox1->value());
    settings_dialog->sendNumber('P', '5', ui->uVoltageBox2->value());
    settings_dialog->sendNumber('P', '6', ui->uVoltageBox3->value());

    text = logger->writeData("ACC -> Corriente y[k-1]: " +
QString::number(ui->yCurrentBox0->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Corriente y[k-2]: " +
QString::number(ui->yCurrentBox1->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Corriente y[k-3]: " +
QString::number(ui->yCurrentBox2->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);

    text = logger->writeData("ACC -> Corriente u[k]: " +
QString::number(ui->uCurrentBox0->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Corriente u[k-1]: " +
QString::number(ui->uCurrentBox1->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Corriente u[k-2]: " +
QString::number(ui->uCurrentBox2->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Corriente u[k-3]: " +
QString::number(ui->uCurrentBox3->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);

    text = logger->writeData("ACC -> Voltaje y[k-1]: " +
QString::number(ui->yVoltageBox0->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Voltaje y[k-2]: " +
QString::number(ui->yVoltageBox1->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Voltaje y[k-3]: " +
QString::number(ui->yVoltageBox2->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);

    text = logger->writeData("ACC -> Voltaje u[k]: " +
QString::number(ui->uVoltageBox0->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Voltaje u[k-1]: " +
QString::number(ui->uVoltageBox1->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Voltaje u[k-2]: " +
QString::number(ui->uVoltageBox2->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("ACC -> Voltaje u[k-3]: " +
QString::number(ui->uVoltageBox3->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
}
```

Archivo “mainwindow.cpp” (continuación)

```
// Envía lo parámetros del compensador del control modo Tensión
void MainWindow::changeVMParams(void)
{
    QString text;
    settings_dialog->sendNumber('L', '0', ui->yVMBox0->value());
    settings_dialog->sendNumber('L', '1', ui->yVMBox1->value());
    settings_dialog->sendNumber('L', '2', ui->yVMBox2->value());

    settings_dialog->sendNumber('L', '3', ui->uVMBox0->value());
    settings_dialog->sendNumber('L', '4', ui->uVMBox1->value());
    settings_dialog->sendNumber('L', '5', ui->uVMBox2->value());
    settings_dialog->sendNumber('L', '6', ui->uVMBox3->value());

    text = logger->writeData("Modo Tensión -> y[k-1]: " +
QString::number(ui->yVMBox0->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("Modo Tensión -> y[k-2]: " +
QString::number(ui->yVMBox1->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("Modo Tensión -> y[k-3]: " +
QString::number(ui->yVMBox2->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);

    text = logger->writeData("Modo Tensión -> Corriente u[k]: " +
QString::number(ui->uVMBox0->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("Modo Tensión -> Corriente u[k-1]: " +
QString::number(ui->uVMBox1->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("Modo Tensión -> Corriente u[k-2]: " +
QString::number(ui->uVMBox2->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
    text = logger->writeData("Modo Tensión -> Corriente u[k-3]: " +
QString::number(ui->uVMBox3->value()), Logger::sendMode);
    ui->plainTextEdit->insertPlainText(text);
}
```

Archivo “mainwindow.h”

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QSerialPort>
#include <QFileDialog>
#include <QtMath>
#include <QDebug>
#include "logger.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class Sci;

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:

    void openSerialPort(void);
    void closeSerialPort(void);
    void fileDirectory(void);

    void openDialog(void);
    void hexChecked(int state);

    void enviardatos(void);
    void recibirdatos(void);

    void enviarParametros(void);

    void changeControlMode(int mode);
    void changeDutyCycle(void);
    void changeACCParams(void);
    void changeVMParams(void);

private:
    Ui::MainWindow *ui = nullptr;
    Sci *settings_dialog = nullptr;
    Logger *logger;

    QByteArray buffer;
    int estado = 0;

};
#endif // MAINWINDOW_H
```

Archivo “mainwindow.ui”

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1054</width>
    <height>605</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QGroupBox" name="loggroupBox">
    <property name="enabled">
     <bool>false</bool>
    </property>
    <property name="geometry">
     <rect>
      <x>0</x>
      <y>0</y>
      <width>411</width>
      <height>521</height>
     </rect>
    </property>
    <property name="title">
     <string>Log</string>
    </property>
    <widget class="QCheckBox" name="HexCheckBox">
     <property name="geometry">
      <rect>
       <x>230</x>
       <y>490</y>
       <width>72</width>
       <height>19</height>
      </rect>
     </property>
     <property name="text">
      <string>Hex</string>
     </property>
    </widget>
    <widget class="QPlainTextEdit" name="plainTextEdit">
     <property name="geometry">
      <rect>
       <x>10</x>
       <y>30</y>
       <width>391</width>
```

Archivo “mainwindow.ui” (continuación)

```
<height>451</height>
      </rect>
     </property>
     <property name="readOnly">
      <bool>true</bool>
     </property>
     <property name="textInteractionFlags">
      <set>Qt::NoTextInteraction</set>
     </property>
    </widget>
    <widget class="QLineEdit" name="lineEdit">
     <property name="geometry">
      <rect>
       <x>90</x>
       <y>490</y>
       <width>113</width>
       <height>21</height>
      </rect>
     </property>
    </widget>
   </widget>
   <widget class="QGroupBox" name="groupBox_BuckParams">
    <property name="enabled">
     <bool>false</bool>
    </property>
    <property name="geometry">
     <rect>
      <x>420</x>
      <y>0</y>
      <width>251</width>
      <height>191</height>
     </rect>
    </property>
    <property name="title">
     <string>Parámetros Buck</string>
    </property>
    <property name="checkable">
     <bool>false</bool>
    </property>
    <widget class="QPushButton" name="pushButton">
     <property name="geometry">
      <rect>
       <x>80</x>
       <y>150</y>
       <width>80</width>
       <height>21</height>
      </rect>
     </property>
     <property name="text">
      <string>Enviar Datos</string>
```

Archivo “mainwindow.ui” (continuación)

