Prototipo a escala de un sistema estacionario para aprovechamiento de energía de frenado regenerativo para trenes eléctricos basado en ultracapacitores

Daniel VILLEGAS, Andrés E. DÍEZ

Trabajo de grado para optar al título de ingeniero electrónico

Andrés E. Diez
Ing. Electricista, Magíster en Transmisión y Distribución de Energía Eléctrica, Doctor en Ingeniería

Universidad Pontificia Bolivariana
Escuela de Ingenierías
Ingeniería electrónica
Pregrado
Medellín
2013

Dedicatoria

A mi familia.

Contenido

Lista de Figuras

Glosario

IGBT. Insulated Gate Bipolar Transistor o en español Transistor Bipolar de Compuerta Aislada, es un dispositivo semiconductor de potencia de tres terminales, se utiliza en situaciones donde se requiere conmutación controlada por voltaje.

ADC. Analog to Digital Converter, conversor análogo a digital en español.

Filtro LC. Es un circuito electrónico pasivo conformado por un condensador y una bobina.

PWM: Pulse Width Modulation, Modulación por Ancho de Pulso.

Filtro RC: Es un circuito electrónico pasivo conformado por un condensador y una resistencia.

RS-232: Recommended Standard -232.

BDM: Background Debug Mode. Es un protocolo para programar microcontroladores utilizado por *Freescale*™.

Bus: Hace referencia a un conductor que se encuentra a un potencial con respecto a una referencia de voltaje, de este conductor se conectan equipos eléctricos en paralelo.

Apagado: Cuando se hable de que un *IGBT* u otro dispositivo electrónico de potencia se encuentran apagados se hace referencia a que la resistencia entre el colector y el emisor es muy grande o que no conduce corriente a través de estas terminales (o de las respectivas terminales del dispositivo).

Encendido: Cuando se hable de que un *IGBT* u otro dispositivo electrónico de potencia se encuentran encendidos se hace referencia a que la resistencia entre el colector y el emisor es muy pequeña y por lo tanto permite que exista una corriente a través de estas terminales (o de las respectivas terminales del dispositivo).

Controlador PI: Sistema de control proporcional-integral.

Resumen

Desde hace algunos años, muchas ciudades han optado por utilizar medios de transporte masivos eléctricos. El frenado regenerativo les da a estos sistemas la posibilidad de reutilizar parte de la energía que se pierde en el frenado, energía que normalmente sería transformada en calor. En este proyecto se implementará un sistema estacionario para el almacenamiento de energía del frenado regenerativo, cuyo prototipo ha contribuido a la investigación que desarrollan la Universidad Pontificia Bolivariana y el Metro de Medellín, para la implementación de este tipo de sistemas en la red de trenes Metropolitanos.

Palabras clave: Sistema estacionario de frenado regenerativo, frenado regenerativo, tren eléctrico, conversor de corriente directa, ultracapacitor, sistema eléctrico de transporte masivo.

Abstract

Electrical mass transportation systems are widely used today in modern cities. Regenerative braking gives to these systems the possibility of using part of the energy of the braking, energy that would be turned into heat in absence of this system. Here a small scale prototype of a stationary-storage regenerative braking system is designed and implemented; the prototype was used as part of a research between Universidad Pontificia Bolivariana and Metro Medellín, for the implementation of this kind of systems in the Metropolitan train network.

Keywords: Stationary-storage regenerative braking system, regenerative braking, electric train, DC-DC converter, ultracapacitor, electric mass transportation system.

INTRODUCCIÓN

En los últimos años se ha buscado aumentar cada vez más la eficiencia de los sistemas de transporte; éste es un problema que por sus características ha sido abordado desde diversas áreas de la ingeniería como la eléctrica, la electrónica, la mecánica, la aerodinámica y más recientemente, la nanotecnología. Con los años se han detectados y mejorado varias características de estos sistemas, estas mejoras consisten en identificar puntos donde existan perdidas de energía y realizar los cambios necesarios para disminuirlas, En este trabajo se tratará una característica que comparten todos los sistemas de transporte, que ya había sido identificada por la ciencia y la ingeniería como uno de los puntos donde más energía se pierde, el frenado.

La energía cinética es la energía que tiene un cuerpo en virtud de su velocidad y masa, tradicionalmente el proceso de frenado se ha llevado a cabo mediante sistemas regenerativos que permiten retornar parte al sistema de alimentación de los vehículos hasta que el sistema deja de ser receptivo a la misma (se limita por el voltaje), y entonces la energía en exceso termina siendo disipada principalmente en calor. Aprovechar esta energía siempre ha presentado retos tecnológicos, porque por un lado los motores de combustión interna no tienen la capacidad de recuperar de manera directa dicha energía y los motores eléctricos que si permiten recuperar la energía cinética y volverla a transformar en energía eléctrica, encuentran restricciones en los medios de almacenamiento de energía eléctrica, que en la actualidad no permiten almacenar grandes cantidades de energía ni hacerlo de manera rápida; y como ya se mencionó, en el caso de vehículos que operan conectados a la red, ésta recepción está restringida por los niveles de tensión. El desarrollo de los condensadores de doble capa o ultracapacitores y las recientes mejoras en las baterías parecen ofrecer una solución para los problemas anteriormente mencionados.

En este proyecto se pretende implementar a escala un conversor de corriente directa que permita realizar experimentos de frenado regenerativo en laboratorio, con finalidades pedagógicas, de investigación y de entrenamiento para ingeniero. Esto se hace como apoyo a la investigación en frenado regenerativo que desarrollan en conjunto la Universidad Pontificia Bolivariana con el Metro de Medellín, donde se pretende estudiar la pertinencia del uso de estos sistemas en la red especifica del metro de Medellín.

En la primera sección se hace una introducción a los modelos de los conversores *buck* y *boost* de corriente directa, en la segunda sección se retoman los modelos presentados en la primera sección y se discute el comportamiento del conversor conectado al *bus de DC* y como debe ser el sistema de control para lograr una compensación en corriente directa. En la tercera sección se presenta un procedimiento para el diseño del conversor de corriente directa. En las secciones 4, 5 y 6 se presentan los circuitos que se implementaron y sus características. En la sección 7 se presenta el software de control y como fue diseñado. Por

ultimo en la sección 8 se presentan una serie de consideraciones de seguridad que surgen como conclusión de todo el trabajo experimental que se realizó.

1. Conversores de corriente directa

Los conversores de corriente directa o conversores *DC-DC* son circuitos electrónicos que se utilizan para aumentar o disminuir un voltaje en corriente directa, puede entenderse como el análogo del transformador en corriente alterna. Existen varios tipos de conversores de corriente directa, sin embargo aquí solo se trataran tres tipos, el conversor *boost*, que sirve para aumentar el voltaje, el conversor *buck* que sirve para disminuir el voltaje y el conversor *buck-boost* bidireccional que es una combinación de los anteriores y permite aumentar el voltaje en una dirección y disminuirlo en la dirección contraria. Debido a que en sistemas de tracción de flujos reversibles de energía, entre sistemas de diferente nivel de voltaje, este tipo de convertidores presenta un especial interés.

1.1. Conversor buck de corriente directa

El conversor *buck* de corriente directa se utiliza para llevar a cabo reducciones de voltaje. En la Figura 1 se presenta el diagrama del circuito conversor *DC-DC buck.*

El sistema de conmutación está conformado por un *IGBT* u otro transistor de potencia y un diodo, el *IGBT* controla el paso de corriente desde la fuente hacia el filtro LC y la carga, el diodo permite la descarga de la energía almacenada en la bobina mientras el *IGBT* se encuentra apagado (Erickson & Maksimovic, 2001).

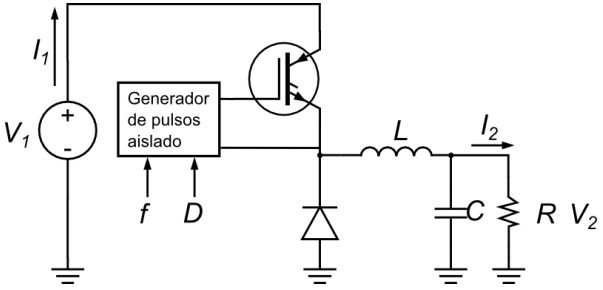


Figura 1. Diagrama del conversor *DC-DC buck*

El circuito opera a una frecuencia *f* con un ciclo útil *D*, donde *D* es un número positivo entre cero y uno. Idealmente el voltaje a la entrada del conversor, o sea V_1 es reducido por un factor *D*

$$V_2 = DV_1. \tag{1}$$

En realidad los semiconductores, la bobina y los conductores involucrados en el circuito tienen asociada una resistencia que produce calentamiento, pérdidas de energía y una disminución en el voltaje de salida, el voltaje de salida puede ser modelado de manera más exacta mediante la expresión

$$V_2 = \frac{R}{(R+R_l)} DV_1, \tag{2}$$

donde *R* es la resistencia de la carga y R_l la resistencia asociada a los elementos del circuito por perdidas de energía.

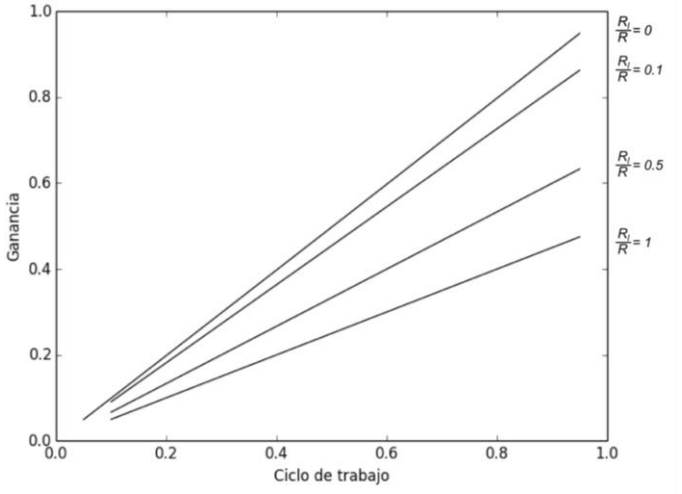


Figura 2. Ganancia vs ciclo de trabajo en el conversor buck

En la Figura 2 se presenta la relación entre la ganancia del conversor *buck* y el ciclo de trabajo para varios valores de resistencias R_l y R, esta grafica se obtuvo realizando varias simulaciones en el programa PSCAD™.

1.2. Conversor boost de corriente directa

El conversor *boost* se utiliza para aumentar el voltaje, en la Figura 3 se presenta el esquema del circuito.

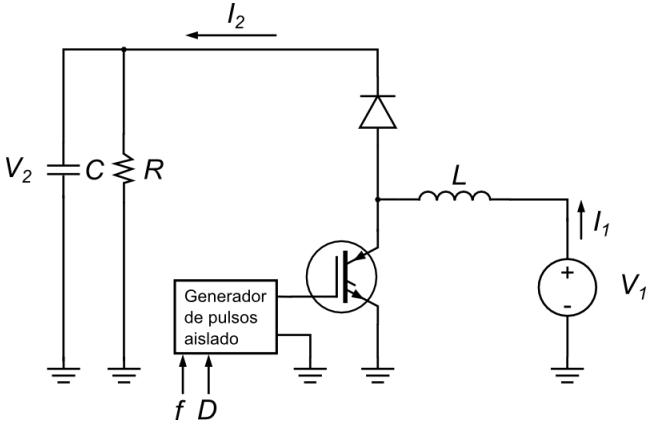


Figura 3. Diagrama del conversor DC-DC boost

El sistema de conmutación está conformado por los mismos dispositivos que conforman el sistema de conmutación del conversor *buck* pero dispuestos de manera diferente. En el conversor *boost* el *IGBT* carga la bobina mientras se encuentra encendido, al apagarlo la energía almacenada en la bobina se descarga a través del diodo (Erickson & Maksimovic, 2001).

Idealmente el voltaje a la salida, V_2 y el voltaje a la entrada, V_1 se relacionan mediante la siguiente expresión

$$V_2 = \frac{1}{1-D} V_1, \qquad (3)$$

D es el ciclo de trabajo del *IGBT*.

En el caso del conversor *boost* también influye la resistencia R_l asociada a las perdidas en el voltaje de salida del circuito, la siguiente expresión considera el efecto producido por dicha resistencia

$$V_2 = \frac{1}{(1-D)(1+\frac{R_l}{R(1-D)})} V_1, \quad (4)$$

R es la resistencia de la carga.

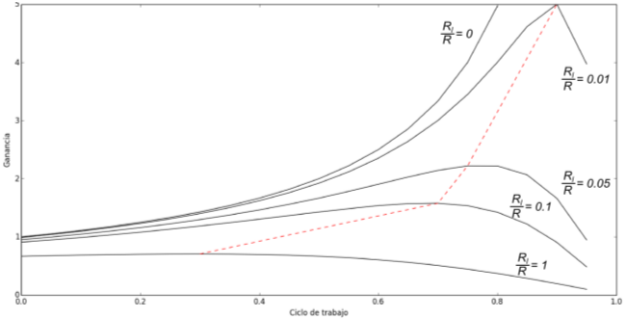


Figura 4. Ganancia vs ciclo de trabajo en el conversor *boost* y curva de valores máximos para cada configuración

A continuación se presentan dos expresiones que muestran la relación entre el valor máximo de la ganancia en un conversor boost de corriente directa y la proporción entre la resistencia serie de la bobina R_l y la resistencia de la carga *R*.

La ganancia máxima de un conversor de corriente directa es

$$A = \frac{1}{2}\sqrt{\frac{R}{R_l}}, \quad (5)$$

donde *A* es la ganancia máxima del conversor y ocurre cuando el ciclo útil de la señal de conmutación del *IGBT* es

$$D = 1 - \sqrt{\frac{R_l}{R}}. \quad (6)$$

1.3. Conversor buck-boost bidireccional de corriente directa

Como puede verse en la Figura 5, gracias a la similitud que existe en la topología de los circuitos conversores de corriente *buck* y *boost*, es posible unirlos para crear el conversor *buck-boost* bidireccional (Killer, 2012). Las ecuaciones que modelan el comportamiento de este circuito son iguales a las de los dos conversores mencionados anteriormente, a pesar de esto hay ciertas consideraciones de diseño.

- Las tierras de los generadores de pulsos deben ser independientes para evitar hacer un corto circuito a través de la tierra del circuito generador de pulsos del *IGBT* de la sección *buck.*
- Los *IGBT* no pueden encenderse de manera simultánea ya que esto causaría un corto circuito.

- Existe riesgo de hacer cortos circuitos a través de la tierra de sensores e instrumentos de medida.

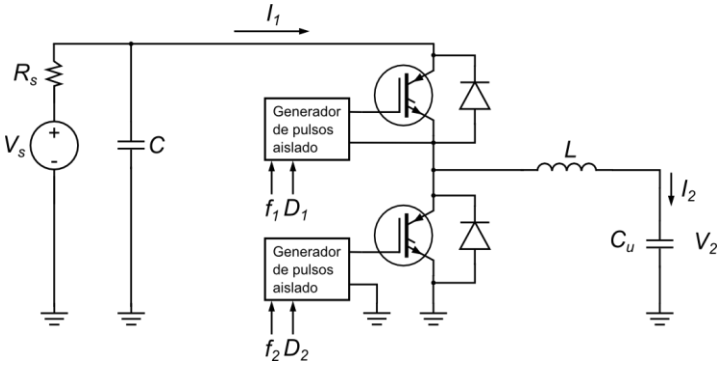


Figura 5. Diagrama del conversor *DC-DC buck-boost* bidireccional

2. Control de compensación de voltaje en el *bus* de corriente directa

En esta sección se presenta una propuesta de sistema de control para la compensación de voltaje en el *bus de DC* y un control para el voltaje en el ultracapacitor, además se hablarán de algunos criterios utilizados en el diseño de los controladores.

