UNIVERSIDAD SAN FRANCISCO DE QUITO

Sistema Steer By-Wire: Construcción y control de plataforma de investigación

Paulo Guerra Figueiredo

Tesis de grado presentada como requisito para la obtención del título de Ingeniería en Eléctrica y Electrónica

Quito, abril de 2008

UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio Politécnico

HOJA DE APROBACION DE TESIS

Sistema Steer By-Wire: Construcción y control de plataforma de investigación

Laurent Sass, PhD. ……...
Director de la Tesis

Santiago Navarro, PhD. ……...
Miembro del Comité de Tesis

Javier Dávila, PhD. ……...
Miembro del Comité de Tesis

Fernando Romo, M.S ……...
Decano del Colegio Politécnico

Quito, abril de 2008

Resumen

El proyecto “Sistema Steer By-Wire: Construcción y control de plataforma de investigación” consiste en el estudio y desarrollo de un sistema de dirección controlado por completo de forma electrónica, eliminando la conexión mecánica entre el volante y las ruedas del vehículo. El proyecto se inicia con una introducción de los antecedentes en la investigación y desarrollo de sistemas By-Wire, para a partir de los mismos proponer y construir una plataforma de investigación que cuente con las características de un sistema de dirección By-Wire dentro de los límites del presupuesto y piezas disponibles en el Ecuador. Se incluyen especificaciones sobre retroalimentación de fuerza del volante, el control de posición de los motores utilizados para orientación de las ruedas del sistema, y el uso de la aplicación de monitoreo del sistema.

Abstract

The project “Sistema Steer By-Wire: Construcción y control de plataforma de investigación” consists in the study and development of a steering system completely controlled electronically, eliminating the mechanical linkages between the steering wheel and the wheels of the vehicle. The project begins with an introduction about the background investigation and development of By-Wire systems. The construction of an investigation platform that includes By-Wire steering system characteristics within the budget and parts available in Ecuador, is then proposed. Detailed specifications are included for force feedback at the steering wheel, position control of the motors used to orientate the wheels, and the monitoring software used by the system.

Tabla de Contenido

Lista de Figuras

Capítulo 1

Introducción

Como introducción a este proyecto, se hará un breve recorrido por la evolución de los sistemas de dirección desde el invento de los automóviles. Una vez conociendo el grado de evolución en el que se ubican los sistemas de dirección By Wire, se procede a introducir las principales características de la tecnología By Wire y sus ventajas. Por último, basándose en la información de los requerimientos de un sistema By Wire se introduce la arquitectura del sistema de dirección By Wire que se ha propuesto y construido como resultado de este proyecto.

1.1 Evolución de los sistemas de dirección

Desde el momento en que se ideó el automóvil hasta hoy en día la evolución del mismo y su diseño, ha sido muy significativa. En los primeros automóviles, como se puede observar en la figura 1.1, la dirección se accionaba por una palanca o manúbrio. A medida que cambiaron las características de las vías por donde transitaban los vehículos, así como las velocidades que los mismos alcanzaban, los sistemas de dirección fueron sufriendo modificaciones. Fue así como se adoptó la forma circular para el volante y las diversas formas de reducción para operar el sistema con menor esfuerzo por parte del conductor. A lo largo de los años, este tipo de modificaciones han producido sistemas de dirección más suaves, precisos y seguros; además de sistemas más sensibles para que el conductor pueda percibir a través de ellos las características del camino por el que transita (Sistema de Dirección, Atiko Estudio). A medida que las condiciones en las que se conducen un vehículo cambian, también cambian los diseños de sus componentes. Las velocidades a las que se conduce hoy en día superan por mucho las velocidades a las que

se circulaba hasta finales de los años treinta. De igual forma la conciencia sobre el impacto ambiental de un vehículo se ha desarrollado en las últimas décadas. Esto ha provocado que, hoy en día, la seguridad y el impacto ambiental de un automóvil, sean características tomadas como importantes referencias para un conductor el momento de escoger su vehículo. Como respuesta a estas exigencias, los sistemas de dirección también han tenido que pasar por varios cambios.

Figura 1.1 Primeros sistemas de Dirección (Eyewitness to history)

El papel del sistema de dirección está claro: convertir una orden provista por el conductor del vehículo en una orientación correspondiente de las llantas del vehículo y por lo tanto en un cambio en la dirección de movimiento del mismo. Hoy en día, generalmente se utilizan sistemas compuestos por columna de dirección, cajas de reducción, piñones y cremallera, encargados de transmitir las fuerzas mecánicas necesarias para cambiar la orientación de las llantas. Esta sección presenta de manera introductiva la evolución presentada por los sistemas de dirección comunes en los vehículos de hoy en día, para

comprender cuán actuales y modernos resultan los diseños de sistemas de dirección By-Wire, foco de estudio en este proyecto.

1.1.1 Sistemas de dirección manual

El sistema manual de dirección fue uno de los primeros sistemas de dirección, y aún es utilizado por la gran mayoría de vehículos antiguos que se encuentran en circulación. Este sistema se caracteriza por la conexión completamente mecánica entre el volante y las ruedas del vehículo. Depende completamente de la fuerza aplicada por el conductor a través del volante para alterar la dirección del vehículo. Las magnitudes de las fuerzas que el conductor necesita aplicar dependen de varios factores como el tamaño del volante, tamaño de las llantas, peso del vehículo, y la resistencia ofrecida por el camino sobre el que se mueve el vehículo.

La principal desventaja de un sistema de dirección manual tiene justamente que ver con la fuerza requerida del conductor para activar el sistema. Muchas veces, dichas fuerzas pueden ser demasiado grandes y causar molestias e incomodidad al conductor el momento de girar el volante en ciertas situaciones como al estacionar. Sin embargo, la elección del mismo como el sistema de dirección para la mayoría de vehículos producidos durante muchos años, se dio debido a la falta de alternativas para reemplazarlo manteniendo sus características de retroalimentación provista al conductor. Al existir una conexión mecánica desde las ruedas hasta el volante, el conductor puede sentir las características del camino sobre el que se desplaza a través del sistema de dirección, permitiéndole detectar cualquier cambio en las condiciones del mismo a través de la fuerza sentida en el volante. Esto se debe a que la presencia de engranajes introduce relaciones lineales entra la fuerza

impuesta por la interacción entre las llantas y el camino, y la fuerza sentida por el conductor en el volante (Chou, 4).

En la figura 1.2, se puede observar la estructura de un sistema de dirección manual tradicional.

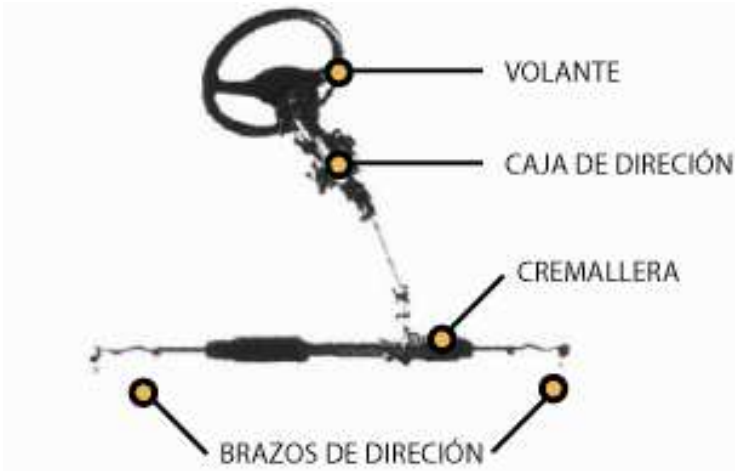


Figura 1.2 Dirección Manual (Ebner, 3)

1.1.2 Sistemas de dirección asistida

Los sistemas de dirección asistida, están basados en los sistemas de dirección manual, pero con la mejora de incluir algún tipo de fuente de potencia adicional que asista al conductor en la transmisión de la fuerza necesaria para mover las ruedas del vehículo. De tal forma, el sistema asistido complementa la acción manual y reduce la magnitud de las fuerzas que el conductor necesita aplicar para activar el sistema de dirección. Las fuentes de potencia adicional utilizadas en los vehículos con sistemas de dirección asistida, pueden ser hidráulicas o eléctricas.

Los sistemas asistidos hidráulicamente utilizan una bomba hidráulica, junto a un pistón, para proveer al sistema una fuerza adicional a la ejercida por el conductor. Como se puede observar en la figura 1.3, el pistón se ubica como parte de la cremallera justamente

para asistir hidráulicamente en el movimiento de ésta. La válvula rotativa indicada en la figura, se encarga de hacer llegar el líquido hidráulico a un lado o al otro del pistón, a través de las líneas de fluido, según el sentido de giro de la columna de dirección. El piñón se ubica al final de la columna de dirección, al igual que en los sistemas manuales, para transmitir el movimiento de la columna de dirección a la cremallera (Manual CEAC del Automóvil, 692).

La bomba se encarga de generar la alta presión en el líquido hidráulico necesaria para el funcionamiento del sistema de asistencia hidráulica. Los sistemas de dirección hidráulicos se clasifican dependiendo de cómo es activada su bomba hidráulica. Las bombas hidráulicas que se utilizan para este tipo de sistemas pueden ser activadas a través del motor del vehículo, o por un motor eléctrico.

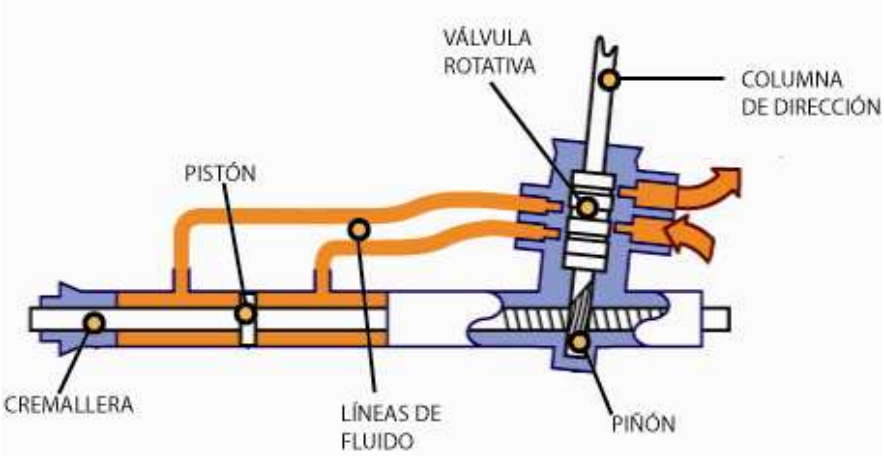


Figura 1.3 Cremallera de un sistema de dirección asistido hidráulicamente

Los sistemas con bombas activadas por el motor del vehículo son bombas que son puestas en funcionamiento por el movimiento de giro del cigüeñal en el motor del vehículo. Esta característica, las vuelve muy poco eficientes ya que se encuentran en

funcionamiento continuamente sea o no necesaria su asistencia. Además, el funcionamiento continuo de la bomba quita potencia al motor.

Las bombas manejadas por motores eléctricos, tienen un motor eléctrico controlado electrónicamente que permite que la bomba se ponga en funcionamiento únicamente cuando es necesaria su asistencia. Ciertamente este tipo de sistemas son mucho más eficientes, y proveen características de controles programables y adaptables al tipo de conductor. Sin embargo, su costo resulta mayor debido al motor eléctrico que se adiciona específicamente para controlar la bomba hidráulica.

La figura 1.4, muestra un sistema de dirección asistida hidráulicamente. Se puede observar su semejanza con el sistema manual, pero con la adición de la bomba hidráulica característica de este sistema.

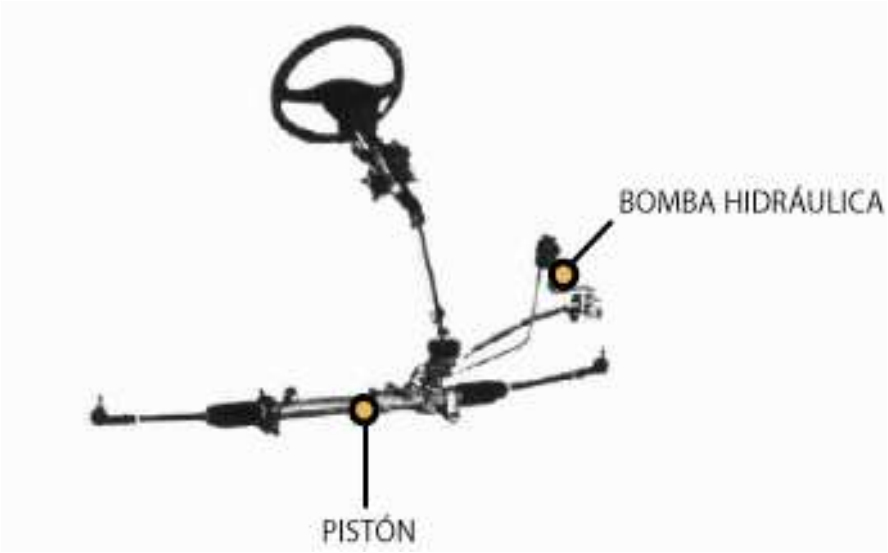


Figura 1.4 Dirección Asistida Hidráulicamente (Ebner, 4)

Otro tipo de dirección asistida elimina la bomba hidráulica como fuente de potencia adicional y la reemplaza con un motor eléctrico, que mediante una caja de reducción se une a la columna de dirección para asistir al conductor con la fuerza que sea necesaria. Este

sistema necesita, además, un controlador electrónico y sensores. Durante la operación del sistema, los sensores se encargan de medir y enviar señales proporcionales al torque provisto por el conductor y la posición del volante, al controlador del motor eléctrico (Chou, 5). Basándose en las señales de los sensores el controlador ubica al motor en un punto de operación dado, dependiendo del grado de asistencia necesario. El sistema puede incrementar el número de variables medidas, como la velocidad del vehículo, para mejorar la respuesta, eficiencia y adaptabilidad del sistema.

1.2 ¿Por qué Steer By Wire?

Se puede afirmar que la electrónica ha revolucionado muchos aspectos de nuestra vida diaria en las últimas décadas, y sin duda continuará haciéndolo por muchas más. Los automóviles forman parte fundamental de la vida diaria de millones de personas. La proliferación de sistemas electrónicos utilizados como ayuda, y muchas veces como reemplazo, para los sistemas mecánicos convencionales, es una tendencia que se puede observar claramente hoy en día. Sistemas de seguridad automotriz como sistemas de freno antibloqueo (ABS) , sistemas de control de tracción (TCS), así como la mejora y combinación de los mismos en el programa de estabilidad electrónica (ESP), son muestras de la importante influencia de la electrónica en el desarrollo de los sistemas utilizados por los vehículos modernos.

Tomó varias décadas, desde la invención del automóvil, para que la electrónica sea utilizada como soporte para los sistemas de dirección. Sin embargo, poco a poco, los sistemas electrónicos ingresaron en el campo de la dirección, primero ayudando en el control de bombas hidráulicas utilizadas en los sistemas de dirección asistidos hidráulicamente, luego con la introducción de motores eléctricos en las direcciones

asistidas por completo de forma electrónica, y en la actualidad con el estudio de sistemas completamente electrónicos con la tecnología Steer By Wire.

A pesar de que los sistemas de dirección asistida representan un avance respecto al sistema de dirección manual, todos ellos parten del sistema manual como base, y por lo tanto mantienen algunas de las desventajas de dicho sistema. Los sistemas tradicionales de dirección dejan de ser satisfactorios debido a desventajas como las enumeradas a continuación:

- La columna de dirección condiciona el diseño del compartimento del motor por lo que resulta inconveniente para el diseño e investigación de nuevas estructuras de seguridad contra accidentes.
- Transmisión de fuertes impactos desde la carretera hacia el conductor a través de las conexiones mecánicas.
- Cada vez existen mayores regulaciones ambientales sobre el tratamiento de los desechos de líquidos a base de petróleo, como el aceite de dirección, utilizados en los sistemas hidráulicos y conocidos por contaminar el suelo y el agua. (hydraulicspneumatics.com)
- La estructura mecánica convencional resulta inconveniente para el desarrollo de sistemas de ayuda al conductor que se basen en el sistema de dirección, como sistemas de estacionamiento automático.

(Ebner, 5)

Debido a desventajas como las mencionadas, las investigaciones actuales se enfocan en un nuevo sistema de dirección que rompe por completo con el esquema de las conexiones mecánicas utilizadas por los sistemas tradicionales. Ese nuevo sistema de dirección está basado en la tecnología By Wire, y es el cual se propone investigar con la plataforma que este proyecto tiene como objetivo construir.

Sin profundizar en su definición y características, el término By Wire hace referencia a sistemas cuyas conexiones están basadas en una red de señales eléctricas, distinguiéndose por completo de los sistemas tradicionales basados en conexiones mecánicas o hidráulicas. Enfocándose específicamente en la dirección, un sistema Steer By Wire elimina por completo cualquier conexión mecánica entre el conductor y las ruedas del vehículo. Como reemplazo, ubica actuadores en las llantas para controlar su orientación, y sensores de posición tanto en el volante como en los actuadores. Además, se ubican sensores de corriente y/o sensores de fuerza de los actuadores. Una unidad central electrónica se encarga de procesar las señales de los sensores, realizar los cálculos necesarios y enviar señales de control a los actuadores para orientar las ruedas según sea necesario de acuerdo con la posición del volante, además de retroalimentar fuerza al volante, mediante un motor conectado al eje del mismo. La retroalimentación de fuerza es utilizada para que el conductor no pierda la sensación de la resistencia al giro de las llantas provocada por el camino. La figura 1.5 presenta un esquema general de la estructura mencionada para un sistema Steer By Wire.

Figura 1.5 Esquema general de un sistema Steer By Wire

Queda claro que la tecnología By Wire se ubica en el punto más actual en la evolución de los sistemas de dirección. Conociendo este hecho, y en busca de responder a necesidades concretas y actuales de investigación dentro de la USFQ es que se ha escogido la construcción de una plataforma de investigación de un sistema de dirección By Wire como tema para este proyecto. A continuación se introduce un poco de la historia de la tecnología By Wire y de las ventajas que la misma tiene dentro del sector automotriz.

1.3 La tecnología By-Wire y sus ventajas

Hacia 1960, la compañía McDonell Douglas reemplazó los sistemas mecánicos por completo con sistemas electrónicos en sus aviones (Leppin, 1). Este reemplazo, con tecnología conocida como By-Wire en inglés, cambió por completo la perspectiva de lo que se podría alcanzar con la electrónica en un vehículo y encaminó un gran número de

investigaciones a la implementación de tal tecnología en automóviles. La tecnología By-Wire es una tecnología que involucra que cada mecanismo o sistema que cuente con la misma, sea controlado electrónicamente. Esta forma de control permite la eliminación de las conexiones mecánicas dentro de un sistema o entre sistemas, ya que permite conocer el estado de cada sistema y ordenar movimientos al mismo, únicamente por medio de señales eléctricas.

Los sistemas mecánicos e hidráulicos utilizados en un inicio por los aviones tenían las desventajas de ser bastante pesados y requerir de cuidados minuciosos, además de poseer poca adaptabilidad a cambios en las condiciones aerodinámicas enfrentadas por el avión. Para los nuevos diseños, introducir un sistema que elimina una gran cantidad de conexiones mecánicas, como es el caso de un sistema con tecnología By-Wire, significó importantes avances en lo que respecta a la disminución de peso, y a la ampliación de las capacidades de controlar las características aerodinámicas del avión.

La presencia de la electrónica en los sistemas de control de los aviones, permitió que dichos sistemas evolucionen a sistemas automatizados que hoy en día complementan el trabajo del piloto, y que en muchos casos llegan a reemplazarlo. La automatización ha permitido el desarrollo de sistemas robóticos capaces de controlar las características aerodinámicas de los aviones, interviniendo electrónicamente en el posicionamiento de mecanismos como los alerones.

Aplicaciones de control con gran capacidad de cálculo, han permitido el desarrollo de sistemas de pilotaje automático a distintos niveles. Los aviones no tripulados se ubican en el tope más alto en la evolución de los sistemas de pilotaje automático. La electrónica, permite que un sistema de pilotaje avanzado a bordo del avión interprete órdenes, que son transmitidas desde un computador portátil en tierra con GPS, y envíe

señales a motores electrónicos ubicados para controlar la posición tanto de alerones como del timón posterior (elmundo.es).

Ciertamente, se espera que en el caso de los automóviles la tecnología By Wire abra el mismo camino hacia la automatización, junto con los beneficios que ésta implicó para los aviones.

El término By-Wire se aplica generalmente a sistemas que utilizan diseños de controladores en los cuales se interpone un sistema computarizado de control entre el operador y el actuador. Este tipo de control permite que los comandos enviados por el operador sean tratados por programas de control antes de enviar una orden de acción al actuador que el operador desee controlar. La ventaja de procesar la orden del operador por medio de una aplicación de software es que es posible desarrollar tales aplicaciones al nivel de asegurar un punto de operación de máxima eficiencia y a la vez resguardar por completo la seguridad del operador y usuarios del vehículo que lleva instalado un sistema con tecnología By-Wire. Además las aplicaciones pueden ser actualizadas, con nuevas y mejores versiones de programas de control, con la misma facilidad con la que se instala un programa en la computadora. Queda claro que un sistema By Wire resulta mucho más versátil, ya que una pequeña actualización del programa podría mejorar por completo las características del sistema a un pequeño costo, algo prácticamente imposible si el vehículo utiliza un sistema convencional de dirección en el cual alterar las características podría involucrar el reemplazo total del sistema de dirección a un costo muy elevado.

En los aviones, los primeros sistemas By-Wire, hacían uso de controladores basados en computadoras analógicas. Las computadoras analógicas eran dispositivos eléctricos o hidráulicos diseñados para procesar los datos de entrada en términos de cantidades físicas como niveles de voltaje o presiones hidráulicas. En una computadora analógica eléctrica típica, las entradas se convierten en tensiones que pueden sumarse o

multiplicarse empleando elementos de circuitos de diseño especial. Un ejemplo común es el tacómetro, el cual convierte el giro de un motor en una señal de voltaje proporcional al número de revoluciones por minuto a las que gira el motor (Ciberhábitat, párr.6). Este tipo de controladores, se encargaban de manejar servo válvulas mecánicas en el caso de los sistemas hidráulicos o servo válvulas controladas eléctricamente en el caso de los sistemas eléctricos. El principal inconveniente con dichos sistemas era su poca precisión característica afectada por su dependencia de los valores leídos por los componentes análogos con respecto a factores externos como la temperatura ambiente y el desgaste sufrido por el uso del sistema. Además, los sistemas de transmisión de movimiento dependían de complejas estructuras mecánicas e hidráulicas que implicaban altos costos en comparación con el costo del cableado encargado de transmitir el movimiento a los actuadores en un sistema By Wire.