```
</property>
    </widget>
    <widget class="QWidget" name="layoutWidget">
     <property name="geometry">
      <rect>
       <x>20</x>
       <y>40</y>
       <width>213</width>
       <height>91</height>
      </rect>
     </property>
     <layout class="QGridLayout" name="gridLayout_2">
      <item row="0" column="0">
       <widget class="QLabel" name="label_2">
        <property name="font">
         <font>
          <pointsize>10</pointsize>
         </font>
        </property>
        <property name="text">
         <string>Frecuencia</string>
        </property>
       </widget>
      </item>
      <item row="0" column="1" colspan="2">
       <spacer name="horizontalSpacer_4">
        <property name="orientation">
         <enum>Qt::Horizontal</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>40</width>
          <height>20</height>
         </size>
        </property>
       </spacer>
      </item>
      <item row="1" column="0">
       <widget class="QLabel" name="label_9">
        <property name="font">
         <font>
          <pointsize>10</pointsize>
         </font>
        </property>
        <property name="text">
         <string>Tensión</string>
        </property>
       </widget>
      </item>
      <item row="1" column="1" colspan="2">
```

Archivo “mainwindow.ui” (continuación)

```
<spacer name="horizontalSpacer_3">
        <property name="orientation">
         <enum>Qt::Horizontal</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>40</width>
          <height>20</height>
         </size>
        </property>
       </spacer>
      </item>
      <item row="2" column="0" colspan="2">
       <widget class="QLabel" name="label_10">
        <property name="font">
         <font>
          <pointsize>10</pointsize>
         </font>
        </property>
        <property name="text">
         <string>Resistencia Carga</string>
        </property>
       </widget>
      </item>
      <item row="2" column="2" colspan="2">
       <spacer name="horizontalSpacer_2">
        <property name="orientation">
         <enum>Qt::Horizontal</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>40</width>
          <height>20</height>
         </size>
        </property>
       </spacer>
      </item>
      <item row="2" column="4">
       <widget class="QDoubleSpinBox" name="rLoadBox">
        <property name="alignment">
         <set>Qt::AlignCenter</set>
        </property>
        <property name="keyboardTracking">
         <bool>true</bool>
        </property>
        <property name="suffix">
         <string> Ω</string>
        </property>
        <property name="minimum">
         <double>1.800000000000000</double>
```

Archivo “mainwindow.ui” (continuación)

```
</property>
         <property name="maximum">
          <double>7.500000000000000</double>
         </property>
         <property name="singleStep">
          <double>0.010000000000000</double>
         </property>
         <property name="value">
          <double>7.000000000000000</double>
         </property>
        </widget>
       </item>
       <item row="1" column="4">
        <widget class="QDoubleSpinBox" name="vRefBox">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="suffix">
          <string> V</string>
         </property>
         <property name="maximum">
          <double>9.000000000000000</double>
         </property>
         <property name="value">
          <double>1.000000000000000</double>
         </property>
        </widget>
       </item>
       <item row="0" column="4">
        <widget class="QDoubleSpinBox" name="frequencySwitchBox">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="specialValueText">
          <string/>
         </property>
         <property name="suffix">
          <string> kHz</string>
         </property>
         <property name="minimum">
          <double>50.000000000000000</double>
         </property>
         <property name="maximum">
          <double>500.000000000000000</double>
         </property>
         <property name="value">
          <double>200.000000000000000</double>
         </property>
        </widget>
       </item>
```

Archivo “mainwindow.ui” (continuación)

```
</layout>
    </widget>
   </widget>
   <widget class="QGroupBox" name="groupBox_6">
    <property name="geometry">
     <rect>
      <x>680</x>
      <y>0</y>
      <width>361</width>
      <height>191</height>
     </rect>
    </property>
    <property name="title">
     <string>Lecturas</string>
    </property>
    <widget class="QWidget" name="layoutWidget">
     <property name="geometry">
      <rect>
       <x>20</x>
       <y>40</y>
       <width>321</width>
       <height>131</height>
      </rect>
     </property>
     <layout class="QGridLayout" name="gridLayout">
      <item row="0" column="2">
       <widget class="QLabel" name="label_11">
        <property name="font">
         <font>
          <pointsize>10</pointsize>
         </font>
        </property>
        <property name="text">
         <string>Voltaje LV</string>
        </property>
        <property name="alignment">
         <set>Qt::AlignCenter</set>
        </property>
       </widget>
      </item>
      <item row="0" column="0">
       <widget class="QLabel" name="label_12">
        <property name="font">
         <font>
          <pointsize>10</pointsize>
         </font>
        </property>
        <property name="text">
         <string>Voltaje HV</string>
        </property>
```

Archivo “mainwindow.ui” (continuación)

