Revista Politécnica ISSN 1900-2351 (Impreso), ISSN 2256-5353 (En línea), Año 20, Número 40, páginas 154-172, Julio-Diciembre 2024

DOI: https://doi.org/10.33571/rpolitec.v20n40a10

DISEÑO E IMPLEMENTACIÓN EN LA PLATAFORMA ARDUINO DE ESTIMADORES DE ESTADO PARA CONVERTIDORES ELECTRÓNICOS DC-DC

Alexandre Dimnet[1], Héctor Antonio Botero-Castro[2]

[1.] M.Sc. Universidad Nacional de Colombia, Facultad de Minas, Departamento de Energía Eléctrica y Automática, Colombia. Correo de correspondencia: adimnet@unal.edu.co

[2.] Ph.D. Universidad Nacional de Colombia, Facultad de Minas, Departamento de Energía Eléctrica y Automática, Colombia.

RESUMEN

En este trabajo se realiza la implementación de observadores de estado en una plataforma integrada que combina Matlab®, Simulink® y Arduino®, mediante la cual se logra estimar con precisión las variables de estado de un convertidor electrónico. La planta es emulada en Matlab - Simulink, mientras que los observadores son implementados en Arduino. Para ello, se diseñan tres tipos de observadores: el observador de Luenberger, el filtro de Kalman y el observador por modos deslizantes, para un convertidor DC-DC boost. Los resultados son evaluados, permitiendo resaltar sus ventajas y desventajas para la obtención de las variables de estado del convertidor. La metodología utilizada ha demostrado ser fácilmente reproducible, lo que abre la posibilidad de futuras investigaciones y aplicaciones prácticas en el campo de la electrónica de potencia y el control.

Palabras clave: Estimación de estado, convertidor electrónico de potencia, observador de Luenberger, filtro de Kalman, observador por modos deslizantes, linealización.

Recibido: 22 de febrero de 2024. Aceptado: 25 de mayo de 2024
Received: February 22, 2024. Accepted: May 25, 2024

DESIGN AND IMPLEMENTATION OF STATE ESTIMATORS FOR DC-DC CONVERTERS BY USING ARDUINO BOARD

ABSTRACT

In this work, the implementation of state observers in an integrated platform combining Matlab®, Simulink® and Arduino® is carried out, by means of which the state variables of an electronic converter can be accurately estimated. The plant is emulated in Matlab - Simulink, while the observers are implemented in Arduino. For this purpose, three types of observers are designed: the Luenberger observer, the Kalman filter and the sliding mode observer, for a DC-DC boost converter. The results are evaluated, allowing to highlight their advantages and disadvantages for obtaining the state variables of the converter. The methodology used has proved to be easily reproducible, which opens the possibility of future research and practical applications in the field of power electronics and control.

Keywords: *State estimation, power electronic converter, Luenberger observer, Kalman filter, sliding mode observer, linearization.*

Cómo citar este artículo: A. Dimnet, H. Botero-Castro. "Diseño e implementación en la plataforma Arduino de estimadores de estado para convertidores electrónicos DC-DC", Revista Politécnica, vol.20, no.40 pp.154-172, 2024. DOI:10.33571/rpolitec.v20n40a10

1. INTRODUCCCIÓN

Los convertidores electrónicos son utilizados para modificar la tensión y la corriente en un sistema, es decir son equivalentes a los transformadores pero en corriente continua, lo cual permite una mayor eficiencia y flexibilidad en el diseño de sistemas electrónicos. Estos dispositivos electrónicos son utilizados para convertir una tensión continua de un valor a otro y son ampliamente utilizados en sistemas electrónicos para alimentar circuitos desde diferentes fuentes de energía, como baterías, generadores de corriente continua, paneles solares y sistemas de distribución eléctrica [1] [2]. Existen diferentes tipos de convertidores DC-DC, cada uno con sus propias ventajas e inconvenientes en términos de rendimiento, costo y complejidad. En ese sentido están divididos en tres familias: buck, boost y buck-boost [1]. Los del tipo buck reducen la tensión de entrada para adaptarla a la tensión de salida deseada, mientras que los boost aumentan la tensión de entrada. Los tipo buck-boost pueden utilizarse para ambos tipos de conversión de tensión.

Como caso particular, un convertidor boost es un circuito electrónico que convierte una fuente de energía de baja tensión en una fuente de energía de alta tensión. El funcionamiento básico de este componente se basa en el uso de un inductor y un capacitor para almacenar y liberar energía eléctrica. Sin embargo, estos dispositivos tienen limitaciones en cuanto a su precisión y estabilidad, ya que son sensibles a los cambios en el voltaje de la fuente de alimentación, los cambios en la carga y las variaciones en sus parámetros [3]. Como el rendimiento de estos dispositivos puede ser afectado por dichas variaciones, se necesitan sistemas de control que permitan mantener las variables en los niveles requeridos. Estos sistemas de control requieren a su vez la medición de las variables de estado, lo cual no siempre es fácil de conseguir debido al retardo y al ruido. Durante mucho tiempo se usaban sensores para medir la tensión de entrada y la carga de salida gracias a la medición de la corriente de salida. Pero usar un sensor tiene muchas desventajas, ya que el uso de un sensor aumenta el costo de compra y hay que adquirir equipamiento adicional. Esto también conlleva gastos de mantenimiento y una menor vida útil de todo el sistema. Además, aumenta la complejidad del diseño a la vez que reduce la confiabilidad. Por estas razones, hoy en día se utilizan sensores virtuales en la mayoría de los casos [4]. Estos sensores virtuales pueden parametrizarse y, por tanto, ajustarse al sistema en el que están integrados, pero también pueden actualizarse fácilmente para sustituirlos si el sistema cambia o queda obsoleto. Además, es más económico porque no es hardware sino una combinación de hardware y software. Esos sensores virtuales son conocidos como observadores o estimadores de estado. Los estimadores de estado son algoritmos utilizados para estimar el estado de un sistema dinámico, como los convertidores DC-DC. Estos algoritmos permiten seguir las variaciones de la tensión de entrada y la corriente de salida en tiempo real, utilizando métodos de filtrado y control para estimar las variables de estado del sistema. Los estimadores de estado suelen basarse en modelos matemáticos del sistema, como ecuaciones diferenciales o modelo de espacio de estados [5].

Existen diferentes tipos de estimadores de estado, cada uno con sus propias ventajas e inconvenientes en términos de precisión y complejidad. Los observadores de estado son algoritmos similares a los estimadores de estado, pero enfocados en estimar las variables de estado ocultas del sistema utilizando las mediciones disponibles de las entradas y salidas. Estos algoritmos son utilizados para detectar errores de medición y anomalías en el sistema, lo que contribuye a evitar daños al convertidor y a mejorar su confiabilidad [6].

El artículo [7] presenta un diseño de un observador de estado mediante modos deslizantes que aplica un movimiento deslizante a las variables medidas y una dinámica asintótica para las variables no medidas. Además, con esta técnica se logra estimar la resistencia de la carga cuando se mide el voltaje y la corriente de salida del convertidor. En la fuente [8] se describe un observador robusto que puede trabajar con incertidumbre en la resistencia de la carga. Este observador utiliza un modelo considerando varios parámetros de pérdidas eléctricas del convertidor. Se utiliza la técnica de modos deslizantes para estimar la corriente de forma robusta. En el artículo [9], se propone un observador

basado en el filtro de Kalman. Allí, se utiliza un módulo de eliminación de la variación de la resistencia de carga y se estima la resistencia mediante el uso de la ecuación estacionaria que relaciona la corriente del inductor con la corriente de la carga, basándose en los valores estimados de voltaje y corriente de salida.

Un observador de Luenberger no lineal se diseña en el artículo [10]. La estructura del observador incluye una copia del modelo y un factor de corrección, pero este tiene la desventaja de solo afectar la dinámica del voltaje en el condensador que es la variable medida. En [6], se propone un observador de corriente robusto no lineal para convertidores boost, buck y buck-boost, que es evaluado mediante el criterio de Lyapunov para determinar su estabilidad. Las pruebas muestran un buen rendimiento ante variaciones en la carga en diferentes tipos de convertidores, y fueron realizadas tanto mediante simulación como análisis experimental.

Aunque mucho del trabajo sobre estimadores de estado en convertidores se hace mediante simulación, varios trabajos han sido hechos para implementar estos en plataformas de desarrollo. En [11] se implementó una plataforma utilizando Matlab-Simulink y una tarjeta Advantech para la estimación y el control de un convertidor. El estimador de estado trabaja con el principio de inmersión invarianza. Este trabajo fue mejorado en [12], donde se implementó un observador de estado para estimar tanto el voltaje de la fuente como la resistencia de la carga de un convertidor boost. En [13] se implementó un prototipo de control por modos deslizantes adaptativo robusto. Todas estas publicaciones están al alcance de un público muy amplio, sin embargo, los detalles relacionados con la implementación de los observadores no son explícitos. Por lo tanto, existe todavía un vacío en este campo, el cual se trata de resolver con este artículo.

