Facultad de Ingeniería - UdelaR

Instituto de Ingeniería Eléctrica

Proyecto de fin de Carrera

Setiembre 2014/Setiembre 2015

Proyecto: Wireless Qi Charger

Nombre Corto: WiQi

Nicolás Alves

Carlos Anza

Rodrigo Espiga

Tutor: Juan Pablo Oliver

Montevideo, Uruguay

Setiembre 2015

Resumen

El presente documento tiene como objetivo recopilar la información referente al Proyecto de Grado **Wireless Qi Charger**, del Instituto de Ingeniería Eléctrica - Facultad de Ingeniería - Universidad de la República. En este proyecto se diseñó de forma íntegra (hardware y software) un sistema de transferencia inalámbrica de energía de baja potencia, para su funcionamiento como cargador inalámbrico de baterías de dispositivos móviles. A estos efectos se siguieron los lineamientos definidos en la norma *Qi*[1], que apunta a estandarizar este tipo de cargadores para lograr compatibilidad independientemente de los fabricantes.

Las dos grandes áreas en las que se puede desglosar el sistema desarrollado son hardware y software. Para la implementación hardware se seleccionó la plataforma a ser utilizada para implementar el controlador, se realizó el diseño y se fabricó el PCB. Por último, se llevó a cabo el ensamblado del sistema. El software se diseñó e implementó pensando primero en el manejo de la dinámica del sistema mediante el uso de Statecharts. Luego, se implementaron directamente en código *C* las funciones que llevan adelante las acciones del sistema, previamente definidas en la dinámica, y se adaptó la estructura software obtenida a la configuración del microcontrolador utilizado.

El resultado final de este proyecto es un sistema de transferencia inalámbrica de energía, capaz de realizar la carga de baterías de dispositivos móviles compatibles con el estándar *Qi*. Cada una de las etapas de dicho sistema se diseñó para satisfacer los requerimientos del estándar. En tal sentido, se logró alcanzar un sistema que selecciona y activa la bobina con mejor acople para la transmisión en caso de que haya un receptor sobre la superficie, pero que se mantiene en estado de stand-by en caso contrario. A su vez, el sistema desarrollado es capaz de recibir, demodular y decodificar la comunicación proveniente de un receptor compatible, y tomar decisiones en base a los mensajes recibidos, logrando también, como consecuencia, un sistema eficiente en términos energéticos. Otra de sus características es la capacidad de modificar el punto de funcionamiento durante el proceso de carga según lo solicite el receptor, de forma de no acortar la vida útil de la batería. Sin embargo, cabe acotar con respecto a este último punto que, dadas las características del sistema implementado, no se logró alcanzar la resolución requerida por la norma para la variación del ciclo de trabajo de la señal de potencia a los efectos de controlar la potencia transferida.

Índice

Agradecimientos

Se agradece por su colaboración a través de sus aportes de conocimientos técnicos al Ing. Fernando Chiaramello y al Dr. Ing. Leonardo Steinfeld. Por su colaboración práctica en instancias de ensamblado hardware, también se agradece al Mg. Ing. Sebastián Fernández. De igual forma se agradece al Dr. Ing. Juan Pablo Oliver, por sus aportes técnicos y orientación a lo largo de este proyecto.

Se agradece, también, por su colaboración a lo largo del desarrollo del proyecto a Eveline Karolyi, y por su invalorable colaboración con la presente documentación a Carla Rapetti.

Capítulo 1

Introducción

¿Qué nivel de carga tiene la batería de su teléfono móvil en este momento?
¿Cuándo fue la última vez que necesitó conectar alguno de sus dispositivos móviles a una fuente de energía?

En ocasiones, la respuesta a estas preguntas es de suma importancia, ya que el avance de la tecnología ha generado que los seres humanos concibamos muchos dispositivos móviles como parte esencial de nuestra vida cotidiana. La gran cantidad de aplicaciones y servicios que estos aparatos brindan los ha convertido en objetos imprescindibles y de constante necesidad. En efecto, no disponer de ellos por extravío, olvido o falta de carga en la batería puede generar situaciones de preocupación, desorientación o falta de comunicación. Esta infinidad de aplicaciones y usos que ofrecen los dispositivos móviles, y en particular los celulares, también los transforma en dispositivos de alto consumo de energía, razón por la cual se presenta un *trade-off* entre su portabilidad (tamaño y peso) y su autonomía. De cualquier forma, y sin que el aumento de su tamaño parezca incidir, la cantidad de dispositivos móviles se ha incrementado en forma tal que en la actualidad todos tenemos, al menos, un dispositivo de este tipo todo el tiempo con nosotros.

Por otra parte, el mencionado aumento de tamaño de estos dispositivos (sin desmedro de la portabilidad) parece estar alcanzando su límite, por lo que la posibilidad de continuar aumentando el tamaño de la correspondiente batería también ha alcanzado su tope. Con esto, la necesidad de recarga de la batería de los teléfonos móviles se ha vuelto constante, tornando imprescindible llevar también el cargador en todo momento. Además de esta necesidad, también se generan algunos problemas a raíz del uso intensivo del cargador: rápido deterioro del puerto conector del cargador (agravando el problema el hecho de ser el mismo puerto que el de datos); incomodidad al momento de tener el dispositivo conectado por largos períodos de tiempo; poca practicidad en los casos de dispositivos con cierre hermético (cada vez más común para lograr teléfonos resistentes al agua y polvo), etc. Entonces, respondiendo a esta masificación de los dispositivos móviles y a los problemas aquí mencionados, la industria tecnológica ha centrado esfuerzos en el desarrollo de dispositivos para la transferencia inalámbrica de energía orientada a la carga de baterías. Esta tendencia apunta a la inundación de dispositivos de carga en espacios públicos de la mano de una universalización de estos sistemas, con el objetivo de generar un fácil acceso a la carga de móviles para los usuarios en todo momento y en todo lugar.

Aplicación del Proyecto

El proyecto aquí descrito tuvo por objetivo el diseño e implementación, tanto hardware como software, de un sistema de transferencia inalámbrica de energía de baja potencia para su funcionamiento como cargador inalámbrico de baterías de dispositivos móviles compatibles. Este tipo de sistema de transferencia de potencia basa su funcionamiento en el principio de inducción electromagnética, presentando un funcionamiento similar al de un transformador con núcleo de aire y haciendo efectiva la carga solamente a pequeñas distancias, logradas mediante el apoyo del receptor sobre una superficie de carga. Una idea general de lo que este concepto implica puede apreciarse en la Fig. 1.1 (Imagen tomada de la publicación *Microsoft, Samsung 'take a leading role' backing Qi wireless charging* por Mihaita Bamburic, para el sitio web http://betanews.com/; año 2013), donde la base emisora de potencia (sistema diseñado en este proyecto) es, en definitiva, una superficie emisora de potencia (primario del transformador), que recibe alimentación cableada y lleva a cabo los procesos eléctricos necesarios hasta lograr la emisión de energía electromagnética. El sistema se completa y, por ende la transferencia se lleva a cabo, cuando se cuenta con un dispositivo receptor (secundario del transformador) apoyado sobre la superficie.

Figura 1.1: Sistema de carga inalámbrica.

1.1. Fundamento teórico

En esta sección se presentan, en primera instancia y de forma general, distintos tipos de transferencia inalámbrica de energía y conceptos físicos en los cuales se basa la tecnología utilizada en sistemas de este tipo. Luego se hace particular énfasis en las bases de la transferencia por Inducción Electromagnética, ya que es este el tipo de transferencia de interés a los efectos de este proyecto. Por último, se realiza una breve descripción del

comportamiento en resonancia de los circuitos oscilatorios forzados, dada su utilidad para la detección de dispositivos sobre la superficie de carga.

1.1.1. Transferencia inalámbrica de energía

La expresión *transferencia inalámbrica de energía* refiere a un grupo de tecnologías para la transferencia de energía mediante el uso de campos electromagnéticos variables. En general, cualquier transferencia de este tipo consiste en un emisor (conectado a una fuente de energía eléctrica) que transforma la energía en campo electromagnético, y uno o más receptores que reciben este campo transformándolo nuevamente en energía eléctrica para alimentar una carga. En estos sistemas, la transformación de energía eléctrica en campos electromagnéticos se lleva a cabo por algún tipo de antena, la cual es alimentada por una señal eléctrica acorde a dicha antena y al campo electromagnético a utilizar como portador de energía. En la tabla 1.1 se presenta una lista de distintas tecnologías de transferencia inalámbrica de energía y algunas de sus características.

Tecnología	**Alcance**	**Frecuencia**	**Antena**
Acople Inductivo	Medio	Hz - GHz	Bobinas sintonizadas, resonadores
Acople Capacitivo	Corto	kHz - MHz	Electrodos
Magnetodinámica	Corto	Hz	Magnetos rotatorios
Microondas	Largo	GHz	Parabólicas
Ondas lumínicas	Largo	THz	Lasers, lentes, fotocélulas

Tabla 1.1: Tecnologías de transferencia inalámbrica de energía.

Entre estas características, una de las más importantes es la frecuencia de oscilación de las ondas electromagnéticas utilizadas, ya que esta es inversamente proporcional a la longitud de onda ($\lambda = c/f$), y es este parámetro el que determina qué tipo de transferencia puede llevarse a cabo dependiendo de la distancia emisor-receptor.

Las leyes del electromagnetismo determinan que cargas estáticas y corrientes continuas generan campos eléctricos y magnéticos estáticos, respectivamente. Estos campos son portadores de energía, pero no son portadores de potencia por ser, justamente, estáticos. Sin embargo, los campos electromagnéticos variables, generados a partir de cargas o corrientes variables, sí son portadores de potencia y pueden ser discriminados en dos regiones, dependiendo de la distancia hasta la antena del sistema de transferencia inalámbrica que los genera. Esto se da ya que estos campos tienen distintas características en estas regiones, y son distintas las tecnologías que pueden ser usadas para la transferencia de potencia:

Campo cercano o región no irradiante: Esta región es la circundante a la antena, hasta una longitud de onda de distancia. Aquí, la potencia puede ser transferida con campos eléctricos, vía acople capacitivo (inducción electrostática) entre electrodos metálicos, o con campos magnéticos, vía acople inductivo (inducción electromagnética) entre bobinados de cable. Estos son campos no irradiantes, lo que significa que la energía se mantiene en el espacio cercano a la antena y que su comportamiento es

tal que, en caso de no haber un receptor o material absorbente dentro del espacio de acoplamiento, no hay disipación de la potencia emitida por la antena en esta zona. Estos campos son de corto alcance, ya que presentan la particularidad de que la potencia transferida decae rápidamente con la distancia, por lo que si la distancia entre antenas emisor-receptor es mucho mayor que el diámetro de dichas antenas, poca potencia será transmitida.

Campo lejano o región irradiante: Esta es la región alejada más de una longitud de onda de distancia a la antena. En este caso los campos eléctrico y magnético son perpendiculares y se propagan como una onda electromagnética (las ondas de radio, microondas y ondas de luz se encuentran dentro de este grupo). Estas son ondas irrandiantes, por lo que hay emisión de potencia desde el transmisor independientemente de la presencia de receptores, y la energía que no es absorbida por un receptor se pierde. Este tipo de ondas puede ser utilizado para transferencias de largo alcance ya que, si bien la emisión es omnidireccional, mediante el uso de sistemas ópticos o antenas de alta ganancia, la radiación puede ser concentrada en un haz acotado y direccionado.

En el caso de la transferencia de campo cercano (NF, por sus siglas en inglés), la resonancia representa una característica diferencial respecto de la distancia dentro de la cual la recepción de potencia resulta efectiva, ya que esta incrementa considerablemente la eficiencia de la transferencia. Entonces, la transferencia NF pude dividirse en dos tipos:

- *Corto alcance:* Donde la transferencia de energía se hace efectiva a distancia entre antenas no mayor al diámetro de estas, mediante inducción electromagnética o capacitiva de campos no resonantes.
- *Alcance medio:* Donde la transferencia de energía se hace efectiva a distancia entre antenas hasta 10 veces mayor al diámetro de estas, mediante inducción electromagnética o capacitiva de campos resonantes.

Presenta particular importancia a los efectos de este documento la transferencia inalámbrica de energía por inducción electromagnética, de campo cercano y corto alcance, dado que es esta la tecnología utilizada por el sistema diseñado.

Los conceptos vertidos en esta sección (1.1.1) fueron tomados de la web Wikipedia, https : //en.wikipedia.org/wiki/Wireless_power

1.1.2. Inducción Electromagnética

En la transferencia inalámbrica de energía por inducción electromagnética, la potencia es transferida entre bobinas de conductores y mediante un campo magnético como portador de energía. Para esto, la bobina del transmisor es alimentada con una corriente alterna creando así un campo magnético variable, según la ley de Ampere. Una porción de este campo magnético (dada por el acople entre bobinas) pasará a través de la bobina del receptor, induciendo en esta una tensión (FEM), según la ley de Faraday, que creará una corriente alterna en el conductor del bobinado en caso de contar con un circuito cerrado. De esta forma, se da la transferencia inalámbrica de potencia entre un emisor y uno, o

varios, receptores, pudiendo la potencia en el secundario alimentar una carga.

En efecto, los bobinados del transmisor y el receptor, juntos, configuran un transformador con núcleo de aire, con la mutua determinada por el acople magnético existente entre las bobinas, donde este acople depende de la geometría del sistema de transformación, o sea, de las características físicas de los bobinados (forma, tamaño, relación de tamaño entre ellos y del número de vueltas de las bobinas) y la distancia y ángulo de orientación entre ellos. Un diagrama representativo de estos conceptos se presenta en la Fig. 1.2 (Imagen tomada de la web del Wireless Power Consortium)[1], donde D y D_2 son los diámetros de las bobinas L_1 y L_2 respectivamente, z la distancia entre estas y B el campo magnético. En definitiva, el acople mencionado no es más que el coeficiente de acoplamiento k utilizado en el estudio de transformadores, calculado como $k = M/\sqrt{L_p L_s}$ (con L_i la autoinductancia del bobinado), que conceptualmente representa la porción del flujo magnético generado por el primario (bobinado del transmisor en este caso) que pasa por el secundario (bobinado del receptor en este caso). Cuando los bobinados se encuentran en el mismo eje, tan cerca como sea posible uno de otro y si sus radios son iguales, teóricamente será el 100 % del flujo el que pasará por el bobinado del receptor, alcanzando el máximo acople posible ($k = 1$).

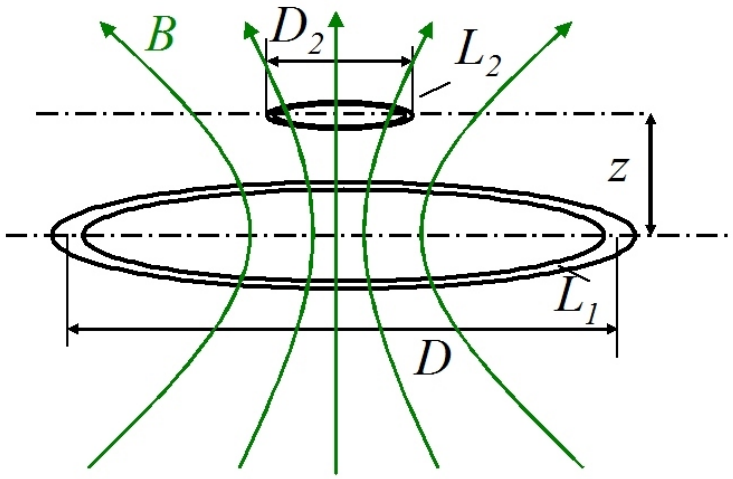


Figura 1.2: Esquema de acople entre bobinas.

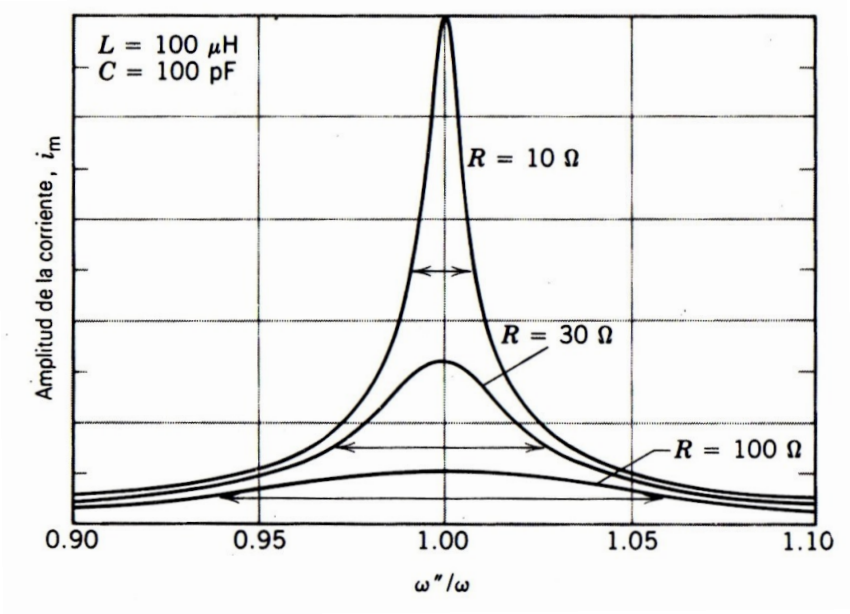
Basado en los principios físicos aquí presentados es que se establecen las directivas detalladas en la norma *Qi*, para implementar un sistema de transferencia de potencia a través de una superficie bien delimitada donde depositar el receptor, buscando un acople suficiente para alcanzar una eficiencia en la transferencia inalámbrica por encima del 86 %[2].

[1] http://www.wirelesspowerconsortium.com/data/images/1/3/3/figure1.jpg.

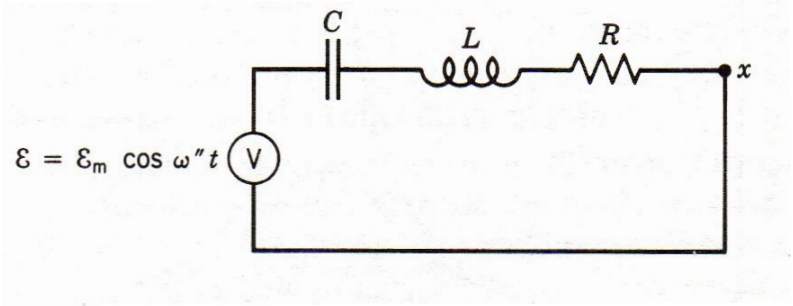
[2] Fuente: http://www.wirelesspowerconsortium.com/data/downloadables/1/2/1/1/why-not-a-wire-the-case-for-wireless-power.pdf

1.1.3. Resonancia de circuito oscilatorio forzado

Como se mencionó en la introducción a esta sección, el comportamiento de los circuitos oscilatorios forzados es el fenómeno físico en el cual se basa la detección de posibles receptores sobre la superficie de carga. Para la comprensión de este proceso resulta fundamental observar el comportamiento de estos sistemas a frecuencias cercanas a la frecuencia de resonancia, y el comportamiento de sistemas de características similares con distinta impedancia. A estos efectos, en la Fig. 1.3(a) se presenta una descripción gráfica de sistemas de este tipo y su respuesta en corriente frente a cambios en la frecuencia y la carga para un circuito resonante como el mostrado en la Fig. 1.3(b) (Imágenes extraídas del libro Física Vol.2, Versión Extendida, David Halliday, Robert Resnik y Kenneth Krane, 4ta edición, año 1992).



(a) Respuesta en corriente frente a cambios en frecuencia



(b) Circuito RLC

Figura 1.3: Comportamiento de circuito resonante RLC.

Con base en este principio se diseña un sistema de detección de elementos ferromagnéticos sobre la superficie de carga, dado el cambio en la impedancia del sistema que representa la presencia de un posible receptor. Para esto el sistema realizado releva periódicamente

la superficie de carga durante el estado stand-by en busca de posibles receptores, energizando las bobinas por un cortísimo período de tiempo con una señal a frecuencia de resonancia. En este período de tiempo sensa la corriente consumida por la etapa de potencia identificando la presencia o no de un receptor a partir del valor de la amplitud de la corriente o, lo que es igual, la impedancia asociada (altura de la campana).

1.1.4. Modulación de Retrodispersión

El sistema desarrollado recibe mensajes desde el receptor a través de una señal de comunicación modulada sobre la onda de potencia, siendo la modulación backscatter el tipo de modulación utilizada . Esta modulación puede tener dos variantes: modulación en amplitud ASK (Amplitude Shift Keying), o modulación en fase PSK (Phase Shift Keying). La modulación elegida para este tipo de sistemas es ASK, y es realizada por el controlador del receptor de potencia modificando la cantidad de potencia que absorbe del emisor mediante conexión y desconexión de una carga extra, de forma de generar las variaciones deseadas en la amplitud de la señal del primario. La señal resultante (vista por el emisor) luego de una modulación de este tipo se muestra en la Fig. 1.4, extraída de [2].

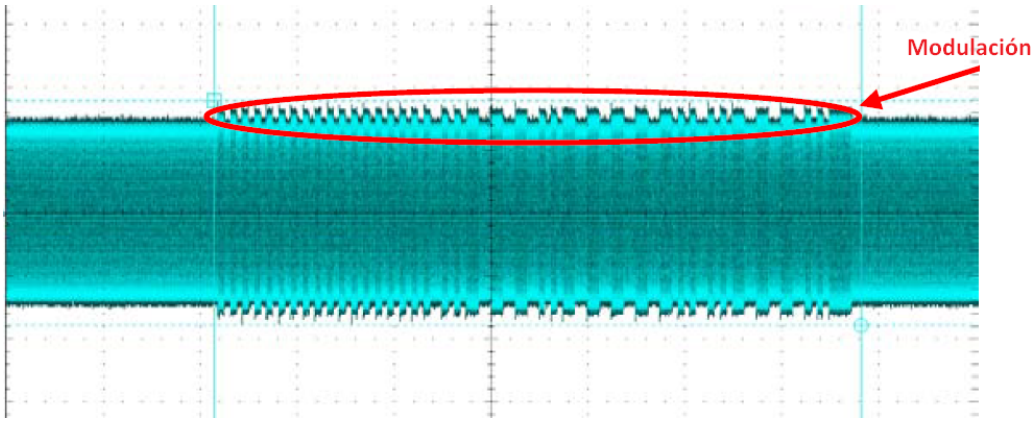


Figura 1.4: Señal modulada con técnica de retrodispersión ASK.

1.2. Marco normativo

La realización del proyecto se centró en el diseño e implementación del cargador inalámbrico según el estándar *Qi*. Dicho estándar detalla aspectos de diseño y constructivos que deben cumplir los emisores y receptores de energía para cumplir con el protocolo. Además, dentro de los aspectos de diseño se explicitan tanto las características del sistema de control del emisor como de la comunicación que debe existir entre emisor y receptor para llevar a cabo la realimentación necesaria para dicho control. Entre los detalles constructivos se halla la especificación de la forma y tamaño de las bobinas requeridas para generar el flujo magnético mediante el cual se transmitirá energía y se modulará la señal de comunicación, así como también se detalla su disposición para los casos en los que el

diseño posea más de una bobina. Dentro de las distintas opciones de cargadores definidos en la norma se diferencian dos grandes grupos:

- El grupo de los cargadores *tipo A*, que consta de los cargadores que energizan una sola bobina para realizar la carga y, por ende, cargan un solo dispositivo a la vez.
- El grupo de los cargadores *tipo B*, que agrupa a los cargadores que poseen grandes arreglos de bobinas y pueden cargar varios dispositivos en simultáneo sin restricciones de posicionamiento, pero sumando al control la tarea de evitar interferencia destructiva entre bobinas activas.

Dado el grado de dificultad que representaba la implementación de un cargador *tipo B*, se optó por centrarse en el diseño de un cargador *tipo A*, eligiéndose dentro de este grupo el *tipo A6*, variante que utiliza un arreglo de tres bobinas para ampliar el área de la superficie de carga. Además del hecho de que implementar un cargador *tipo A* es más viable que implementar uno *tipo B*, por la complejidad de este último, fue relevante para esta decisión la existencia en el mercado de cargadores *tipo A* y de todos los componentes necesarios para su ensamblado (cosa que no ocurre con los *tipo B*). En particular, se adquirieron bobinados con las características definidas en la norma para un *tipo A6*, evitando la necesidad de diseñar y construir uno, y un par cargador[3]-receptor para el relevamiento de las señales de comunicación y características de la transferencia. Sin embargo, surgió del estudio de la norma la idea de una implementación híbrida que permita la realización de una superficie de carga *tipo B* a través de una implementación modular de un número dado de cargadores *tipo A* y un controlador global.

Previamente a la profundización en el sistema desarrollado, cabe acotar que a lo largo de esta documentación se seguirán las formas presentadas en el estándar, refiriéndose siempre al ***Receptor*** como el dispositivo móvil receptor de potencia, y al ***Trasmisor*** como el trasmisor de potencia, sistema desarrollado en este proyecto.

Tomando esto en cuenta, así como el nivel de detalle descriptivo existente en la norma, es que se consideró contar con una base sólida y buenos lineamientos para la realización de este proyecto.

1.3. Descripción del problema

Del Marco Normativo (1.2) surge una descripción del problema que permite su división en 3 aspectos principales:

Implementación de la Etapa de Potencia

Para lograr la transferencia inalámbrica de energía se debió **diseñar e implementar la etapa de potencia** que, controlada mediante las señales apropiadas (PWM, ENA, V_{DD}), proveyera la señal necesaria para los bobinados. Este circuito, siguiendo lo que se especifica en la norma, debió contar con un inversor *Half-Bridge* y su respectivo controlador, para proveer de la señal necesaria al ramal de la bobina de transferencia y su capacitor en serie.

[3]Tipo A11, de una sola bobina

Programación del Sistema de Control

A los efectos de obtener un sistema con eficiencia energética, seguro y que no acorte la vida útil de la batería a cargar, la norma especifica el sistema de control con el que deben contar los cargadores compatibles. Por ende, otro problema enfrentado fue **la programación del sistema de control en la plataforma a utilizar** a tales efectos.

Procesamiento de señales de comunicación

El sistema de control establecido durante un proceso de carga basa su funcionamiento en la realimentación desde el receptor de energía hacia el cargador a través del envío de mensajes, y esta se lleva a cabo mediante la modulación en amplitud de la onda de potencia. Por lo tanto se debió **procesar la señal de potencia sobre la cual el receptor de potencia modula la comunicación (a través de load-shift keying) para obtener los paquetes de datos, y programar el controlador del sistema para decodificar la información recibida**, con el fin de contar con la realimentación necesaria para llevar adelante el control en la transferencia de energía.

Se habla de un sistema seguro y con eficiencia energética ya que el estándar especifica los comportamientos que debe tener un cargador *Qi* en algunas situaciones particulares, para alcanzar estos objetivos. En resumen, un cargador de este tipo debe sensar la superficie de carga periódicamente en busca de un receptor válido. En caso de detectar la presencia de un posible receptor, intentará establecer una comunicación, y dos posibles casos se pueden dar: el material en cuestión es un material férrico, inductivo, capaz de absorber potencia (y disiparla en forma de calor) pero no es un receptor *Qi* y no establecerá comunicación; el material es un receptor *Qi* válido y comenzará a enviar mensajes de reconocimiento y configuración. En el primer caso, el sistema volverá al estado de reposo y no se emitirá potencia. En el segundo caso, se establecerá una onda de potencia duradera y se comenzará el proceso de carga. Además, a lo largo de un proceso de carga, el cargador recibirá mensajes de parte del receptor, que le informará la potencia que está recibiendo, para que el sistema evalúe si la diferencia entre lo enviado y lo recibido es muy grande. En caso de que efectivamente la diferencia sea demasiada, el emisor supondrá que hay algún otro dispositivo inductivo sobre la superficie y finalizará la emisión de potencia. Además la norma asegura que los sistemas *Qi* compatibles tienen una eficiencia mínima en la transferencia inalámbrica del 86 % (por debajo de cierto nivel de señal no se iniciará el proceso de carga), dotando de una buena eficiencia energética a sistemas de este tipo.

Para enfrentar los problemas anteriormente detallados fue necesaria la resolución de problemas intrínsecos a la implementación de un sistema integral, implicando esto el **diseño y desarrollo tanto del software como del hardware** necesario para alcanzar los objetivos.

1.4. Definición del proyecto

El proyecto *Wireless Qi Charger* podría definirse, entonces, como:

Un proyecto que apuntó al diseño e implementación de un cargador inalámbrico cumpliendo con los lineamientos detallados en el estándar Qi del Wireless Power Consortium, y capaz de obtener la certificación Qi.

1.5. Antecedentes

Como se deduce tanto del Marco Normativo (1.2) como de la Definición y Objetivos (1.4) del proyecto, su principal antecedente es el Estándar *Qi* Versión 1.1.2 definido por la Wireless Power Consortium. En este escenario, cabe mencionar que a lo largo de todo este documento se hará referencia tanto a *la norma*, como al *estándar* para hablar de estándar *Qi* aquí citado.

Otro antecedente es el proyecto final del curso *Sistemas embebidos para tiempo real*, realizado por dos de los integrantes del grupo, que tuvo como objetivo el diseño de la lógica básica de estados del controlador de un cargador Qi. Para ese proyecto se logró una estructura de estados para el controlador que, respondiendo a señales (pulsadores o mensajes I^2C) de un emulador de receptor, cumplía con las transiciones de estados detalladas en el estándar. Esta estructura general de controlador fue realizada en lenguaje C, para un μC (Microcontrolador) MSP430 de Texas Instruments, y desarrollada con Statecharts[4] mediante las herramientas VisualState y Embedded Workbench de IAR Systems. Por esta razón, y dada la utilización de un μC STM8[5] de STMicroelectronics para nuestro diseño, la experiencia adquirida en esa instancia con la plataforma VisualState resultó de gran utilidad.

1.6. Alcance del Proyecto

El controlador diseñado debía cumplir en su totalidad con las especificaciones del Estándar *Qi* en lo que a selección de bobinas por acoplamiento, control y comunicación durante el proceso de carga respecta. En base a esto se implementó un cargador inalámbrico capaz de interactuar satisfactoriamente con un receptor *Qi* adquirido en el mercado.

Como opcional para este proyecto, y sujeto a la disponibilidad de tiempo, se planteó incluir en el diseño la funcionalidad necesaria para que el cargador pueda ser utilizado como un módulo en un sistema que implemente una superficie con comportamiento *tipo B* (a desarrollar), y alcanzar la posibilidad de transmisiones mayores a 5W[6]. Sin embargo, debido a una serie de contratiempos experimentados (detallados en el Anexo D, 9), no

[4]Herramienta de diseño para sistemas embebidos reactivos (4.1)
[5]Microcontrolador que cuenta con plataforma de desarrollo específica provista por IAR
[6]Límite máximo de potencia establecido por la norma para cada bobina.

fue posible trabajar en esta funcionalidad propuesta en un primer momento.

1.7. Objetivos

Para alcanzar el objetivo planteado en este proyecto se hacían necesarias la implementaciones hardware y software necesarias para lograr un sistema con la totalidad de las funcionalidades especificadas en el estándar, detalladas a continuación:

Modo stand-by: Recorrido de bobinas en búsqueda de posible receptor cuando la superficie de carga se encuentre libre.

Ping Analógico: Detección de posible receptor sobre la superficie de carga.

Comunicación: Demodulación de la señal de comunicación, decodificación de datos, validación de paquetes e interpretación de mensajes recibidos.

Selección de bobinas: Establecimiento de comunicación con un receptor válido para la selección de la bobina mejor acoplada.

Realimentación: Interpretación de mensajes recibidos y toma de las acciones correspondientes definidas por el estándar.

Foreign Object Detection: Comparación entre la potencia que el receptor informa estar recibiendo, y la potencia que efectivamente se está enviando.

Control de Potencia: Variación controlada o finalización de la transferencia de potencia en respuesta a pedidos del receptor.

Temporización: Cumplimiento con las restricciones de tiempo definidas.

Sin embargo, al término de este proyecto sólo se alcanzaron parcialmente las funcionalidades requeridas por el estándar Qi, logrando la implementación de un hardware que, a priori, presenta la capacidad de llevar a cabo casi completamente las acciones requeridas (es incapaz de alcanzar algunos niveles de resolución exigidos), y una implementación software con funcionamiento muy restringido. En resumen, la implementación final es capaz de:

Modo stand-by: Se alcanzó completamente la implementación de esta funcionalidad.

Ping Analógico: Se alcanzó completamente la implementación de esta funcionalidad.

Comunicación: Se alcanzó completamente la implementación de esta funcionalidad, logrando recibir y verificar exitosamente los paquetes de datos.

Selección de bobinas: No se logró implementar esta funcionalidad como solicita la norma, seleccionando la bobina mejor acoplada para transferir potencia, eligiendo para la transferencia la primer bobina por la que se recibe la señal de comunicación.

Realimentación: Sólo se logró implementar funciones correspondientes a los paquetes de ajuste del punto de funcionamiento en la transferencia de potencia y de finalización de carga. No se logró la confección o renegociación del contrato de transferencia ni la elección de bobina por mayor acople.

Foreign Object Detection: No se logró poder garantizar la eficiencia en la transferencia de potencia a través de esta funcionalidad.

Control de Potencia: Se alcanzó un control de potencia parcial, logrando la variación de la frecuencia de la onda de potencia en respuesta a un pedido de receptor, pero no mediante la utilización de un PID (PI, para el caso de este tipo de cargadores) como indica el estándar.

Temporización: Sólo se implementaron temporizadores para la decodificación de comunicaciones, no pudiendo lograr el control de tiempo para las transiciones y finalización de transferencia de potencia en caso de error impuestos por la norma.

1.8. Descripción general del Sistema

Buscando cumplir con los objetivos se implementó un sistema como el mostrado en la Fig. 1.5, compuesto por 5 bloques bien definidos: un bloque StepDown DC-DC (3.4.2) conversor de tensión para alimentar los componentes que requieren tensión de alimentación *5VDC* (la alimentación del sistema es *12VDC*); un bloque Sensor de Corriente (3.4.1) para determinar el estado del sistema en momentos dados; una Etapa de Potencia (3.1) encargada de transformar la alimentación DC del sistema en el campo magnético mediante el cual se transfiere la potencia; una Etapa de Comunicación (3.3) encargada de demodular las señal de comunicación; y un Controlador (3.2) central, que a través de la programación del Software (4.2) dota al sistema de la inteligencia necesaria para el funcionamiento buscado.

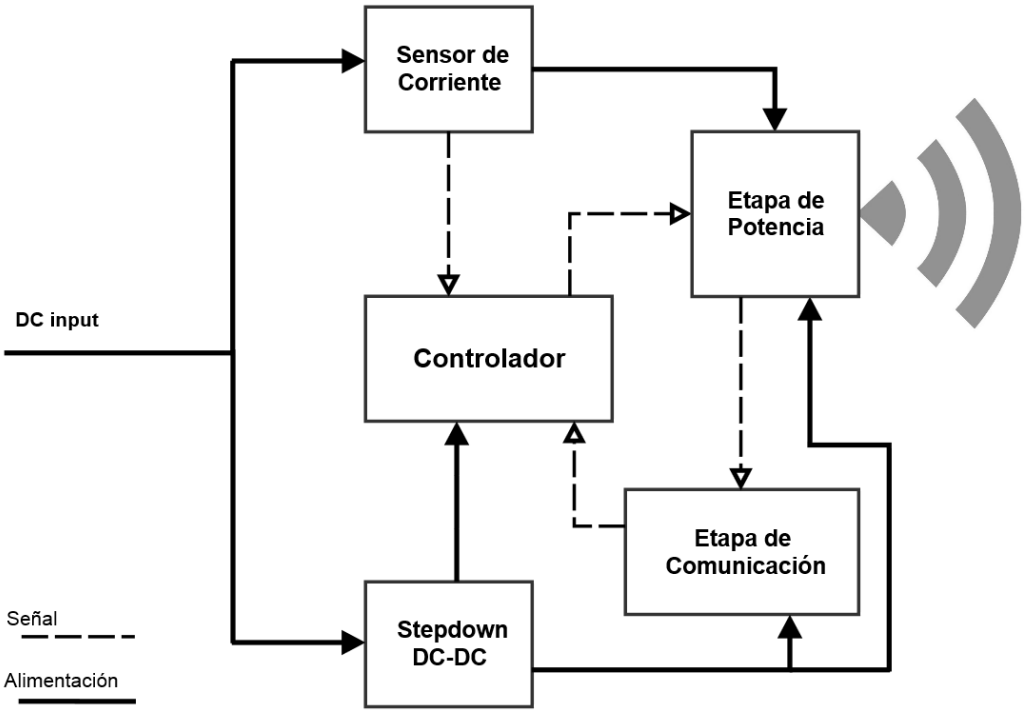


Figura 1.5: Diagrama de bloques del sistema.

1.9. Desarrollo actual de la tecnología y mercado

Respondiendo a la fuerte presencia de dispositivos móviles de gran capacidad de procesamiento, alto consumo de energía y por ende bajísima autonomía, la industria tecnológica ha centrado esfuerzos en el desarrollo de dispositivos para la transferencia inalámbrica de energía. Esta tendencia apunta a la inundación de dispositivos de carga en espacios públicos de la mano de una universalización de estos sistemas, generando así un fácil acceso a la carga de móviles por parte de los usuarios en todo momento y en todo lugar. A raíz de esto es que han surgido Alianzas y Consorcios que incluyen a los más grandes fabricantes de tecnología a nivel mundial, siendo los más importantes en cuanto a desarrollo de tecnologías para carga de dispositivos portables la WPC (Wireless Power Consortium) y la PMA (Power Matters Alliance). Actualmente ambas alianzas nuclean alrededor de 200 empresas multinacionales del área de la tecnología, encontrándose entre éstas grandes productoras de dispositivos móviles o electrónica de consumo y las más relevantes compañías productoras de integrados semiconductores.

Basándose en esta realidad a la hora de la toma de decisiones, luego de un relevamiento de la oferta y el desarrollo en el mercado que presentan estas dos opciones, al notar que ambos conjuntos de empresas tienden a nuclear a los mismos fabricantes (quienes ya ofrecen productos compatibles con ambos protocolos a la vez) y la facilidad de acceso a las especificaciones de la WPC en comparación con el acceso a la especificación de la PMA, se optó por basar el presente trabajo en el estándar *Qi*, establecido por la WPC.

Una vez trazados los lineamientos, se siguió por establecer qué tipo de emisor sería el foco del proyecto, ya que el estándar define un total de 23 tipos de emisores de potencia. En definitiva, se optó por el tipo *A6* buscando un equilibrio entre la complejidad que requerirá el diseño, las plataformas posibles y el tiempo disponible hasta la finalización del proyecto. Por último, para esta etapa previa se realizó una exhaustiva búsqueda de la oferta tanto de dispositivos como de componentes específicos para implementación de cargadores *A6*. Como resultado de esta búsqueda se encontró que sólo existen dos fabricantes que ofrecen cargadores tipo *A6* implementados, mientras son cuatro los fabricantes de IC[7] que ofrecen chips controladores tipo *A6*. En resumen:

Cargadores

- ITian - Origen: India
- TYLT - Origen: USA/China

Controladores

- Texas Instruments - IC con periféricos e implementación sugeridos.
- NXP - IC con periféricos e implementación sugeridos.
- IDT - IC con periféricos e implementación sugeridos.
- Freescale - Microcontrolador con configuración programable.

[7] Circuitos Integrados

Cabe acotar, además, que 5 de los 23 emisores definidos en el estándar *Qi* son los denominados *Type B*, con la característica diferencial de contar con una superficie de carga *true free-positioning* que admitiría más de un dispositivo receptor a la vez[8]. Este tipo de emisores permitiría también la transmisión de potencias mayores a 5W, y es ahí donde reside su mayor potencial. Sin embargo, y si bien ese tipo de emisores era la idea inicial del presente proyecto, la inexistencia de antecedentes en lo que a diseño y producción respecta hacía de esa idea un objetivo demasiado ambicioso para el marco de este proyecto. Dadas estas circunstancias y en vistas del paso previo que se considera que un emisor *A6* representa, se establecieron los Objetivos (1.4) que este proyecto presenta.

A partir de esta información y de un estudio detallado de las prestaciones de cada uno de los controladores y sus implementaciones sugeridas, se establecieron las bases para dar comienzo al proyecto **Wireless Qi Charger**.

[8]Con algunas restricciones de cercanía entre dispositivos

Capítulo 2

Requerimientos y características del Sistema

Al momento de establecer las características del sistema a diseñar, y dado que el objetivo planteado es el diseño de un dispositivo *Qi* compatible, se siguieron de manera estricta los requerimientos impuestos por la norma[1]. A continuación se realiza una descripción detallada de dichos requerimientos desglosando en Hardware, Comunicación y Sistema de Control, haciendo en estas últimas dos secciones un fuerte énfasis en la información contenida en el estándar.

2.1. Hardware

2.1.1. Etapa de Potencia

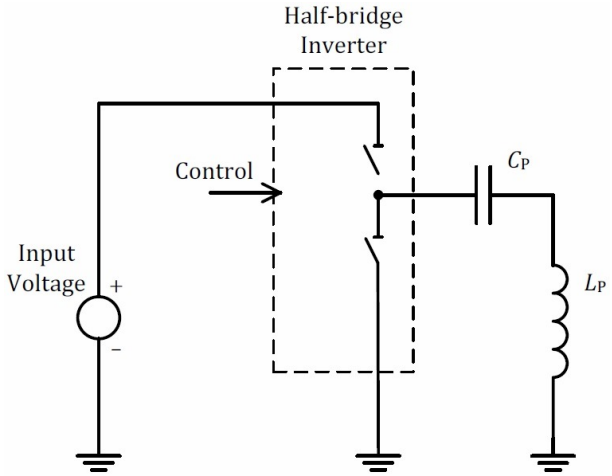
Señal

La evaluación de cómo diseñar la etapa de potencia, y en particular de cómo implementar el control de los puentes H que generan la onda en las bobinas, trajo aparejada la necesidad de consultar al departamento de Electrónica de Potencia del IIE. Esta necesidad fue producto del surgimiento, en reunión con el tutor, de dos soluciones posibles para el manejo de dichos puentes: directamente, conectando pines del controlador a los gates de los FET del puente; o mediante drivers específicos para puentes H de MOSFET.