```
<property name="alignment">
         <set>Qt::AlignCenter</set>
        </property>
       </widget>
      </item>
      <item row="1" column="2">
       <widget class="QLineEdit" name="voltage_LVEdit">
        <property name="alignment">
         <set>Qt::AlignCenter</set>
        </property>
        <property name="readOnly">
         <bool>true</bool>
        </property>
       </widget>
      </item>
      <item row="3" column="2">
       <widget class="QLineEdit" name="current_LVEdit">
        <property name="alignment">
         <set>Qt::AlignCenter</set>
        </property>
        <property name="readOnly">
         <bool>true</bool>
        </property>
       </widget>
      </item>
      <item row="2" column="2">
       <widget class="QLabel" name="label_14">
        <property name="font">
         <font>
          <pointsize>10</pointsize>
         </font>
        </property>
        <property name="text">
         <string>Corriente LV</string>
        </property>
        <property name="alignment">
         <set>Qt::AlignCenter</set>
        </property>
       </widget>
      </item>
      <item row="1" column="0">
       <widget class="QLineEdit" name="voltage_HVEdit">
        <property name="alignment">
         <set>Qt::AlignCenter</set>
        </property>
        <property name="readOnly">
         <bool>true</bool>
        </property>
       </widget>
      </item>
```

Archivo “mainwindow.ui” (continuación)

```
<item row="3" column="1">
       <spacer name="horizontalSpacer">
        <property name="orientation">
         <enum>Qt::Horizontal</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>40</width>
          <height>20</height>
         </size>
        </property>
       </spacer>
      </item>
     </layout>
    </widget>
   </widget>
   <widget class="QGroupBox" name="groupBox_Control">
    <property name="geometry">
     <rect>
      <x>420</x>
      <y>200</y>
      <width>631</width>
      <height>321</height>
     </rect>
    </property>
    <property name="title">
     <string>Control</string>
    </property>
    <widget class="QGroupBox" name="groupBox_NC">
     <property name="enabled">
      <bool>false</bool>
     </property>
     <property name="geometry">
      <rect>
       <x>30</x>
       <y>130</y>
       <width>121</width>
       <height>151</height>
      </rect>
     </property>
     <property name="title">
      <string>Lazo Abierto</string>
     </property>
     <property name="flat">
      <bool>false</bool>
     </property>
     <property name="checkable">
      <bool>false</bool>
     </property>
     <property name="checked">
```

Archivo “mainwindow.ui” (continuación)

```
<bool>false</bool>
     </property>
     <widget class="QWidget" name="layoutWidget">
      <property name="geometry">
       <rect>
        <x>10</x>
        <y>30</y>
        <width>101</width>
        <height>111</height>
       </rect>
      </property>
      <layout class="QGridLayout" name="gridLayout_5">
       <item row="0" column="0">
        <widget class="QLabel" name="label_16">
         <property name="font">
          <font>
           <pointsize>10</pointsize>
          </font>
         </property>
         <property name="text">
          <string>Ciclo de trabajo</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="1" column="0">
        <widget class="QDoubleSpinBox" name="dutyBox">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="buttonSymbols">
          <enum>QAbstractSpinBox::NoButtons</enum>
         </property>
         <property name="maximum">
          <double>1.000000000000000</double>
         </property>
         <property name="singleStep">
          <double>0.010000000000000</double>
         </property>
         <property name="value">
          <double>0.500000000000000</double>
         </property>
        </widget>
       </item>
       <item row="2" column="0">
        <widget class="QPushButton" name="sendDuty">
         <property name="text">
          <string>Enviar Datos</string>
```

Archivo “mainwindow.ui” (continuación)

```
</property>
        </widget>
       </item>
      </layout>
     </widget>
    </widget>
    <widget class="QLabel" name="label">
     <property name="geometry">
      <rect>
       <x>60</x>
       <y>50</y>
       <width>51</width>
       <height>21</height>
      </rect>
     </property>
     <property name="font">
      <font>
       <pointsize>10</pointsize>
      </font>
     </property>
     <property name="text">
      <string>Control</string>
     </property>
     <property name="alignment">
      <set>Qt::AlignCenter</set>
     </property>
    </widget>
    <widget class="QComboBox" name="controModeBox">
     <property name="enabled">
      <bool>true</bool>
     </property>
     <property name="geometry">
      <rect>
       <x>30</x>
       <y>80</y>
       <width>121</width>
       <height>21</height>
      </rect>
     </property>
     <property name="font">
      <font>
       <pointsize>10</pointsize>
      </font>
     </property>
     <property name="currentIndex">
      <number>2</number>
     </property>
     <item>
      <property name="text">
       <string>Lazo Abierto</string>
```

Archivo “mainwindow.ui” (continuación)

```
</property>
     </item>
     <item>
      <property name="text">
       <string>Modo Tensión</string>
      </property>
     </item>
     <item>
      <property name="text">
       <string>Corriente Media</string>
      </property>
     </item>
    </widget>
    <widget class="QGroupBox" name="groupBox_VM">
     <property name="enabled">
      <bool>false</bool>
     </property>
     <property name="geometry">
      <rect>
       <x>160</x>
       <y>20</y>
       <width>171</width>
       <height>281</height>
      </rect>
     </property>
     <property name="title">
      <string>Control Modo Tensión</string>
     </property>
     <widget class="QWidget" name="layoutWidget">
      <property name="geometry">
       <rect>
        <x>10</x>
        <y>20</y>
        <width>151</width>
        <height>251</height>
       </rect>
      </property>
      <layout class="QGridLayout" name="gridLayout_3">
       <item row="0" column="0" colspan="2">
        <widget class="QLabel" name="label_17">
         <property name="sizePolicy">
          <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
           <horstretch>0</horstretch>
           <verstretch>0</verstretch>
          </sizepolicy>
         </property>
         <property name="font">
          <font>
           <pointsize>10</pointsize>
          </font>
```

Archivo “mainwindow.ui” (continuación)