Las computadoras digitales, a diferencia de las analógicas, procesan todas sus señales como datos numéricos, usando tantos bits sean necesarios para asegurar la precisión de sus cálculos, algo clave para la seguridad de los sistemas By Wire. Las computadoras digitales, abren el camino para aumentar significativamente la complejidad de los algoritmos de control implementados respecto a las computadoras analógicas. Estas características, llevaron a que las computadoras analógicas de los aviones sean reemplazadas por computadoras digitales para encargarse de todo el procesamiento de las señales. Ciertamente, la digitalización volvió a los sistemas By Wire de los aviones mucho más flexibles y estables para recibir las señales de todos los sensores ubicados en el sistema, interpretarlas y ejecutar los algoritmos de control respectivos con velocidad, precisión y seguridad.

Hoy en día, empresas como BMW, Mazda, General Motors, Fiat, Bosch y Delphi conducen las principales investigaciones en prototipos de automóviles con sistemas de

dirección By Wire. En el año 2003 Mazda presentó el prototipo Washu, en el cual el volante se encuentra montado sobre el tablero en una configuración "steer-by-wire" para poder esconderlo cuando no se utiliza; similar al mando de un moderno avión, tiene un equipo multifunción que controla las distintas funciones de la conducción (Autocosmos, párr.3). Antes, en el 2002, General Motors presentó el concepto Hy-Wire, un chasis estilo monopatín que funciona con celdas de hidrógeno y utiliza la tecnología By Wire en todos sus sistemas (Wikipedia, párr.1). Una imagen de este concepto, se puede observar en la figura 1.6.

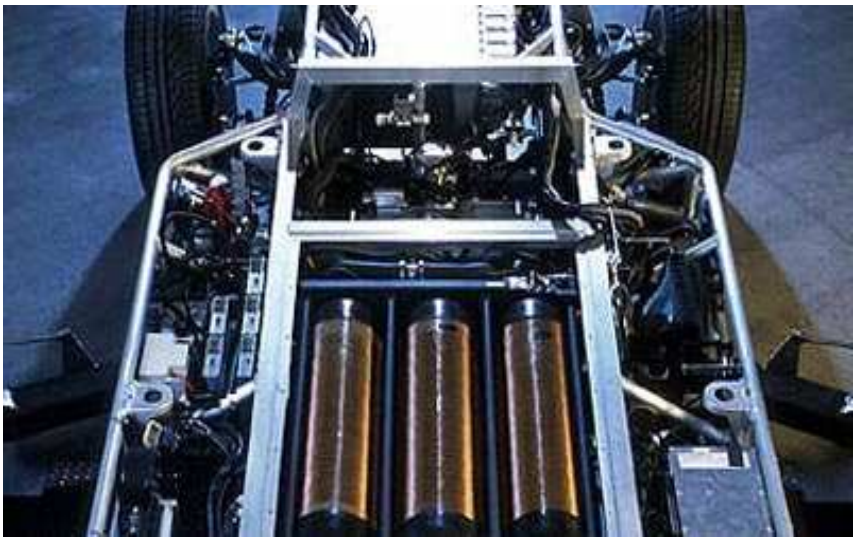


Figura 1.6 Chasis General Motors Hy-Wire (How Stuff Works, General Motors)

El hecho de que los sistemas desarrollados por las grandes empresas mencionadas aún se encuentren fuera del mercado se debe a que los mayores cuestionamientos para este tipo de tecnología tienen que ver con su seguridad, argumentando que una pequeña falla en el sistema eléctrico puede ocasionar desastres como la pérdida total del control del vehículo. La electrónica y el uso de computadoras, permite responder a tales cuestionamientos con la implementación de sistemas redundantes que trabajan con canales

independientes por los cuales se transmiten exactamente los mismos datos, de tal forma que si uno de estos canales falla, el sistema pueda seguir funcionando con los otros canales sin comprometer la seguridad del operador. Al mismo tiempo, este método pone bajo aviso de las fallas presentes en el sistema, con el suficiente tiempo para hacerse cargo de su reparación.

Los sistemas redundantes han sido implementados exitosamente en los aviones con sistemas Fly By Wire, pero su implementación en vehículos entra en duda debido a su costo. El número de automóviles producidos en el mundo es mucho mayor que el de aviones. Las empresas aeronáuticas recuperan el capital invertido en la implementación de sistemas redundantes en sus aviones gracias a la cantidad de pasajeros que pagan por volar en los mismos, pero las empresas automotrices aún no encuentran forma de justificar el elevado precio que tendría un vehículo que implemente sistemas redundantes para su seguridad. Existen otras alternativas de seguridad como el tener respaldos mecánicos o hidráulicos que entran en acción en caso de una falla en el sistema By-Wire, sin embargo estas alternativas son poco deseadas ya que con ellas se pierden muchas de las ventajas de los sistemas By-Wire, como por ejemplo la reducción del peso del sistema. Para evitar hacer uso de tales alternativas, normalmente se incrementa el número de canales independientes de comunicación permitiendo manejar un mayor número de fallas del sistema, pero su costo de implementación sigue siendo el mayor obstáculo para la salida al mercado de tales sistemas.

El reemplazar por completo un sistema convencional de dirección por un sistema de dirección By-Wire tendrá importantes ventajas para los vehículos que implementen dicho sistema. Sólo tener en cuenta que el nuevo sistema elimina por completo la columna de dirección abre espacio para pensar en cuanto más simple podría ser el diseño del compartimento del motor del vehículo. Además en un accidente que involucre un choque

frontal, las posibilidades de que el volante invada el habitáculo del vehículo amenazando la integridad del conductor se reducen considerablemente al no existir un medio de transmisión del impacto, como constituye la columna de dirección. El volante, al perderse la conexión mecánica con las ruedas, podría convertirse en un accesorio completamente portátil y adaptable al tipo de conductor, pudiendo localizarlo a la derecha u izquierda del panel del automóvil, según las necesidades, en cuestión de segundos. Cambiar el sistema de dirección By-Wire, podría llegar a ser tan sencillo como es cambiar de radio en los vehículos actuales. De igual forma, la eliminación de la conexión mecánica entre volante y ruedas, trae ventajas adicionales como es la eliminación de la transmisión de molestos golpes como los que se siente cuando el vehículo golpea una vereda o choca con otro. La característica de retroalimentación de fuerza por medio de un motor que simula las fuerzas que se generan debido a la interacción entre las llantas y el terreno sobre el que circula el vehículo, permitiría desarrollar un sistema en el que el conductor selecciona el nivel de retroalimentación que desea recibir, controlando así el nivel de ruido y vibración de la carretera que siente a través del volante. Adicionalmente, al incluir una computadora que procese toda la información del sistema de dirección, es posible ajustar infinitamente parámetros como la relación entre el giro del volante y el de las ruedas, así como el esfuerzo necesario para girar el volante, optimizando tanto la respuesta del sistema de dirección como la sensación de conducción para el operador.

La tecnología By Wire aplicada a la dirección de un vehículo abre la posibilidad para equipar los automóviles con la capacidad de dirección activa (active steering en inglés). El término dirección activa hace referencia a la capacidad de controlar electrónicamente el nivel de influencia que tiene en la dirección de las ruedas el comando dado por el conductor del vehículo a través del volante. Dicha capacidad involucra que el sistema interceda por el conductor si se detecta que el vehículo llega a un punto cercano a

la pérdida de maniobrabilidad por parte del conductor. Igualmente la dirección activa, permite alterar la influencia del conductor en la respuesta final de los actuadores dependiendo de las condiciones en las que se maneja el vehículo; por ejemplo a altas velocidades, que involucran una potencial pérdida de control de vehículo. La computadora, tomaría un mayor control sobre el vehículo, alterando la respuesta del sistema de dirección respecto a las órdenes enviadas por el conductor.

El uso de motores eléctricos para controlar el movimiento de los actuadores del sistema de dirección provee de respuestas más rápidas a un sistema By Wire, y de un uso de energía y espacio mucho más eficiente que los sistemas convencionales. El avance en cuanto a tiempos de respuesta para un sistema By Wire se encuentra en que al eliminar las conexiones mecánicas también se reducen etapas de transferencia de energía, que generalmente involucran pérdidas por fricción y flexión de los materiales, y las reemplaza por etapas de transmisión de señales eléctricas que ciertamente viajan mucho más rápido y con pérdidas mucho menores.

Por último, y ciertamente como otra clara ventaja, un sistema con tecnología By Wire resulta más económico en varios aspectos. Ya que una unidad electrónica de control es la encargada de procesar todas las señales en el sistema, las capacidades de diagnóstico de problemas en el mismo aumentan significativamente al poder monitorear todo el sistema a través de dicha unidad. Esto se traduce en economía en costos de servicio técnico. Ciertamente al eliminar conexiones mecánicas, un sistema By-Wire requiere de la instalación de menos partes, por lo que su tiempo de producción y su costo es menor que el de los sistemas mecánicos o hidráulicos convencionalmente utilizados por un vehículo (Whitfield, párr 4).

Es importante considerar que, si bien los sistemas Steer By Wire serán muy económicos el momento de salir al mercado, su etapa de desarrollo no resulta para nada

económica, siendo otro motivo fundamental para que estos sistemas aún se encuentren distantes de su producción a nivel comercial. Esto se debe principalmente a que las investigaciones sobre sistemas Steer By Wire para automóviles a nivel comercial, significan importantes inversiones para las instituciones que las auspician. Lo cual, ocasiona que el desarrollo de este tipo de sistemas con todas sus ventajas, se encuentre limitado por la capacidad para financiar los altos presupuestos que las mismas implican.

1.4 Arquitectura del sistema Steer By Wire propuesto

Este proyecto se enfoca únicamente en la aplicación de la tecnología By Wire al sistema de dirección. El objetivo es construir una pequeña plataforma de investigación que implemente un sistema de dirección By Wire, eliminando toda interfase mecánica entre el conductor y el mecanismo de orientación de las ruedas. Tomando en cuenta las limitaciones de presupuesto disponible para el proyecto y las piezas disponibles dentro del país, se diseñó una plataforma que contara con los elementos básicos necesarios para realizar investigación referente a la dirección By Wire. El diseño, sin profundizar en detalles técnicos y considerando los principales requerimientos de un sistema de dirección By Wire, se expone a continuación.

Existen varias opciones al momento de controlar la dirección de un vehículo por medio de actuadores eléctricos. Una opción consiste en eliminar la conexión entre el volante e instalar un único motor eléctrico que mueva la columna de dirección. La misma idea se puede utilizar para eliminar por completo la columna de dirección, instalando el motor a la altura del piñón para iniciar el movimiento de las ruedas. Este tipo de sistema ya ha sido estudiado por estudiantes de Instituto Politécnico de Virginia, implementándolo en un Cadillac SRX de General Motors (Leppin, 52).

Si bien la opción expuesta en este proyecto elimina la conexión mecánica del volante con el resto del sistema de dirección, aún mantiene varias conexiones mecánicas del sistema convencional de dirección, ya que utiliza una caja de reducción y un pequeño sistema piñón-cremallera por cada llanta para permitir que los motores eléctricos controlen los brazos de dirección. Ya que este proyecto busca iniciar investigaciones encaminadas a sistemas de dirección cien por ciento By Wire, se justifica la utilización de un motor por cada rueda, para permitir ampliar con facilidad investigaciones a sistemas de cuatro ruedas direccionales.

Decidida la forma de actuar sobre la dirección de las ruedas con los actuadores eléctricos, la arquitectura del resto del sistema se diseñó basándose en las características generales de un sistema By Wire, por lo que el mismo cuenta, además de los actuadores eléctricos para las ruedas, con un volante con capacidad de retroalimentación de fuerza para simular la resistencia al giro de las llantas, una unidad central electrónica conformada por una computadora portátil, una tarjeta de adquisición de datos y tres microcontroladores, además de la etapa de potencia que permite al sistema manejar la corriente requerida por los motores. La gráfica presentada en la figura 1.7, muestra una visión general de la arquitectura del sistema propuesto a implementarse como parte de la plataforma de investigación.

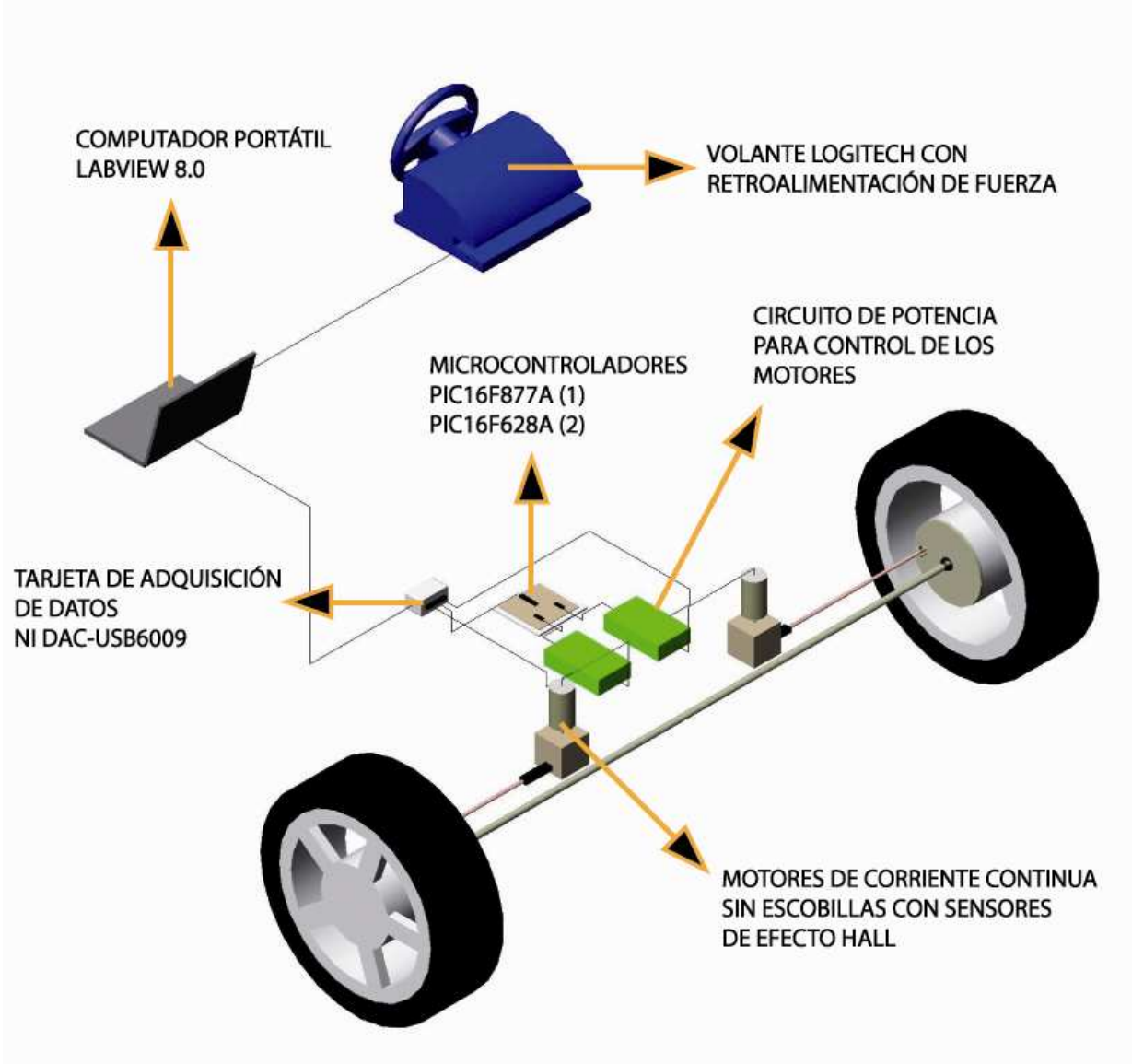


Figura 1.7 Arquitectura del sistema Steer By Wire propuesto

Una vez determinada la arquitectura del sistema se plantearon las etapas a ser cumplidas para que el sistema implementado en la plataforma se desempeñe como un sistema de dirección By Wire. La primera etapa se enfoca en controlar la posición de los motores eléctricos que determinan la posición de cada rueda. En la segunda etapa se estudia la forma de retroalimentar fuerza al volante, para que el conductor no pierda la sensación de control sobre la dirección del vehículo y esté conciente de fuerzas que se opongan al movimiento de las ruedas. En la última etapa se detalla el desarrollo de una aplicación de control en la que se unen y sincronizan las etapas de control de posición con la retroalimentación de fuerza para poner completamente en funcionamiento el sistema By Wire.

A continuación, cada una de las etapas es tratada en un capítulo independiente, en los que se profundizan los detalles técnicos y pasos seguidos para completarlas con éxito a lo largo del desarrollo del proyecto.

Capítulo 2

Control de posición de las ruedas

El control de posición de los actuadores eléctricos es clave en un sistema de dirección By Wire, ya que son estos actuadores los encargados de direccionar las ruedas correctamente evitando que el conductor pierda control sobre el vehículo. Considerando que, con fines demostrativos, la plataforma podría expandirse al punto de controlar un sistema de dirección a escala reducida, se dio importancia a la selección de actuadores cuya posición pudiera ser controlada con precisión y que a la vez entregaran un torque mediano, alrededor de 1 N.m, suficiente para orientar ruedas de un modelo a escala. Tomando en cuenta estos requerimientos, se consideraron dos posibles tipos de motores eléctricos para ser usados como actuadores para direccionar las ruedas. El primer tipo de motores considerados fueron motores paso a paso y el segundo tipo fue de motores de corriente continua sin escobillas. Al final se seleccionaron los motores de corriente continua sin escobillas, debido a su menor costo, su disponibilidad en el mercado, y mayor torque para el mismo tamaño en comparación con un motor paso a paso. Es importante notar que si se quisiera implementar el sistema en un vehículo convencional, la mejor opción también estaría dada por los motores de corriente continua sin escobillas, ya que el rango de torque de los motores paso a paso no llega al nivel que se requeriría para orientar las ruedas en un sistema de dirección a escala real. A continuación, se revisan fundamentos teóricos sobre el funcionamiento de los motores de corriente continua sin escobillas y su control. Además, se revisa la implementación práctica del control de posición de los motores utilizados en este proyecto.

2.1 Introducción al funcionamiento de un motor DC sin escobillas

Los motores de corriente continua sin escobillas (BrushLess Direct Current Motors en inglés), son motores de tipo síncrono, es decir que se caracterizan porque el campo magnético generado por el estator gira a la misma frecuencia que el campo magnético generado por el rotor, por lo que no presentan el deslizamiento que se presenta en tipos de motores como los de inducción. Los motores BLDC (por sus siglas en inglés), pueden tener una, dos o tres fases, pero la configuración de tres fases es la más común. Como se mencionó anteriormente, este proyecto hizo uso de dos motores BLDC trifásicos como actuadores independientes para cada rueda, por lo que esta introducción teórica se enfoca en los motores de corriente continua sin escobillas de tres fases.

El funcionamiento de un motor BLDC está directamente relacionado con la forma de funcionar de un motor de corriente continua con escobillas, conocidos como motores DC. En los motores DC tradicionales, al hacer circular corriente por un bobinado dentro de un campo magnético, generado por imanes permanentes ubicados en su estator, se produce una fuerza electromagnética, la cual genera torque en el bobinado provocando el giro del rotor. Este giro, tiende a alinear la normal al plano del bobinado con el campo magnético. Ya que, como se puede observar en la figura 2.1, la normal al plano del bobinado se alinea con el campo magnético solo con dar media vuelta al rotor, es necesario cambiar la dirección de la corriente cada media vuelta para que el rotor complete una vuelta entera. El cambio de dirección en la corriente, cuando el rotor alcanza una posición determinada, produce un cambio en el campo electromagnético del motor, y se conoce como conmutación.

La necesidad de conmutar cada media vuelta está relacionada con el torque generado y la dependencia del mismo con el ángulo que se forma entre la normal al

bobinado y el campo magnético de los imanes. La figura 2.1 muestra un perfil del bobinado y el ángulo en cuestión.

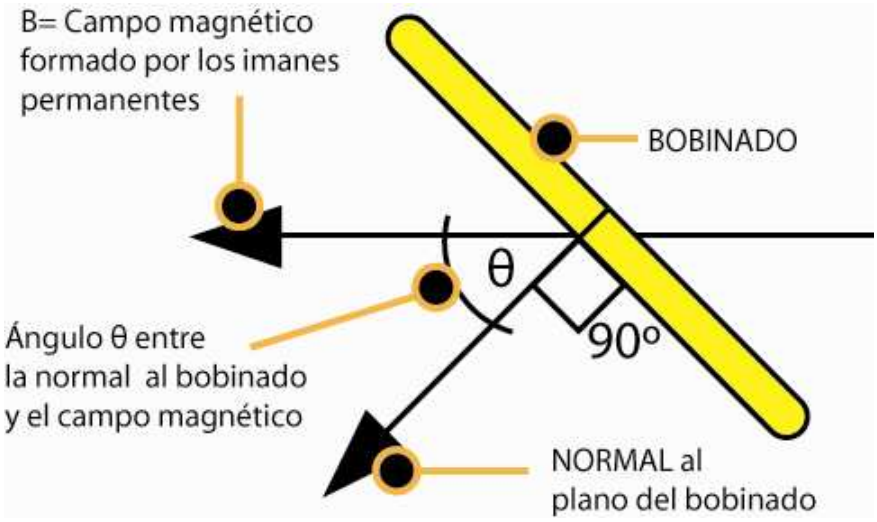


Figura 2.1 Esquema para deducir el torque de un MotorDC (DC Electric Motors)

Ya que el torque se define como fuerza por el brazo de fuerza, la fórmula para calcular el torque que hace girar el motor dependiendo de la posición del rotor se expresa de la siguiente manera:

$$\text{Torque} = \text{Fuerza} \times (\text{Ancho del bobinado}/2) \sin \theta$$

A partir de la fórmula expuesta y su dependencia del seno del ángulo θ, se explica por qué el bobinado daría solo media vuelta si no existiera conmutación. Sin θ es una función que se hace cero cada 180 grados, por esto el torque es cero cada media vuelta y es necesario conmutar en dicho punto para generar torque que permita que el rotor complete una vuelta. Por otro lado sin θ, alcanza su máximo en θ = 90º, lo que quiere decir que el torque de giro del rotor es máximo cuando existe un ángulo de 90º entre el campo magnético y la normal al plano del bobinado. Este punto es conocido como plano neutral (Hubert, 359).