En ese sentido, el objetivo de este trabajo es utilizar una tarjeta Arduino para la implementación de estos observadores, aprovechando la capacidad que tienen estos dispositivos para manejar cálculos y operaciones complejas, así como su facilidad de programación y su bajo costo. Además, es una plataforma abierta y con una gran comunidad, lo que facilita el desarrollo y la implementación de proyectos relacionados con el control y el procesamiento de señales. Asimismo, para facilitar el uso de la plataforma a cualquier usuario, el convertidor es emulado usando Matlab – Simulink, logrando una simulación muy cercana del circuito real, lo cual permite probar los observadores antes de su implementación en el sistema real [3] [14] [15] [16] [17]. Lo anterior posibilita extender el resultado a otro tipo de plantas diferentes a la que se usa aquí, por ejemplo a procesos químicos o biológicos.

El resto del artículo está organizado como se explica a continuación. Inicialmente se aborda el funcionamiento del convertidor DC-DC. Esto implica la caracterización y modelado del sistema. Una vez establecido el modelo del sistema, se agrupan los observadores de estado relevantes y se integran en los cálculos. El siguiente paso consiste en implementar este modelo y su observador en un entorno de simulación Matlab (Simulink), y en una plataforma de prototipado Arduino. Por último, los resultados obtenidos en las pruebas son analizados y comparados para evaluar la eficacia de las diferentes soluciones propuestas. Esto permite determinar la mejor configuración para el observador, así como los puntos de mejora potenciales para el sistema. Finalmente se explican las conclusiones detalladas sobre los resultados obtenidos, incluyendo observaciones y recomendaciones para investigaciones futuras.

2. MATERIALES Y MÉTODOS

En esta sección se mostrará el modelo del convertidor y de los observadores de estado que van a ser implementados en Arduino. Como el enfoque de este artículo se centra en la implementación, solo se describen de manera parcial los diseños de los observadores, los cuales pueden ser consultados en la referencia [16] de esta misma revista.

2.1 Modelo del convertidor

El circuito de un convertidor DC-DC boost consta de cuatro componentes principales: un interruptor (generalmente un transistor MOSFET), una bobina, un diodo y un condensador. En la Figura 1 se puede ver una representación del mismo.

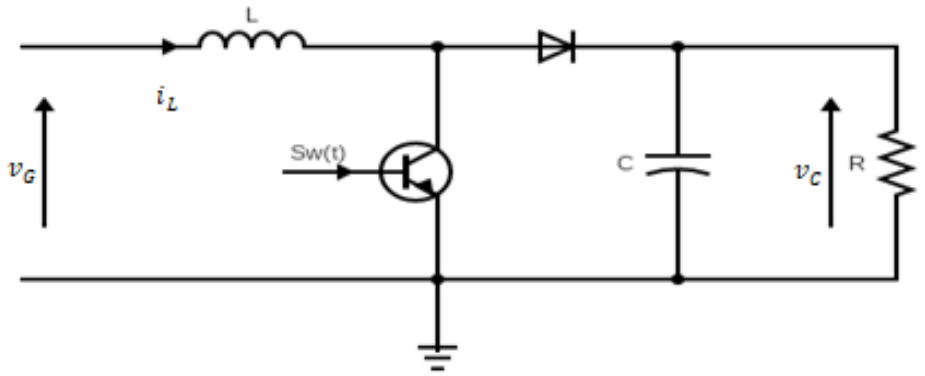

Figura 1. Circuito de un convertidor boost con diodo y transistor.

El convertidor de potencia DC-DC boost al ser un sistema conmutado genera cambios en la dinámica del sistema, dependiendo de la posición del interruptor. Por lo tanto, el modelo completo del convertidor incluye una variable relacionada con la apertura o cierre del interruptor que es idealmente una función a tramos de dos valores constantes: cero y uno. Debido a que en este caso el interés es conocer los valores medios de las variables de estado del convertidor de potencia, se realiza una simplificación al modelo y se utiliza un modelo promediado que remplaza el transistor por un interruptor. Cabe recalcar que este modelo promediado se usa especialmente en la representación en el espacio de estados para el análisis dinámico. Para el diseño del convertidor, es decir, para encontrar los valores de la inductancia, la capacitancia, el tipo de transistor, entre otros, se requiere conocer cómo es la variación de la dinámica de las variables de estado respecto a su valor medio, conocida como rizado [16].

El Modelo promediado reducido proporciona una descripción matemática que involucra las dinámicas del convertidor. La necesidad del mismo surge a raíz de la complejidad del modelo completo, que incluye detalles como las resistencias internas de los componentes, diodos y transistores. Estos detalles pueden ser innecesarios para ciertas aplicaciones, y su inclusión puede resultar en ecuaciones demasiado complejas para la simulación en tiempo real. Por lo tanto, se trabaja aquí con un equivalente reducido que permite una simulación más rápida y sencilla. En este se ignoran las resistencias internas de los componentes, el diodo y se reemplaza el transistor por un simple interruptor. Estas simplificaciones permiten una implementación más sencilla del modelo y facilitan la implementación del observador de estado en una plataforma como Arduino.

Las ecuaciones del modelo quedan entonces de la forma (1) y (2):

$$\frac{di_L}{dt} = \frac{(D-1)v_C + v_G}{L} \tag{1}$$

$$\frac{dv_C}{dt} = \frac{(1-D)i_L - \frac{v_C}{R}}{C} \tag{2}$$

De esta forma se presenta el modelo presentado en los artículos [3] y [16]. Para ello se utilizarán como variables de estado: $X(t) = \begin{bmatrix} i_L(t) \\ v_C(t) \end{bmatrix}$

Y como variables de entrada: $U(t) = \begin{bmatrix} v_G(t) \\ D(t) \end{bmatrix}$

El modelo es bilineal porque hay multiplicación entre variables de estado y entradas. Por lo tanto, se linealiza en un punto de equilibrio ($\mathrm{I_{Lee}} = 0.4\ \mathrm{A}, \mathrm{V_{Cee}} = 2\ \mathrm{V}, \mathrm{V_{Gee}} = 4\ \mathrm{V}, \mathrm{D_{ee}} = 0{,}5$) obteniéndose la representación en el espacio de estados:

$$\dot{X}(t) = A(t)X(t) + B(t)U(t) \quad (3)$$
$$Y(t) = C(t)X(t) + D(t)U(t) \quad (4)$$

Tomando los valores de los parámetros de [16]:

$$\mathrm{R} = 20\ \Omega\ , \mathrm{L} = 120\ \mu\mathrm{H}\ , \mathrm{C} = 75\ \mu\mathrm{F}$$

Se obtienen las matrices A y B evaluadas en este punto de equilibrio:

$$A_{ee} = \begin{bmatrix} 0 & \frac{-0.5}{120\,.\,10^{-6}} \\ \frac{0.5}{75\,.\,10^{-6}} & -\frac{1}{1.5\,.\,10^{-3}} \end{bmatrix}$$

$$B_{ee} = \begin{bmatrix} \frac{1}{120\,.\,10^{-6}} & \frac{1}{30\,.\,10^{-6}} \\ 0 & -\frac{0.4}{75\,.\,10^{-6}} \end{bmatrix}$$

$$C_{ee} = [0\ 1]$$

$$D_{ee} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Los valores propios de la matriz A_{ee}, son: $\lambda_{1,2} = -333.33\ (+/-)\ 5259.92i$ lo cual origina un tiempo de establecimiento de aproximadamente 0.015 s. Este dato se utilizará en la selección del tiempo de muestreo y las dinámicas de los estimadores.

2.2. Observadores de estado en Matlab-Simulink

En el caso del convertidor DC-DC boost, el observador se implementa utilizando la retroalimentación de la tensión de salida del convertidor $v_C(t)$ y la inyección de las variables de entrada $v_G(t), D(t)$. Esto permite calcular y estimar la corriente en el inductor, que es una variable de estado de difícil medición. Los cálculos para los observadores se siguieron según la referencia [16]. En ese sentido, tres observadores de estado fueron diseñados como se explica a continuación.