```
</property>
          <property name="text">
           <string>Parámetros</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
         </widget>
        </item>
        <item row="1" column="0">
         <widget class="QLabel" name="label_19">
          <property name="font">
           <font>
            <pointsize>9</pointsize>
           </font>
          </property>
          <property name="text">
           <string>y[k-1]</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
         </widget>
        </item>
        <item row="1" column="1">
         <widget class="QDoubleSpinBox" name="yVMBox0">
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
          <property name="decimals">
           <number>3</number>
          </property>
          <property name="maximum">
           <double>999.990000000000009</double>
          </property>
         </widget>
        </item>
        <item row="2" column="0">
         <widget class="QLabel" name="label_20">
          <property name="font">
           <font>
            <pointsize>9</pointsize>
           </font>
          </property>
          <property name="text">
           <string>y[k-2]</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
```

Archivo “mainwindow.ui” (continuación)

```
         </widget>
        </item>
        <item row="2" column="1">
         <widget class="QDoubleSpinBox" name="yVMBox1">
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
          <property name="decimals">
           <number>3</number>
          </property>
          <property name="maximum">
           <double>999.990000000000009</double>
          </property>
         </widget>
        </item>
        <item row="3" column="0">
         <widget class="QLabel" name="label_24">
          <property name="font">
           <font>
            <pointsize>9</pointsize>
           </font>
          </property>
          <property name="text">
           <string>y[k-3]</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
         </widget>
        </item>
        <item row="3" column="1">
         <widget class="QDoubleSpinBox" name="yVMBox2">
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
          <property name="decimals">
           <number>3</number>
          </property>
          <property name="maximum">
           <double>999.990000000000009</double>
          </property>
         </widget>
        </item>
        <item row="4" column="0">
         <widget class="QLabel" name="label_21">
          <property name="font">
           <font>
            <pointsize>9</pointsize>
           </font>
          </property>
```

Archivo “mainwindow.ui” (continuación)

```
    <property name="text">
           <string>u[k]</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
         </widget>
        </item>
        <item row="4" column="1">
         <widget class="QDoubleSpinBox" name="uVMBox0">
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
          <property name="decimals">
           <number>3</number>
          </property>
          <property name="maximum">
           <double>999.990000000000009</double>
          </property>
         </widget>
        </item>
        <item row="5" column="0">
         <widget class="QLabel" name="label_22">
          <property name="font">
           <font>
            <pointsize>9</pointsize>
           </font>
          </property>
          <property name="text">
           <string>u[k-1]</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
         </widget>
        </item>
        <item row="5" column="1">
         <widget class="QDoubleSpinBox" name="uVMBox1">
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
          <property name="decimals">
           <number>3</number>
          </property>
          <property name="maximum">
           <double>999.990000000000009</double>
          </property>
         </widget>
        </item>
        <item row="6" column="0">
```

Archivo “mainwindow.ui” (continuación)

```
<widget class="QLabel" name="label_23">
         <property name="font">
          <font>
           <pointsize>9</pointsize>
          </font>
         </property>
         <property name="text">
          <string>u[k-2]</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="6" column="1">
        <widget class="QDoubleSpinBox" name="uVMBox2">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="7" column="0">
        <widget class="QLabel" name="label_26">
         <property name="font">
          <font>
           <pointsize>9</pointsize>
          </font>
         </property>
         <property name="text">
          <string>u[k-3]</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="7" column="1">
        <widget class="QDoubleSpinBox" name="uVMBox3">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
```

Archivo “mainwindow.ui” (continuación)

```
<property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="8" column="0" colspan="2">
        <widget class="QPushButton" name="sendParamVM">
         <property name="text">
          <string>Enviar Datos</string>
         </property>
        </widget>
       </item>
      </layout>
     </widget>
    </widget>
    <widget class="QGroupBox" name="groupBox_ACC">
     <property name="enabled">
      <bool>true</bool>
     </property>
     <property name="geometry">
      <rect>
       <x>340</x>
       <y>20</y>
       <width>281</width>
       <height>281</height>
      </rect>
     </property>
     <property name="title">
      <string>Control Modo Corriente Media</string>
     </property>
     <widget class="QWidget" name="layoutWidget">
      <property name="geometry">
       <rect>
        <x>10</x>
        <y>20</y>
        <width>261</width>
        <height>251</height>
       </rect>
      </property>
      <layout class="QGridLayout" name="gridLayout_4">
       <item row="4" column="2">
        <widget class="QDoubleSpinBox" name="uCurrentBox0">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
```

Archivo “mainwindow.ui” (continuación)

```
  </property>
        </widget>
       </item>
       <item row="7" column="2">
        <widget class="QDoubleSpinBox" name="uCurrentBox3">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="2" column="2">
        <widget class="QDoubleSpinBox" name="yCurrentBox1">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="5" column="1">
        <widget class="QDoubleSpinBox" name="uVoltageBox1">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="4" column="1">
        <widget class="QDoubleSpinBox" name="uVoltageBox0">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
```

Archivo “mainwindow.ui” (continuación)

```
<property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="5" column="0">
        <widget class="QLabel" name="label_8">
         <property name="font">
          <font>
           <pointsize>9</pointsize>
          </font>
         </property>
         <property name="text">
          <string>u[k-1]</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="6" column="2">
        <widget class="QDoubleSpinBox" name="uCurrentBox2">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="8" column="0" colspan="3">
        <widget class="QPushButton" name="sendParamACC">
         <property name="text">
          <string>Enviar Datos</string>
         </property>
        </widget>
       </item>
       <item row="4" column="0">
        <widget class="QLabel" name="label_7">
         <property name="font">
          <font>
           <pointsize>9</pointsize>
          </font>
         </property>
         <property name="text">
          <string>u[k]</string>
         </property>
```

Archivo “mainwindow.ui” (continuación)