Para modelar el sistema deben diferenciarse dos casos fundamentales, el primero, cuando actúa el conversor buck y el segundo, cuando actúa el conversor boost. Para el caso particular de este sistema es difícil lograr un modelo que caracterice de manera general su comportamiento, aquí se presentarán algunos casos particulares que ayuden a entender el comportamiento del sistema.

El sistema tiene dos variables de entrada o acciones de control, que corresponden al ciclo de trabajo de la señal de cada uno de los dos *IGBT*. Estas dos variables son números reales entre 0 y 1 o entre 0% y 100%. Una de las características de funcionamiento de este circuito es que siempre que uno de los dos *IGBT* se encuentre en estado de conmutación (o sea conmutando entre encendido y apagado a cierta frecuencia) el otro *IGBT* estará apagado, esta característica permite que las dos variables se unan en una que toma valores entre -1 y 1 o entre -100% y 100%, cuando la acción de control es positiva el dispositivo funciona como conversor *buck* y cuando la acción de control es negativa el dispositivo funciona como conversor *boost*.

La salida del sistema es el voltaje en el *bus* de corriente directa medido justamente a la salida del sistema de compensación de corriente directa (es importante tener en cuenta que para algunos sistemas la resistencia del *bus de DC* no es despreciable, particularmente cuando hay corrientes altas y distancias grandes).

Además de las variables de entrada y salida del sistema existen diversos factores que influyen en el comportamiento del sistema como tal, por ejemplo para un mismo ciclo de trabajo la corriente que ingresa al conversor *buck* desde el *bus de DC* varía, debido a

que a medida que el nivel de carga del ultracapacitor aumenta la corriente disminuye. A continuación se presentan los factores principales que influyen en el modelo.

- Número de trenes frenando y acelerando simultáneamente en la línea.
- Peso de cada tren, esto depende del número de pasajero y por lo tanto de la hora a la que ocurre.
- Voltaje del ultracapacitor.
- Parámetros del *bus de DC* y de la fuente.
- Distancia a la que frena cada tren, pues esto modifica la topología de la red.

Según el número de trenes que frenan y aceleran simultáneamente el voltaje del *bus de DC* aumenta o disminuye. Un tren frenando entrega energía a un tren acelerando, y en general varios trenes frenando entregan energía al conjunto de trenes que están acelerando. Bajo estas condiciones se distinguen tres situaciones diferentes:

- La energía que entregan los trenes que frenan es mayor a la energía que consumen los trenes que aceleran más las pérdidas, en este caso el voltaje del *bus de DC* aumenta.
- La energía que entregan los trenes que frenan es menor a la energía que consumen los trenes que aceleran más las pérdidas, en este caso el voltaje del *bus de DC* disminuye.
- La energía que entregan los trenes que frenan es igual a la energía que consumen los trenes que aceleran más las pérdidas, en este caso el voltaje del *bus de DC* permanece igual al voltaje de la fuente, es muy poco probable que este caso ocurra por intervalos largos de tiempo, cuando ocurre el aprovechamiento de energía es máximo y el compensador no actúa.

Es posible obtener un modelo lineal de un sistema de tracción, por ejemplo en (Iannuzzi, Pighetti, & Tricoli , 2010) se presenta un modelo en el cual los trenes se presentan como fuentes de corriente, las estaciones de tracción como fuentes de voltaje reales (o sea en serie con una resistencia), el sistema estacionario de almacenamiento para frenado regenerativo es presentado como una fuente de corriente y el *bus de DC* como resistencias que varían su valor según la posición de los trenes. Teniendo en cuenta lo anterior, es posible encontrar un circuito equivalente en las terminales del sistema de frenado regenerativo.

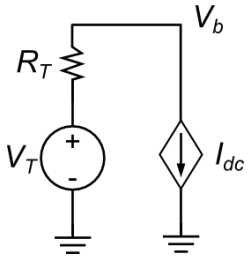


Figura 6. Modelo del sistema de frenado regenerativo conectado al sistema de tracción.

En la Figura 6 se presenta un modelo del sistema donde se relaciona la corriente que entra al conversor de corriente directa y el voltaje del *bus de DC*. V_T y R_T conforman el circuito equivalente formado por los trenes, las estaciones de tracción, V_b es el voltaje del *bus de DC* e I_{dc} es la corriente que entra al conversor de corriente directa (sí la corriente es negativa significa que la corriente no entra sino que sale), la expresión

$$V_b = V_T - R_T I_{dc}(V_{ultacap}, D) \quad (7)$$

describe el comportamiento de este modelo.

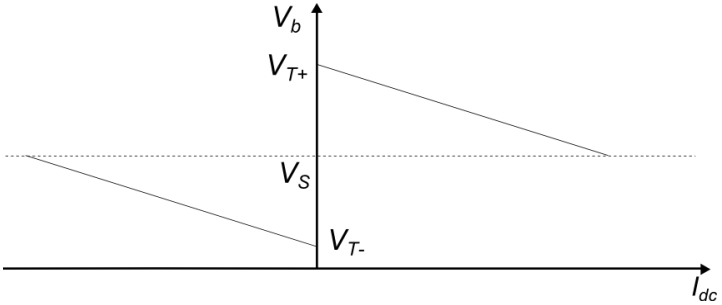


Figura 7. Relación entre voltaje y corriente del modelo del sistema.

En la Figura 7 se presenta la relación entre voltaje y corriente, en el semieje positivo de la corriente puede verse el comportamiento del sistema cuando actúa el conversor *buck*, en este caso V_T es mayor a V_s que es el voltaje promedio, en la figura aparece como V_{T+} para diferenciarlo. En el semieje negativo de la corriente se presenta el comportamiento del sistema cuando actúa el conversor *boost*, esta situación se presenta cuando el voltaje V_T es menor a V_S, en este caso se presenta V_T como V_{T-}.

2.1. Comportamiento del sistema cuando actúa el conversor buck

El objetivo de esta sección es presentar una descripción del comportamiento del sistema de frenado regenerativo cuando está actuando el conversor *buck*.

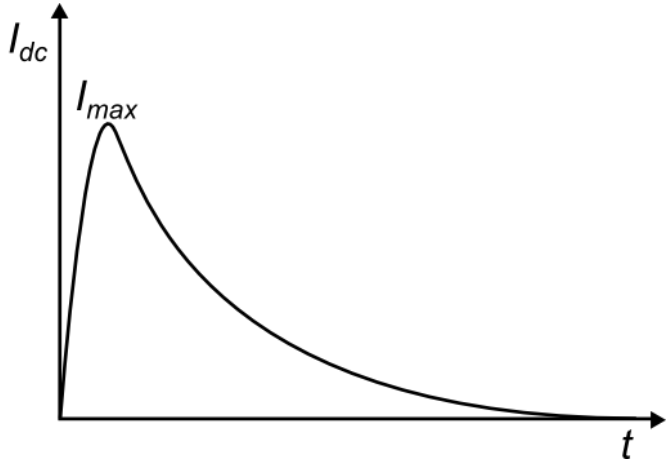


Figura 8. Respuesta en estado transitorio de la corriente que entra al conversor de corriente directa en modo *buck*

En este caso no es posible establecer una relación entre el ciclo de trabajo de la señal de conmutación y la corriente ya que la corriente que ingresa al conversor en estado estable es cero o casi cero. En cambio se presentará un modelo simplificado que ayude a entender el comportamiento de la corriente en el tiempo y con respecto al ciclo de trabajo.

En la Figura 8 se presenta el comportamiento de la corriente que entra al conversor, I_{max} es el valor máximo de corriente que toma la señal, este valor aumenta proporcionalmente con el ciclo de trabajo D, es necesario aclarar que en la Figura 8 solo se presenta el comportamiento promedio de la corriente, en realidad la corriente es una señal conmutada de alta frecuencia. De esta respuesta y considerando el modelo del conversor *buck* que va a ser utilizado puede intuirse que la velocidad a la cual decrece la corriente que entra al sistema disminuye con la capacitancia del ultracapacitor, esto se comprobó mediante simulaciones y experimentalmente, en la Figura 9 se presenta una simulación como ejemplo a la situación anterior, la curva de color verde fue generada utilizando un condensador de 3 F, la azul con un condensador de 1 F.

Para explicar el comportamiento de la corriente que ingresa al conversor de corriente directa hay que considerar el comportamiento del voltaje en el ultracapacitor. Como se presenta en la Figura 5, al establecer el ciclo de trabajo D_1 en un valor mayor a 0 y menor a 1, el voltaje V_2 en las terminales del ultracapacitor aumentará a razón de $\frac{i_2(t)}{c_u}$, la carga termina cuando $V_2 = D_1 V_1$, V_1 no se muestra de manera explícita pero se refiere al voltaje en las terminales del conversor *DC-DC*, idealmente cuando la carga termina no ingresa corriente al sistema. En este punto se conoce que la corriente inicia en un cero, luego de iniciar la carga debe alcanzar un punto máximo y luego debe disminuir hasta terminar en cero, además que el sistema debe ser de orden mayor o igual a dos pues tiene por lo menos dos elementos que

almacenan energía (ultracapacitor y bobina), esto permite concluir que el comportamiento de la corriente debe ser como se muestra en las Figuras 8 y 9.

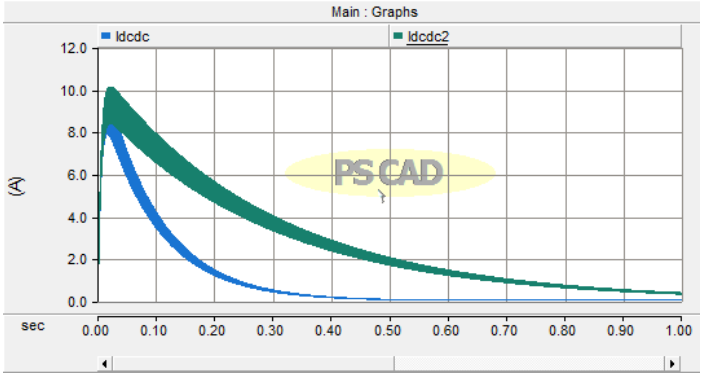


Figura 9. Simulación del conversor buck en estado transitorio

2.2. Comportamiento del sistema cuando actúa el conversor boost

Para el caso del conversor *boost* no es tan relevante considerar comportamiento del sistema en un estado transitorio, en cambio se mostrará cómo se relaciona el voltaje del *bus de DC* con respecto al ciclo de trabajo del conversor.

Primero es necesario revisar la Figura 4, en esta se muestra el comportamiento general de un conversor *boost*, con base en esto se debe analizar cómo sería el comportamiento del mismo conectado a un *bus de DC*. En el modelo general del conversor *boost* no es tenido en cuenta el hecho de que el sistema está conectado por medio de un *bus de DC* a una fuente que entrega energía y unos trenes que consumen. En la Figura 3 se presenta el esquema del conversor *boost*, puede verse que el diodo es el elemento que permite el paso de corriente desde el conversor hacia el *bus de DC*, el diodo solo conduce cuando se encuentra polarizado directamente, para esto es necesario que la corriente almacenada en la bobina sea lo suficientemente grande, en la expresión (4) se presenta la relación entre el los voltajes de entrada, salida y el ciclo de trabajo del conversor *boost*, de manera intuitiva puede verse qué el diodo va a estar directamente polarizado cuando $f(D_2)V_2 > V_s$ donde V_s es el voltaje medido en las terminales del conversor de corriente directa y el *bus de DC* mientras el conversor se encuentra apagado (o sea que el conversor no consume ni entrega corriente) y $f(D_2)$ es la función que representa la ganancia de voltaje del conversor, esta se muestra en (4). En la Figura 10 se presenta un modelo aproximado del comportamiento del voltaje del *bus de DC* con respecto al ciclo útil de conversor *boost*.

Cuando el conversor *boost* actúa, el voltaje en el ultracapacitor disminuye, este comportamiento no se explorará con profundidad ya que el sistema se diseña para que el ultracapacitor almacene mucha energía o sea que las variaciones de voltaje son

relativamente pequeñas. El sistema de control debe ser diseñado de manera que si el ultracapacitor se descarga por debajo del voltaje mínimo establecido, el conversor *boost* sea desactivado hasta que por medio del conversor *buck* el ultracapacitor recupere energía y aumente su voltaje a un valor dado.

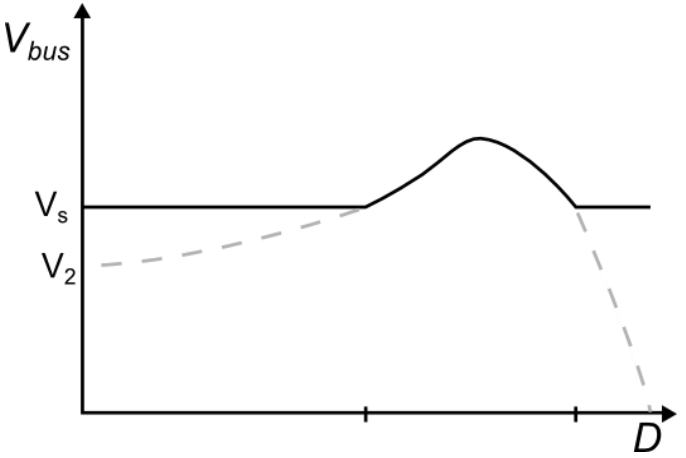


Figura 10. Relación entre el voltaje del bus de DC y el ciclo de trabajo del conversor *boost*

2.3. Control de compensación de voltaje para el conversor de corriente directa

Para el sistema de compensación de voltaje es necesario implementar dos sistemas de control, uno mantiene el voltaje del ultracapacitor dentro de un rango de voltajes especificado en el diseño y otro que para compensar las variaciones que producen las cargas en el *bus de DC*.

El sistema de control de voltaje del ultracapacitor se utiliza para asegurar que el voltaje del ultracapacitor permanezca dentro de un rango previamente establecido y que satisfaga las condiciones de diseño del sistema de compensación. Existen diversos factores que influyen en que la energía que es almacenada y la energía que entregada por el sistema sean diferentes, uno de ellos por ejemplo es que la carga es variable.

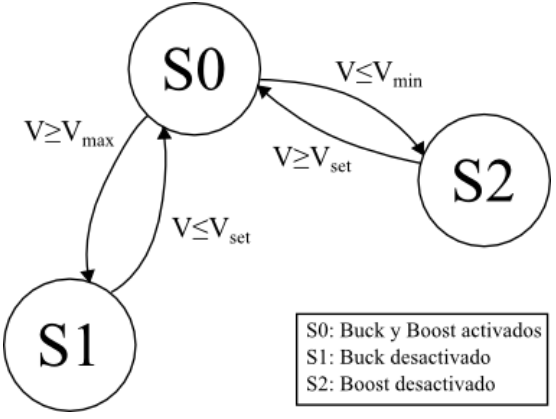


Figura 11. Diagrama de transición de estados el control de voltaje en el ultracapacitor.

El límite inferior del voltaje del ultracapacitor se calcula a partir del estado de carga (S_oC) que será presentada de forma más amplia en la siguiente sección y el voltaje máximo debe seleccionarse menor que el voltaje máximo posible del ultracapacitor.

La estrategia de control que se propone en este proyecto para controlar el voltaje del ultracapacitor es un control *on-off* modificado. El funcionamiento consiste en monitorear el voltaje sobre un rango que se encuentra dentro del rango de los límites de operación, cuando el voltaje se salga de dicho rango se procederá a desactivar uno de los dos conversores (el *buck* o el *boost*) hasta que el voltaje regrese a un voltaje dentro del rango, éste punto puede ser especificado, su valor puede ser el valor medio entre los límites de operación o cualquier otro voltaje que se encuentre dentro del rango. En la Figura 11se presenta el diagrama de transición de estados que describe el sistema de control de voltaje en el ultracapacitor, V es el voltaje medido en el ultracapacitor, V_{min} y V_{max} son respectivamente los limites mínimo y máximo y V_{set} es un voltaje mayor a V_{min} y menor a V_{max}.