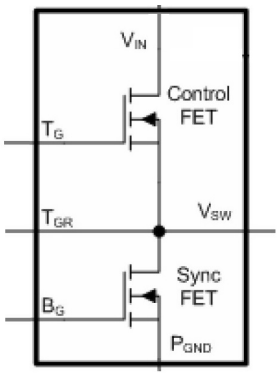
Previamente a la reunión coordinada con el Ing. Fernando Chiaramello[1] se tomó conciencia del problema existente con respecto al uso directo del controlador para manejar los puentes. Como deriva del circuito de la Fig. 2.1(a), el punto medio del puente tendrá una tensión variable y flotante respecto a tierra, hecho que se confirmó al observar la implementación de un puente nMOS Fig. 2.1(b) (imagen extraída de [3]). En esta imagen se observa que se cuenta con un pin de tensión de referencia entre source y gate de los transistores, de forma de poder asegurar el ΔV_{GS} necesario para activar el transistor de arriba independientemente del valor de la tensión V_{SW}. Efectivamente, en consulta con el docente, se recibió la enfática recomendación de controlar los puentes con drivers por la

[1] Asistente Grado 2 del Departamento de Potencia del IIE y docente del Taller de Electrónica de Potencia.

razón antes mencionada, además de la necesidad de separar las etapas de potencia y de control por razones de seguridad a la hora de probar el circuito.



(a) Diagrama eléctrico básico representado en el estándar [1]



(b) Puente implementado con nMOS

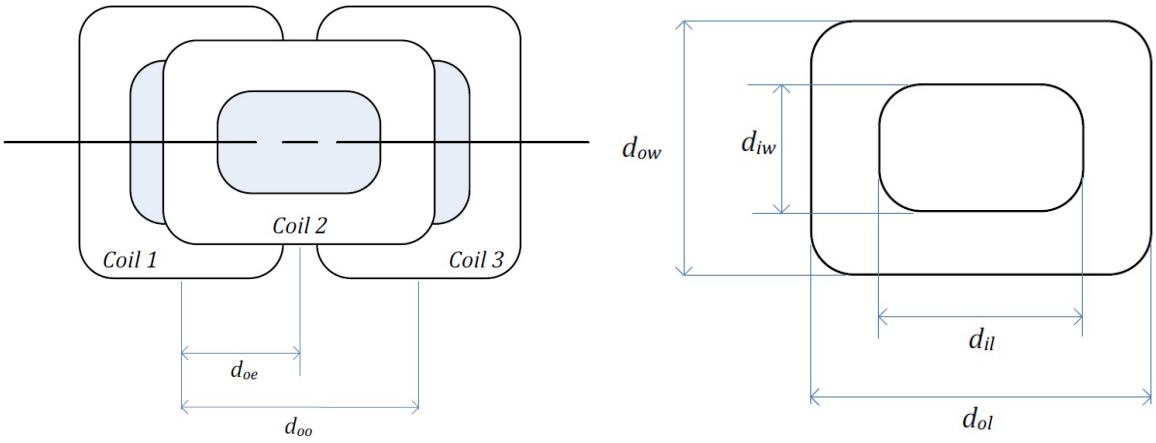
Figura 2.1: Análisis de la etapa de potencia

Para alimentar el bobinado (cuya configuración se muestra en la Fig. 2.2(a)) con la onda necesaria, y cumpliendo con las especificaciones impuestas por la norma para esta etapa, se requiere implementar el circuito que se ve en la Fig. 2.1(a) con las inductancias definidas para las bobinas, y capacitores de valores definidos en el estándar: $C_p = 0{,}136^{\pm 5\,\%}\mu F$ en serie con las bobinas laterales, y $C_p = 0{,}147^{\pm 5\,\%}\mu F$ en serie con la bobina central. Además, la norma informa que, en resonancia, la tensión a través de los capacitores puede llegar a $100V_{pp}$. En este punto es donde se presentó el primer problema práctico, ya que si bien existen registros de la existencia de capacitores de 136 y $140pF$ para $100V$, no se consiguieron estos valores, razón por la cual se utilizan en la implementación práctica dos paralelos de capacitores de $C_p = 0{,}068^{\pm 5\,\%}\mu F$, y un paralelo de esos valores con un capacitor de $C_p = 0{,}015^{\pm 5\,\%}\mu F$.

Bobinado

Con la decisión de comenzar por esta etapa, y dados los requerimientos presentes en la norma para los cargadores A6, era necesario contar con bobinados con la disposición mostrada en la Fig. 2.2(a), definiéndose $d_{oo} = 49{,}2^{\pm 4}mm$ y $d_{oe} = 24{,}6^{\pm 2}mm$, y con las características geométricas mostradas en la Fig. 2.2(b) y especificadas en la tabla 2.1. Además, en la norma se detallan los requerimientos respecto a las características eléctricas de cada bobina con el bobinado ya ensamblado, requiriéndose una autoinductancia para las bobinas laterales de $L_p = 12{,}5^{\pm 10\,\%}\mu H$, y una autoinductancia para la bobina central de $L_p = 11{,}5^{\pm 10\,\%}\mu H$. Con estos valores de inductancias (definidos por el estándar), y los valores de los capacitores detallados anteriormente (dentro de los rangos de valores posibles definidos por el estándar), se configuran tres circuitos resonantes con frecuencia de resonancia $f_{res}^{lat} = 122kHz$ para las bobinas laterales, y $f_{res}^{cen} = 120kHz$ para la bobina

central.



(a) Disposición del bobinado en cargadores A6 (b) Especificaciones geométricas de cada bobina

Figura 2.2: Especificaciones del Bobinado detalladas en la norma [1]

Parámetro	**Símbolo**	**Valor**	**Unidad**
Largo exterior	d_{ol}	$53{,}2^{\pm 0{,}5}$	mm
Largo interior	d_{il}	$27{,}5^{\pm 0{,}5}$	mm
Ancho exterior	d_{ow}	$45{,}2^{\pm 0{,}5}$	mm
Ancho interior	d_{iw}	$19{,}5^{\pm 0{,}5}$	mm
Espesor	d_c	$1{,}5^{\pm 0{,}5}$	mm
Número de vueltas	N	12	*vueltas*

Tabla 2.1: Valores geométricos requeridos para las bobinas.

Los demás componentes de este circuito son necesarios para la implementación del puente de la etapa de potencia, esto es: un puente *Half Bridge* y un driver para su manejo. Dado que el rango de señal que manejará esta etapa deberá ser $f_{op} = 115...205kHz$, tanto los puentes como los drivers deberán ser capaces de trabajar a frecuencias máximas de al menos $205kHz$. A su vez, dado el circuito de esta etapa, las corrientes máximas que debe manejar el puente rondan los $4A$, por lo que también se debe tener en cuenta este valor a la hora de elegir este componente.

En el capítulo *Hardware* (3) se hace referencia a la implementación general del hardware del sistema, y en la sección *Etapa de potencia* (3.1) y Anexo A - Profundización Hardware (6), se hace referencia a esta etapa en particular, detallando y analizando los componentes adquiridos con ese fin.

2.1.2. Controlador

En base a los requerimientos detallados por la norma con respecto al comportamiento que el sistema debe presentar, en términos de eficiencia energética y control a lo largo del proceso de transferencia de potencia, se hace evidente la necesidad de contar con un sistema dotado de cierta inteligencia. En efecto, el sistema debe ser capaz de decodificar e interpretar los mensajes recibidos y tomar decisiones en base a esta información y al estado en que se encuentra al recibirla. Dentro de las posibles acciones que el sistema debe ejecutar al recibir determinados mensajes, se encuentran el ajuste del punto de funcionamiento en la transferencia de potencia, la finalización del proceso de carga (por batería completa o por retiro del receptor de la superficie de emisión) y la detección de objetos extraños, entre otras cosas.

En este escenario se decidió para este proyecto la inclusión de un controlador en el sistema de transferencia, de forma de poder implementar la lógica necesaria para las acciones antes descritas mediante su programación. Así, ahondando en los requerimientos particulares del sistema, se definen las necesidades que el controlador deberá satisfacer:

- Contar con un mínimo de 2 pines de entrada, de los cuales por lo menos uno deberá ser capaz de levantar una interrupción en el μC al recibir una comunicación.
- Contar con un mínimo de 4 pines de salida con el fin de controlar los drivers de los puentes H; uno para la señal PWM global y cada uno de los 3 restantes para la señal de *enable* de cada uno de los drivers.
- Contar con un ADC de n bits, con $n > log_2(286)$, para la medición de corrientes de hasta 2A rms con 7mA de resolución, por lo que un ADC de 9 bits cumplirá con los objetivos.
- Posibilidad de generar una señal PWM de frecuencia variable, entre 115 y $205kHz$, y que se mantenga estable independientemente de que el controlador esté realizando otros procesos (decodificado de comunicaciones, sensado de corriente o procesos de control).
- Capacidad de generación de la señal PWM de forma de lograr el cumplimiento del requerimiento de la norma respecto a la resolución de un 0,1 % en el manejo del ciclo de trabajo de la señal de potencia.
- Capacidad de procesamiento tal que permita satisfacer las restricciones de tiempo del protocolo en cuanto a transiciones de estado y tomar de decisiones respecta, siendo capaz de tomar las acciones de control requeridas, en un tiempo máximo de $10ms$.
- Capacidad de procesamiento tal que permita decodificar la señal de comunicación cumpliendo con los tiempos de señal especificados por la norma.

Considerando las necesidades antes expuestas, en el capítulo *Hardware*, sección *Controlador* (3.2), se discutirá a fondo la decisión sobre la plataforma que se utilizó para la implementación del controlador del sistema.

2.2. Comunicación

En el capítulo de comunicaciones del estándar *Qi* se especifican los requerimientos de las señales a efectos de decodificar los mensajes recibidos en el emisor desde el receptor. Para llevar a cabo esta comunicación se utiliza modulación de *backscattering*[2] sobre la señal de potencia, y es sobre esta señal que se definen los parámetros de variación para la identificación de mensajes. En la Fig. 2.3 se muestra uno de los dos posibles diseño de receptor[3] detallados en la norma, donde puede observarse la carga (RC_{cm}) que el controlador utiliza para la modulación de la onda de potencia, según lo explicado en 1.1.4. En otras palabras, los mensajes se envían en amplitud modulada de la señal de potencia, estableciendo así un canal de comunicación y teniendo entonces la realimentación necesaria para el Sistema de Control.

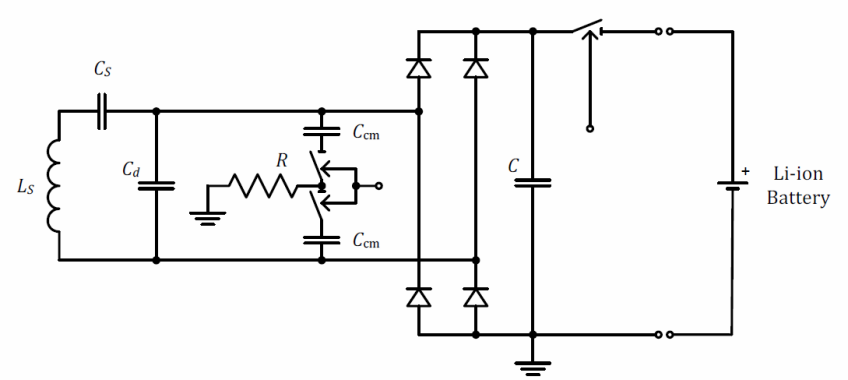


Figura 2.3: Esquema del diseño de un receptor detallado en [1].

Mediante este canal el receptor de potencia envía información al emisor, a partir de la cual este toma alguna de las siguientes acciones:

- Comienzo de transmisión de potencia luego de identificación de receptor válido.
- Identificación del dispositivo receptor.
- Determinación de la potencia a transferir (punto de funcionamiento).
- Reajuste del punto de funcionamiento en base al nivel de carga.
- Finalización de la emisión de potencia por ausencia de comunicación luego de $t_{timeout}$.
- Finalización de la emisión de potencia por diferencia considerable entre potencia emitida y potencia recibida.
- Finalización de la emisión de potencia por carga completa.

[2]Modulación de *retrodispersión*, en español.

[3]Diseño que escapa al alcance de este proyecto, dado que el receptor compatible se adquirió ya implementado.

2.2.1. Capa Física

A los efectos de establecer parámetros claros para la comunicación, se fijan los niveles de la onda de potencia de forma de parametrizar la modulación realizada y se definen los estados Hi y Lo en el primario, tanto en tensión como en corriente. Además, se determinan los tiempos de establecimiento y duración de los niveles para la modulación, cuyos valores se detallan en la tabla 2.2. A su vez, en la Fig. 2.4 se muestran estos requerimientos gráficamente, además de los valores definidos para las máximas variaciones permitidas dentro de un mismo nivel de señal (Δ detallados en la tabla).

Para una carga y un acople entre primario y secundario apropiados, uno de los estados posibles está determinado si la amplitud de la señal se mantiene un tiempo t_S dado y siempre y cuando se cumpla una de las siguientes condiciones, mediante las cuales se define la profundidad de modulación (*Modulation Depth* en la figura):

- La diferencia de la amplitud de la **corriente** en la bobina activa del primario entre los estados Hi y Lo es de, al menos, **15mA**.
- La diferencia de la amplitud de la **corriente** en la bobina activa del primario entre los estados Hi y Lo, y **medida en tiempos correspondientes a un cuarto de ciclo** del driver del puente H, es de al menos **15mA**.
- La diferencia de la amplitud de la **tensión** en la bobina activa del primario entre los estados Hi y Lo es de, al menos, **200mV**.

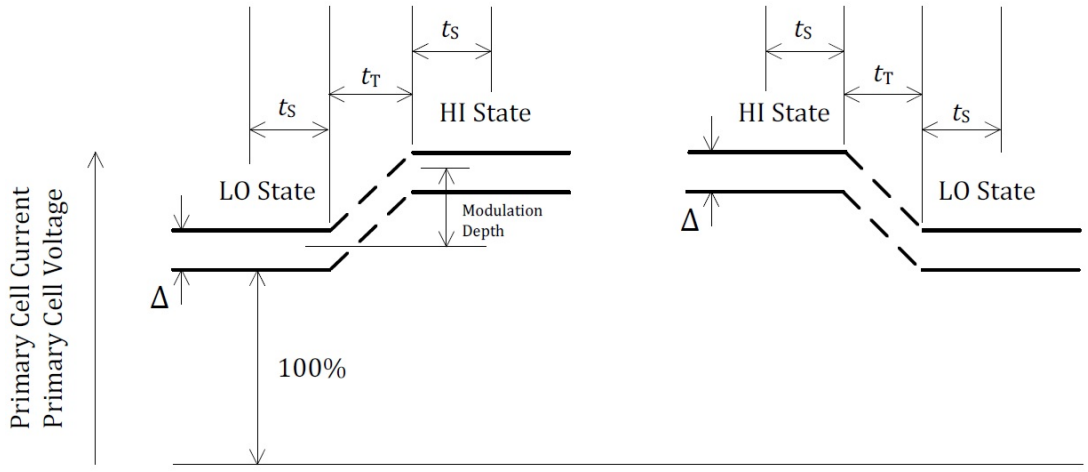


Figura 2.4: Especificación de los niveles de señal detallados en [1]

Parámetro	Símbolo	Valor	Unidad
Máximo tiempo de transición	t_T	100	μs
Tiempo estable mínimo	t_S	150	μs
Ripple máximo de corriente	ΔI_{nivel}	8	mA
Ripple máximo de tensión	ΔV_{nivel}	110	mV

Tabla 2.2: Valores de los parámetros para la modulación de la señal de potencia.

2.2.2. Codificación de datos

Con la capa física, y por ende los parámetros necesarios para tener un canal de comunicación definido, el estándar establece los esquemas de codificación para el flujo de datos desde el receptor hacia el emisor de energía.

Bits

Con respecto a la codificación de bits, el estándar establece la codificación que debe utilizar el receptor de potencia con el fin de establecer la comunicación a través de la onda modulada en amplitud. Basándose en esto es que se asume que cualquier receptor compatible debe sincronizar cada bit de datos con un período completo de un reloj interno, de forma tal que los flancos de subida del reloj y de los bits coincidan. Además, se define una frecuencia fija para este reloj interno tal que $f_{CLK} = 2^{\pm 4\,\%} kHz$.

Como se observa en la Fig. 2.5, la codificación diferencial utilizada codifica un UNO lógico con dos transiciones[4] de nivel de la señal de potencia durante un período de reloj, de forma que cada una coincida con uno de los flancos de f_{CLK}. A su vez, esta codificación identifica un CERO lógico si sólo se da una transición de la señal de potencia durante un período de reloj y es coincidente con el flanco de subida de f_{CLK}. La modulación de señal está a cargo del receptor de potencia, por lo que en este proyecto sólo será necesaria su interpretación.

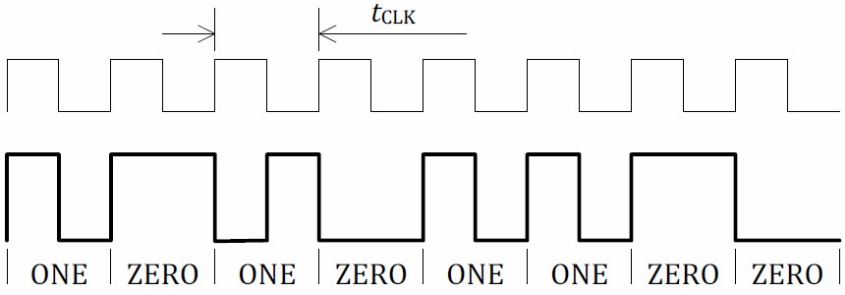


Figura 2.5: Codificación diferencial bifásica [1]

[4]Sin importar si el orden es subida-bajada o bajada-subida.

Bytes

Para la codificación de bytes, el estándar especifica que el receptor de potencia debe utilizar un formato de paquetes de 11-bits en serie asíncronos por byte a transmitir. El formato consta de un *Start bit*, los 8 bits de datos, un bit de paridad y un *Stop bit*. Como se muestra en la Fig. 2.6, con la codificación bifásica correspondiente, este formato tiene las siguientes características:

Start Bit: Este bit debe ser un CERO.

Orden: El primer bit de datos transmitido es el bit menos significativo.

Paridad: Este bit es de paridad **impar**. Es decir, el bit de paridad deberá ser un UNO si el byte contiene un número par de bits de valor UNO; y deberá ser CERO en otro caso.

Stop Bit: Este bit debe ser un UNO.

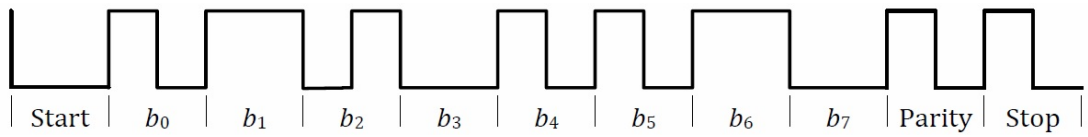


Figura 2.6: Byte codificado con el formato especificado - valor 0x35 para este ejemplo

Estructura de paquetes

La comunicación debe hacerse desde el receptor de potencia hacia el emisor, mediante el uso de paquetes. Estos paquetes se componen de 4 partes: el preámbulo, el encabezado, el mensaje y el checksum. Como se observa en la Fig. 2.7, dentro de cada paquete se respeta este orden de envío. El preámbulo consiste en un tren de entre 11 y 25 bits de valor UNO, codificados como se detalla en la sección Bits 2.2.2. Este preámbulo tiene como objetivos la sincronización por parte del emisor y la detección precisa del *Start bit* del encabezado. El conjunto encabezado-mensaje-checksum está compuesto por bytes codificados como se detalla en la sección Bytes 2.2.2, siendo siempre un byte de encabezado, un byte de checksum y por lo menos un byte de mensaje.

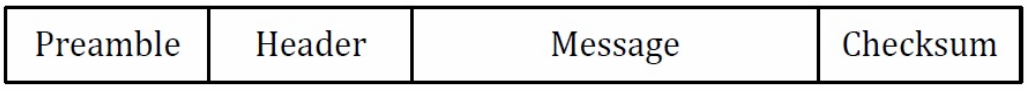


Figura 2.7: Formato de los paquetes utilizados para la comunicación

El emisor de potencia considerará un paquete recibido como válido si:

- Detecta al menos 4 bits de preámbulo seguidos de un *Start bit*.

- No detecta un error en la paridad de todos los bytes que comprenden al paquete. Esto es, los bytes de mensaje, el encabezado y el checksum.
- Ha detectado el *Stop bit* del *checksum byte*.
- Se ha determinado que el *checksum byte* es consistente.

En caso de no recibir correctamente un paquete, el emisor de potencia lo debe descartar, sin usar la información que contenga.

Una descripción general de las estructuras del preámbulo, el encabezado, los mensajes y el checksum, se presenta a continuación:

Preámbulo:

El preámbulo consiste en una sucesión de entre 11 y 25 bits "1", y tiene como objetivos la sincronización del emisor de potencia con la señal de comunicación y la detección precisa del *start bit* del encabezado.

Encabezado:

El encabezado consiste en un único byte mediante el cual se informa qué tipo de paquete se está enviando, y a partir del cual se puede calcular el número de bytes en el mensaje truncando a un entero el valor correspondiente según la tabla 2.3.

Encabezado	**Tamaño del mensaje*(*)***
0x00..0x1F	$1 + (Encabezado - 0)/32$
0x20..0x7F	$2 + (Encabezado - 32)/16$
0x80..0xDF	$8 + (Encabezado - 128)/8$
0xE0..0xFF	$20 + (Encabezado - 224)/4$

Tabla 2.3: Tamaño de mensaje según el valor del encabezado.

(*)El tamaño del mensaje corresponderá a la parte entera del resultado correspondiente.

Mensajes:

El mensaje dependerá del tipo de paquete que se esté enviando y deberá ser consistente con el tipo de mensaje detallado en el encabezado. Consistirá en por lo menos un byte, comenzando por el byte B_0, y podrá tener un largo máximo de 27 bytes.

Checksum:

El checksum consiste en un solo byte que le permite al emisor de potencia corroborar si hubo errores en la transmisión de datos. Para esto, el emisor debe calcular el checksum de la siguiente forma: $C = H \oplus B_0 \oplus B_1 \oplus .. \oplus B_{last}$. Siendo H el encabezado, B_i cada uno de los bytes del cuerpo del mensaje, y C el valor del checksum. En caso de existir coincidencia entre el checksum calculado y el recibido, el emisor de potencia deberá determinar que el checksum recibido es inconsistente.

Encabezado	**Tipo de Paquete**	**Tamaño de Mensaje**
Ping		
0x01	Signal Strength	1
0x02	End Power Transfer	1
Identification & Configuration		
0x06	Power Control Hold off	1
0x51	Configuration	5
0x71	Identification	7
0x81	Extended Identification	8
Power Transfer		
0x02	End Power Transfer	1
0x03	Control Error	1
0x04	Received Power	1
0x05	Charge Status	1
Identification & Configuration y *Power Transfer*		
0x18	Proprietary	1
0x19	Proprietary	1
0x28	Proprietary	2
0x29	Proprietary	2
0x38	Proprietary	3
0x48	Proprietary	4
0x58	Proprietary	5
0x68	Proprietary	6
0x78	Proprietary	7
0x84	Proprietary	8
0x$A4$	Proprietary	12
0x$C4$	Proprietary	16
0x$E2$	Proprietary	20

Tabla 2.4: Tipos de paquetes de datos definidos para la comunicación.

En la tabla 2.4 se detallan los tipos de paquetes definidos en la norma, diferenciados por el valor del encabezado, y lo tamaño del mensaje que cada paquete tendrá. Los valores de encabezado que no están en esta lista corresponden a paquetes reservados.

Capítulo 3

Hardware

En este capítulo se detallan los diseños hardware realizados, así como los componentes correspondientes adquiridos, a los efectos de la implementación física del sistema objetivo. La presentación de las distintas etapas se realiza por secciones dentro de este capítulo, y siguiendo el orden cronológico de avance. Esta presentación de las distintas etapas del sistema se condice con la forma modular de realización del proyecto, el cual se llevó a cabo haciendo en cada etapa de avance especial énfasis en uno de los siguientes puntos:

- Etapa de potencia
- Controlador
- Etapa de comunicación
- Etapas auxiliares
- Implementación del Hardware definitivo - Circuito impreso

A continuación de este capítulo, en el capítulo *Software* (4), respetando también el orden cronológico del desarrollo del proyecto, se detalla la etapa final la Programación del Software del Controlador.

3.1. Etapa de potencia

La *Etapa de Potencia* es la porción del sistema cuya función es transformar la tensión de alimentación, $12V$ DC, en la señal necesaria para lograr la inducción electromagnética mediante la cual se realiza la transferencia de potencia[1]. Considerando su importancia[2] es que se decidió comenzar enfrentando el desafío de su implementación. Dicha importancia reside en que tanto la transferencia de energía, como la comunicación y la verificación de presencia de receptor válido se llevan a cabo a través de esta etapa. Por ende, en las secciones *Diseño* (3.1.1), *Pruebas y Prototipo* (3.1.2) y *Resultados* (3.1.3) se detallan los drivers para los medios puentes H utilizados, los medios puentes H que manejan la señal provista a las bobinas, el bobinado utilizado para la transferencia y el diseño general de la *Etapa de Potencia*.

[1]Señal que también se utiliza para modular la comunicación, hecho que se detalla en la sección Etapa de Comunicación (3.3).

[2]El resto del sistema y el éxito de este proyecto dependen del buen funcionamiento de la etapa de potencia.

3.1.1. Diseño

Bobinados

En la norma se puede encontrar un detalle minucioso de las características constructivas que las bobinas de un emisor *Qi* deben presentar, especificando el conductor a utilizar para el bobinado (tipo, sección y cantidad de de hilos), la cantidad de vueltas de conductor, sus dimensiones físicas y la ubicación relativa entre ellas (para los casos de múltiples bobinas). Además, también se especifica el blindaje magnético que debe separar el bobinado del resto del sistema a los efectos de evitar interferencia o inducción no deseada. Por ende, dada la existencia en el mercado de bobinados para cargadores *Qi* y la complejidad que la construcción de un bobinado representaba, se tomó la primera definición con respecto al hardware a adquirir. Entonces, de forma de asegurarse de cumplir los requerimientos existentes respecto a las bobinas específicas, se optó por adquirir un conjunto bobinado/blindaje fabricado para cargadores *tipo A6*.

Tomada la decisión antes descrita y con el objetivo de adquirir el bobinado, una de las opciones evaluadas en principio fue la compra online[3] de bobinados A6, pero en el único caso que existía la opción de pedidos mínimos menores de 100 bobinados el envío resultaba excesivamente costoso. Por esto, se solicitaron muestras a Würth Electronik, obteniendo de forma totalmente gratuita varios bobinados *WE-WPCC Wireless Power Charging Transmitter Coil*[4] *Qi* compatibles, como el de la Fig. 3.1.

En las figuras 3.2(a) y 3.2(b) se presentan las características geométricas y eléctricas detalladas en la hoja de datos de este bobinado, y que confirman su utilidad para este proyecto.

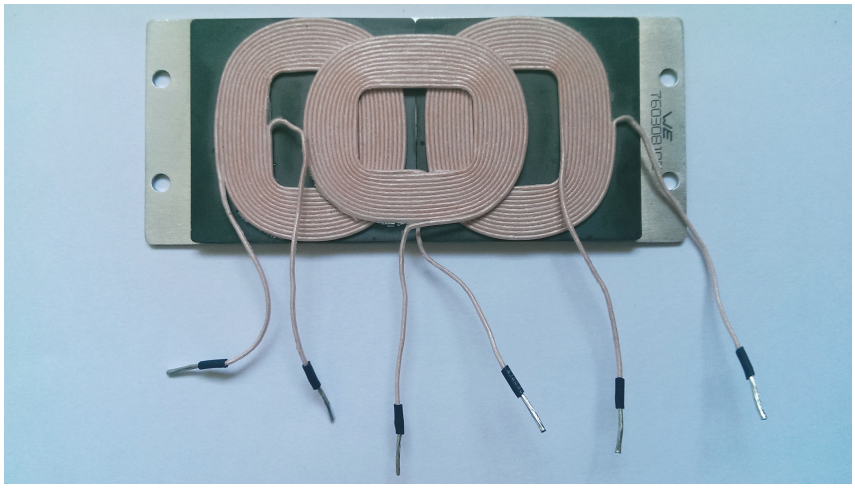
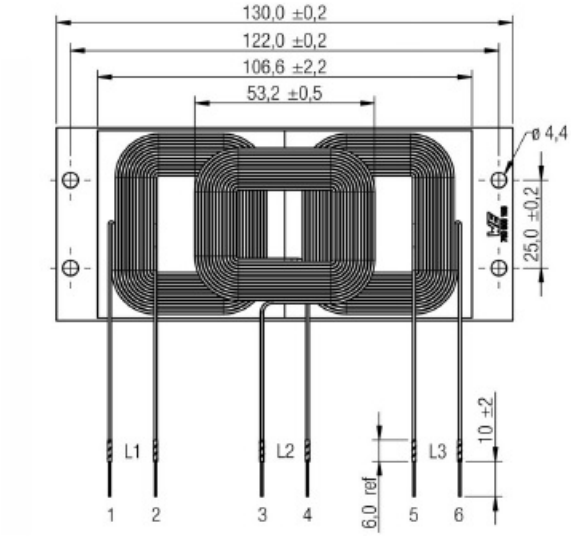


Figura 3.1: Bobinado A6 adquirido.

[3]Dado que estos bobinados no se consiguen en nuestro país.
[4]Model 760308106 A6

Generación de señal

Considerando los *Requerimientos del sistema* (2.1.1), e incluso buscando tener márgenes de sobra (en lo que a corriente y frecuencia máximas respecta) para obtener un circuito de mayor robustez, se adquirieron medios puentes H[5] y sus respectivos drivers[6] de Texas Instruments. Las características relevantes de estos integrados a los efectos de este proyecto, que se encuentran en las hojas de datos correspondientes, se presentan en las tablas 3.1 y 3.2.



(a) Detalles geométricos

Properties	Test conditions		Value			Unit	Tol.
			1	2	3		
Inductance	125 kHz/ 10 mA	L	12.5	11.5	12.5	µH	±10%
Q-factor	125 kHz/ 10 mA	Q	125	125	125		typ.
Rated Current	ΔT = 40 K	IR	9.0	9.0	9.0	A	max.
Saturation Current		I_{sat}	10.0	10.0	10.0	A	typ.
DC Resistance	@ 20°C	R_{DC}	55	55	55	mΩ	typ.
DC Resistance	@ 20°C	R_{DC}	65	65	65	mΩ	max.
Self resonant frequency		f_{res}	14	14	14	MHz	

(b) Especificaciones eléctricas

Figura 3.2: Características presentadas en la hoja de datos de los bobinados Wurth adquiridos.

Half-Bridge Power Block:

Parámetro	MIN	MAX	Unidad
V_{GS} - Tensión Gate-Source	4,5	8	V
V_{IN} - Tensión de alimentación	-	27	V
f_{SW} - Frecuencia de inversión	-	1500	kHz
I - Corriente de operación	-	25	A
T_J - Temperatura de operación	-	125	^oC

Tabla 3.1: Condiciones recomendadas de operación del puente a 25^o.

[5]Texas Instruments CSD87352Q5D $NexFET^{TM}$[4]
[6]Texas Instruments TPS28225-Q1[3]

High Frequency power block Driver:

Parámetro	MIN	MAX	Unidad
V_{DD} - Tensión de alimentación	4,5	8,8	V
V_{PWM}, $V_{EN/PG}$ - Tensión de señal	$-0{,}3$	13,2	V
f_{SW} - Frecuencia de operación	-	2	MHz
T_J - Temperatura de operación	-40	125	^oC

Tabla 3.2: Condiciones recomendadas de operación del driver.

Por lo que se deduce de la comparativa entre los *requerimientos para la etapa de potencia* (2.1.1) y el hardware aquí descrito, efectivamente se cumple lo mínimo necesario e incluso se cuenta con amplios márgenes para trabajar con menores riesgos en la implementación. Un estudio detallado de la exigencia que este sistema impone a los componentes recientemente detallados puede ser consultado en el Anexo A - Profundización Hardware (6).

En primera instancia, y dada la presentación modular de las bobinas adquiridas (ver Fig. 3.1), se diseñó un circuito tipo módulo de prueba de la etapa de potencia con el fin de conectarlo al bobinado y a un generador de ondas. El objetivo de este circuito fue centrarse en la etapa de potencia en general para probar su funcionamiento y, en particular, en la formación de la onda que debe realizar el conjunto puentes-drivers para las bobinas a partir de la alimentación DC y sin necesidad del controlador. Se apuntó, con esto, a tener en primera instancia un sistema *dummy* que logre transferir potencia eléctrica a un receptor apto, sin tener en consideración la comunicación proveniente desde este receptor y aprovechando que es el emisor el encargado de controlar[7], a partir de esa comunicación, el funcionamiento del sistema.

Al diseñar el PCB se notó que el aspecto más restrictivo es el ancho de pista requerido por los integrados y la separación necesaria entre estas. Como puede verse en las figuras 3.3 y 3.4 (tomadas de [4] y [3] respectivamente) el ancho mínimo de pista necesario es de 0.35 mm y su separación mínima es de 0.3 mm, y en ambos requerimientos el más restrictivo es el driver[8].

En la Fig. 3.5 se presenta el esquemático del circuito diseñado siguiendo las recomendaciones presentes en las hojas de datos de los componentes y tomando en cuenta las necesidades para este proyecto. En base a eso, y para la fabricación de este prototipo de prueba, se recurrió a un proveedor de plaza para llevar a cabo el diseño presentado en la Fig. 3.6.

[7] Decidiendo si comenzar la transferencia, modificando punto de funcionamiento o potencia transferida y finalizando la carga.

[8] TPS28226-Q1

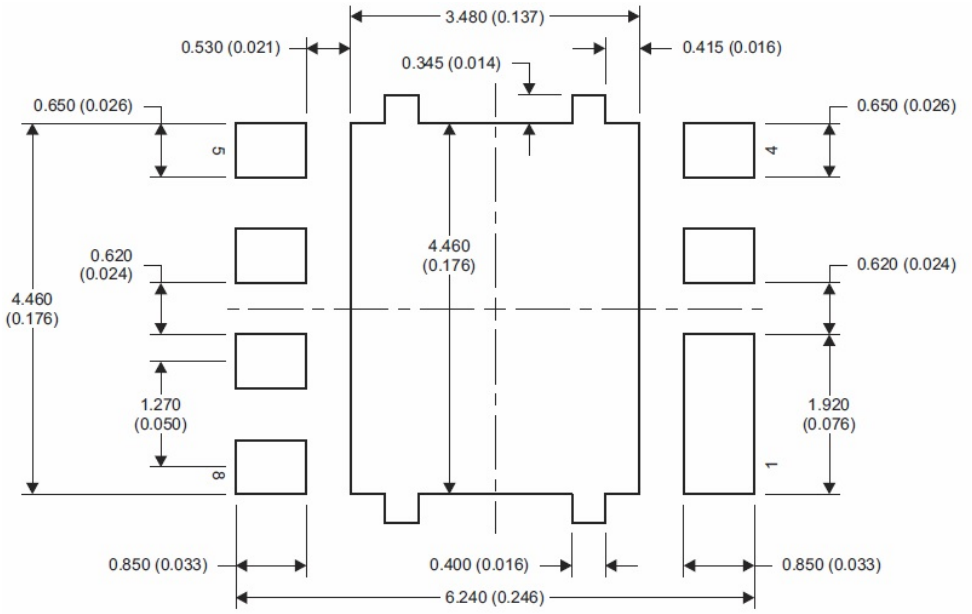


Figura 3.3: *Land pattern* del Puente H.

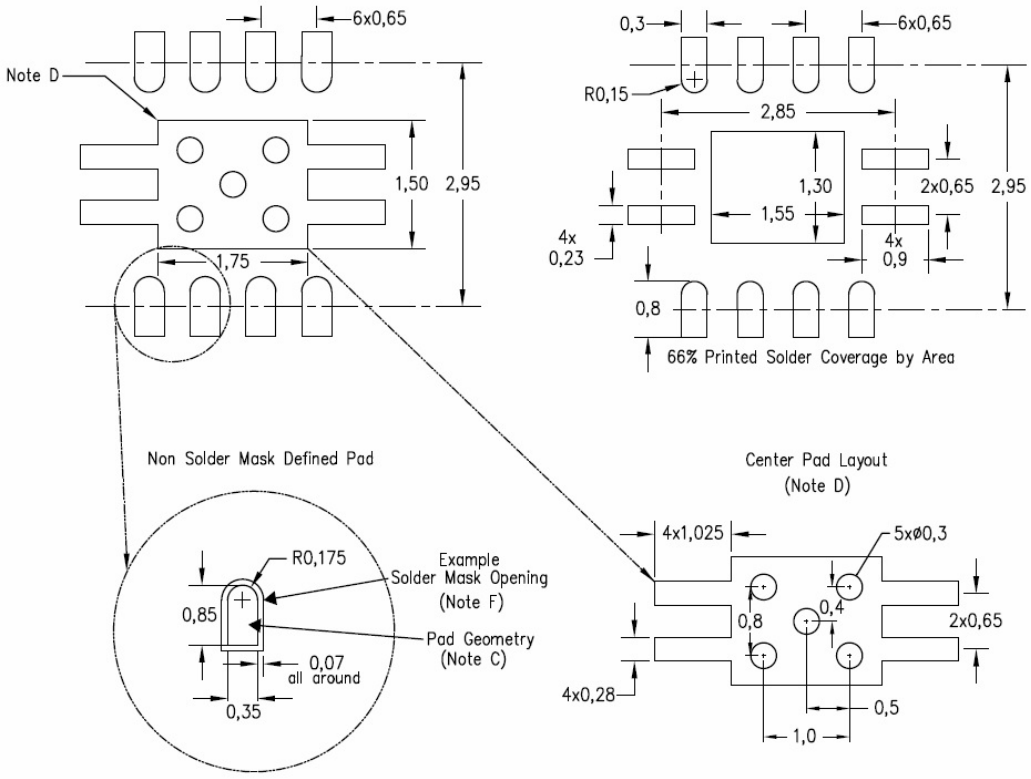


Figura 3.4: *Board layout* del driver de los puentes.

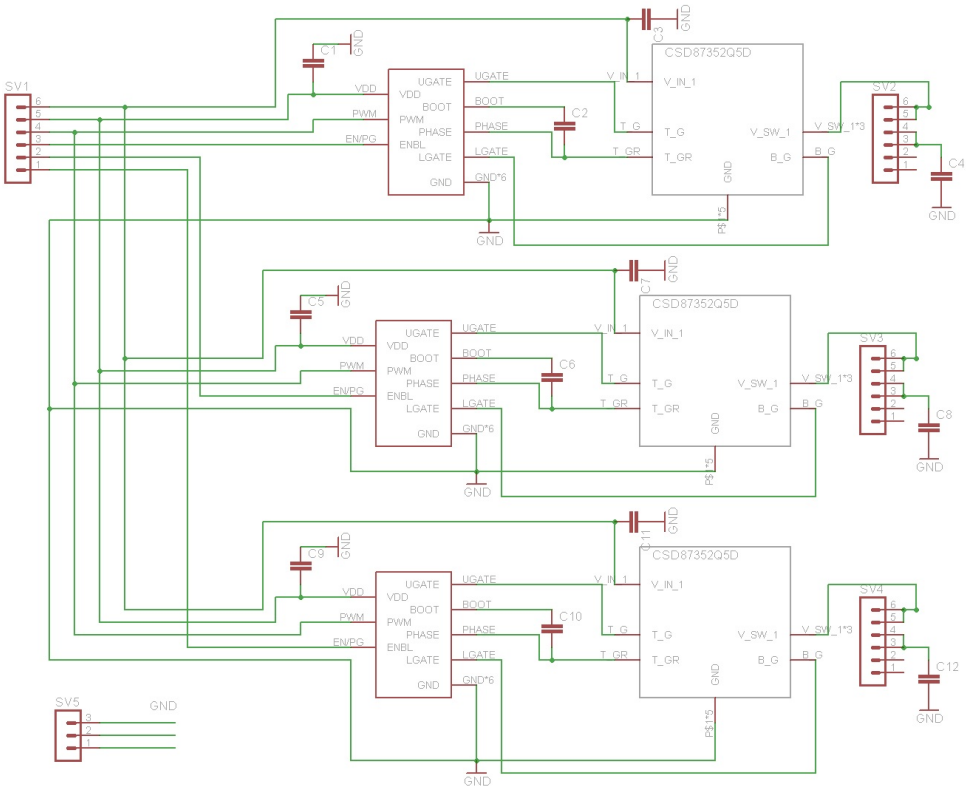


Figura 3.5: Esquemático del circuito del manejo de bobinas en la Etapa de potencia.