```
<property name="alignment">
        <set>Qt::AlignCenter</set>
       </property>
      </widget>
     </item>
     <item row="7" column="0">
      <widget class="QLabel" name="label_25">
       <property name="font">
        <font>
         <pointsize>9</pointsize>
        </font>
       </property>
       <property name="text">
        <string>u[k-3]</string>
       </property>
       <property name="alignment">
        <set>Qt::AlignCenter</set>
       </property>
      </widget>
     </item>
     <item row="5" column="2">
      <widget class="QDoubleSpinBox" name="uCurrentBox1">
       <property name="alignment">
        <set>Qt::AlignCenter</set>
       </property>
       <property name="decimals">
        <number>3</number>
       </property>
       <property name="maximum">
        <double>999.990000000000009</double>
       </property>
      </widget>
     </item>
     <item row="6" column="0">
      <widget class="QLabel" name="label_15">
       <property name="font">
        <font>
         <pointsize>9</pointsize>
        </font>
       </property>
       <property name="text">
        <string>u[k-2]</string>
       </property>
       <property name="alignment">
        <set>Qt::AlignCenter</set>
       </property>
      </widget>
     </item>
     <item row="2" column="1">
      <widget class="QDoubleSpinBox" name="yVoltageBox1">
```

Archivo “mainwindow.ui” (continuación)

```
<property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="0" column="2">
        <widget class="QLabel" name="label_4">
         <property name="font">
          <font>
           <pointsize>10</pointsize>
          </font>
         </property>
         <property name="text">
          <string>Corriente</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="3" column="2">
        <widget class="QDoubleSpinBox" name="yCurrentBox2">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="0" column="1">
        <widget class="QLabel" name="label_3">
         <property name="font">
          <font>
           <pointsize>10</pointsize>
          </font>
         </property>
         <property name="text">
          <string>Voltaje</string>
         </property>
         <property name="alignment">
```

Archivo “mainwindow.ui” (continuación)

```
<set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="3" column="0">
        <widget class="QLabel" name="label_18">
         <property name="font">
          <font>
           <pointsize>9</pointsize>
          </font>
         </property>
         <property name="text">
          <string>y[k-3]</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="1" column="1">
        <widget class="QDoubleSpinBox" name="yVoltageBox0">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="6" column="1">
        <widget class="QDoubleSpinBox" name="uVoltageBox2">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
       <item row="2" column="0">
        <widget class="QLabel" name="label_6">
         <property name="font">
          <font>
     <pointsize>9</pointsize>
```

Archivo “mainwindow.ui” (continuación)

```
</font>
          </property>
          <property name="text">
           <string>y[k-2]</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
         </widget>
        </item>
        <item row="1" column="0">
         <widget class="QLabel" name="label_5">
          <property name="font">
           <font>
            <pointsize>9</pointsize>
           </font>
          </property>
          <property name="text">
           <string>y[k-1]</string>
          </property>
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
         </widget>
        </item>
        <item row="1" column="2">
         <widget class="QDoubleSpinBox" name="yCurrentBox0">
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
          <property name="decimals">
           <number>3</number>
          </property>
          <property name="maximum">
           <double>999.990000000000009</double>
          </property>
         </widget>
        </item>
        <item row="7" column="1">
         <widget class="QDoubleSpinBox" name="uVoltageBox3">
          <property name="alignment">
           <set>Qt::AlignCenter</set>
          </property>
          <property name="decimals">
           <number>3</number>
          </property>
          <property name="maximum">
           <double>999.990000000000009</double>
          </property>
         </widget>
```

Archivo “mainwindow.ui” (continuación)

```
</item>
       <item row="0" column="0">
        <widget class="QLabel" name="label_13">
         <property name="font">
          <font>
           <pointsize>10</pointsize>
          </font>
         </property>
         <property name="text">
          <string>Parámetros</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
       </item>
       <item row="3" column="1">
        <widget class="QDoubleSpinBox" name="yVoltageBox2">
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
         <property name="decimals">
          <number>3</number>
         </property>
         <property name="maximum">
          <double>999.990000000000009</double>
         </property>
        </widget>
       </item>
      </layout>
     </widget>
    </widget>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menubar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>1054</width>
     <height>20</height>
    </rect>
   </property>
   <widget class="QMenu" name="menuArchivo">
    <property name="title">
     <string>Archivo</string>
    </property>
    <addaction name="actionSave_As"/>
   </widget>
   <addaction name="menuArchivo"/>
```

Archivo “mainwindow.ui” (continuación)