Por otro lado, para el control de compensación de voltaje en el *bus de DC* se utilizará un controlador PI. En la Figura 12 se presenta el diagrama completo del sistema de control, además de lo que ya se mencionó se agrega una polarización en el conversor *boost* la finalidad de este bloque es evitar la zona muerta que puede verse en la Figura 10 y dos bloques limitadores, el bloque del conversor *boost* se calcula de manera que no sea posible exceder el voltaje máximo del bus de DC y el limitador del conversor *buck* se calcula a partir del voltaje máximo del ultracapacitor.

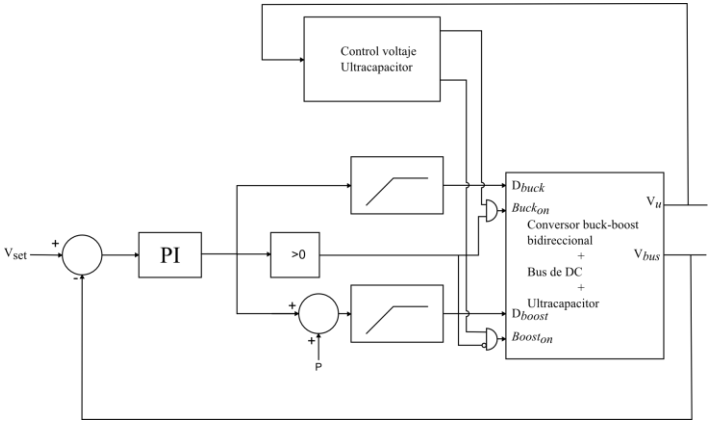


Figura 12. Diagrama en bloques del control de compensación de DC.

A continuación en la

Figura 13 se presenta una simulación del control de compensación de voltaje, la curva roja representa el comportamiento de la carga, cuando alcanza los 140V está entregando energía al *bus de DC* y cuando se encuentra en 0V está consumiendo energía, la figura superior es una simulación con el compensador de corriente directa activado, en contraste, en la parte inferior puede verse el comportamiento del voltaje en el *bus de DC* cuando se desactiva el compensador.

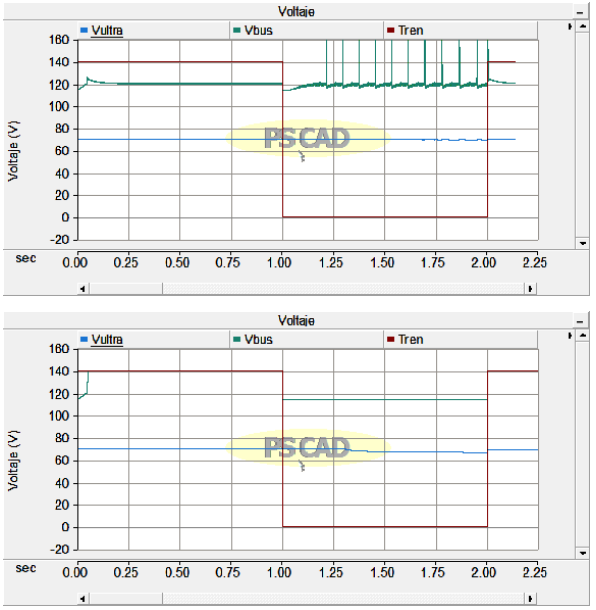


Figura 13. Simulación de control de compensación de voltaje.

3. Diseño del conversor de corriente directa

La forma de determinar los parámetros del conversor de corriente directa está basada en las metodologías planteadas en (Killer, 2012) y (Erickson & Maksimovic, 2001) para el diseño de conversores de corriente directa pero se harán algunas variaciones basadas en algunas consideraciones de diseño y en el trabajo experimental.

A continuación se enlistan los parámetros iniciales de diseño:

- Frecuencia del conversor (f_{conv}.) La frecuencia del conversor está dada por la menor frecuencia máxima de conmutación de todos los semiconductores.
- Estado de carga mínimo ($S_o C_{min}$.) El estado de carga mínimo es el porcentaje de energía mínimo admisible en el ultracapacitor. El estado de carga mínimo se relaciona con la ganancia máxima por medio de la expresión

$$S_o C_{min} = \frac{1}{A_{max}^2} \tag{8}$$

- Voltaje máximo del ultracapacitor ($V_{total\ max}$).
- Voltaje máximo del *bus de DC* ($V_{in\ max}$).
- Voltaje promedio del *bus de DC* (V_{set}).
- Corriente máxima de carga del ultracapacitor ($I_{total\ max}$).
- Voltaje de rizado en modo *boost* ($V_{rip\ boost}$).
- Capacitancia del sistema en el *bus de DC* (C_{bus}).

Ahora con base en estos parámetros debe diseñarse el conversor de corriente directa, primero se calculan los ciclos de trabajo del conversor basado en los voltajes del *bus de DC* y del ultracapacitor.

El valor máximo del ciclo de trabajo del conversor *buck* considerando el voltaje máximo del *bus de DC* es

$$D_{\min buck} = \frac{V_{total\ max}}{V_{in\ max}} \tag{9}$$

y el máximo ciclo de trabajo del conversor *boost*

$$D_{\max boost} = 1 - \frac{V_{total\ max}\sqrt{S_o C_{min}}}{V_{set}}. \tag{10}$$

La inductancia mínima admisible para el conversor *buck* es

$$L_{\min buck} = \frac{D_{\min buck}(V_{in\ max} - V_{total\ max})}{2 f_{conv} I_{total\ max}} \tag{11}$$

y para el conversor boost

$$L_{\min boost} = \frac{V_{set} D_{\max boost}(1 - D_{\max boost})^2}{2 f_{conv} I_{total\ max}}. \tag{12}$$

En este punto es posible tomar dos caminos, por un lado puede agregarse un capacitor en paralelo con el conversor para lograr los requerimientos de voltaje de rizado máximo admitido en el *bus de DC* o puede estimarse el voltaje de rizado máximo que se produciría con base en la capacitancia total conectada al *bus de*

DC. La capacitancia en paralelo puede calcularse mediante la siguiente expresión

$$C_{\min boost} = \frac{I_{total\,max}\, D_{\max boost}}{f_{conv}\, V_{rip\,boost}} \qquad (13)$$

o como en este proyecto en el que se optó por no agregar un condensador en paralelo al conversor para no aumentar la corriente de corto circuito del sistema, el voltaje de rizado puede estimarse con la expresión

$$V_{rip\,boost} = \frac{I_{total\,max}\, D_{\max boost}}{f_{conv}\, C_{bus}}. \qquad (14)$$

Ahora hay que calcular los parámetros de la bobina, la inductancia de la bobina es la mayor entre los valores calculados en las ecuaciones 11 y 12; el otro parámetro de la bobina que es relevante en el diseño del conversor es la resistencia relacionada con las pérdidas en el conductor. Luego de establecer la geometría del núcleo (solenoide, toroide, etc.) se debe calcular el número de vueltas del alambre utilizando las ecuaciones estándar del diseño de inductores, por lo general con esta información es posible calcular la longitud del cable. Como se vio en la ecuación 5, la resistencia serie del inductor, los semiconductores y el ultracapacitor limitan la ganancia máxima del conversor *boost*, el único valor variable en este punto es la resistencia del inductor, los otros dos valores dependen del ultracapacitor y de los semiconductores que se elijan, esta resistencia debe calcularse de manera que sea posible lograr la ganancia máxima, utilizando las expresiones 5, 8 y teniendo en cuenta que R_l es igual a la suma de las resistencias del ultracapacitor, del inductor y de los semiconductores puede hallarse

$$R_b \leq \frac{R}{4}\, S_o C_{min} - R_{IGBT} - R_u. \qquad (15)$$

Donde R_b es la resistencia serie del inductor, R_{IGBT} la resistencia de los semiconductores y R_u es la resistencia serie del ultracapacitor y R es la resistencia de carga a la salida del conversor *boost*; si el lado derecho de la expresión es menor a cero entonces es necesario variar los parámetros de diseño iniciales del conversor o de ser posible agregar semiconductores o ultracapacitores en paralelo para disminuir R_u y R_{IGBT}. Por último, con base en el valor calculado de R_b y la longitud del alambre (que puede calcularse por medio del número de espiras y la geometría del inductor) se puede calcular el radio del alambre

$$r = \sqrt{\frac{\rho l}{\pi R_b}}. \qquad (16)$$

el radio puede elegirse mayor o igual al calculado, ρ es la resistividad del material del alambre y l la longitud del alambre.

4. Interfaz electrónica entre el sistema de potencia y el sistema de control

El proyecto consta de dos componentes principales, el sistema de potencia, encargado de realizar la conmutación de los elementos de potencia para lograr el efecto de la conversión de voltajes en corriente directa y el sistema de control, encargado de general los

pulsos que controlan la conmutación en el sistema de potencia, el sistema de control además mide señales provenientes del sistema de potencia. Para lograr el sistema descrito anteriormente fue necesario implementar una interfaz electrónica entre ambos sistemas, para este proyecto se estudiaron y probaron dos posibilidades que se describen a continuación.

Al final, se optó por el circuito con fuentes independientes que se presenta en la Figura 17 ya que permitía una mayor flexibilidad a la hora de implementar el controlador.

4.1. Circuito con dos fuentes aisladas galvánicamente

La implementación de este circuito utiliza el circuito integrado TLP250 que es un opto-acoplador especializado en el disparo de IGBT y MOSFET de potencia, para este circuito se requieren dos transformadores que permiten lograr un aislamiento de las tierras de cada uno de los opto-acopladores. El circuito se presenta en la Figura 17.

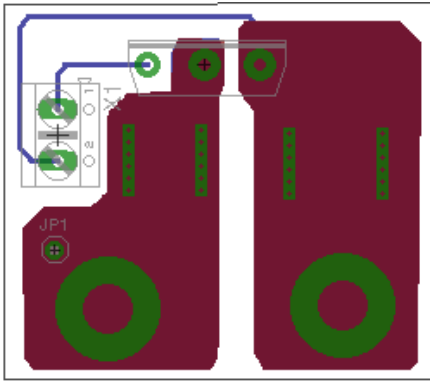
4.2. Circuito con fuente única

El circuito integrado IR2110 ofrece la posibilidad de manejar la compuerta de dos IGBT o MOSFET independientes utilizando únicamente una fuente de DC, esta aparente ventaja se ofrece a cambio de que el modo de operación sea discontinuo, es decir uno de los dos transistores de potencia solo puede trabajar de manera continua durante intervalos cortos de tiempo, luego de esto el transistor es apagado y permanecerá así por un periodo de tiempo similar al que permaneció encendido (este comportamiento se debe a que el circuito debe cargar y descargar un condensador de "bootstrap"). El circuito se presenta en la Figura 18.

5. Componentes de potencia

En esta sección se presentan los elementos de potencia utilizados durante el proyecto.

Para el proyecto fue necesario construir una bobina con características especiales, el objetivo era alcanzar una inductancia de 1mH aproximadamente con un factor de calidad alto, se logró obtener una bobina con una inductancia de .5mH la cual funcionó correctamente, para lograr un factor de calidad alto se utilizó un alambre de 10AWG que redijera la resistencia al mínimo; el núcleo que se utilizó fue un toroide de ferrita. En la Figura 15 puede verse la bobina que se construyó durante el proyecto.



14 Circuito impreso para los semiconductores de potencia.

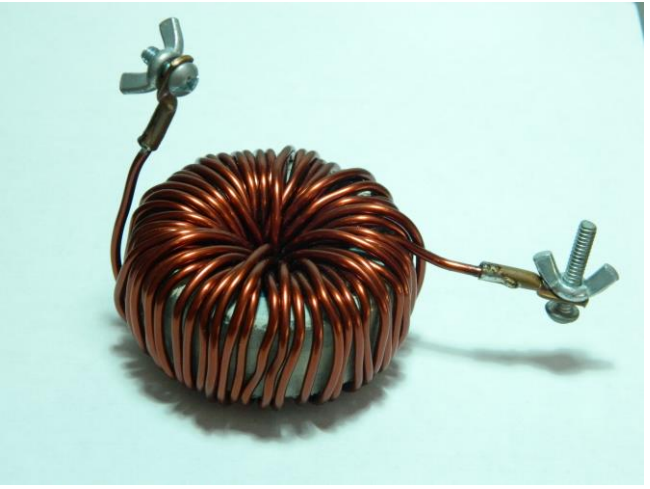


Figura 15. Bobina del conversor de corriente directa.

Los semiconductores de potencia que se utilizaron fueron módulos compuestos por un *IGBT* con capacidad de conducir corrientes de 25A a 100ºC y un diodo de características similares conectado en configuración antiparalelo, referencia del dispositivo es IRG4PC50UD. Se diseñó un circuito impreso donde se conectan los semiconductores y además se acoplaron a disipadores térmicos. Una fotografía de los elementos mencionados puede verse en Figura 16, cada semiconductor además está conectado a un fusible de 30A.

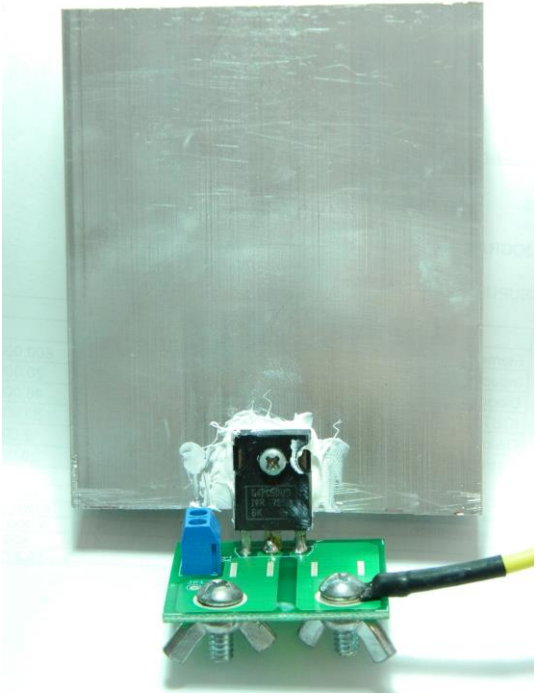


Figura 16. Semiconductor, disipador térmico y PCB.