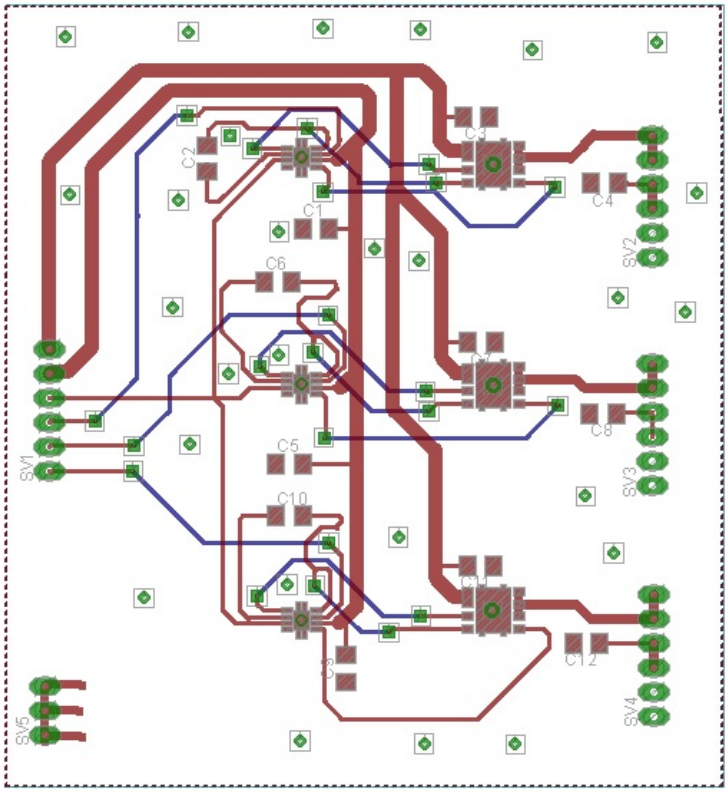
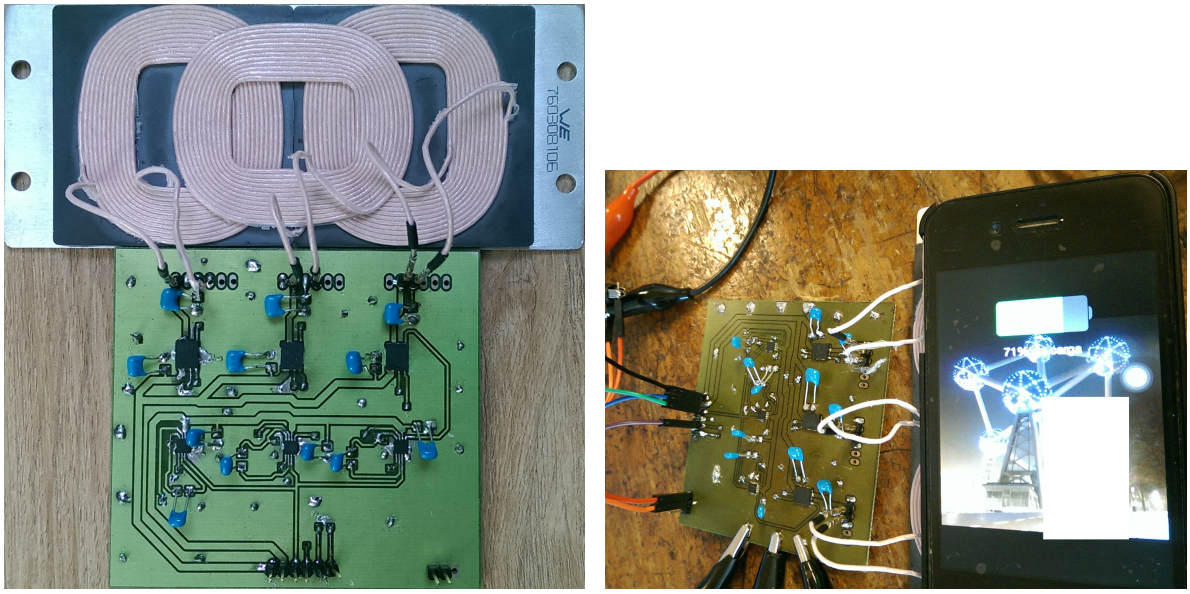


Figura 3.6: Layout para el impreso del circuito del manejo de bobinas.

3.1.2. Pruebas y prototipo

En la Fig. 3.7(a) se muestra el prototipo implementado a los efectos de probar la etapa de potencia, y sobre el cual se realizaron testeos para comprobar su correcto funcionamiento y llevar a cabo ajustes, en caso de ser necesarios. Estas pruebas consistieron en conectar un generador de señal para proveer la señal PWM necesaria, una fuente DC para alimentar a los medios puentes H ($12V$) y a los drivers ($5V$), y controlar el circuito a través de los drivers con las respectivas señales *Enable* ($5V$).



(a) Etapa de potencia implementada.

(b) Transferencia de potencia exitosa.

Figura 3.7: Etapa de potencia y transferencia exitosa en laboratorio.

3.1.3. Resultado

El éxito comprobable en esta instancia fue obtener una recepción de energía exitosa por parte de un receptor compatible, y el paso intermedio a tales efectos fue comprobar la llegada de las señales en las condiciones necesarias a los puntos del circuito. Con este fin, fue de mucha utilidad el análisis del circuito en funcionamiento mediante un osciloscopio, así como también lo fue para identificar los trenes de paquetes de comunicación sobre la señal de potencia enviados por el receptor durante una transferencia de potencia. En la Fig. 3.7(b) se muestra el momento en que el receptor identificó la recepción de potencia y carga de batería correspondiente.

Con esta etapa en estado funcional y apta para interactuar con un controlador que genere las señales de control correspondientes para cada etapa del proceso de carga, se pudo dar paso a la implementación de dicho controlador en la plataforma seleccionada, detallada en la sección *Controlador* (3.2). Además, con la validación del diseño de esta etapa, se pasó a contar con el diseño hardware del primer módulo necesario para la implementación final del PCB, y esta comprobación particular de su funcionamiento significó un paso importante hacia los objetivos generales del proyecto.

3.2. Controlador

En base a los requerimientos del sistema presentados en la sección *Requerimientos del Sistema* (2.1.2) y, por ende, a las prestaciones mínimas que debe presentar el controlador, se tomó la decisión de la plataforma a utilizarse para implementar dicho controlador. Si bien, como se menciona en la sección (3.3), la opción de utilizar un ADC para el procesamiento de la señal recibida en la bobinas y, de esta forma, obtener los correspondientes paquetes de datos contenidos en la señal de comunicación fue descartada, la opción de utilizar un μC resultó de igual forma ser la más conveniente. Los fundamentos para esta decisión, además de la persistente necesidad de contar con un ADC para el sensado de corriente, residen en la preponderante necesidad del manejo de estados y procesamientos de alto nivel, frente a la necesidad de contar con un gran número de señales independientes entre sí, e independientes del estado del controlador. Respecto a esta característica en particular, jugó un papel fundamental la posibilidad de generar la señal PWM con alguno de los timers presentes en la mayoría de los μC. Además de esto, y aportando aún más en el sentido de la elección de esta plataforma, se observó que dados los requerimientos de tiempos impuestos por la norma, la frecuencia de la señal de comunicación y la frecuencia de procesamiento de la mayoría de los μC de bajo costo con los que se podía contar para la realización de este proyecto, se hacía perfectamente manejable la decodificación de los mensajes, sensado de corriente, control del sistema y ajustes de señal de potencia dentro de prácticamente todos los parámetros requeridos por el estándar.

3.2.1. Plataforma

Buscando llevar a cabo la mejor implementación posible se consideraron, en primera instancia, dos opciones posibles respecto a la plataforma a utilizar para el controlador del sistema, presentando cada una de ellas sus ventajas y desventajas:

FPGA

Esta opción presentaba ventajas en los casos que fuera necesario realizar manejo de señales de bajo nivel con requerimientos exigentes respecto a estabilidad, slew rate y/o precisión. Además, la independencia de sus pines generales de I/O brindaba la posibilidad de realizar procesos en paralelo para distintas señales, característica que podía resultar conveniente a la hora de procesar las señales de comunicación mientras se mantenían las señales de control de la etapa de potencia (brindar un mayor ancho de banda para manejo de señales). Sin embargo, esta opción no resultaba ser la más conveniente cuando de la realización de procesos de alto nivel se tratara, ya que los cálculos complejos, vectoriales y de punto flotante requieren una excesiva cantidad de compuertas lógicas para ser alcanzados.

Microcontrolador

La opción de utilizar un μC presentaba, como ventaja primordial, la posibilidad de aprovechar la experiencia en el uso de Statecharts[9], considerando la conveniencia

[9]Experiencia adquirida al realizar el proyecto final del la asignatura *Sistemas Embebidos para Tiempo Real*.

que esto implica a la hora de implementar máquinas de estados concurrentes para sistemas reactivos. Además, presenta ventajas notorias frente al FPGA respecto al procesamiento de alto nivel. Su desventaja, como parece razonable, es que su funcionamiento se rige por hilos de ejecución, y cada ISR[10] detiene la ejecución del programa principal, o sea, no realiza procesos en paralelo como ocurre en los FPGA. También presenta desventajas en lo que a manejo de señales de bajo nivel respecta, problema que puede ser paliado, en parte, con el uso de los periféricos con los que cuentan los μC (Timers, por ejemplo).

Otro aspecto comparativo a resaltar, a la hora de evaluar qué opción es más conveniente, es el hecho de que las implementaciones de este tipo de cargadores son realizadas tanto con controladores específicos para tales fines, como mediante el uso μC, pero no se encontraron implementaciones realizadas con FPGA.

Dadas las consideraciones técnicas mencionadas, y consideraciones respecto a los costos, pensando en una posible producción comercial futura del sistema en desarrollo[11], es que se buscó utilizar un μC de bajo costo. En este sentido, se evaluaron las opciones que tuvieran compatibilidad con la plataforma de desarrollo *IAR Embedded Workbench*, de forma de aprovechar la experiencia con la que se contaba y de poder utilizar la herramienta *IAR VisualState*. Así, la decisión se redujo a elegir entre un μC MSP430 de Texas Instruments y un STM8 de ST Microelectronics, siendo el MSP430 el único de estos con el cual se tenía experiencia. Sin embargo, al evaluar tanto los costos como las prestaciones de ambos, y su relación con los requerimientos de este proyecto, resultó más conveniente la utilización del STM8 (similitud en prestaciones, y un costo aproximadamente 6 veces menor); y al consultar con el Ing. Leonardo Steinfeld[12] sobre las dificultades que podría traer aparejada esta decisión, aconsejó seguir este rumbo por no encontrar grandes diferencias entre la programación de ambos μC.

STM8S

La línea de μC de ST Microelectronics que mejor se adaptó a los requerimientos del sistema, sin apartarse demasiado de los objetivos de este proyecto, resultó ser la STM8S Series. Esta serie, basada en tecnología de 130nm, con arquitectura CISC-Harvard y Core STM8 de 8-bits, presenta las siguientes características relevantes a los efectos de este proyecto:

- Reloj interno de hasta $16MHz$, u oscilador externo de hasta $24MHz$.
- Timers con múltiples canales y capacidad de generación de PWM.
- Conversores analógico-digitales.
- Entre 4 y $128kB$ de memoria Flash.
- Entre 1 y $6kB$ de memoria RAM.

[10] Interrupt Service Routine

[11] En el Anexo C - Evaluación de costos (8) puede encontrarse información sobre esto.

[12] Profesor Adjunto Grado 3 del Departamento de Electrónica del IIE y docente de las asignaturas Sistemas Embebidos para Tiempo Real y Redes de Sensores Inalámbricos

- Entre 128 y 2048$bytes$ de EEPROM.
- Pines I/O de propósito general (GPIO).

En particular, el modelo de μC adquirido para la implementación de este proyecto es el STM8S105K6[5], que cuenta con 32$kbytes$ de memoria Flash, 2$kbytes$ de RAM y 1$kbyte$ de EEPROM, 8 canales de timers (2 con posibilidad de generar PWM), ADC de 10$bits$ y hasta 28 GPIO. Con relación a este último punto cabe mencionar que, luego de recibidos los μC, se evaluó la posibilidad de usar un modelo con iguales características pero de 20 pines y sólo 12 GPIO, dado que en definitiva sólo fueron utilizados 7 pines GPIO en el diseño final; sin embargo no se encontró una versión con esa cantidad reducida de pines y la cantidad de memoria necesaria. De todas formas dado el tamaño de los bobinados A6, las dimensiones del PCB estaban determinadas, razón por la cual no se experimentó falta de espacio para la implementación del diseño, por lo que el uso de la versión del micro con 32 pines sólo representa un desaprovechamiento de recursos. Además, aprovechando esta disponibilidad de recursos, se agregaron al diseño 4 conectores ruteados hacia pines GPIO libres, con el objetivo de ser utilizados para debugging o futuros desarrollos sobre este diseño.

Un punto relevante respecto al controlador implementado con este μC es **la imposibilidad de cumplir con uno de los requerimientos impuestos por la norma**, sobre la resolución que se debe tener en el manejo del ciclo de trabajo de la señal PWM. La norma especifica que "... un emisor de potencia tipo A6 **debe** controlar el ciclo de trabajo de la señal de potencia (PWM) con una resolución de 0,1 % o superior", y el manejo de la señal de potencia en este tipo de cargadores se hace efectivo variando la frecuencia entre 115 y 205kHz con un ciclo de trabajo del 50 % y el ciclo de trabajo de la señal entre 10 % y 50 % a 205kHz. Por ende, la resolución del ciclo de trabajo debe ser de por lo menos 0,1 % a 205kHz, haciendo que el timer que genera la señal PWM requiera, al menos, una frecuencia de reloj de 205MHz para satisfacer este requerimiento. Al evaluar este punto en conjunto con el tutor del proyecto se concluyó que la utilización de un procesador de tales características, con el único objetivo de cumplir con sólo un requerimiento específico, sería desmedida, mientras que los objetivos globales del presente proyecto pueden ser satisfechos con un μC de poco consumo y bajo costo, como el que se utilizó.

Más detalles sobre este requerimiento y sus implicancias, así como el alcance de otros requerimientos y evaluaciones del hardware implementado, pueden ser consultados en el Anexo A - Profundización Hardware (6).

3.2.2. Diseño

Como se mencionó previamente en esta sección, el plan para llevar adelante el diseño del controlador fue, desde el comienzo, la utilización de StateCharts. Esta idea surgió luego de la implementación, mediante esta herramienta, de un sistema básico de control de estados para un emulador de cargador *Qi*, en el marco de la asignatura *Sistemas Embebidos para Tiempo Real*. En esa instancia se tuvo un primer acercamiento a la utilización y prestaciones de esta herramienta mediante el software IAR VisualState[6], a raíz de lo cual este software pasó a ser la opción más conveniente al momento del diseño del controlador para este proyecto. De todas formas, y a raíz de la charla *Máquinas de estados UML y el Framework RKH^{TM}* y el taller *Framework RKH^{TM}: la práctica de programación dirigida por eventos con Statecharts*[13], que tuvieron lugar al mismo tiempo que se estaba en condiciones de generar la estructura del código del controlador a partir de la herramienta VisualState, se manejó la opción de utilizar el Framework, dado que se veía con buenos ojos la utilización de una plataforma abierta, en lugar de un software privativo. Sin embargo, luego de una evaluación profunda de las opciones[14], las adaptaciones del trabajo ya realizado a la fecha necesarias para este cambio de rumbo, las ventajas y desventajas de cada opción de implementación, y el tiempo que requeriría el aprendizaje del Framework, se decidió seguir la idea original e implementar la estructura del sistema con la herramienta IAR VisualState. Más información sobre la implementación del diseño y el software del controlador puede ser consultada en el capítulo Software (4).

3.3. Etapa de comunicación

Se denomina *Etapa de Comunicación* a la porción del sistema que toma la señal proveniente del bobinado de la *Etapa de Potencia* (3.1) a su entrada, y a su salida le entrega al Controlador del sistema (3.2) una señal de comunicación binaria, codificada según los lineamientos establecidos en el capítulo *Communications Interface* del estándar, con los paquetes de mensajes que el receptor modula sobre dicha onda de potencia. Esta etapa está compuesta por un reductor de tensión, seguido por un filtro pasabajos y, por último, un comparador, cuya salida se conectará directamente a uno de los pines del controlador del sistema.

Al momento de optar por la plataforma a utilizarse para la implementación del controlador, se tuvo en consideración los requerimientos que esta debía cumplir para el procesamiento de las señales de comunicación. La opción de un μC que contara con un ADC a tales efectos fue la que, a priori, pareció más conveniente, ya que entregándole al micro esta señal para que hiciera el procesamiento se mostraba como la solución más sencilla. Sin embargo, un paso previo ineludible para esta implementación sería la reducción de tensión de la señal proveniente de las bobinas, que presenta una amplitud que ronda los $20V_{pp}$, pudiendo incluso llegar a tensiones de $100V_{pp}$ según el estándar, mientras que los

[13] Actividades organizadas por el Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería y el Capítulo Uruguay de la IEEE Circuits & Systems Society (CAS) en Julio 2015, dictadas por los creadores del Framework RKH^{TM}, Ing. Leandro Francucci e Ing. Darío Baliña.

[14] Con la colaboración de Leonardo Steinfeld y Leandro Francucci.

máximos valores de input permitidos para un controlador son V_{cc} (3,3 o $5V$). A raíz de esto se buscó bibliografía relacionada al tratamiento de las señales que los fabricantes de integrados específicos implementan, encontrando que todos[15] delegan el procesamiento de la señal a un bloque externo al controlador, dejando para este sólo las tareas de demodulación y decodificación binarias. Si bien en general las *application notes* no detallan cómo implementar esta etapa de decodificación, sí se encontró una *application note* de Freescale[2] muy detallada sobre el procesamiento de señales realizado para la comunicación de un emisor *Qi*. En base a esto, y dado que la salida de este bloque es directamente la señal binaria de los paquetes de mensajes enviados por el receptor de potencia, y codificada según el esquema antes descrito (2.2.2), se procedió a realizar un prototipo de este bloque y analizar su funcionamiento.

Cabe acotar, además, que esta opción de implementar un bloque independiente para el procesamiento de las señales de comunicación sigue la línea general de modularización de etapas que este proyecto presenta, ya que con esto se le entregará al controlador solamente los paquetes de mensajes. Entonces, a continuación se describirán el *Diseño y la Implementación* para el procesamiento de señales, el *Prototipo* de pruebas realizado y por último se mostrarán los *Resultados* obtenidos en las pruebas sobre el prototipo de la *Etapa de Comunicación*.

3.3.1. Diseño

Como se explicó en *Requerimientos del Sistema - Comunicación* (2.2) este sistema utiliza modulación de retrodispersión sobre la señal de potencia para llevar a cabo la comunicación desde el receptor de potencia, hacia el emisor. Las características de esta señal están claramente definidas, y bien establecidos sus rangos de frecuencia, ya que la señal modulada es de frecuencia fija, $f_{com} = 2kHz$, y la onda de potencia (portadora) puede variar en el rango entre 115 y $205kHz$. A partir de la señal con estas características, y con un bloque que implemente las funciones presentadas en el diagrama de bloques de la Fig. 3.8, se obtendrá a la salida una señal binaria correspondiente al paquete de mensaje transmitido.

Figura 3.8: Diagrama de bloques del procesamiento de la señal.

Entonces, la etapa deberá implementar las tres funciones siguientes:

- Reductor de tensión
- Filtro pasabajos
- Comparador

[15]Se estudiaron hojas de datos de controladores de Texas Instruments, NXP, IDT y Freescale

A continuación se detalla la implementación realizada para alcanzar cada una de estas funcionalidades, para lo que se recurrió casi exclusivamente a la *application note* de Freescale [2].

Reductor de tensión

La tensión en las bobinas varía, generalmente, entre los 16 y los $32V_{pp}$, pudiendo alcanzar por momentos valores superiores a los $50V_{pp}$. Es por esto que la principal función de esta etapa es llevar la tensión de la onda proveniente de las bobinas a valores manejables por el controlador, respetando la forma de dicha onda a los efectos de conservar la información contenida. Además, se busca centrar el recorrido de la tensión de salida en $2.5V$, y que varíe entre 0 y $5V$. En la Fig. 3.9 se muestra el esquemático del circuito a implementar con este fin.

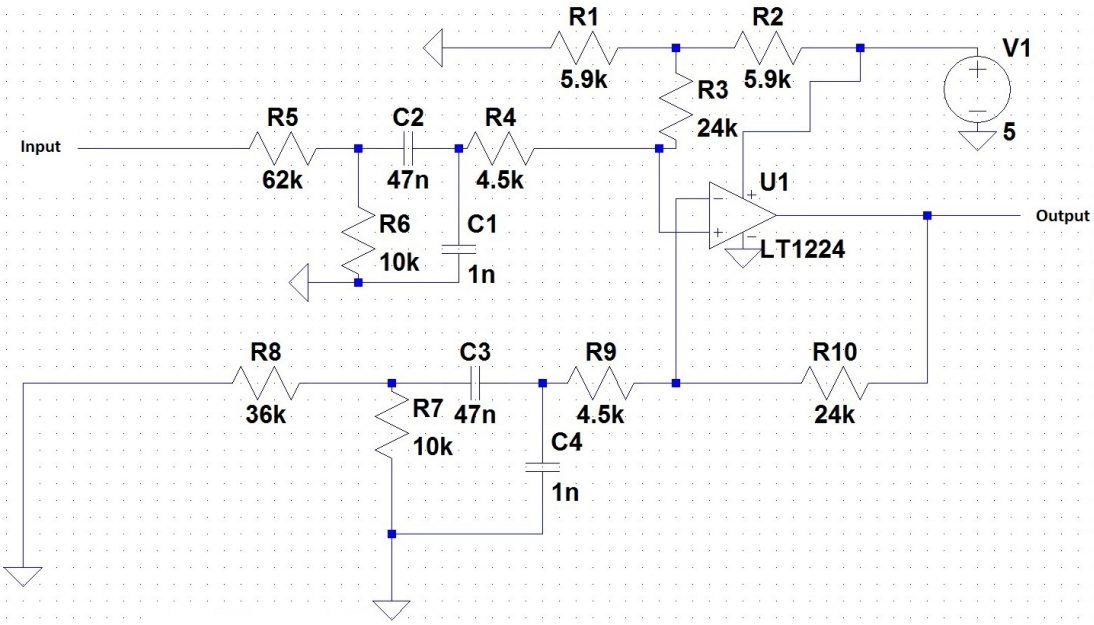


Figura 3.9: Esquemático del circuito reductor de tensión.

Una característica distintiva de la implementación realizada por Freescale, respecto a la aquí presentada, es su utilización en modo diferencial para ajustar el punto medio de salida en $2.5V$ para una señal de entrada completa. Sin embargo, dada la rectificación previa existente en el sistema a implementarse en este proyecto, se decidió utilizar el sistema en modo simple contra tierra, obteniendo de esta forma (como se muestra en *Resultados* (3.3.3)) también una señal centrada en $2.5V$, asimétrica, pero conservando de igual forma la información de los paquetes de datos.

Filtro Pasabajos

Dado que en lo que a la comunicación respecta, la señal de interés es la de baja frecuencia ($2kHz$) que está modulada sobre una portadora de frecuencias por encima de los

$115kHz$, el claro objetivo de esta etapa es el filtrado pasabajos de la señal recibida. Con esto se busca obtener, con la menor atenuación posible, solamente una señal que responda a las excitaciones realizadas por parte del receptor de potencia a los efectos del envío de paquetes de datos. En la Fig. 3.10 se muestra un esquemático del circuito a implementarse para cumplir con el filtrado pasabajos de la señal proveniente de las bobinas.

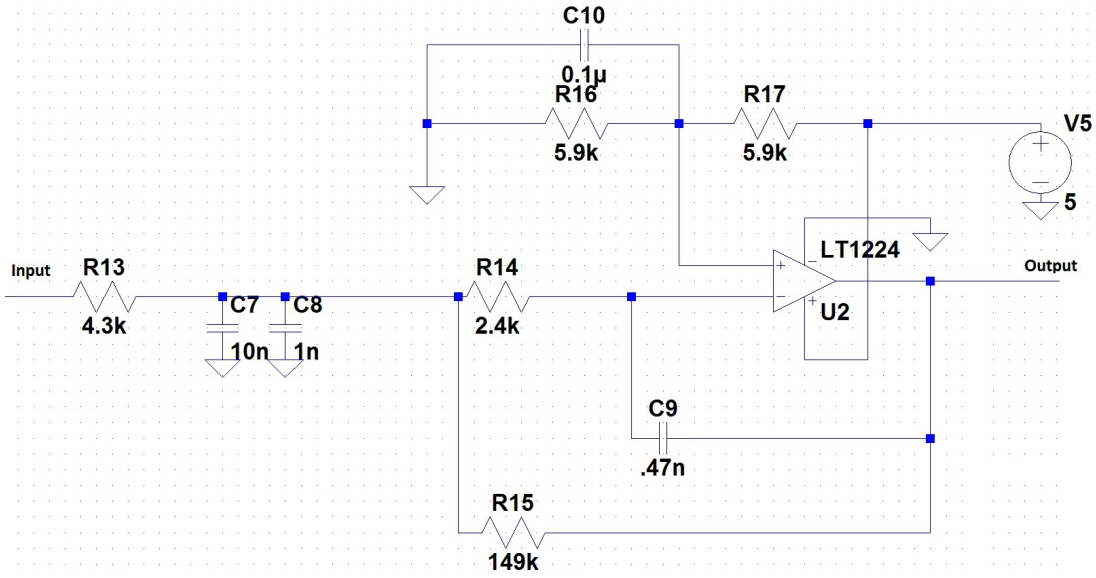


Figura 3.10: Esquemático del filtro pasabajos de segundo orden.

Los valores de los componentes presentados en el esquemático de la Fig. 3.10 fueron directamente tomados de la *application note* de Freescale, y a partir de ellos se obtiene un filtro pasabajos de segundo orden con una ganancia en banda pasante de $15dB$ y una frecuencia de corte -3dB de $4.9kHz$. En la Fig. 3.11 se muestra el resultado de filtrar una señal como la mostrada en 1.4 con este filtro.

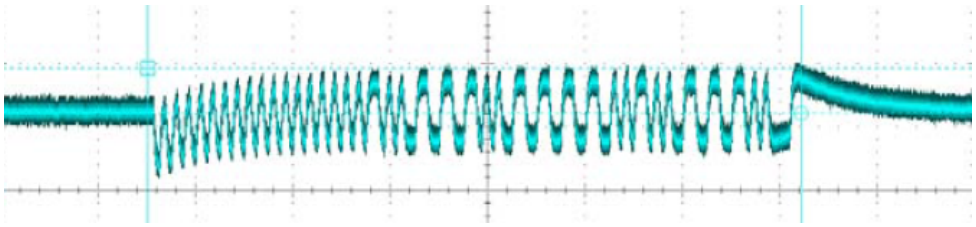


Figura 3.11: Señal obtenida luego del filtrado (imagen extraída de [2]).

Comparador

Por último, y con el objetivo de entregar al controlador una señal cuadrada que responda a la señal binaria modulada sobre la onda de potencia, se implementa un comparador. Dada la tensión media de $2.5V$ fijada en la etapa de reducción de tensión, se determina dicha tensión como el umbral para la inversión de la salida del comparador. Además, como

puede observarse en la Fig. 3.12, la configuración del amplificador es tal que su output swing[16] es el rango $[0..5V]$, de manera que esta señal de salida puede ser directamente leída por el controlador.

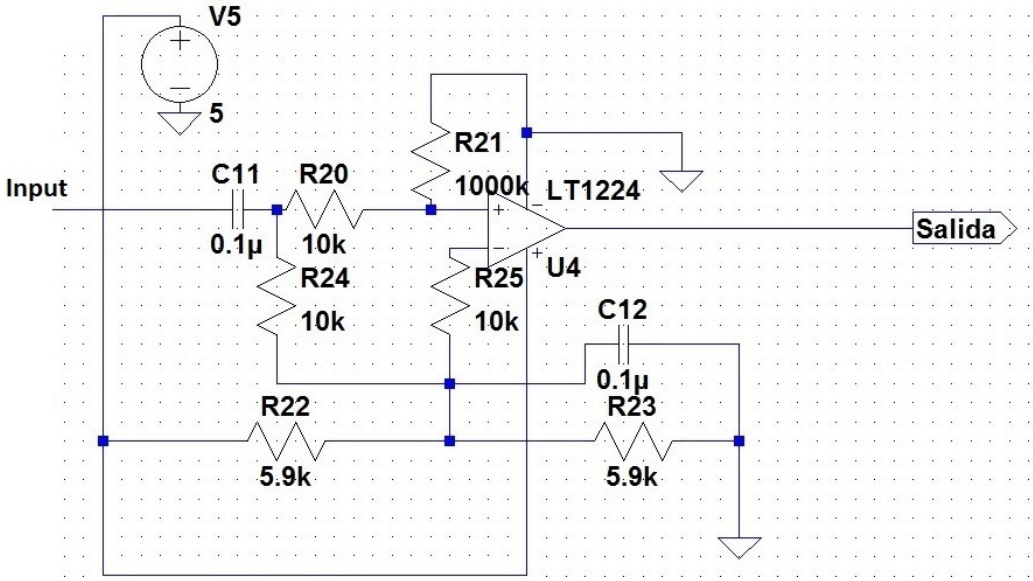


Figura 3.12: Esquemático del comparador.

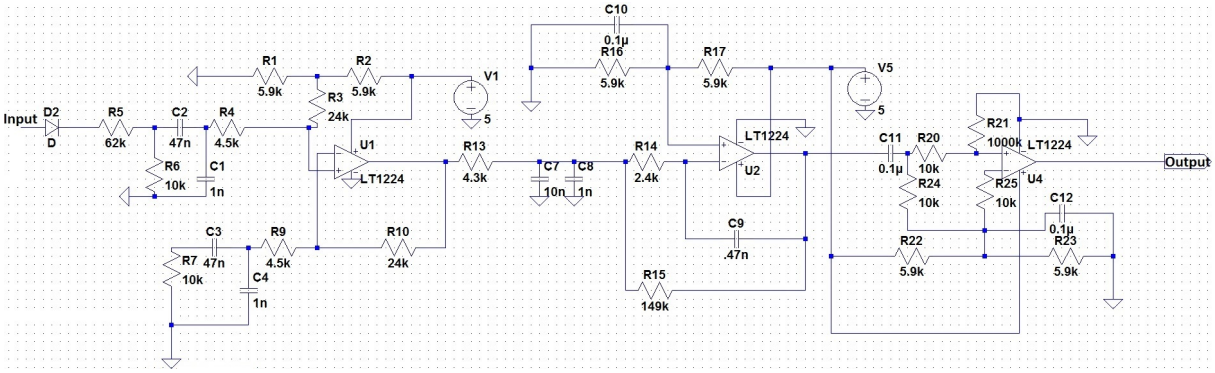
La señal que se obtiene a la salida de esta etapa es la señal resultante luego del procesado por esta etapa de comunicación, y una reconstrucción de datos adquiridos de esta señal puede verse en las figuras 3.14.

Al observar comparativamente el diseño del sistema de procesamiento de señales implementado por Freescale y el aquí descrito, puede notarse que se suprimieron algunos bloques que se consideraron innecesarios para este sistema. Esto se debe, principalmente, a una diferencia entre el objetivo del sistema propuesto en la *application note* del fabricante y el de este proyecto, ya que el sistema propuesto por Freescale apunta a demodular la comunicación recibida en un sistema de una sola bobina, mientras que el presente sistema se diseñó para un modulo de carga tipo A6, de 3 bobinas. Por esta razón, el sistema WiQi, como cualquier otro sistema de bobinado múltiple, cuenta con un diodo en la toma de señal de cada bobina (previo al capacitor de descarga a tierra), de forma de evitar una retroalimentación de la onda de potencia desde la bobina activa hacia las inactivas en el nodo de unión previo al procesamiento por parte del bloque de comunicación. Así, siendo que la señal a ser procesada por el bloque de comunicación en el presente diseño ya tiene una etapa de rectificación previa, y que la salida del comparador está en el rango tensiones de entrada del μC (definido en la sección *Controlador*, 3.2) del capítulo *Hardware* (3), sólo resultan necesarios los bloques descritos en esta sección.

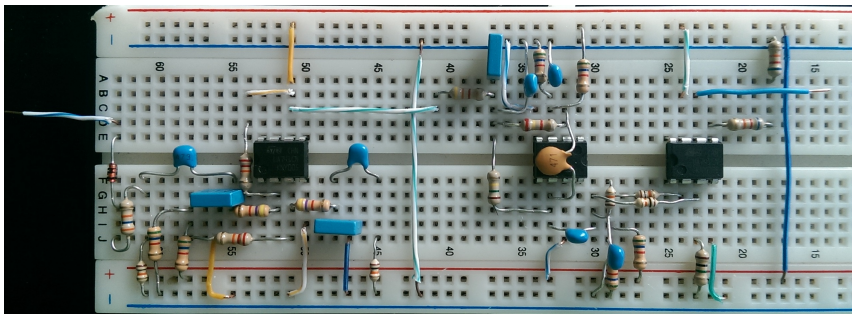
[16]Recorrido de la tensión de salida de un amplificador.

3.3.2. Pruebas y prototipo

A partir de lo anterior, y luego de realizadas las simulaciones Spice sobre el esquemático de la etapa completa mostrada en la Fig. 3.13(a), se procedió a la implementación del prototipo de esta etapa, presentado en la Fig. 3.13(b), con el objetivo de probar su funcionamiento en un proceso de carga efectivo. Para esto se utilizó un par emisor-receptor[17] *Qi compliant* adquirido para pruebas y análisis, tomando la señal a la salida de la bobina y relevando la señal resultante luego del proceso realizado por el prototipo con una adquisidora de datos.



(a) Esquemático completo de la etapa de comunicación.



(b) Prototipo de la etapa de comunicación.

Figura 3.13: Proceso de prototipado de la etapa de comunicación.

3.3.3. Resultados

Con el prototipo implementado, se procedió a constatar si efectivamente se obtenía a la salida del bloque de comunicación una señal apta para entregar al controlador, según las características explicadas al inicio de esta sección *Etapa de Comunicación* (3.3). Como puede observarse en la Fig. 3.14(a) se obtuvo una señal dentro de los límites de tensión aceptables para el controlador, y de la frecuencia esperada. Además, estudiando en detalle la señal procesada, como se ve en la Fig. 3.14(b), se puede observar que se obtienen paquetes de datos listos para entregar al controlador del sistema, con la estructura esperada (explicada en *Estructura de Paquetes* 2.2.2):

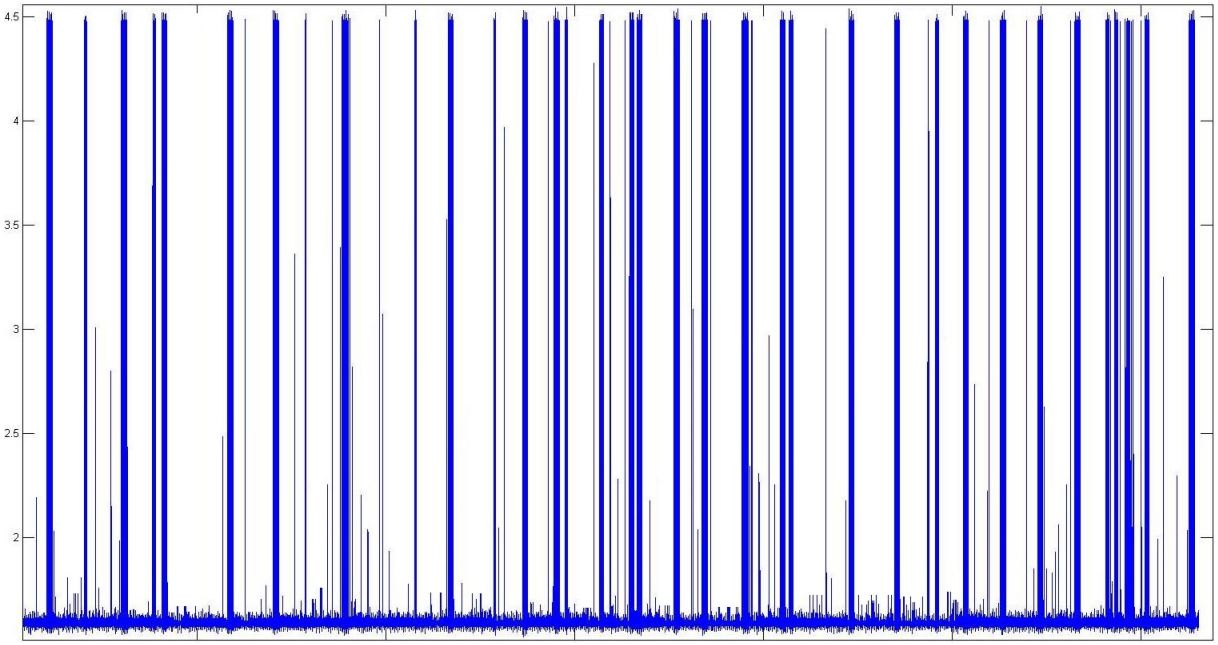
[17] Cargador tipo A11 de bobina simple, y receptor tipo carcasa para iPhone 4.

Preámbulo: 22 bits “1”.

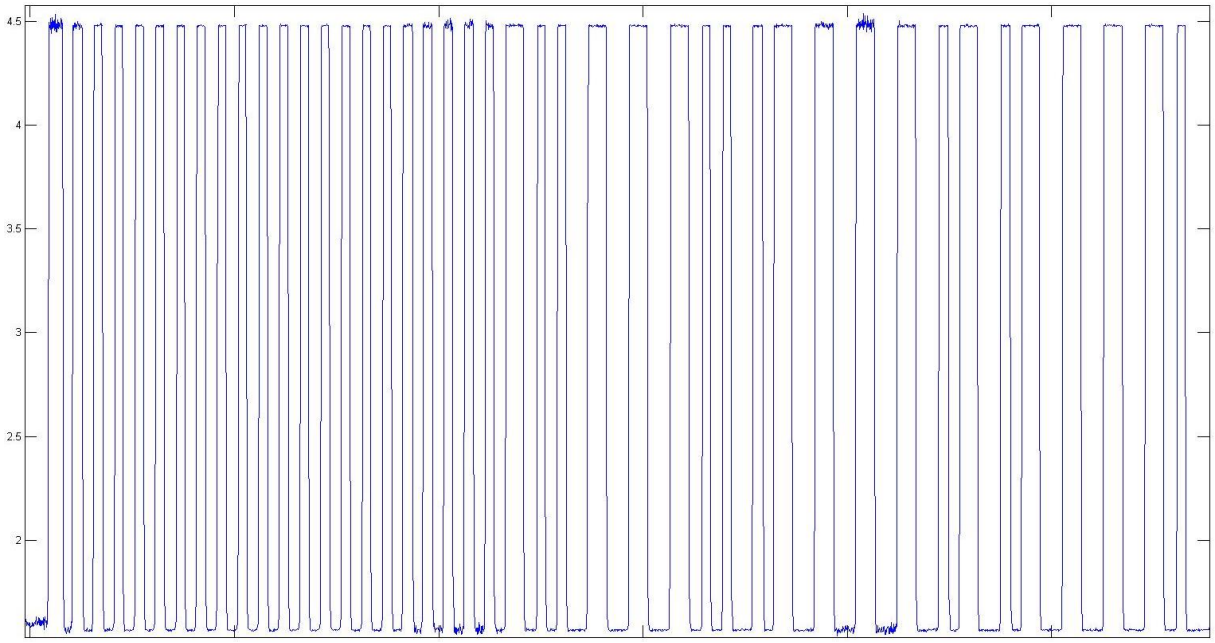
Encabezado: *Start bit*, byte 0x03, paridad “1”, *stop bit*.

Mensaje: *Start bit*, byte 0x01, paridad “0”, *stop bit*.

Checksum *Start bit*, byte 0x02, paridad “0”, *stop bit*.



(a) Datos adquiridos durante 6 segundos de relevamiento de una transferencia de potencia.



(b) Paquete de datos presente en la señal demodulada.

Figura 3.14: Graficado en MatLab de los datos adquiridos a la salida del bloque de demodulación.

Evaluados los resultados, se consideró exitoso el diseño del bloque de procesamiento de la señal de potencia que permite obtener los paquetes de datos de la comunicación. Esto se traduce en un paso más en el avance hacia el diseño Hardware final del sistema objetivo de este proyecto.

3.4. Etapas Auxiliares

Además de los tres bloques principales recientemente presentados, resultan esenciales para la implementación del sistema dos bloques secundarios con funciones bien definidas. Una de estas etapas implementa un sensado de corriente, tarea necesaria para cumplir con los requerimientos del estándar, ya que a partir de la corriente se determina si hay o no un objeto inductivo sobre las bobinas. La otra es una etapa que presta un servicio al sistema central, a los efectos de obtener una alimentación adecuada para algunos componentes a partir de la alimentación general del sistema. A continuación se presenta el bloque *Sensor de Corriente* (3.4.1), y posteriormente el bloque *StepDown DC-DC* (3.4.2).

3.4.1. Sensor de Corriente

La base en la que se apoya la utilidad de esta etapa es que la corriente consumida por un sistema de transformación en vacío es distinta a la corriente consumida si existe una carga en el secundario; y su fundamento teórico es el comportamiento en resonancia de circuitos oscilatorios forzados. A raíz de esto, y de que el sistema de transferencia de energía es (en definitiva) un transformador de núcleo de aire, la norma requiere evaluar la presencia de un posible receptor (secundario con carga acoplada) midiendo la corriente consumida por las bobinas y diferenciando dos posibles casos:

La corriente es igual a la corriente de vacío: No hay elemento inductivo apoyado sobre la superficie de carga, caso en el cual se volverá al estado de reposo hasta la siguiente evaluación de esta corriente.

La corriente es distinta a la corriente de vacío: En la superficie hay algún elemento inductivo, pudiendo ser un receptor válido (secundario con carga acoplada) o algún elemento metálico inductivo disipando energía, caso en el cual se dará paso a la etapa de identificación.

A su vez, esta etapa juega un papel fundamental en la implementación de un punto de importancia presentado en la norma: Eficiencia en la transferencia de energía, a través del control de potencia durante el proceso de carga, implementando a su vez en esta etapa de control el sistema FOD (Foreign Object Detection) para evitar inducción en objetos no deseados. Este control se lleva a cabo, como se explica en la sección 4.4.4, a través de un algoritmo Proporcional-Integral-Derivativo (PID) que utiliza el valor de la corriente consumida por las bobinas, así como la información de potencia recibida enviada por el receptor, como parámetros de cálculo.