En la figura 2.2, se pueden observar las dos escobillas del motor DC. Las escobillas son partes fundamentales para la conmutación en un motor DC. Cada escobilla se conecta a una de las dos terminales de una fuente de alimentación DC. Como se puede observar en la figura, los extremos del bobinado se conectan a dos semicírculos separados que hacen contacto con las escobillas. Estos semicírculos giran cuando gira el rotor, y por lo tanto cada media vuelta, los semicírculos cambian la terminal de la fuente de alimentación DC a la que se encuentran conectados por medio de las escobillas. Esto produce la conmutación deseada para el giro del motor, permitiendo alcanzar un torque máximo cada media vuelta del rotor. El partir un disco en dos mitades formando los semicírculos, es una forma mecánica de asegurar que la conmutación se produzca cada media vuelta del bobinado.

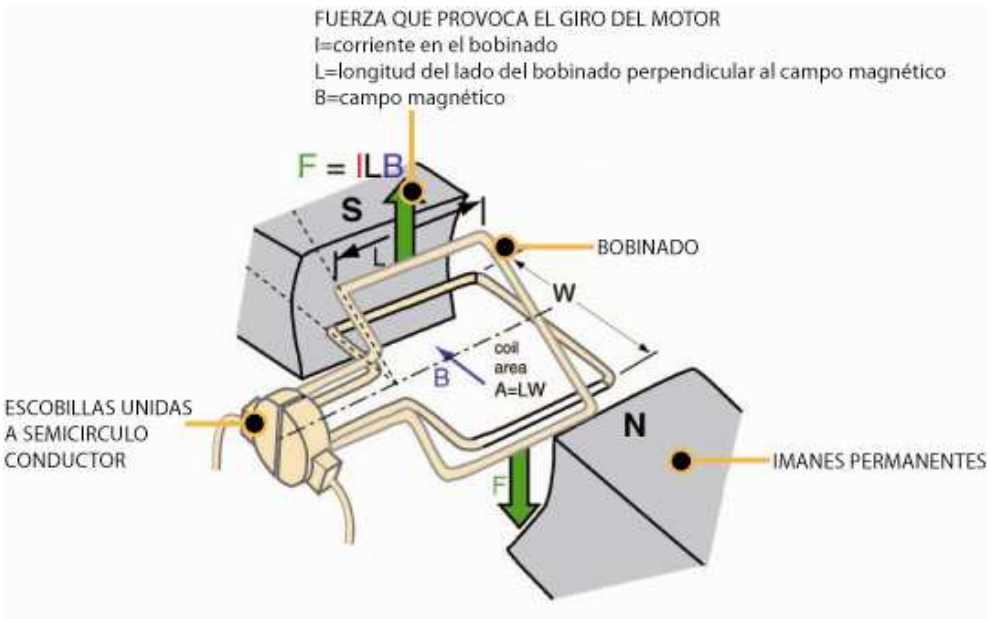


Figura 2.2 Esquema funcionamiento de un MotorDC (DC Electric Motors)

Ahora, es importante notar que en la práctica los motores DC no tienen un solo bobinados. El ejemplo de un solo bobinado es importante para explicar la necesidad de la conmutación y el comportamiento del torque obtenido de acuerdo a la posición del

bobinado. Pero un motor de un solo bobinado resulta poco práctico debido a que no es posible obtener su torque máximo continuamente, como se requiere en la mayoría de aplicaciones.

Al aumentar el número de bobinados, cada bobinado pasa por el plano neutral en diferentes instantes. A medida que un bobinado se mueve fuera del plano neutral, el siguiente bobinado está entrando en dicho plano (Hubert, 360). Esto provoca que prácticamente en cada instante durante la vuelta del rotor, exista siempre un bobinado alineándose con el plano neutral, y por lo tanto, generando un torque máximo en el rotor.

Las figuras 2.3 y 2.4, presentan gráficas aproximadas del torque obtenido en función de la posición del rotor para un motor de un solo bobinado y para un motor de varios bobinados, respectivamente. Las gráficas permiten observar con mayor claridad la diferencia entre un motor de un solo bobinado, que alcanza su torque máximo cada 180º de giro del rotor, y un motor de varios bobinados, que alcanza el torque máximo varias veces durante los mismos 180º. A partir de la figura 2.4, es claro que mientras mayor sea el número de bobinados más tenderá el torque a mantenerse constante en su valor máximo.

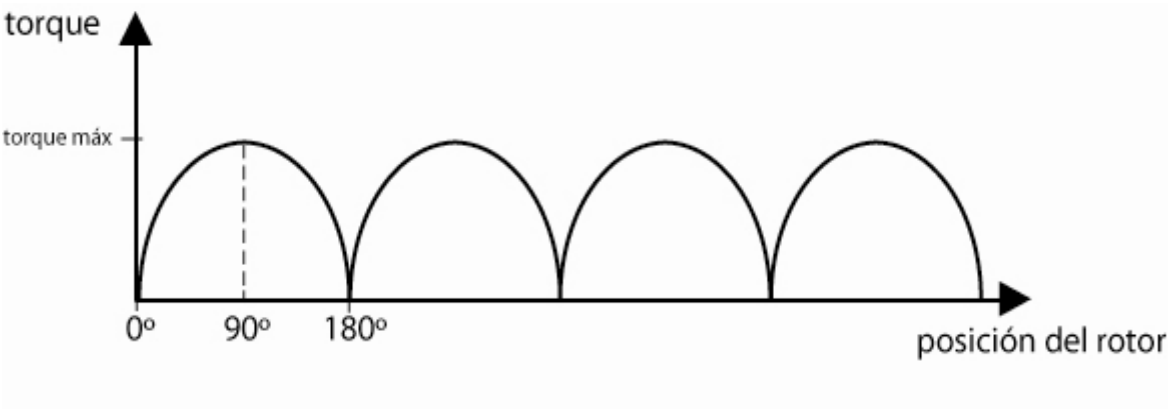


Figura 2.3 Torque en función de la posición del rotor para un motor DC de un solo bobinado

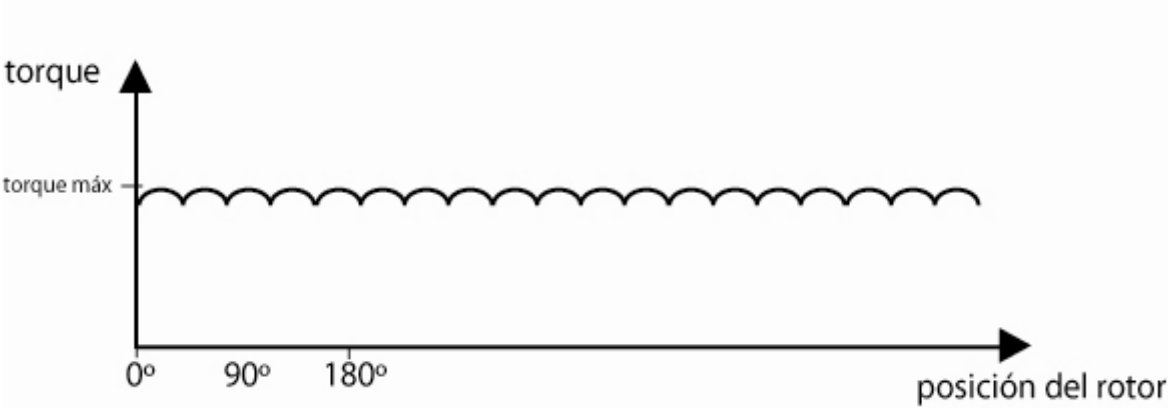


Figura 2.4 Torque en función de la posición del rotor par un motor DC con varios bobinados

Una vez comprendidos los fundamentos de operación de un motor DC con escobillas, es posible entrar en detalles de los motores BLDC. A diferencia de los motores DC con escobillas, los motores BLDC tienen los bobinados en el estator y los imanes permanentes en el rotor, pero la teoría del giro provocado por la fuerza generada por la interacción entre la corriente que circula por los bobinados y el campo magnético de los imanes permanentes, se mantiene igual.

Como se puede observar en la figura 2.5, el estator de un motor BLDC está hecho de láminas de acero ubicadas en su periferia circular, las cuales son cortadas axialmente para formar espacios alternados donde se ubican los grupos de bobinados del estator. Los grupos de bobinados se distribuyen en la periferia del estator de tal forma que generan un número par de polos. Los motores BLDC pueden ser agrupados en dos grupos según el tipo de conexión que tienen sus bobinados en el estator. La primera configuración, conecta los bobinados en configuración delta, lo que significa que los bobinados se conectan en serie y se alimenta cada una de las conexiones. El segundo tipo de configuración, es la conexión en Y, la cual conecta todos los bobinados a un punto central y la alimentación se aplica en los extremos libres de cada bobinado. Las especificaciones de voltaje de un

motor BLDC hacen referencia al voltaje de control máximo para el que está diseñado el motor. Si bien existen motores de mayor voltaje, los motores de 48V o menos son utilizados normalmente en aplicaciones automotrices.

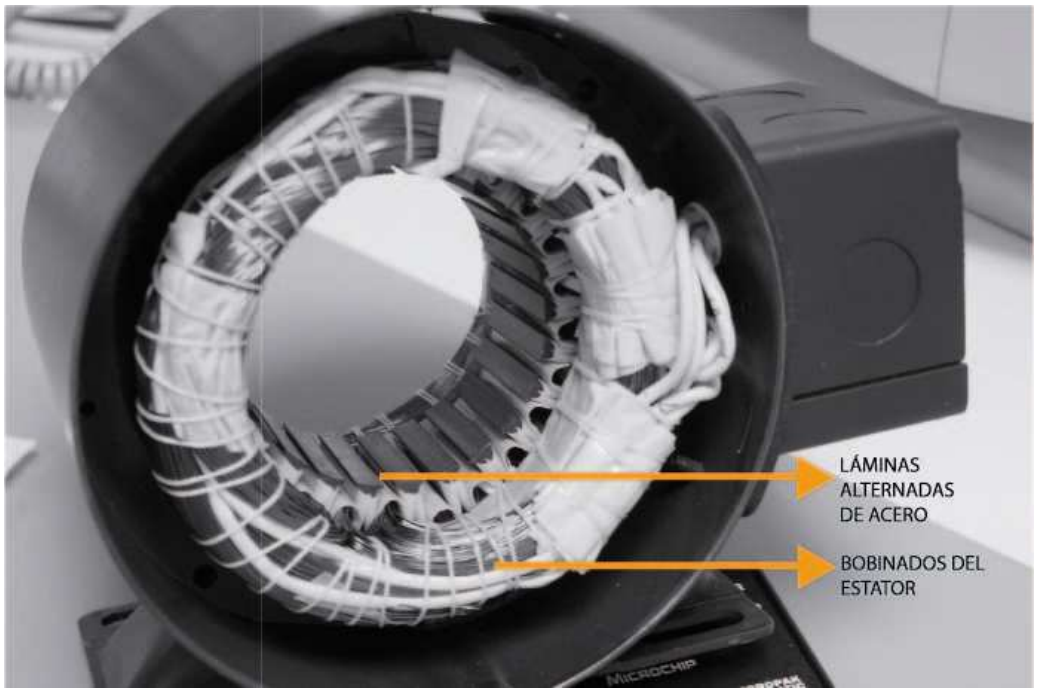


Figura 2.5 Estator de un motor BLDC (Yedamale, 3)

El rotor de un motor BLDC está hecho de imán permanente. En un motor BLDC el rotor puede presentar desde 2 hasta 8 pares de polos alternados, dependiendo del diseño. El material que se utiliza para hacer los imanes utilizados en estos rotores depende de la densidad del campo magnético que se necesita. Tradicionalmente se utiliza la ferrita, pero hoy en día las aleaciones están siendo tomadas en cuenta, ya que a pesar de ser más costosas, en comparación con la ferrita, permiten mayor densidad de flujo en un volumen dado. Esto, a la vez, significa un mayor torque para un motor hecho de aleaciones que para un motor, del mismo tamaño, hecho de ferrita. Algunas de las aleaciones utilizadas en los

rotores son el Neodimio (Nd), Cobalto de Samario (SmCo) y la aleación de Neodimio, Ferrita y Boro (NdFeB) (Yedamale, 4).

Ya que en un motor BLDC los bobinados se encuentran en el estator, que es la parte inmóvil del motor; ya no existe movimiento de los bobinados que pueda ser utilizado para controlar mecánicamente la conmutación en el motor. Debido a esta característica, en un motor BLDC, es posible prescindir de las escobillas, de allí su nombre, pero la alimentación de los bobinados correspondientes a cada fase tiene que ser controlada electrónicamente, es decir, alimentando las fases según una secuencia específica. Al controlar electrónicamente la secuencia de alimentación de las fases, se hace necesario encontrar una forma de conocer la posición del rotor para que la conmutación electrónica se realice correctamente, consiguiendo el giro continuo que se espera de un motor.

Los motores BLDC ofrecen algunas ventajas sobre los motores DC con escobillas. Entre las principales, se encuentran: la necesidad muy baja de manutención debido a la ausencia de escobillas, una vida útil mayor, características de torque-velocidad que permiten su operación a cualquier velocidad bajo una carga nominal, alta eficiencia debido a la ausencia de escobillas que producen caídas de voltaje, mejor disipación de calor debido a que los bobinados se encuentran cerca de la carcaza del motor, reducción del ruido eléctrico producido. El costo de producción y la complejidad del control necesarios para un motor BLDC, podrían considerarse como los únicos contras respecto a un motor DC con escobillas.

2.2 Funcionamiento detallado de un motor DC sin escobillas

La figura 2.6 muestra ejemplos gráficos de las diferentes señales que se pueden medir en un motor de corriente continua sin escobillas. A partir de esta figura se puede explicar brevemente el funcionamiento de este tipo de motor y confirmar su relación con el funcionamiento de los motores DC con escobillas. En cada paso de la secuencia de alimentación, una de las fases es conectada a la terminal positiva de la fuente de alimentación, la segunda fase es conectada a la terminal negativa, mientras que la tercera fase se la deja flotante es decir sin energizar. En tales condiciones de alimentación, se produce torque debido a que fluye corriente por el bobinado alimentado, generando un campo magnético que interactúa con el campo magnético generado por los imanes permanentes del rotor. Al igual que en un motor DC con escobillas, el torque máximo se genera cuando los dos campos mencionados se encuentran a 90º entre ellos. Es necesario continuar con la secuencia de alimentación de los bobinados para mantener el rotor girando y obtener continuamente su torque máximo.

Queda claro, que la rotación de un motor de corriente continua sin escobillas de tres fases, se consigue alimentando sus fases en una secuencia determinada con el fin de generar un campo magnético giratorio. Y que además, para controlar la rotación de los motores adecuadamente, es necesario conocer la posición del rotor antes de energizar el siguiente bobinado, evitando así pérdida de pasos o vibraciones debido a cambios bruscos de dirección por fallas en la secuencia de energización. Debido a esta necesidad, se han desarrollado diversos métodos para conocer la posición del rotor de un motor. A continuación se tratan los dos métodos más comúnmente utilizados para un motor BLDC.

El primer método utilizado para conocer la posición del rotor, es posible si el motor se construye con sensores ubicados en el estator. En la actualidad, la mayoría de los motores BLDC son construidos con sensores de efecto Hall para facilitar la medición de la

posición de su rotor. Los sensores se ubican en el lado del estator más alejado de donde se coloca la carga, y los mismos generan señales lógicas altas y bajas dependiendo si el polo del rotor que pasa cerca del sensor es Norte o Sur. Refiriéndose nuevamente a la figura 2.6, se puede observar que la secuencia de alimentación, también conocida como secuencia de conmutación, consta de 6 pasos. El cambio en la alimentación de los bobinados se ve representado por el cambio de la corriente por fase cada 60 grados eléctricos. Correspondiendo a los cambios en la alimentación de las fases, las señales de los sensores de efecto Hall, al igual que la magnitud de la fuerza contra electromotriz, cambian su estado también cada 60º eléctricos.

Al combinar las señales de los sensores se puede determinar la ubicación exacta del rotor con una resolución igual a 360º divididos para el número de pasos que completan una revolución mecánica del motor. En el ejemplo de la figura 2.6, se observó que el motor da 6 pasos por ciclo eléctrico y 2 ciclos eléctricos por revolución mecánica. Es decir el motor da un total de 12 pasos por revolución mecánica. Esto da una resolución de 360º/12=30º en la posición del rotor.

Es importante conocer que un ciclo eléctrico no necesariamente corresponde a una revolución mecánica del rotor. El número de ciclos eléctricos que corresponden a una revolución mecánica es igual al número de pares de polos que tiene el rotor del motor. En este caso el motor tendría 4 polos, es decir 2 pares de polos, ya que por cada revolución mecánica se completan dos ciclos eléctricos. Como solo uno de los tres sensores de efecto Hall cambia de estado cada 60º eléctricos, se genera un código binario al juntar los 3 bits que representan la señal de cada sensor. Es este código que se utiliza para conocer la posición del rotor y el paso de la secuencia de alimentación que corresponde ejecutar a continuación. En la figura 2.6, la parte correspondiente a las señales de los sensores de

efecto Hall, muestra un ejemplo del código binario que se genera para el primer ciclo eléctrico del motor.

Los motores utilizados en este proyecto tienen 2 pares de polos, por lo que su resolución es de 30° y completa 2 ciclos eléctricos por revolución mecánica, igual que el motor utilizado como ejemplo a partir de la figura 2.6.

El segundo método para conocer la posición del rotor, consiste en medir la fuerza contra electromotriz producida por el motor. De acuerdo a la Ley de Lenz, la fuerza contra electromotriz corresponde al voltaje generado por cada uno de los bobinados, cuya polaridad es contraria al voltaje que energiza los bobinados (Yedamale, 12). La fuerza contra electromotriz (fcem), depende principalmente de la velocidad angular del rotor, el campo magnético producido por los polos del rotor y del número de vueltas con las que cuentan los bobinados en el estator. Ya que una vez que el motor ha sido fabricado, el número de vueltas de los bobinados se mantiene constante, el valor de la fuerza contra electromotriz es dominado por la velocidad angular y puede ser aproximada a partir de la siguiente relación:

$$FCEM = RPM / Kv$$

Donde Kv es la constante de voltaje del motor y RPM es la velocidad del motor en revoluciones por minuto. Para todos los motores, su hoja técnica detalla la fuerza contra electromotriz correspondiente a la velocidad nominal. Con dichos datos se puede obtener la constante Kv, y luego utilizarla para estimar la fuerza contra electromotriz para una velocidad dada del motor.

En la figura 2.6, se pueden observar las gráficas para la fuerza contra electromotriz medida entre fases para cada paso en la secuencia de alimentación del motor. Al relacionar estás gráficas con las de los sensores de efecto Hall, se puede observar que las señales de los sensores de efecto Hall cambian de estado cada vez que la polaridad de una de las

señales de la fcem cambia de positivo a negativo o de negativo a positivo. Así el cruce por cero de las señales de la fcem puede ser utilizado para determinar la posición del rotor. Debido a la proporcionalidad de la fcem respecto a la velocidad del rotor, este método presenta inconvenientes el momento de emplearlo a velocidades muy bajas donde la fcem es prácticamente nula dificultando la detección de su cruce por cero. Normalmente este problema se soluciona manejando el motor en lazo abierto hasta que alcanza una velocidad en la que la magnitud de la fcem sea suficiente para medir su cruce por cero, y poder controlarlo en base a la posición del rotor.

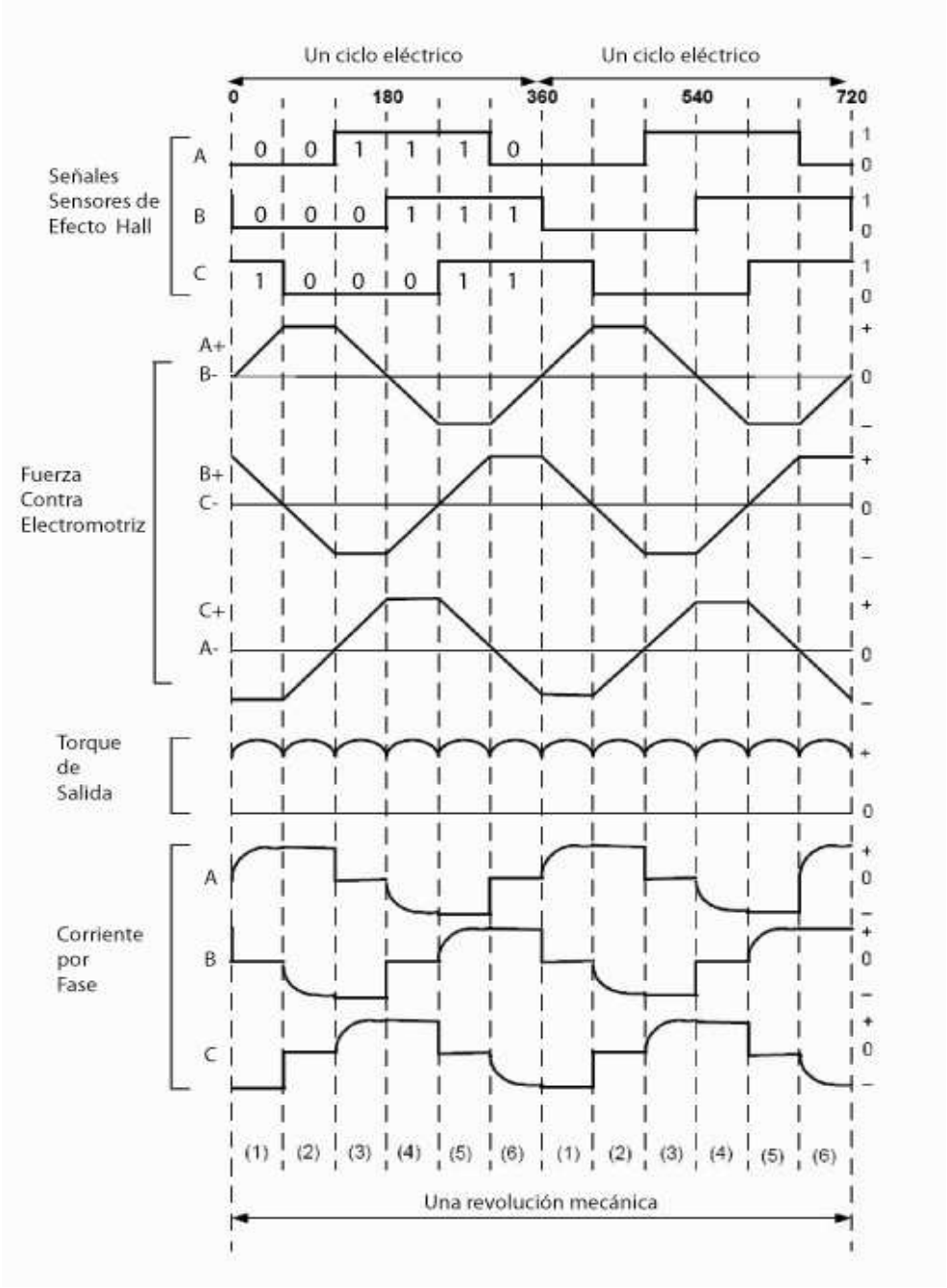


Figura 2.6 Sensores efecto Hall, FCEM, Torque de Salida y Corriente por Fase (Yedamale, 9)

En la figura 2.7, se muestra un esquema gráfico de la configuración típica de un motor de corriente continua sin escobillas construido con sensores de efecto Hall.