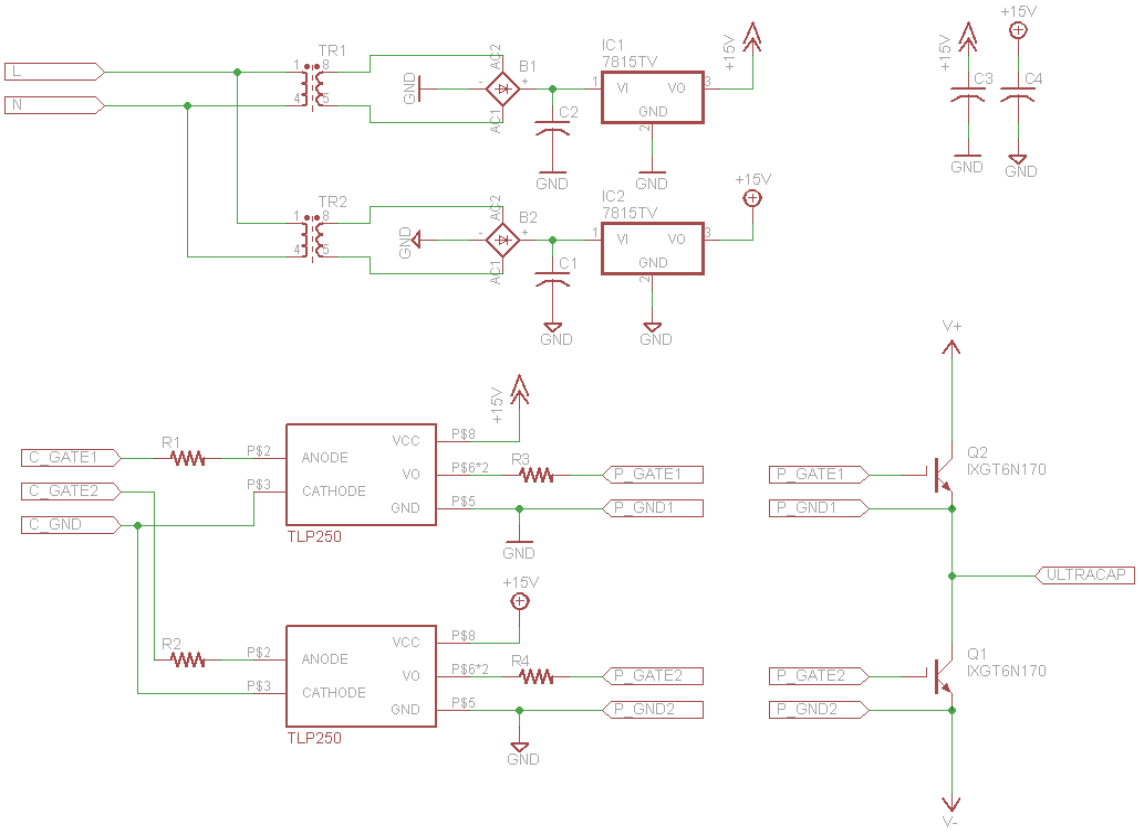


Figura 17. Circuito Interfaz con dos fuentes independientes

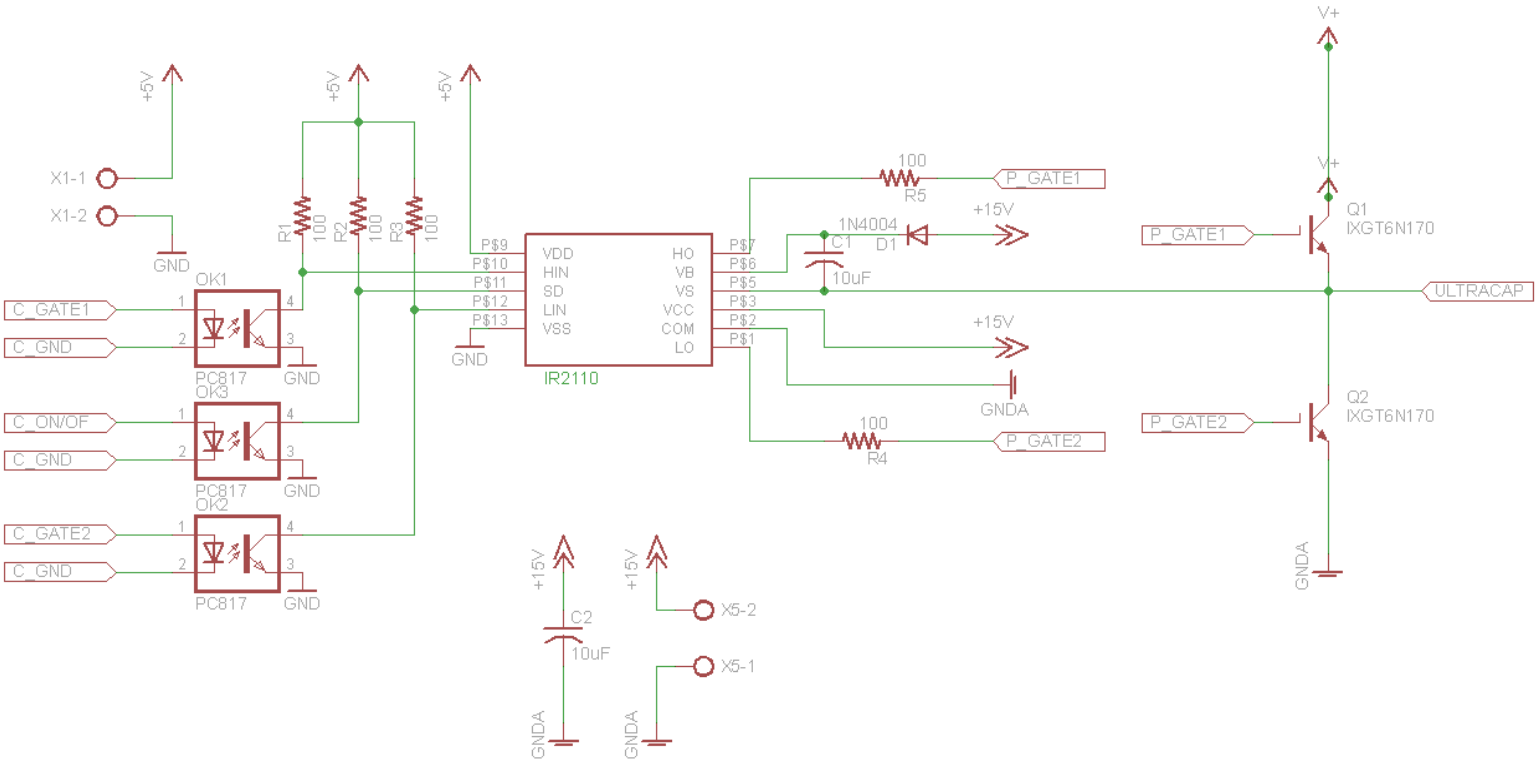


Figura 18. Circuito con fuente única

6. Circuito electrónico de control

En esta sección se hará una descripción del circuito electrónico en el cual se implementará el sistema de control. El circuito se diseñó de manera flexible para permitir futuras mejoras y modificaciones que no se consideraron o no hacen parte de los alcances de este proyecto (PCB y esquemáticos en los anexos).

Características del circuito:

- Dos entradas análogas con interfaz para medir altos voltajes en corriente directa.
- Dos salidas digitales con acople óptico para manejar los *IGBT* estas salidas permiten utilizar *PWM* a frecuencias de hasta 30 kHz.
- Siete pines de uso libre, pueden ser entradas análogas, entradas digitales o salidas digitales.
- Comunicación serial aislada.
- Conexión de cable *BDM* para programar el microcontrolador.
- Microcontrolador *Freescale*™ MCF51JM128de 32 bits.

A continuación se hace una descripción detallada de cada una de las partes del circuito.

6.1. Entradas análogas

El circuito cuenta con 9 entradas análogas, 7 de ellas se encuentran conectadas directamente al microcontrolador, el voltaje máximo que pueden medir es de 5V con respecto a la referencia y el voltaje mínimo es 0V.

Las otras 2 entradas análogas están conectadas a un circuito divisor de voltaje, a un diodo *Zener* que limita el voltaje a 5V y a un *filtro RC* de 500kHz, estableciendo una frecuencia de muestreo mínima de 1kHz para estas dos entradas, en la Figura 19 se presenta el esquema de estas entradas.

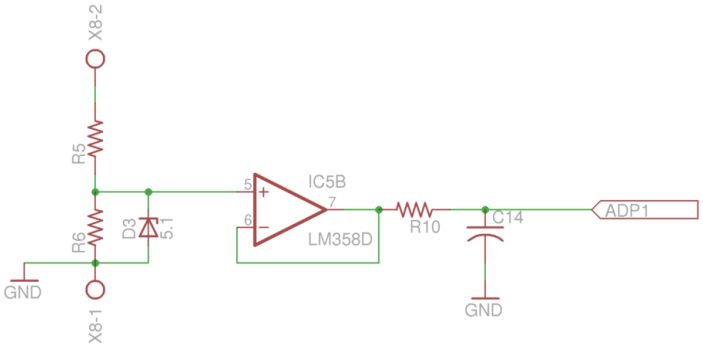


Figura 19. Esquema para las entradas analogas con divisor de voltaje, limitador y filtro.

6.2. Entradas/salidas digitales

Las entradas/salidas son de uso libre y pueden ser utilizadas bajo los parámetros que permite el microcontrolador, estos pines no fueron conectados a semiconductores para permitir que se utilizaran como pines de entrada o salida. Para utilizarlos como salidas para manejar elementos de potencia como relés y contactores es necesario conectar una interfaz de potencia, por ejemplo un *optotriac* o un transistor.

Dos salidas se encuentran conectadas a circuitos integrados con acoples ópticos que sirven de interfaz entre las compuertas de los *IGBT* y el sistema de control. Estas salidas están diseñadas para trabajar como *PWM* de hasta 30 kHz.

6.3. Comunicación serial aislada

En muchas aplicaciones puede ser útil o indispensable contar con comunicación serial, ya sea para acceder a la información que leen los sensores o para implementar algoritmos de control avanzados que requieran mayor poder de cómputo y no puedan ser implementados directamente en el microcontrolador, en la Figura 20 se presenta el esquema de la conexión serial.

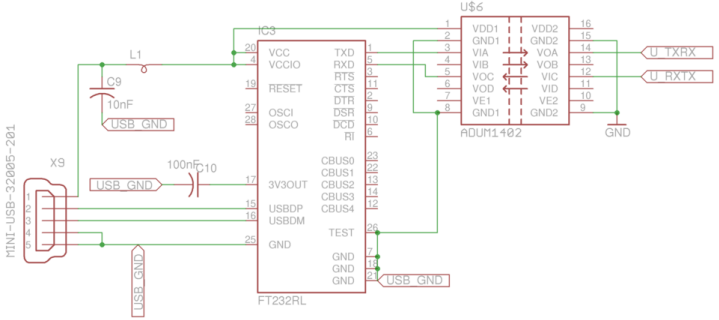


Figura 20. Esquema de la conexión serial aislada.

Físicamente no se utiliza una conexión serial *RS-232*, en cambio se utiliza una conexión *USB Mini* conectada a un circuito conversor de *USB* a Serial. Para la fecha es más común encontrar en un computador conexiones de *USB* que conexiones *DB-25* o *DE-9*, esto hace que sea más práctico utilizar un puerto *USB Mini* y un conversor de *USB* a serial que utilizar los conectores propuestos por *RS-232*.

La conexión entre el conversor *USB* a serial y el microcontrolador se hace mediante un circuito que aísla eléctricamente las tierras de las señales, esto permite que se conecte un computador al puerto serial sin correr el riesgo de que se produzca un corto circuito entre las tierras de los dispositivos. El conversor de *USB* a serial

es alimentado por el cable *USB* desde el computador u otro dispositivo.

6.4. Conexión BDM (Freescale™, 2009)

Es la conexión estándar recomendada por *Freescale™* para programar el microcontrolador, esta conexión no es aislada, el microcontrolador no debe programarse mientras el circuito de potencia se encuentre activo o conectado (esto incluye los sensores y la referencia).

7. Software del dispositivo

Se implementaron dos programas diferentes, uno para el manejo del conversor *DC-DC* por medio de un puerto serial, el otro es el software específico del compensador de corriente directa (Programas en lenguaje C en los anexos).

7.1. Software del conversor DC-DC

En este programa se implementó una comunicación por puerto serial para controlar remotamente el conversor de corriente directa, el programa puede utilizarse manualmente o puede comunicarse con un software de control en un equipo externo. El programa se hizo con tres estados: apagado, boost y buck.

A continuación se describen los comandos predefinidos de comunicación serial, en los comandos se utiliza el símbolo # para representar un carácter variable, por ejemplo ## puede ser cualquier número entre 00 y 99.

- "buck ###\r": El programa pasa al estado *buck* y enciende una señal cuadrada con un ciclo de trabajo definido por ### en el pin correspondiente del microcontrolador, el ciclo de trabajo es una décima parte del valor especificado en ###, así si la instrucción es "buck 235\r", el ciclo de trabajo será del 23.5%. Si el programa se encuentra en el estado *buck*, permanecerá en este pero cambiará el ciclo de trabajo al valor especificado. Cuando el dispositivo se encuentra en este estado la señal que va a la compuerta del *IGBT* del conversor *boost* se pone en cero.
- "boost ###\r": El programa pasa al estado boost y enciende una señal cuadrada con un ciclo de trabajo definido por ### en el pin correspondiente del microcontrolador, el ciclo de trabajo es una décima parte del valor especificado en ###. Si el programa se encuentra en el estado *boost*, permanecerá en éste pero cambiará el ciclo de trabajo al valor especificado. Cuando el dispositivo se encuentra en este estado la señal que va a la compuerta del *IGBT* del conversor *buck* se pone en cero.
- "stop\r": Pasa el programa al estado apagado, en este estado las dos señales que van a las compuertas de los *IGBT* se encuentran apagados.

- "sensor #\r": Devuelve un valor de 12 *bits* (de 0 a 4095) leído en el canal del *ADC* especificado por #. El dispositivo utiliza el comando "sensor <numero>\n" para responder a ésta instrucción.
- "state\r": Lee el estado y el ciclo de trabajo y responde con el valor establecido, por ejemplo si el conversor *boost* se encuentra encendido con un ciclo de trabajo del 53% la respuesta será "boost 53.0%\n", por otro lado si se encuentran ambos conversores apagados la respuesta será "off\n".

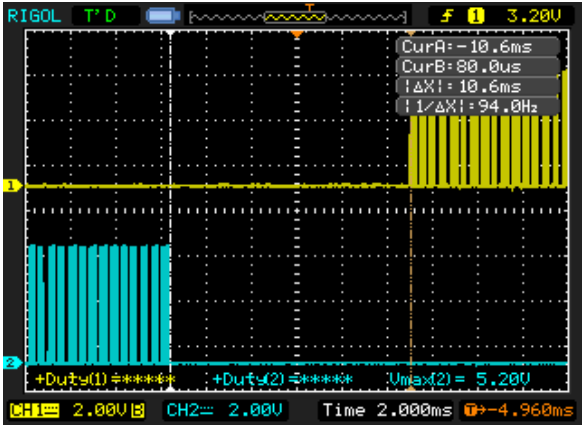


Figura 21. Transición entre estados *buck* y *boost*

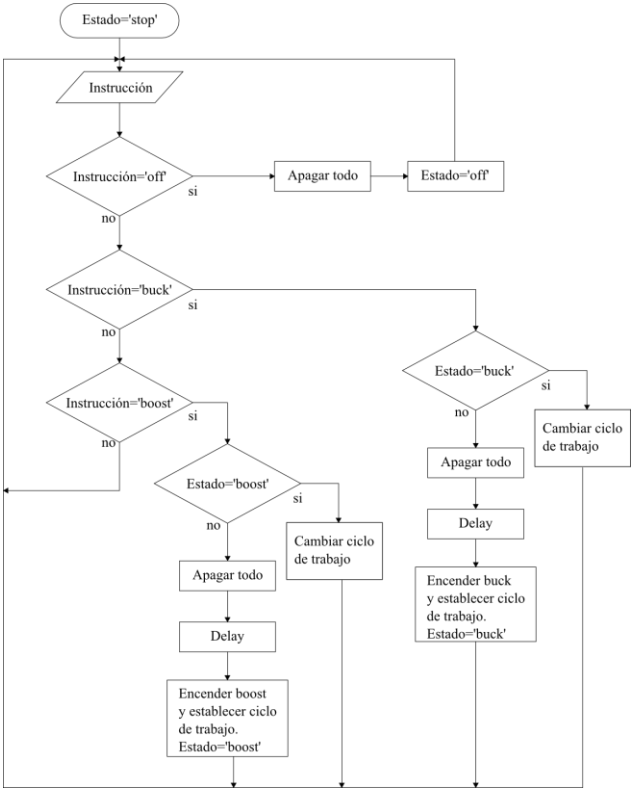


Figura 22. Diagrama de transición entre estados.