```
</widget>
  <widget class="QStatusBar" name="statusbar"/>
  <widget class="QToolBar" name="toolBar">
   <property name="windowTitle">
    <string>toolBar</string>
   </property>
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
   <addaction name="actionConnect"/>
   <addaction name="separator"/>
   <addaction name="actionDisconnect"/>
   <addaction name="separator"/>
   <addaction name="actionSerial_Settings"/>
   <addaction name="separator"/>
  </widget>
  <action name="actionConfigurate">
   <property name="icon">
    <iconset resource="resources.qrc">
     <normaloff>:/imagenes/settings-
gears.png</normaloff>:/imagenes/settings-gears.png</iconset>
   </property>
   <property name="text">
    <string>Configuración</string>
   </property>
   <property name="toolTip">
<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Configuración&lt;/p
&gt;&lt;/body&gt;&lt;/html&gt;</string>
   </property>
  </action>
  <action name="actionConnect">
   <property name="icon">
    <iconset resource="resources.qrc">
<normaloff>:/imagenes/connected.png</normaloff>:/imagenes/connected.png</
iconset>
   </property>
   <property name="text">
    <string>Connect</string>
   </property>
   <property name="toolTip">
<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Connect&lt;/p&gt;&l
t;/body&gt;&lt;/html&gt;</string>
   </property>
  </action>
  <action name="actionDisconnect">
   <property name="icon">
    <iconset resource="resources.qrc">
<normaloff>:/imagenes/remove.png</normaloff>:/imagenes/remove.png</iconse
t>
   </property>
   <property name="text">
    <string>Disconnect</string>
```

Archivo “mainwindow.ui” (continuación)

```
</property>
   <property name="toolTip">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Disconnect&lt;/p&gt
;&lt;/body&gt;&lt;/html&gt;</string>
   </property>
  </action>
  <action name="actionSerial_Settings">
   <property name="icon">
    <iconset resource="resources.qrc">

<normaloff>:/imagenes/setting.png</normaloff>:/imagenes/setting.png</icon
set>
   </property>
   <property name="text">
    <string>Serial Settings</string>
   </property>
   <property name="toolTip">
    <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Serial
Settings&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
   </property>
  </action>
  <action name="actionSave_As">
   <property name="text">
    <string>Guardar Como</string>
   </property>
  </action>
 </widget>
 <resources>
  <include location="resources.qrc"/>
 </resources>
 <connections/>
</ui>
```

Archivo “sci.cpp”

```
#include "sci.h"

Sci::Sci(QWidget *parent) :
    SettingsDialog(parent),
    serial_port(new QSerialPort(this))
{
}

Sci::~Sci()
{
    delete serial_port;
}

// Abre el puerto con la configuración deseada. Devuelve 1 si lo consigue
y 0 si falla
bool Sci::openPort()
{
    Settings current_setting = getCurrentSettings();
    serial_port->setPortName(current_setting.name);
    serial_port->setBaudRate(current_setting.baudRate);
    serial_port->setDataBits(current_setting.dataBits);
    serial_port->setParity(current_setting.parity);
    serial_port->setStopBits(current_setting.stopBits);
    serial_port->setFlowControl(QSerialPort::NoFlowControl);

    if(!serial_port->open(QIODevice::ReadWrite))
    {
        QMessageBox::critical(this, tr("Error"), serial_port-
>errorString());
        return false;
        qDebug() << "Error Puerto";
    }

    qDebug() << "Puerto Abierto";
    return true;
}

// Cierra el puerto si está abierto
void Sci::closePort()
{
    if(serial_port->isOpen())
        serial_port->close();
    qDebug() << "Puerto Cerrado";
}
```

Archivo “sci.cpp” (continuación)

```
// Envía el parámetro a modificar
void Sci::sendNumber(char command, char parameter, double number)
{
    QByteArray param, data;
    int n_etx;

    // Envía el parámetro que se va a modificar
    param.append(0x02);
    param.append(command);
    param.append(parameter);
    param.append(0x03);
    serial_port->write(param);

    // Envía el valor del parámetro modifcado
    data.append(0x02);
    data.append(QByteArray::number(number));

    // Calcula los espacios necesarios a rellenar de la última trama (4
bytes)
    n_etx = 4 - ((QString::number(number).length() + 1) % 4);

    // Rellena los espacios sobrantes con el carácter de fin de trama
(0x03)
    for(int i = 0; i < n_etx; i++)
        data.append(0x03);

    serial_port->write(data);
    qDebug()<< "qbytearray: " << data;
}

// Envía el tipo de control seleccionado
void Sci::sendControl(char command, char parameter)
{
    QByteArray data;
    data.append(0x02);
    data.append(command);
    data.append(parameter);
    data.append(0x03);
    serial_port->write(data);
}

QSerialPort *Sci::getSerial_port() const
{
    return serial_port;
}
```

Archivo “sci.h”

```
#ifndef SCI_H
#define SCI_H

#include "settingsdialog.h"
#include <QObject>
#include <QMessageBox>

class Sci : public SettingsDialog
{
    Q_OBJECT

public:
    explicit Sci(QWidget *parent = nullptr);
    ~Sci();

    QSerialPort *getSerial_port() const;
    void sendNumber(char command, char parameter, double number);
    void sendControl(char command, char parameter);

public slots:
    bool openPort();
    void closePort();

private:
    QSerialPort *serial_port = nullptr;
};

#endif // SCI_H
```

Archivo “settingsdialog.cpp”

```
#include "settingsdialog.h"
#include "ui_settingsdialog.h"

SettingsDialog::SettingsDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::SettingsDialog)
{
    // Rellena las configuraciones posibles del puerto serie
    ui->setupUi(this);
    fillDataBits();
    fillListPorts();
    fillParity();
    fillStopBits();

    // Si detecta que hay un puerto serie lo configura como
predeterminado
    if(ports.length() > 0)
    {
        fillBaudRates(0);
        ui->baudRateBox->setCurrentIndex(6);
        updateSettings();
    }
    connect(ui->applyButton, &QPushButton::clicked, this,
&SettingsDialog::apply);
}

SettingsDialog::~SettingsDialog()
{
    delete ui;
}

// Actualiza la configuración del puerto serie y esconde la ventana de
configuración
void SettingsDialog::apply()
{
    updateSettings();
    hide();
}