La implementación de este bloque sensor de corriente se realizó como se muestra en la Fig. 3.15, donde la resistencia R_{SHUNT} es una resistencia de precisión de 0,33Ω y el

resto del circuito implementa dos divisores resistivos[18] conectado a un filtro pasabajos[19] para luego llevar la señal a un amplificador diferencial (con tierra como tensión de referencia para la salida). La salida de este bloque se obtiene en *I_SENSE*, presentando una ganancia de $250V/V$ (dada por amplificador diferencial en combinación con los divisores resistivos) respecto a la caída de tensión medida entre las terminales de R_{SHUNT}, y está conectada a una de las entradas analógicas del μC, donde se calcula la corriente. Para esto la señal obtenida en el pin 15 del micro ($PB1/AIN1$), es procesada con el ADC de $10bits$[20] y a partir de la ganancia del circuito, y del valor de R_{SHUNT}, estimar la corriente.

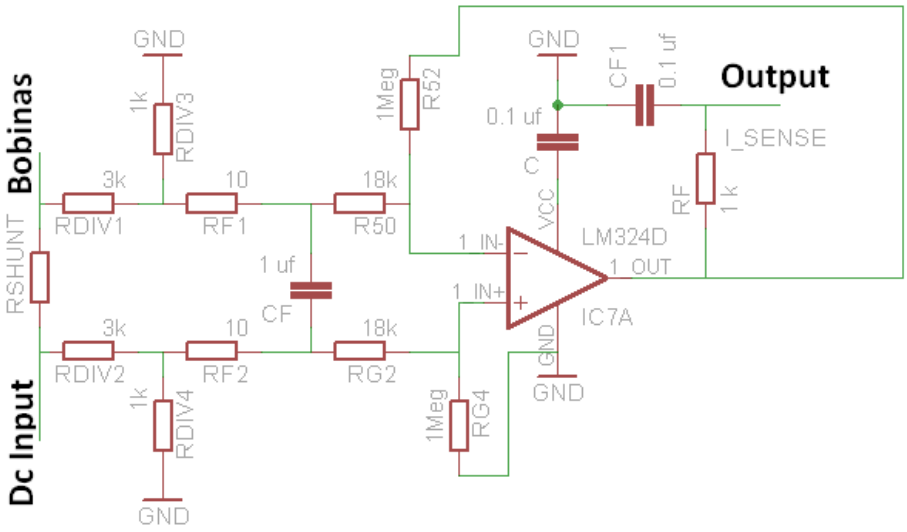


Figura 3.15: Bloque sensor de corriente.

Es de orden mencionar que el diseño de esta etapa se basó en el diseño presentado en un *Application Schematic*[7] de un chip controlador para cargadores *Qi - Tipo A6* de Texas Instruments. Algunos otros detalles, simulaciones y funciones de componentes de este bloque pueden ser consultadas en el Anexo A - Profundización Hardware, Etapas Auxiliares, Sensor de Corriente (6.1.1).

3.4.2. Conversor DC-DC

De acuerdo con lo que se define como *Etapa de Potencia* en la sección correspondiente (3.1), y siguiendo las especificaciones para los cargadores tipo A6 detalladas en la norma, donde textualmente se especifica que "La tensión de entrada al inversor half-bridge es de $12V^{\pm 5\,\%}$", la tensión de alimentación del sistema a ser implementado quedó determinada en este valor. Sin embargo, tanto los drivers para el manejo de los puentes como el μC que será utilizado como controlador del sistema tienen rangos de alimentación recomendada por debajo de esta tensión: entre 4,5 y $8V$ para el driver, y 3,3 o $5V$ para le micro. Por

[18]Este divisor tiene como objetivo reducir la tensión en modo común (12V) de forma de llevarla a niveles manejables por el Amplificador Operacional.

[19]Este filtro tiene el objetivo de evitar el ripple en la señal y quedarse con la caída de tensión promedio

[20]La efectivización de este proceso es determinada por el software que la implementa, explicado en la subsección Lectura de Corriente (4.4.2), en la sección Funciones Centrales del capítulo Software.

esto, se decidió incluir en el sistema una etapa Step Down DC-DC, que provea $5V$, a partir de los $12V$ de alimentación general, aptos para la alimentación tanto del μC como de los drivers. A estos efectos se utilizó un Step Down[21], cuyos parámetros de funcionamiento se presentan en la tabla 3.3.

Parámetro	MIN	MAX	Unidad
V_{DD} - Tensión de alimentación	3,5	28	V
V_O - Tensión de salida	0,9	5	V
I_O - Corriente a la salida	-	2	A
T_J - Temperatura de operación	-40	150	^oC

Tabla 3.3: Parámetros de funcionamiento del StepDown DC-DC utilizado.

Para la estabilización de la tensión de salida y control de la corriente provista, el bloque utilizado requiere del diseño de un circuito periférico, para el cual se detallan características y requerimientos en la hoja de datos correspondiente[8]. Por detalles sobre el diseño implementado para lograr la tensión de alimentación requerida, ver Anexo A - Profundización Hardware, Etapas Auxiliares, Diseño conversor DC-DC 6.1.2.

3.5. Versión Final

Al momento de seleccionar el bobinado tipo A6 que se presentó en la subsección Diseño (3.1.1), de la sección Etapa de potencia (3.1), las dimensiones físicas del sistema a diseñar quedaron determinadas. Con esto, y dadas las características de dicho bobinado, se decidió que el área del circuito impreso a diseñar fuera igual al área del bobinado adquirido, y es a partir de esta definición que se comenzó el proceso de diseño. A continuación se explicará el proceso de diseño del PCB final y luego se presentará el resultado obtenido.

3.5.1. Diseño

De la misma forma que se presentó el desarrollo y análisis del sistema a implementar en tres grandes bloques, siendo estos la Etapa de Potencia (3.1), la Etapa de Comunicación (3.3) y el Controlador (3.2), más dos etapas auxiliares, el diseño del circuito impreso se pensó e implementó en estos bloques o sectores.

En primera instancia, y respondiendo a que ya se contaba con la experiencia adquirida al realizar el prototipo de la Etapa de Potencia, se redujo el área del diseño con el que se contaba para el prototipo, se cambió a su vez el tipo de empaquetado de los drivers (de conectores inferiores a patas laterales) dada la dificultad que presentó la soldadura durante el prototipado, y se decidió su ubicación en el área central del circuito, dado el corto alcance de los cables del bobinado utilizado. Luego se definió la ubicación del conector de alimentación $12VDC$ en una de las esquinas de la placa, por lo que definiendo el recorrido que tendría la pista que alimenta las bobinas ya quedó definida la ubicación

[21] StepDown DC-DC Converter de Texas Instruments, TPS54231[8]

del bloque Sensor de Corriente (3.4.1). Dado que la Etapa de Comunicación utiliza 3 amplificadores operacionales (AmpOp), al igual que el bloque sensor de corriente, que un AmpOp, y que la entrada de esta etapa es tomada de las conexiones con el bobinado, se definió la ubicación de esta etapa entre la Etapa de Potencia y el bloque sensor. Con esta configuración se logró concentrar los AmpOps necesarios para la implementación del sistema en la misma área de la placa, pudiendo utilizar entonces un empaquetado[22] con la cantidad de AmpOps necesarios (4). Con este nivel de avance del diseño, sólo restaba por ubicar el Controlador y el bloque DC-DC (3.4.2), y si bien el área del PCB cubierta no era demasiada como para que se presentaran problemas a la hora de ubicar los componentes restantes, se debía resolver algunos recorridos de pistas para efectivizar conexiones necesarias. En este sentido, se resolvió ubicar el bloque DC-DC a un lado de la etapa de comunicación, y lo más cerca posible del conector de alimentación, para tener una vía de alimentación $5VDC$ que recorriera la placa hacia el controlador, pasando por los drivers de los puentes. Por último, en la zona lateral restante, se ubicó el controlador (en el centro), dos pines necesarios para su programación (en la esquina opuesta al conector de alimentación) y 4 pines conectados a patas GPIO del micro para uso opcional (debugging, por ejemplo), debido a que se tenía espacio disponible. Luego de completado el diseño, se decidió agregar 5 LED indicativos: 2 rojos (uno para verificación de alimentación $5VDC$ y otro conectado a una pata del μC configurable a tales efectos) y 3 verdes (cada uno conectado al *Enable* de cada driver, para indicar cuál está activo). En la Fig. 3.16 se muestra el layout del PCB diseñado, hallándose del lado superior izquierdo en esa figura el conector de alimentación.

En la Fig. 3.17 se muestra el esquemático completo del sistema, detallando en él los bloques descritos a lo largo de este capítulo. A su vez, se recuadra en color azul la interfaz entre la etapa de potencia y la etapa de comunicación, con la conexión necesaria para obtener la señal de cualquiera de las tres bobinas, a partir de la cual demodular la comunicación, sin causar una retroalimentación de la señal desde la bobina activa hacia las bobinas inactivas.

3.5.2. Fabricación Hardware

Dado que la experiencia con el hardware logrado en instancias del prototipado[23] no fue la mejor, al momento de la fabricación del PCB definitivo se optó por cambiar de proveedor. En tal sentido, y luego de evaluar, consultar y estudiar las distintas opciones con las que se contaba, se tomó la decisión de llevar a cabo la implementación hardware del PCB en el exterior[24]. El resultado obtenido en la fabricación de la placa por parte del proveedor seleccionado superó las expectativas, siendo una placa sólida, resistente a altas temperaturas y con una máscara antisoldante de muy buena calidad. Luego de recibir la placa, el primer paso necesario fue la separación de las placas, ya que el encargo fue realizado como un arreglo para optimizar la relación cantidad/costo. Seguidamente se procedió al proceso de ensamblado del circuito definitivo, tarea que resultó sensiblemente más fácil de lo que se preveía en base a la experiencia del ensamblado del prototipo, debido a la

[22] El empaquetado utilizado es el LM324A[9] de Texas Instruments

[23] Como se explica en el Anexo D - Contratiempos Experimentados 9

[24] Advanced Circuits - www.4pcb.com

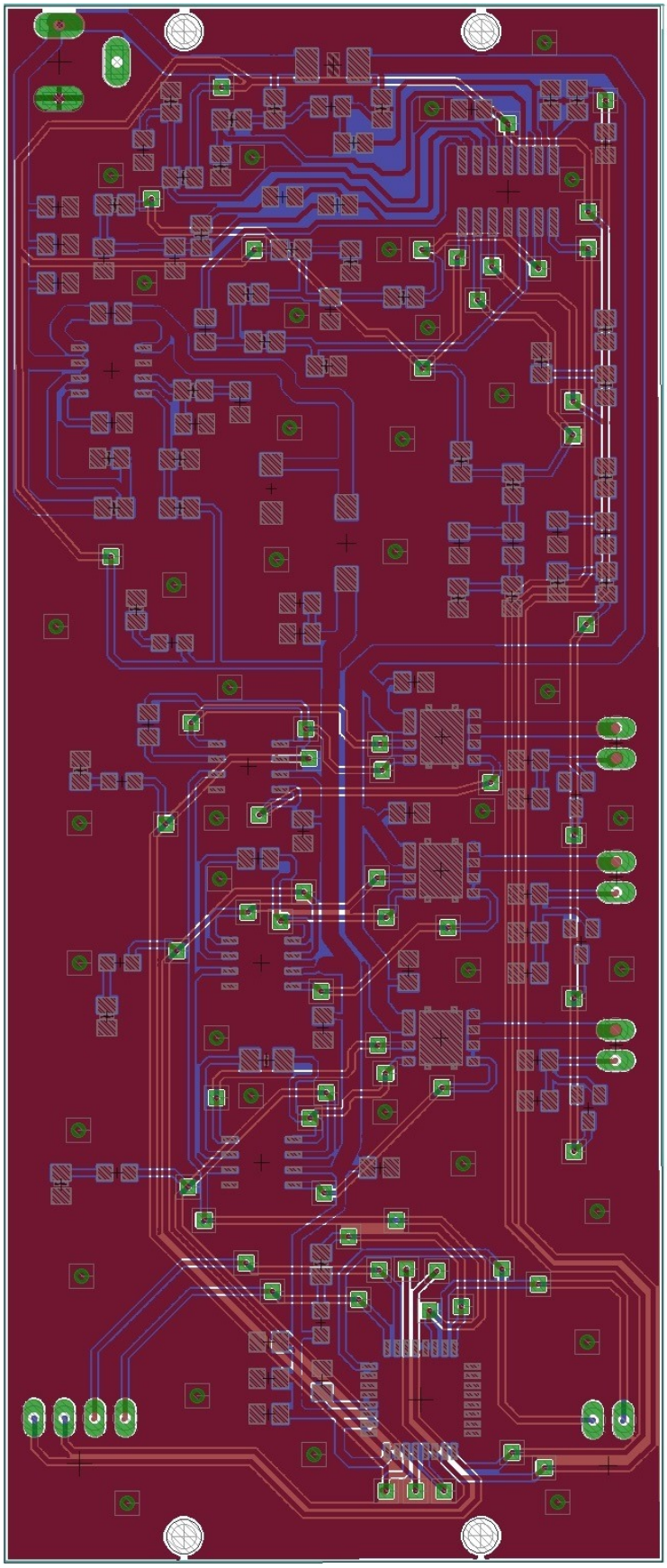


Figura 3.16: Layout del PCB del sistema.

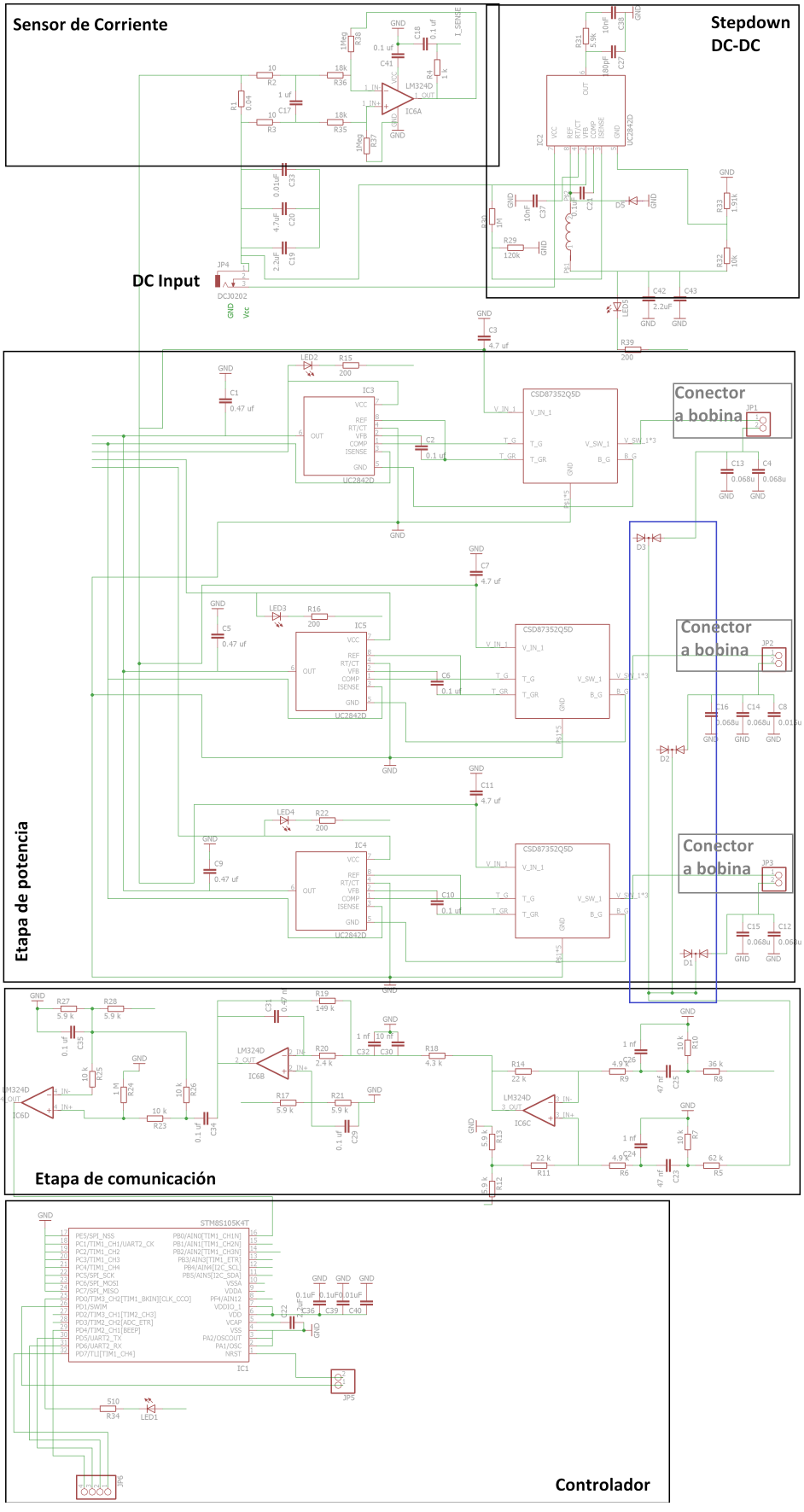
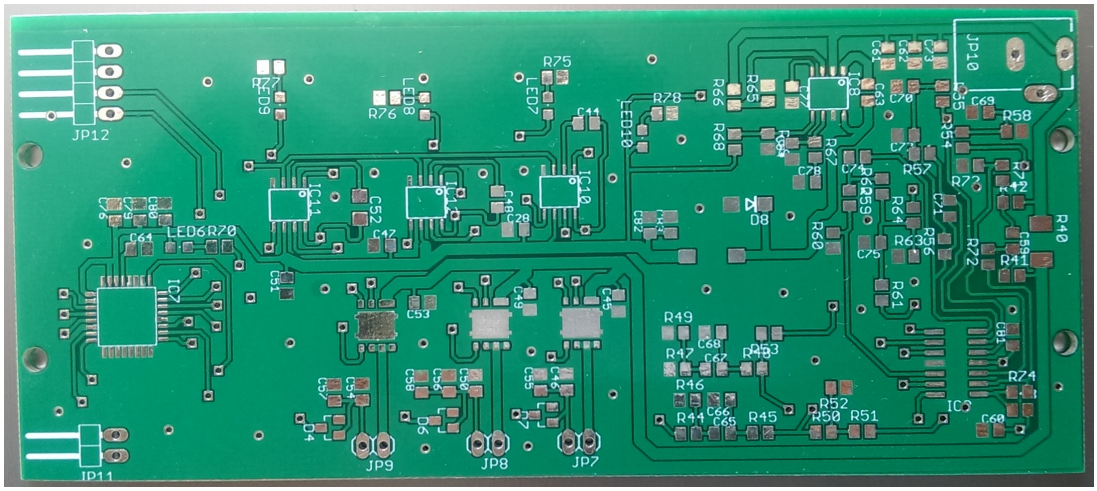
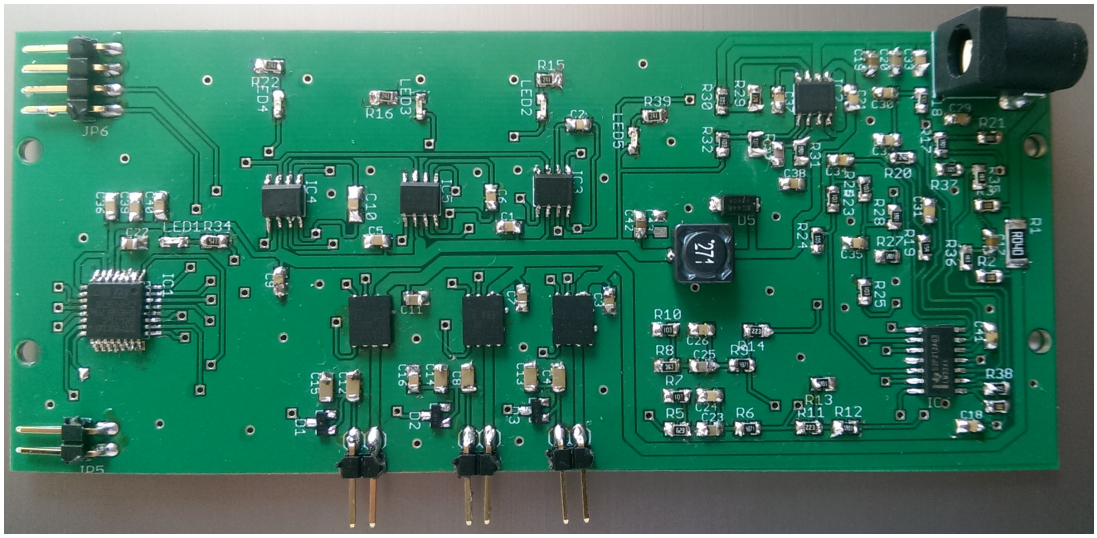


Figura 3.17: Esquemático del sistema.

buena calidad de la máscara antisoldante. Otra de las razones que facilitaron este proceso fue la posibilidad de utilizar una superficie a alta temperatura, sin que se deteriorara la placa, y donde apoyar las placas para fundir el estaño y de esta forma lograr soldar los medios puentes H[25]. En la Fig. 3.18(a) se muestra la placa construida, previo al soldado de componentes, mientras que en la Fig. 3.18(b) se puede apreciar el resultado hardware final (sin el bobinado conectado).



(a) Placa.

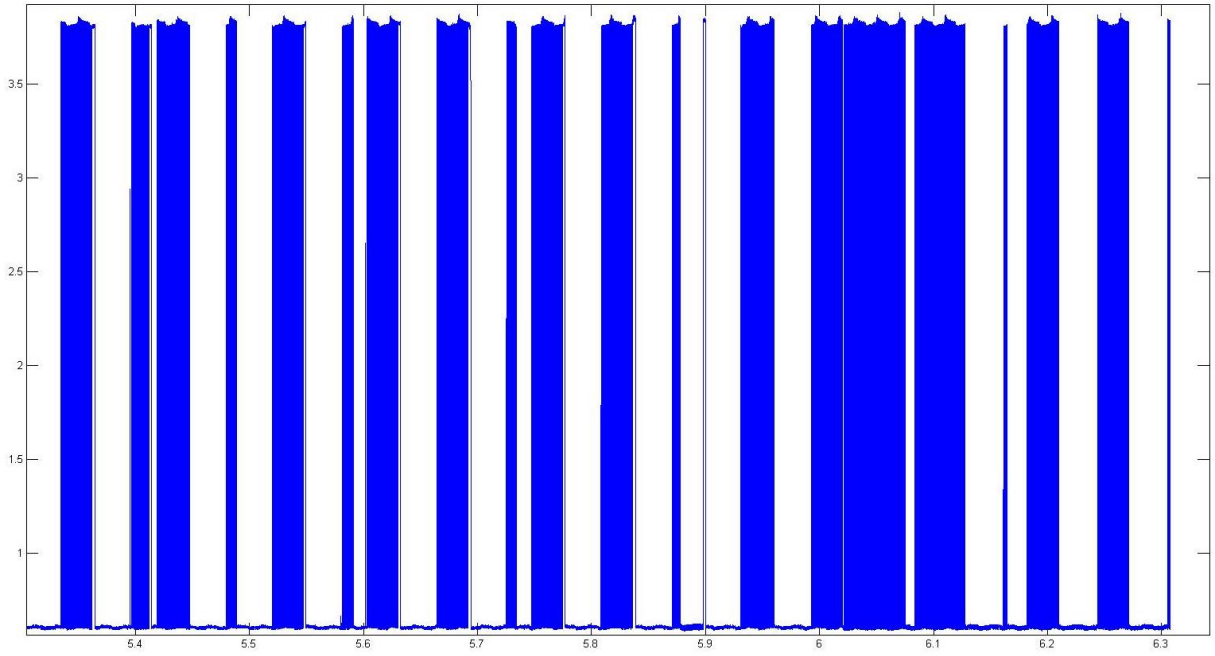


(b) Circuito.

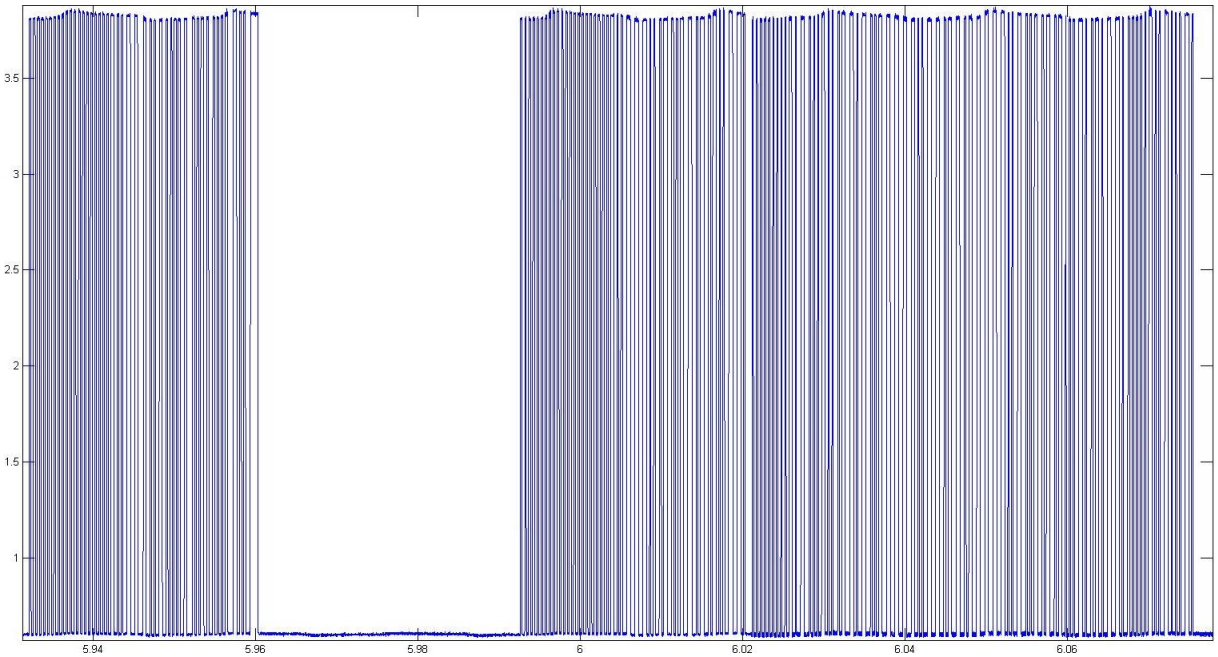
Figura 3.18: Implementación Hardware del sistema.

En este punto, cabe mencionar que el proceso de diseño y fabricación debió ser realizado 3 veces, por diferentes problemas (uno hardware y otro software, detallados en el Anexo D - Contratiempos Experimentados 9), para lograr una versión exitosa. De todas formas, incluso con las versiones previas al circuito definitivo, ya se pudo proceder a realizar los

[25]Componentes con contactos eléctricos en la cara inferior del empaquetado.



(a) Señal global.



(b) Detalle de la señal recibida.

Figura 3.19: Relevamiento de señal de comunicación en el sistema final.

relevamientos necesarios para corroborar el comportamiento físico esperado por parte del sistema, mediante la programación del controlador con un software básico que proveyera la señal PWM inicial definida en la norma (de frecuencia $175kHz$ y ciclo de trabajo $50\,\%$), y habilitara una de las vías driver-puente . En tal sentido, los principales puntos de interés fueron la señal de comunicación recibida por el μC (resultado del filtrado de la señal de potencia), relevamiento cuyo resultado se muestra en las figuras 3.19 que muestra una señal limpia, coherente y dentro de los límites esperados (3.19(a)) y paquetes de datos bien definidos al observar la señal en detalle (3.19(b)); y la medida de corriente en resonancia obtenida por el bloque sensor de corriente. Con estos comportamientos hardware corroborados, se contaba con la plataforma sobre la cual desarrollar el software del controlador del sistema a implementar, y es sobre el diseño y la implementación de esa etapa, así como los resultados obtenidos, que se profundiza en el capítulo dedicado al *Software* (4).

Capítulo 4

Software

En el presente capítulo se detallan las características del software desarrollado a los efectos de implementar el controlador del sistema, conforme a los lineamientos establecidos en el estándar *Qi*. Con este objetivo, a continuación se realiza una breve introducción a los Statecharts y al IAR VisualState (aplicación de diseño utilizada), a partir de lo cual se presentará el diseño de la estructura del software que rige la dinámica del controlador, y sobre el cual posteriormente se implementaron las funciones específicas. Luego, se desarrollarán las funciones centrales del sistema de control, dentro de las cuales se ejecutan procesos de suma importancia (como el sensado de corriente, PID o decodificación y almacenado de mensajes), así como lineamientos de la estructura del software y resultados de pruebas y debugging. Los detalles del código y dinámica del sistema controlador quedan por fuera de este capítulo, pudiendo encontrarse esta información en el Anexo B - Profundización Software (7).

4.1. StateCharts

Una de las características fundamentales de los μC es contar con una Unidad Central de Proceso (CPU, por sus sigas en inglés), Core STM8 en este caso, que es la unidad encargada de ejecutar las acciones necesarias, mediante la aplicación de alguna(s) de sus instrucciones según lo requiera el caso. En un sinfín de aplicaciones implementadas con este tipo de procesadores resulta de utilidad tener un sistema reactivo, cuyo modelo dinámico presente concurrencia o paralelismo entre estados, incluso si de hecho la ejecución de procesos fuera secuencial. Por lo tanto, haciendo uso de la alta velocidad de procesamiento que presentan los μC, en comparación con los tiempos de respuesta necesarios en un sistema reactivo a parámetros físicos, los Statecharts resultan una potente herramienta para lograr una dinámica de este tipo con un procesador secuencial.

David Harel, creador de los Statecharts, en relación a estos afirma: *"A behavioral description of a system specifies dynamic aspects of the entire system or of a particular function, including control and timing. It specifies the states and modes that the system might reside in and the transitions between them. It also describes what causes activities to start and stop, and the way the system reacts to various events. A natural technique for describing the dynamic of a system is to use a finite state machine. The described system of*

function is always in one of a finite set of states. When an event occurs, the system reacts by performing actions. The events causing the reaction are called triggers" (Texto extraído de la web de RKH - RKH Reference Manual: http://rkh-reactivesys.sourceforge.net/). Entonces, se puede afirmar que esta herramienta "Constituye un formalismo visual para describir los estados y transiciones de manera modular, permitiendo la agrupación y concurrencia, fomentando la capacidad de moverse fácilmente entre diferentes niveles de abstracción". (Fragmento extraído de las diapositivas de la charla *Máquinas de Estados UML y el framework RKH* dictada por el Ing. Leandro Francucci en la Facultad de Ingeniería, en Julio 2015). Respecto a las máquinas de estado tradicionales, los Statecharts incorporan:

- Anidamiento jerárquico de estados
- Concurrencia
- Transiciones condicionadas
- Acciones de entrada y salida de estados
- Pseudoestados

La notación y semántica definida originalmente por los Statecharts fue adoptada en la formalización UML Statecharts[1], que la transforman en una representación gráfica modularizable del sistema, facilitando su comprensión, documentación y modificación (muy útil en iteraciones de un proceso de debugging). Otra de las ventajas presentadas por el lenguaje UML reside en la formalización que se detalla para el diseño gráfico de la dinámica del sistema, a los efectos de generar, a partir de este, el código necesario para el manejo de esta dinámica en el hardware correspondiente.

El sistema a implementar, con el fin de cumplir con las especificaciones y directivas presentes en la norma *Qi*, cumple con las características ideales para ser manejado a través de Statecharts, ya que es un sistema reactivo, que presenta la necesidad de tener estados concurrentes y exigencias de tiempo aptas para ser manejado por un μC ejecutando esta dinámica. Es respondiendo a esto, a las ventajas en la representación gráfica de la dinámica del sistema y a la optimización del proceso de debugging que los Statecharts representan que se optó por estos para llevar adelante el diseño del controlador del sistema.

4.1.1. Diseño con Statecharts

En este apartado se explica brevemente el uso de statecharts tomando como referencia un diseño primario realizado para este sistema, descartado en última instancia. A tales efectos, la Fig. 4.1 muestra un diseño realizado en Statecharts, con la utilización de gran parte de las posibilidades que esta herramienta ofrece. A continuación se realiza una breve descripción de las características que este diseño se detallan:

[1]Unified Modeling Language Statecharts, siendo esta una formalización y variación basada en objetos y extendida de los Statecharts

Estados

En este tipo de diagramas se distinguen los estados **simples** y los estados **compuestos**, pudiendo ambos contener niveles de jerarquía inferiores (subestados y, por ende, submáquinas), con la característica diferencial de que los estados compuestos pueden contener más de una región, conteniendo cada una de ellas submáquinas concurrentes o, lo que es igual, de funcionamiento paralelo. Un estado compuesto es el denominado *DecoMensajes* del ejemplo, mientras que el resto de los estados allí presentes son estados simples. Características comunes a todos los estados, son las múltiples propiedades internas que estos pueden tener:

Entry: En esta definición se agrupan las acciones que se desea que se ejecuten al entrar al estado. En efecto, y hablando estrictamente de cómo se efectúan dichas acciones, esta definición sería equivalente a llamar a la ejecución de las mismas acciones en todas las transiciones que tienen como destino el estado donde se define este conjunto. Por lo tanto, para este grupo no se definen *triggers* ni guardas, ya que la condición de ejecución es la entrada al estado.

Internal: Este grupo (o grupos, ya que se pueden definir varios) es un conjunto de acciones a ejecutar mientas se esté en un estado, y en caso que se den los *triggers* y/o guardas definidas. Su existencia nace a partir de la posibilidad de que estando en un estado, se reciba un evento que no genera una transición de salida, pero que deba tener un efecto sobre el sistema y que deba entonces ser atendido.

Exit: El objetivo y funcionamiento de este grupo, es igual al del grupo *Entry*, pero relacionado a las transiciones de salida del estado.

Transiciones

Al igual que en cualquier diseño de máquinas de estado, o diagramas de flujo, las transiciones tienen un origen y un destino bien definidos. Sin embargo, los statecharts enriquecen las transiciones (en relación a este otro tipo de diseños) dotándolas de guardas (condicionales) y acciones asociadas, además de los *triggers*. Por lo tanto, cada transición tiene las siguientes características asociadas:

Disparador o *trigger*: Los disparadores para las transiciones son eventos (pudiéndose definir un conjunto de eventos posibles) o señales (eventos particulares generados por otra máquina del sistema). Si bien su definición para una transición no es obligatoria (existen transiciones *triggerless*), deberá existir un disparador o una guarda, o una combinación de ambos asociada a cada transición.

Guarda: Esta característica define condicionales para una transición, y se define con condiciones booleanas a partir de variables del sistema. Al igual que los disparadores, su existencia es complementaria pero no obligatoria en cada transición.

Acciones: Las acciones de transición definen funciones asociadas a dichas transiciones, con un funcionamiento análogo a las definiciones de *Entry* y *Exit*, como se explicó anteriormente, es decir, se ejecutan a la salida del estado origen o entrada al estado destino. Su existencia no es obligatoria para la definición de una transición.

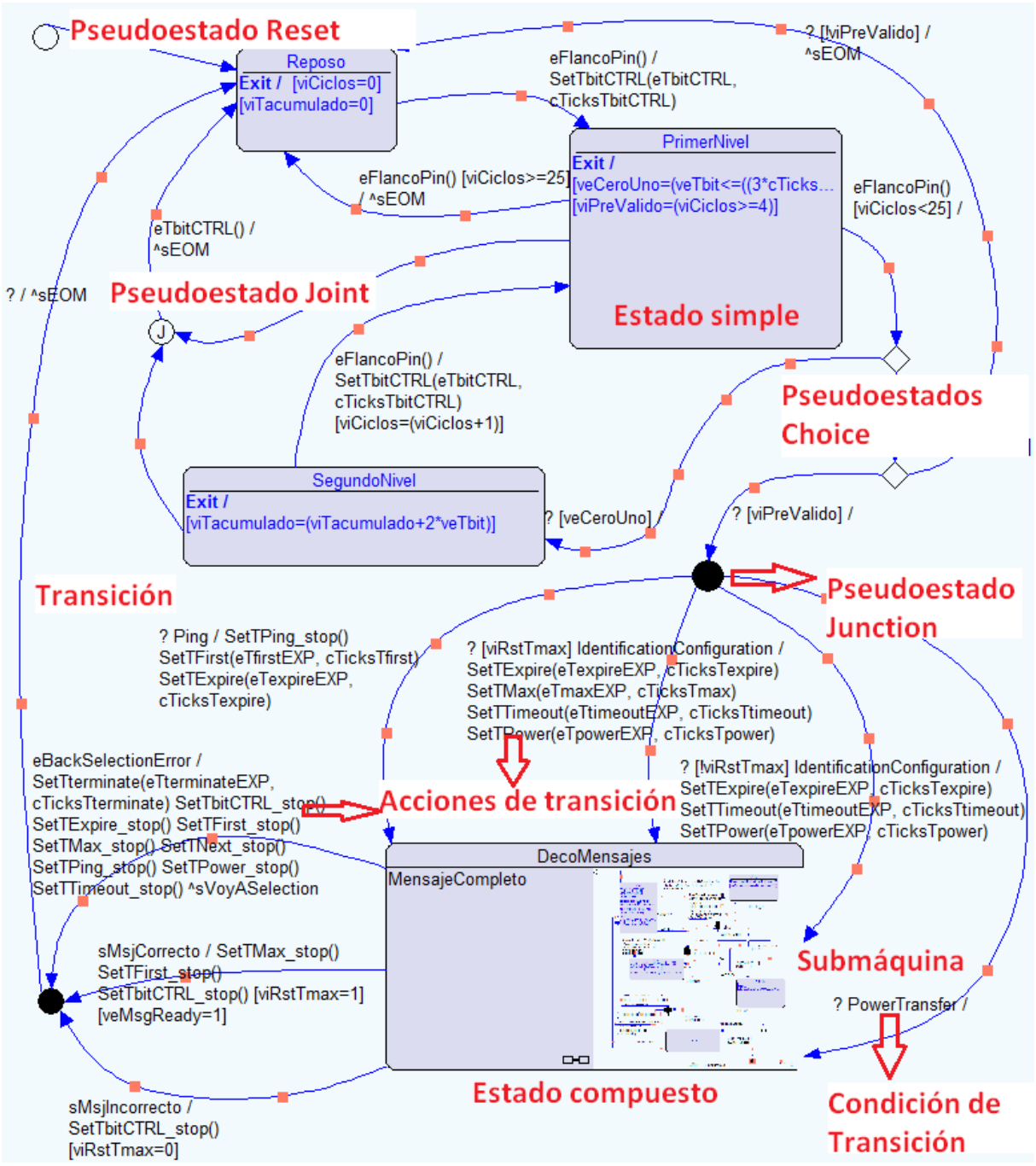


Figura 4.1: Estructura de diagramas Statecharts.

Pseudoestados

Estos estados especiales tiene el objetivo de enriquecer las transiciones y la dinámica de las máquinas diseñadas. Con estos pseudosestados es posible simplificar el entendimiento y la descripción del diseño aún utilizando los procesos de transición con mayor complejidad, ya que (como se observa en la Fig. 4.1) con algunos de ellos es posible unificar varias transiciones y sustituirlas por una transición simple, en serie con varias transiciones condicionales. Algunos de estos pseudoestados son:

Reset: Este pseudoestado tiene la simple y fundamental función de indicar estado inicial de cada máquina del diseño, sin importar el nivel de jerarquía. Por lo tanto, cada diseño tendrá tantos pseudoestados reset como máquinas estado tenga.

Joint: Este pseudoestado tiene como objetivo unificar las transiciones con el mismo destino e iguales características, pero distinto origen. En el ejemplo, puede observarse que dos transiciones parten de los estados *PrimerNivel* y *SegundoNivel*, sin disparador aparente, y se unen en este estado, dando paso a una transición con evento disparador y acción asociada (en este caso, encolar una señal). Esta combinación sería equivalente a dos transiciones simples, partiendo de los estado origen respectivos, y llegando al estado *Reposo*.

Choice: Su función es bifurcar transiciones, evaluando condiciones dadas para hacerlo. Su particularidad, que la diferencia de los estados *Junction*, es la de requerir completitud en sus condiciones de salida, dado que las condiciones se evalúan **en el estado** luego de la transición de llegada, y en caso de no cumplirse ninguna de las condiciones de salida la máquina quedaría estancada. Para asegurar esto se recomienda el uso de la condición *else*, de forma de tener al menos una ruta posible asegurada. Dada esta forma de evaluación de parámetros, estos estados resultan de mucha utilidad a la hora de tomar decisiones en función de la evaluación de condiciones que se establecen durante la transición de llegada.

Junction o fork: La diferencia fundamental de estos estados respecto de los *Choice* es la evaluación de las condiciones de transición. En este caso, las transiciones se evalúan **antes de dejar el estado origen** de la transición, por lo que si no se cumple ninguna de ellas no se dará la transición. Estos pseudoestados representan, entonces, una simplificación en la complejidad del diagrama, ya que al igual que en el caso de los estados *Joint*, la configuración con estos pseudoestados podría ser sustituida con un conjunto de transiciones, en esta caso con una compleja estructura de disparadores y guardas.

Un dato relevante respecto a estos diseños, que se mencionó al explicar las transiciones, es que el manejo de la dinámica para estos diseño está orientado a la respuesta frente a eventos recibidos por el sistema. Por esta razón, en una implementación software para un diseño de este tipo se deberá estructurar un sistema de manejo de eventos, con el objetivo de derivar al sistema estas excitaciones de forma ordenada y completa, es decir sin pérdida de eventos a causa de que el sistema esté ocupado al momento en que estos ocurren.

4.1.2. VisualState

Como se mencionó con anterioridad, la experiencia en el uso de la aplicación IAR VisualState para la utilización de Statecharts la posicionó como la mejor opción respecto al medio a utilizar para llevar adelante el diseño del controlador. No obstante, esta herramienta no presenta una versión gratuita apta para las dimensiones del sistema a diseñar y el tiempo que la realización del mismo requirió, ya que las versiones de licencia gratuita limitan o en cantidad de tiempo de uso (30 días) o en tamaño del proyecto a realizar(30 estados como máximo). De todas formas, no se contaba con otra opción posible[2], por lo que para la utilización de este software se tuvo que trabajar en la versión limitada por cantidad de estados, hasta que el proyecto alcanzó una dimensión superior a la permitida por esta versión, trabajando luego en distintos equipos adquiriendo licencias por tiempo.