Uno de los aspectos más relevantes de ésta implementación es que se puede establecer un tiempo en la transición entre dos estados, de esta manera es posible asegurar que los dos *IGBT* no van a estar encendidos simultáneamente. En la Figura 21 se presenta una medición que se hizo del tiempo de transición entre un estado y otro y en la Figura 22 un diagrama de flujo que especifica cómo se implementó en el programa.

7.2. Software de control de compensación de voltaje

En la Figura 23 se presenta el diagrama de flujo del algoritmo utilizado para el proceso de compensación de voltaje, es una implementación del control propuesto en la Figura 12.

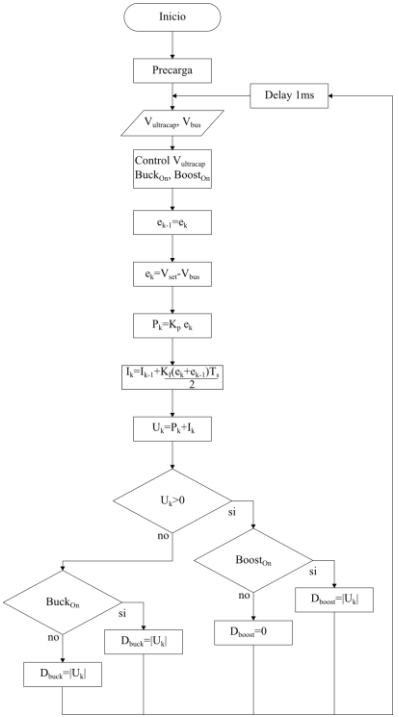


Figura 23. Diagrama de flujo del algoritmo de compensación de voltaje

8. Consideraciones de seguridad, protecciones del sistema e instrumentación

En esta sección se presentan aspectos técnicos de la implementación del sistema de frenado regenerativo, algunas de las consideraciones que se presentarán a continuación están fundamentadas en el trabajo experimental que se realizó, se dará una explicación detallada que explique cada parte del diseño que se propone.

En la Figura 24 se presenta el circuito con los elementos de protección y los instrumentos de medida, S_1, S_2 y S_3 son interruptores, F_1, F_2 y F_3 fusibles, hay dos circuitos de descarga sobre los cuales no se hará énfasis y 5 sensores, 3 de voltaje y 2 de corriente.

Las siguientes situaciones producen fallas en el sistema:

- Los dos *IGBT* conducen simultáneamente.
- El *IGBT* inferior o sea el del conversor *boost* permanece encendido mientras el ultracapacitor está cargado.
- El *IGBT* superior permanece encendido mientras el sistema está conectado al bus de DC.
- El voltaje del ultracapacitor supera el voltaje máximo.
- La corriente del ultracapacitor supera la corriente máxima.
- El *IGBT* inferior opera a un ciclo de trabajo muy alto, esta situación se explica brevemente al final de esta sección.

Comenzando por los sensores, las variables mínimas requeridas para implementar el sistema de control son el sensor V_{bus} que mide el voltaje del *bus de DC* y V_c que mide el voltaje del ultracapacitor. Los sensores de corriente I_1 e I_c no son requeridos para el sistema de control, sin embargo, el sensor I_1 es necesario si se implementa un control de corriente para el conversor *buck*, también, considerando que el objetivo del sistema es ahorrar energía, es buena idea tener los sensores de corriente ya que sin ellos no es posible calcular la energía que entra, la energía que entrega y la eficiencia del conversor.

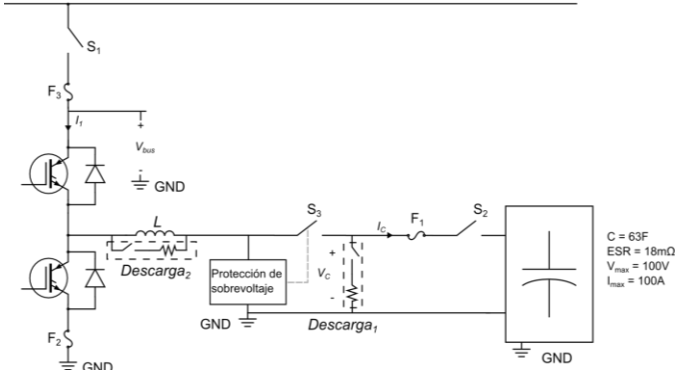


Figura 24. Diagrama del circuito con protecciones e instrumentos de medida.

La ubicación de los sensores V_{bus} e I_1 se eligió de manera que queden desconectados del *bus de DC* si F_3 o S_1 se abren, ya sea por falla o por mantenimiento, igualmente los sensores V_c e I_c se encuentran ubicados de manera que puedan ser desconectados por F_1 o S_2 del ultracapacitor que bajo ciertas circunstancias podría permanecer cargado. Además de eso, en caso de que los sensores V_{bus} o V_c fallaran y provocaran un corto circuito, los Fusibles F_3 y F_1 protegerían al sistema.

Como se vio en la primera sección y de manera más específica, teniendo en cuenta el comportamiento descrito por la expresión (4), la resistencia en serie con la bobina afecta ampliamente la ganancia máxima del conversor *boost*, por este motivo es recomendable considerar utilizar sensores de corriente de efecto Hall, estos sensores son menos invasivos y no aumentan la resistencia del sistema.

Hay un tercer sensor de voltaje en el sistema, el sensor del sistema de protección de sobrevoltaje, éste puede ser V_c, el mismo que se utilizó para medir el voltaje del ultracapacitor, la utilización de un sensor independiente depende principalmente de como se quiera implementar la protección de sobrevoltaje y si se quiere o no independizar del sistema principal de control.

El ultracapacitor está diseñado para un voltaje máximo, debe protegerse contra voltajes superiores a los que puede soportar. Parte de esta protección puede hacerse por software ya que el voltaje V_c es conocido, sin embargo, es una buena práctica agregar un circuito independiente de protección como un relé de sobrevoltaje, diodos *Zener*, etc. El circuito de protección para sobrevoltaje controla el interruptor S_3.

Durante los experimentos que se realizaron con el ultracapacitor se vio la necesidad de un circuito para descargar completamente el ultracapacitor, para evitar problemas luego de finalizar su operación, o cuando se presentan fallas en los *IGBT*. Una solución práctica a este problema es agregar en paralelo con el ultracapacitor un circuito que conecte el mismo a una resistencia que lo descargue. Una de las situaciones que debe considerarse es el mantenimiento de los equipos, por seguridad se requiere que todos los elementos que tengan la capacidad de almacenar energía puedan ser descargados, los circuitos de descarga integrados al

sistema permiten que se haga de manera más automática, sin poner en contacto a un operario con el elemento en cuestión mientras está cargado. En el sistema hay dos elementos que pueden almacenar energía, la bobina y el ultracapacitor, ambos cuentan con un circuito de descarga.

El interruptor S_1 se encarga de desconectar completamente el sistema, el interruptor S_2 desconecta el ultracapacitor, este interruptor es necesario cuando F_1 se abre, en este estado es imposible descargar el ultracapacitor y para remplazarlo la opción es desconectarlo por medio de S_2, éste además puede utilizarse en labores de mantenimiento menores que no requieran la descarga del ultracapacitor.

El fusible F_1 está diseñado para la corriente máxima pico del ultracapacitor, F_2 y F_3 protegen los semiconductores y deben calcularse para la mínima de las corrientes pico que puede soportar cada uno de los semiconductores, es decir que se diseñan para proteger al semiconductor que menos soporte.

Hay un caso especial que debe considerarse y que fue mencionado al principio de la sección. Cuando el *IGBT* del conversor *boost* opera con un ciclo de trabajo muy alto la eficiencia del conversor llega casi a cero, en este estado se disipa una gran cantidad de energía en dicho *IGBT*, esta situación puede evitarse por medio del software que controla el ciclo de trabajo de este *IGBT*.

CONCLUSIONES

Se implementó exitosamente un conversor *buck-boost* bidireccional de corriente directa que permite la carga y descarga de un ultracapacitor conectado a un *bus de DC*.

Se comprobó que la estrategia de diseño propuesta en (Killer, 2012) para conversores de corriente directa puede ser utilizada en la práctica, además, se complementó con detalles del diseño de la bobina y la selección de los semiconductores, en especial se resaltó la importancia de minimizar la resistencia serie producida por la bobina y los semiconductores.

Se simuló exitosamente un control PI para la compensación de voltaje en el *bus de DC* el control se implementó pero no produjo resultados satisfactorios, se presume que es necesario utilizar una instrumentación más especializada para lograr los resultados vistos en la simulación.

Con base en la experiencia adquirida durante el proyecto, se diseñaron dos guías de laboratorio para aprender sobre el frenado regenerativo y conversores de corriente directa.

TRABAJO POSTERIOR

En este proyecto se realizó una primera aproximación a la implementación y a los aspectos técnicos de los conversores *DC-DC*, durante el proyecto se encontraron diferentes puntos de los que pueden surgir trabajos posteriores y derivados, a continuación se exponen estos puntos.

Durante el proyecto se presentaron diversos problemas con el manejo de la temperatura del conversor *DC-DC*, esto plantea entonces la necesidad de profundizar más en el manejo de la temperatura para los semiconductores de potencia.

Como pudo verse en este documento, la bobina del conversor de corriente directa juega un papel muy importante y el resultado final depende considerablemente de este elemento. El diseño y la construcción de la bobina constituyen una rama aparentemente importante para los conversores *DC-DC* en la que puede profundizarse de manera que se pueda lograr un elemento que funcione eficientemente en la banda de frecuencia especificada y con un factor de calidad alto.

El sistema de control plantea un área muy grande de investigación donde se puede profundizar mucho más, quedan varios problemas abiertos: transición entre conversores *buck* y *boost*, la búsqueda de un sistema de control más eficiente, un sistema de control que no corrija el comportamiento inestable que se presenta en el conversor *boost* cuando el ciclo de trabajo sobrepasa el punto de máxima ganancia de voltaje (este comportamiento puede verse en la Figura 4) y una estrategia de control para mantener la carga del ultracapacitor dentro de un rango especificado (en este caso se utiliza un control *on-off*).

AGRADECIMIENTO

Al director de éste proyecto Andrés E. Diez y a todos los que participaron y aportaron al proyecto durante las diferentes revisiones y presentaciones que se hicieron.

Para este proyecto se utilizó software libre o versiones especiales para estudiantes.

PSCAD™ Standard Free, este software se utilizó para simular el conversor de corriente directa diseñado, para probar el sistema de control y para generar los datos utilizados en las Figuras 2 y 4.

Python™ y los paquetes NumPy™, Matplotlib™ fueron utilizados para analizar datos realizar pruebas varias y en particular para generar las Figuras 2 y 4.

INKSCAPE™ se utilizó para generar los gráficos contenidos en las Figuras 1, 2 y 5.

CadSoft EAGLE PCB Design Software™, versión gratuita, fue utilizado para diseñar el circuito impreso y los planos esquemáticos que se presentan en el texto.

CodeWarrior Development Tools, versión gratuita, fue utilizado para programar el microcontrolador que se utilizó.

REFERENCIAS

Erickson, R. W., & Maksimovic, D. (2001). *Fundamentals of Power Electronics.* New York: Kluwer Academic.

Freescale™. (06 de 2009). *MCF51JM128 ColdFire® Integrated Microcontroller Reference Manual.* Recuperado el 20 de Octubre de 2013, de Sitio web de Freescale™: http://www.freescale.com/files/32bit/doc/ref_manual/MCF51JM128RM.pdf

Iannuzzi, D., Pighetti, P., & Tricoli , P. (2010). A study on Stationary Supercapacitor sets for Voltage Droops Compensation of Streetcar Feeder Lines. *Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS).*

Killer, A. (2012). Ultracapacitor Assisted Regenerative Braking. En A. Killer, *Ultracapacitor Assisted Regenerative Braking* (págs. 9-10). Medellín, Antioquia, Colombia: Universidad Pontificia Bolivariana.

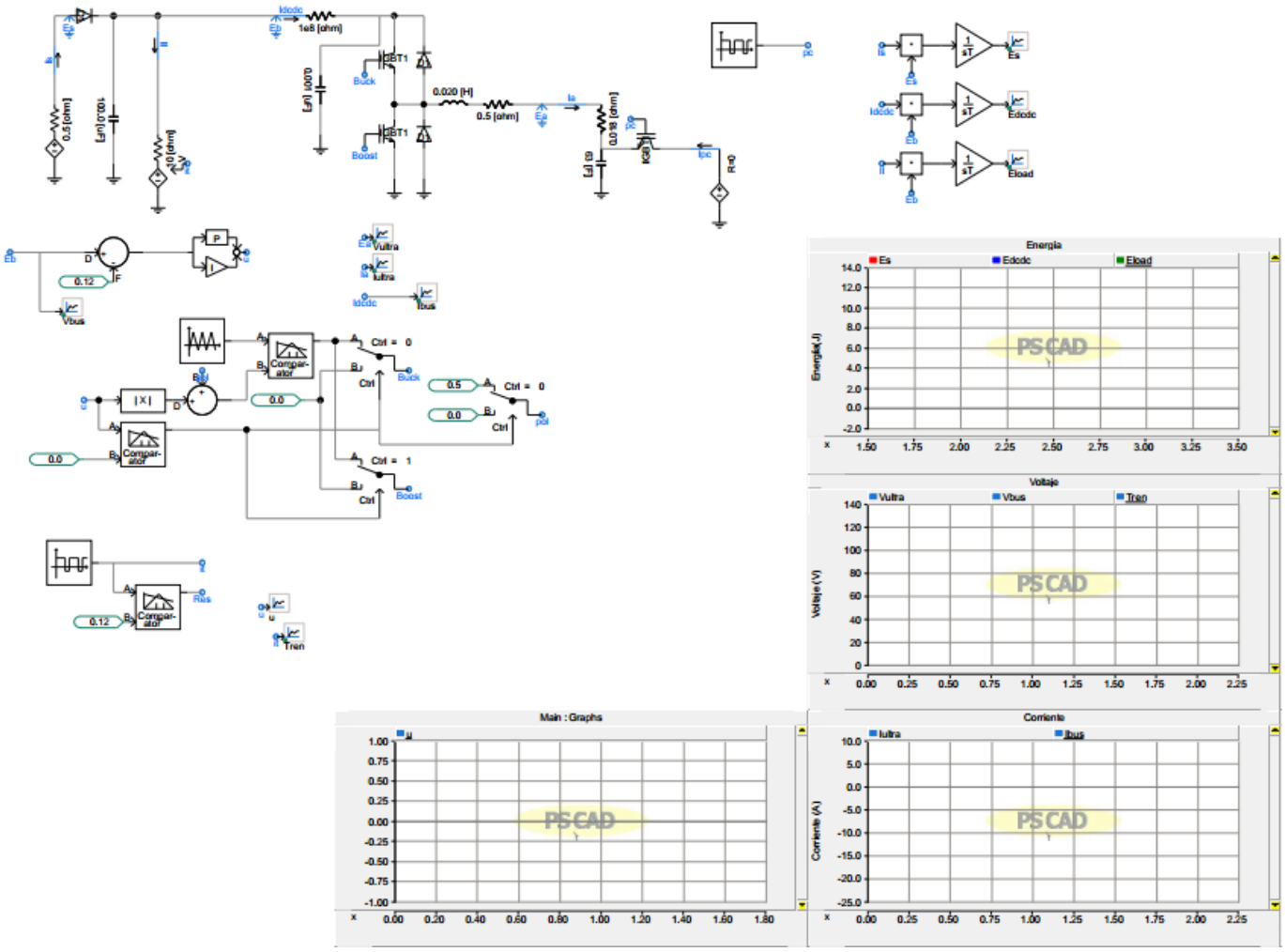
AUTORES

Andrés E. DÍEZ RESTREPO, Ingeniero Electricista de la Universidad Pontificia Bolivariana –UPB-
Máster en ingeniería de la UPB en 2005.
Doctor en Ingeniería de la UPB en 2010 con tesis Estrategia para la electrificación del transporte en Colombia, en convenio con la Universidad de Ciencias Aplicadas de Kempten, Alemania, y dirigida por el Profesor-Doctor Helmuth Biechl.
Docente ocasional Universidad Nacional de Colombia 2006-2009
Docente-Investigador de pregrado y posgrado de la UPB desde 2002