// Devuelve los parámetros de la configuración del puerto serie
SettingsDialog::Settings SettingsDialog::getCurrentSettings() const
{
    return currentSettings;
}

// Llenamos el combobox "serialPortbox" con los puertos disponibles
void SettingsDialog::fillListPorts()
{
    QString serialNumber;

    // Borramos el contenido del combobox
    ui->serialPortBox->disconnect();
    ui->serialPortBox->clear();
```

Archivo “settingsdialog.cpp”

```
// Borramos el contenido de la lista de puertos
    ports.clear();

    // Obtenemos los puertos disponibles
    ports = QSerialPortInfo::availablePorts();

    qDebug() << "Número puertos disponibles: " << ports.length();

    // Añadimos los datos al combobox
    for(int i = 0; i < ports.length(); i++)
    {
        serialNumber = ports[i].serialNumber();
        ui->serialPortBox->addItem(ports[i].portName());
        ui->baudRateBox->addItem(ports[i].portName());
    }

    // Actualiza los valores de baudios disponibles dependiendo del
puerto
    connect(ui->serialPortBox,
QOverload<int>::of(&QComboBox::currentIndexChanged), this, [=](int
index){fillBaudRates(index);});
}


// Muestra los baudios soportados por el objetivo en el combobox
"baudRateBox"
void SettingsDialog::fillBaudRates(int index)
{
    QList<qint32> baudRates;

    ui->baudRateBox->clear();

    baudRates = ports[index].standardBaudRates();

    for(int i = 0; i < baudRates.length(); i++)
    {
        ui->baudRateBox->addItem(QString::number(baudRates[i]) + " baud",
baudRates[i]);
    }
}

// Muestra la lista del tamaño de datos de cada trama en el combobo
"dataBitsBox"
void SettingsDialog::fillDataBits()
{
    ui->dataBitsBox->addItem(QStringLiteral("5"), QSerialPort::Data5);
    ui->dataBitsBox->addItem(QStringLiteral("6"), QSerialPort::Data6);
    ui->dataBitsBox->addItem(QStringLiteral("7"), QSerialPort::Data7);
    ui->dataBitsBox->addItem(QStringLiteral("8"), QSerialPort::Data8);

    ui->dataBitsBox->setCurrentIndex(3);
}
```

Archivo “settingsdialog.cpp”

```
// Muestra la lista de paridad en el combobox "parityBox"
void SettingsDialog::fillParity()
{
    ui->parityBox->addItem(QStringLiteral("No usar"),
QSerialPort::NoParity);
    ui->parityBox->addItem(QStringLiteral("Impar"),
QSerialPort::EvenParity);
    ui->parityBox->addItem(QStringLiteral("Par"),
QSerialPort::OddParity);
    ui->parityBox->addItem(QStringLiteral("Marcada"),
QSerialPort::MarkParity);
    ui->parityBox->addItem(QStringLiteral("Espaciada"),
QSerialPort::SpaceParity);
}

// Muestra la lista del n° de bits de parada en el combobox "stopBitsBox"
void SettingsDialog::fillStopBits()
{
    ui->stopBitsBox->addItem(QStringLiteral("1"), QSerialPort::OneStop);
    ui->stopBitsBox->addItem(QStringLiteral("1.5"),
QSerialPort::OneAndHalfStop);
    ui->stopBitsBox->addItem(QStringLiteral("2"), QSerialPort::TwoStop);
}

// Actualiza la configuración del puerto serie
void SettingsDialog::updateSettings()
{
    currentSettings.name = ui->serialPortBox->currentText();
    currentSettings.baudRate = static_cast<qint32>(ui->baudRateBox-
>currentData().toInt());
    currentSettings.dataBits = static_cast<QSerialPort::DataBits>(ui-
>dataBitsBox->currentData().toInt());
    currentSettings.parity = static_cast<QSerialPort::Parity>(ui-
>parityBox->currentData().toInt());
    currentSettings.stopBits = static_cast<QSerialPort::StopBits>(ui-
>stopBitsBox->currentData().toInt());

    qDebug() << "Current name: " << currentSettings.name;
    qDebug() << "Current baudRate: " <<
QString::number(currentSettings.baudRate);
    qDebug() << "Current dataBits: " <<
QString::number(currentSettings.dataBits);
    qDebug() << "Current parity: " <<
QString::number(currentSettings.parity);
    qDebug() << "Current stopBits: " <<
QString::number(currentSettings.stopBits);
}
```

Archivo “settingsdialog.h”

```
#ifndef SETTINGSDIALOG_H
#define SETTINGSDIALOG_H

#include <QDialog>
#include <QSerialPort>
#include <QSerialPortInfo>
#include <QDebug>

namespace Ui {
class SettingsDialog;
}

class SettingsDialog : public QDialog
{
    Q_OBJECT

public:

    struct Settings {
        QString name;
        qint32 baudRate;
        QSerialPort::DataBits dataBits;
        QSerialPort::Parity parity;
        QSerialPort::StopBits stopBits;
    };

    explicit SettingsDialog(QWidget *parent = nullptr);
    ~SettingsDialog();

    Settings getCurrentSettings() const;
    void fillListPorts();

private:

    void fillDataBits();
    void fillParity();
    void fillStopBits();
    void updateSettings();

private slots:
    void fillBaudRates(int);
    void apply();

protected:
    QList<QSerialPortInfo> ports;
    Settings currentSettings;

private:
    Ui::SettingsDialog *ui = nullptr;
};