Observador de Luenberger:

La Figura 2 muestra el diagrama de bloques del observador de Luenberger simulado con el modelo lineal del convertidor. Una de las principales ventajas del Observador de Luenberger es su simplicidad y facilidad de implementación. Además, este observador no requiere conocimientos previos sobre el ruido y las perturbaciones en el sistema, lo que lo hace muy adecuado para sistemas en los que no se tiene información detallada sobre ello. Sin embargo, una de las principales limitaciones es que no es adecuado para sistemas en los que las variables de estado cambian rápidamente, ya que la retroalimentación de la variable de salida puede introducir errores en la

estimación del estado. La idea del diseño es tener un observador con una dinámica más rápida que el sistema real, modificando la matriz de ganancia L.
En este caso se usa una ubicación de polos que sea dos veces el valor absoluto de la magnitud de los polos de la planta. Eso significa que los polos del observador son elegidos más alejados y por lo tanto más rápidos que los de la matriz A del sistema. Las matrices B y C del observador de Luenberger son iguales que las del sistema. De esta manera se puede calcular la matriz de ganancia L con la función acker() de Matlab. El observador tiene la forma siguiente [18]:

$$\dot{\hat{X}} = A\hat{X}(t) + BU(t) + L\left(y(t) - C\hat{X}(t)\right) \tag{5}$$

Donde $\hat{X}$ es una estimación del X real y L es la matriz de ganancia para que el observador se corrija adecuadamente. En este caso, $L = \begin{bmatrix} 12500 \\ 20415.18 \end{bmatrix}$. Un montaje de Simulink que representa este observador se muestra en la Figura 2.

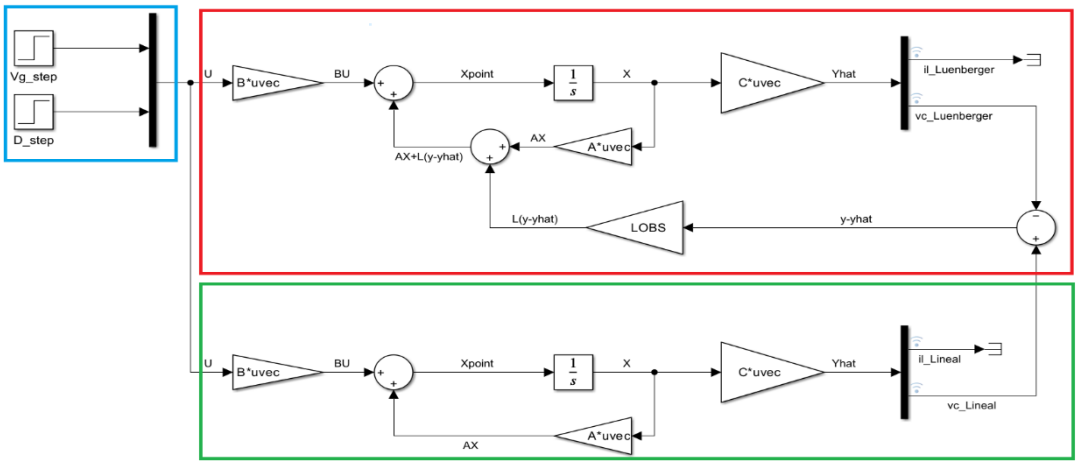

Figura 2. Diagrama de Simulink para el observador de Luenberger

Filtro de Kalman

El filtro de Kalman es un tipo de observador que utiliza un modelo probabilístico del sistema y del ruido para realizar la estimación del estado. Este observador fue desarrollado por Rudolf Kalman en 1960 y se utiliza ampliamente en la teoría de control y en aplicaciones de procesamiento de señales. Este observador se basa en el modelo matemático del sistema y utiliza mediciones de las entradas y salidas del sistema para estimar el estado interno. El diseño del filtro se basa en el uso de matrices de covarianza para modelar la incertidumbre en el sistema y el ruido de medición. Estas matrices representan las incertidumbres en las mediciones y en el modelo matemático del sistema.

En el caso del convertidor DC-DC boost, el filtro de Kalman se puede utilizar para estimar el estado del sistema a partir de las mediciones de voltaje de entrada y del ciclo de trabajo en la entrada, y la tensión de salida del sistema. La implementación de tal observador en el sistema requiere de un procesamiento computacional significativo, ya que se deben calcular las matrices de covarianza y actualizar la estimación del estado en tiempo real. Sin embargo, una vez que se ha diseñado e implementado correctamente, el observador de Kalman puede proporcionar una estimación precisa y robusta del estado del sistema, incluso en presencia de ruido y otras perturbaciones. La Figura 3 muestra el diagrama de bloques del observador de Kalman implementado en el convertidor.

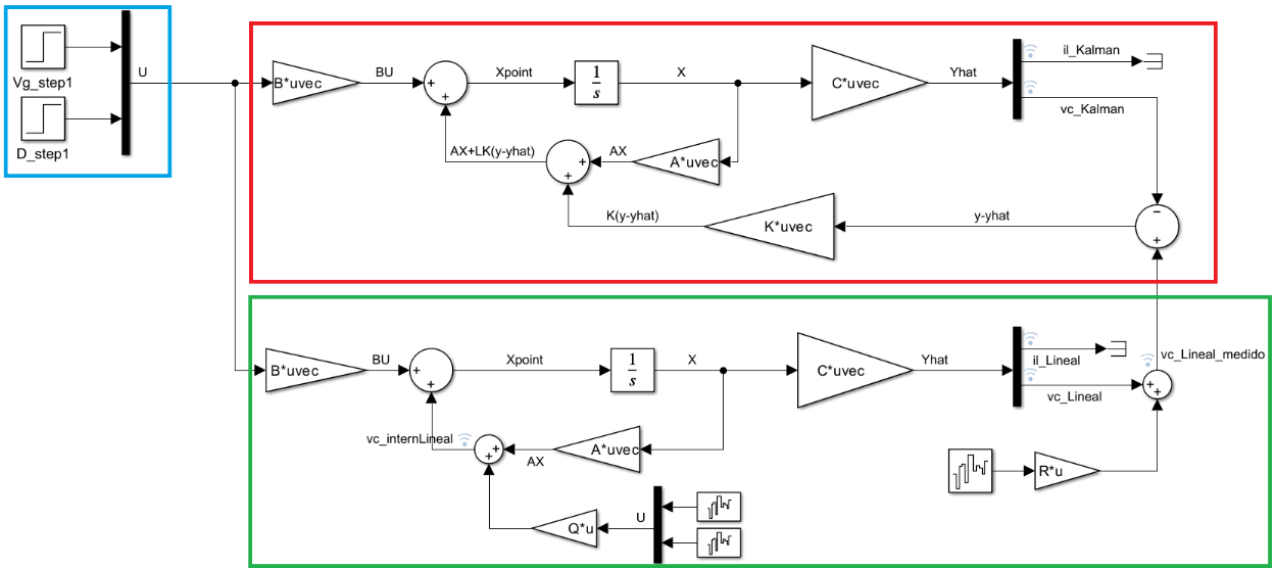

Figura 3. Diagrama de Simulink para el filtro de Kalman

Como el filtro de Kalman usa datos estadísticos para corregirse, se añaden ruidos en el sistema w_1 y w_2 que representan el ruido de proceso del sistema para ambas variables de estado y también se considera un ruido de medición v_1 que se aplica a la salida. Así que el sistema se escribe como se muestra en (6) a (8):

$$\frac{dv_C}{dt} = \frac{(1-D)i_L}{C} - \frac{v_c}{RC} + w_1 \tag{6}$$

$$\frac{di_L}{dt} = \frac{(D-1)v_C}{L} + \frac{v_G}{L} + w_2 \tag{7}$$

$$y = v_C + v_1 \tag{8}$$

De manera formal, w_1 y w_2 tienen covarianzas asociadas a una matriz *Q y* v_1 asociada a una matriz R. Como la ecuación de v_C contiene la resistencia de la carga, la cual es variable, se considera que esta ecuación tiene más ruido de proceso es decir incertidumbre. Por lo tanto las matrices son tomadas de la siguiente forma:

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.2 \end{bmatrix} \tag{9}$$

$$R = R_1 = 0.1 \tag{10}$$

Con la función kalman() de Matlab se obtiene la matriz de ganancia K del filtro. Como se dijo antes, el filtro de Kalman calcula estas ganancias en cada tiempo de muestreo, lo cual requiere bastante potencia de cálculos, y en este proyecto se va a utilizar un hardware limitado en tiempo como Arduino. Para evitar entonces un bloqueo, se utiliza un observador Kalman estacionario, es decir, las ganancias del filtro se calculan una sola vez y se guardan para el resto de la ejecución. Los valores obtenidos son:

$$K_1 = 43885.67$$

$$K_2 = 24680.43$$

Observador por modos deslizantes

El observador por modos deslizantes es especialmente útil para sistemas con discontinuidades en su comportamiento, como el sistema de control del convertidor y se basa en la idea de una "superficie deslizante" que permite la estimación del estado interno. La superficie deslizante es una función matemática que se construye para que la dinámica del sistema se "deslice" sobre ella. La función de superficie deslizante se elige de tal manera que se cancelan los términos no lineales del sistema, y se obtiene una dinámica lineal para la estimación del estado.