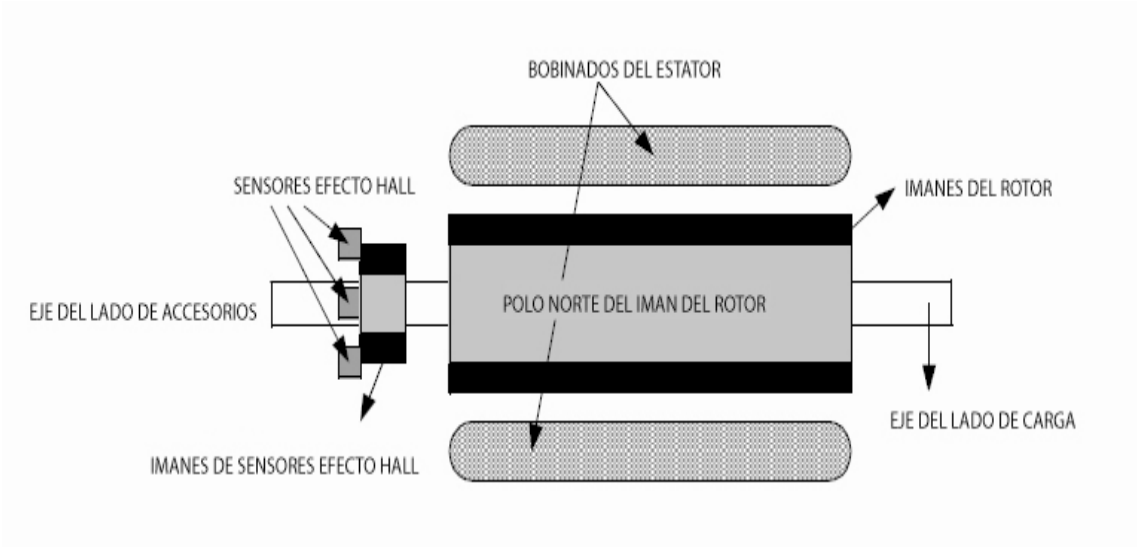


Figura 2.7 Esquema de un motor de corriente continua sin escobillas (Yedamale, 5)

2.3 Características Torque-Velocidad de motores BLDC

Las características de torque-velocidad son otro elemento importante en la teoría de los motores de corriente continua sin escobillas. Las características del torque de un motor BLDC normalmente están descritas por su Torque Pico y su Torque Nominal. Un motor que opera continuamente solo puede soportar carga hasta el valor de su torque nominal. El torque se mantiene constante hasta alcanzar la velocidad nominal del motor. En ese momento si se continúa aumentando la velocidad del motor, el torque comienza a disminuir. Generalmente los motores pueden alcanzar una velocidad igual al 150% de su velocidad nominal. Existen aplicaciones para los motores BLDC que necesitan que los mismos operen intermitentemente, con inicios y paradas, así como cambios de dirección

frecuentes en un lapso corto de tiempo. En algunos de eso casos, el torque necesario puede ser mayor que el torque nominal, principalmente si el motor tiene carga y arranca después de estar completamente parado. El torque extra es necesario para superar la inercia de la carga sobre el rotor. El motor puede alcanzar su Torque Pico siempre y cuando siga su curva de torque-velocidad. La figura 2.8 muestra una gráfica utilizada como referencia de las zonas de funcionamiento de un motor BLDC en función de su velocidad.

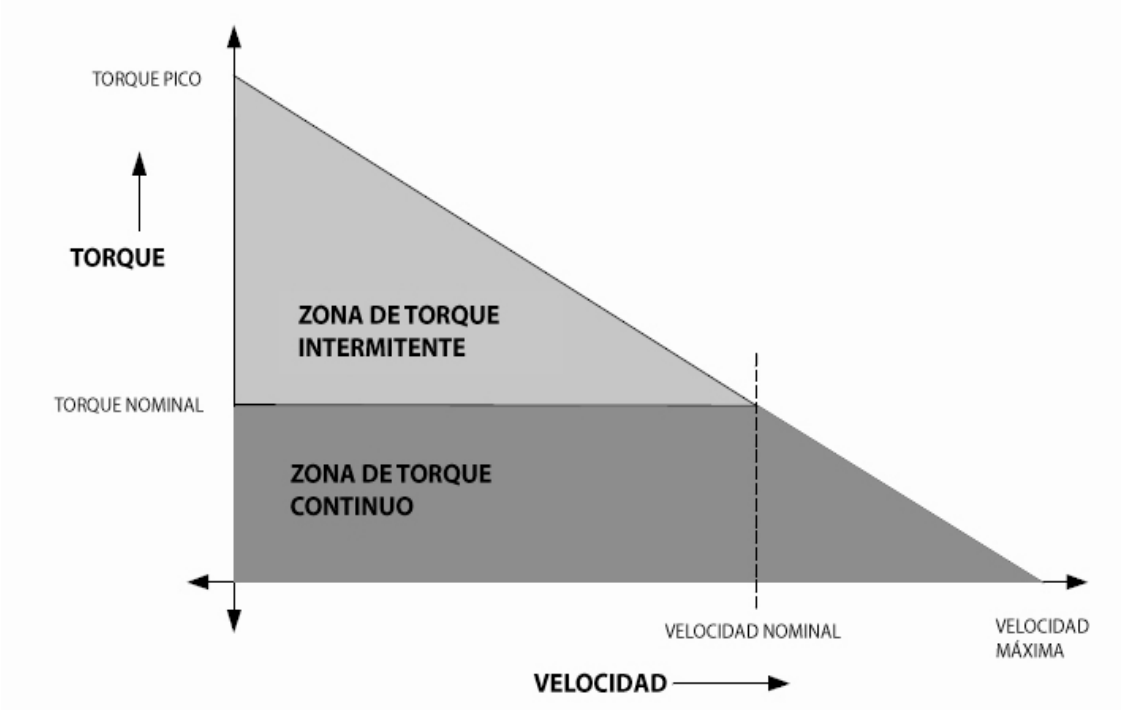


Figura 2.8 Referencia zonas de funcionamiento para un motor BLDC (Yedamale, 6)

2.4 Fundamentos para el control de motores DC sin escobillas

Esta sección revisa teóricamente los requisitos para construir un controlador para un motor de corriente continua sin escobillas. El enfoque del controlador solo será para motores BLDC con sensores de efecto Hall, ya que estos serán los utilizados en el proyecto.

La figura 2.9 muestra dos esquemas simplificados para un motor BLDC. El primero de ellos muestra un diagrama de los bobinados del estator junto al imán del rotor y el código binario generado por los sensores de efecto Hall para cada posición del rotor. El segundo diagrama muestra los bobinados junto a los pasos de la secuencia de conmutación a seguir para conseguir que el motor gire correctamente.

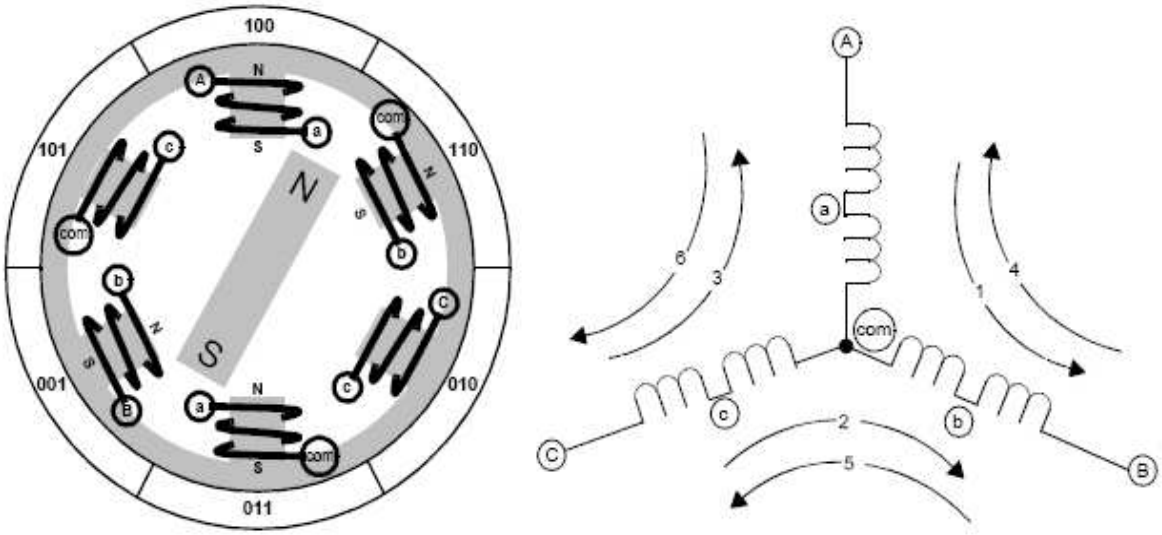


Figura 2.9 Diagramas simplificados de un motor BLDC (Brown, 1)

A partir de los diagramas expuestos resulta más fácil comprender la disposición de los bobinados formando tres circuitos electromagnéticos que se conectan en un punto en común. Cada uno de los circuitos electromagnéticos se divide en la mitad, (puntos a, b y c), para permitir que el rotor se ubique en la mitad del campo magnético inducido. Como los

motores con los que se trabajará tienen sus tres fases conectadas en estrella al igual que en el diagrama, los mismos se controlarán energizando dos de sus fases en cada paso de la secuencia de conmutación.

Como se mencionó en la primera sección de este capítulo, conocer la posición del rotor es clave para poder controlar correctamente la conmutación del motor y energizar las fases para que el motor produzca el mayor torque disponible. Teóricamente, los bobinados correctos son activados si el rotor se encuentra a 120º de alinearse con el campo magnético producido por dichos bobinados. Los mismos bobinados serán correctamente desactivados cuando el rotor se encuentre a 60º de alinearse. En dicho punto se activará el siguiente circuito magnético, y dicha secuencia es la que se repite continuamente para mantener girando el rotor. La presencia de sensores facilita considerablemente los cálculos que corresponderían para cumplir con los requerimientos que se acaban de mencionar para activar y desactivar correctamente los circuitos magnéticos del estator (Yamadale, 5).

Los motores utilizados en el proyecto constan de 3 sensores de efecto Hall, los cuales tienen como salida un 1 lógico durante 180º de rotación eléctrica, y un 0 lógico para los otros 180º de rotación. Como cada sensor se encuentra ubicado a 60º eléctricos de desfases respecto uno del otro, cada sensor está alineado con uno de los circuitos magnéticos. En la figura 2.10, se puede observar el efecto del desfase de los sensores en las señales de salida que se obtiene de los mismos y su relación con los pasos de la secuencia de alimentación. Además se observa que fases se encuentran alimentadas correspondiendo a cada estado del rotor dado por el código binario de 3 bits generado por la sobreposición de las señales de los sensores de efecto Hall. El bit más significativo corresponde al sensor C y el menos significativo al sensor A. En la misma figura se puede observar que cada estado de alimentación, corresponde a la conexión de una de las fases a la terminal positiva de la fuente alimentación, de la segunda fase a la terminal negativa y a dejar la fase

sobrante en estado flotante, es decir sin conexión a ninguna de las terminales de la fuente de alimentación.

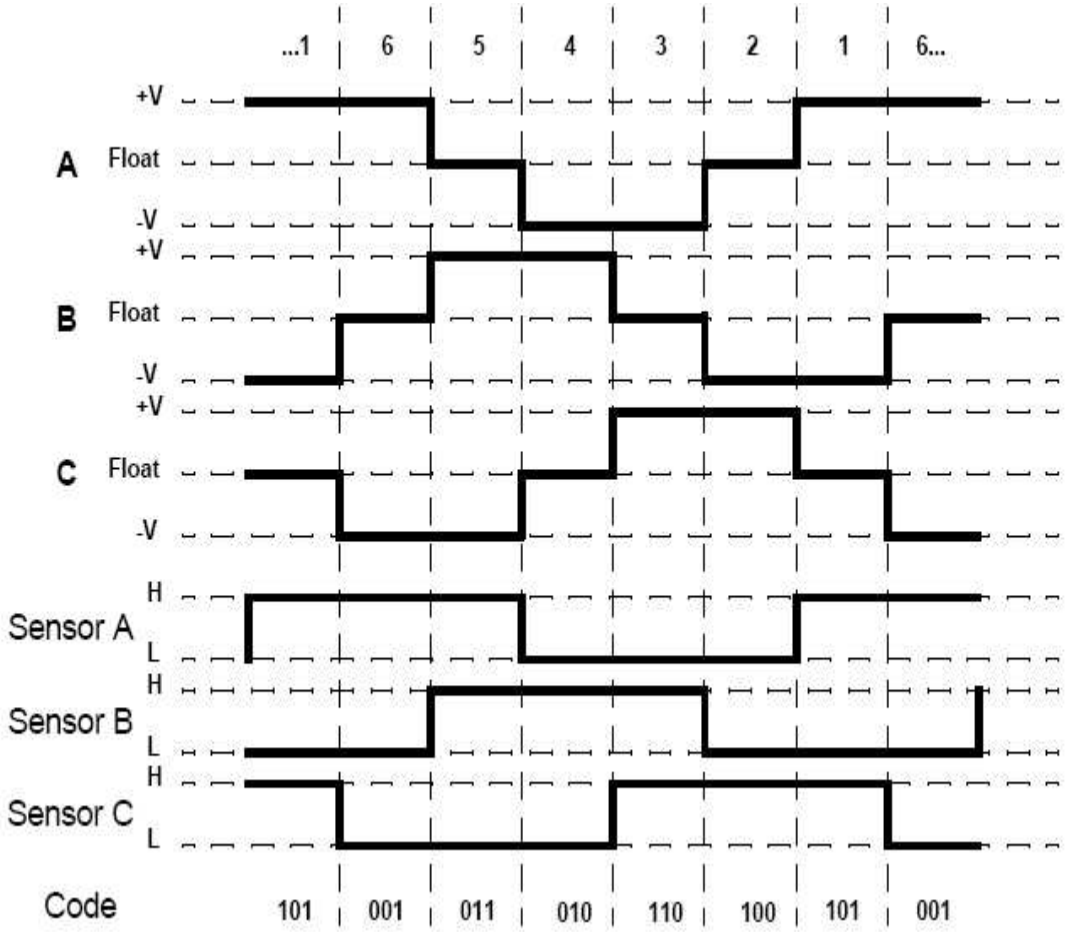


Figura 2.10 Relación entre fases alimentadas y señales de los sensores de un motor BLDC (Brown, 2)

Conociendo las necesidades de alimentación de cada fase, se pasa al diseño de un circuito eléctrico capaz de satisfacer tales necesidades. Desde una visión muy elemental, dicho circuito constaría de 6 interruptores, 2 interruptores por fase. El primer interruptor se conecta entre la terminal positiva de la fuente de alimentación y la fase correspondiente, y el segundo interruptor se conecta entre la fase y la terminal negativa de la fuente de alimentación. Está disposición de los interruptores es conocida como medio puente H. Ya

que cada motor cuenta con tres fases, su control se logra con un circuito de tres medios puentes H. De esta manera, si los dos interruptores están abiertos, la fase no se encuentra conectada a ningún lado y se consigue el estado flotante de la misma. Si se cierra el primer interruptor, dejando abierto el segundo, se conecta la fase a la terminal positiva de la fuente de alimentación. Por último, al abrir el primer interruptor y cerrar el segundo interruptor, la fase se conecta a la terminal negativa de la fuente de alimentación. De esta manera se consiguen los tres estados requeridos por fase para poder controlar el motor. Es importante tomar en cuenta que si se llegan a cerrar los dos interruptores se provocaría un corto circuito, por lo que es importante que la aplicación de control se asegure de que nunca se alcance dicho estado de los interruptores.

Como es conocido, para poder hacer uso de controladores electrónicos que manejen las altas velocidades de conmutación requeridas por un motor, la electrónica reemplaza los interruptores por transistores. La figura 2.11 muestra un esquema simplificado del circuito de control para un motor de corriente continua sin escobillas, en el que se utilizan dos transistores por fase.

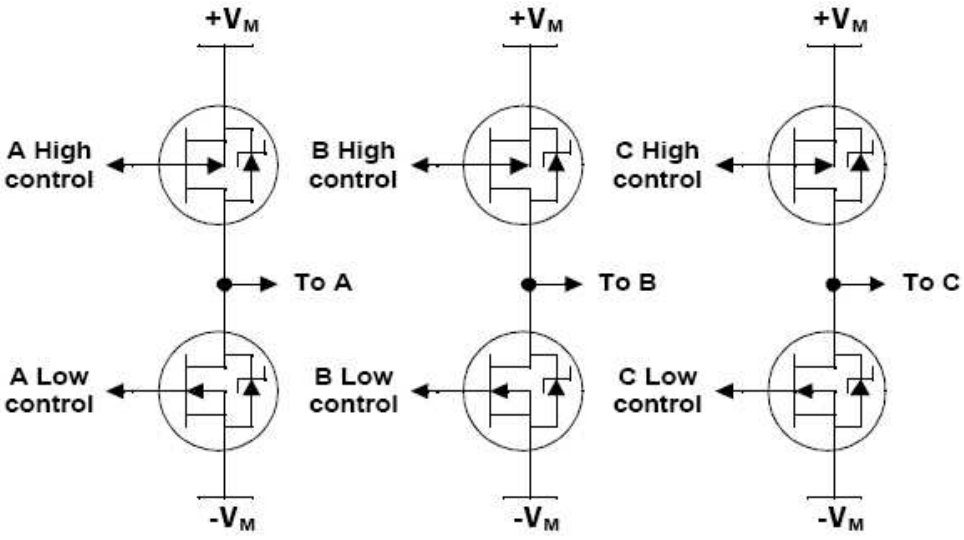


Figura 2.11 Esquema del circuito de control de un motor BLDC (Brown, 3)

Al aplicar este circuito para el control de un motor BLDC, y siempre que la programación de la secuencia de alimentaciónsea correcta, no es necesario preocuparse por la posibilidad de que el transistor del lado bajo se encuentre encendido al mismo tiempo que el transistor del lado alto. Esto se puede comprender fácilmente al regresar y observar la gráfica presentada en la figura 2.10 donde se ve que ninguna fase pasa directamente del estado bajo al alto, o viceversa; sino que siempre existe un lapso de tiempo en el que los dos transistores por fase se apagan antes de realizar cualquiera de los cambios de estados mencionados en los que se podría generar un corto circuito.

Una vez claros los fundamentos básicos necesarios para controlar la posición de un motor de corriente continua sin escobillas, se puede continuar con el diseño y la implementación práctica del controlador que será utilizado en la plataforma de investigación. Este proceso cuenta con dos etapas: una dedicada al diseño y construcción del circuito de potencia, y otra dedicada al diseño y construcción del circuito de control.

2.5 Diseño y construcción del circuito de potencia para los motores BLDC del sistema Steer By Wire propuesto

En esta sección se revisan los pasos seguidos en el desarrollo de la etapa de potencia necesaria para acoplar el circuito de control a los motores de corriente continua sin escobillas utilizados en este proyecto.

Primero, es necesario conocer las características de los motores que se utilizarán y que se planea controlar. Los motores utilizados son motores de corriente continua sin escobillas de la marca MAE. La hoja técnica del tipo de motor utilizado se encuentra en el Anexo A de este documento. Los motores utilizados son de la serie BM05/3. Esta serie tiene un torque continuo de 0,32 Nm a su velocidad nominal de 4000 rpm. La potencia de

los motores es de 133 Watts. El dato más importante a tomar en cuenta el momento de diseñar el circuito de control es la corriente pico máxima, ya que esta es la corriente que tienen que soportar los transistores que se utilicen en el circuito. La corriente pico máxima de los motores utilizados es de 13A. Los motores funcionan a 60V y su fuerza contra electromotriz a la velocidad nominal es de 28V. Además, cada motor está equipado con tres sensores de efecto Hall ubicados a 120º eléctricos el uno del otro, y que pueden ser alimentados con un mínimo de 4V y un máximo de 24V.

Debido a limitaciones en el presupuesto la plataforma contará inicialmente con únicamente 24V de alimentación. Por este motivo, una vez conocidas las especificaciones técnicas de los motores, el circuito de potencia se diseñó para soportar hasta 13A y funcionar con los 24V disponibles para la alimentación. Para esto, se partió de la forma básica de tres medios puentes H revisada en la sección de fundamentos para el control de motores DC sin escobillas. Para cumplir con los requerimientos de corriente se decidió trabajar con MOSFETS que soportan una corriente máxima de 22A. Estos transistores se prenden y apagan dependiendo del voltaje que existe entre la puerta y la fuente del transistor. Entonces, el diseño del circuito se enfocó en los circuitos de prendido y apagado de los MOSFETS.