La importancia de esta herramienta para el proceso de diseño del sistema fue notoria, ya que la representación gráfica de las transiciones de estados, con la definición de los eventos o señales[3] facilita sensiblemente este proceso. Sin embargo, el alcance que esta herramienta tiene es acotado, ya que sólo pueden ser escritas funciones básicas en lenguaje C tanto en las transiciones como en los estados, entregando sólo la definición de funciones más complejas, como las funciones de timers, en el código generado. Por esta razón, las funciones deberán ser agregadas a las librerías del proyecto general luego de haber generado el código con la aplicación. Otra característica que presenta este software es su generalidad respecto a la plataforma que se utilizará, por lo que tampoco se puede implementar la configuración HW del micro en la etapa de diseño. De todas formas, esta característica dota de una portabilidad muy conveniente al código generado por este software, cosa que se aprovechó al momento de necesitar cambiar la plataforma del sistema en primera instancia elegida (más detalles a este respecto a se presentan en el Anexo C - Contratiempos Experimentados, 9).

4.2. Diseño

En los capítulos 5 (*System Control*) y 6 (*Communications Interface*) del estándar se establecen las directivas que un sistema *Qi* compatible debe satisfacer, en lo que a lógica de funcionamiento, orden de acciones, temporización, codificación de comunicaciones e interpretación de mensajes respecta. Por esto, y dados los objetivos de este proyecto, es que el software del controlador del sistema se diseñó obedeciendo estas directivas, resultando, como se mencionó en la sección previa a los efectos de describir el entorno de trabajo y tal como se muestra en la Fig. 4.2, en el diseño de un sistema controlador (*SisCtrl*) dividido en dos estados paralelos, básicos: *CommBlock* y *CtrlBlock*. Estos bloques tienen funcionalidades complementarias respecto al Sistema General, que se detallan en esta sección.

[2]Debido al tiempo que implicaría el aprendizaje del manejo de una nueva herramienta, y que cada software utiliza una versión modificada del lenguaje UML, no siguiendo las especificaciones exactamente.

[3]Eventos autogenerados por el sistema para influir en otra máquina de estados del sistema

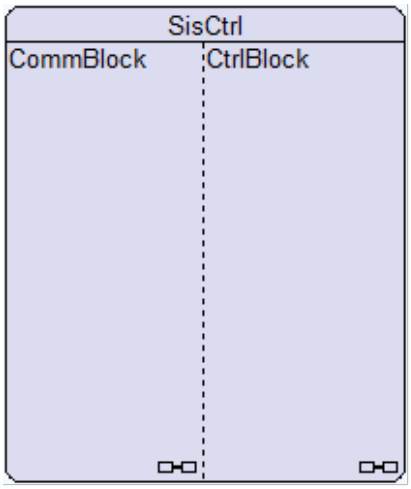


Figura 4.2: Máquina *SisCtrl* con dos regiones paralelas, Top Level del sistema.

4.2.1. CtrlBlock

El objetivo funcional del bloque de control es lograr que el transmisor monitoree la superficie de carga periódicamente en busca de un posible receptor[4] para, habiéndolo encontrado, proceder a la identificación de un receptor válido del cual se reciba la información suficiente, con el fin de establecer algunos parámetros, y poder dar inicio al proceso de carga. Este bloque, como puede verse en la Fig. 4.4, implementa directamente la dinámica de fases definida en el apartado *System Control* del estándar (mostrada en la Fig. 4.3) y controla el funcionamiento global del sistema, definiendo estados según las acciones que esté llevando a cabo el transmisor, los cuales se desarrollan a continuación.

Descripciones detalladas del diseño de *CtrlBlock* y todos sus estados y submáquinas, incluyendo variables utilizadas, condiciones de transición, descripción de estados y sus acciones vinculadas pueden consultarse en el Anexo B - Profundización Software, Diseño de Statecharts, CtrlBlock (7.1.1).

Selection

El estado *Selection* está relacionado con el monitoreo periódico de la superficie en busca de un receptor y, para el caso particular del sistema *WiQi*, se implementa el recorrido por las 3 bobinas del sistema para cubrir toda el área de la superficie de carga en este relevamiento. La implementación de este comportamiento se hizo efectiva mediante un estado compuesto por dos subestados concurrentes, con objetivos bien diferenciados:

Selection_Ctrl:

Esta submáquina, mostrada en la Fig. 4.5(a), es la que efectivamente lleva a cabo el monitoreo a través de la bobina activa (en todo momento se tiene una, o ninguna, bobina activa en el sistema) mediante el método de cambio de resonancia[5]. A estos

[4] A los efectos de no realizar una continua e ineficiente emisión de potencia.

[5] Definido en el anexo B del estándar [1], y basado en los fundamentos explicados en 1.1.3.

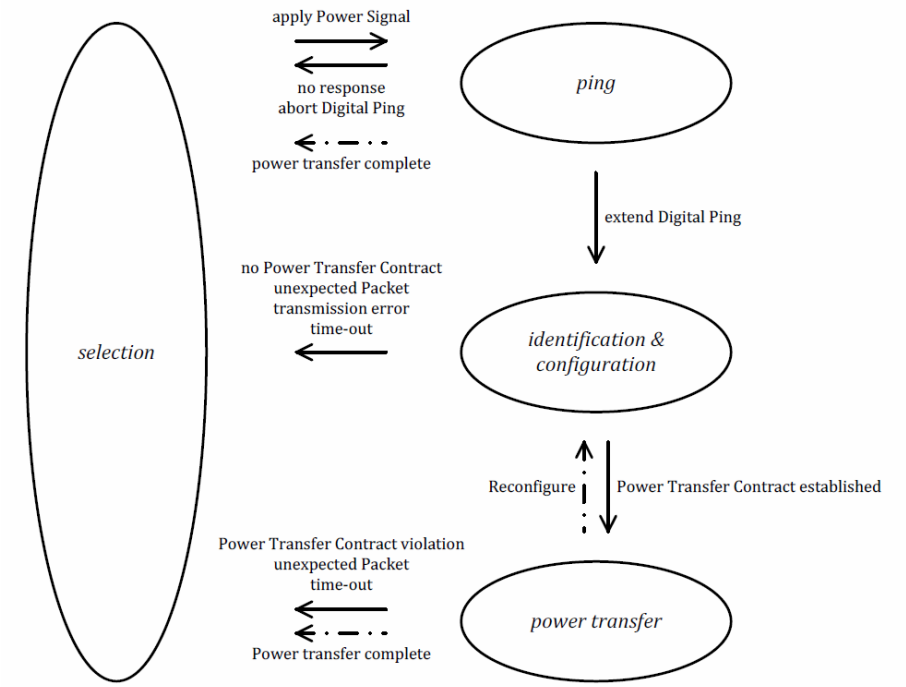


Figura 4.3: Diagrama de fases definido en el estándar.

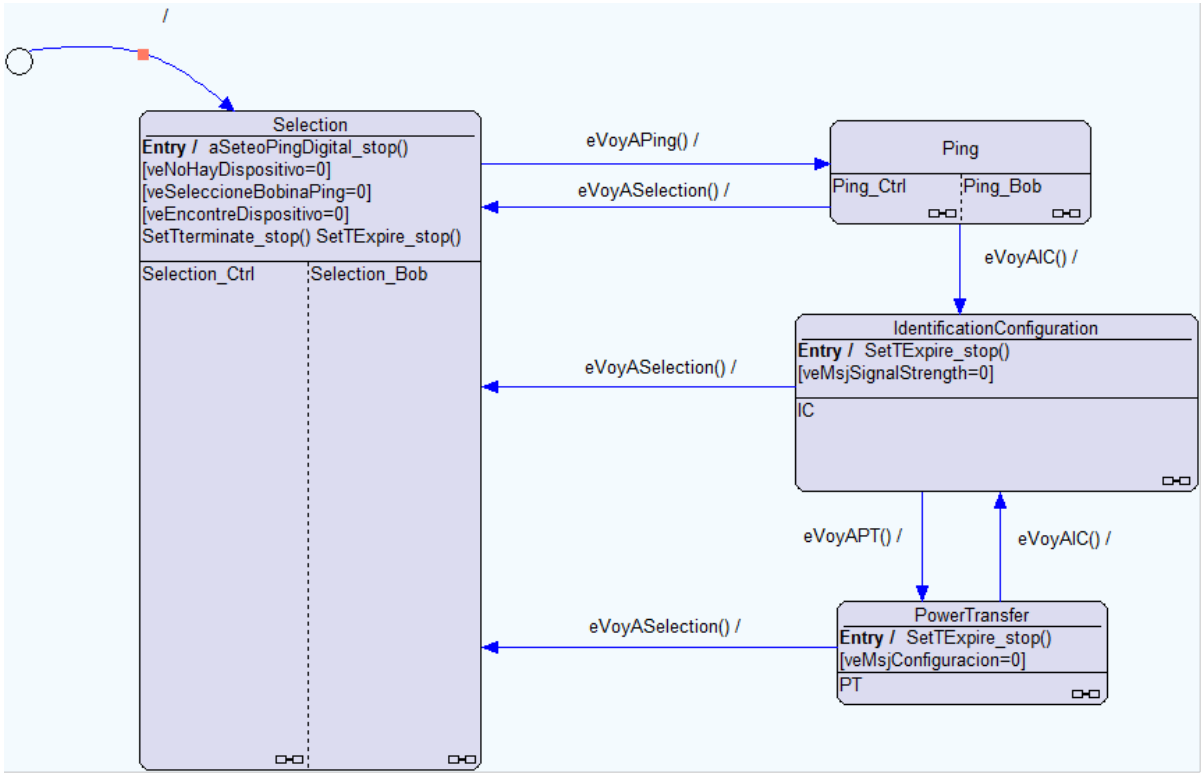


Figura 4.4: Máquina del control global, *CtrlBlock*.

efectos, envía un pulso de corta duración a la bobina, a frecuencia de resonancia, midiendo al mismo tiempo la corriente consumida. En caso de que la corriente sensada sea menor a la corriente esperada (corriente de resonancia), se asume la presencia de un receptor y se da paso al estado *Ping*; si por el contrario, la corriente es la de resonancia, se asume que la bobina está libre y por lo tanto no hay presencia de receptor, dando por finalizado el monitoreo de la bobina hasta el próximo período de evaluación.

Selection_Bob:

La función de esta submáquina, complementaria a *Selection_Ctrl*, es realizar el cambio de la bobina activa para lograr el barrido de toda la superficie de carga verificando las tres bobinas, de a una por vez. Este cambio se hace efectivo al recibir una señal, desde la máquina *Selection_Ctrl*, que indica que se finalizó el monitoreo de la bobina activa y se debe pasar a la siguiente bobina. En la Fig. 4.5(b) se puede ver el diagrama de esta máquina.

Con estas submáquinas, y los temporizadores que en ellas se definen, se obtiene el bloque *Selection* para realizar el mencionado barrido de toda la superficie en busca de un receptor de forma periódica. Este comportamiento configura el modo *stand-by* del sistema y tiene como objetivo la optimización su consumo energético, dada la reducción del gasto de energía que significa en comparación con la simple emisión constante de potencia, haya o no un receptor en la superficie de carga.

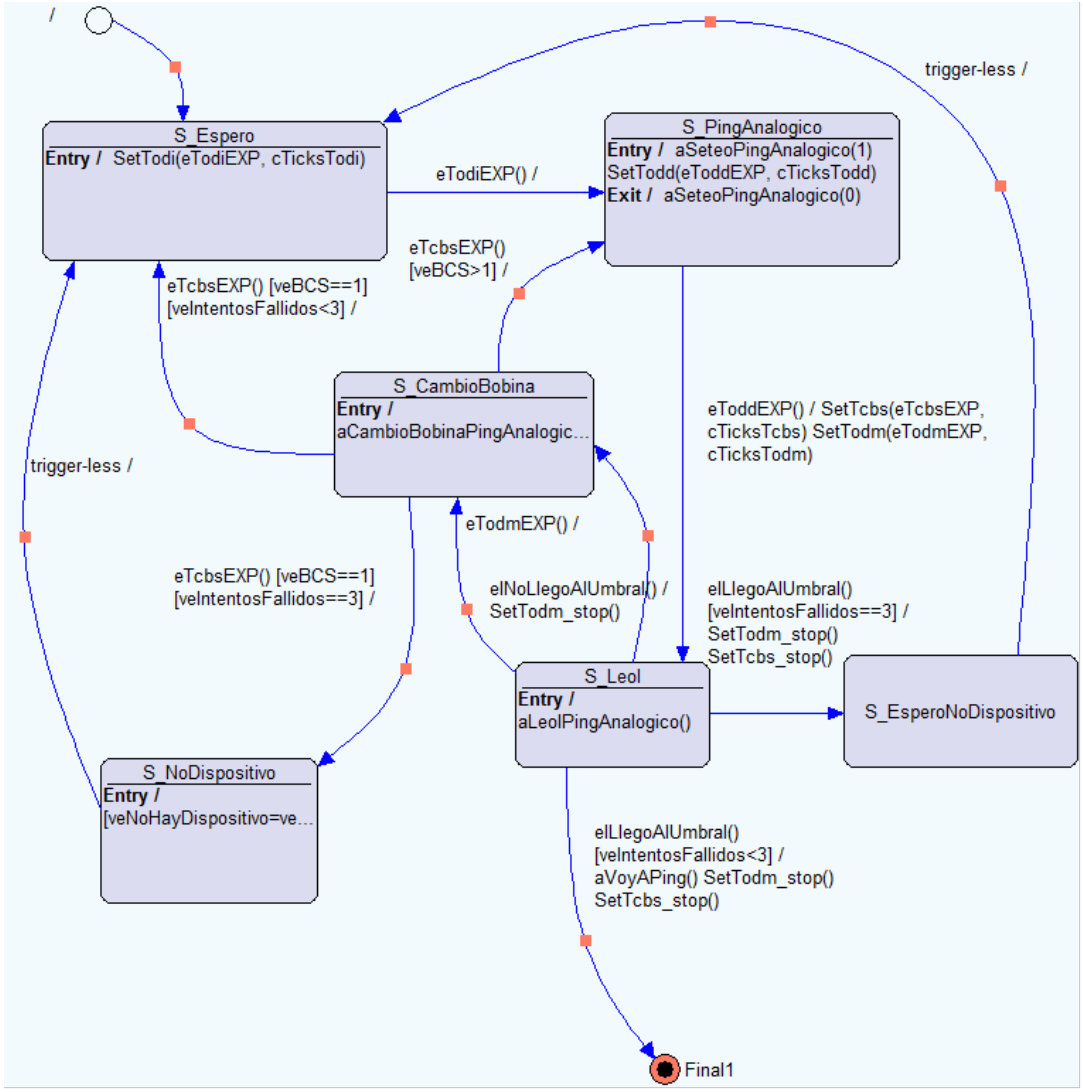
Ping

El estado *Ping* está vinculado con el inicio de comunicación con el receptor y es a través de dicha comunicación que se determina el nivel de acoplamiento de las bobinas. Los objetivos primordiales de esta etapa son identificar un receptor válido, madiante de la recepción de mensajes modulados en la onda de potencia, y seleccionar la bobina mejor acoplada con el fin de realizar la transferencia de potencia más eficiente posible. Al igual que *Selection*, este estado es un estado compuesto por dos submáquinas concurrentes, con objetivos específicos:

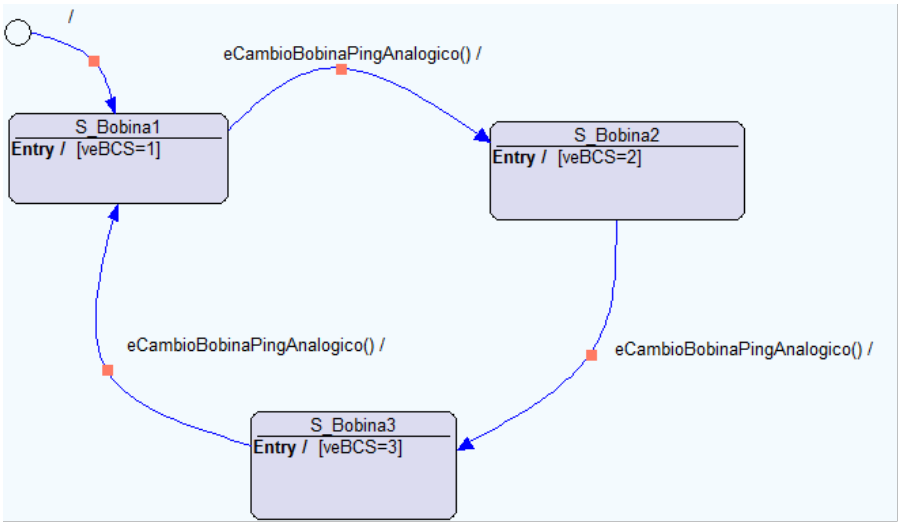
Ping_Ctrl:

Esta submáquina tiene la tarea primaria de excitar la bobina activa con una onda de potencia de $175kHz$[6] a los efectos de proporcionar el canal necesario para que el receptor pueda comunicarse. Luego de establecida la onda de potencia, sobre la cual el receptor podrá modular la señal de comunicación, se espera un tiempo determinado (t_{ping}) a la espera del comienzo de preámbulo de un paquete de datos. En caso de que transcurra t_{ping}, y no se haya recibido un flanco en el pin asignado para recibir la señal de comunicación, se quita la señal de excitación y se asume que no hay un receptor válido para esta bobina. Estos mismos pasos se siguen en los casos que se reciba un paquete que no sea de *Signal Strength*, o si se recibe

[6]Esto lo lleva a cabo el controlador, estableciendo la señal *enable* en el driver correspondiente a la bobina activa, y excitándolo con una onda PWM de $175kHz$ y ciclo de trabajo 50 %



(a) Diagrama StateChart de la Submáquina Selection_Ctrl para el monitoreo de bobinas.



(b) Diagrama StateChart de Selection_Bob para el barrido de superficie.

Figura 4.5: Detalle de las submáquinas del bloque Selection.

un paquete de *End Power Transfer*. En caso de recibir exitosamente un paquete indicador de acople de bobina (*Signal Strength*) se almacena esta información para proseguir con el proceso. El diagrama Statechart de esta máquina se muestra en la Fig. 4.6.

Ping_Bob:

Al igual que *Selection_Bob*, esta submáquina tiene la tarea de realizar el cambio de la bobina activa para lograr el recorrido de las tres bobinas en el proceso de *Ping*, trabajando conjuntamente con *Ping_Ctrl*. En otras palabras, esta máquina hace efectivo el cambio de bobina activa al recibir una señal desde la máquina *Ping_Ctrl*, que indica que se completó el ping de esa bobina y se debe cambiar a la siguiente. El diagrama Statechart que detalla la implementación de esta máquina puede observarse en la Fig. 4.7.

Luego de realizado el ping en las tres bobinas, se procede a una transición de estado dependiendo de la comunicación recibida: en caso de no recibir ningún paquete o recibir un paquete que no sea *Signal Strength*, se apaga la señal de potencia y se devuelve al sistema al estado *Selection*; en caso de recibir al menos un paquete *Signal Strength* se da la transición de estado hacia *Identification & Configuration*, reestableciendo la señal de potencia de $175kHz$ en la bobina con mejor acople (paquete *Signal Strength* de mayor valor). Cabe mencionar que cada vez que aquí se habla de recepción de mensajes, se está haciendo referencia implícita al otro gran bloque del diseño, *CommBlock*, ya que es esa máquina la encargada de la recepción de mensajes, su almacenamiento y el aviso a *Ctrl-Block* de que hay un mensaje válido en un lugar de memoria preestablecido.

Identification & Configuration

Este estado está vinculado a la identificación del receptor, la recepción e interpretación de información para la configuración del sistema y, en base a dicha información, el establecimiento de los parámetros necesarios para la transferencia de potencia. A diferencia de los estados anteriores, este estado es un estado simple (se compone de una sólo máquina de estados) llamado *IC*, y la máquina se muestra en la Fig. 4.8. Con el fin de llevar a cabo su función, esta máquina espera la recepción de mensajes (a través de CommBlock) y al recibirlos verifica si cada mensaje cumple ciertas reglas de orden y tipo. En caso de recibir un mensaje no esperado, en un orden no correspondiente, o de que expire alguno de los tiempos de espera, se devuelve el sistema al estado *Selection*, apagando la señal de potencia. En caso de recibir la secuencia de mensajes que cumplen con las reglas esperadas (secuencia que finaliza con el mensaje *Configuration*) se utiliza la información recibida para establecer un Contrato de Transferencia de Potencia, donde se establecen límites para ciertos parámetros de la transferencia de potencia. Luego de establecido el Contrato de Transferencia de Potencia, se da la transición hacia el estado *Power Transfer*.

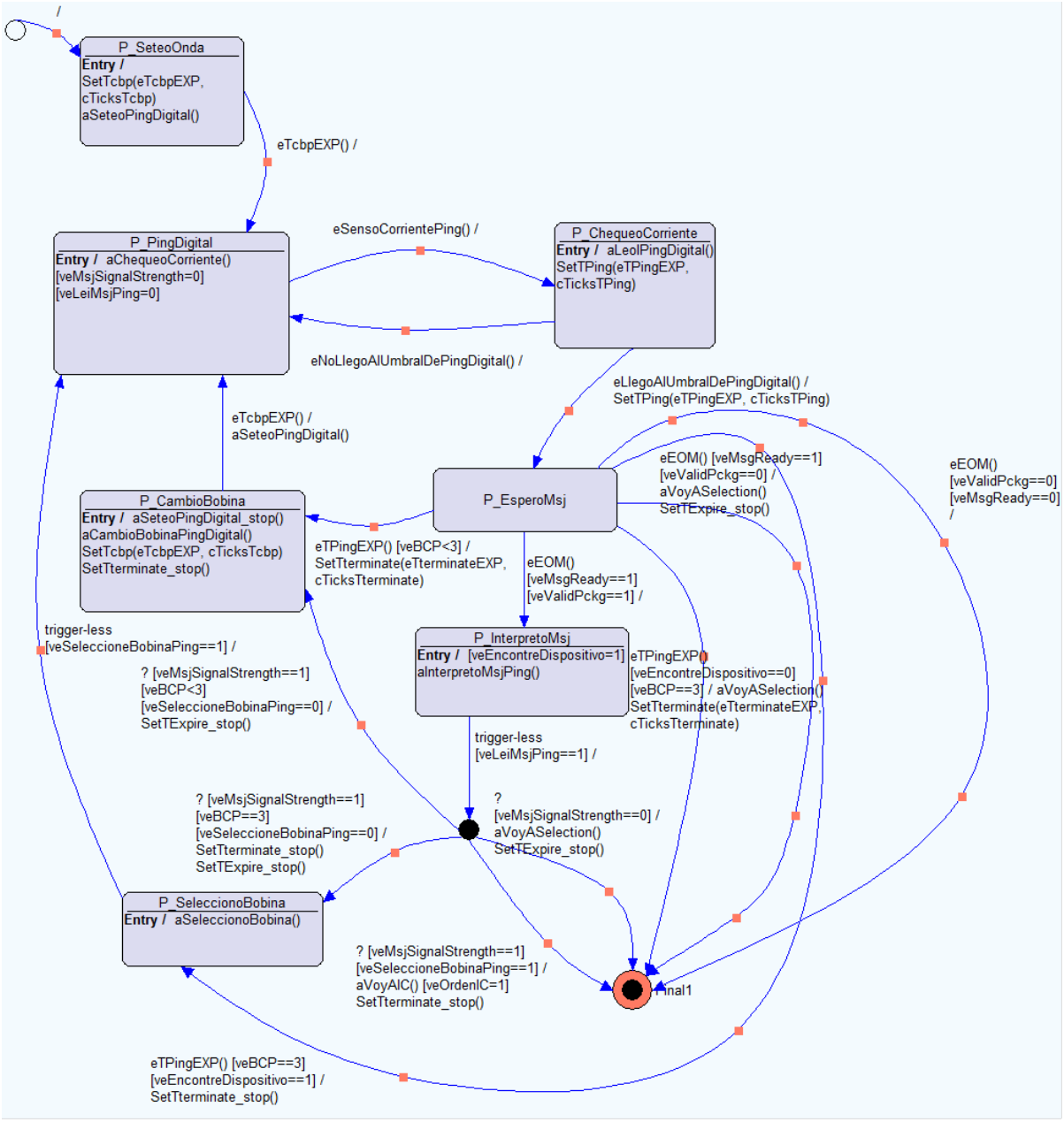


Figura 4.6: Diagrama StateChart de la submáquina Ping_Ctrl para el ping de bobinas.

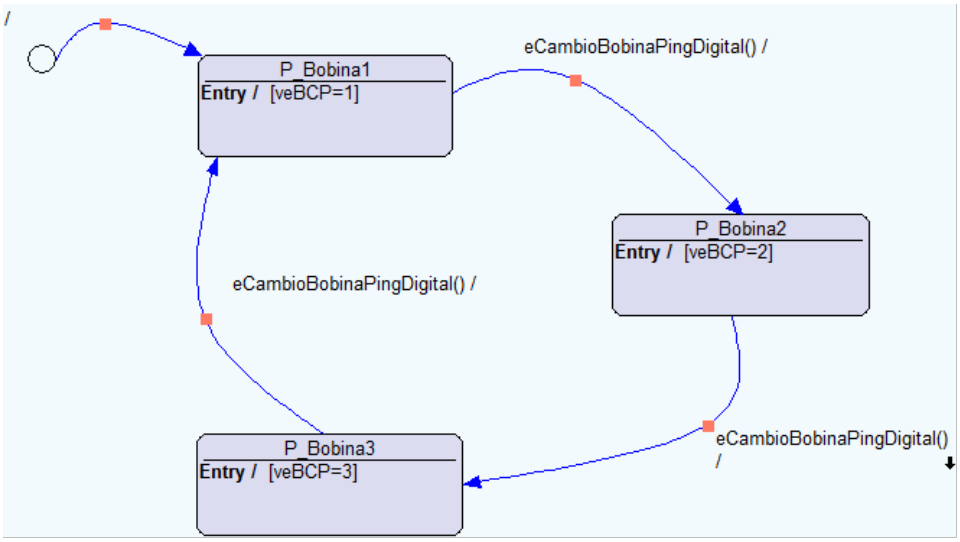


Figura 4.7: Diagrama StateChart de Ping_Bob para el barrido de superficie.

Power Transfer

Este es el estado destinado a la trasferencia de potencia, al correspondiente control y sus posibles reajustes de la potencia transferida, y al cumplimiento del Contrato de Transferencia de Potencia. Su implementación consta de una sóla máquina y su diseño puede verse en la Fig. 4.9. El objetivo del control de la potencia transferida se alcanza mediante la recepción de mensajes *Control Error*, que indican el ajuste en el punto de funcionamiento de la transferencia de potencia que el receptor solicita. A partir de este mensaje el controlador deberá modificar la frecuencia o el ciclo de trabajo, según corresponda, de la onda PWM de manejo de la señal de potencia. Este ajuste implementa un bloque *PID*[7], que toma como parámetros de ajuste el valor recibido en el paquete *Control Error* y los sucesivos valores de corriente sensados en cada iteración del proceso *PID*. Además, en todo momento se evalúan los parámetros necesarios para la verificación del cumplimiento del Contrato de Transferencia de Potencia, y en caso de una violación de dicho contrato se devuelve al sistema al estado *Selection*. En caso de recibir un mensaje de *End Power Transfer* de un valor específico (0x07) se deberá devolver al sistema al estado *Identification & Configuration*, para renegociar el Contrato de Transferencia de Potencia, o finalizar la transferencia de potencia y devolver el sistema al estado *Selection* en caso de recibir cualquier otro valor en un mensaje *End Power Transfer*. Esta última acción también deberá ser realizada en caso de que transcurra un tiempo predeterminado (t_{power}) y no se haya recibido comunicación de parte del receptor, asumiendo que se lo ha retirado de la superficie de carga.

[7]Mecanismo de control realimentado, Proportional Integral Derivative.

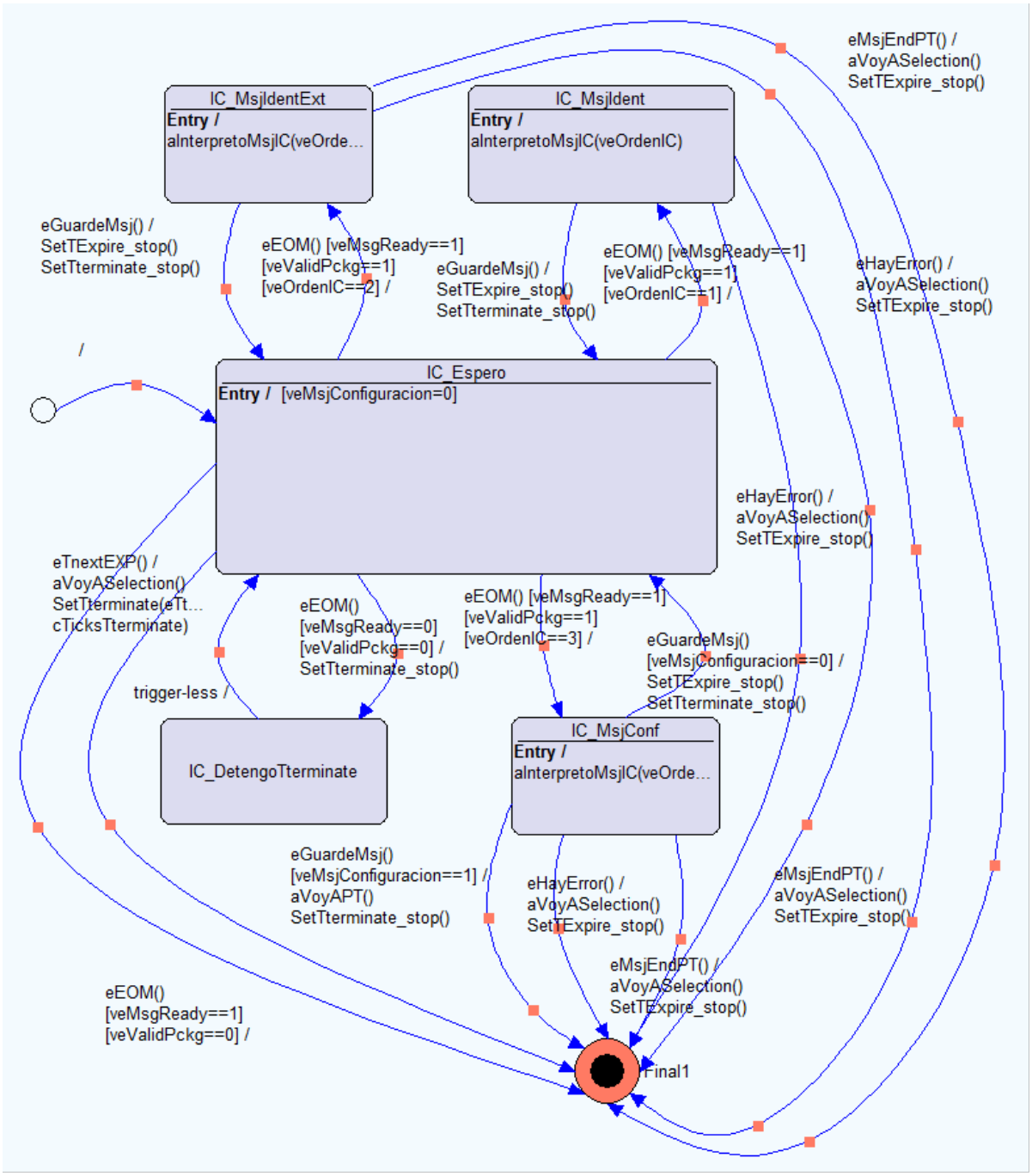


Figura 4.8: Diagrama Statechart de la máquina *IC*.

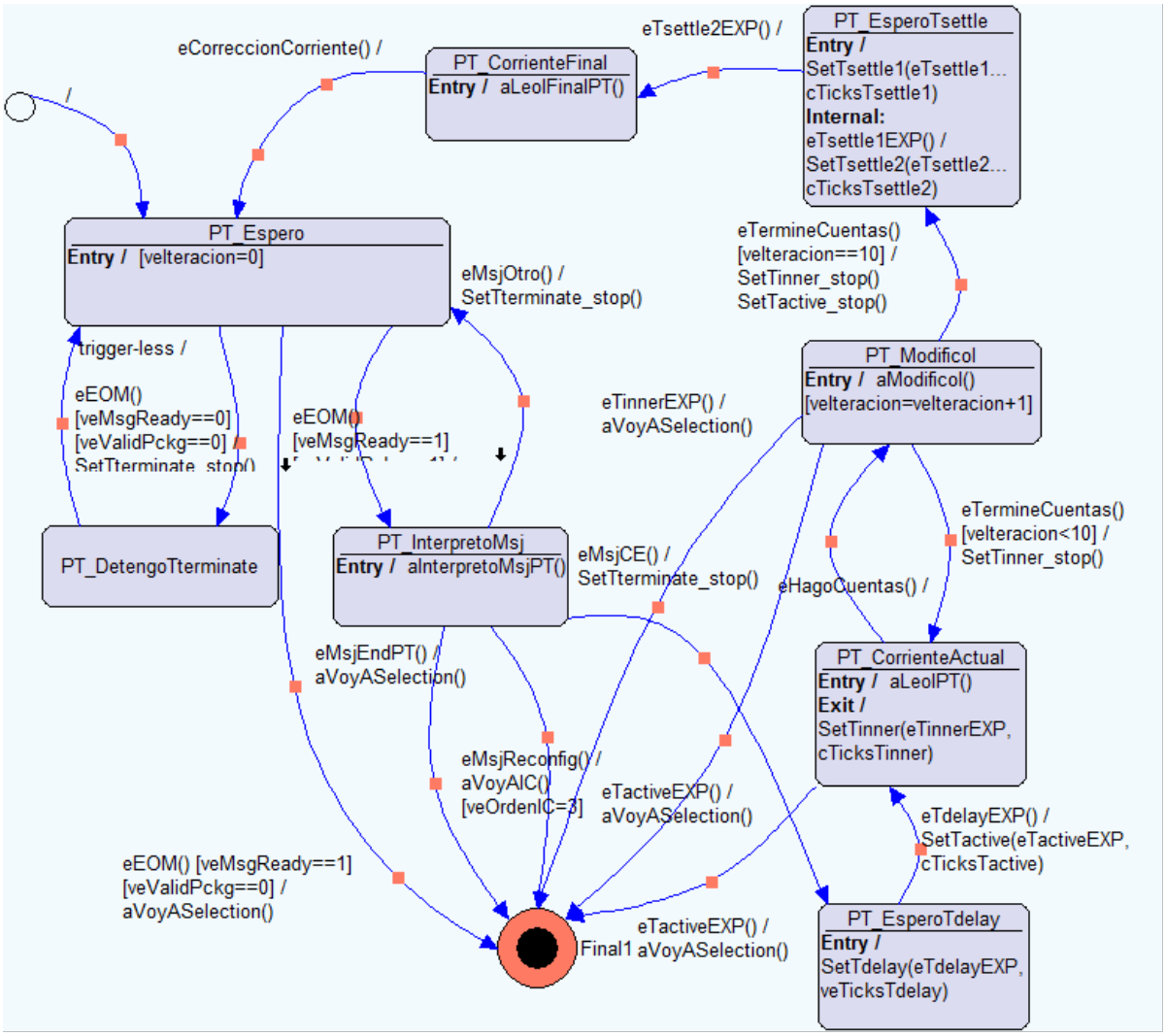


Figura 4.9: Diagrama Statechart de la máquina *PT*.

4.2.2. CommBlock

Este bloque, mostrado en la Fig. 4.10, lleva a cabo la decodificación de la señal de comunicación, con los controles de error correspondientes, según las especificaciones detalladas en el capítulo *Communications Interface*, y deposita los bytes recibidos en un lugar de memoria predeterminado avisando a *CtrlBlock* cuando se completa exitosamente la recepción de un paquete. Durante la recepción y decodificación de los mensajes esta máquina y sus submáquinas implementan controles de tiempos de recepción de mensajes, buscando seguir las restricciones a este respecto establecidas en el capítulo *System Control* del estándar. A medida que se van recibiendo los flancos en la señal de comunicación, interpretando los bits que estos conforman, almacenando los bytes de datos y configurando así los paquetes de mensajes, este bloque realiza controles de duración de bit[8] y verificación de paridad de los bytes, como indica la norma. Luego de terminado el procesamiento de la señal de comunicación, se llevan a cabo controles del checksum del paquete y de la validez del paquete recibido, verificando la correspondencia de la información recibida con el estado en el que se encuentra el sistema. En caso de detectar un error de cualquier tipo en la comunicación, o en los datos recibidos, se descarta el paquete correspondiente, aumentando la probabilidad de expiración de algunos de los tiempos de espera de mensaje.

La implementación del diseño de este bloque se lleva a cabo con una máquina compuesta por dos estados, uno de los cuales es un estado simple de espera de inicio de comunicación. Al recibir un flanco en la señal de comunicaciones se da la transición hacia el estado *DecoMensajes*, llamando a la función central de procesamiento de comunicaciones: *aCOMMReceptor*. El objetivo de esta implementación es que al momento del inicio de la recepción de un paquete, y hasta el final de su recepción y almacenamiento, el resto del sistema esté en espera. Este diseño es producto de la experiencia adquirida con un diseño realizado previamente, incapaz de procesar la señal de comunicaciones de forma exitosa, sobre la cual se presentan detalles en el Anexo D - Contratiempos Experimentados (9).

***Reposo*:** En este estado la máquina está esperando recibir un flanco en el pin de la señal comunicaciones para comenzar a procesarla. Al recibir esta señal, se da la transición hacia el estado *DecoMensaje*, disparando la función de procesamiento de comunicaciones.

***DecoMensaje*:** En este estado, la submáquina recibe el paquete de datos desde la función *aCOMMReceptor* y, al contar con esta información, corrobora la validez del paquete recibido y verifica la validez del *Checksum* correspondiente. Luego vuelve al estado *Reposo* enviando la señal de aviso de mensaje recibido a *CtrlBlock*, seteando las banderas de validez y los timers correspondientes en función de la validez del mensaje.

[8] A los efectos de evitar esperas de cambio de señal demasiado largas, y evitar que el sistema se "cuelgue".

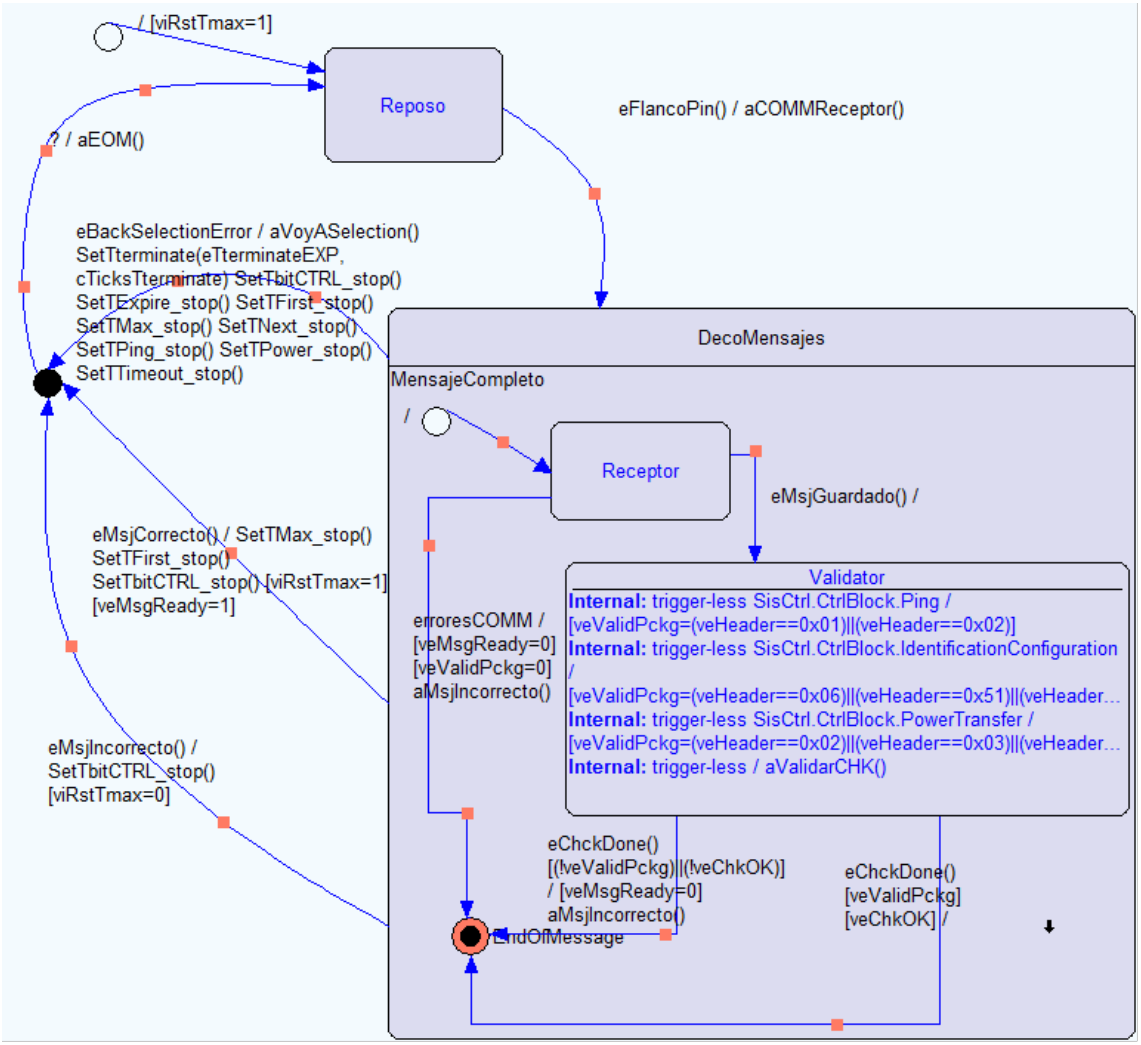


Figura 4.10: Máquina de procesamiento de comunicaciones y mensajes, *CommBlock*.