El diseño de este observador implica la elección de la función de superficie deslizante y la selección de los parámetros del observador para garantizar la estabilidad del sistema y una buena estimación del estado interno. El observador se puede representar matemáticamente como un sistema de dos ecuaciones diferenciales no lineales. La primera ecuación describe la evolución de la superficie deslizante, mientras que la segunda ecuación describe la evolución del estado estimado. En el caso del convertidor DC-DC boost, el observador se puede implementar utilizando una función de superficie deslizante que tenga en cuenta la dinámica no lineal del convertidor y la carga. Una vez que se ha seleccionado la función de superficie deslizante, se pueden determinar los parámetros del observador para garantizar la estabilidad del sistema. La estimación del estado se realiza utilizando la superficie deslizante y las entradas y salidas del sistema.

En el aspecto matemático, el observador está representado como otro sistema en una representación de estado. Pero la idea es tener un observador que usa la salida del sistema y una matriz de ganancia L para corregirse, es decir tipo Luenberger, pero con un modo deslizante incorporado. El observador genera entonces dinámicas independientes para las variables de estados medibles y no medibles de la forma siguiente:

$$\dot{\hat{X}} = A\hat{X}(t) + BU(t) + Lsign\big(y(t) - \hat{y}(t)\big) \tag{11}$$

Donde $\hat{X}$ es una estimation del X real, L es una matriz de ganancia de la forma siguiente [19]:

$$L = \begin{bmatrix} L_2 L_1 \\ L_1 \end{bmatrix} \tag{12}$$

De manera más detallada se puede escribir el observador de esta manera:

$$\frac{\widehat{di_L}}{dt} = A_{11}\hat{y} + A_{12}\hat{x} + B_1 u + L_2 L_1 sign(y - \hat{y}) \tag{13}$$

$$\frac{\widehat{dv_C}}{dt} = A_{21}\hat{y} + A_{22}\hat{x} + B_2 u + L_1 sign(y - \hat{y}) \tag{14}$$

Como la salida del modelo, es decir la variable medida, corresponde a v_C, se toma como superficie deslizante este error igualado a cero ($e_{v_C} = 0$), cuya dinámica se obtiene restando (2) y (14):

$$\frac{de_{v_C}}{dt} = -\frac{1}{RC}e_{v_C} + \frac{1-D}{C}e_{i_L} - L_1 sign\big(e_{v_C}\big) \tag{15}$$

La ganancia L_1 se selecciona de manera tal que exista un compromiso entre el chattering y la precisión requerida. La selección de L_1 se hace por medio de simulación, verificando que si L_1es pequeña se le quita importancia al modo deslizante y si es grande se aumenta el chattering, lo cual hace que el observador no se pueda acercar al valor final sino que oscile entre uno superior y uno inferior. Para la selección de L_2 se aplica el concepto de control equivalente descrito en [19], y se obtiene la dinámica del error de la parte no medible así:

$e_{v_C} = 0 \Leftrightarrow \frac{de_{v_C}}{dt} = 0$, en la ecuación (15) produce:

$$0 = 0 + \frac{1-D}{C} e_{i_L} - L_1 sign\left(e_{v_C}\right)_{eq} \quad \Leftrightarrow \quad L_1 sign\left(e_{v_C}\right)_{eq} = \frac{1-D}{C} e_{i_L}$$

$$\frac{de_{i_L}}{dt} = \frac{D-1}{L} e_{v_C} - L_2 L_1 sign\left(e_{v_C}\right) \tag{16}$$

$$\frac{de_{i_L}}{dt} = \frac{D-1}{L} e_{v_C} - L_2 \frac{1-D}{C} e_{i_L} \tag{17}$$

Cuando la trayectoria de estado alcanza la superficie deslizante el $e_{v_C} = 0$, entonces:

$$\frac{de_{i_L}}{dt} = 0 - L_2 \frac{1-D}{C} e_{i_L} \quad \Leftrightarrow \quad \frac{de_{iL}}{dt} = -\alpha e_{i_L}$$

Para determinar L_2 es necesario que α sea un polo más rápido que los del sistema original para que la corrección del error sea bastante rápida para converger. En este caso se elige dos veces más rápido para este observador, así que:

$$L_2 = 2 \times |polo(A)| \times \frac{C}{1-D} \approx 1.58$$

Con $L_1 = 100$ los resultados obtenidos cumplen el criterio definido arriba respeto a chattering y error. En la Figura 4 se puede ver cómo está representado el Observador en Simulink.

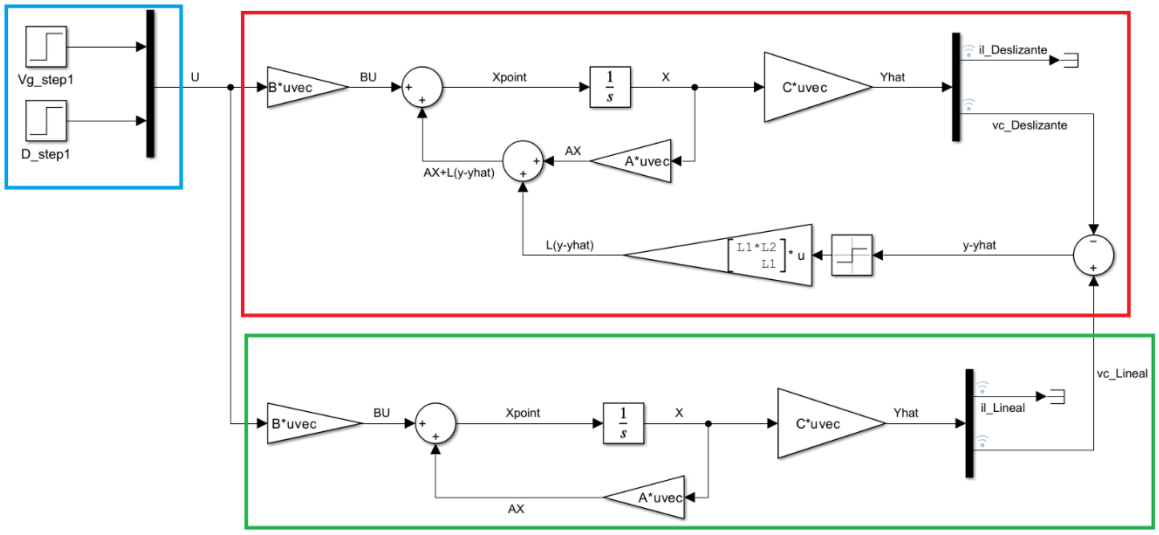

Figura 4. Diagrama de Simulink para el observador por modos deslizantes

Los observadores descritos hasta aquí serán simulados para verificar si la implementación de los mismos en Arduino funciona correctamente o no. Esta para se verificará en la Sección 3.

2.3. Implementación de observadores en Matlab-Arduino

Arduino Uno es una placa de microcontrolador, de código abierto, que ha ganado popularidad debido a su simplicidad y versatilidad en una amplia variedad de aplicaciones. El procesador está diseñado para ser fácil de programar y utilizar, por lo cual este ha sido ampliamente adoptado por aficionados, estudiantes, y profesionales en campos como la electrónica, la robótica y la automatización. La importancia de la comunicación entre Arduino Uno y Matlab radica en la capacidad de aprovechar

las ventajas de ambas plataformas para realizar de forma simultánea la implementación de los estimadores en Arduino Uno y la simulación del modelo en Matlab-Simulink. Por lo tanto, el código de los estimadores se alberga en el procesador de Arduino, mientras que Matlab-Simulink sirve como entorno de programación del modelo del convertidor DC-DC y simulación (emulación de la planta), a la vez que ofrece poderosas herramientas para el análisis y procesamiento, así como la generación de gráficos y visualizaciones.

La comunicación entre Arduino Uno y Matlab se realiza mediante bloques de Simulink. Al establecer una comunicación fluida entre estas dos plataformas, es posible aprovechar las fortalezas individuales de cada una y combinarlas para lograr resultados en un entorno simulado cercano a la realidad. Las especificaciones y características de Arduino Uno aparecen el manual del fabricante el cual puede ser consultado en Internet. El esquema de implementación del modelo del convertidor junto con la comunicación se ilustra en la Figura 5.