El diseño inició considerando que el circuito de control se construirá a partir de microcontroladores. La corriente máxima que puede proveer un pin utilizado como salida en un microcontrolador PIC es de 20mA. Para asegurar que los pines del microcontrolador no se quemen, se decidió utilizar opto acopladores para separar el circuito de control del de potencia. Colocando una resistencia de 330Ω a la entrada del opto acoplador, se aseguró que al poner el pin del microcontrolador en 5V el circuito no exija más de 15mA al mismo.

Cada medio puente H utilizado para controlar una fase del motor, tiene dos entradas de control: una para controlar el MOSFET del lado alto del puente y la otra para el control

del lado bajo del puente. La secuencia de alimentación se encarga de evitar que se prendan el transistor del lado alto y el del lado bajo del medio puente H al mismo tiempo. Esto evita que se produzca un corto circuito en la fuente de alimentación.

El voltaje mínimo para el encendido de los MOSFET utilizados es de 4V entre la puerta y la fuente, y el máximo es de 20V. Tomando esto en cuenta, los circuitos para prendido de los MOSFET, hacen uso de divisores de tensión para colocar el voltaje correcto de prendido en la puerta del transistor. Sabiendo que la alimentación es de 24V, para los transistores de tipo P utilizados en los lados altos de los medios puentes H se utilizaron divisores de tensión que fijen el voltaje de encendido alrededor de los 15V. Esto da una diferencia de voltaje de 9V entre la puerta y la fuente del transistor, asegurando el encendido del mismo. Para los transistores de tipo N utilizados en los lados bajos de los medios puentes H, los divisores de tensión fijan el voltaje de encendido en 8V. Como la fuente de los transistores tipo N se encuentra conectada a tierra, los 8V en la puerta son suficientes para su encendido.

El diseño hace uso de las salidas de los opto acopladores para controlar transistores de señal, encargados de activar y desactivar los divisores de tensión, y por lo tanto de prender y apagar los MOSFET.

Terminado el diseño, se implementó el mismo en un software para simulación de circuitos electrónicos, para confirmar que se desempeñe como se esperaba ante la secuencia de conmutación. La figura 2.12 muestra el diseño implementado en el software de simulación. Esta figura muestra únicamente un medio puente H, sin embargo para las simulaciones se utilizaron tres de estos circuitos, tal como tendría el circuito final construido.

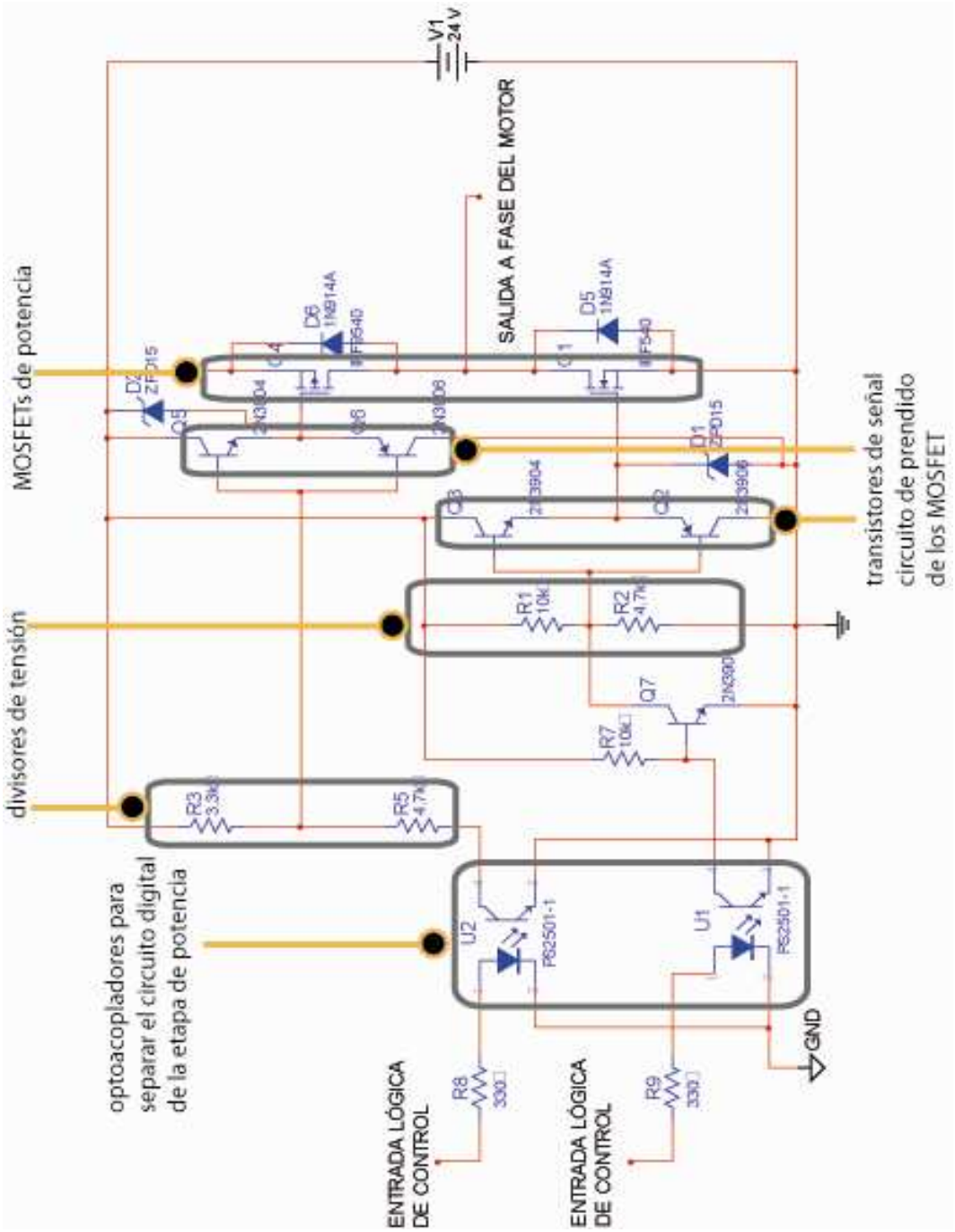


Figura 2.12 Esquemático de medio puente H utilizado para el control de una fase del motor BLDC

Para la simulación de la secuencia de conmutación se utilizaron generadores de ondas cuadradas con ancho de pulso variable, con un voltaje mínimo de 0V y un máximo de 5V, tal como se obtendrían de un microcontrolador. El ancho de pulso se estableció en 2ms, para un periodo de 6ms. A las señales de control de los transistores del lado alto del circuito se dio retrasos de 0.3ms, 0.5ms y 0.1ms, respectivamente. Mientras que para las señales del lado bajo se utilizaron retrasos de 0ms, 0.2ms y 0.4ms. La figura 2.13, muestra como con tales retrasos se logró reproducir la secuencia de conmutación con exactitud para la simulación.

En la figura 2.14, se muestran los resultados para la simulación. Cada una de las señales presentadas corresponde a la salida de un medio puente H del circuito, es decir al punto señalado como SALIDA A FASE DEL MOTOR en el circuito presentado en la figura 2.12. En estas salidas es donde se conectan las tres fases de los motores para energizarlas de acuerdo a la secuencia de conmutación. Como se puede observar, cada una de las tres fases se conecta a los 24V de alimentación en diferentes instantes. Lo mismo ocurre con la conexión a tierra de cada fase. El estado flotante de cada fase, en el que no se conecta a ninguna terminal de la fuente de alimentación, se observa en las pequeñas variaciones que sufre la señal de la salida el momento en que otra salida es conectada a la misma terminal a la que se encontraba conectada la fase anterior, ya que no pueden existir dos fases conectadas a una misma terminal en un mismo instante.

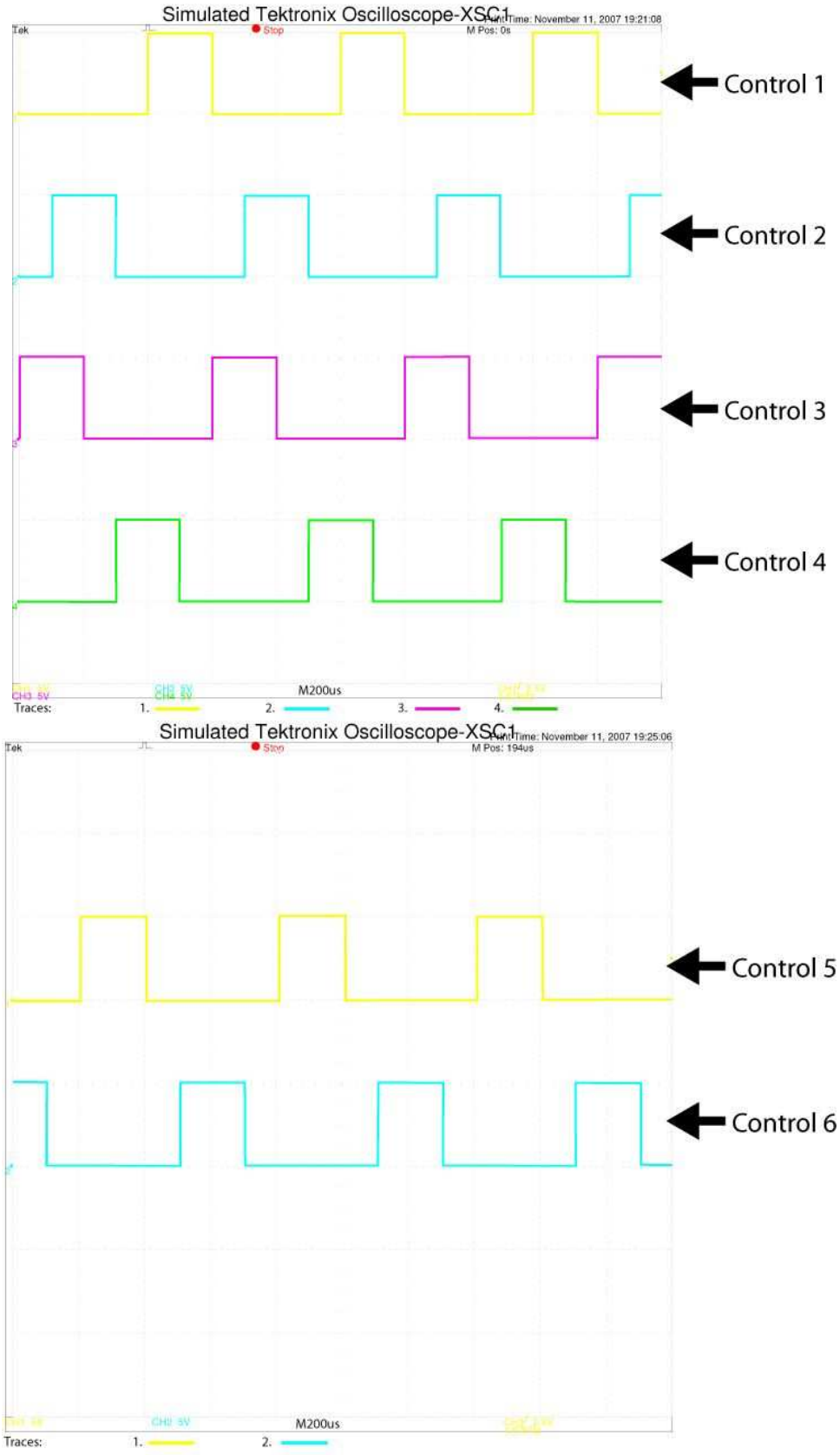


Figura 2.13 Secuencia de conmutación simulada

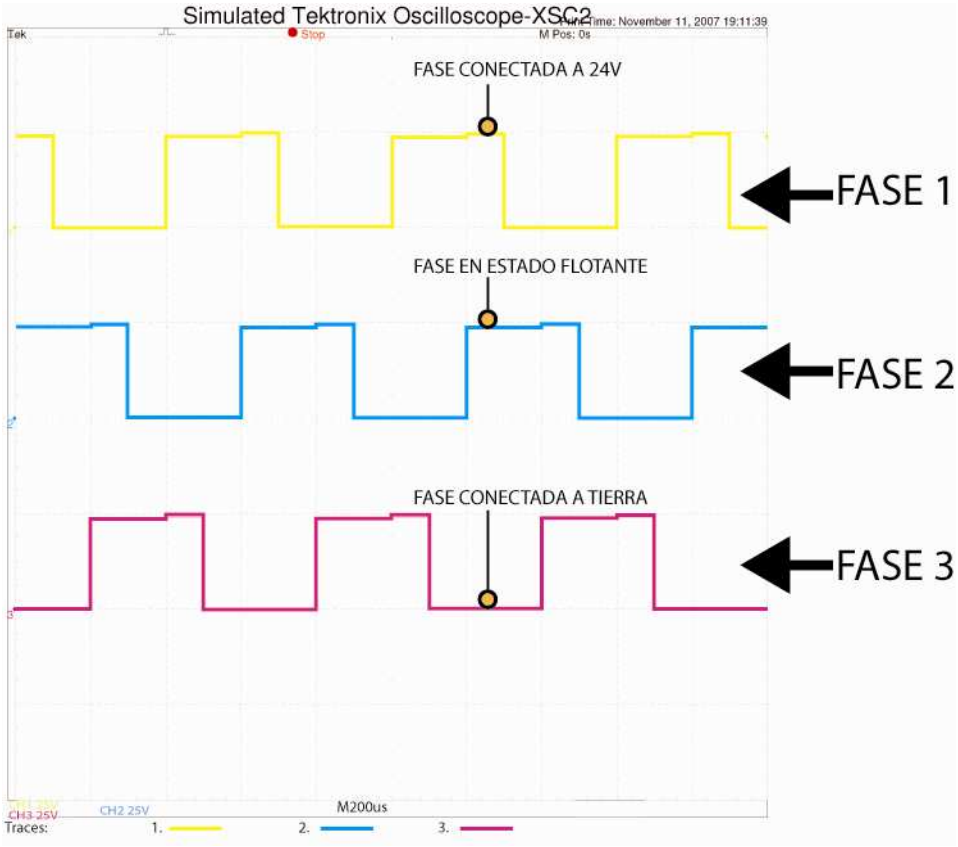


Figura 2.14 Resultado de la simulación del circuito de potencia para un motor BLDC

Una vez confirmado el funcionamiento del circuito en el software de simulación, se prosiguió con la etapa de construcción del circuito. Para esto se buscaron los componentes, disponibles en el mercado, que fueran adecuados para cumplir con los requerimientos del circuito. Los MOSFETS utilizados en la rama final que soporta hasta 23A, son el MOSFET tipo P, IRF9540, para el lado alto del puente, y el MOSFET tipo N, IRF540, para el lado bajo del puente. Los transistores de señal utilizados para activar y desactivar los divisores de tensión son del tipo 2N3904 y 2N3906. Los opto acopladores que se utilizaron fueron los PC817 fabricados por la compañía SHARP. El resto del circuito consta de resistencias, diodos y capacitores, con valores determinados en el diseño. Los componentes se adquirieron en el Laboratorio Técnico ubicado en la esquina de la Avenida

República y 10 de Agosto, en Quito. El costo total del circuito fue de aproximadamente veinte y cinco dólares. La figura 2.15, muestra el circuito terminado.

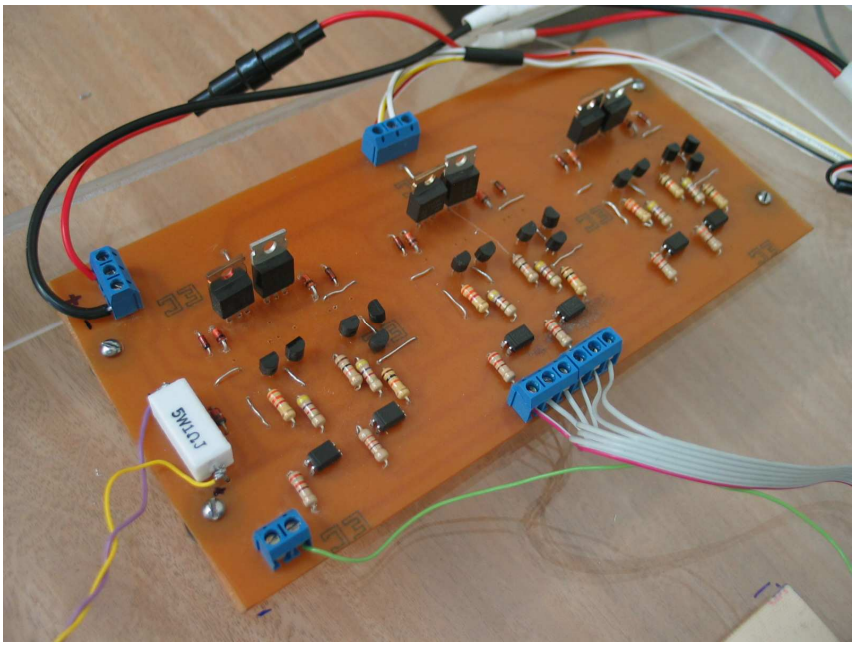


Figura 2.15 Circuito de potencia para acoplamiento del control de un motor BLDC

2.6 Diseño y construcción del circuito de control para los motores BLDC del sistema Steer By Wire propuesto

Terminado el circuito de potencia, se pasó al diseño y construcción del circuito de control que genere la secuencia de conmutación para los motores de acuerdo a una referencia de posición. Como se mencionó en la introducción a la arquitectura del sistema, para esta unidad electrónica de control se decidió utilizar tres microcontroladores. El primer microcontrolador es un PIC16F877A de 40 pines, el mismo que se encarga de recibir la referencia análoga de posición, convertirla a una señal digital, y realizar los cálculos necesarios del número de pasos que tiene que dar el motor para alcanzar la posición deseada. El microcontrolador principal recibe también la información de los

sensores de efecto Hall para determinar el momento que el motor ha alcanzado la posición deseada. Además de este microcontrolador, se utilizaron dos microcontroladores secundarios del tipo PIC16F628 de 18 pines, que se encargan de generar la secuencia de conmutación para cada uno de los dos motores, en base a la lectura de los sensores de efecto Hall. Estos microcontroladores son secundarios ya que solo funcionan al recibir la orden de encendido y dirección de movimiento proveniente del PIC16F877A, el microcontrolador principal.

Como el proyecto propuesto está direccionado a la construcción de una plataforma de investigación, el circuito diseñado tiene conectores con acceso a todo los pines de los tres microcontroladores mencionados, y se incluye un espacio extra en caso de que se desee utilizar un PIC16F628 adicional, en futuras aplicaciones. La figura 2.16, presentada a continuación, muestra el circuito terminado. El código de control para cada microcontrolador será detallado en el Capítulo 4, donde se revisa el desarrollo de la aplicación de control.

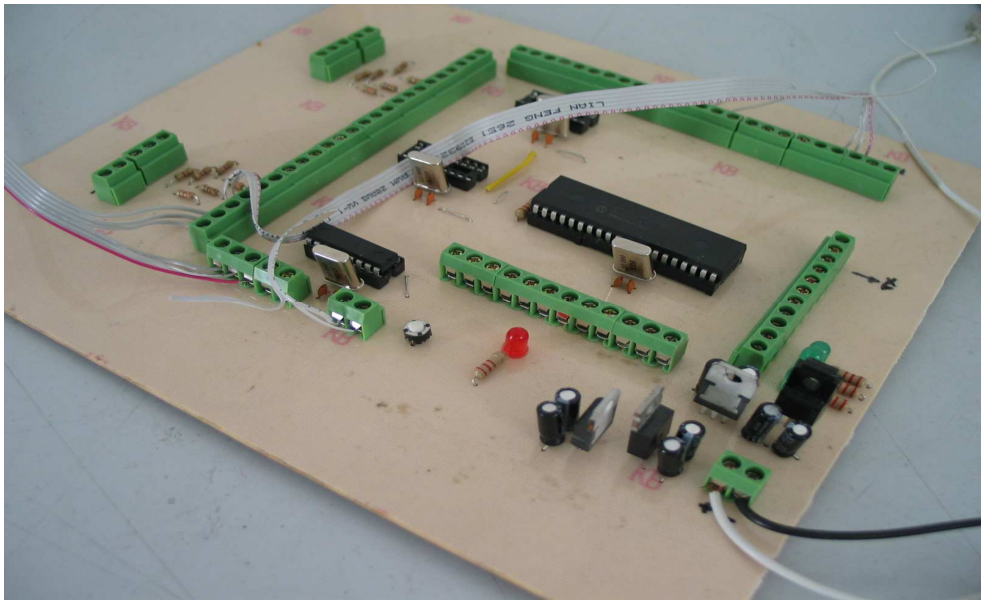


Figura 2.16 Circuito de microcontroladores para control de posición de un motor BLDC

Capítulo 3

Retroalimentación de fuerza al volante

Para resolver el problema de proveer al usuario con información útil sobre el estado del sistema de dirección, los sistemas Steer By Wire hacen uso de la retroalimentación de fuerza al volante. La retroalimentación de fuerza consiste en ejercer una contrafuerza en el volante en realciòn con la fuerza que ejercen los motores para orientar las ruedas. Para la implementación de la retroalimentación se tomaron en cuenta dos alternativas. La primera de ellas consistía en la utilización de un motor de corriente continua con escobillas que por medio de una banda opusiera cierta resistencia al volante, cuya magnitud sería proporcional al esfuerzo realizado por los motores del sistema de dirección en ese instante. La segunda alternativa surgió el momento en que se adquirió el volante de juegos Logitech para ser utilizado como el volante de la plataforma de investigación. El volante de juegos cuenta con retroalimentación de fuerza por lo que incluye un motor de corriente continua que se encarga de proveer dicha retroalimentación al conductor dependiendo de sus acciones. Ya que usar un motor de corriente continua, externo, como requeriría la primera alternativa, significaría un desperdicio de los recursos disponibles en el volante Logitech, se optó por implementar la segunda alternativa.

Debido a que el volante Logitech es un dispositivo de juegos para computadora, el control del mismo se realiza a través de aplicaciones y drivers de computador. Como se planeó incluir un computador en el sistema desde un inicio con el fin de registrar datos del comportamiento del sistema mediante el software LabView y una tarjeta de adquisición de datos, una nueva meta se fijó en controlar el volante a través de la misma aplicación desarrollada en LabView. Leer la posición del volante en Labview resulta relativamente

sencillo, ya que cuenta con elementos de su biblioteca construidos específicamente para comunicarse y leer datos de un joystick. Sin embargo, las cosas se dificultan considerablemente al intentar escribir información desde la computadora al volante, ya que Labview no trae instrumentos virtuales construidos para escribir información al puerto donde se conecta el joystick. Buscar información sobre los drivers de control por defecto del joystick, no dio resultados ya que Logitech no da información alguna acerca del código que contienen sus drivers. Ante esta dificultad, la única solución viable resultó construir por completo un driver que se comunicara con el volante, tanto leyendo como escribiendo información desde la computadora, y que fuese llamado por la aplicación de control de Labview.