4.3. Entorno de trabajo

Respecto al entorno de trabajo de la aplicación VisualState, cabe decir que esta define los niveles jerárquicos presentando la estructura del sistema, con las distintas máquinas, submáquinas y estados simples, en una estructura de árbol. Luego, dentro de cada nivel de jerarquía, se trabaja en la interfaz gráfica de la aplicación diseñando/modificando las máquinas, los estados, las transiciones, las acciones y las variables. En la Fig. 4.2 se muestra la Top Level Machine del sistema, que cuenta con dos regiones concurrentes, las cuales tienen tareas bien diferenciadas pero se complementan en el global, ya que el sistema de control basa parte de su funcionamiento en los mensajes recibidos. Para lograr niveles de abstracción a la hora del diseñar, y tener la posibilidad de evaluar máquinas de ciertos niveles jerárquicos, independientemente del funcionamiento del resto de los niveles, es que se puede optar tener el tipo visualización que presenta dicha figura. En las figuras 4.4 y 4.10 se presentan en detalle las máquinas dentro de la Top Level Machine o, en otras palabras, un nivel del jerarquía por debajo de dicha máquina. De igual forma, existen máquinas definidas en niveles de jerarquía inferiores, dentro de las regiones *DecoMensajes*, *Selection-Ctrl*, *Selection-Bob*, *Ping-Ctrl*, *Ping-Bob*, *I&C* y *PT* , que se ven en las figuras. Una mayor profundización en el diseño de la dinámica del sistema, y niveles de jerarquía inferiores, pueden ser encontrados en la sección Diseño (4.2).

Luego de finalizado el proceso de diseño, se llevan a cabo las etapas de Validación y Verificación, etapas en las que se corrobora gráficamente la dinámica del sistema generando manualmente los eventos que dan lugar a las transiciones de estados y se comprueba aritméticamente la lógica del sistema y la inexistencia de ambigüedades, respectivamente. Por último, con el proyecto libre errores en las etapas anteriores, se genera el código correspondiente al diseño realizado. Este código, si bien presenta las limitantes antes mencionadas, es la implementación de la dinámica del sistema, con la importancia que esto implica. Luego del proceso de debugging y de la vinculación correcta entre los eventos reales del sistema y los eventos virtuales definidos en la etapa del diseño, dicho código presenta la robustez de vincular correctamente los estados, sus transiciones y las acciones correspondientes. Además de las prestaciones recién mencionadas, IAR VisualState presenta una herramienta de documentación que resulta muy útil a la hora de realizar esta tarea, ya que presenta una descripción profunda de los estados y transiciones (tanto como se las haya descrito en el proceso de diseño), además de realizar listas por categoría de los elementos que forman parte del diseño.

Los archivos C obtenidos mediante la generación de código de la aplicación representan la estructura general del sistema de control sobre la cual trabajar, y para el caso de este proyecto en particular, se contaba con:

Estados del sistema: Los estados están representados en código en tablas de valores ya que esta fue la opción elegida, apuntando a la optimización del código y los procesos de ejecución en desmedro de la legibilidad. La opción alternativa para la codificación de estados es mediante una estructura switch-case.

Transiciones de estados: En la propia estructura de estados se definen las transiciones posibles.

Eventos y señales: Se definen (*#define*) los identificadores de eventos numerándolos, para luego, al recibirlos, disparar las acciones correspondientes al evento con número igual al que entregó la cola de eventos. Siguiendo esta numeración, se numeran las señales para ser tratadas como cualquier otro evento proveniente de la cola.

Acciones básicas: Se implementan las acciones básicas de la forma que fueron definidas en la aplicación.

Variables y constantes: Se definen las variables y constantes que maneja la dinámica del sistema, y que fueron definidas en la aplicación.

Declaraciones de funciones: Se declaran las funciones que fueron definidas en la aplicación, dejando la definición para ser realizada en el entorno de escritura de código por el usuario.

A partir de los archivos *C* así generados, las bibliotecas específicas de archivos *C* del μC y los archivos que implementan y manejan la cola de eventos y señales, *eventHandler.h*, *simpleEventHandler.h* y *simpleEventHandler.c*, se creó el proyecto sobre el cual trabajar hacia el código definitivo para la programación del Controlador.

En definitiva, el diseño realizado a través de esta herramienta tiene como resultado un paquete de archivos de código en el cual se define la estructura del sistema. Esta estructura está compuesta por: la dinámica de transiciones entre estados, con los eventos que las generan y sus condiciones y acciones asociadas; los estados y submáquinas, con los llamados a funciones que deben ser declaradas durante el diseño (incluidas las funciones de disparo de timers); la generación y encolado de señales de para interacción entre estados paralelos; y la definición de todas las variables y constantes que tengan que ver con esta estructura. Entonces, para obtener un software completo y funcional luego de obtenido el código correspondiente al diseño realizado con VisualState, es necesario implementar software para: manejo de eventos, para lo que se implementaron funciones de encolado de eventos y se utilizó código encontrado en librerías de prueba de la aplicación que implementa una cola de eventos; el cuerpo de las funciones declaradas en el diseño; las funciones de disparo y parada de timers; y para la inicialización del sistema y los servicios del μC.

Por detalles sobre el código y su estructura, el proyecto realizado y el software en general, recurrir al Anexo B - Profundización Software (7)

4.4. Funciones centrales

En esta sección se detallan funciones trascendentes que se implementaron en código a partir de la definición realizada en el diseño de Statecharts, insertada en la estructura del proyecto con el código generado por la aplicación. Estas funciones trabajan directamente con la plataforma, actualizando valores en los registros o tomando datos de los periféricos del μC, además de realizar cálculos de considerable complejidad, razones por las cuales no pudieron ser implementadas en la aplicación de diseño. Estas funciones se encargan de:

- Decodificación de la señal de comunicación y armado de los bytes de datos.
- Realizar la lectura de corriente
- Interpretar los mensajes recibidos en cada estado de *CtrlBlock*
- PID para la corrección de corriente
- Selección, habilitación y excitación de bobina
- Manejo de timers
- Sincronización emisor-receptor

Debido a la estructura del software generado a partir del diseño de StateCharts, todas las transiciones de estado que estas funciones generan se efectivizan mediante el encolado del evento asociado a dicha transición. El código *C* de la implementación de estas funciones no será presentado en esta sección, pero podrá ser consultado en el Anexo B - Profundización Software, Código - Funciones Centrales 7.2.

4.4.1. Procesamiento de la señal de comunicación

Teniendo en cuenta los requerimientos necesarios para el procesamiento de la señal de comunicaciones y la exigencia que el código generado a partir del diseño StateChart impone al μC, se tomó la decisión de implementar una función (*aCOMMReceptor*) en código para decodificar la señal recibida. De esta forma, ubicando el llamado a la función *aCOMMReceptor()* por parte de la máquina de estados en el instante de recibir el evento relacionado a un flanco en el pin de comunicaciones, y sólo volviendo a tener transiciones de estado una vez finalizada la recepción del paquete de datos, se logra obtener paquetes válidos con una probabilidad directamente proporcional al nivel de acople de las bobinas. A continuación se desarrollan los conceptos en los estados definidos en la función *aCOMM-Receptor()*, dejando los detalles del código para el Anexo B - Profundización Software, sección Código - Funciones Centrales, Procesamiento de la Señal de Comunicación (7.2).

State 0 - Preámbulo:

Este es el estado inicial para la recepción de un paquete al cual se entra luego de un flanco de comienzo de preámbulo. Dado que, si bien existen límites establecidos por la norma, el largo del preámbulo es desconocido de antemano, este estado se mantiene en loop mientras no se reciba un *Start Bit* ("0") y se esté por debajo del largo máximo admitido para un preámbulo. Al verificar un bit largo, correspondiente a un "0", se sale del lazo de ejecución, se realiza un promedio de la duración de los bits de preámbulo, se corrobora que el preámbulo sea válido (por lo menos 4 bits de preámbulo) y se va al estado de recepción del encabezado.

State 1 - Encabezado:

En este estado se procesan 10 bits, ya que el *Start Bit* correspondiente a este byte fue procesado por el estado de preámbulo, quedando sólo los 8 bits de datos, el bit de paridad y el *Stop Bit*. A estos efectos se establece la variable *veBitsLeft* en 8, se llama a la función *procesaBit()* que agrega un "1" o un "0" al byte en función de la duración de nivel de señal e invirtiendo la paridad por cada bit "1" recibido, y se

decrementa *veBitsLeft* en uno, repitiendo este proceso hasta que no queden bits por recibir. Luego de recibido el byte, se invierte el orden de sus bits[9] y se espera por el siguiente bit estableciendo la variable de bit de paridad *viBitP* en función de su valor. Seguido a esto se comparan el bit de paridad recibido con el bit calculado, y en caso de discordancia se sale de la función reseteando la bandera *CommOK*. Si la paridad del byte recibido es la esperada, se calcula la cantidad de bytes de datos que se recibirán a partir del valor del encabezado, según indica la norma, y se espera el *Stop Bit* para proceder al estado de recepción de mensajes.

State 2 - Mensajes:

En el estado *Mensajes* se implementan dos loops de largo predefinido. El primero es de largo definido en función de la cantidad de bytes a recibir, valor que se calcula en el estado *Encabezado* y determina la cantidad de veces que se ejecutará un loop idéntico al que guarda el encabezado, en el estado correspondiente, pero para guardar cada byte de mensaje. Luego de archivados los mensajes en el arreglo *Mensaje[]*, y en caso de no haberse dado ningún error de paridad, se procede al estado de recepción de Checksum.

State 3 - Checksum:

Este estado archiva sólo el byte final, correspondiente al *Checksum* de verificación enviado por el receptor. Los pasos para llevar a cabo esta acción son iguales a los que se realizan para la recepción del encabezado. Luego de la recepción, y verificación de paridad, se encola el evento *eMsjGuardado* y se procede al estado final.

State 4:

Este estado sólo existe a efectos de resetear la bandera *CommOnCourse* de recepción de mensaje en curso y salir así del loop principal de la función.

4.4.2. Lectura de corriente

El sistema basa partes centrales de su funcionamiento en los datos recibidos del bloque sensor de corriente ya que, como se explicó en el capítulo Hardware - Etapas Auxiliares - Sensor de Corriente (3.4.1) como la detección de presencia de un posible receptor, el ajuste en la potencia transferida y la detección de objetos no deseados en la superficie se basan en una evaluación de la corriente consumida por las bobinas. A nivel de software, la función que implementa esta acción es *LecturaADC()* y esta se ejecuta cada vez que *CtrlBlock* necesita el valor de la corriente y lo solicita. Al iniciar su ejecución enciende el ADC y se itera 16 veces el siguiente proceso: se espera a que el ADC realice la conversión del valor analógico de tensión que se recibe en el pin *AIN1* a un valor representado con 10bits, momento en el cual se establece la flag de fin de conversión, y se guarda el valor. Luego se implementa un filtro pasabajos por software, promediando estos 16 valores de lectura, con el objetivo de reducir la sensibilidad al ruido en la señal.

[9]Esta inversión de bits se lleva a cabo dado que el orden de llegada de bits es $b_0..b_7$, según se define en la norma.

4.4.3. Interpretación de Mensajes

La interpretación de los mensajes enviados por el receptor de potencia es una parte fundamental para efectivizar la realimentación en la que se basa el control del sistema. Dado esto es que en los tres estados activos del bloque *CtrlBlock* (*Ping*, *Identification & Configuration* y *Power Transfer*), es decir en los que se supone presencia de receptor, existen funciones de interpretación de los mensajes. Estas funciones, llamadas *aInterpretoMsjPing*, *aInterpretoMsjIC* y *aInterpretoMsjPT*, tienen en común la característica de tomar el encabezado del mensaje para verificar que este corresponde al grupo de mensajes posibles para ese estado y terminar el proceso de carga devolviendo al sistema al estado *Selection* en caso negativo. La diferencia que presentan estas funciones son las acciones que toman al verificar que el mensaje recibido es válido e identificar el encabezado del mensaje, por lo general haciendo uso de los datos contenidos en el cuerpo del mensaje. En *aInterpretoMsjPing*, las acciones posibles con mensajes válidos son: Acumular un contador de intentos fallidos o guardar el dato *Signal Strength* de la bobina actual. En *aInterpretoMsjIC* se verifica el orden de recepción y la cantidad de los mensajes (ya que son características detalladas en la norma), y las acciones a tomar son: Actualización del mensaje siguiente esperado, establecimiento de los parámetros del Contrato de Transferencia de Potencia y verificación de la cantidad de mensajes recibidos contra la cantidad declarada en el último mensaje. En cambio, en *aInterpretoMsjIC* al recibir mensajes válidos se realizan las siguientes acciones: Ajustes del punto de funcionamiento de la transferencia de potencia, a partir de los datos recibidos en los mensajes de *Control Error*, o devolución del sistema al estado *Identification & Configuration* al recibir cierto tipo de mensajes *End Power Transfer* con el fin de renegociar el Contrato de Transferencia de Potencia.

4.4.4. Ajuste del punto de funcionamiento - PID

Al recibir un mensaje de *Control Error* se debe ajustar el punto de operación de la transferencia de potencia dentro de los límites de tiempo establecidos por la norma (t_{active}). Dadas las características del sistema implementado, este ajuste se realizará en caso de que el *Control Error* recibido corresponda, como mínimo, al paso mínimo establecido para la variación de corriente. A su vez, para cumplir con este objetivo es necesario contar con la lectura de la corriente consumida por las bobinas, acción que también tiene determinado cierto límite de tiempo (para poder cumplir con la temporización del ajuste), sin embargo se debe esperar un tiempo t_{delay} para que la corriente se estabilice. Una vez obtenido el valor del sensado de la corriente, se debe ajustar el punto de operación de la transferencia, en un tiempo máximo determinado (t_{active}), para asegurar que finalizado este proceso el sistema está apto para continuar recibiendo mensajes, luego de esperar un tiempo para la estabilización de la corriente (t_{settle}).

Cada ajuste del punto de funcionamiento se lleva a cabo mediante la ejecución de varias iteraciones de un proceso PID, donde se deben cumplir relaciones entre la cantidad de iteraciones y la duración de cada una de ellas para no exceder las restricciones impuestas por la norma: $t_{active} = t_{inner} * It_{max}$. Cada iteración del PID consta de los siguientes pasos en el orden en que se muestran:

- $I_{new} = I_{actual} * (1 + CtrlError/128)$

- $Error = I_{new} = I_{actual}$. En la primera iteración del PID $I_{actual} = I_{AlRecibirMsjCE}$, luego I_{actual} va variando.
- $P = K_p * Error$. Parte proporcional del PID.
- $I_{actual} = I_{anterior} + (K_i * Error * t_{inner})$. Parte integral del PID. Los límites de integración definidos en la norma son definidos para cada tipo de emisor. En la primera iteración $I_{anterior} = 0$.
- $PID = P + I$.
- $V_{actual} = V_{anterior} - (S_v * PID)$. V es la variable a modificar, para el sistema *WiQi* (por ser un *tipo A6*) V es la frecuencia, que tiene a 205 kHz como valor máximo posible; luego de llegar a este valor se modifica el ciclo de trabajo. S_v es un factor de escala que se varía para cada tipo de emisor.
- Se establece el nuevo punto de funcionamiento con la nueva frecuencia.
- Por último se sensa la corriente, un tiempo $t_{active} + t_{delay} + t_{settle}$ luego de recibido el mensaje de *Control Error* a los efectos de verificar el ajuste realizado.

4.4.5. Manejo de bobinas

Las características del sistema diseñado hacen que sea de suma importancia tener un manejo adecuado de las tres bobinas de emisión de potencia ya que, para evitar interferencias en las señales de potencia y comunicación, en ningún momento debe haber más de una bobina activa. Además, dado que las propias bobinas son tanto antenas de recepción para la señal de comunicación, como el medio físico para efectivizar la onda de potencia con una frecuencia modificable, se vuelve necesario un cuidadoso manejo de las señales que las controlan. Para esto se implementaron funciones de control para la señal PWM (de construcción de onda) y para la habilitación de los drivers (de selección de bobina):

PWM: La función *SetPWMBobina(float Frec,float DutyCicle,uint8_t Bobina)* setea el timer al valor de frecuencia de la señal PWM en un valor necesario (recibida como parámetro), con un ciclo de trabajo necesario (recibido como parámetro) y en la bobina elegida (recibida como parámetro).

Ping Analógico - recorrido de bobinas: La función *aSeteoPingAnalogico()* establece, si se la habilita, la frecuencia de la onda PWM (a través de la función *SetPWMBobina*) en un valor cercano a resonancia y con ciclo de trabajo de 50 % para el ping analógico, que resultará en un valor de corriente conocido si está en vacío, pudiéndose detectar a partir de esto la presencia de un posible receptor. Además, la bobina habilitada estará determinada por el estado de la máquina *Selection* en el que sea llamada esta función.

Ping Digital - recorrido de bobinas: La función *aSeteoPingDigital()* tiene un funcionamiento similar a *aSeteoPingAnalogico()*, excepto que la frecuencia de la onda PWM se establece en un valor intermedio de los valores posibles ($175kHz$), y con el objetivo de brindar al receptor un canal para establecer la comunicación.

Ping Digital - selección por acople: La función *aSeleccionoBobina()* tiene el objetivo de determinar cuál de las tres bobinas tiene mejor acople con el receptor. Esto se lleva a cabo haciendo una comparativa entre los tres valores recibidos como mensajes *Signal Strength* (uno por cada bobina) y estableciendo la variable *vsBCP* en el valor correspondiente a la bobina con mejor acople.

Habilitación de bobina: La función *EnableBobina(uint8_t Bobina)* habilita seteando en "1" el pin correspondiente al enable de la bobina recibida como parámetro.

4.4.6. Manejo de timers

A lo largo de todo el capítulo 5, *System Control*, de la norma *Qi* se detalla el funcionamiento que debe tener el sistema de control en cuanto a acciones a tomar como resultado de los datos recibidos dependiendo del estado en que se encuentre el sistema. Sin embargo, todas y cada una de las acciones, decisiones, recepciones de mensaje o transiciones de estado tienen restricciones respecto al tiempo de cumplimiento. Para cumplir con estas restricciones se implementaron varias funciones haciendo uso de los 4 timers con los que cuenta el μC, cada uno configurado a la frecuencia más conveniente para los tiempos a medir. En la tabla 4.1 se muestra una lista de restricciones de tiempo a cumplir por el sistema.

Las funciones en código *C* que implementan los manejos de los timers para contar los tiempos descritos en la tabla 4.1 son las funciones *SetXXXX*[10]*(SEM_EVENT_TYPE event, unsigned int ticks)* para inicial el timer que encolará el evento *event* luego de un tiempo $ticks * f_{timer}$, y *SetXXXX_stop(void)* para detener el timer.

[10]Donde XXXX corresponde al nombre del tiempo a medir

Nombre	tiempo	Objetivo
Selection		
t_{odi}	$500ms$	Período de sensado de bobinas
t_{odd}	$70\mu s$	Duración de sensado de superficie
t_{odm}	$19{,}5\mu s$	Ventana de lectura de corriente
Ping		
t_{ping}	$70ms$	Tiempo máximo de espera por comienzo de mensaje
t_{first}	$20ms$	Tiempo máximo de recepción de paquete
$t_{terminate}$	$28ms$	Tiempo de corte de la señal para volver a *Selection*
t_{expire}	$90ms$	Tiempo máximo de corte de señal sin hubo transición de estado
Identification & Configuration		
t_{next}	$21ms$	Tiempo entre el fin de un mensaje y el comienzo del siguiente
t_{max}	$170ms$	Tiempo de espera para la finalización de un paquete en recepción
$t_{terminate}$	$28ms$	Tiempo de corte de la señal para volver a *Selection*
t_{expire}	$90ms$	Tiempo máximo de corte de señal sin transición de estado
Power Transfer		
$t_{timeout}$	$1500ms$	Tiempo de espera entre paquetes *Control Error*
t_{power}	$23000ms$	Tiempo de espera entre paquetes *Received Power*
t_{active}	$20ms$	Tiempo para corrección de la corriente
t_{delay}	$5..205ms$	Tiempo de espera de espera para estabilizar la corriente
t_{settle}	$5ms$	Tiempo de estabiliazación de la corriente
t_{inner}	$1..5ms$	Tiempo de iteración del PID

Tabla 4.1: Restricciones de tiempo a cumplir en cada estado.

4.4.7. Sincronización emisor-receptor

El tipo de comunicación establecida entre el emisor y un receptor comienza siendo asíncrona, dado que no se cuenta con un reloj de referencia. A los efectos de reducir los posibles errores que esto pueda generar en la recepción de datos la norma define que cada paquete debe contar con un preámbulo con el fin de que el emisor pueda sincronizarse con esta señal. Esta sincronización se llevó a cabo a través del sensado del pin de comunicaciones a lo largo del preámbulo, ejecutando los siguientes pasos:

- Levantando una interrupción con cada cambio de flanco detectado en el pin;
- Guardando el valor de la cuenta y redisparando el timer en la ISR;
- Incrementando en uno el contador de ciclos de preámbulo;
- Identificando, en base el valor del timer guardado, si se obtuvo un "1" (sigue el preámbulo) o un "0" (*Start Bit*);
- Acumulando el tiempo total del preámbulo a los efectos de promediar el valor de período.

La implementación software para el sensado de pin se realizó mediante la función *INTERRUPT_HANDLER(EXTI_PORTB_IRQHandler, 4)*.

Luego de recibido un flanco que identifique un *Start Bit*, se utiliza el valor promedio de período calculado para continuar evaluando de forma síncrona el valor de la tensión en el pin correspondiente. Esta sincronización se hace mediante la función *INTERRUPT_HANDLER(TIM4_UPD_OVF_IRQHandler, 23)*, que dispara interrupciones cuando expira el timer *TIM4*, de la forma que se explicó en la sección Diseño, CommBlock (4.2.2) de este capítulo.

Capítulo 5

Conclusiones y Resultados

5.1. Resultados

En términos de hardware, una vez finalizado el proyecto, el resultado obtenido es un sistema que -en líneas generales- coincide con el planteado en los objetivos iniciales, dado que el mismo llevaría a cabo la función de realizar la carga inalámbrica de dispositivos móviles, en cumplimiento parcial de los requisitos establecidos por la norma *Qi*. Hablamos de cumplimiento parcial dado que para el desarrollo del trabajo existieron ciertas restricciones que se detallan a continuación.

La norma *Qi* establece que el sistema debe presentar un paso de frecuencia, entre los 105 y los $205kHz$, y de ciclo de trabajo entre el 50 % y el 10 % (una vez alcanzados los $205kHz$) que redunden en pequeñas variaciones de potencia. Tomando en cuenta esta información, se puede afirmar que la primera limitante al cumplimiento se vio impuesta por el tipo de microcontrolador utilizado para el desarrollo del prototipo (ver Sección Hardware - Controlador), ya que -en términos de la resolución alcanzada para el manejo del ciclo de trabajo de la señal- no se logra cumplir con lo que la norma requiere. Una segunda limitación está dada por la opción de implementar el bloque sensor de corriente con un amplificador operacional convencional, puesto que se contaba con el chip LM324 de Texas Instruments -que contiene 4 amplificadores de los que sólo se utilizaban 3-, en lugar de un INA199 integrado, también de Texas Instruments pero específico para el sensado de corrientes. Con esto, se generó una pérdida de resolución en la detección de variaciones de corrientes, no pudiendo alcanzar los $7mA$ de resolución que la norma solicita, y, en consecuencia, una pérdida en la resolución de la variación de la potencia transferida. Entonces, a los efectos de lograr un sistema que realice una carga con una curva de variación de potencia similar a la requerida por la norma, en el sistema desarrollado se establecen 9 pasos de variación de frecuencia y 3 pasos de variación del ciclo de trabajo.

Siguiendo la línea de resultados hardware cabe afirmar que el prototipo cuenta, además, con un sistema de procesamiento de comunicaciones capaz de demodular la señal de comunicación a partir de la señal de potencia. En este sentido, el bloque de comunicación brinda a la salida una señal apta para ser introducida directamente en el controlador, y este es capaz de decodificar la señal y armar los bytes de datos recibidos.

5.1.1. Datos del sistema

Características eléctricas del sistema

Parámetro	Descripción	Valor
V_{IN}	Tensión de alimentación	$12VDC$
η_{Mag}^{min}	Eficiencia en la transferencia inalámbrica	$86\,\%(*)$
$P_{standby}$	Potencia consumida en vacío	184mW
P_{comm}	Potencia entregada para establecer comunicación	1.91W $(**)$
P_{carga}^{avg}	Potencia promedio consumida cargando	$(***)$

$(*)$Eficiencia mínima en la transferencia inalámbrica de potencia (tomando en cuenta sólo las bobinas), asegurada por la norma dado el acople mínimo aceptable para realizar la carga.

$(**)$Potencia que se empieza entregando al sensar un posible receptor a la espera del *Control Error* necesario para ajustar la transferencia a través del PI.

$(***)$Dato que se planeaba obtener promediando la potencia a lo largo de un proceso completo de carga, y no se obtuvo dado que no se logró funcionalidad completa.

Características físicas

Parámetro	Valor (mm)
Largo	130
Ancho	54,5
Espesor	18
Superficie de carga	93x52

5.2. Conclusiones

Finalizado el trabajo requerido para lograr los objetivos de este proyecto, y evaluados los contratiempos experimentados y la experiencia obtenida, es posible sacar en limpio algunas conclusiones. En primer lugar, se concluye que el proceso de implementación hardware resulta ser un proceso iterativo y más costoso de lo que a priori parece. Esto responde a la alta probabilidad de errores que el diseño hardware presenta (sobre todo si no se cuenta con experiencia previa), y al hecho de que para obtener una placa de buena calidad la misma debe ser fabricada en el exterior, tornando este proceso aún más engorroso, lento y costoso. En segundo lugar, se concluye que fue una buena decisión la utilización de Statecharts para el diseño del sistema de control, dada la robustez que el código resultante presenta para el manejo de la dinámica del sistema, así como el ahorro de tiempo y la claridad para el diseño que esta herramienta brinda. Sin embargo, no se considera haber tomado una buena decisión al intentar implementar la decodificación de la señal de comunicaciones a través de un diseño Statecharts, dado que se tomaron en cuenta sólo los requerimientos hardware impuestos por la norma para elegir la plataforma de control, pero no se evaluó cuánto exigiría al microcontrolador el software generado por la herramienta. Por último, y como aprendizaje obtenido en base a los errores cometidos, se concluye que en un proceso de diseño integral como el que aquí se implementó, es conveniente el paralelismo temporal entre los desarrollos hardware y software, o en otro caso, priorizar el desarrollo del software. Esto responde a que, desarrollando primero el hardware del sistema, puede ocurrir (como sucedió durante este trabajo) que se cometan errores en la estimación de los requerimientos que el software impondrá al hardware, obligando a cambiar hardware y, posiblemente, tener que rediseñarlo. En cambio, invirtiendo el orden de las tareas, se podrá avanzar en el diseño y desarrollo del software, al menos, hasta la etapa de debugging o adaptación al hardware, pero se contará ya con una estimación más cercana de los requerimientos de hardware para dicho software.

Otra conclusión que es posible extraer en relación con el hardware, es que para proyectos en los que se hace necesario el desarrollo de este tipo de prototipos, debe tenerse en cuenta que muchos de los componentes necesarios para su óptimo funcionamiento no se encuentran en plaza. Esto hace imprescindible que, para determinar plazos acertados para el proyecto, se deban tomar en consideración los tiempos adicionales de espera de la mercadería proveniente del exterior.

Con el diseño del software del controlador del sistema finalizado, previo a la etapa de debugging, se creía tener un software bastante avanzado en cuanto al desarrollo. Sin embargo, no se contaba con argumentos de peso para creer que el sistema podía ser funcional en tiempo de ejecución dado que, si bien las herramientas de prueba con las que cuenta el software de diseño fueron de utilidad testeando submáquinas de pocos estados, presentaron múltiples bugs y errores al momento de probar diseños de tamaño considerable. En este punto del desarrollo se consideró la utilización de un framework existente para la implementación de Statecharts (framework RKH), pero en consulta con los desarrolladores de la plataforma, y considerando el tiempo con el que se disponía para finalizar este proyecto, se descartó la idea. También se contaba con la posibilidad de utilizar la herramienta de vínculo entre la plataforma de diseño Statechart y el software para desa-

rrollo de sistemas embebidos, para poder llevar a cabo un debugging on-chip. Pero por problemas explicados en la sección Contratiempos Experimentados (9) esta tampoco fue una opción válida para la finalización y correcciones del diseño. A su vez, al no contar con un set completo de herramientas de debbuging se demoró demasiado en concluir que era necesario cambiar el diseño del software de procesamiento de comunicaciones debido a la incapacidad del microcontrolador de cumplir con los tiempos de señal y el manejo del software generado. A raíz de estos inconvenientes, sumados a los problemas ya expuestos en Conclusiones, no fue posible finalizar el software del sistema de forma de llevarlo a un estado funcional de acuerdo al comportamiento esperado.

Por ende, el sistema final **es capaz** de transferir potencia a un receptor depositado sobre la superficie por cualquiera de sus tres bobinas, siendo capaz incluso de identificar cuál de sus bobinas es la que tiene mejor acople con el receptor y seleccionarla para realizar la transferencia de potencia. Esto lo lleva a cabo utilizando de forma exitosa otro de los bloques que lo componen: el sensor de corriente en conjunto con el ADC del controlador. También es capaz de establecer un canal de comunicación con un receptor compatible *Qi* a través de la onda de potencia enviada, demodulando la señal de potencia para extraer la señal de comunicación. Es capaz, también, de cumplir con los tiempos de la comunicación y decodificar la señal recibida para interpretar el valor binario que recibe del receptor, armando a partir de esto los bytes de datos y verificación, logrando corroborar exitosamente la paridad de cada byte y el checksum de los paquetes de datos.

Sin embargo, el sistema no es capaz, con el software desarrollado a la fecha, de tomar las acciones correspondientes a los mensajes recibidos. Por lo tanto, **es incapaz** de realizar el ajuste sobre la señal de potencia que solicita el receptor a través de la comunicación o, en otras palabras, el control sobre la transferencia de potencia durante el proceso de carga. En consecuencia, el sistema tiene la capacidad para ajustar la potencia transferida modificando la frecuencia o ciclo de trabajo de la señal, pero no lo hace respondiendo a los pedidos del receptor.

En el mismo sentido, y dado que la carga con este sistema se realiza a frecuencia constante y sin capacidad de comparar la potencia transmitida y la potencia solicitada, **no es capaz** de detectar la presencia - durante el proceso de carga - de elementos inductivos no deseados sobre la superficie. Por último, y también relacionado con la imposibilidad de interpretar los paquetes de datos recibidos, el sistema desarrollado tampoco sabrá determinar cuándo debe finalizar la transferencia de potencia como consecuencia de que la batería del receptor haya alcanzado su carga máxima.

Luego de una evaluación global de la gestión del proyecto y del avance alcanzando, se concluye que se equivocaron los métodos de desarrollo e implementación. Se dedicó demasiado tiempo al diseño, desarrollo y testeo del hardware por etapas, menospreciando la importancia que el software logrado tendría en la unificación de los bloques, y por ende en las capacidades globales del sistema. Con esa óptica, se planificó muy poco tiempo para la etapa de debugging del software del sistema, partiendo de la base de que sólo serían necesarios ajustes y correcciones sobre el software diseñado, sin tener en cuenta la posibilidad de tener que modificar grandes bloques por incompatibilidad con el hardware, como en definitiva ocurrió.

En consecuencia, los planes a futuro con el desarrollo realizado a lo largo de este proyecto se centran, en primera instancia, en seguir trabajando en el código desarrollado para el controlador del sistema. En base a esto se intentará lograr un cargador que, dentro de sus limitaciones hardware, pueda implementar todos los requerimientos establecidos por el estándar *Qi*, llevando a cabo cargas de dispositivos compatibles de forma segura y eficiente. En una etapa más avanzada, y pensando en mejorar el sistema planificado, se podrá rediseñar el sistema, a los efectos de introducir los cambios necesarios para obtener la sensibilidad a los cambios de corriente requerida por la norma. De igual forma, se podrá investigar las opciones existentes para lograr la precisión requerida en la variación del ciclo de trabajo de la señal de control de potencia, sin que el sistema pierda características esenciales: bajo costo, bajo consumo y dimensiones físicas reducidas.

Por último, y basado en las estimaciones realizadas en el anexo dedicado a evaluación de costos (Anexo C), existe la posibilidad de buscar fondos de inversión o apoyo a emprendedores, una vez se cuente con el sistema terminado y se logre un funcionamiento adecuado. Este proyecto se basa en una comparativa del costo estimado, con el precio en el mercado de cargadores del mismo tipo[1], cuyo valor de venta en origen es de $U\$s60$. Si bien al diseño aquí realizado habría que agregarle un diseño de carcasa y una presentación adecuada, impactando esto en costos de producción, se considera que existen métodos a mejorar y hacer más eficientes en términos de la producción, que pueden redundar en una reducción de costos.

En definitiva, se considera que este proyecto puede representar un puntapié inicial de cara al desarrollo de cargadores inalámbricos en nuestro país, generando un ámbito para el desarrollo posterior de otros sistemas de carga inalámbrica apuntados a distintos dispositivos móviles eléctricos de uso masivo.

[1] http://www.tylt.com/vu-wireless-charger

Capítulo 6

Anexo A - Profundización Hardware

6.1. Etapas Auxiliares

6.1.1. Sensor de Corriente

En esta sección se detallan características y simulaciones respecto del bloque auxiliar Sensor de Corriente, mostrado en la Fig. 3.15.

Para medir la corriente se utiliza una resistencia shunt de $0,33\Omega$. Esta resistencia se coloca en serie con la entrada de potencia a los Mosfet, por lo que solo se verá la corriente consumida por los Mosfet. El voltaje entre los bornes de la resistencia, cuando no hay dispositivos o algún elemento ferromagnético sobre la bobina se muestra en la Fig. 6.1.

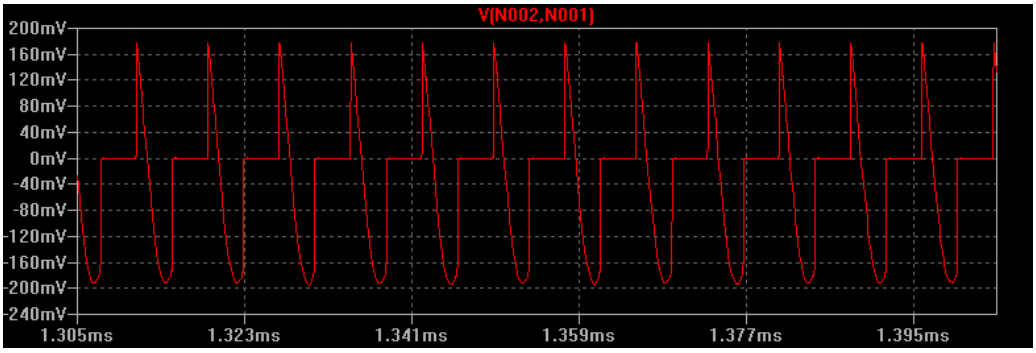


Figura 6.1: Tensión en bornes del Shunt.

Como se observa, la tensión tiene una forma irregular y es de amplitud muy baja. Se quiere observar el valor medio de la corriente. Por lo tanto se coloca un filtro pasabajos con frecuencia de corte mucho menor a la frecuencia mínima de switcheo del inversor (115kHZ). Las resistencias de 10Ω conectadas a los bornes del Shunt, y el capacitor en paralelo con este, implementan un filtro pasabajos con frecuencia de corte de 8 kHz.

El valor de salida del pasabajos (V_f) cuando la entrada es la señal de la Fig. 6.1 se observa en la Fig. 6.2.

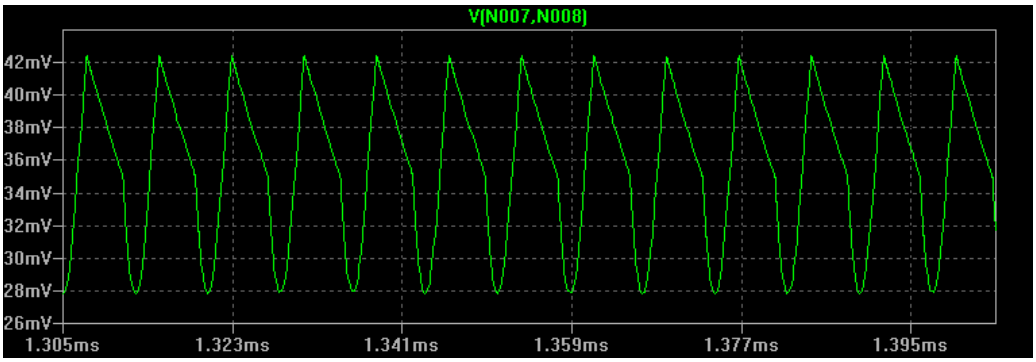


Figura 6.2: Tensión en bornes del Shunt filtrada.

Se observa que el valor de salida es pequeño con respecto a la resolución del ADC del microcontrolador, recoredemos que el ADC del Microcontrolador es de 10 bits, siendo el máximo valor la tensión V_{DDA} que es de 5V. Por lo tanto el paso del ADC es de $5/1024 = 4,88mV$. Entonces el valor de tensión a la salida del filtro deberá ser amplificado para poder tener una mayor escala. Para esto se coloca un amplificador diferencial con ganancia 250V/V y una impedancia de entrada $Z_{in} = 2R + R_{in} = 2x1000 + R_{in}$. Donde R_{in} es la resistencia de entrada del amplificador operacional, por lo tanto no afecta a la salida del filtro pasabajos ya que este tiene resistencia de salida muy baja. La salida del amplificador se observa en la Fig. 6.3.

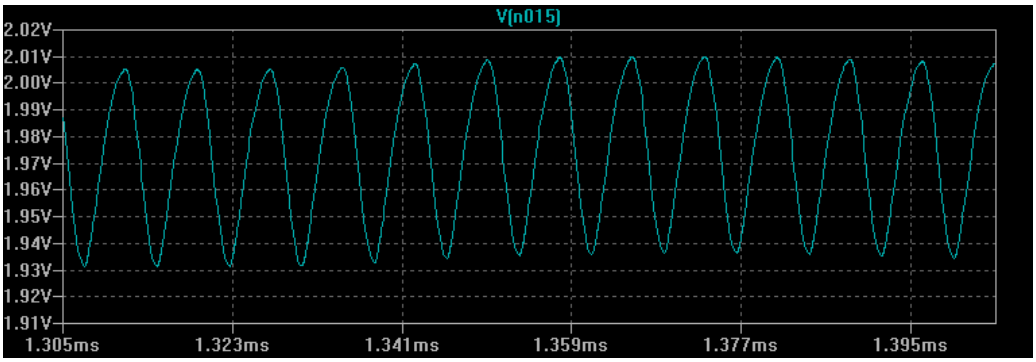


Figura 6.3: Tensión a la salida del amplificador diferencial.

Sobre esta señal se le aplicará nuevamente un filtro pasabajos RC, pero este de frecuencia de corte 1600 Hz, para que el ADC solo mida el valor medio. Este filtro está implementado por la resistencia de valor un $1k\Omega$ y el capacitor de valor $0,1\mu F$ a la salida del bloque. La salida de este filtro se muestra en la Fig. 6.4

6.1.2. Diseño conversor DC-DC

Como se explicó en la sección Etapas Auxiliares (3.4), del capítulo dedicado a Hardware, los componentes elegidos para el diseño del sistema hacían necesaria la inclusión de un bloque conversor DC-DC, a los efectos de proveer la alimentación adecuada para gran parte del circuito. A tales efectos se utilizó un integrado específico para esto, y se diseñó el circuito periférico necesario para obtener la tensión y corriente deseadas. En la Fig. (6.5)

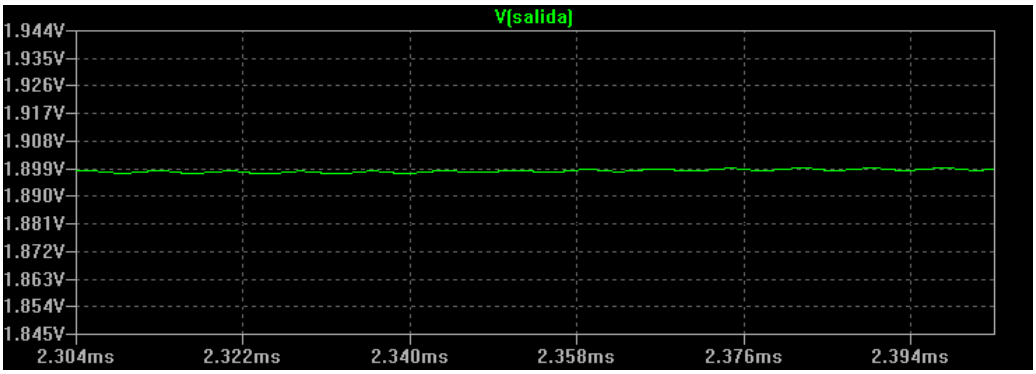


Figura 6.4: Tensión a la salida del amplificador diferencial filtrada.

se muestra un esquemático de la configuración típica encontrada en la hoja de datos del bloque TP54231[8] utilizado. Si bien este esquemático presenta un esquemático diseñado para proveer una tensión 3.3VDC y 2A máximo, en la hoja de datos del integrado se presentan valores típicos necesarios para algunas características en la tensión de salida, y los cálculos necesarios para determinar los valores necesarios en función de la tensión y corriente deseadas.

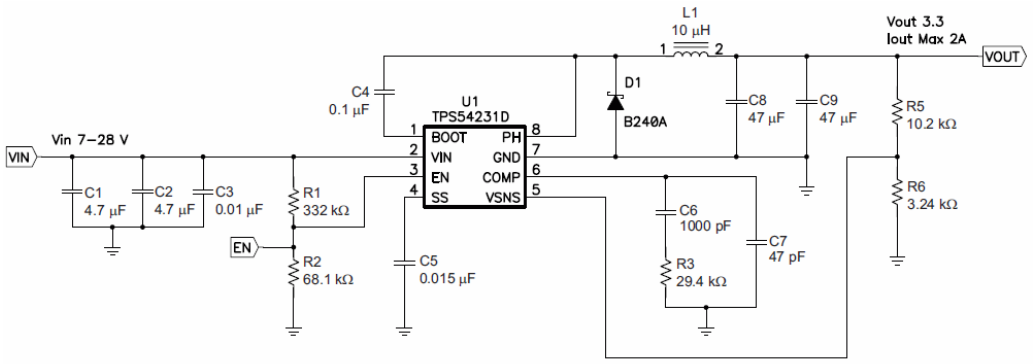


Figura 6.5: Esquemático de aplicación típica del bloque step-down, extraída de [8].