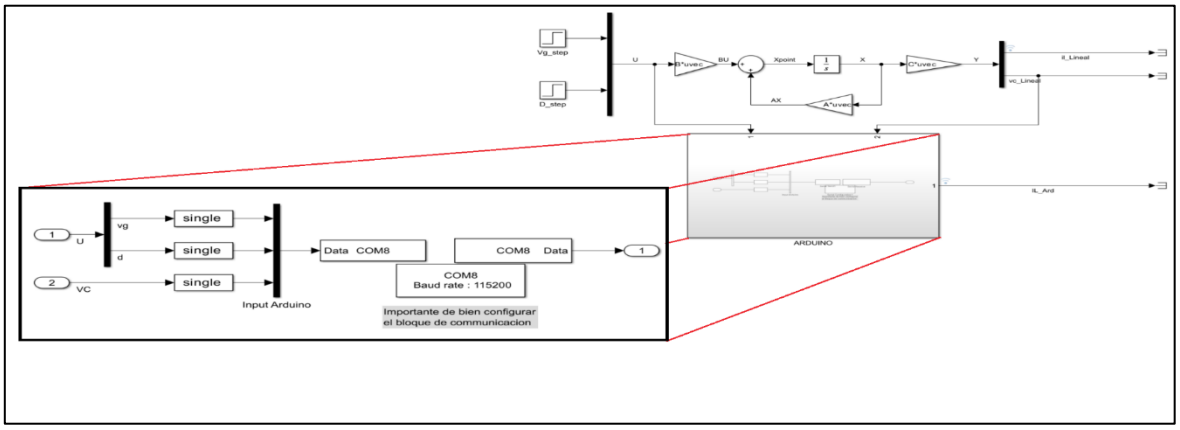

Figura 5. Diagrama del enlace de Simulink con el Arduino UNO

Para la implementación se debe guardar en Arduino el algoritmo discretizado del observador y por lo tanto el procesador debe ser capaz de recibir y de transmitir datos. Además, debe recibir las entradas del sistema v_G, D y también el valor de salida medible del sistema v_C. Con esto puede calcular el estado interno del sistema, en particular el valor estimado de i_L, el cual se debe transmitir hacia Matlab para su graficación. Una comunicación exitosa entre Arduino Uno y Simulink es esencial, para ello es necesario garantizar que los datos enviados desde Arduino Uno estén en el formato esperado por Simulink. Esto puede requerir la conversión de datos, como flotantes a bytes o viceversa. Se pueden utilizar uniones (unions) o punteros para convertir datos entre diferentes formatos y realizar el envío y recepción de manera adecuada. El cálculo de la variable de estado observada v_C se realiza después de la recepción de v_C y de v_G, D. Los parámetros del sistema (matrices, ganancias, constantes) se almacenan directamente en el Arduino.

La comunicación entre Arduino Uno y Matlab/Simulink se basa en el protocolo de comunicación serial. El protocolo serial es un estándar de comunicación ampliamente utilizado que permite la transferencia de datos de manera secuencial, bit por bit, a través de un enlace de comunicación. El protocolo serial utiliza dos líneas de comunicación principales: una línea para transmitir datos (Tx) y otra para recibir datos (Rx). En el caso de Arduino Uno, estas líneas están asociadas a los pines digitales 0 (Rx) y 1 (Tx) respectivamente, y se utilizan para la comunicación con el puerto serie (Serial). Algunos aspectos clave del protocolo de comunicación serial incluyen:

- Baud Rate: Es la tasa de baudios que define la velocidad de transmisión de datos en bits por segundo (bps). Tanto Arduino Uno como Matlab/Simulink deben estar configurados con el mismo baud rate para garantizar una comunicación adecuada.

- Formato de datos: El formato de datos especifica cómo se transmiten los bits de datos. Los formatos comunes incluyen la cantidad de bits de datos por mensaje, la paridad (opcional) para detección de errores y el número de bits de parada.
- Sincronización: La sincronización es fundamental para garantizar que los datos se transmitan y se interpreten correctamente. Se utilizan bits de inicio y bits de parada para indicar el comienzo y el final de cada mensaje. Además, se utiliza una señal de reloj para sincronizar la velocidad de transmisión entre el emisor y el receptor.

La configuración del protocolo de comunicación serial en Arduino Uno se realiza utilizando la biblioteca Serial, incorporada en el entorno de desarrollo de Arduino. Se pueden establecer la velocidad de transmisión (baud rate) y otros parámetros utilizando la función Serial.begin(). En el código de Arduino, se utiliza la función Serial.begin() para configurar la comunicación serial. Esta función se coloca en el bloque setup() y se utiliza para especificar el baud rate y otros parámetros de comunicación. Por ejemplo, para configurar la comunicación a una velocidad de 115200 baudios que es la más rápida, se utiliza “Serial.begin(115200);”. La recepción de datos enviados desde Simulink, se utiliza la función Serial.available() para verificar si hay datos disponibles en el búfer de recepción. Una vez que se confirma la disponibilidad de datos mediante Serial.available(), se puede utilizar la función Serial.read() para leer los datos del búfer de recepción. Por ejemplo, “byte data = Serial.read();”, lee un byte de datos enviado desde Simulink. Para enviar datos desde Arduino Uno a Matlab/Simulink, se utiliza la función Serial.print() o Serial.write(). Estas funciones permiten enviar datos en diferentes formatos, como enteros, flotantes o caracteres

En Matlab/Simulink, la comunicación serial se logra utilizando la biblioteca “Instrument Control Toolbox”, el bloque “Serial Configuration” para crear un objeto de comunicación serial y configurar los parámetros (velocidad de transmisión). Luego, se pueden utilizar los bloques “Serial Recieve” y “Serial Send” para simplificar aún más la comunicación entre Matlab/Simulink y Arduino Uno. Estos bloques proporcionan una interfaz gráfica intuitiva para enviar y recibir datos a través de la comunicación serial. La comunicación entre Arduino Uno y Matlab/Simulink es bidireccional, lo que significa que puede enviar y recibir datos en ambos sentidos para una interacción completa entre las plataformas. En la siguiente sección se detallará un programa en el que se muestra cómo hacer esta comunicación.

3. RESULTADOS Y ANÁLISIS DE RESULTADOS

En esta sección se explicarán los casos de aplicación de los observadores implementados en Arduino Uno y Matlab-Simulink. Más específicamente, se hace la implementación de los observadores calculados anteriormente y se comparan con los simulados en Simulink. Por lo tanto, se explicará cómo lograr la adaptación de los algoritmos de los observadores al entorno de Arduino y cómo utilizar la comunicación con Matlab/Simulink para recibir y enviar datos relevantes para poder graficar los resultados.

3.1. Discretización del modelo

La discretización es un proceso fundamental que permite representar un sistema continuo en el dominio discreto, lo cual es necesario para la implementación de observadores en Arduino Uno. En efecto, la transmisión de datos entre Simulink y el Arduino es secuencial, entonces no se puede usar el sistema no lineal continuo, sino el sistema discretizado con un tiempo de muestreo conocido. Para implementar los observadores se utiliza el método de Euler, que es un método numérico comúnmente utilizado para aproximar soluciones de ecuaciones diferenciales ordinarias. Este método se basa en la aproximación de la derivada utilizando la pendiente de la recta tangente en cada punto. El primer paso para discretizar el sistema dinámico es obtener su modelo de espacio de estados continuo. Una vez que se tiene el modelo de espacio de estados continuo, el cual se obtuvo

en la Sección 2, se puede aplicar el método de Euler para obtener las ecuaciones de estado como sigue.

Con $t = kT$ se tiene que el modelo en el espacio de estado se puede aproximar a:

$$\dot{x}(t) = \frac{dx}{dt} \approx \frac{1}{T}\left(x_{(k+1)} - x_k\right) \approx Ax_k + BU_k \tag{18}$$

$$x_{(k+1)} \approx x_k + T(Ax_k + BU_k) \tag{19}$$

Es importante tener en cuenta que la elección del tiempo de muestreo T es crucial. En efecto, este debe tener un valor adecuado para capturar las dinámicas y así obtener una buena aproximación del sistema. Se puede calcular esta variable a partir de los polos del sistema, los cuales representan la ubicación de los puntos críticos en el plano complejo. Un tiempo de muestreo adecuado garantiza una representación precisa del sistema discretizado y entonces, permite hacerlo funcionar en un Arduino. En este caso, el tiempo de muestreo se determinó a partir de la constante de tiempo del sistema, que es igual al valor absoluto del inverso de la parte real del polo complejo. Entonces, para determinar este tiempo se divide por 10 para obtener un tiempo de muestreo correcto. Con esos cálculos se obtiene: $Ts \approx 3.0\,.\,10^{-4}$ s. Se aproxima a $1.0\,.\,10^{-5}\ s$ para facilitar los cálculos numéricos y garantizar un mejor comportamiento.