Una vez establecido el objetivo de controlar la retroalimentación de fuerza disponible en el volante Logitech mediante el desarrollo de un driver, fue necesario investigar para encontrar una herramienta que permitiera hacerlo. El componente DirectInput de la Microsoft, utilizado para programar aplicaciones mediante la interfase Microsoft DirectX, resultó ser la mejor opción, por lo que el driver desarrollado como resultado de esta investigación está desarrollado completamente en dicha interfase con la ayuda de Visual C++.

Este capítulo, revisa primero los fundamentos necesarios para trabajar con DirectInput y la retroalimentación de fuerza, para posteriormente revisar la implementación práctica del driver para el control del volante Logitech desde LabView.

3.1 Desarrollo de Drivers con DirectInput

Para poder hacer uso de la retroalimentación de fuerza de un joystick, es fundamental conocer como funciona la interfase DirectInput de Microsoft. DirectInput es un componente que permite a una aplicación leer datos de un dispositivo de entrada como puede ser el mouse, el teclado o el joystick conectado a una computadora; y a la vez da soporte completo para dispositivos que cuentan con retroalimentación de fuerza entre sus características. Existen cuatro términos básicos que son necesarios comprender en el momento de desarrollar un controlador con DirectInput (Microsoft Help: DirectInput). Estos son:

Objecto DirectInput

Es la interfase raíz de DirectInput.

Dispositivo

Un teclado, mouse, joystick, u otro dispositivo de entrada.

Objeto DirectInputDevice

Código que representa un teclado, mouse, joystick, u otro dispositivo de entrada.

Device object

Código que representa una tecla, botón, disparador, etc. que puedan ser encontrados en un *Objeto DirectInputDevice*.

En el caso particular tratado en este proyecto, el volante con capacidad de retroalimentación de fuerza ha ser utilizado es visto como un joystick capaz tanto de recibir instrucciones desde el computador como de enviar información al mismo. El volante Logitech MOMO® Racing Force Feedback Wheel se conecta directamente al puerto USB del computador, y es reconocido como un dispositivo de interfase humana. Los drivers por defecto, se encargan de controlarlo cuando se utiliza el volante en los juegos de

computadora, sin embargo para que la aplicación de control del sistema By Wire tenga acceso al joystick y control sobre el mismo, es necesario escribir un propio driver que sea ejecutado al correr la aplicación de control. Los pasos básicos a seguir para tener acceso a un joystick son los siguientes (Microsoft Help: DirectInput).

1. Crear un objeto del tipo DirectInput para poder reconocer los dispositivos conectados al computador que soportan DirectInput.

2. Enumerar los dispositivos que soportan DirectInput. Este paso devuelve un identificador único para cada dispositivo encontrado.

3. Crear un objeto del tipo DirectInputDevice para el dispositivo que se quiere utilizar.

4. Configurar el dispositivo. Para cada dispositivo, primero se configura el nivel de cooperación, que determina si el dispositivo puede ser compartido con otras aplicaciones o con el sistema. También se configura el formato en el que se leen los datos de los objetos del dispositivo como son los ejes y botones de un joystick. Opcionalmente se pueden configurar los rangos de los valores para cada eje del joystick.

5. Adquirir el dispositivo, enviando la orden a la interfase de DirectInput de que la aplicación está lista para comenzar a leer datos del dispositivo.

6. Leer los datos del dispositivo, como puede ser el estado actual de sus botones y posición respecto a sus ejes.

7. Actuar a partir de los datos leídos. Una vez leído los datos se puede responder a la acción sobre un botón o eje. Si existe el soporte por parte del dispositivo, se puede escribir datos al dispositivo, como correspondería en un dispositivo que soporte retroalimentación de fuerza.

8. Cerrar la interfase DirectInput. Esto significa liberar todos los dispositivos y objetos de tipo DirectInput.

Estos son los mismos pasos que se seguirán al momento de implementar prácticamente el código para el driver que se debe desarrollar. Para la retroalimentación de fuerza nos enfocaremos en el paso número 7: la escritura de datos al dispositivo adquirido por medio de DirectInput. Para que una aplicación pueda escribir datos de retroalimentación de fuerza a un dispositivo, el mismo debe ser adquirido en modo exclusivo por la aplicación. Al adquirir de modo exclusivo el dispositivo, la aplicación tiene acceso a la escritura de datos al mismo, siempre que se encuentre activa sobre el resto de aplicaciones que estén corriendo en el sistema. Una vez que se adquiere de modo exclusivo el dispositivo, la aplicación puede comenzar a escribir órdenes de retroalimentación de fuerza según la disponibilidad de las mismas en la librería de DirectInput.

Las órdenes de retroalimentación de fuerza, son conocidas y clasificadas como efectos dentro de la librería de DirectInput. Existe un gran número de estos efectos disponibles en DirectInput, pero se los puede clasificar en dos grupos básicos: los efectos definidos por una forma de onda y los efectos condicionales. En los efectos definidos por una forma de onda, la magnitud de la fuerza varía de acuerdo a una forma de onda especificada a lo largo del tiempo, independiente de la posición del joystick. En los efectos condicionales la magnitud de la fuerza depende de la posición del joystick a lo largo de uno de sus ejes.

DirectInput mide la magnitud de la fuerza que se envía al joystick en unidades que se ubican en un rango lineal que va de 0 a 10000, donde 0 significa que la fuerza es nula, y 10000 corresponde a la máxima fuerza disponible en el dispositivo. En C++, la magnitud

máxima para la fuerza se define como DI_FFNOMINALMAX. Los valores negativos simplemente representan un cambio en la dirección de la fuerza. Todos los efectos de fuerza disponibles tienen su duración medida en microsegundos.

Cuando se trabaja con efectos de fuerza, también es necesario seguir un procedimiento para que el efecto sea reproducido en el joystick. Antes que nada, para que el usuario pueda sentir los efectos de retroalimentación de fuerza, es necesario desactivar cualquier tipo de autocentrado en el joystick. Después se debe crear una estructura del tipo de fuerza que utiliza el efecto que se quiere reproducir. En dicha estructura se definen las propiedades como magnitud y dirección de la fuerza. Con esta estructura definida, se crea una nueva estructura de tipo efecto, en la que se definen características como la duración, el tipo de coordenadas, ejes, dirección, ganancia aplicada a la fuerza, y la referencia a la estructura de fuerza creada previamente.

Definido completamente el efecto que se quiere reproducir, se inicia con el proceso de su creación, transmisión al joystick y reproducción. En la creación del efecto se crea una instancia del efecto. Esta instancia recibe la configuración especificada en la estructura de tipo efecto y transmite estas especificaciones al joystick. Por último, una vez que la información del efecto se encuentra en el joystick, el mismo solo espera por la orden de inicio para comenzar con la reproducción del efecto. Si durante la ejecución de la aplicación se desea cambiar parámetros del efecto, no es necesario crear todo el efecto nuevamente. Sólo es necesario reconfigurar el parámetro que se desea alterar, como puede ser la magnitud de la fuerza, y transmitir el cambio al joystick para que lo reproduzca.

A continuación se presenta un ejemplo concreto del código y procedimiento a seguir para generar un efecto de fuerza en el volante de juegos Logitech.

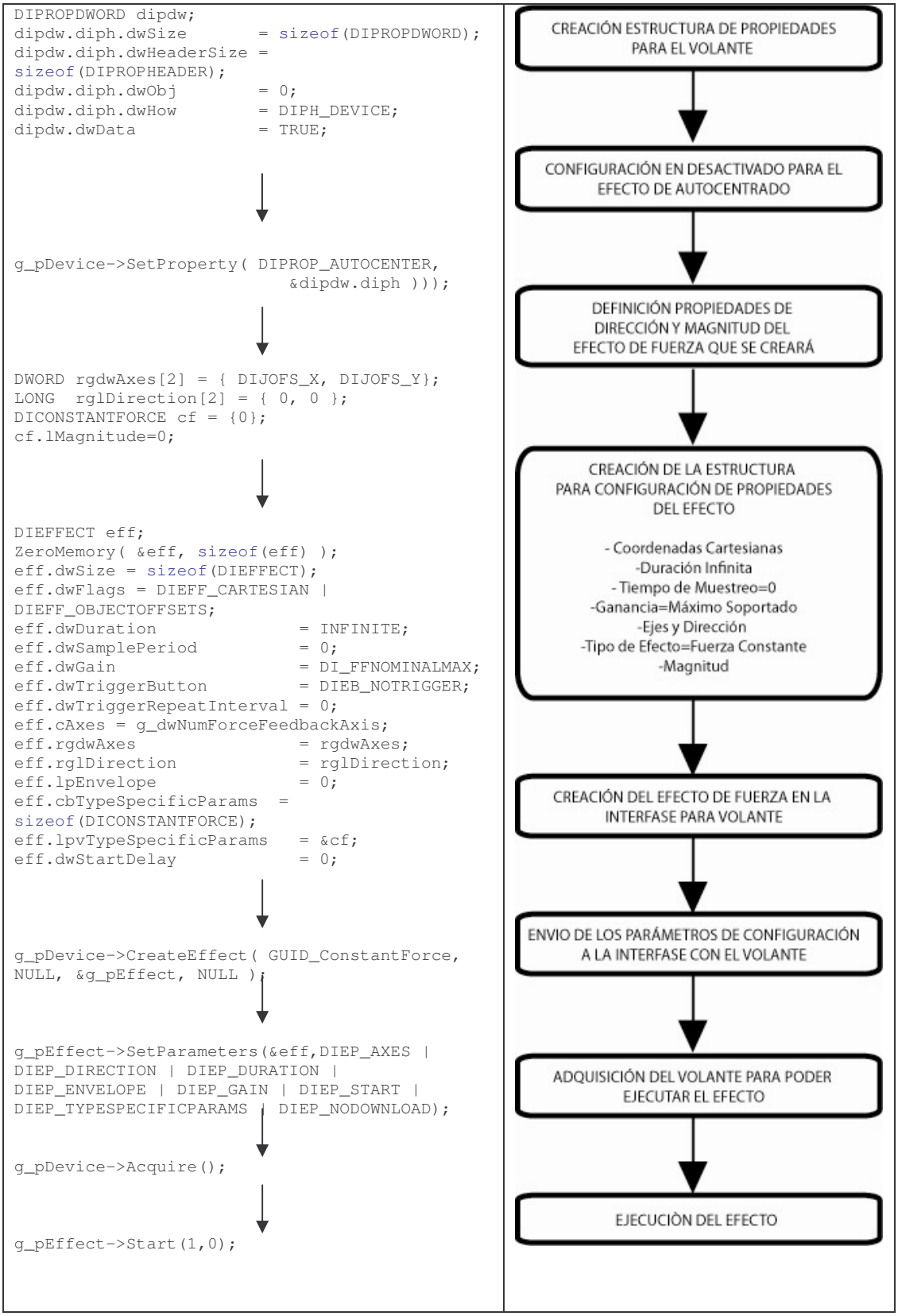


Figura 3.1 Código y diagrama de la secuencia seguida para genera un efecto de fuerza en el volante

3.2 Intregración del driver en Labview

Todos los fundamentos mencionados hasta el momento, permitirían construir un driver que funcione correctamente por sí solo. Sin embargo, para confirmar que el mismo funciona, es necesario que sea incluido en una aplicación que lo ejecute. Ya que la aplicación para el sistema propuesto en este proyecto será desarrollada en LabView, se revisarán los fundamentos de la inclusión de un driver dentro de una aplicación desarrollada en LabView.

LabView es un software de programación gráfica. Debido a la facilidad de lectura del programa visualmente e interacción con el mismo, la programación gráfica permite desarrollar prototipos de programas eficientes y en menor tiempo que el que tomaría escribir su código. Los programas en LabView se organizan con módulos conocidos como Instrumentos Virtuales (VIS). Estos programas se separan en un Panel Frontal donde se ubica la interfase con el usuario del programa, y un Diagrama de Bloques donde se estructura el programa bajo el que funciona la interfase con el usuario.

Lamentablemente, LabView no soporta directamente la interfase DirectInput. Sin embargo, contiene Instrumentos Virtuales para ejecutar código desarrollado en un lenguaje de programación externo (LabView8.2 Help/Input Device Control VIs). Para que la aplicación desarrollada en LabView pueda hacer uso del driver para tener acceso a la retroalimentación de fuerza del joystick, se hará uso de un Instrumento Virtual que llame una biblioteca de enlace dinámico (DLL por sus siglas en inglés), en la que se incluirá el código del driver desarrollado. Enlace dinámico es un mecanismo que enlaza la aplicación a una biblioteca únicamente el momento que se ejecuta la aplicación. Por lo tanto las librerías de enlace dinámico son archivos separados del archivo ejecutable de la aplicación, y que permiten establecer interfases entre una aplicación y otras librerías que no pueden ser

relacionadas directamente. En este caso, el DLL tiene el trabajo de enlazar la aplicación con las capacidades de las librerías de DirectInput para retroalimentación de fuerza.

Para implementar un DLL en LabView, es necesario usar la función Call Library, cuyo ícono se muestra en la figura 3.2. Lo más importante al momento de utilizar esta función, es que los parámetros de entrada a la misma coincidan con los que se espera entren y sean utilizados por el código escrito. La función Call Library puede ser configurada mediante una ventana de configuración, en la que se detallan el nombre del archivo del DLL, el nombre de la función dentro del código escrito a la que se hará referencia, y los tipos de variables que se utilizan como entrada y salida de la función.

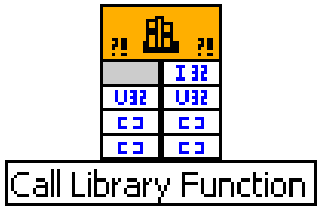


Figura 3.2 Función de LabView Call Library (LabView 8.2 Help/ Call Library Function)

Los fundamentos revisados en esta sección son básicos para conseguir implementar una aplicación en LabView capaz de manejar un joystick con retroalimentación de fuerza. En la siguiente sección, se detalla la implementación práctica del driver en Visual C++, y la aplicación en LabView, para controlar la retroalimentación de fuerza del volante Logitech MOMO® Racing Force Feedback Wheel utilizado en el sistema Steer By Wire propuesto.

3.3 Implementación práctica de retroalimentación de fuerza en el sistema Steer By Wire propuesto

3.3 a Generación de un DLL en Microsoft Visual Studio .NET 2003

Para la implementación práctica del driver y su inclusión en la aplicación de LabView, se comenzó por la escritura del código en Visual C++ que viene incluido en el paquete de Microsoft Visual Studio .NET 2003. Para poder hacer uso de las librerías de Microsoft DirectInput, es necesario bajar e instalar el Microsoft DirectX System Development Kit, que es un kit para el desarrollo de sistemas con soporte para Microsoft DirectX.

En Visual C++, se comienza por crear un nuevo proyecto Win32Console Project. En Application Settings, se debe cambiar el tipo de aplicación a DLL. El programa crea un proyecto con tres archivos básicos: driver.cpp, stadfx.cpp, como archivos de código fuente, y stdafx.h como archivo de cabecera. Para que el DLL pueda exportar funciones a una aplicación desarrolladas en otro lenguaje, es necesario agregar un archivo de código fuente, driver.def, donde se especifican las funciones exportadas. Además, es necesario agregar un archivo cabecera driver.h.

Una vez hecho esto, es necesario configurar las propiedades del DLL en Project -> Properties. En la pestaña Linker, se selecciona Input y en Additional Dependencies se deben agregar las librerías winmm.lib, dxerr.lib, dxguid.lib, dinput8.lib, comctl32.lib, d3d9.lib, d3dx9.lib para que el DLL funcione con la interfase DirectInput. Además se debe especificar driver.def para ModuleDefinitionFile. Con esta configuración, el DLL se construirá para ser incluido directamente en la aplicación de LabView (DirectX SDK Samples: FFConst Sample).

Una vez configurado el proyecto se puede iniciar con la escritura del código para el driver. Todo el código va dentro del archivo driver.cpp. Detalles como INCLUDES y DEFINES necesarios pueden revisarse en el apéndice donde se anexa el código completo del driver. El código se inicia con la declaración de las variables en las que se guardarán la interfase raíz a DirectInput creada por el driver, la interfase al joystick y la interfase a los efectos de fuerza. Además se declaran variables para guardar el número de ejes disponibles del joystick y el nombre que identifica al joystick.

Después de declarar las variables, se debe especificar los prototipos de las funciones que se utilizarán. Es fundamental que todas las funciones que se planea utilizar desde la aplicación de LabView se declaren de forma distinta a las funciones que solo se utilizan dentro del driver. El siguiente ejemplo muestra la declaración del prototipo de la función "terminar" la cual se utiliza para liberar el joystick y su interfase DirectInput, cuando se cierra la aplicación de Labview.

```
extern "C"_declspec (dllexport) VOID terminar (HWND hwnd);
```

Lo importante es especificar `extern "C"_declspec (dllexport)` antes del prototipo de la función que se desea exportar.

Declarados los prototipos de las funciones, se inició con el desarrollo del código para especificar el papel de cada función en el driver. Se inició siguiendo los pasos revisados en los fundamentos para trabajar con DirectInput. La primera función creada es la función InitDirectInput. Esta función crea una instancia de DirectInput para el driver. Después de crear la instancia, la función la inicializa y enumera los dispositivos conectados al computador con soporte para retroalimentación de fuerza. A continuación, se especifica el formato de datos con los que se manejarán los datos del joystick reconocido. Para el volante Logitech, el formato de los datos debe ser `&c_dfDIJoystick`. Una vez establecido el formato, se debe especificar el nivel cooperativo del driver. Como se mencionó

anteriormente, la retroalimentación de fuerza solo funcionará si el volante se adquiere de modo exclusivo por el driver y la aplicación que lo ejecuta. La instrucción `SetCooperativeLevel( hwnd,DISCL_EXCLUSIVE | DISCL_FOREGROUND)` , donde hwnd es el identificador de la ventana donde se corre la aplicación, adquiere el volante de modo exclusivo. El siguiente paso, es desactivar el autocentrado del volante y leer el número de botones y ejes de los que se dispone en el joystick. La última parte de esta función, crea un efecto de fuerza constante y lo envía al joystick. De esta forma, la aplicación no tiene acceso para crear nuevos efectos sino que solo hará uso de una función para cambiar el valor de los parámetros de la fuerza que se envía al volante. Así todos los efectos pueden ser generados en LabView, facilitando la investigación y el desarrollo de nuevos proyectos.

Después de crear la función para inicializar DirectInput y el joystick, se crearon tres funciones adicionales para ser exportadas a la aplicación desarrollada en LabView. La más básica de ellas, es la función `terminarDirectInput` que se encarga de liberar los dispositivos y cerrar todas las interfases de DirectInput. Esta función es muy importante ya que de no existir, la aplicación no tendría la posibilidad de cerrar correctamente todos los dispositivos lo que podría causar que la aplicación se congele poniendo en riesgo el control del sistema de dirección. La siguiente función a la que tiene acceso la aplicación, es la función `info`, la cual se encarga de leer el estado del joystick y enviar la información actualizada de cada botón y eje en arreglos para que sean leídos por el usuario. La tercera función que se exporta desde el driver, es la función `iniciarefectorawforce,` la cual es la función básica que será utilizada por la aplicación de control mientras esté ejecutándose. Esta función recibe el valor de fuerza que el usuario o el programador desea enviar al volante y transmite el cambio del efecto al joystick.

3.3 b Ejecución del DLL en la aplicación desarrollada en LabView

Hasta este punto, se habían seguido todos los pasos para la implementación del driver y se esperaba que funcione correctamente. Sin embargo, surgieron ciertas dificultades que retrasaron el trabajo con la retroalimentación de fuerza y requirieron de código adicional para solucionarlas. Al implementar una aplicación de prueba en LabView, el driver no tuvo problemas en leer el estado del joystick, pero si en enviar comandos de retroalimentación de fuerza. Haciendo un debug del programa, se encontró que el problema tenía que ver con la adquisición en modo exclusivo por parte de la aplicación.

Tras investigar el problema, se llegó a la conclusión de que la dificultad tenía que ver con la referencia a la ventana de la aplicación y como se estaban manejando los mensajes de la ventana dentro del driver. La referencia a la ventana de una aplicación es un identificador único, que el sistema operativo asigna a cada ventana que se abre. Esta referencia corresponde a la variable `hwnd` utilizada en el código, y que se utiliza en la instrucción `SetCooperativeLevel`, utilizada para establecer el nivel cooperativo del joystick en modo exclusivo. Una falla en el código, hizo que en lugar de un identificador a la ventana de la aplicación, se pase el identificador del DLL el momento de asignar el modo exclusivo. Entonces, el siguiente objetivo se fijó en encontrar una forma de obtener el identificador de la ventana de la aplicación en LabView y manejar sus mensajes, para poder acceder exitosamente al joystick en modo exclusivo.

Para obtener el identificador de la ventana de la aplicación desarrollada en LabView, se creó una nueva función, `crearventana`, en la cual se utiliza la instrucción FindWindow para encontrar la ventana de la aplicación y asignar su identificador a la variable hwnd. El momento que se tiene el identificador de la ventana, es necesario suplantar el proceso del sistema que maneja los mensajes que recibe la ventana con un proceso propio creado en el código del DLL. El proceso que se ocupa de las ventanas

abiertas en un sistema operativo, se encarga de manejar mensajes como los dados al pulsar un botón del mouse, presionar una tecla, cerrar, maximizar, minimizar la ventana, etc. El código del proceso se encarga de tomar la acción adecuada para cada caso. Al suplantar el proceso, con uno propio creado en el DLL, se tiene acceso para que el DLL maneje algunos mensajes. El proceso creado, WndProc, se encarga únicamente de manejar el caso en que la ventana de la aplicación se encuentra activa, es decir cuando la ventana se encuentra sobre cualquier otra aplicación y ha sido activada por un click izquierdo del mouse. El código del DLL, cuando se recibe el mensaje de que la ventana está activa, se encarga de que la aplicación tenga completo acceso al joystick para adquirirlo y enviar cualquier efecto de retroalimentación de fuerza. Para cualquier otro tipo de mensaje, el proceso llama por defecto al proceso del sistema operativo, para que se encargue de manejarlo. Mayor información acerca de cómo conseguir manejar los procesos de las aplicaciones para adquirir un dispositivo en modo exclusivo puede ser encontrada en la ayuda de Microsoft para las funciones `FindWindow`, `GetWindowLongPtr`, `SetWindowLongPtr y CallWindowProc` (Microsoft Help: The Window Procedure).