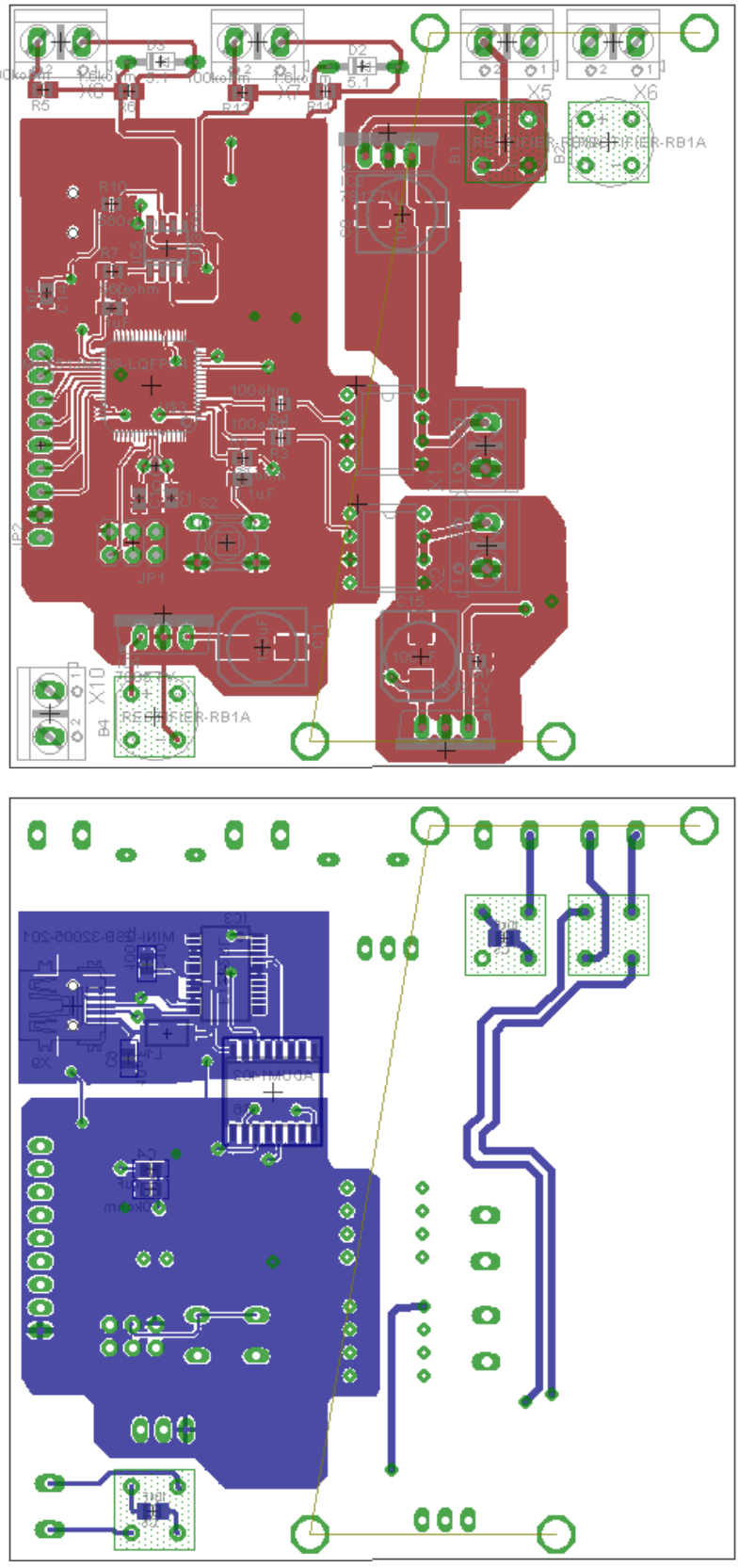
Daniel VILLEGAS POSADA, nacido en Medellín, Colombia en 1990. Estudiante de ingeniería electrónica en la Universidad Pontificia Bolivariana.