3.2. Implementación de los observadores en Arduino

Con el sistema discretizado, se puede implementar el observador en Arduino Uno utilizando las ecuaciones discretizadas. Esto implica calcular las ganancias del observador y realizar los cálculos necesarios para estimar las variables de estado del sistema en cada muestra de tiempo. Para los cálculos de las ganancias, se utiliza Matlab, y posteriormente los valores obtenidos se guardan en el código del Arduino. Los métodos para estos cálculos dependen del observador diseñado, como se explicó en la Sección 2. La ecuación general de los estimadores queda de la forma siguiente:

$$\hat{x}_{(k+1)} \approx \hat{x}_k + T\left(Ax_k + BU_k + O(y(t) - C\hat{x}_k)\right) \tag{20}$$

Donde *O* representa el observador, puede ser una ganancia que multiplica la diferencia entre la salida del modelo lineal y la del observador, pero también puede ser una función como en el caso del observador por modos deslizantes. A partir de esto se tiene todo lo necesario para hacer las implementaciones de los observadores en Arduino. Uno de los códigos en Arduino, el del filtro de Kalman, se muestra en el Anexo.

3.3. Resultados de la implementación

Con el fin de probar el comportamiento dinámico de los observadores y su convergencia hacia el valor esperado, se toman las condiciones iniciales de estos diferentes a las del conversor DC-DC, el cual se supone en un estado estacionario. En este caso las condiciones iniciales de los observadores son $\mathrm{I_{Lob}} = 0.5\ \mathrm{A}$ y $\mathrm{V_{Cob}} = 4.1\ \mathrm{V}$, mientras que el conversor está con $\mathrm{I_L} = 0.4\ \mathrm{A}$ y $\mathrm{V_C} = 4.0\ \mathrm{V}$. Adicionalmente se introducen escalones en las entradas v_G y D en $0.002\ s$ y $0.04\ s$ respectivamente, con un incremento del 10% del valor del punto de equilibrio. La simulación tiene un tiempo total de $0{,}08\ s$ y usa como tiempo de muestreo $Ts = 1.0\,.\,10^{-5}\ s$, lo que está muy debajo del tiempo de muestreo mínimo para que converja el modelo discretizado.

Para obtener los resultados se realizó la implementación del observador de Luenberger y se generaron los resultados de las Figuras 6 y 7. En estas figuras y las siguientes se llama IL Real a la corriente obtenida desde el modelo de la planta en Matlab – Simulink, IL Arduino a la que se obtiene en la plataforma de Arduino con el código de implementación, e IL Simulink a la que se obtiene con el observador simulado, correspondiente a las Figuras 2, 3 y 4.

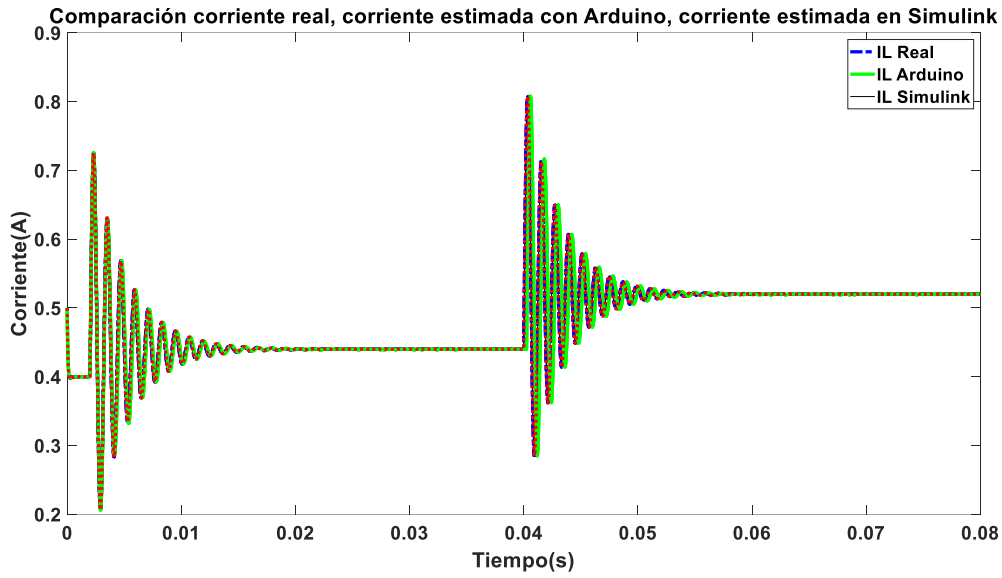

Figura 6. Resultados del observador de Luenberger.

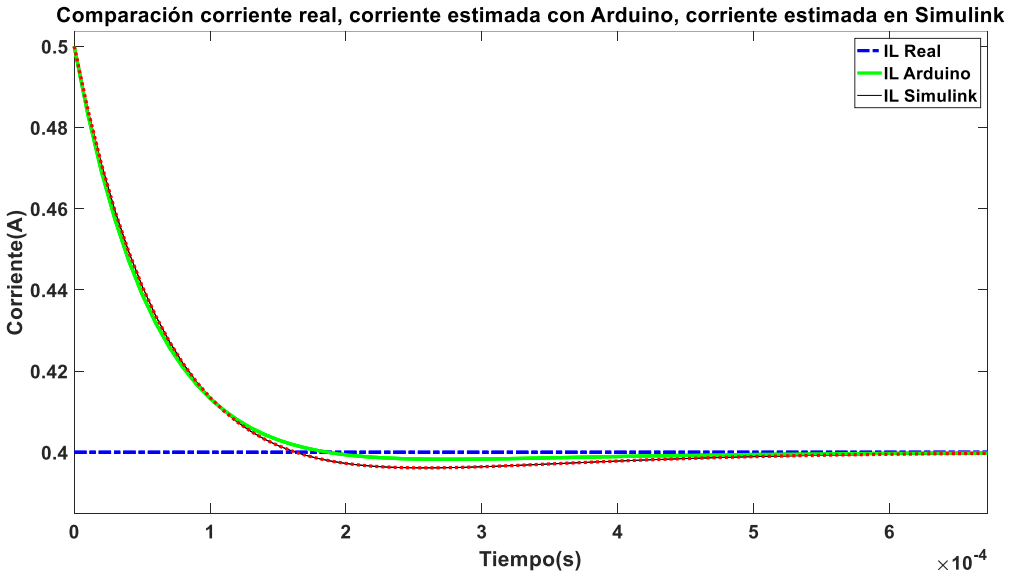

Figura 7. Resultados del observador de Luenberger (Detalle del transitorio inicial).

En la Figura 6 se puede ver que el comportamiento del observador programado en Arduino funciona correctamente, ya que sigue el mismo patrón del que se simuló en Simulink. En la Figura 7 se puede ver en detalle la convergencia del observador ante una condición inicial diferente al estado estacionario de la planta, allí se nota que se logra un tiempo de establecimiento de $0.0005\ s$, es decir más pequeño que el de la planta, como se tenía previsto en el diseño.

En segundo lugar, se implementó el filtro de Kalman, los resultados aparecen en las Figuras 8 y 9.

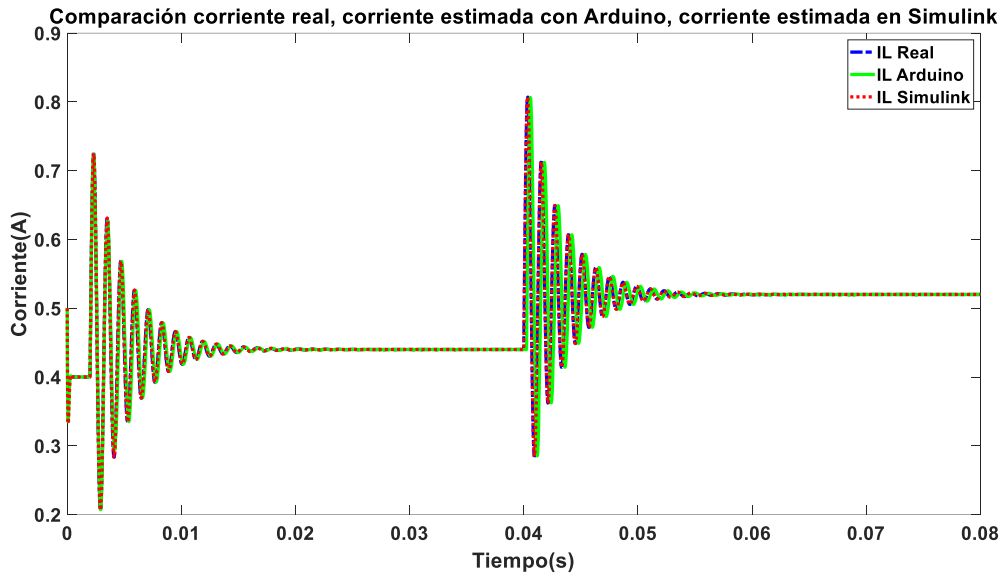

Figura 8. Resultados del filtro de Kalman.