Con el problema de acceso exclusivo al joystick solucionado, se utilizó la pequeña aplicación de prueba en LabView para confirmar que todo funcionaba correctamente. La aplicación tiene una interfase para el usuario, en la que el mismo puede observar el estado de todos los botones del joystick, así como la posición a lo largo de sus ejes. Además, el usuario tiene acceso para cambiar el valor de la fuerza que se envía al volante. Cambiando la magnitud de la fuerza se puede lograr que el volante se mueva por sí solo, confirmando que el driver tiene acceso al joystick y que es capaz de alterar la magnitud de la fuerza en sus efectos. La siguiente figura muestra el panel frontal de la aplicación de prueba desarrollada.

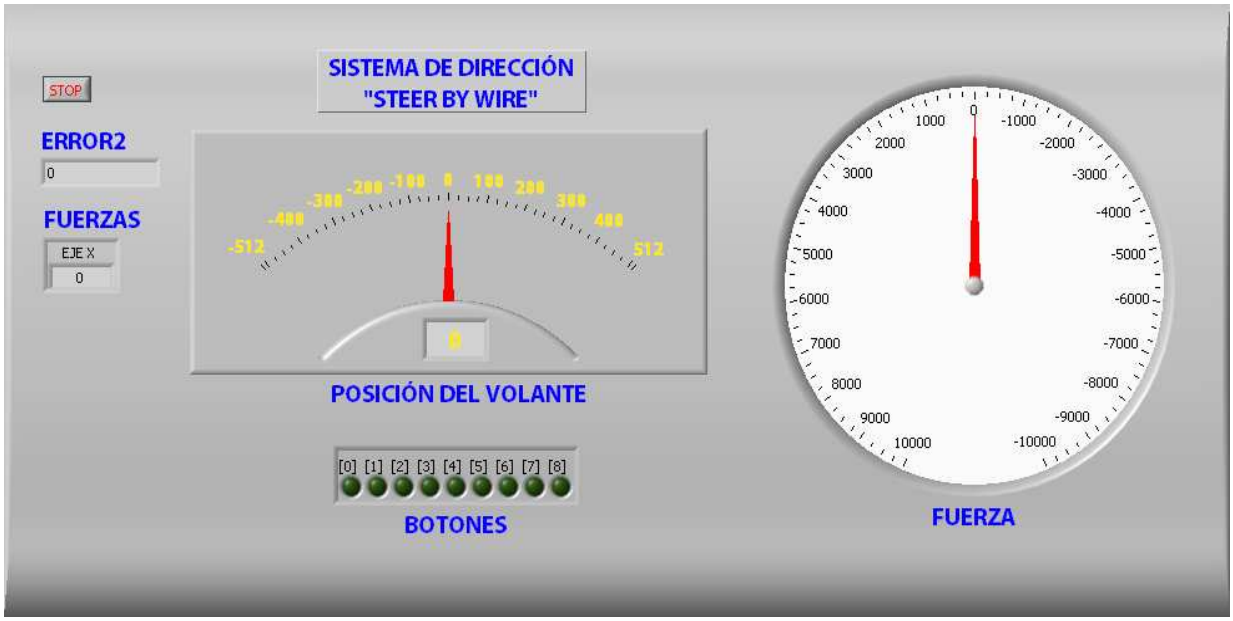


Figura 3.3 Panel Frontal de la aplicación de prueba para retroalimentación de fuerza

Con esta aplicación se puso fin a las dos etapas iniciales del proyecto que se enfocaban en controlar la posición de los motores de corriente continua sin escobillas, por un lado, y en tener acceso a la retroalimentación de fuerza al volante utilizado para la plataforma, por el otro. La siguiente etapa, consistió en fusionar los logros alcanzados en las dos etapas mencionadas, elaborando una aplicación de control que se encargue tanto de posicionar los motores de acuerdo a la posición del volante, como de retroalimentar fuerza al volante en base al estado de los motores.

Capítulo 4

Desarrollo de la aplicación de control

El desarrollo de la aplicación de control es clave para el funcionamiento de la plataforma como un sistema de dirección By Wire. La aplicación de control, es la encargada de sincronizar el movimiento del volante con el de los motores, y de hacer que la retroalimentación de fuerza en el volante corresponda con la situación de los motores y la dirección de las ruedas. La aplicación de control se subdivide en varias etapas que incluyen el programa de procesamiento y adquisición de datos en el computador portátil, así como cada uno de los programas en los microprocesadores que intervienen en el sistema de control. Este capítulo revisa la estructura de cada uno de los programas mencionados, pero no profundiza en detalles del código escrito, ya que los mismos pueden ser encontrados en los anexos C y D de este documento.

La figura 4.1, muestra un esquema de la estructura y flujo de señales tomado como referencia para el desarrollo de las aplicaciones de control.

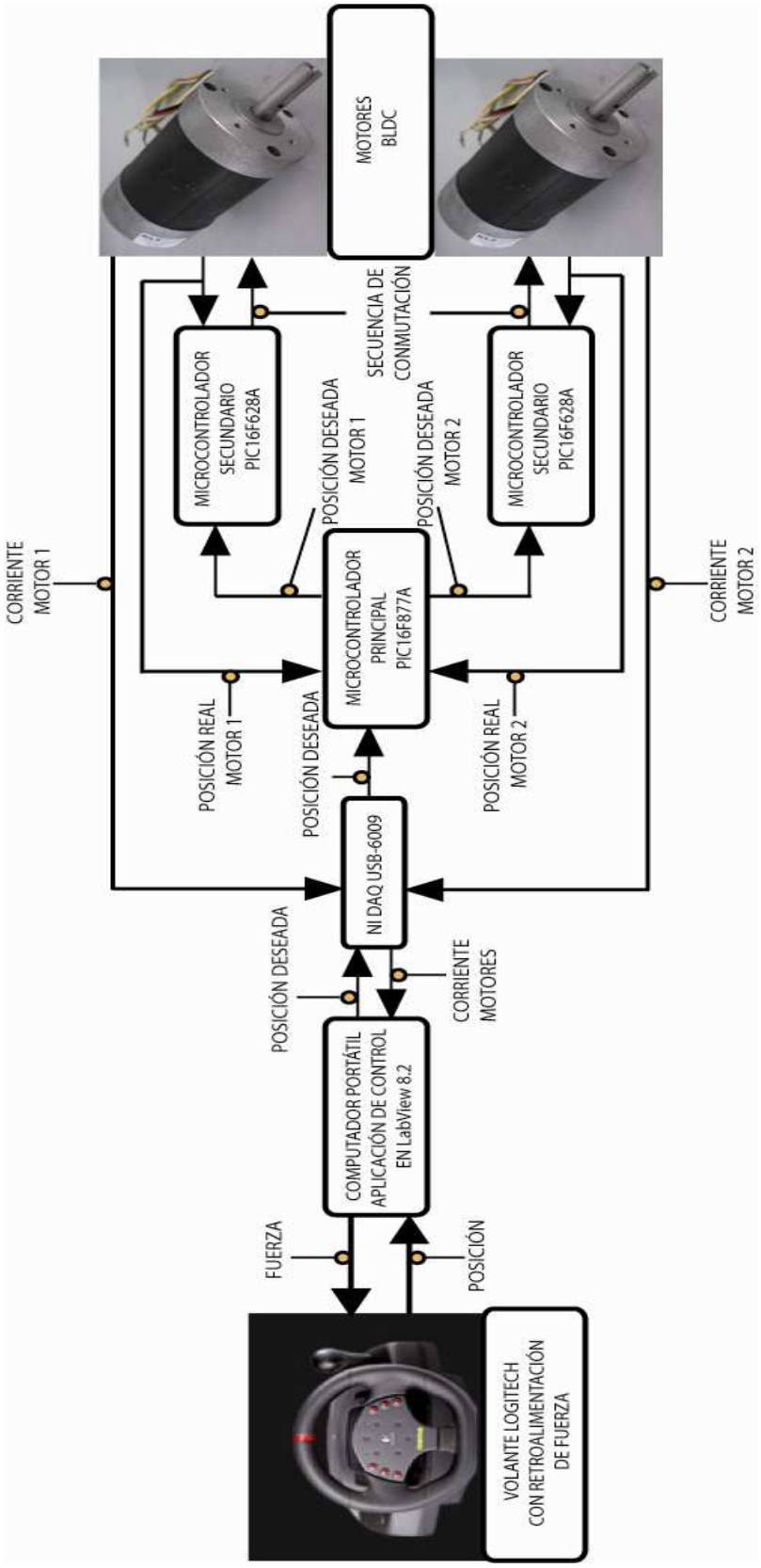


Figura 4.1 Estructura de comunicación entre componentes del sistema Steer By Wire

4.1 Estructura de la aplicación de control para el computador portátil

El desarrollo de la aplicación de control para el computador portátil, tomó como base la aplicación de prueba mencionada en el capítulo 3 y que se utilizó para probar el driver de control para la retroalimentación de fuerza al volante del sistema. Los datos claves con los que trabaja el computador portátil, son la posición del volante y la corriente consumida por los motores en las ruedas como entradas, y, la posición requerida para los motores y la fuerza de retroalimentación para el volante, como salidas. La aplicación construida en LabView, lee la posición del volante y la información de los sensores de corriente, y envia una referencia análoga para posicionar los motores, además de retroalimentar fuerza al volante a través del driver desarrollado. Este driver, se ejecuta continuamente mientras la aplicación esté en funcionamiento. Para compartir la información con el resto del sistema, la aplicación se comunica con una tarjeta de adquisición de datos, la DAQ NI USB-6009 de National Instruments.

La aplicación consiste de una etapa de inicialización en la que se llama al driver desarrollado para el volante, el cual inicializa la interfase DirectInput e intercepta el proceso de control de la ventana en la que se ejecuta el programa para tener acceso exclusivo a la retroalimentación de fuerza. Además, en esta etapa se coloca el volante en la posición central y se desactiva el efecto de autocentrado para poder sentir la retroalimentación de fuerza. Resulta importante conocer que el volante no está reservado para ser utilizado únicamente en la plataforma de investigación, sino que una vez que se libera su interfase de DirectInput, el mismo puede utilizarse para otras investigaciones o para que el usuario lo utilice en juegos de computador, sin que esto afecte la ejecución normal de la aplicación de control en el computador portátil del sistema.

En esta etapa también se utiliza una salida digital de la tarjeta de adquisición de datos, la cual se configura como un uno digital para comunicar a los microcontroladores

que la tarjeta se encuentra lista y que el sistema se ha puesto en marcha. Esto permite que los microcontroladores identifiquen cuando comenzar a leer datos análogos, como son las señales de 0 a 5V proporcionales a la posición del volante, en su conversor análogo digital. Esto evita que los motores se muevan durante la etapa de inicialización e introduzcan errores en el control de posición.

Una vez realizadas todas las acciones de inicialización, el programa entra en un lazo while el cual solo es detenido cuando el usuario pulsa el botón STOP en el panel frontal. Este lazo consiste básicamente de dos módulos, uno que se encarga del procesamiento de todo lo que corresponde al control de posición, y otro que procesa todo lo relacionado a la retroalimentación de fuerza. Ya que este programa no se encarga directamente de controlar la posición de los motores, el módulo de control de posición únicamente se encarga de leer la posición de volante y generar una señal análoga de voltaje entre 0 y 5V proporcional a dicha posición, para ser transmitida a los microcontroladores a través de la DAQ. El codificador del giro del volante Logitech, provee una señal entre -512 y 511 para especificar la posición del volante. Una lectura de -512, significa que el volante se encuentra en el tope de su giro a la izquierda, 511 representa el tope de giro a la derecha, y 0 corresponde a la posición central. A partir de esto, la aplicación suma 512 a la posición del volante, y luego la divide para 204.8, transformándola proporcionalmente a la señal de 0 a 5V necesaria para que el conversor análogo digital del microcontrolador la pueda procesar. Está señal está disponible en una salida análoga de la DAQ USB-6009.

El módulo de control de fuerza se encarga de actualizar continuamente la fuerza que se retroalimenta al volante por medio del driver de control del mismo. El sistema cuenta con un sistema de retroalimentación de fuerza que es proporcional a la posición del volante. Es decir, mientras más alejado se encuentra el volante de su posición central mayor es la fuerza que se opone a dicho movimiento. De esta forma se consigue el efecto

de autocentrado que se experimenta en los sistemas tradicionales de dirección mecánica cuando el vehículo se encuentra en movimiento. Además, el sistema de retroalimentación de fuerza hace uso de la tarjeta de adquisición de datos y dos de sus entradas análogas para leer señales de voltaje proporcionales a la corriente consumida por cada uno de los dos motores utilizados en el sistema. Esta es una característica importante para el sistema, ya que lo provee con la posibilidad de retroalimentar información sobre obstáculos en la trayectoria de posicionamiento de las llantas del vehículo. Si, por ejemplo, la llanta se encuentra con una vereda que obstaculiza su movimiento, el esfuerzo realizado por el motor en el intento de posicionarse de acuerdo a la orden recibida desde el volante, provocará un incremento en la corriente consumida por el mismo. Una resistencia de potencia se encarga de tomar muestras de la corriente consumida en el motor y traducirlas a medidas de voltaje. La señal de voltaje es leída únicamente por la tarjeta de adquisición de datos, y el programa en el computador portátil se encarga de tomar las acciones necesarias para poner al usuario sobre aviso del aumento de corriente.

Por último, en caso de que el usuario pulse el botón STOP del panel frontal, la aplicación se encarga de detener la tarjeta de adquisición para que la salida digital que indica su estado envíe un cero digital indicando que no se debe leer más las señales análogas y por lo tanto los motores deben ser detenidos. Con el fin de su ejecución, la aplicación libera la interfase de DirectInput, disponibilizando el volante para su uso en otras aplicaciones.

A continuación, la figura 4.2, presenta un diagrama en el que se especifica el flujo del programa de control en el computador portátil.

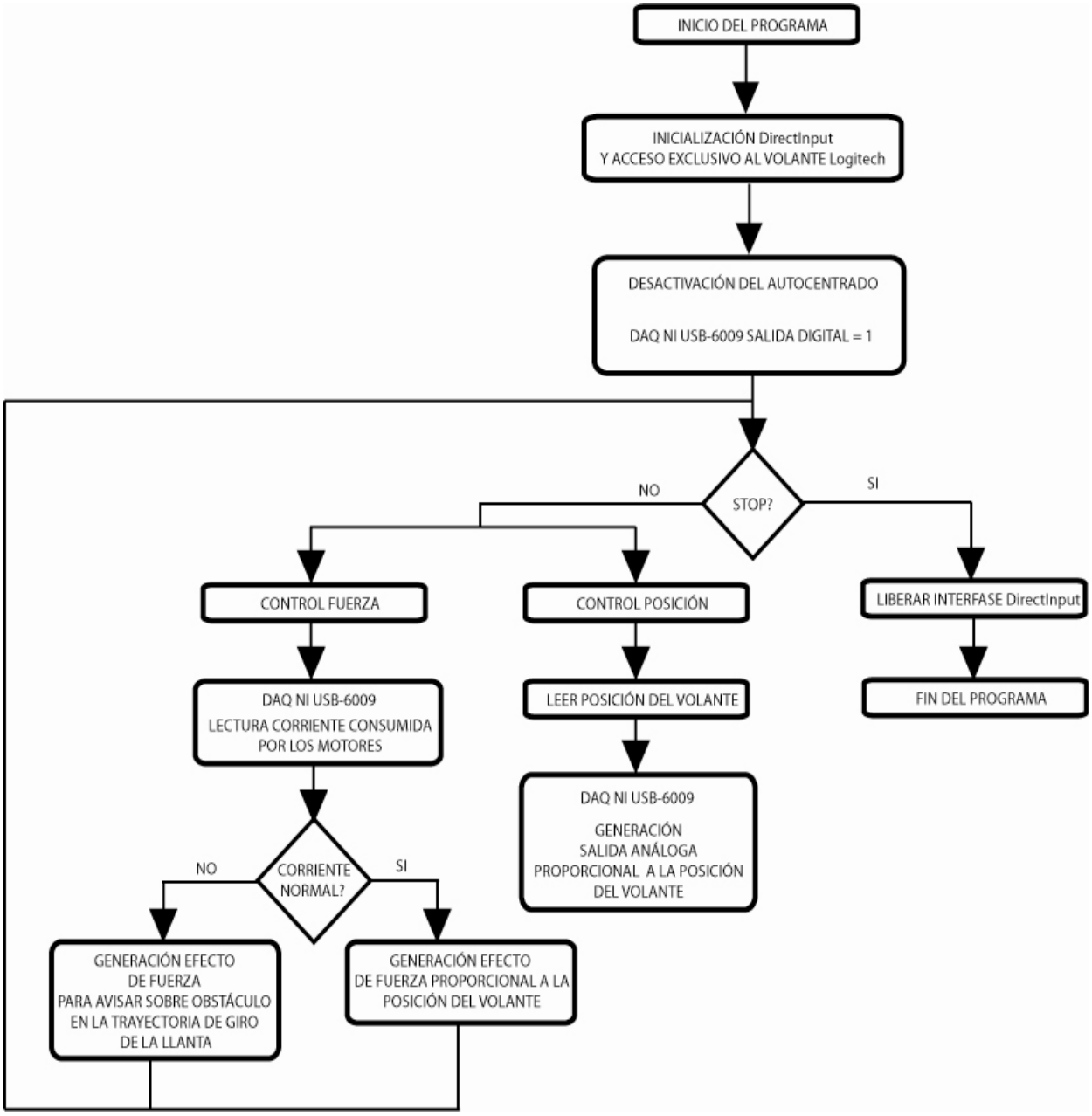


Figura 4.2 Diagrama de bloques programa de control computador portátil

La aplicación consta con un panel frontal, en el que el usuario dispone de diferentes herramientas para verificar y controlar el funcionamiento del sistema. Además, en el panel frontal se cuentan con todas las opciones necesarias para realizar las pruebas que demuestran las capacidades del sistema, y que serán detalladas en el capítulo 5 de este documento. Cada vez que se inicializa la aplicación, una ventana indicará si el driver pudo

inicializar la interfase DirectInput con éxito. Después de aceptar el mensaje, el usuario debe dar un clic izquierdo en la ventana de la aplicación para activarla y acceder exclusivamente al sistema.

4.2 Estructura del programa para el microcontrolador principal

El microcontrolador principal, un PIC16F877A es la interfase entre el computador portátil y los microcontroladores que controlan los motores que direccionan las llantas del sistema By Wire. Debido a esta característica, el microcontrolador tiene acceso tanto a datos de posición del volante como a los datos de posición de los motores. La posición del volante es recibida en forma de la señal análoga de voltaje entre 0 y 5V generada por la aplicación de control que se ejecuta en el computador portátil. La posición de los motores es decodificada a partir de las señales de los sensores de efecto Hall de los dos motores de corriente continua sin escobillas.

La primera parte del programa se encarga de leer el canal de entrada análogo en el que se recibe la señal enviada por la tarjeta de adquisición de datos de National Instruments. Este canal es leído continuamente durante la operación del sistema, con el fin de detectar cualquier cambio en la posición del volante. El conversor análogo digital del microcontrolador convierte la señal análoga a una señal digital con una resolución de 8 bits. Esto significa que tiene 256 niveles de resolución. Como el recorrido del volante de tope a tope tiene 1024 niveles, el microcontrolador principal solo detecta cambios de posición del volante cada 4 niveles. En el microcontrolador una señal de 0 equivale al tope de giro a la izquierda, 255 equivale al tope de giro a la derecha, mientras que 127 representa la posición central del volante.

Una vez interpretados estos datos, se calcula el número de vueltas que tiene que dar el motor para corresponder a la posición del volante. En este instante, entra en juego una de las ventajas más importante de los sistemas de dirección By-Wire: el hecho de poder cambiar el número de vueltas que dan los motores por vuelta del volante. Esta característica será explicada en detalle como parte de las pruebas realizadas en el capítulo 5, sin embargo es importante conocer que el programa en el computador portátil tiene cuatro relaciones de giro predeterminadas de las que puede escoger el usuario. Dependiendo de cada selección, el microcontrolador recibe un código binario de dos bits que se traduce en un cambio en el factor por el que se multiplica la señal digital proporcional a la posición del volante, en el cálculo del número de vueltas que deben dar los motores. Es más, el programa podría adaptar dicha relación a diferentes condiciones de manejo, algo imposible en un sistema de dirección mecánica, donde cambiar la relación entre las vueltas del volante y cuanto se mueven las llantas, requeriría un cambio de engranajes.

Las relaciones de giro predeterminadas así como su código correspondiente se indican en la tabla 4.1.

CÓDIGO (2 bits)	**RELACIÓN DE GIRO (movimiento detectado del volante: vueltas motores)**
00	1:20
01	1:25
10	1:30
11	1:35

Tabla 4.1 Código para cambio de relación de giro en el microcontrolador principal

El programa, almacena el número de vueltas en una variable de tipo BYTE (8 bits), con lo que el máximo de vueltas que puede dar un motor es de 255 vueltas para cada lado. El programa siempre necesita almacenar la posición actual del motor en vueltas. La posición almacenada es comparada con la nueva orden que se recibe cuando el volante es movido, y así se calcula la dirección y el número de vueltas que tiene que dar el motor para ubicarse correctamente.

Calculado el número de vueltas, el microcontrolador principal tiene que encargarse de controlar los microcontroladores secundarios. Una vez que los enciende, el microcontrolador principal utiliza las señales de los sensores de efecto Hall para contar el número de vueltas dadas por los motores y apagarlos el momento preciso en que estos alcancen la posición deseada. Ya que el microcontrolador principal necesita indicar la dirección hacia la que tienen que moverse los motores, se utilizan dos pines del microcontrolador por motor para enviar la orden de encendido y apagado de los motores junto con la dirección. Así, la orden consta de dos bits, dando lugar a cuatro posibilidades.