Para este caso, se requería tener a la salida de esta etapa un voltaje de 5VDC, tensión que se logra ajustando las resistencias R5 y R6 del esquemático a partir de las relaciones 6.1 y 6.2. Además el fabricante recomienda que la resistencia R_5 sea de valor cercano a los $10k\Omega$, por lo que entonces se obtuvo:

- $R_5 \approx 10k\Omega$
- $V_{out} = 5V$
- $V_{Ref} = 0{,}8V$
- $R_6 = 1{,}91k\Omega$

$$R_6 = (R_5 * V_{Ref})/(V_{out} - V_{Ref}) \tag{6.1}$$

$$V_{out} = V_{Ref} * ((R_5/R_6) + 1) \tag{6.2}$$

El valor mínimo de la inductancia se obtuvo de la relación 6.3, donde los valores utilizados para este diseño fueron:

- $V_{out,max} = 5{,}3V$, ya que $5{,}5V$ es la máxima tensión de entrada admitida por el μC.
- $V_{in,max} = 12{,}5V$.
- $K_{ind} = 0{,}2$, recomendación del fabricante cuando se utiliza un condensador de salida con alta ESR[1].
- $I_{max} = 150mA$.
- $f_{sw} = 570kHz$.
- $L_{min} = 185\mu Hy$.

$$L_{min} = (V_{out,max} * (V_{in,max} - V_{out}))/(V_{in,max} * K_{ind} * I_{max} * f_{sw}) \tag{6.3}$$

Para esta implementación, se solicitaron muestras de inductancias a Würth Electronik de al menos el valor calculado, a raíz de lo que se obtuvieron inductancias de $L = 270\mu Hy$. Estas inductancias cumplían con lo se pretendía, y generan un ripple de corriente máximo tal que $I_{LPP} = 24mA$, a la frecuencia f_{sw}.

Para el cálculo del capacitor de salida, el fabricante recomienda que la frecuencia de corte del circuito LC sea, cómo máximo, $f_c = 25kHz$, y dado el cálculo de la frecuencia de corte de un circuito RC ($\omega = 1/(RC)$), se obtuvo una $C_{O,min} = 127nF$, tomando $R_O = (V_O/I_O) = 5/0{,}1 = 50\Omega$. Por lo que se utilizó un capacitor de valor $C_O = 2{,}2\mu F$, con una ESR de $3m\Omega$.

Entonces, con estos valores y la ecuación 6.4, siendo $D = V_O/V_I$, se obtuvo el ripple en la tensión de salida: $V_{OPP} = 0{,}45mV$.

$$V_{OPP} = I_{LPP}[((D - 0{,}5)/(4 * f_{sw} * (C_O)) + R_{ESR})] \tag{6.4}$$

El resto de los componentes del circuito se calcularon para obtener una tensión de enable apta para asegurar el funcionamiento de esta etapa cuando se energizara el sistema. Para este cálculo, se toman los valores de tensión máximo y mínimo para los cuales se quiere que el sistema trabaje: $V_{Start} = 11V$ y $V_{Stop} = 14V$. Con esos valores, y las ecuaciones 6.5 y 6.6 dadas por el fabricante, se obtiene: $R_{en1} = 1M\Omega$ y $R_{en2} = 116k\Omega$.

$$R_{en1} = (V_{Start} - V_{Stop})/(3\mu A) \tag{6.5}$$

$$R_{en2} = (V_{EN})/[((V_{Start} - V_{Stop})/R_{en1}) + 1\mu A] \tag{6.6}$$

[1]Resistencia en Serie Equivalente

Por último, se calculó el condensador C_{ss}, que tiene como objetivo el levantamiento retardado de la tensión de salida. El fabricante recomienda utilizar un capacitor con capacitancia entre $2{,}5nF$ y $25nF$, y en este caso se utilizó un capacitor de $10nF$, obteniendo un retardo de $4ms$, realizando el cálculo a partir de la ecuación 6.7 con los valores $I_{SS} = 2\mu A$ y $V_{Ref} = 0{,}8V$.

$$T_{ss}(ms) = (C_{ss}(nF) * V_{Ref}(V))/I_{SS}(\mu A)] \tag{6.7}$$

6.2. Disipación - Medios Puentes H

Aquí se presenta el cálculo realizado para obtener la disipación de potencia que los medios puente H pueden presentar las condiciones de funcionamiento en este sistema. A raíz de los resultados obtenidos en esta estimación, es que se decidió no incluir un sistema de disipación de calor en el sistema.

Los medio puentes H utilizados en este sistema son los chips CSD87352q5d[4] de Texas Instruments, y en su hoja de datos se especifica que el valor de la resistencia térmica de su encapsulado es $R_{\Theta JA} = 82^oC/W$. Este valor, según se detalla, está determinado en condiciones de trabajo sobre un PCB de $1"^2$, mientras que las dimensiones del PCB de este circuito son 13,0x5,5cm (5,118x2,165"), o sea $11{,}08"^2$, hecho que redunda en una mayor superficie de disipación de calor y una consecuente disminución del en el valor de $R_{\Theta JA}$. Sin embargo, el PCB de este sistema reúne 3 medios puente H, por lo que se tomó el valor de $R_{\Theta JA}$ brindado por la hoja de datos para realizar los cálculos de disipación, evaluando así un peor caso.

En la hoja de datos se especifica, también, una temperatura máxima de operación de la juntura de 125^oC, aunque la temperatura máxima que soporta sean 150^oC. A su vez, para evaluar un caso de exigencia, se tomó una temperatura ambiente de 40^oC. Entonces, a partir de los datos del fabricante, las suposiciones realizadas y la ecuación 6.8, se puede calcular la potencia máxima disipada por el encapsulado: $P_{max} = 1{,}04W$.

$$P_{max} = [T_{Jmax} - T_A]/R_{\Theta JA} = [125 - 40]/82 \tag{6.8}$$

A su vez, la potencia disipada por el encapsulado se compondrá de por la potencia disipada por el MOSFET superior, más la potencia disipada por el MOSFET inferior del puente. Y la potencia que disipa cada MOSFET, se puede separar en tres aportes:

1. Potencia de MOSFET encendido
2. Potencia de MOSFET apagado
3. Potencia de conducción

Para este sistema, la potencia disipada por el MOSFET encendido se calculó a partir de la ecuación 6.9, y las siguientes características:

- t_r - Tiempo de encendido del MOSFET (obtenido de la hoja de datos)

- E - Tensión de alimentación de los puentes (12V)
- I_O - Corriente de régimen alcanzada por el MOSFET en conducción ($2A$)
- f - Frecuencia de switcheo del puente ($205kHz$, tomando la máxima frecuencia a la que funciona el sistema)

$$P_{MOSFETon} = (1/2) * [t_r * E * I_O * f] \tag{6.9}$$

MOSFET superior	**MOSFET inferior**
$t_r = 11ns$	$t_r = 7ns$
$P_{MOSFETon} = 27mW$	$P_{MOSFETon} = 17mW$

Tabla 6.1: Potencia disipada por ambos MOSFET del puente H en estado ON.

Mientras que la potencia disipada por el MOSFET apagado se calculó a partir de la ecuación 6.10, y las siguientes características:

- t_f - Tiempo de apagado del MOSFET (obtenido de la hoja de datos)
- E - Tensión de alimentación de los puentes (12V)
- I_O - Corriente de régimen alcanzada por el MOSFET en conducción ($2A$)
- f - Frecuencia de switcheo del puente ($205kHz$, tomando la máxima frecuencia a la que funciona el sistema)

$$P_{MOSFEToff} = (1/2) * [t_r * E * I_O * f] \tag{6.10}$$

MOSFET superior	**MOSFET inferior**
$t_r = 2ns$	$t_f = 2{,}7ns$
$P_{MOSFETon} = 4{,}9mW$	$P_{MOSFETon} = 6{,}6mW$

Tabla 6.2: Potencia disipada por ambos MOSFET del puente H en estado OFF.

Por último, la potencia disipada por el MOSFET en conducción se calculó a partir de la ecuación 6.11, y las siguientes características:

- R_{DSon} - Resistencia del MOSFET en conducción (obtenido de la hoja de datos)[2]
- I_O - Corriente de régimen alcanzada por el MOSFET en conducción ($2A$)
- δ - Ciclo de trabajo (se tomó peor caso: 50 %)

[2]Se consideró el MOSFET funcionando a $125°C$, por lo que resistencia encontrada en la hoja de datos para $25°C$ hay que multiplicarla por 1,3, parámetro tomado de la hoja de datos.

$$P_{cond} = R_{DSon} * {I_O}^2 * \delta \tag{6.11}$$

MOSFET superior	**MOSFET inferior**
$R_{DSon} = 9 * 1{,}3 = 11{,}7m\Omega$	$R_{DSon} = 2{,}8 * 1{,}3 = 3{,}6m\Omega$
$P_{cond} = 23{,}4mW$	$P_{cond} = 7{,}28mW$

Tabla 6.3: Potencia disipada por ambos MOSFET del puente H en conducción.

Para todas las evaluaciones se tomó $I_O = 2A$ de forma de considerar el peor caso, aunque en la práctica esta corriente efectivamente sea menor. Partiendo de estos cálculos se estimó una disipación de potencia total del encapsulado igual a $P_{Total} = 86{,}2mW$, muy por debajo del valor de la potencia máxima $P_{max} = 1{,}04W$, calculado en la ecuación 6.8. A raíz de estos resultados es que se tomó la decisión de no agregar un disipador para los medios puentes H del sistema, de forma de ahorrar en el tamaño y el costo del sistema.

Capítulo 7

Anexo B - Profundización Software

A lo largo de esta documentación se presentaron los diseños Statechart realizados para este sistema. Este tipo de diseño, presenta la ventaja de tener una representación gráfica modularizable y de fácil entendimiento, por lo que con una explicación a alto nivel de los objetivos de cada máquina y los diagramas correspondientes, se logra la comprensión de su funcionamiento. Sin embargo, el código generado por la aplicación utilizada resulta de una complejidad avanzada, sobre todo por la codificación utilizada por el programa que utiliza identificadores propios, eliminando los nombres de estados y máquinas asignadas en el proceso de generación de código. Por esto, en la siguiente sección (Explicación diseños StateChart 7.1) se desarrollan los conceptos detrás de los diseños Statecharts realizados para el sistema, profundizando en la explicación de los objetivos de las transiciones, los estados y las acciones definidas en los diseños, intentando dejar más claro el diseño del controlador.

Luego, en la sección Código - Funciones Centrales (7.2), se presentan las funciones implementadas en código *C* a los efectos de lograr la adaptación del código genérico con la plataforma del controlador del sistema.

7.1. Explicación diseños StateChart

7.1.1. CtrlBlock

La máquina CtrlBlock sensa la superficie de carga, establece un medio para la comunicación y regula la transmisión de potencia hacia el receptor, en función de mensajes enviados por el propio receptor y detectados por CommBlock . Para llevar a cabo estas tareas, se realizan transiciones entre los estados *Selection*, *Ping*, *IdentificationConfiguration* y *PowerTransfer*.

Selection

La máquina CtrlBlock inicia su funcionamiento en el estado Selection donde sensa la superficie por dispositivos a cargar. Sale de este estado al detectar una corriente menor a la esperada en vacío, para una frecuencia cercana a la resonancia, lo que significa que existe acoplamiento electromagnético. Se regresa siempre a este estado en caso de que

ocurra un error (incumplimiento de tiempos, mensaje erróneo o no apto para el estado donde se recibe, etc.) o bien si se terminó con el proceso de carga. Además, se restringe la carga en caso que haya habido tres intentos fallidos de carga. Para realizar estas acciones, Selection cuenta con dos submáquinas: *Selection_Ctrl y Selection_Bob*.

Selection_Ctrl

Esta máquina, mostrada en la Fig. 4.5(a), genera pulsos cortos a una frecuencia cercana a la de resonancia del sistema y luego sensa la corriente mediante el ADC. Posteriormente, en función del valor obtenido, setea variables y genera la transición hacia Ping o no.

S_Espero:

La máquina *Selection_Ctrl* incia su funcionamiento en este estado seteando un timer de espera entre sensados de corriente. Luego de este tiempo, pasa al estado *S_PingAnalogico*.

S_PingAnalogico:

Se energiza la bobina mediante una señal PWM a una frecuencia cercana a la de resonancia. Se setea un timer para asegurar que el pulso de excitación sea suficientemente largo para detectar cambios en la corriente si hay o no acoplamiento, pero a su vez suficientemente corto para no dañar el circuito por las altas tensiones y corrientes. Una vez que expira el timer, se apaga la onda y se pasa al estado *S_LeoI* seteando dos timers. Uno de ellos representa el tiempo máximo permitido para obtener la corriente y la función del otro (*Tcbs*) es dar un tiempo suficiente para que el sensor de corriente esté listo para volver a sensar.

S_LeoI:

Se lee la corriente mediante el valor del ADC. Si la corriente es mayor o igual a una corriente umbral, se considera que no hay dispositivos a cargar o que expiró el tiempo de lectura y se va a la estado *S_CambioBobina*. Si la corriente es menor al umbral, pero hubo tres intentos fallidos, se pasa al estado *S_EsperoNoDispositivo*. Por último, si la corriente es menor al umbral y no hubo tres intentos fallidos se genera una transición hacia el estado *Ping*.

S_CambioBobina:

Se le da la orden a *Selection_Bob* para que cambie de bobina y se espera a que expire el timer *Tcbs*. Si no se chequearon las 3 bobinas, se vuelve al estado *S_PingAnalogico*. En cambio, si las 3 bobinas ya fueron chequeadas, se discrimina entre dos situaciones: la primera es que haya habido 3 intentos fallidos de carga, por lo que se pasa al estado *S_NoDispositivo*; la segunda posible situación sería que se hayan chequeado las 3 bobinas pero que no haya habido 3 intentos fallidos de carga, por lo que se vuelve al estado *S_Espero* y se vuelve a iniciar el loop de sensado.

S_No dispositivo:

Mediante el uso de una variable se corrobora que efectivamente no haya dispositivo exigiendo que el sistema pase dos veces por este punto antes de volver a cero la traba de los tres intentos fallidos.

S_EsperoNoDispositivo:

Este estado mantiene un loop cerrado yendo hacia *S_Espero* e impidiendo que se salga del estado *Selection_Ctrl* hasta que en *S_NoDispositivo* se setee en cero la variable que representa los tres intentos fallidos.

Selection_Bob

En esta máquina hay un loop cerrado que pasa por tres estados análogos llamados *S_Bobina1*, *S_Bobina2* y *S_Bobina 3*. Las transiciones se producen mediante eventos generados en *Selection_Ctrl*, particularmente en *S_CambioBobina*. Cada estado tiene una variable asociada que permite discriminar entre las bobinas para realizar el PWM.

Ping

Una vez detectado un posible dispositivo a cargar, se realiza la transición hacia el estado Ping donde se espera recibir un mensaje (proveniente de CommBlock), relacionado al acoplamiento entre el trasmisor de potencia y el receptor. Es en Ping donde se determina mediante qué bobina se realizará la carga. En este estado hay dos transiciones posibles: si se logró determinar exitosamente qué bobina posee mayor acoplamiento y no hubo errores, se sigue al estado IdentificationConfiguration; si, por el contrario, hubo un error en las comunicaciones o no se encuentra un dispositivo para cargar, se vuelve al estado Selection.

Ping_Ctrl

Esta máquina genera una señal a 175kHz en una de las bobinas y espera una respuesta por parte del receptor. Ya sea que se haya recibido el mensaje esperado o se haya vencido el tiempo de espera, se cambia de bobina y se vuelve a setear la onda a 175kHz. Luego de haber energizado a las 3 bobinas, se determina cuál es la que posee mayor acoplamiento, analizando los mensajes recibidos, y se vuelve a energizar esta bobina pasando al estado de *IdentificationConfiguration*.

P_SeteoOnda:

Para lograr que el sensor de corriente tenga, como mínimo, el mismo tiempo desde que se energiza una bobina hasta que se energiza la siguiente, se setea un timer a la entrada de este estado y cuando el mismo expira, se setea la onda. Este mismo timer es utilizado en el estado *P_CambioBobina*. Luego de setear la onda se pasa al estado *P_PingDigital*.

P_PingDigital:

Se lee la corriente mediante la lectura del ADC y se encola un evento pasando al estado *P_ChequeoCorriente*.

P_ChequeoCorriente:

En este estado se compara la corriente sensada contra un umbral que representa la mitad del valor de la corriente estable cuando hay un dispositivo. En caso de que la corriente pase el umbral, se pasa al estado *P_EsperoMsj* y se setea un timer que

indica el tiempo máximo que se tiene hasta recibir el comienzo de un mensaje del receptor. Si no se pasa el umbral, se vuelve al estado *P_PingDigital* para tomar una nueva medida de corriente.

P_EsperoMsj:

En este estado se distinguen varias situaciones posibles, en caso de que expire el tiempo para recibir el inicio del mensaje y no se hayan sensado las tres bobinas, se pasa al estado *P_CambioBobina*. En caso de que haya expirado el tiempo para recibir el inicio del mensaje, se haya recibido al menos un mensaje para una de las bobinas y se haya chequeado las tres bobinas, se pasa al estado *P_SeleccionoBobina*. En caso de recibir un mensaje correcto por parte del receptor, se pasa al estado *P_InterpretoMsj*. Por último, ya sea que se reciba un mensaje incorrecto, expire el tiempo de espera para el inicio del mensaje para las tres bobinas o se haya recibido un mensaje de fin de transferencia de potencia, se vuelve al estado Selection.

P_CambioBobina:

Cuando se entra a este estado se ejecuta una acción que genera que el estado *Ping_Bob* cambie de bobina.

P_SeleccionoBobina:

Este estado ejecuta una acción que compara los niveles de acoplamiento declarados por el receptor, seleccionando la bobina que tenga mayor acoplamiento. Una vez que selecciona la bobina, setea una variable que permite el pasaje hacia el estado *IdentificationConfiguration*.

P_InterpretoMsj:

La función de este estado es generar transiciones dependiendo de qué tipo de mensaje haya llegado. Dado que para el estado *Ping* sólo hay dos mensajes posibles, las decisiones tomadas por este estado son muy básicas: se genera una transición hacia Selection si el mensaje es de fin de transferencia de potencia, o bien se guarda el valor si el mensaje es referido al acoplamiento.

Ping_Bob

Funcionamiento igual a *Selection_Bob*, con distintas señales de transición de estados.

Identification & Configuration

IdentificationConfiguration posee una sola máquina de estados llamada *IC* que recibe una sucesión de mensajes provenientes de *CommBlock* y, dependiendo del orden y valores de estos mensajes, se realiza un contrato de transferencia de energía el cual será monitoreado posteriormente en el proceso de carga propiamente dicho. Cuando el contrato de transferencia de potencia se realiza adecuadamente se genera la transferencia hacia el estado *PowerTransfer*.

IC_Espero:

Este es el estado inicial de esta máquina de estados y además es el estado central, ya que cada vez que llega un mensaje y se procesa se vuelve a este estado a esperar el siguiente mensaje. En caso que expire un timer de tiempo máximo entre mensajes o *CommBlock* determine que hubo un error, se genera una transición que vuelve al estado *Selection*.

IC_MsjIdent:

En caso de llegar a *IdentificationConfiguration* desde *Ping*, el primer mensaje esperado es un mensaje de identificación. Por este motivo, al recibir un mensaje de forma correcta se pasa del estado *IC_Espero* al estado *IC_MsjIdent*. Dado que el mensaje de identificación puede involucrar un mensaje de identificación extendida, uno de los campos del mensaje de identificación declara si se debe esperar o no uno de identificación extendida. En caso de que el mensaje haya sido efectivamente un mensaje de identificación y que se deba esperar un mensaje de configuración extendida, se vuelve al estado *IC_Espero* seteando una variable que forzará que el próximo mensaje recibido de forma exitosa lleve a la máquina hacia el estado al estado *IC_MsjIdentExt*. Si no se debe esperar un mensaje de identificación extendida o se llegó a *IdentificationConfiguration* desde *PowerTransfer*, el próximo mensaje esperado es un mensaje de configuración o mensajes de configuración propietarios, por lo que se vuelve al estado IC_Espero seteando una variable que forzará que el próximo mensaje recibido de forma exitosa lleve a la máquina hacia el estado *IC_MsjConf*. Si se recibe un mensaje diferente a un mensaje de identificación o expira el tiempo máximo entre mensajes, se vuelve al estado de *Selection*.

IC_MsjIdentExt:

Como ya fue mencionado en *IC_MsjIdent*, no necesariamente se debe pasar por este estado en todos los procesos de carga, sino que depende exclusivamente del receptor. En caso de haber llegado al estado *IC_MsjIdentExt* y recibir efectivamente un mensaje de identificación extendida, se setea la variable que conduce a *IC_MsjConf* y se vuelve a *IC_Espero*. En caso de que se reciba un mensaje diferente o que expire el tiempo máximo entre mensajes, se vuelve al estado *Selection*.

IC_MsjConf:

En este estado se pueden recibir varios tipos de mensajes propietarios, algunos de los cuales no tienen incidencia para el transmisor y otros sí la tienen. El único

mensaje que siempre debe recibirse es el mensaje de configuración con el cual se genera el contrato de transferencia de potencia y, por ende, es el más relevante de los mensajes esperados. A medida que se van recibiendo de forma correcta los mensajes propietarios, se los va contando y se almacena los datos útiles. Los mensajes propietarios pueden ser como máximo 7 y, además, deben coincidir con la cantidad declarada en uno de los campos del mensaje de configuración (que es el último mensaje recibido en *IdentificationConfiguration*).

Power Transfer

El estado *PowerTransfer* consta de una única máquina llamada *PT* que realiza la transferencia de potencia propiamente dicha, regulándola mediante mensajes de *ControlError* y chequeando los parámetros contenidos en el contrato de transferencia de potencia. En caso de que alguno de esos parámetros sea violado se vuelve a *Selection*.

PT_Espero:

Al igual que en *IC*, esta máquina empieza esperando mensajes y realizando acciones a raíz de los mensajes recibidos. Una vez recibido un mensaje correctamente, se pasa al estado *PT_InterpretoMsj*. Si se recibe un mensaje erróneo, se vuelve al estado *Selection*.

PT_InterpretoMsj:

En este estado se compara el mensaje recibido con una lista de mensajes posibles. Se observan los parámetros incluidos en el contrato de transferencia de potencia verificando que no se violen los límites del mismo. En caso de que no ocurran violaciones a los parámetros se vuelve al estado *PT_Espero*. Si, por el contrario, se halla una violación hacia esos parámetros o el mensaje recibido es un mensaje de fin de transmisión de potencia, se genera una transición hacia el estado *Selection*. En caso de que el mensaje recibido sea un mensaje de *Control Error*, se pasa al estado *PT_EsperoTdelay*. Por último, si el mensaje recibido es un mensaje de reconfiguración, se vuelve al estado *IdentificationConfiguration* para renegociar los parámetros del contrato de transferencia de energía.

PT_EsperoTdelay:

En este estado se espera el tiempo t_{delay} permitiendo que la señal llegue a un valor estable para su sensado y modificación. Una vez expirado este tiempo, se pasa al estado *PT_CorrienteActual* iniciando un timer (t_{active}) que expresa el tiempo máximo para la corrección de la corriente actual hacia el valor pedido en el mensaje de *Control Error*.

PT_CorrienteActual:

El estado *PT_CorrienteActual* y el estado *PT_ModificoI* conforman un PID. El estado *PT_CorrienteActual* sensa la corriente y inicializa un timer que establece los límites temporales para la corrección de un paso del PID. Luego de leer la corriente se pasa al estado *PT_ModificoI* donde se realiza una iteración del PID y se vuelve al estado *PT_CorrienteActual*. Este loop continúa hasta que se realicen los

pasos establecidos del PID. En caso que expire el timer t_{active}, se vuelve al estado *Selection*.

PT_ModificoI:

Además de lo mencionado en *PT_CorrienteActual*, *PT_ModificoI* lleva un control de los pasos del PID efectuados. Si se cumplieron las restricciones de tiempo y se llegó al número de iteraciones, se pasa al estado *PT_EsperoTsettle*. Mientras que no se complete la cantidad de iteraciones necesarias, luego de realizar la modificación de la corriente correspondiente al paso del PID, se vuelve al estado *PT_CorrienteActual*. Si en el transcurso de alguna de las iteraciones expira el timer dedicado a controlar el tiempo por iteración o el timer t_{active}, se genera una transición a *Selection*.

PT_EsperoTsettle:

Este estado tiene como función permitir que la corriente llegue a un valor estable para volver a ser sensada. Para esto, activa un timer y cuando este expira, se genera una transición hacia el estado *PT_CorrienteFinal*.

PT_CorrienteFinal:

Este estado realiza una última lectura de corriente y luego genera una transición al estado *PT_Espero*.

7.1.2. CommBlock

La máquina *CommBlock* presenta un diseño extremadamente sencillo debido a las razones expuestas en el capítulo Software, sección Diseño, CommBlock (4.2.2). Este inicia su funcionamiento en el estado *Reposo* esperando por un flanco[1] en el pin asignado para recibir la señal de comunicación. Al recibir el evento que identifica esta excitación, desencadena la transición de estado hacia *DecoMensaje* y comienza la recepción de un nuevo paquete de datos a través de la función *aCOMMReceptor()* en el estado *Receptor* de la submáquina. El resto del sistema está a la espera del fin del procesamiento del paquete, por lo que sólo se da una nueva transición de estado cuando el sistema toma el evento encolado por esta función *eMSJGuardado*. En esta transición se lleva a la submáquina al estado *Validator* en caso de no haberse dado ningún error en la comunicación (paridad, señal colgada, etc.), o se va a *EndOfMessage* en caso contrario. Una vez en el estado *Validator*, se corrobora la validez del tipo de paquete recibido (identificado por su header en la variable *veHeader*) para el estado en que se encuentra *CtrlBlock* y se verifica el checksum, para devolver la máquina al estado *Reposo* encolando el evento de fin de mensaje *EOM* con las banderas de validez de mensaje correspondientes (*veValidPckg* y *veMsgReady*) seteadas en función de los resultados.

[1] La implementación para esta recepción se explica en el capítulo Software, Funciones Centrales, Sincronización emisor-receptor (4.4.7).

7.2. Código - Funciones Centrales

Procesamiento de la señal de comunicación

```
extern VS_VOID aCOMMReceptor(){
state = 0;
bitsPre=0;

veBitsLeft=0;
veCeroUno=1;
veBytesLeft=0;
veByteDeMensaje=0;
veByteActual=0x00;
CommOK=1;
flancoPin=0;
viPreValido=0;

int tran=1;
uint8_t varByte;
int CommOnCourse=1;

for(int j=0; j8; j++){ //Inicialización del vector de bytes de mensajes
Mensaje[j] = 0x00;
}

if ((Estado==212) —— (Estado==310) —— (Estado==410)){
while (CommOK==1 & CommOnCourse==1){ //Mientras no haya error de comunicación, y falte información por procesar
switch (state){

case 0: ////////////////// PREÁMBULO ////////////////////
while (veCeroUno==1){
if (bitsPre  26){
while(flancoPin==0){} //Espero un nuevo flanco
flancoPin=0;
if (tran==1){ //Nivel de Transición de un 1
tran = 0;
if(veTbit=48){
veCeroUno = 1;
}else{
veCeroUno = 0;
}
viPreValido = (bitsPre =4);
}else{ //Primer nivel de un 1
tran = 1;
if (veTbit 48) {
SEQ_AddEvent(eErrorCOMM);
CommOK=0;
break;}
bitsPre++;
}
}else{ //Preámbulo muy largo, asumo que es ruido
SEQ_AddEvent(eErrorCOMM);
CommOK=0;
break;}
if(CommOK==0){
veCeroUno=0;
break;
}
}

if ((viPreValido==1) & (bitsPre25)){ //Cuando recibo un 0, continúo según validez del preámbulo
state = 1;
veBitsLeft = 8; //Se procesan 8 bits
TimersDriver(Estado);
}else{
SEQ_AddEvent(eErrorCOMM);
CommOK=0;
}
break;

case 1: ////////////////// HEADER ////////////////////
//El StartBit lo procesó el estado de preámbulo
viParidad=1;
veByteActual=0;
while(veBitsLeft 0){
varByte=veByteActual;
veByteActual=procesaBit(veByteActual); //Arma el byte con una función específica
if ((veByteActual/varByte)!=2){
viParidad=(!viParidad);}
veBitsLeft–;
}
veHeader = aMSJRevert(veByteActual); //Guarda header en memoria, invirtiendo bits del byte en proceso
while(flancoPin==0){} //Espero bit de paridad
flancoPin=0;
if (veTbit=48){ //Si es 1, establezco bandera de bit de paridad
viBitP=1;
while(flancoPin==0){} //Es 1, así que espero un flanco para ir al stopbit
flancoPin=0;
}else{ //Si es 0, reseto bandera de bit de paridad
viBitP=0;
}
viByteValido = (viBitP==viParidad);
```

```
while(flancoPin==0){} //Espero flanco de stopbit
flancoPin=0;
if (viByteValido==0){
SEQ_AddEvent(eErrorCOMM);
CommOK=0;
break;
}
state = 2;
while(flancoPin==0){} //Espero segundo flanco de stopbit
flancoPin=0;
veBytesLeft = cantidadPaquetes(veHeader); //Cálculo del largo del paquete a partir del header
CommOK=1;
break;

case 2: ////////////////// MENSAJES ////////////////////
while(veBytesLeft 0){ //Mientras queden bytes del paquete
viParidad=1;
veByteActual=0;
veBitsLeft=8;
while(flancoPin==0){} //Espero flanco de startbit
flancoPin=0;
while(veBitsLeft 0){
varByte=veByteActual;
veByteActual=procesaBit(veByteActual);
if ((veByteActual/varByte)!=2){
viParidad=(!viParidad);}
veBitsLeft−;
}
Mensaje[veByteDeMensaje] = aMSJRevert (veByteActual); //Guarda byte actual inviertido, como byte de mensaje
while(flancoPin==0){} //Espero bit de paridad
flancoPin=0;
if (veTbit=48){ //Si es 1, establezco bandera de bit de paridad
viBitP=1;
while(flancoPin==0){} //Es 1, así que espero un flanco para ir al stopbit
flancoPin=0;
}else{ //Si es 0, reseto bandera de bit de paridad
viBitP=0;
}
viByteValido = (viBitP==viParidad);
while(flancoPin==0){} //Espero flanco de stopbit
flancoPin=0;
if (viByteValido==0){
SEQ_AddEvent(eErrorCOMM);
CommOK=0;
break;
}
while(flancoPin==0){} //Espero segundo flanco de stopbit
flancoPin=0;
veByteDeMensaje++;
veBytesLeft−;
}
state = 3;
CommOK=1;
break;

case 3: ////////////////// CHECKSUM ////////////////////
while(flancoPin==0){} //Espero flanco de startbit
flancoPin=0;
veBitsLeft=8;
veByteActual=0;
viParidad=1;
while(veBitsLeft 0){
varByte=veByteActual;
veByteActual=procesaBit(veByteActual);
if ((veByteActual/varByte)!=2){
viParidad=(!viParidad);}
veBitsLeft−;
}
while(flancoPin==0){} //Espero bit de paridad
flancoPin=0;
if (veTbit=48){ //Si es 1, establezco bandera de bit de paridad
viBitP=1;
while(flancoPin==0){} //Es 1, así que espero un flanco para ir al stopbit
flancoPin=0;
}else{ //Si es 0, reseto bandera de bit de paridad
viBitP=0;
}
viByteValido = (viBitP==viParidad);
while(flancoPin==0){} //Espero flanco de stopbit
flancoPin=0;
if (viByteValido==0){
SEQ_AddEvent(eErrorCOMM);
CommOK=0;
break;
} veChecksum=aMSJRevert(veByteActual);
while(flancoPin==0){} //Espero segundo flanco de stopbit
flancoPin=0;
state = 4;
SEQ_AddEvent(eMsjGuardado);
CommOK=1;
break;

case 4: //Fin de recepción. Encolo evento y bajo bandera de comunicación.
CommOnCourse = 0;
```

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

}
```

Procesador de Bits

```
extern uint8_t procesaBit(uint8_t byte){

uint8_t aux=0;
uint8_t Byte=byte;
while(flancoPin==0){} //Espero una iterrupción que me indique un cambio en la señal de comunicaciones
flancoPin=0;
if(veTbit=45){
veCeroUno = 1;
}else{
veCeroUno = 0;
}
if(veCeroUno==1){
aux = (Byte*2);
Byte = aux+1;
while(flancoPin==0){} //Espero una iterrupción que me indique un cambio en la señal de comunicaciones
flancoPin=0;
if(veTbit 45){
SEQ_AddEvent(eErrorCOMM);
CommOK=0;
}
}else{
aux = (Byte*2);
Byte = aux;
}
return Byte;
}
```

Lectura de Corriente

```
extern uint16_t LecturaADC(void){
uint16_t temph = 0;
uint16_t tempanterior=0;
uint8_t i=0;
while (i≤15){
i++;
ADC1_Init(ADC1_CONVERSIONMODE_SINGLE,ADC1_CHANNEL_1,
ADC1_PRESSEL_FCPU_D4,ADC1_EXTTRIG_TIM,DISABLE,
ADC1_ALIGN_RIGHT,ADC1_SCHMITTTRIG_CHANNEL1, DISABLE);
ADC1_StartConversion();
while (!(ADC1->CSR &0x80));// wait for end of conversion
temph = ADC1_GetConversionValue();
ADC1->CSR& =∼ ADC1_CSR_EOC; // clear end of conversion flag
ADC1->CR1& =∼ ADC1_CR1_ADON;
if (temph>tempanterior){
tempanterior=tempanterior+temph;
}
}
tempanterior=tempanterior/16;
ADC1->CSR& =∼ ADC1_CSR_EOC; // clear end of conversion flag
ADC1->CR1& =∼ ADC1_CR1_ADON; // stop ADC
return (tempanterior);

}
```

Intérpretes de mensajes

Ping

```
extern VS_VOID aInterpretoMsjPing (VS_VOID){
if (veHeader == 0x01){
veIntentosFallidos = 0;
veMsjSignalStrength = 1;
if (veBCP==1){
SignalStrengthBobina1 = Mensaje[0];
} else if (veBCP==2){
SignalStrengthBobina2 = Mensaje[0];
} else if (veBCP==3){
SignalStrengthBobina3 = Mensaje[0];
};
} else if (veHeader == 0x02){
veIntentosFallidos = veIntentosFallidos + 1;
veMsjSignalStrength=0;
} else {
SEQ_AddEvent(eMsjEndPT);
}
```

```
};
```

Identification & Configuration:

```
extern VS_VOID aInterpretoMsjIC (VS_UINT8 OrdenIC){
switch (OrdenIC){
case 1:
if (veHeader == 0x71){
MsjID =(Mensaje[3] & 0x80);
if (MsjID == 0x80){
veOrdenIC = 2;
SEQ_AddEvent(eGuardeMsj);
} else {
veOrdenIC = 3;
SEQ_AddEvent(eGuardeMsj);
};
} else if (veHeader == 0x2){
veIntentosFallidos = veIntentosFallidos + 1;
SEQ_AddEvent(eMsjEndPT);
} else {
SEQ_AddEvent(eHayError);
};
break;
case 2:
if (veHeader == 0x81){
veOrdenIC = 3;
SEQ_AddEvent(eGuardeMsj);
} else if (veHeader == 0x2){
veIntentosFallidos = veIntentosFallidos + 1;
SEQ_AddEvent(eMsjEndPT);
} else {
SEQ_AddEvent(eHayError);
};
break;
case 3:
if (veHeader == 0x06){
if (MsjsConfig < 7){
MsjsConfig = MsjsConfig + 1;
Tdelay = Mensaje[0];
SEQ_AddEvent(eGuardeMsj);
} else {
SEQ_AddEvent(eHayError);
};
} else if (veHeader == 0x51) {
CantMsjsConfig = (Mensaje[2] & 0x07);//Msj_Conunt B2, b2 b1 b0
if (CantMsjsConfig == MsjsConfig) {
veMsjConfiguracion=1;
SEQ_AddEvent(eGuardeMsj);
} else {
SEQ_AddEvent(eHayError);
};
} else if (veHeader == 0x02){
veIntentosFallidos = veIntentosFallidos + 1;
SEQ_AddEvent(eMsjEndPT);
} else if ((veHeader == 0x71) || (veHeader == 0x81)){
SEQ_AddEvent(eHayError);
} else {
if (MsjsConfig < 7){
MsjsConfig = MsjsConfig + 1;
SEQ_AddEvent(eGuardeMsj);
} else {
SEQ_AddEvent(eHayError);
};
};
break;
default:
SEQ_AddEvent(eMsjEndPT);
break;
};
};
```

Power Transfer:

```
extern VS_VOID aInterpretoMsjPT (VS_VOID){
if (veHeader == 0x04){
RecivedPower = Mensaje[0];
SendPower = 12 * CorrienteActual;
if ((SendPower-RecivedPower)¿UMBRAL){
veIntentosFallidos = veIntentosFallidos + 1;
SEQ_AddEvent(eMsjEndPT);
} else {
SEQ_AddEvent(eMsjOtro);
};
} else if (veHeader == 0x03){
MsjError = Mensaje[0];
SEQ_AddEvent(eMsjCE);
} else if (veHeader == 0x02){
if (Mensaje[0] == 0x07){
SEQ_AddEvent(eMsjReconfig);
} else {
```

```
veIntentosFallidos = veIntentosFallidos + 1;
SEQ_AddEvent(eMsjEndPT);
};
} else {
SEQ_AddEvent(eMsjOtro);
};

};
```

PID

```
extern VS_VOID aModificoI (VS_VOID){
if (veIteracion == 1){
CorrienteNueva = CorrienteActual*(1+MsjError/128);
IteracionAnterior = 0;
};
ErrorEnI = CorrienteNueva-CorrienteActual;
P = Kp*ErrorEnI;
I = IteracionAnterior + Ki * ErrorEnI * Tinner;
if (I>M){
I = M;
} else if (I<-M){
I = -M;
};
IteracionAnterior = I;
PID = P+I;
if (PID>MPID){
PID = MPID;
} else if (I<-MPID){
PID = -MPID;
};
if (Frec < (205)){
DC = 0.5; // va 0.5
if ((Frec ≥ 115) && (Frec < 140)){
Sv = 1.5; // va 1.5
} else if ((Frec ≥ 140) && (Frec < 160)){
Sv = 2; // va 2
} else if ((Frec ≥ 160) && Frec < 180){
Sv = 3; // va 3
} else if ((Frec ≥ 180) && (Frec < 205)){
Sv = 5; // va 5
};
Frec = Frec-Sv*PID;
if (Frec > 205){
Frec = 205;
};
} else if (Frec == 205){
DC = DC-0.01*PID;
};
SetPWMBobina (Frec,DC,veBCP);
SEQ_AddEvent(eTermineCuentas);
};
```

Manejo de Bobinas

PWM:

```
void SetPWMBobina (float Frec,float DutyCicle,uint8_t Bobina){
float Periodo = 0;
uint8_t Flanco = 0;
Periodo = roundf(16000/Frec);
Flanco = (uint8_t)roundf(Periodo * DutyCicle);
TIM3_DeInit();
/* Set TIM3 Frequency to 16Mhz y el periodo de PWM en 175kHz */
TIM3_TimeBaseInit(TIM3_PRESCALER_1,(uint8_t)(Periodo));
/* Channel 1 PWM configuration @brief Initializes the TIM3 Channel1 according to the specified parameters. */
TIM3_OC1Init(TIM3_OCMODE_PWM1, TIM3_OUTPUTSTATE_ENABLE,Flanco, TIM3_OCPOLARITY_LOW );
TIM3_OC1PreloadConfig(ENABLE);
/* Enables TIM3 peripheral Preload register on ARR */
TIM3_ARRPreloadConfig(ENABLE);
TIM3_Cmd(ENABLE);
/*Seteo bobina*/
EnableBobina(Bobina);
}
```

Ping Analógico - Recorrido de Bobinas:

```
extern VS_VOID aSeteoPingAnalogico (VS_UINT8 SeteoPingAnalogico){
if (SeteoPingAnalogico == 1){
Frec = 140; //Cercano a frecuencia de resonancia
DC = 0.5;
SetPWMBobina (Frec,DC,veBCS);
}else{
GPIO_WriteLow(GPIOB,GPIO_PIN_3);
GPIO_WriteLow(GPIOB,GPIO_PIN_4);
```

```
GPIO_WriteLow(GPIOB,GPIO_PIN_5);
};

};
```

Ping Digital - Recorrido de Bobinas:

```
extern VS_VOID aSeteoPingDigital (VS_VOID){
Frec = 175; // va 175
DC = 0.5; // va 0.5
SetPWMBobina (Frec,DC,veBCP);
GPIO_WriteHigh(GPIOD,GPIO_PIN_4);

}
```

Ping - Selección por acople:

```
extern VS_VOID aSeleccionoBobina (VS_VOID){
if ((SignalStrengthBobina1 ≥ SignalStrengthBobina2) && (SignalStrengthBobina1 ≥ SignalStrengthBobina3)){
veBCP=1;
} else if ((SignalStrengthBobina2 ≥ SignalStrengthBobina1) && (SignalStrengthBobina2 ≥ SignalStrengthBobina3)){
veBCP=2;
} else {
veBCP=3;
};

};
```

Habilitación de Bobinas:

```
void EnableBobina(uint8_t Bobina){
switch ( Bobina ) {
case 1:
GPIO_WriteHigh(GPIOB, GPIO_PIN_3);
break;
case 2:
GPIO_WriteHigh(GPIOB, GPIO_PIN_4);
break;
case 3:
GPIO_WriteHigh(GPIOB, GPIO_PIN_5);
break;
}

}
```

Timers

Función de seteo del timer con la cantidad de ticks dada como parámetro, y que encolará el evento también dado como parámetro:

```
extern void SetTXXXX (SEM_EVENT_TYPE event, unsigned int ticks){
uint16_t C = TIM#_GetCounter();
TIM#_SetCompare*2(C + ticks);
TIM#_OC2PreloadConfig(ENABLE);
T#C*=event;

}
```

Función de detención de timer:

```
extern void SetTXXXX_stop (void){
TIM#_OC*PreloadConfig(DISABLE);
}
```

En ambos casos el símbolo # indica el timer a manejar (el μC utilizado tiene timers TIM1, TIM2, TIM3 y TIM4), y el símbolo $*$ indica el canal del timer sobre el que se quiere accionar (4 canales TIM1, 3 canales TIM2, 1 canal TIM3 y TIM4).

Sincronización

```
INTERRUPT_HANDLER(EXTI_PORTB_IRQHandler, 4){
uint16_t TiempoActual=TIM1_GetCounter();
uint8_t Cuenta;
Cuenta=TIM4_GetCounter();
if (HuboFlanco==1){
if (Cuenta>15){ //esto es 1/4 de período y se hace para evitar espurios
veTbit=Cuenta;
SEQ_AddEvent(eFlancoPin);
}
}
else {
if(TiempoAnterior!=TiempoActual){//Limpio en caso de que lleguen muchos espurios
SEQ_AddEvent(eFlancoPin);
}
}// Aca hay que ver cuando se recibe el primer bit y se tiene el timer apagado
TiempoAnterior=TiempoActual;
}
INTERRUPT_HANDLER(TIM4_UPD_OVF_IRQHandler, 23){
if(T4==eTbitCTRL){
veVP=GPIO_ReadInputPin(GPIOB, GPIO_PIN_0);
SEQ_AddEvent(eTbitCTRL);
GPIO_Init(GPIOB, GPIO_PIN_0, GPIO_MODE_IN_PU_IT);// se desactivan las
//interrupciones por flanco en puerto de comunicaciones
}else if(T4==eBitPre){
SEQ_AddEvent(T4);
}
TIM4_ClearITPendingBit(TIM4_IT_UPDATE);
TIM4_Cmd(DISABLE);
HuboFlanco=0;
}
```

Capítulo 8

Anexo C - Evaluación de costos

A los efectos de evaluar la viabilidad de la producción comercial de dispositivos de este tipo, es de suma importancia estimar los costos fijos que su fabricación tendría. Es un hecho que los costos que un prototipo tiene distan bastante de los costos que una producción a mediana escala tendrá, dada la diferencia en las cantidades requeridas para uno y otro caso y la notoria reducción del costo por unidad al aumentar la cantidad. Sin embargo, esta estimación puede servir como punto de partida para la evaluación de una inversión primaria que una producción con fines comerciales requeriría. Con este objetivo, en la tabla 8.1 se presenta una lista de los componentes hardware del sistema.

Claramente, otro de los puntos sensibles para el cálculo del costo de un sistema de este tipo son las horas hombres requeridas para el ensamblado y terminaciones que un cargador de estas características requeriría. Y se suma a esto las horas de dedicación necesaria para las terminaciones del diseño software, así como las mejoras y el mantenimiento que este requeriría. Pero todos estos son costos variables y que pueden ser considerados como trabajo base en caso de apuntar a una *start-up* a partir de este sistema, por lo que no se considerarán estos puntos como un costo asociado.

Relevando los costos de los componentes adquiridos, mostrados en la tabla 8.1, a excepción de los elementos inductivos que fueron muestras gratis enviadas por Wurth Electronik, se estima un costo total de $U\$s16{,}58$. A su vez, el costo de las muestras recibidas es de $U\$s2{,}27$ la unidad de inductores y $U\$s31{,}25$ el bobinado *A6*. Respecto a este punto, cabe mencionar que existen opciones menos costosas de otros fabricantes: bobinado TDK $U\$s16{,}8$ la unidad y bobinado Abracon LLC $U\$s8{,}46$ la unidad; ambos de iguales características eléctricas que el bobinado utilizado y con blindaje para protección del circuito. Otro punto a considerar a este respecto es el costo del PCB, el cual ascendió a $U\$s6{,}6$ cada placa. Todos los costos aquí presentados son en origen, por lo que se deberá sumar costos de envío al costo total. En resumen, el costo hardware en origen asciende a $U\$s25{,}45$ **para el sistema desarrollado**, sin tener en cuenta el costo del bobinado. En este sentido, la opción más conveniente parece ser el bobinado Abracon LLC, que sumado a lo anterior resulta en un **costo total hardware de** $U\$s33{,}91$. Estimando la reducción de costos con la compra de grandes cantidades, y sumando costos de envío estimados, se puede calcular un costo total hardware aproximado de unos $U\$s50$.

Elemento	#	Descripción	Fabricante
Driver			
TPS28225	3	MOSFET synchronous Driver	Texas Instruments
LTST-S220KGFT	3	LED GREEN CLEAR RT ANG 0805	Lite-ON
R=200 ohm	3	Resistencia SMD	
C= 0,1 uf	3	Capacitor SMD	
C= 0,47 uf	3	Capacitor SMD	
Puente			
CSD87352Q5D	3	MOSFET 2N-CH 30V 25A 8SON	Texas Instruments
C=4,7 uf	3	Capacitor SMD	
Circuito resonante			
Bobinado 760308106	1	L=12,5μf; L=11,5μf; L=12,5μf	Würth Electronik
C= 0,068 uf	6	Capacitor SMD	
C=0,015 uf	1	Capacitor SMD	
Sensor de corriente			
LM324 ($*$)	1	1 de 4 Amp.Op. utilizados	Texas Instruments
R=1 kohm	4	Resistencia SMD	
R=2,95 kohm	2	Resistencia SMD	
R=10 ohm	2	Resistencia SMD	
R=1 Mohm	2	Resistencia SMD	
Rshunt=0,33 ohm	1		
R=1 kohm	1	Resistencia SMD	
C=0,1uf	2	Capacitor SMD	
CL21F105ZAFNNNE	1	CAP CER 1UF 25V Y5V 0805	
Conversor DC/DC			
TPS54231	1	Conversor DC-DC StepDown	Texas Instruments
74477792700	1	L=270 μHy	Würth Electronik
R=200 ohm	1	Resistencia SMD	
R=120 kohm	1	Resistencia SMD	
R=1 Mohm	1	Resistencia SMD	
R= 5,9 kohm	1	Resistencia SMD	
R= 10 kohm	1	Resistencia SMD	
R= 1,91 kohm	1	Resistencia SMD	
C= 10 nf	2	Capacitor SMD	
C= 180 pf	1	Capacitor SMD	
C=0,1uf	1	Capacitor SMD	
C=2,2uf	2	Capacitor SMD	
C=4,7 uf	1	Capacitor SMD	
C=0,01 uf	1	Capacitor SMD	
LTST-S220KRKT	1	LED SUPR RED CLR RT ANG 0805	Lite-ON
B240A-FDICT-ND	1	DIODE SCHOTTKY 40V 2A SMA	Diodes Inc.

Demodulador			
LM324 (∗)	1	3 de 4 Amp.Op. utilizados	Texas Instruments
BAS16,215	3	DIODE GEN PURP 100V 215MA SOT23	NXP
R=10 kohm	5	Resistencia SMD	
R=36 kohm	1	Resistencia SMD	
R=62 kohm	1	Resistencia SMD	
R= 4,9 kohm	2	Resistencia SMD	
R= 22 kohm	2	Resistencia SMD	
R= 5,9 kohm	6	Resistencia SMD	
R= 4,3 kohm	1	Resistencia SMD	
R= 2,4 kohm	1	Resistencia SMD	
R= 1 Mohm	1	Resistencia SMD	
C= 47 nf	2	Capacitor SMD	
C= 1 nf	3	Capacitor SMD	
C=10 nf	1	Capacitor SMD	
C=0,1 uf	1	Capacitor SMD	
C=0,47 nf	1	Capacitor SMD	
C=0,01 uf	3	Capacitor SMD	
Microcontrolador			
STM8S105K6T6C	1	IC MCU 8BIT 32KB FLASH 32LQFP	ST Micorelectronics
LTST-S220KGFT	1	LED GREEN CLEAR RT ANG 0805	Lite-ON
R=510	1	Resistencia SMD	
C=0,01 uf	1	Capacitor SMD	
C=0,1 uf	2	Capacitor SMD	
C=2,2 uf	1	Capacitor SMD	

Tabla 8.1: Componentes hardware utilizados en el sistema.

Capítulo 9

Anexo D - Contratiempos Experimentados

9.1. Hardware

En relación con los contratiempos experimentados en lo que a hardware respecta, cabe mencionar que, en primera instancia, se pensó en utilizar la fresadora del IIE con el layout mostrado en la Fig. 3.6 y con una placa doble capa provista también por el Instituto para el prototipo de la etapa de potencia. Sin embargo, resultó imposible realizar esto tal como estaba planeado, dado que en las fechas previstas para el prototipado del proyecto, la fresadora se encontraba fuera de funcionamiento y a la espera de reparaciones. Como consecuencia, hubo que recurrir a un proveedor de PCB de plaza para solucionar este inconveniente. Con esto, se obtuvo la ventaja de contar con un PCB con máscara antisoldante que, a pesar de no ser de muy buena calidad, fue de utilidad a la hora de soldar los componentes. Sin embargo esto implicó un retraso considerable en el tiempo estipulado para comenzar a realizar pruebas sobre hardware.

Otro punto a tener en cuenta a la hora de evaluar los contratiempos experimentados se detalla extensamente en las Conclusiones (5.2), y refiere a la demora en los plazos que trae aparejada la producción hardware en nuestro país. Más aún en un proceso iterativo de diseño-corrección-rediseño, y con alguna posible retención de pedidos en aduana, caso en el cual se podrán tener meses de estancamiento en el avance del desarrollo.

Por último, y en relación con lo que se explicó en el capítulo Conclusiones y Resultados, Resultados (5.1), otro contratiempo se generó a raíz de la utilización de un amplificador operacional convencional en la implementación hardware del bloque sensor de corriente. Dado que el diseño de esta etapa del circuito se basó en un diseño que utiliza un tipo de amplificadores específicos para el sensado de corriente en un shunt, que trabaja con tensiones de entrada en modo común muy por encima de su tensión de alimentación, se derivó en un error de diseño que implicó una pérdida de tiempo considerable. En el diseño desarrollado el amplificador está alimentado con 5V, y su máximo valor de entrada de tensión en modo común es de Vcc - 1.5V según la hoja de datos. Sin embargo, por las características del circuito y la tensión que se necesitaba medir, se estaba obteniendo

una tensión en modo común de 12VDC, y por ende solamente una señal ruidosa con $V_{avg} = 50mV$ a la salida del bloque. Previo a tener conocimiento de las causas reales del problema, se invirtió un tiempo considerable en intentar solucionar problemas inexistentes en circuito, incluso llegando a armar otro circuito completamente.

9.2. Software

Para la etapa de debugging del software se contaba con el uso de la herramienta C-SPYLINK de IAR, que establece una comunicación entre el Visualstate y el IAR Embedded Workbench. Esta herramienta es de gran utilidad ya que permite una visualización gráfica del funcionamiento del controlador en tiempo real, facilitando así el proceso de corrección de software y el propio debugging. No obstante, por motivos que no son del todo claros (dado que no se encontró bibliografía al respecto del problema que el programa presentaba), no se pudo hacer uso de esta herramienta. De esta forma, se vio la necesidad de llevar a cabo el debugging con breakpoints en código y agregando variables de evaluación de estado. El debugging así realizado en el código C se hizo muy difícil puesto que el código generado con el Visualstate, a partir del diseño de Statecharts, es muy complejo y no utiliza los nombres de estado definidos en el diseño, con la consecuente pérdida de tiempo que esto acarreó.

Además de lo mencionado anteriormente, cabe destacar otro contratiempo experimentado en términos de software. Como ya se explicó a lo largo de la documentación, dado que el sistema desarrollado se trata de un sistema reactivo se decidió su diseño a través de Statecharts. Este tipo de diseños, de lenguaje UML, definen señales que, según las especificaciones de dicho lenguaje, deberían ser tratadas en tiempo de ejecución igual que los eventos, aunque no sean generadas por eventos físicos externos al controlador, sino internamente -por las máquinas de estados o el propio software-. No obstante, en la etapa de debugging, se vio que si bien los eventos eran debidamente tomados por el sistema, las señales no estaban presentando el comportamiento esperado, ya que se encolaban o perdían prioridad frente a eventos generados posteriormente. Dado que no se encontró explicación de por qué estaba sucediendo este inconveniente, y en consecuencia no se encontró una solución, se vio la necesidad de cambiar el diseño original e implementar funciones que encolaran eventos de la misma forma que lo hace el sistema, es decir, respondiendo a excitaciones externas.

Sin embargo, el contratiempo más grande tuvo que ver con el diseño del software de procesamiento de comunicaciones y los requerimientos hardware que este diseño requería. A priori, se consideró que la velocidad de procesamiento del μC era suficiente para los requerimientos del sistema, dado que decodificar una señal de $4kHz$ con un procesador de $16MHz$ no debería presentar mayores inconvenientes. Sin embargo, luego de las pruebas presentadas al final de esta sección, se concluyó que el código generado a partir de los diseños StateChart realizan un consumo excesivo de los recursos del μC, utilizando una cantidad desmedida de ciclos de máquina (más de los 5000 correspondientes) para el manejo de las máquinas de estado entre flancos de señal. En las secciones siguientes se presentan el diseño original del bloque de comunicaciones y las pruebas realizadas sobre

el sistema con ese diseño.

9.2.1. CommBlock - Diseño Original

En primera instancia, el diseño del bloque de comunicaciones *CommBlock* se realizó de forma íntegra mediante StateCharts, obteniendo la máquina de estados presentada en la Fig. 9.1. Este diseño debió ser descartado al no lograr su correcto funcionamiento luego de la etapa de debugging, habiendo agotado todas las posibilidades al alcance para esto. A continuación se presenta una descripción sintética del funcionamiento de la máquina original, que resulta de utilidad dado que en dicho funcionamiento se basó la estructura del código final. En la subsección siguiente, Código Generado - Pruebas y Testeos (9.2.2) se presentan los resultados obtenidos en las pruebas realizadas sobre el diseño original que llevaron a la conclusión de descartarlo.

MensajeCompleto

Esta submáquina tiene funciones dependientes de los resultados que obtenga de la máquina paralela *DecoByte*, y estas son: Cálculo del largo del paquete a partir del encabezado de paquete recibido, almacenamiento de los bytes de datos recibidos y verificación del Checksum (su diseño se muestra en la Fig. 9.2). Luego de recibido el encabezado, calculado el largo de mensaje y adicionado el valor del Header a la cuenta del Checksum, esta máquina toma cada byte recibido[1] e invierte su contenido ($b_0..b_7$ a $b_7..b_0$), almacenando luego el byte invertido, y adiciona su valor a la cuenta de Checksum. Por último, luego de recibir el byte de Checksum se verifica la coincidencia del Checksum recibido y el calculado, enviando una señal de aviso de paquete listo a *CtrlBlock* en caso de que la verificación sea exitosa. Si, por el contrario, alguno de los pasos de verificación arroja un error, o se recibe una señal de error desde la máquina *DecoByte*, se descarta el mensaje.

DecoByte

En esta máquina se procesa la señal de comunicación recibida, discriminando según los cambios de nivel de señal durante un período para identificar un bit "0" o "1", para la conformación de los bytes. Esto se lleva a cabo a partir de la recepción del *Start Bit*, en sincronismo con la señal, sensando el estado del pin de comunicaciones en 1/4 y 3/4 del período, identificando un "0" si el nivel de tensión recibido es el mismo y "1" en caso contrario. Además, también se hace el manejo de varios temporizadores detallados en la norma (4.4.6), en respuesta a los valores que se vayan recibiendo y dependiendo del estado en que esté *CtrlBlock*. Luego de recibido el byte, se verifica su paridad calculada con la recibida en el bit de paridad (bit siguiente al último bit del byte), y en caso de éxito de esta verificación se espera por el *stop bit* para enviar a la máquina *MensajeCompleto* una señal de fin de byte. En caso de existir algún error en la recepción del byte (error de paridad) se envía una señal de byte inválido a la máquina *MensajeCompleto*. El diagrama Statechart del diseño de esta máquina se muestra en la Fig. 9.3, y está compuesto por dos submáquinas paralelas: *SensorDeNivel* (mostrada en la Fig. 9.4) que implementa la

[1] La máquina *DecoByte* envía una señal a *MensajeCompleto* cada vez que recibió exitosamente un byte.

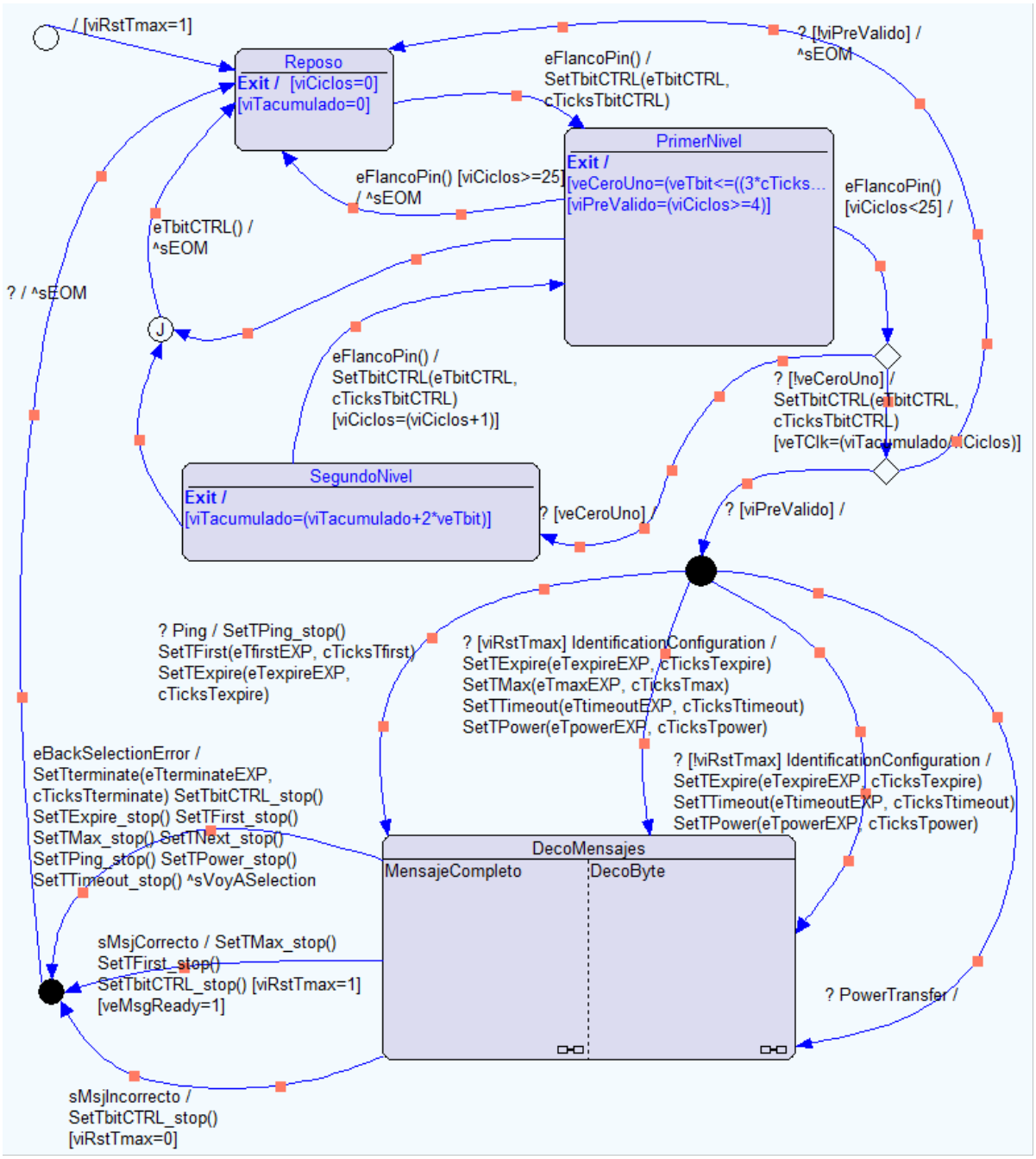


Figura 9.1: Diseño original de la máquina CommBlock.

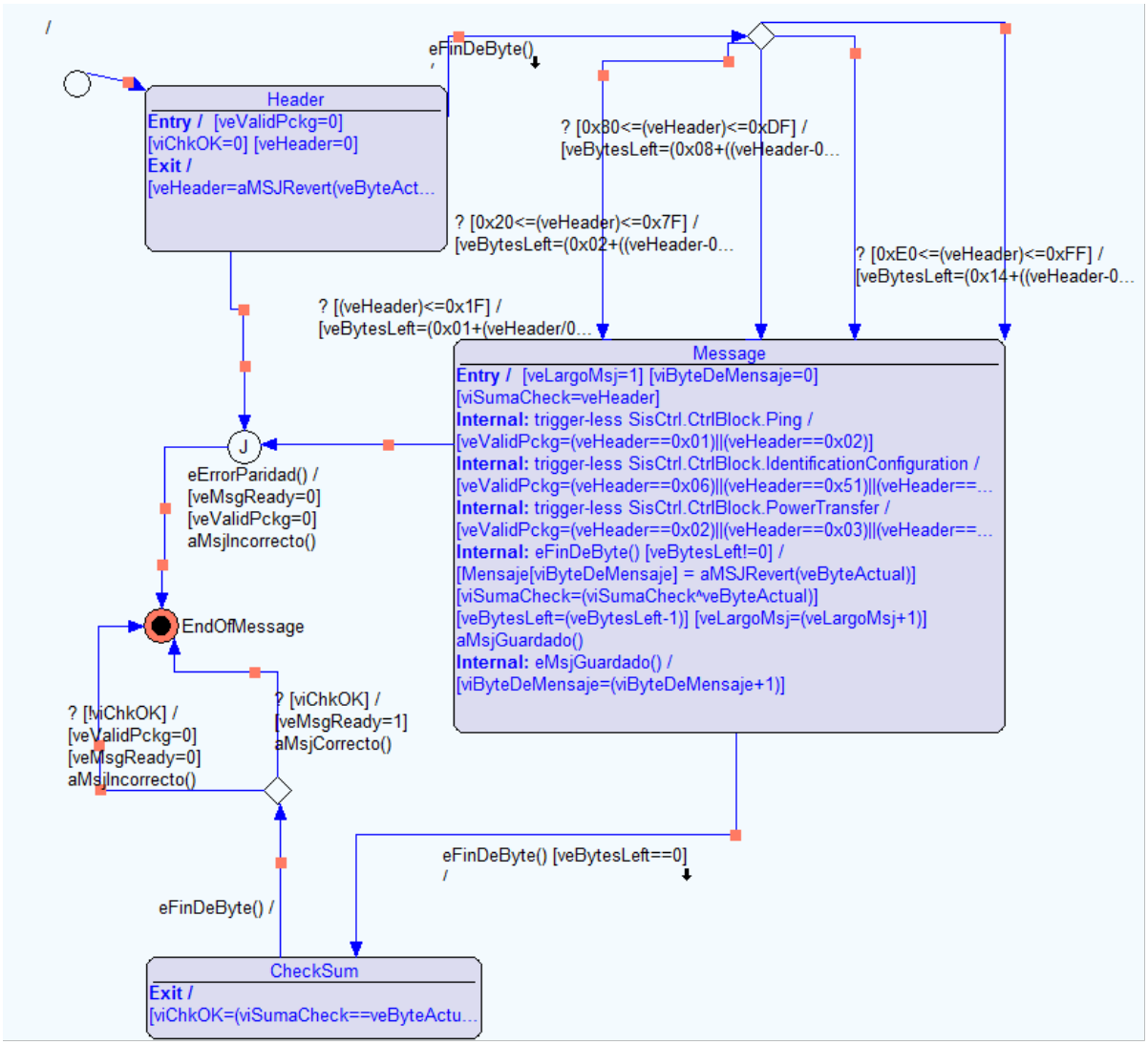


Figura 9.2: Diagrama StateChart de la submáquina *MensajeCompleto* de almacenamiento de paquetes de datos.

lectura del pin, e *InterpreteByte* (mostrada en la Fig. 9.5) que arma los bytes con los bits que se identifican en *SensorDeNivel*.

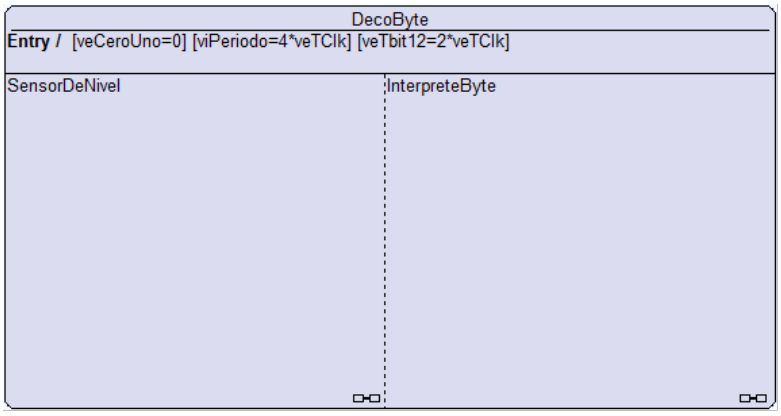


Figura 9.3: Diagrama StateChart de la submáquina *DecoByte* de decodificación de bits y armado de bytes.

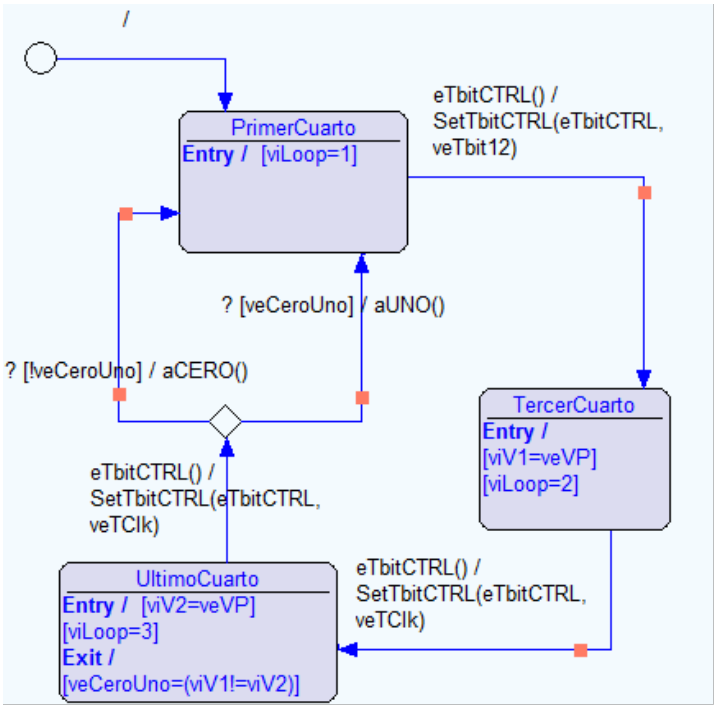


Figura 9.4: Diagrama StateChart de la submáquina *SensorDeNivel* para identificación de bits.

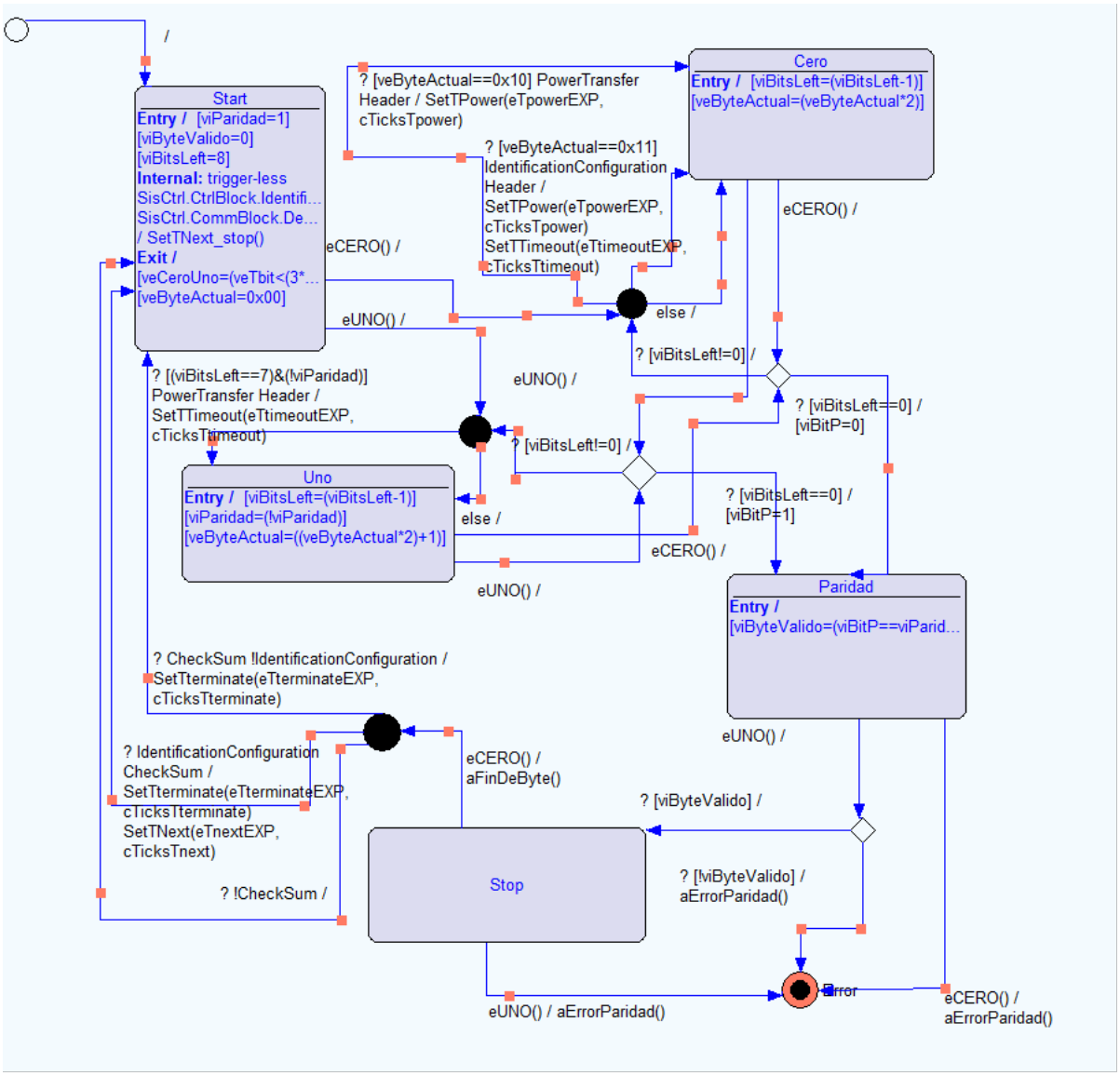


Figura 9.5: Diagrama StateChart de submáquina *InterpreteByte* para el armado de bytes.

9.2.2. Código Generado - Pruebas y Testeos

Frente a la constante recepción errónea de los mensajes enviados por el receptor, habiendo corroborado el comportamiento esperado de la señal a través de datos adquiridos en laboratorio, y luego de reducir al mínimo la posibilidad de estar evaluando espurios como flancos efectivos de la señal, se comenzó a trabajar sobre la sospecha de la imposibilidad del μC de cumplir con los requerimientos. Dado que la versión del software IAR Embedded Workbench para STM8 no cuenta con la herramienta *cycle counter*, que permite evaluar los ciclos de máquina que requieren las instrucciones ejecutadas, se comenzó por utilizar uno de los pines de debugging a los efectos de obtener una señal medible a partir de la cual evaluar la velocidad con la que el controlador interpretaba un bit en respuesta a variaciones en la señal de comunicaciones. En la Fig. 9.6 se muestra el resultado de la prueba que reforzó la teoría de que efectivamente el sistema diseñado no estaba interpretando bien la señal recibida debido a su velocidad de respuesta; prueba que consistió en evaluar la respuesta del sistema frente a la recepción de la señal de comunicación. En esta prueba, se hizo una comparativa entre la señal de comunicación (cuya frecuencia es de $2kHz$) recibida en el pin del μC luego de la demodulación (curva negra), y la repuesta del controlador frente a esta excitación. Para esto, se incluyó en el código una instrucción para que el micro invirtiera el estado (High a Low, o viceversa) de uno de los pines destinados a debugging cada vez que interpretara la recepción de un bit, esta es la señal lenta que se observa en la figura (curva azul).

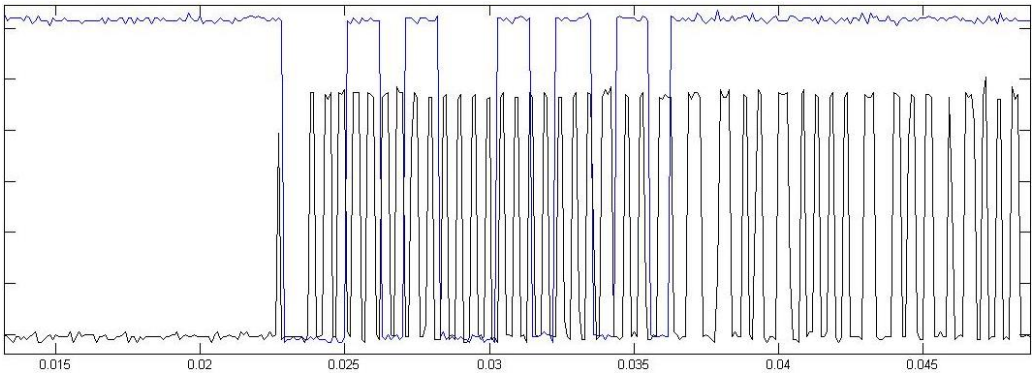


Figura 9.6: Comparativa de la velocidad de respuesta del sistema, frente a la frecuencia de señal.

Efectivamente, observando en detalle la figura, pudo concluirse que el controlador implementado en esas condiciones era incapaz de procesar la señal de comunicación, dado que en el tiempo que ser recibían 20 bits de preámbulo, un *start bit* y 4 bits de datos, el controlador sólo interpretaba 10 bits.

Para hilar un poco más fino, e intentar saber a ciencia cierta el origen de este gran retardo en la velocidad de respuesta del sistema, se realizó el mismo procedimiento mostrado anteriormente, pero cambiando el valor lógico del pin de debugging en el ISR disparado por un flanco recibido en la señal, luego de los procesos de validación del flanco[2]. El re-

[2]El proceso de validación consiste en acotar la ventana de recepción de flanco a rangos cercanos a los

sultado de esta prueba se muestra en la Fig. 9.7, y probó que se estaba implementando de buena forma la evaluación de los cambios de nivel de señal y, por ende, la recepción hardware de los bits, incluso descartando espurios existentes en tiempos cercanos a los cambios de señal.

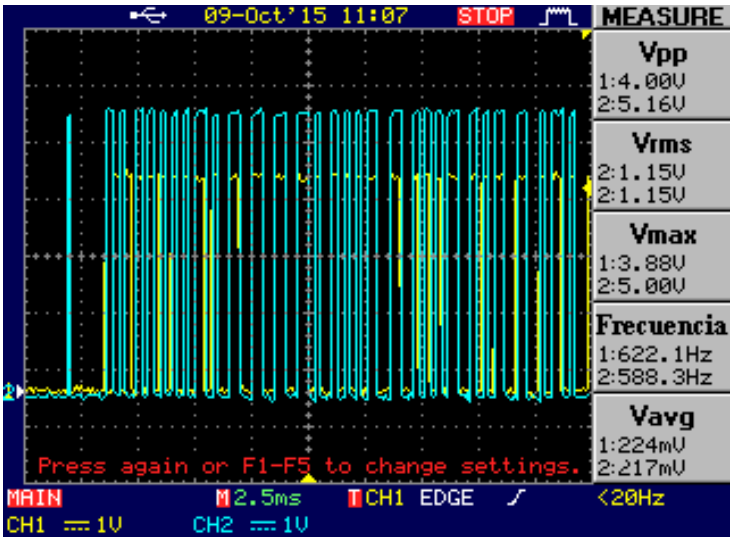


Figura 9.7: Velocidad de respuesta del servicio de atención a interrupciones.

Como última prueba, cuyo resultado se muestra en la Fig. 9.8, se procedió a evaluar el estado de la cola de eventos que implementa la dinámica del sistema durante el proceso de recepción y decodificación de comunicaciones, obteniendo resultados que dejaron en evidencia la saturación de los recursos del microcontrolador al intentar realizar este proceso. El evento identificado con valor 8 es el evento encolado por la ISR disparada a raíz de la detección de un flanco en el pin de comunicaciones. En el estado mostrado en la figura, la cola de eventos estaba en estado de no aceptación de nuevos eventos y el resto del sistema sin capacidad de tomar los eventos a la velocidad a la que se generaban. En base a esto, se concluyó que el sistema estaba descartando eventos relacionados a nuevos cambios en la señal de comunicaciones y que, al atender eventos relacionados a cambios ya ocurridos, el valor de la variable que identificaba el largo de bit correspondía a algún bit posterior al que generó el evento.

Basándose en estos resultados, se concluyó que el software generado con la aplicación VisualState sobreexige los recursos del μC, haciéndolo incapaz de procesar señales de la frecuencia manejada en la comunicación. En consecuencia, se procedió a rediseñar el sistema de procesamiento de comunicaciones realizando un diagrama Statecharts básico, que se encontrara en estado de espera mientras una función implementada en código recibe y procesa la señal de comunicación correspondiente a un paquete de datos.

tiempos de cambio de nivel, es decir, cerca de un medio del tiempo de período y de un período completo.

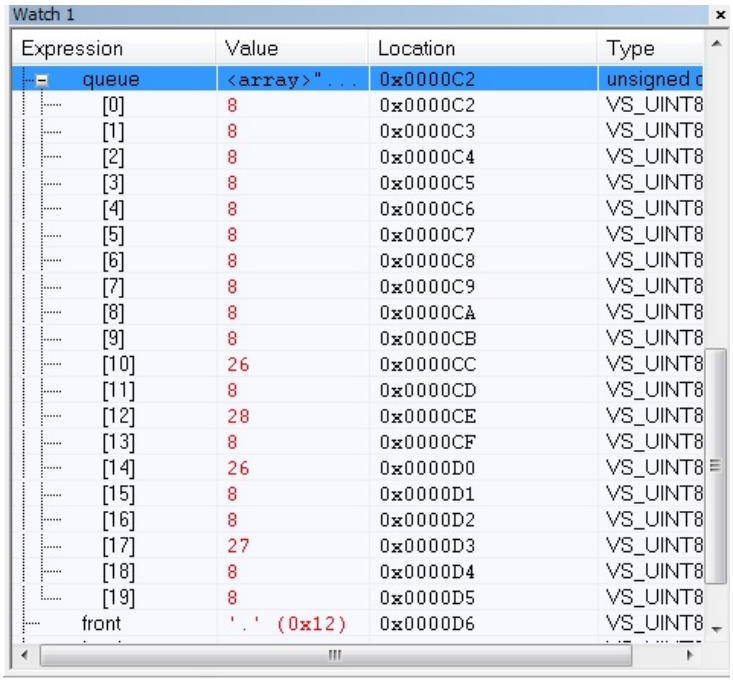


Figura 9.8: Estado de la cola de eventos al procesar comunicaciones con el diseño original.

Capítulo 10

Anexo E - Gestión de tiempos

La gestión real de tiempos en este proyecto distó considerablemente de los planificado tanto al inicio, como luego de los ajustes realizado en las instancias de presentación de avance. Incluso los buffers de tiempo considerados al momento de la replanificación, a instancias de la segunda presentación de avance (cuyo diagrama se muestra en la Fig. 10.1), no fueron suficientes para la previsión de los retrasos experimentados a causa de los motivos ya explicados.

De todas formas, y quitando el foco de los contratiempos experimentados, se evalúa de forma negativa las decisiones tomadas en la planificación. Esta evaluación deriva de que el tiempo planificado para dedicar al desarrollo de software, y sobre todo para el debugging, fue notoriamente insuficiente, redundando en un alcance final de los objetivos insuficiente. De igual forma, se evalúa negativamente la incapacidad de avance en paralelo frente a retrasos experimentados.

Figura 10.1: Replanificación realizada en Junio 2015.

Capítulo 11

Bibliografía

[1] WPC, "Qi specification - system description - wireless power transfer," vol. I: Low Power, 2013. [Online]. Available: http://www.wirelesspowerconsortium.com/downloads/wireless-power-specification-part-1.html

[2] "Demodulating communication signals of qi-compliant low-power wireless charger using mc56f8006 dsc," 2013. [Online]. Available: cache.freescale.com/files/microcontrollers/doc/app_note/AN4701.pdf

[3] "Hoja de datos de driver tps28225-q1," 2011. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps28225-q1.pdf

[4] TexasInstruments, "Hoja de datos de puente mosfet csd87352q5d," 2011. [Online]. Available: http://www.ti.com/lit/ds/symlink/csd87352q5d.pdf

[5] "Hoja de datos de μc stm8s105k6," 2015. [Online]. Available: http://www.st.com/web/en/resource/technical/document/datasheet/CD00200092.pdf

[6] "Iar visualstate user guide," 2014. [Online]. Available: ftp://ftp.iar.se/WWWfiles/vs/UserGuide.pdf

[7] "Application schematic - qi a6 controller bq500412," 2013. [Online]. Available: http://www.ti.com/lit/df/sluc522/sluc522.pdf

[8] "Hoja de datos de stepdown tps54231," 2008. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps54231.pdf

[9] "Hoja de datos de bloque de amplificadores operacionales lm324a," 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm224.pdf