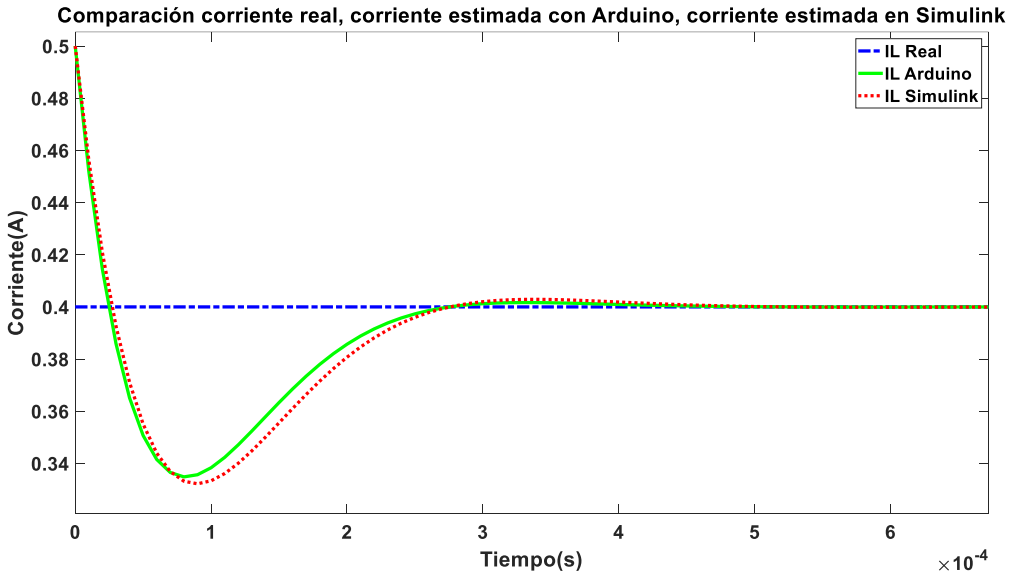

Figura 9. Resultados del filtro de Kalman (Detalle del transitorio inicial).

En los resultados se puede apreciar la convergencia del error a cero de manera asintótica, en este caso el tiempo de establecimiento es de $0.0004\,\mathrm{s}$. Se nota además el seguimiento que realiza la variable estimada a la variable de la planta durante el transitorio. Así mismo, se nota la coincidencia del observador programado en Arduino con el observador simulado en Simulink. Sin embargo, este observador presenta un sub impulso que hace que su comportamiento sea un poco peor que el anterior debido a esta desviación transitoria en la variable estimada.

Por último, se realiza la implementación del observador por modos deslizantes, los resultados se muestran en las Figuras 10 y 11.

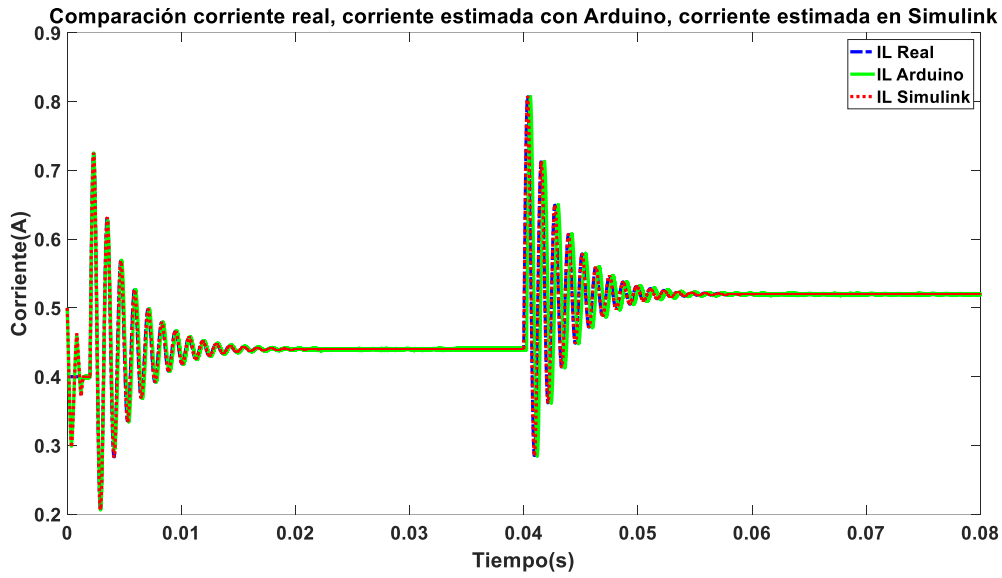

Figura 10. Resultados del observador deslizante.

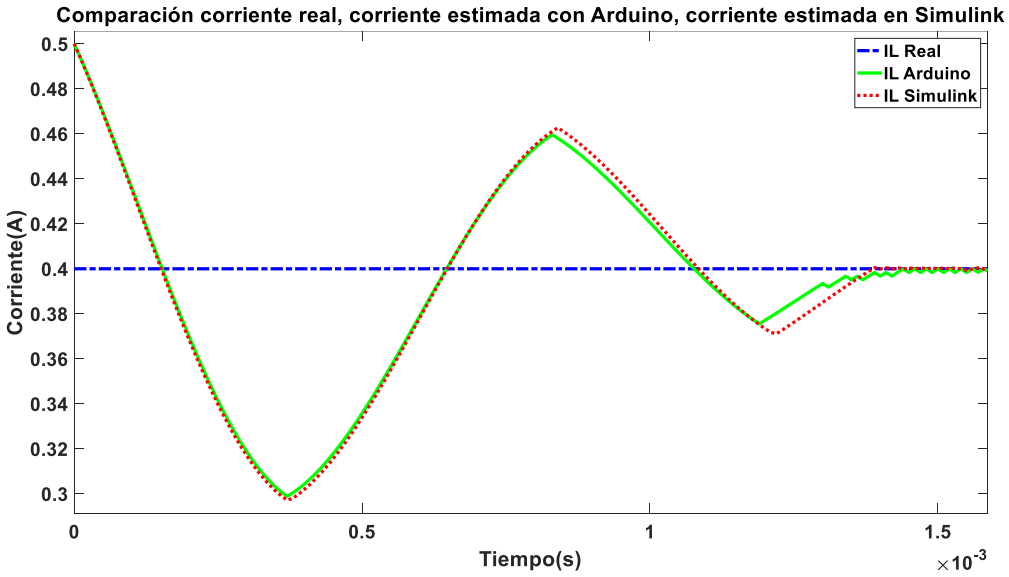

Figura 11. Resultados del observador deslizante (Detalle del transitorio inicial).

En la Figura 10 puede notarse que durante los transitorios este observador sigue la variable que se desea estimar. El comportamiento del observador programado en Arduino es similar al del simulado en Simulink. En la Figura 11 se puede ver la convergencia del observador ante la condición inicial, pero con algunas oscilaciones que corresponde al tiempo que se tarda en alcanzar la superficie deslizante, el cual está alrededor de $0.0012\,s$. Luego de este tiempo puede verse la convergencia asintótica del error hacia cero en $0.0013\,s$. La Figura 11 muestra además que el castañeo (chattering) tiene un valor de 0.0016 de amplitud. Este observador es el que peor desempeño mostró debido al alto tiempo de establecimiento y a las oscilaciones iniciales.

Se nota que en todas estas pruebas aparece un pequeño retraso entre la variable estimada (IL Arduino) y la de la planta (IL Real) debido a la sincronización entre el dato que entrega Arduino y el

graficador de Simulink. Este retraso se puede reducir mejorando dicha sincronización, por ejemplo usando las funciones de tiempo real que Matlab tiene. Sin embargo, esto implica una herramienta adicional que los estudiantes deberían manejar. Debido a que este trabajo se pensó para estudiantes de los primeros cursos de control no se profundizó en este aspecto y se deja para una mejora futura.

4. CONCLUSIONES

En este artículo se logró la implementación de tres observadores para un convertidor DC-DC boost en la plataforma Arduino, la cual es de fácil manejo, bajo costo y de uso común en el entorno académico. Para evidenciar el correcto funcionamiento de la implementación, se realizó una comunicación serial entre MATLAB/Simulink y Arduino Uno, la cual ha demostrado ser efectiva para la simulación y estimación del sistema. Se analizaron tres tipos de observadores, mostrando las diferencias cualitativas en las respuestas, sin embargo este no es el centro de este trabajo, ya que los diseños pueden ser modificados. Lo que sí se explicó aquí es una descripción detallada de los pasos para lograr la implementación de los mismos en una plataforma de Arduino, lo cual proporciona un valioso recurso educativo para estudiantes interesados en sistemas de control y electrónica. Los enfoques adoptados en el diseño, implementación y evaluación de los observadores de estado permiten a los estudiantes aprender y visualizar casos de aplicaciones prácticas en el contexto de un convertidor electrónico. Además, la metodología y la implementación presentadas en este proyecto son fácilmente reproducibles, lo que facilita su aplicación en futuras investigaciones y proyectos relacionados. Los resultados obtenidos aquí tienen el potencial de ser aplicados en diversas áreas de la ingeniería eléctrica y electrónica.