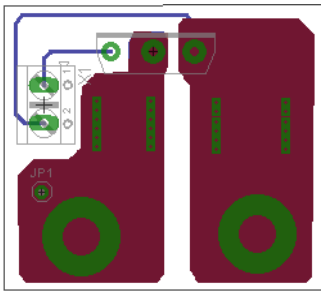
Anexos

Simulación en PSCAD

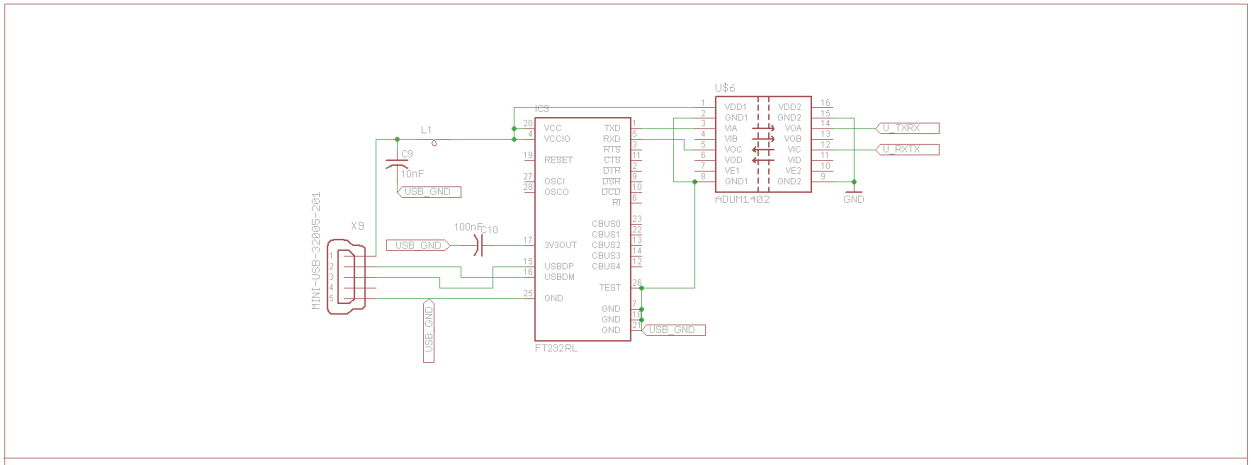


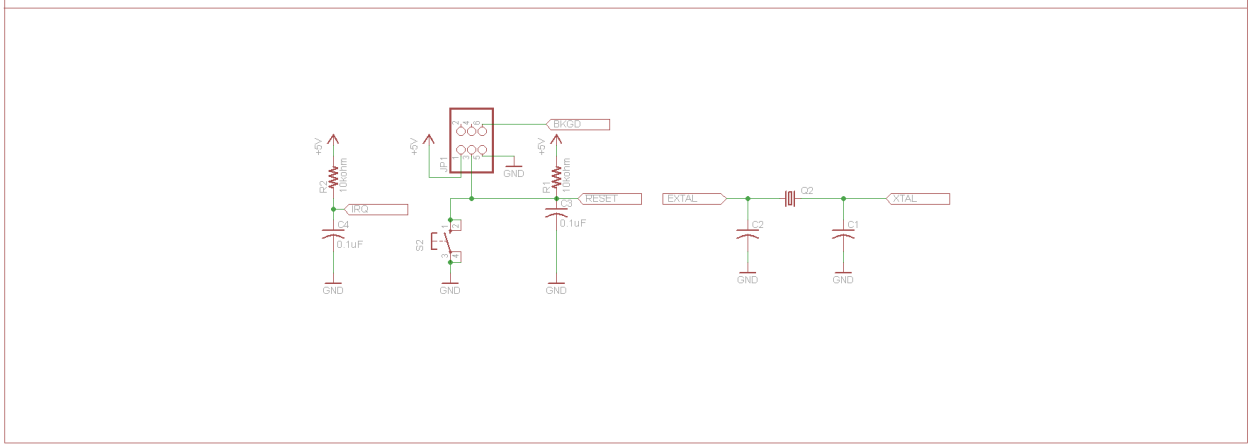
Capa superior del circuito impreso

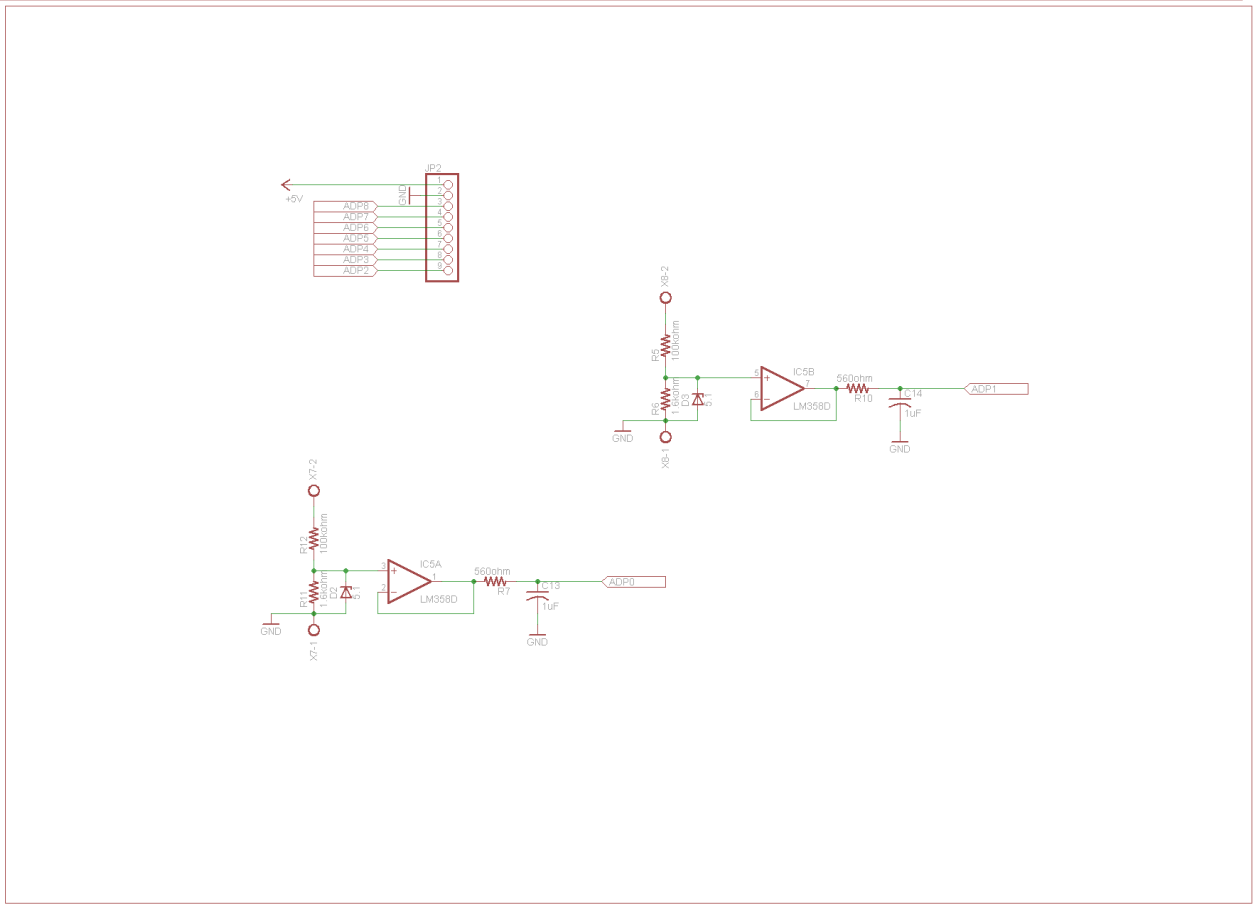


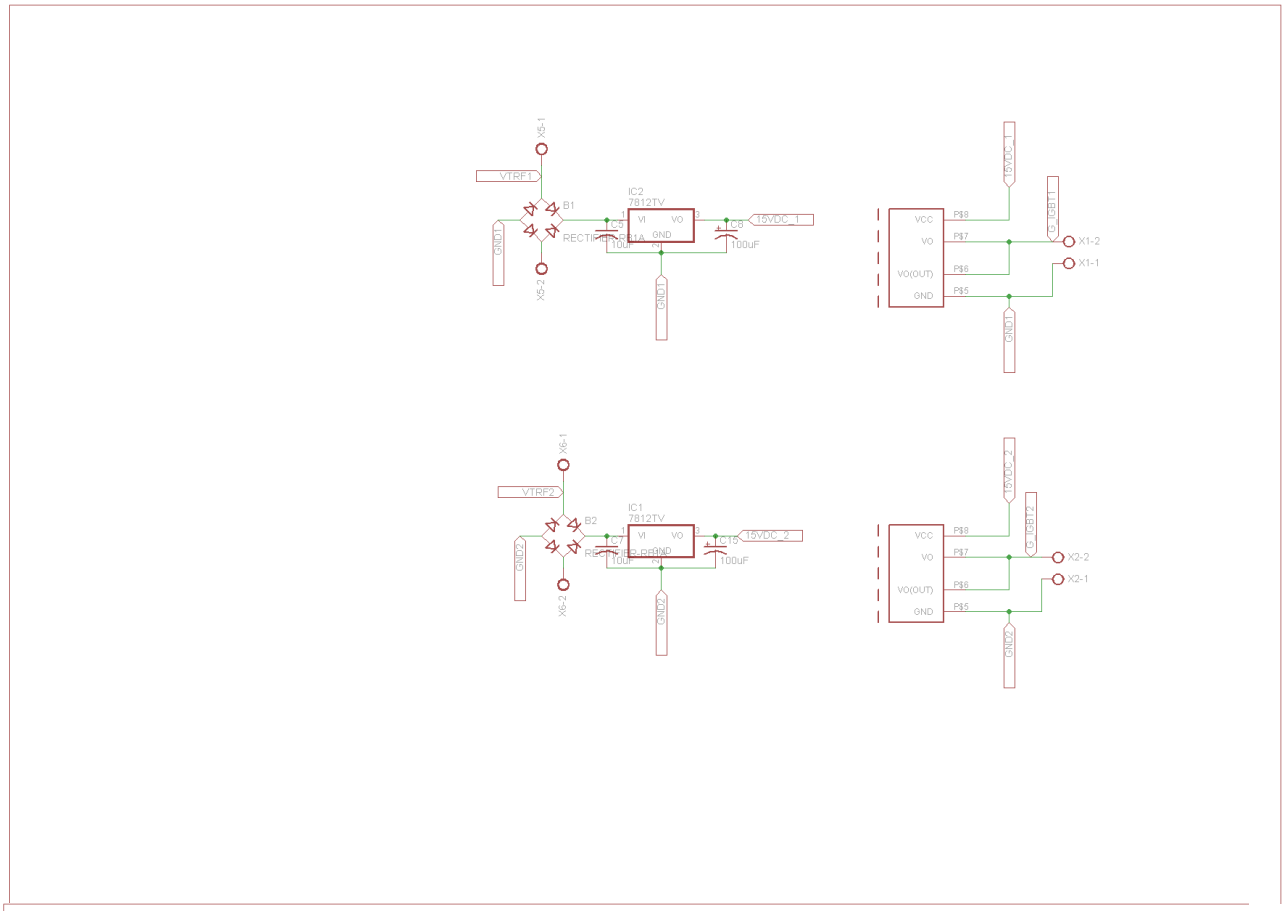

JP1

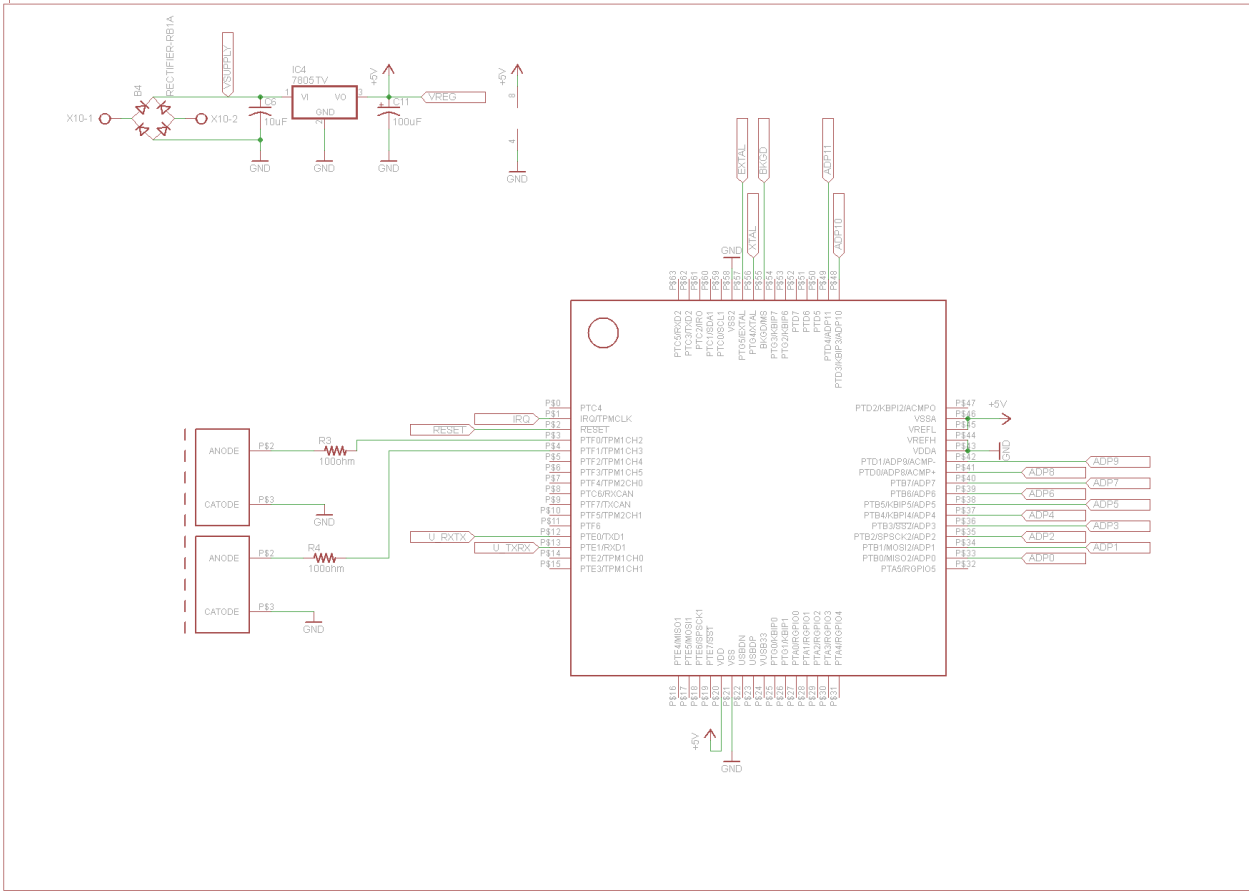
Planos del circuito








X5-1
VTRF1
B1
IC2
7812TV
VI
VO
GND
C5
100uF
C6
100uF
15VDC_1
GND1
X5-2
VCC
VO
VO(OUT)
GND
P$8
P$7
P$6
P$5
X1-2
X1-1
15_ICBT1
X6-1
VTRF2
B2
IC1
7812TV
C7
100uF
C10
100uF
15VDC_2
GND2
X6-2
15_ICBT2
X2-2
X2-1


X10-1
B4
RECTIFIER-RB1A
X10-2
VSUPPLY
IC4
7805TV
C8
10uF
C11
100uF
VREG
+5V
GND
EXTAL
BKGD
ADP11
ADP10
XTAL
RESET
IRQ
U_RXTX
U_TXRX
R3
R4
100ohm
ANODE
CATODE
ADP9
ADP8
ADP7
ADP6
ADP5
ADP4
ADP3
ADP2
ADP1
ADP0
PTC4
IRQ/TPMCLK
RESET
PTF0/TPM1CH2
PTF1/TPM1CH3
PTF2/TPM1CH4
PTF3/TPM1CH5
PTF4/TPM2CH0
PTC6/RXCAN
PTF7/TXCAN
PTF5/TPM2CH1
PTF6
PTE0/TXD1
PTE1/RXD1
PTE2/TPM1CH0
PTE3/TPM1CH1
PTD2/KBPI2/ACMPO
VSSA
VREFL
VREFH
VDDA
PTD1/ADP9/ACMP-
PTD0/ADP8/ACMP+
PTB7/ADP7
PTB6/ADP6
PTB5/KBIP5/ADP5
PTB4/KBPI4/ADP4
PTB3/SS2/ADP3
PTB2/SPSCK2/ADP2
PTB1/MOSI2/ADP1
PTB0/MISO2/ADP0
PTA5/IRQPIO5

Software del conversor DC-DC

```
#include <hidef.h>
#include "derivative.h"
#define BUSCLK 4274000
#define SERIAL_TIMEOUT 500
#define TRANSFER_DELAY 10
//MCG definitions


unsigned char init_clock(void);
void MCG_FEI(unsigned char DRS, unsigned char DMX32, unsigned char BDIV);
void MCG_FEE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char
BDIV, unsigned char RANGE, unsigned char DIV32);
void MCG_FBE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char
BDIV, unsigned char RANGE, unsigned char DIV32);
void MCG_PBE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char
DIV32);
void MCG_PEE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char
DIV32);


void init_timer(void);
void change_duty_buck(unsigned short int d);
void change_duty_boost(unsigned short int d);
void change_freq(unsigned short int f);
void init_rtc(void);
void rtcISR(void);
void serialISR(void);
void init_serial(int baudrate);
int int_pow(int a, int b);
void init_adc(void);
void send_char(char chr);
void send_str(const char *str);
void clear_buffer(void);
void write_serial_buffer(char chr);
int str_cmp(const char *str1, const char *str2, char ignore, char limiter);
int str2int_x3(char *str);
void stop_dcdc(void);
void num2str(unsigned int num, char *str, int len);
unsigned int read_adc(char channel);
void delay_ms(unsigned int delay);
char buck_boost_state = 0;
char serial_buffer[30];
int serial_buffer_idx = 0;
char echo_enable = 0;
unsigned long int time_ms = 0;
unsigned int timeout_ms = 0;
unsigned int transfer_ms = 0;
char duty[5];



/*
 * 0 -> off
 * 1 -> buck
 * 2 -> boost
 */
float x = 0, y = 0, k = 0.1, T = 1e-3, A = 0;
void main(void) {
  char num_buffer[5];
  num_buffer[4] = 0;
  EnableInterrupts;
  /* include your code here */
```

```
//  init_clock();
  unsigned int adc_reading;
  unsigned long int i = 0;
  int j = 0;
  A = k * 1000;
  SOPT1_COPT = 0;
  /*** End of Processor Expert internal initialization.                    ***/
  init_timer();
  init_rtc();
  init_adc();
  init_serial(9600);
  change_freq(25000);
  change_duty_buck(0);
  for(;;) {
    __RESET_WATCHDOG();
    if(serial_buffer_idx){
      if(str_cmp("buck ###\r", serial_buffer, '#', '\r')){
            change_duty_buck(str2int_x3(serial_buffer + 5));
            duty[0] = serial_buffer[5];
            duty[1] = serial_buffer[6];
            duty[2] = '.';
            duty[3] = serial_buffer[7];
            duty[4] = '%';
            clear_buffer();
      }
      else if(str_cmp("boost ###\r", serial_buffer, '#', '\r')){
            change_duty_boost(str2int_x3(serial_buffer + 6));
            duty[0] = serial_buffer[6];
            duty[1] = serial_buffer[7];
            duty[2] = '.';
            duty[3] = serial_buffer[8];
            duty[4] = '%';
            clear_buffer();
      }
      else if(str_cmp("stop\r", serial_buffer, '#', '\r')){
            stop_dcdc();
            clear_buffer();
      }
      else if(str_cmp("state\r", serial_buffer, '#', '\r')){
            send_str("state ");
          clear_buffer();
            switch(buck_boost_state){
            case 0:
                  send_str("off\n");
                  break;
            case 1:
                  send_str("buck ");
                  send_str(duty);
                  send_char('\n');
                  break;
            case 2:
                  send_str("boost ");
                  send_str(duty);
                  send_char('\n');
                  break;
            }
      }
      else if(str_cmp("sensor #\r", serial_buffer, '#', '\r')){
            send_str("sensor ");
            adc_reading = read_adc(*(serial_buffer + 7) - '0');
          num2str(adc_reading, num_buffer, 4);
            send_str(num_buffer);
            send_char('\n');
          clear_buffer();
```

```
      }

    }
      } /* loop forever */
  /* please make sure that you never leave main */
}

int str2int_x3(char *str){
      unsigned int val = 0;
      val += (str[0] - '0') * 100;
      val += (str[1] - '0') * 10;
      val += (str[2] - '0');
      return val;
}

int int_pow(int a, int b){
      int r = 1, idx = 0;
      for(idx = 0; idx < b; idx++)
            r *= a;
      return r;
}
void num2str(unsigned int num, char *str, int len){
      unsigned int tmp;
      int idx;
      tmp = num;
      for(idx = len; idx > 0; idx--){
            str[len - idx] = (char) (tmp / int_pow(10, idx - 1));
            tmp -= str[len - idx] * int_pow(10, idx - 1);
            str[len - idx] += '0';
      }
}
void init_rtc(void){
      RTCSC_RTCLKS = 0;
      RTCSC_RTIE = 1;
      RTCSC_RTCPS = 8;
      RTCMOD = 0;
}

void init_adc(void){
      ADCCFG_ADICLK = 1;
      ADCCFG_ADLPC = 0;
      ADCCFG_ADIV = 3;
      ADCCFG_ADLSMP = 1;
      ADCCFG_MODE = 1;
      ADCSC1_ADCH = 0xF;
      ADCSC2_ADTRG = 0;
      APCTL1_ADPC0 = 1;
      APCTL1_ADPC1 = 1;
}

unsigned int read_adc(char channel){
      ADCSC1_ADCH = channel;
      while(!ADCSC1_COCO){};
      return ADCR;
}

interrupt VectorNumber_Vrtc void rtcISR(void){
      RTCSC_RTIF= 1;
      if(time_ms < 100000000L)
            time_ms++;
      else
            time_ms=0;
      if(time_ms - timeout_ms >= SERIAL_TIMEOUT){
            clear_buffer();
```

```
	}
}

void init_serial(int baudrate){
	SCI1C2_TE = 1;
	SCI1C2_RE = 1;
	SCI1C2_RIE = 1;
	SCI1BD = ((BUSCLK*10)/(baudrate*16) + 5)/10;
}

void send_char(char chr){
	while(SCI1S1_TDRE == 0){};
	SCI1D = chr;
	while(SCI1S1_TC==0){};
}

void send_str(const char *str){
	char i = 0;
	while(str[i] != '\0')
		send_char(str[i++]);
}

int str_cmp(const char *str1,const char *str2, char ignore, char limiter){
	int i = 0, ret = 0;
	while((str1[i] != limiter && str2[i] != limiter && str1[i] == str2[i]) || str1[i] ==
ignore || str2[i] == ignore)
		i++;
	ret = str1[i] == str2[i] & str1[i] == limiter & str2[i] == limiter;
	return ret;
}

void write_serial_buffer(char chr){
	serial_buffer[serial_buffer_idx] = chr;
	if(serial_buffer_idx < 30)
		serial_buffer_idx++;
	else
		clear_buffer();
}

void clear_buffer(void){
	int i = 0;
	for(i = 0; i < 30; i++)
		serial_buffer[i] = '\0';
	serial_buffer_idx = 0;
}

interrupt VectorNumber_Vsci1rx void serialISR(void){
	char tmp;
	(void)SCI1S1;
	tmp = SCI1D;
	write_serial_buffer(tmp);
	timeout_ms = time_ms;
	if(echo_enable){
		send_char(tmp);
	}
}

void init_timer(void){
	TPM1C0V = 0;
	TPM1C1V = 0;
	buck_boost_state = 0;
	TPM1SC = 0x88;
	TPM1MOD = 0x100;
	TPM1C1SC = 0b00101000;
```

```
	TPM1C0SC = 0b00101000;
}
void delay_ms(unsigned int delay){
	unsigned int time_delay_0 = time_ms;
	while(delay > time_ms - time_delay_0)
	    __RESET_WATCHDOG();
}
void change_duty_buck(unsigned short int d){
	unsigned short int v = 0;
	int cnt = 0;
	if(buck_boost_state == 2 || buck_boost_state == 0){
		TPM1C0V = 0;
		delay_ms(TRANSFER_DELAY);
		buck_boost_state = 1;
	}
	v = (TPM1MOD * d)/1000;
	TPM1C1V = v;
}

void stop_dcdc(void){
	TPM1C0V = 0;
	TPM1C1V = 0;
	buck_boost_state = 0;
}

void change_duty_boost(unsigned short int d){
	unsigned short int v = 0;
	int cnt = 0;
	if(buck_boost_state == 1 || buck_boost_state == 0){
		TPM1C1V = 0;
		delay_ms(TRANSFER_DELAY);
		buck_boost_state = 2;
	}
	v = (TPM1MOD * d)/1000;
	TPM1C0V = v;
}

void change_freq(unsigned short int f){
	/*
	 * FREQ (Hz)
	 */
	unsigned short int Tmod;
	(void)TPM1MOD;
	Tmod = (BUSCLK/(f))-1;
	TPM1MOD = Tmod;
	while(TPM1MOD != Tmod){
	    __RESET_WATCHDOG();
	}
}

unsigned char init_clock(void){
	SOPT2_CLKOUT_EN = 1;
	MCG_FEI(2, 1, 0);
	MCG_FBE(3, 2, 1, 0, 1, 1);
	MCG_PBE(3, 0, 1, 1);
	MCG_PEE(3, 0, 1, 1);

	return 1;
}

void MCG_FEI(unsigned char DRS, unsigned char DMX32, unsigned char BDIV){
	MCGC1 = 0x04;
	MCGC3_PLLS = 0;
	MCGC4 = (DMX32 << 5)|(DRS);
```

```
	MCGC2_BDIV = BDIV;
}

void MCG_FEE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char
BDIV, unsigned char RANGE, unsigned char DIV32){
	MCGC1 = ((RDIV & 0x7) << 3);
	MCGC2_BDIV = BDIV;
	MCGC2_RANGE = RANGE;
	MCGC3_PLLS = 0;
	MCGC3_DIV32 = DIV32;
	MCGC4 = (DMX32 << 5)|(DRS);

}

void MCG_FBE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char
BDIV, unsigned char RANGE, unsigned char DIV32){
	MCGC2_EREFS = 1;
	MCGC2_ERCLKEN = 1;
	MCGC1_IREFS = 0;
	MCGC1_CLKS = 2;
	MCGC3_DIV32 = DIV32;
	MCGC2_RANGE = RANGE;
	MCGC1_RDIV = RDIV;
	MCGC2_LP = 0;
	MCGC3_PLLS = 0;
	MCGC2_BDIV = BDIV;
	MCGC4 = (DMX32 << 5)|(DRS);
}

void MCG_PBE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char
DIV32){
	MCGC2_EREFS = 1;
	MCGC2_ERCLKEN = 1;
	MCGC1_IREFS = 0;
	MCGC1_CLKS = 2;
	MCGC3_DIV32 = DIV32;
	MCGC2_RANGE = RANGE;
	MCGC1_RDIV = RDIV;
	MCGC2_LP = 0;
	MCGC3_PLLS = 0;
	MCGC2_BDIV = BDIV;
	MCGC3_VDIV = 2;
}

void MCG_PEE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char
DIV32){
	MCGC2_EREFS = 1;
	MCGC2_ERCLKEN = 1;
	MCGC1_IREFS = 0;
	MCGC1_CLKS = 0;
	MCGC3_DIV32 = DIV32;
	MCGC2_RANGE = RANGE;
	MCGC1_RDIV = RDIV;
	MCGC2_LP = 0;
	MCGC3_PLLS = 0;
	MCGC2_BDIV = BDIV;
	MCGC3_VDIV = 2;
}
```