#endif // SETTINGSDIALOG_H
```

Archivo “settingsdialog.ui”

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>SettingsDialog</class>
 <widget class="QDialog" name="SettingsDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>331</width>
    <height>311</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <widget class="QPushButton" name="applyButton">
   <property name="geometry">
    <rect>
     <x>200</x>
     <y>130</y>
     <width>80</width>
     <height>21</height>
    </rect>
   </property>
   <property name="text">
    <string>Aplicar</string>
   </property>
  </widget>
  <widget class="QWidget" name="layoutWidget">
   <property name="geometry">
    <rect>
     <x>60</x>
     <y>30</y>
     <width>91</width>
     <height>231</height>
    </rect>
   </property>
   <layout class="QFormLayout" name="formLayout">
    <item row="0" column="0">
     <widget class="QLabel" name="label">
      <property name="text">
       <string>Puerto</string>
      </property>
     </widget>
    </item>
    <item row="1" column="0">
     <widget class="QComboBox" name="serialPortBox"/>
    </item>
    <item row="2" column="0">
     <widget class="QLabel" name="label_2">
```

Archivo “settingsdialog.ui” (continuación)

```
<property name="text">
       <string>Baudios</string>
      </property>
     </widget>
    </item>
    <item row="3" column="0">
     <widget class="QComboBox" name="baudRateBox"/>
    </item>
    <item row="4" column="0">
     <widget class="QLabel" name="label_3">
      <property name="text">
       <string>N° Bits</string>
      </property>
     </widget>
    </item>
    <item row="5" column="0">
     <widget class="QComboBox" name="dataBitsBox"/>
    </item>
    <item row="6" column="0">
     <widget class="QLabel" name="label_4">
      <property name="text">
       <string>Paridad</string>
      </property>
     </widget>
    </item>
    <item row="7" column="0">
     <widget class="QComboBox" name="parityBox"/>
    </item>
    <item row="8" column="0">
     <widget class="QLabel" name="label_5">
      <property name="text">
       <string>Bits de parada</string>
      </property>
     </widget>
    </item>
    <item row="9" column="0">
     <widget class="QComboBox" name="stopBitsBox"/>
    </item>
   </layout>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

Archivo “logger.cpp”

```
#include "logger.h"

Logger::Logger(QObject *parent) : QObject(parent)
{
    file = new QFile;
    file->setFileName(fileName);
    qDebug() << "Finalizado";
}

Logger::~Logger()
{
    if(file != 0)
        file->close();
    delete file;
}

// Escribe los datos en un txt con la fecha-hora
QString Logger::writeData(QString data, bool i)
{
    QString text, command;

    if(i == receiveMode)    command = "Recibido: ";
    else    {   command = "Enviado: ";}

    file->open(QIODevice::WriteOnly|QIODevice::Text|QIODevice::Append);
    text = QDateTime::currentDateTime().toString("yyyy/MM/dd hh:mm:ss ")
+ command + data + "\n";

    //Escribe los datos
    QTextStream out(file);
    out << text;
    file->close();

    return text;    // Devuelve los datos con la fecha y hora y si es un
dato enviado o recibido
}

// Lee los datos guardados del logger y los imprime en el programa
QString Logger::readData()
{
    QString data ="";

    file->open(QIODevice::ReadOnly|QIODevice::Text);
    QTextStream in(file);

    // Lee los datos disponibles
    while (!in.atEnd()) {
        QString line = in.readLine();
        data += line + "\n";
    }
    file->close();

    return data;
}
```

Archivo “logger.cpp” (continuación)

```
// Guarda los datos del logger en un archivo
void Logger::saveAs()
{
    // Obtiene los datos almacenados actuales
    QString data = readData();

    // Modifica la ruta y el nombre del archivo
    fileName = QFileDialog::getSaveFileName(nullptr,tr("Guardar como"),
"log",
                                            tr("Archivo de texto (*.txt);;All
Files(*)"));
    file->setFileName(fileName);

    // Borra el contenido del nuevo archivo si existe
    if(file->exists())
        file->resize(0);

    // Reescribe los datos
    file->open(QIODevice::WriteOnly|QIODevice::Text|QIODevice::Append);
    QTextStream out(file);
    out << data;
    file->close();
    qDebug() << "fileName: " + fileName;
}
```

Archivo “logger.h”

```
#ifndef LOGGER_H
#define LOGGER_H

#include <QObject>
#include <QPlainTextEdit>
#include <QFile>
#include <QTextStream>
#include <QDateTime>
#include <QDebug>
#include <QFileDialog>

class Logger : public QObject
{
    Q_OBJECT
public:
    explicit Logger(QObject *parent = nullptr);
    ~Logger();

public slots:
    QString writeData(QString data, bool mode);
    QString readData();
    void saveAs(void);

public:
    enum mode{sendMode, receiveMode};

private:
    QFile *file = nullptr;
    QString fileName = "log.txt";

};

#endif // LOGGER_H
```

Archivo “main.cpp”

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Archivo “Interfaz_Control_Digital_Buck.pro”

```
QT       += core gui serialport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the
APIs deprecated before Qt 6.0.0

SOURCES += \
    logger.cpp \
    main.cpp \
    mainwindow.cpp \
    sci.cpp \
    settingsdialog.cpp

HEADERS += \
    logger.h \
    mainwindow.h \
    sci.h \
    settingsdialog.h

FORMS += \
    mainwindow.ui \
    settingsdialog.ui

# Default rules for deployment.
qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    resources.qrc
```