Para futuros avances en este proyecto se puede mejorar el problema del retraso acumulado en la comunicación entre Arduino y MATLAB, lo que permitirá una implementación más robusta y precisa de los observadores de estado. Asimismo, se sugiere la utilización de hardware más potente que permita realizar simulaciones en tiempo real y la implementación de los observadores en un convertidor DC-DC real.

Anexo: Código Arduino del filtro de Kalman

```
#include <Arduino.h>
union BtoF {
  byte b[4];
  float fval;
} u;
// Declaración de las variables necesarias en el formato adecuado
float R = 20;
float L = 120e-6;
float Co = 75e-6;
// Punto de equilibrio
float VG_ee = 2;
float D_ee = 0.5;
float VC_ee = 4;
float IL_ee = 0.4;
// Declaración delas matrices A y B del modelo
float A[2][2] = {
  {0, (D_ee-1)/L},
```

```
  {(1-D_ee)/Co, -1/(R*Co)}
};
float B[2][2] = {
  {1/L, VC_ee/L},
  {0, -IL_ee/Co}
};
float K[2] = {43885.67, 24680.43}; //Ganancias del filtro
float ILob = 0.1;
float VCob = 0.1;
float Ts=0.00001;
const int buffer_size = 64;
float* myVal;
float U1;
float U2;
float VC;

void setup() {
  Serial.begin(9600);
  writeToMatlab(ILob);
}
void loop() {
  if (Serial.available()>11) {
    myVal = readFromMatlab();
    U1=myVal[0];
    U2=myVal[1];
    VC=myVal[2];
    ILob = ILob + Ts*(ILob*A[0][0] + VCob*A[0][1] + U1*B[0][0]+U2*B[0][1] + K[0]*(VC - VCob));
    VCob = VCob + Ts*(ILob*A[1][0] + VCob*A[1][1] + U1*B[1][0]+U2*B[1][1] + K[1]*(VC - VCob));
    writeToMatlab(ILob);
    delete[] myVal;
    Serial.flush();
  }
}
float* readFromMatlab() {
  byte buf[buffer_size];
  int curr_buffer_size = Serial.readBytesUntil('\n', buf, buffer_size);
  float* outputList = new float[3];  // Fixed size for 3 float values

  for (int j = 0; j < 3; j++) {
   for (int i = j * 4; i < 4 + j * 4; i++) {
     u.b[i % 4] = buf[i];
   }
   outputList[j] = u.fval;
 }
```

```
 return outputList;
}
void writeToMatlab(float num) {
  byte* b = (byte*)&num;
  Serial.write(b, 4);
  Serial.write('\n');
}
```

5. REFERENCIAS BILBIOGRÁFICAS

[1] Hossain, M. Z., Rahim, N. A. and Selvaraj, J. A. (2018). Recent progress and development on power DC-DC converter topology, control, design and applications: A review. Renewable and Sustainable Energy Reviews, vol. 81, pp. 205-230.

[2] Forouzesh, M., Siwakoti, Y. P., Gorji, S. A., Blaabjerg, F. and Lehman, B. (2017). Step-Up DC-DC converters: A comprehensive review of voltage-boosting techniques, topologies, and applications. IEEE Transactions on Power Electronics, vol. 32, n° 112, pp. 9143-9178, 12.

[3] Muktiadji, R. F., Ramli, M. A. M., Bouchekara, H. R. E. H., Milyani, A. H., Rawa, M., Seedahmed, M. M. A., & Budiman, F. N. (2022). Control of Boost Converter Using Observer-Based Backstepping Sliding Mode Control for DC Microgrid. Frontiers In Energy Research, 10. https://doi.org/10.3389/fenrg.2022.828978

[4] Yan , S., Yang,Y., Hui, S. Y. and Blaabjerg, F. (2021). A Review on Direct Power Control of Pulsewidth Modulation Converters. IEEE Transactions on Power Electronics, vol. 36, n° 110, pp. 11984-12007, 10.

[5] Bernard, P., Andrieu, V. and Astolfi, D. Observer design for continuous-time dynamical systems. Annual Reviews in Control, vol. 53, pp. 224-248, 1 2022.

[6] Cimini, G., Ippoliti, G., Orlando, G., Longhi, S., & Miceli, R. (2017). A unified observer for robust sensorless control of DC–DC converters. Control Engineering Practice, 61, 21-27. https://doi.org/10.1016/j.conengprac.2017.01.012

[7] Oettmeier, F., Neely, J., Pekarek, S., DeCarlo, R., & Uthaichana, K. (2009). MPC of Switching in a Boost Converter Using a Hybrid State Model With a Sliding Mode Observer. IEEE Transactions On Industrial Electronics, 56(9), 3453-466. https://doi.org/10.1109/tie.2008.2006951

[8] G. Cimini, G. Ippoliti, G. Orlando et M. Pirro. (2013). Current sensorless solutions for PFC of boost converters with passivity-based and sliding mode control. 4th International Conference on Power Engineering, Energy and Electrical Drives, pp. 1175-1180.

[9] Tong, Q., Chen, C., Zhang, Q., & Zou, X. (2015). A Sensorless Predictive Current Controlled Boost Converter by Using an EKF with Load Variation Effect Elimination Function. Sensors, 15(5), 9986-10003. https://doi.org/10.3390/s150509986

[10] Wang, N. D., Zhao, N. Y., Li, N. B., Li, N. B., & Luo, N. J. (2016b). The current Observer design for Buck Converter. En 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT). https://doi.org/10.1109/icsict.2016.7999037, pp. 768-770.

[11] Malekzadeh, M., Khosravi, A., & Tavan, M. (2019). A novel sensorless control scheme for DC-DC boost converter with global exponential stability. The European Physical Journal Plus, 134(7). https://doi.org/10.1140/epjp/i2019-12664-4

[12] Malekzadeh, M., Khosravi, A., & Tavan, M. (2019b). An immersion and invariance based input voltage and resistive load observer for DC–DC boost converter. SN Applied Sciences, 2(1). https://doi.org/10.1007/s42452-019-1880-7

[13] Oucheriah, S., & Guo, L. (2013). PWM-Based Adaptive Sliding-Mode Control for Boost DC–DC Converters. IEEE Transactions On Industrial Electronics, 60(8),3291-3294. https://doi.org/10.1109/tie.2012.2203769.

[14] Neacsu, D. O. (2021). A Low-Cost Observer-Based Evaluation of Slow Performance Deterioration Due to Component Ageing in DC—DC Converters. IEEE Open Journal Of The Industrial Electronics Society, 2, 498-510. https://doi.org/10.1109/ojies.2021.3115242.

[15] Tan, B., Li, H., Zhao, D., Liang, Z., Ma, R., & Huangfu, Y. (2022). Finite-control-set model predictive control of interleaved DC-DC boost converter Based on Kalman observer. eTransportation, 11, 100158. https://doi.org/10.1016/j.etran.2022.100158 .

[16] Giraldo-Ramírez, S. A., Quiroz-Pedraza, D. A., Torres-Sánchez, A. S., & Botero-Castro, H. A. (2022). Diseño y simulación de estimadores de estado lineales y no lineales en un modelo de convertidor de potencia DC/DC. Revista Politécnica, 18(36), 140-161. https://doi.org/10.33571/rpolitec.v18n36a11.

[17] Ahmeid, M., Armstrong, M., Gadoue, S., Al-Greer, M., & Missailidis, P. (2017). Real-Time Parameter Estimation of DC–DC Converters Using a Self-Tuned Kalman Filter. IEEE Transactions On Power Electronics, 32(7), 5666-5674. https://doi.org/10.1109/tpel.2016.2606417

[18] Das, D., Madichetty, S., Singh, B., & Mishra, S. (2019). Luenberger Observer Based Current Estimated Boost Converter for PV Maximum Power Extraction—A Current Sensorless Approach. IEEE Journal Of Photovoltaics, 9(1), 278-286. https://doi.org/10.1109/jphotov.2018.2877418

[19] Drakunov, S., & Utkin, V. (2002). Sliding Mode Observers. tutorial. Proceedings of 34th IEEE Conference on Decision and Control, pp. 3376-3378. https://doi.org/10.1109/cdc.1995.479009.