Software de compensación de voltaje

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */

#define BUSCLK 4274000 // Frecuencia del bus del microcontrolador
#define SERIAL_TIMEOUT 500 // Tiempo en milisegundos para reiniciar buffer de comunicación serial
#define TRANSFER_DELAY 1 // Tiempo en milisegundos para la transición de buck a boost y viceversa
#define V10P 10. // Voltaje del primer paso de la precarga del ultracapacitor
#define V20P 20. // Voltaje del segundo paso de la precarga del ultracapacitor
#define V30P 30. // Voltaje del tercer paso de la precarga del ultracapacitor
#define V40P 40. // Voltaje del cuarto paso de la precarga del ultracapacitor
#define V50P 50. // Voltaje del quinto paso de la precarga del ultracapacitor

#define VultracapSet 60.0 // Setpoint voltaje ultracapacitor
#define VultracapMax 80.0 // Voltaje máximo del ultracapacitor
#define VultracapMin 40.0 // Voltaje mínimo del ultracapacitor


#define VSET_BUS 80.0 // Voltaje setpoint del bus de DC
#define ATENUACIONBUS 32.33333333333333 // Atenuación del sensor de voltaje del bus
#define ATENUACIONULTRACAP 32.33333333333333 // Atenuación del sensor de voltaje del ultracapacitor
#define TSAMP_MS 5 // Periodo de muestreo del sistema de control en milisegundos
#define Kp 50.0 // ganancia proporcional
#define Ki 2.0 // ganancia integral
#define Kanti 0.5 // ganancia anti windup
#define Dmaxboost 200.0 // Limite superior del ciclo de trabajo del conversor boost
#define Dmaxbuck -600.0 // Limite superior del ciclo de trabajo del conversor buck
#define PolarizacionBuck 0
#define PolarizacionBoost 500

unsigned char init_clock(void);
void MCG_FEI(unsigned char DRS, unsigned char DMX32, unsigned char BDIV);
void MCG_FEE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char BDIV, unsigned char RANGE, unsigned char DIV32);
void MCG_FBE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char BDIV, unsigned char RANGE, unsigned char DIV32);
void MCG_PBE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char DIV32);
void MCG_PEE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char DIV32);

float clamp_f(float min, float max, float x);
void init_timer(void);
void change_duty_buck(unsigned short int d);
void change_duty_boost(unsigned short int d);
void change_freq(unsigned short int f);
void init_rtc(void);
void rtcISR(void);
void serialISR(void);
void init_serial(int baudrate);
int int_pow(int a, int b);
void init_adc(void);
void send_char(char chr);
void send_str(const char *str);
void clear_buffer(void);
void write_serial_buffer(char chr);
int str_cmp(const char *str1, const char *str2, char ignore, char limiter);
int str2int_x3(char *str);
void stop_dcdc(void);
```

```
void num2str(unsigned int num, char *str, int len);
unsigned int read_adc(char channel);
void delay_ms(unsigned int delay);
void precarga(void);

char vultracap_state = 0;
/*
 * 0 -> buck on, boost on
 * 1 -> buck on, boost off
 * 2 -> boost on, buck off
 */
char buck_boost_state = 0;
char serial_buffer[30];
int serial_buffer_idx = 0;
char echo_enable = 0;
unsigned long int time_ms = 0;
unsigned int timeout_ms = 0;
unsigned int transfer_ms = 0;
char duty[5];



/*
 * 0 -> off
 * 1 -> buck
 * 2 -> boost
 */
void main(void) {
      float P=0, I=0, U=0, Ua=0, Ekm1=0, Ek=0, Vbus=0;
      float Ai, antiW = 0, Vultracap, Pol;
      unsigned int t_loop=0;
  char num_buffer[5];
  num_buffer[4] = 0;
  Ai = Ki * (TSAMP_MS / 2000.0);
  unsigned int adc_reading;
  EnableInterrupts;
  /* include your code here */
//  init_clock();

  unsigned long int i = 0;
  int j = 0;
  unsigned int adcread;
  //A = k * 1000;
  SOPT1_COPT = 0;
  /*** End of Processor Expert internal initialization.                    ***/
  init_timer();
  init_rtc();
  init_adc();
  change_freq(25000);
  change_duty_buck(0);
  t_loop = time_ms;
  precarga();
  for(;;) {
    __RESET_WATCHDOG();
    if(time_ms - t_loop >= TSAMP_MS){
            Vbus = read_adc(0) * ATENUACIONBUS / 819.;
            Vultracap = read_adc(1) * ATENUACIONULTRACAP / 819.;
      Ekm1 = Ek;
            Ek = (VSET_BUS - Vbus);
      P = Ek * Kp;
      I += Ai * (Ek + Ekm1) + antiW;
      U = P + I;
      Ua = clamp_f( Dmaxbuck ,Dmaxboost, U);
      antiW = (Ua - U) * Kanti;
```

```
	switch(vultracap_state){
	case 0:
		if(Vultracap > VultracapMax)
			vultracap_state = 2;
		else if(Vultracap < VultracapMin)
			vultracap_state = 2;
		else
			vultracap_state = 0;
		break;
	case 1:
		if(Vultracap >= VultracapSet)
			vultracap_state = 0;
		else
			vultracap_state = 1;
	    break;
	case 2:
		if(Vultracap <= VultracapSet)
			vultracap_state = 0;
		else
			vultracap_state = 2;
	    break;
	}
	if(Ua<=0){
		if(vultracap_state == 0 || vultracap_state == 1)
			change_duty_buck((int) ((-1 * Ua) + PolarizacionBuck) );
		else
			change_duty_buck(0);

	}else{
		if(vultracap_state == 0 || vultracap_state == 2)
			change_duty_boost((int) ((1. * Ua) + PolarizacionBoost));
		else
			change_duty_boost(0);
	}
	t_loop = time_ms;
    }
      } /* loop forever */
  /* please make sure that you never leave main */
}

float clamp_f(float min, float max, float x){
	if(x > max)
		return max;
	else if(x < min)
		return min;
	else
		return x;
}
void precarga(void){
	change_duty_buck(100);
	while(read_adc(1) * ATENUACIONULTRACAP / 819.0 < V10P){
		__RESET_WATCHDOG();
	}
	change_duty_buck(200);
	while(read_adc(1) * ATENUACIONULTRACAP / 819.0 < V20P){
		__RESET_WATCHDOG();
	}
	change_duty_buck(300);
	while(read_adc(1) * ATENUACIONULTRACAP / 819.0 < V30P){
		__RESET_WATCHDOG();
	}
	change_duty_buck(400);
	while(read_adc(1) * ATENUACIONULTRACAP / 819.0 < V40P){
		__RESET_WATCHDOG();
```

```
	}
	change_duty_buck(500);
	while(read_adc(1) * ATENUACIONULTRACAP / 819.0 < V50P){
		__RESET_WATCHDOG();
	}
}

int str2int_x3(char *str){
	unsigned int val = 0;
	val += (str[0] - '0') * 100;
	val += (str[1] - '0') * 10;
	val += (str[2] - '0');
	return val;
}

int int_pow(int a, int b){
	int r = 1, idx = 0;
	for(idx = 0; idx < b; idx++)
		r *= a;
	return r;
}
void num2str(unsigned int num, char *str, int len){
	unsigned int tmp;
	int idx;
	tmp = num;
	for(idx = len; idx > 0; idx--){
		str[len - idx] = (char) (tmp / int_pow(10, idx - 1));
		tmp -= str[len - idx] * int_pow(10, idx - 1);
		str[len - idx] += '0';
	}
}
void init_rtc(void){
	RTCSC_RTCLKS = 0;
	RTCSC_RTIE = 1;
	RTCSC_RTCPS = 8;
	RTCMOD = 0;
}

void init_adc(void){
	ADCCFG_ADICLK = 1;
	ADCCFG_ADLPC = 0;
	ADCCFG_ADIV = 3;
	ADCCFG_ADLSMP = 1;
	ADCCFG_MODE = 1;
	ADCSC1_ADCH = 0xF;
	ADCSC2_ADTRG = 0;
	APCTL1_ADPC0 = 1;
	APCTL1_ADPC1 = 1;
}

unsigned int read_adc(char channel){
	ADCSC1_ADCH = channel;
	while(!ADCSC1_COCO){};
	return ADCR;
}

interrupt VectorNumber_Vrtc void rtcISR(void){
	RTCSC_RTIF= 1;
	if(time_ms < 100000000L)
		time_ms++;
	else
		time_ms=0;
	if(time_ms - timeout_ms >= SERIAL_TIMEOUT){
		clear_buffer();
```

```
	}
}

void init_serial(int baudrate){
	SCI1C2_TE = 1;
	SCI1C2_RE = 1;
	SCI1C2_RIE = 1;
	SCI1BD = ((BUSCLK*10)/(baudrate*16) + 5)/10;
}

void send_char(char chr){
	while(SCI1S1_TDRE == 0){};
	SCI1D = chr;
	while(SCI1S1_TC==0){};
}

void send_str(const char *str){
	char i = 0;
	while(str[i] != '\0')
		send_char(str[i++]);
}

int str_cmp(const char *str1,const char *str2, char ignore, char limiter){
	int i = 0, ret = 0;
	while((str1[i] != limiter && str2[i] != limiter && str1[i] == str2[i]) || str1[i] ==
ignore || str2[i] == ignore)
		i++;
	ret = str1[i] == str2[i] & str1[i] == limiter & str2[i] == limiter;
	return ret;
}

void write_serial_buffer(char chr){
	serial_buffer[serial_buffer_idx] = chr;
	if(serial_buffer_idx < 30)
		serial_buffer_idx++;
	else
		clear_buffer();
}

void clear_buffer(void){
	int i = 0;
	for(i = 0; i < 30; i++)
		serial_buffer[i] = '\0';
	serial_buffer_idx = 0;
}

interrupt VectorNumber_Vsci1rx void serialISR(void){
	char tmp;
	(void)SCI1S1;
	tmp = SCI1D;
	write_serial_buffer(tmp);
	timeout_ms = time_ms;
	if(echo_enable){
		send_char(tmp);
	}
}

void init_timer(void){
	TPM1C0V = 0;
	TPM1C1V = 0;
	buck_boost_state = 0;
	TPM1SC = 0x88; //0b0000 1000
	TPM1MOD = 0x100;
	TPM1C1SC = 0b00101000;//
```

```
	TPM1C0SC = 0b00101000;
}
void delay_ms(unsigned int delay){
	unsigned int time_delay_0 = time_ms;
	while(delay > time_ms - time_delay_0)
	    __RESET_WATCHDOG();
}
void change_duty_buck(unsigned short int d){
	unsigned short int v = 0;
	int cnt = 0;
	if(buck_boost_state == 2 || buck_boost_state == 0){
		TPM1C0V = 0;
		delay_ms(TRANSFER_DELAY);
		buck_boost_state = 1;
	}
	v = (TPM1MOD * d)/1000;
	TPM1C1V = v;
}

void stop_dcdc(void){
	TPM1C0V = 0;
	TPM1C1V = 0;
	buck_boost_state = 0;
}

void change_duty_boost(unsigned short int d){
	unsigned short int v = 0;
	int cnt = 0;
	if(buck_boost_state == 1 || buck_boost_state == 0){
		TPM1C1V = 0;
		delay_ms(TRANSFER_DELAY);
		buck_boost_state = 2;
	}
	v = (TPM1MOD * d)/1000;
	TPM1C0V = v;
}

void change_freq(unsigned short int f){
	/*
	 * FREQ (Hz)
	 */
	unsigned short int Tmod;
	(void)TPM1MOD;
	Tmod = (BUSCLK/(f))-1;
	TPM1MOD = Tmod;
	while(TPM1MOD != Tmod){
	    __RESET_WATCHDOG();
	};

}

unsigned char init_clock(void){
	SOPT2_CLKOUT_EN = 1;
	MCG_FEI(2, 1, 0);
	MCG_FBE(3, 2, 1, 0, 1, 1);
	MCG_PBE(3, 0, 1, 1);
	MCG_PEE(3, 0, 1, 1);

	return 1;
}

void MCG_FEI(unsigned char DRS, unsigned char DMX32, unsigned char BDIV){
	MCGC1 = 0x04;
	MCGC3_PLLS = 0;
```

```
	MCGC4 = (DMX32 << 5)|(DRS);
	MCGC2_BDIV = BDIV;
}

void MCG_FEE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char
BDIV, unsigned char RANGE, unsigned char DIV32){
	MCGC1 = ((RDIV & 0x7) << 3);
	MCGC2_BDIV = BDIV;
	MCGC2_RANGE = RANGE;
	MCGC3_PLLS = 0;
	MCGC3_DIV32 = DIV32;
	MCGC4 = (DMX32 << 5)|(DRS);

}

void MCG_FBE(unsigned char RDIV, unsigned char DRS, unsigned char DMX32, unsigned char
BDIV, unsigned char RANGE, unsigned char DIV32){
	MCGC2_EREFS = 1;
	MCGC2_ERCLKEN = 1;
	MCGC1_IREFS = 0;
	MCGC1_CLKS = 2;
	MCGC3_DIV32 = DIV32;
	MCGC2_RANGE = RANGE;
	MCGC1_RDIV = RDIV;
	MCGC2_LP = 0;
	MCGC3_PLLS = 0;
	MCGC2_BDIV = BDIV;
	MCGC4 = (DMX32 << 5)|(DRS);
}

void MCG_PBE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char
DIV32){
	MCGC2_EREFS = 1;
	MCGC2_ERCLKEN = 1;
	MCGC1_IREFS = 0;
	MCGC1_CLKS = 2;
	MCGC3_DIV32 = DIV32;
	MCGC2_RANGE = RANGE;
	MCGC1_RDIV = RDIV;
	MCGC2_LP = 0;
	MCGC3_PLLS = 0;
	MCGC2_BDIV = BDIV;
	MCGC3_VDIV = 2;
}

void MCG_PEE(unsigned char RDIV, unsigned char BDIV, unsigned char RANGE, unsigned char
DIV32){
	MCGC2_EREFS = 1;
	MCGC2_ERCLKEN = 1;
	MCGC1_IREFS = 0;
	MCGC1_CLKS = 0;
	MCGC3_DIV32 = DIV32;
	MCGC2_RANGE = RANGE;
	MCGC1_RDIV = RDIV;
	MCGC2_LP = 0;
	MCGC3_PLLS = 0;
	MCGC2_BDIV = BDIV;
	MCGC3_VDIV = 2;
}
```