La tabla 4.2 se utiliza como referencia para las órdenes enviadas a los motores. Al inicio del programa se envía la orden 00 ó 01 para mantener apagados los motores. Una vez que se mueve el volante y se calcula el número de vueltas, el microcontrolador envía la orden 10 ó 11, dependiendo de la dirección de movimiento, y se basa en las señales de los sensores de efecto hall para saber cuantas vueltas da el motor. El momento que se alcanza el número de vueltas calculado, se envía nuevamente la orden 00 ó 01 para detener el motor en la posición deseada hasta recibir una nueva orden.

ORDEN (2 bits)	MOTOR
00	APAGADO
01	APAGADO
10	ENCENDIDO – HACIA LA IZQUIERDA
11	ENCENDIDO – HACIA LA DERECHA

Tabla 4.2 Interpretación de órdenes enviadas por microcontrolador principal

El código completo, resultado de las características y necesidades de control descritas en esta sección se encuentra en el Anexo D de este documento

4.3 Estructura del programa para los microcontroladores secundarios

4.3 a Perfiles de velocidad

Los microcontroladores secundarios seleccionados para el proyecto, son dos PIC 16F628A. Estos microcontroladores son fabricados por la empresa Microchip, pero únicamente tienen 18 pines, a diferencia de los 40 pines con los que consta el microcontrolador principal PIC16F877A. Se seleccionaron estos microcontroladores, porque su función dentro del control del sistema es bastante sencilla. Los microcontroladores secundarios se encargan de ejecutar la secuencia de conmutación para que el motor gire a la izquierda o a la derecha dependiendo de la orden recibida desde el microcontrolador principal. Como se mencionó, con la orden 00 ó 01, los microcontroladores paran los motores en una posición dada; y las órdenes 10 y 11 se utilizan para girar el motor a la derecha o izquierda.

En el desarrollo del código para los microcontroladores secundarios fue importante considerar que el momento de poner en marcha un motor, la única resistencia al flujo de

corriente está dada por la impedancia de los bobinados. Ya que, por diseño, dicha impedancia suele ser bastante pequeña, la corriente de arranque puede ser bastante grande, llegando a superar por mucho la corriente para la que fue diseñado el circuito de potencia. Esto significa que si no se limita la corriente de arranque los transistores pueden quemarse y echar a perder el circuito. La solución más económica para limitar la corriente que fluye en el arranque del motor, es incrementar la velocidad del motor progresivamente hasta alcanzar la velocidad de operación deseada. De igual forma el momento de detener el motor es importante disminuir la velocidad de forma progresiva. Para conseguir variaciones progresivas de velocidad es necesario programar etapas de aceleración y desaceleración para los motores.

Con el control de aceleraciones y desaceleraciones propuesto se consigue que los motores tenga un perfil de movimiento punto a punto. Esto significa que el motor de estar inicialmente parado, es acelerado a una velocidad constante y luego desacelerado de tal forma que al alcanzar la posición de destino, tanto su aceleración como velocidad son cero. Para cumplir con este tipo de perfiles de movimiento, es necesario programar perfiles de velocidad para el motor. El perfil de velocidad más simple y comúnmente utilizado es el perfil de velocidad trapezoidal.

Un perfil de velocidad trapezoidal consta de tres fases: una etapa inicial de aceleración constante en la que la velocidad se incrementa linealmente, una etapa intermedia sin aceleración en la que la velocidad se mantiene constante al nivel deseado, y una etapa final de desaceleración constante en la que la velocidad disminuye linealmente hasta cero, momento en el que el motor se detiene. Al programar este tipo de perfiles para un motor BLDC, es importante conocer el número de pasos que el motor da en cada una de las etapas del perfil de velocidad, para evitar la pérdida de precisión en su posicionamiento. Sabiendo que el espacio es igual a la integral de la velocidad, el espacio recorrido por el

motor durante cada etapa de su perfil de velocidad puede ser conocido simplemente al calcular el área bajo dicho perfil durante la etapa deseada.

La figura 4.3, muestra gráficas de un perfil de velocidad trapezoidal junto a la aceleración correspondiente en cada una de sus etapas. Refiriéndose a la gráfica de velocidad, se puede observar que el cálculo del área bajo el perfil es bastante sencillo al tratarse de la combinación de las áreas de dos triángulos y un rectángulo. Calculando el área del primer y del segundo triángulo es posible conocer el número de pasos que el motor toma para acelerar y desacelerar respectivamente. Para la aplicación tratada en este documento, se redujo el área durante las etapas de aceleración y desaceleración de tal forma que se mantuvieron las características de limitación de corriente, pero sin afectar la precisión en el posicionamiento de los motores. Al minimizar las áreas bajo las etapas de aceleración y desaceleración se asegura que las pérdidas de precisión en el posicionamiento de los motores no sean mayores a dos vueltas.

El control sobre la velocidad del motor se consigue mediante la modulación de ancho de pulso, ó PWM, aplicada en el control de encendido de los MOSFETS presentes en los medios puentes H del circuito de potencia. La frecuencia del PWM generado debe ser mayor a la frecuencia dada por la constante del tiempo del motor. La constante de tiempo del motor está dada por la división de la inductancia para la resistencia de sus bobinados. Estos datos, normalmente, se especifican en la hoja técnica del motor. Para los motores utilizados en este proyecto, utilizando como referencia el Anexo A de este documento, tenemos una inductancia L = 2.7mH y una resistencia R=3.8Ω. A partir de estos datos la constante de tiempo sería de L/R= 0.710ms. A partir de esta constante de tiempo se puede concluir que la resistencia e inductancia del motor actúan como un filtro con una frecuencia de corte de 1/0.710ms = 1407 Hz. La frecuencia de la PWM generada debe ser por lo menos 10 veces mayor a la frecuencia de corte. Sin embargo, frecuencias

menores a los 18Khz están dentro de nuestro espectro audible y pueden ocasionar ruidos molestos. Debido a esto la frecuencia final con la que se trabajó fue de 20Khz.

Conseguir una frecuencia alta para el PWM es muy importante, y el lenguaje de programación utilizado es determinante debido a su influencia en el tiempo de ejecución tomado por cada instrucción del programa. Tras algunas pruebas se determinó que la mejor opción era utilizar el lenguaje Ensamblador para programar los microcontroladores secundarios. Este es un lenguaje considerado de bajo nivel y que toma aproximadamente dos microsegundos (2us) por instrucción. Con este tiempo de ejecución, fue relativamente simple alcanzar los 20Khz para la generación del PWM.

El código final que se ejecuta en los microcontroladores secundarios hace uso de las interrupciones generadas por uno de sus temporizadores. Para acelerar, el programa incrementa la referencia del PWM cada cierto número de interrupciones. Mientras mayor es la referencia mayor es la velocidad alcanzada por el motor. Así se consigue una variación progresiva desde cero hasta la velocidad máxima del motor. Las interrupciones también son utilizadas para desacelerar el motor. Cada cierto número de interrupciones la referencia de PWM disminuye su valor, logrando que el motor se detenga progresivamente. Para alcanzar la frecuencia de operación de 20Khz, el microcontrolador genera una interrupción cada 50us.

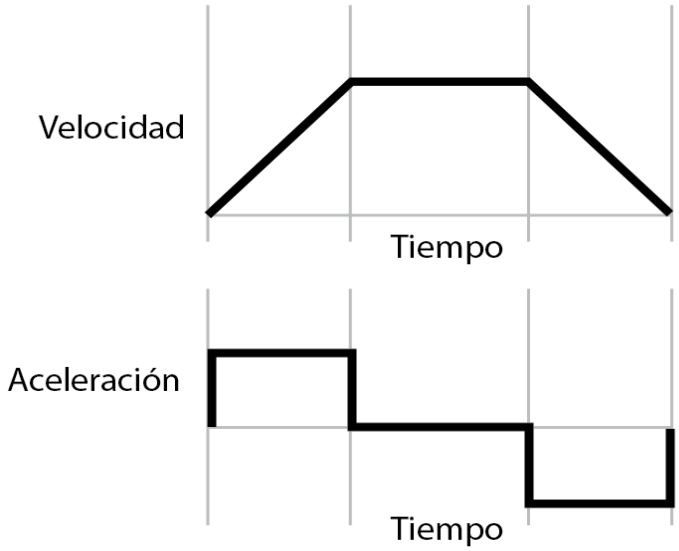


Figura 4.3 Perfil de velocidad trapezoidal y aceleración correspondiente

4.3 b Implementación de la secuencia de conmutación

Un PIC16F628, tiene dos puertos: PUERTOA y PUERTOB. En la aplicación para este sistema, el PUERTOA se encarga de recibir las señales de los sensores de efecto HALL, y las órdenes provenientes del microcontrolador principal. El PUERTOB se encarga de enviar las señales de conmutación al circuito de potencia para controlar el estado de los transistores y, por lo tanto, de alimentación de las fases del motor. Si bien son necesarias únicamente 6 señales para el circuito, el programa utiliza órdenes de 8 bits, para facilitar el trabajo por medio de registros.

Los microcontroladores secundarios se encargan ejecutar el código que genera la secuencia de control descrita y simulada en la sección 2.5 del capítulo 2 de este documento. La figura 4.4, muestra un circuito en el que se identifica la posición de cada transistor Q con un número. A partir de esta referencia se generó la tabla 4.1, en la que se puede observar el estado de cada pin en el PUERTOB del microcontrolador secundario

para cada paso de la secuencia de conmutación. Seis pines del PUERTOB, controlan el encendido y apagado de los transistores del circuito de potencia de los motores. La tabla, cuenta también con el estado de los sensores de efecto Hall, disponibles en los pines 3, 4 y 5 del PUERTOA del microcontrolador, correspondientes para cada uno de los estados del PUERTOB.

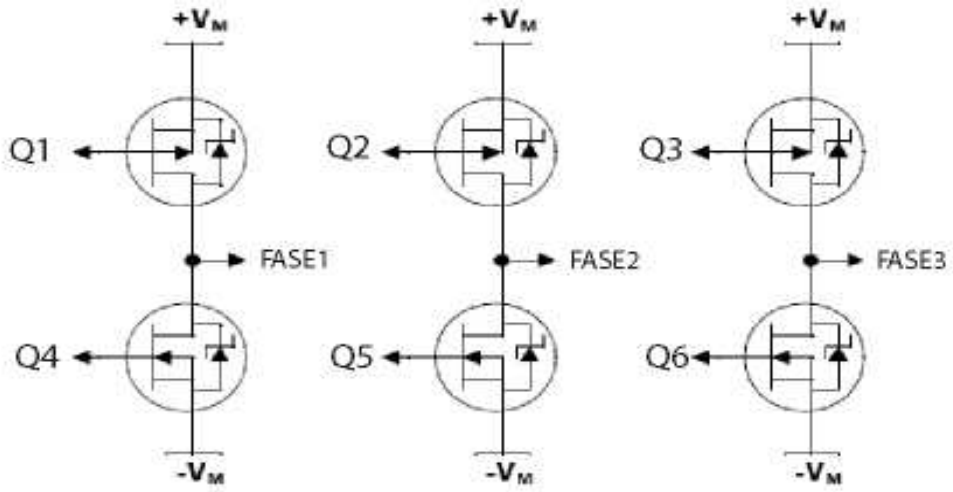


Figura 4.4 Circuito de referencia para programación de microcontrolador secundario

Por último, la tabla 4.3 es básica para la programación correcta del microcontrolador y para que la alimentación de las fases tenga coherencia con el estado de los sensores de efecto Hall, de forma que el sistema no desperdicie el torque de los motores. Para invertir el sentido de giro, sólo es necesario invertir el orden de la secuencia. Es claro, que los cambios de dirección de giro dependerán únicamente de las órdenes recibidas desde el microcontrolador principal.

B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0	A.3	A.4	A.5
X	Q4	Q1	Q5	Q2	Q6	Q3	X	Hall 3	Hall 2	Hall 1
X	1	0	0	0	0	1	X	0	0	1
X	0	1	1	0	0	0	X	0	1	0
X	0	0	1	0	0	1	X	0	1	1
X	1	0	0	1	1	0	X	1	0	0
X	0	0	0	1	0	0	X	1	0	1
X	0	1	0	0	1	0	X	1	1	0

Tabla 4.3 Secuencia de conmutación y relación con sensores de efecto Hall

El código desarrollado para los microcontroladores secundarios es el último nivel en el camino que recorren las señales de control antes de llegar a los motores BLDC, encargados del direccionamiento de las ruedas del vehículo. Implementado este código, todo el sistema se encuentra unificado y controlado por las aplicaciones que se ejecutan tanto en el computador portátil como en cada uno de los microcontroladores. En este punto, la etapa de desarrollo de la plataforma llega a su final, y es el momento de ponerla a prueba. Las pruebas realizadas y los resultados obtenidos para las mismas se detallan en el siguiente capítulo.

Capítulo 5

Pruebas y Resultados

Una vez diseñadas e implementadas las etapas principales en el desarrollo de la plataforma de investigación, es necesario evaluar el desempeño de la misma como un sistema Steer By Wire. Este capítulo describe varias pruebas realizadas en el sistema desarrollado, con el fin de determinar tanto las capacidades como limitaciones de la plataforma de investigación. Adicionalmente, y en un marco más general, las pruebas a realizarse buscan establecer las características, ventajas, novedades y diferencias de los sistemas Steer By Wire ante los sistemas convencionales de dirección.

A lo largo del capítulo, cada prueba se describe individualmente junto a sus resultados. Todas las pruebas pueden ser controladas a partir del panel frontal de la aplicación que se ejecuta en el computador portátil. La figura 5.1, muestra una imagen de la plataforma de investigación terminada que se utilizó para realizar las pruebas detalladas en este capítulo.

Figura 5.1 Plataforma de investigación terminada

La figura 5.2, presenta una imagen del panel frontal de la aplicación de control en el que se encuentran todas las herramientas necesarias para realizar las pruebas descritas a continuación.

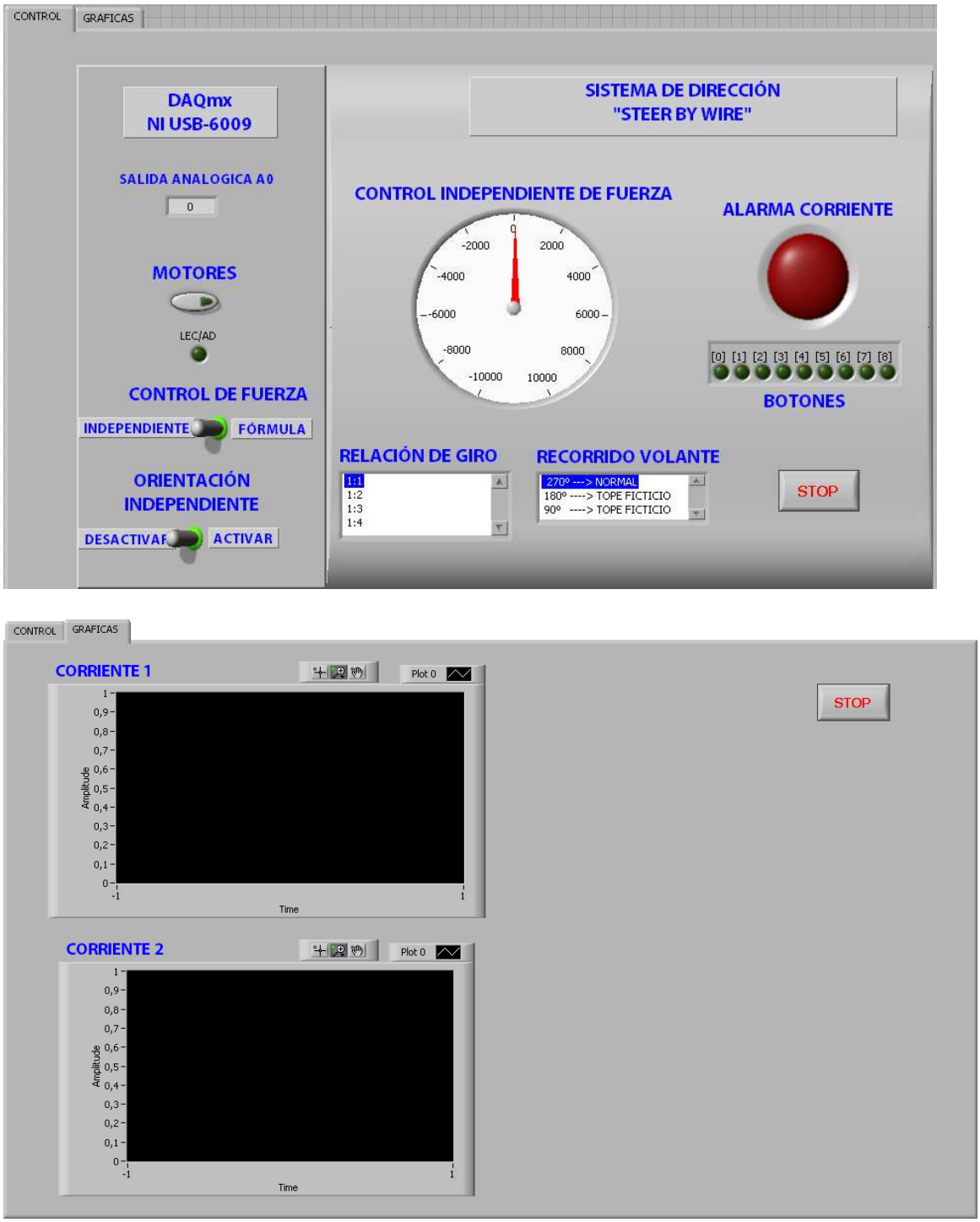


Figura 5.2 Panel frontal utilizado para realización de pruebas

5.1 Control de posición

Esta primera prueba, examina una de las características fundamentales con las que debe contar un sistema Steer By Wire: el posicionamiento de los motores eléctricos basado en la posición del volante.

El posicionamiento de los motores eléctricos, tomando como referencia la posición del volante, es considerado clave en un sistema Steer By Wire, porque es justamente esta característica la que permite que un sistema Steer By Wire elimine todas las piezas mecánicas que conectan el volante con las ruedas en un sistema de dirección convencional.

Para evaluar el posicionamiento de los motores en función de la posición del motor, la prueba se inicia enviando una posición de referencia. Para esto se mueve el volante y se lo deja en una posición fija. Este movimiento debe generar un voltaje análogo que se transmite al microcontrolador principal a través de la tarjeta de adquisición de datos. En el microcontrolador principal se realizan los cálculos necesarios para prender los microcontroladores secundarios hasta que los motores alcancen la posición correspondiente a la posición del volante.

Para medir el número de vueltas dadas por los motores, se utilizó un microcontrolador PIC16F628A adicional encargado únicamente de contar las revoluciones dadas por el motor. La precisión de la operación puede ser determinada a través de la comparación del número de vueltas dadas por el motor para una misma referencia en diferentes repeticiones de la prueba. Las señales de los sensores de efecto Hall de cada motor son retroalimentadas al microcontrolador, el cual ejecuta un programa que se encarga de contar el número de revoluciones del motor, y posteriormente exhibir las revoluciones totales en una pantalla LCD. Esto permite conocer las características del movimiento realizado por el motor. Idealmente el número de revoluciones dadas por el motor para una referencia fija, debería ser siempre la misma sin importar cuántas veces se

repita la prueba. En la práctica, los resultados de la prueba fueron los esperados. Los motores dan el número de vueltas esperado para el mismo cambio en la posición del volante. Rara vez, el posicionamiento del motor falla por máximo una vuelta. Esta prueba deja claro que el posicionamiento de los motores es tan preciso como se esperaba para la plataforma.

5.2 Tope Ficticio

El tope ficticio hace referencia a una característica única en los sistemas Steer By Wire. Esta característica abre la posibilidad de programar mediante software los límites en el recorrido del volante, y por lo tanto de las ruedas. Los sistemas convencionales cuentan únicamente con topes mecánicos, imposibles de ser modificados en tiempo real. El tope ficticio permite que un volante cuyos topes mecánicos determinan un recorrido de 360º, pueda ser programado mediante software para tener un recorrido de únicamente 90º, 180º, o el recorrido que el usuario desee y se adapte mejor a sus necesidades.

La programación de topes ficticios, se realiza en la aplicación ejecutada en el computador portátil. Ya que el volante MOMO de Logitech utilizado en la plataforma tiene un recorrido mecánico de 270º, el software sólo permite programar topes ficticios que involucren un recorrido menor. Para mostrar las capacidades del sistema, el programa incluye dos recorridos ficticios predeterminados: 180º y 90º.

Figura 5.2 Cuadro de selección del recorrido del volante deseado

El usuario escoge cualquiera de estos dos recorridos y al ejecutar la aplicación, el sistema no permite que ni el volante ni los motores alcancen posiciones mayores al recorrido escogido. Idealmente, el momento en que alcanza el tope ficticio, el volante es trabado por el motor eléctrico encargado de la retroalimentación de fuerza al mismo. Sin embargo, el motor eléctrico del volante Logitech no tiene el torque suficiente para trabar el volante. Debido a esto, se ha programado que el volante ejerza una fuerza en dirección contraria a la del movimiento el momento de alcanzar el tope ficticio y que la tarjeta de adquisición de datos no transmita señales que correspondan a posiciones del volante fuera de los topes ficticios, asegurando así que los motores también se posicionen únicamente dentro de sus respectivos límites ficticios.

Las siguientes gráficas, presentadas en la figura 5.3 y figura 5.4, muestran la diferencia en el comportamiento del sistema al establecer topes ficticios. Las gráficas presentadas en la figura 5.3, muestran la posición y la fuerza para el caso en el que no se desea tener un tope ficticio. En dicho caso, para cualquier posición, la fuerza ejercida por el volante es nula. La situación es diferente para el caso en el que se establece un tope ficticio para limitar el recorrido del volante a 180°. La figura 5.4, muestra la retroalimentación de fuerza al establecer topes ficticios. Mientras el volante se mantiene dentro de los topes ficticios, la fuerza retroalimentada es nula, pero el momento en el que el volante llega a uno de dichos topes, la fuerza que se retroalimenta cambia a su magnitud máxima, positiva ó negativa, dependiendo de la dirección de movimiento del volante.

Como se observa la acción del volante es bastante precisa, por lo que se considera que la plataforma cumple esta prueba con éxito.

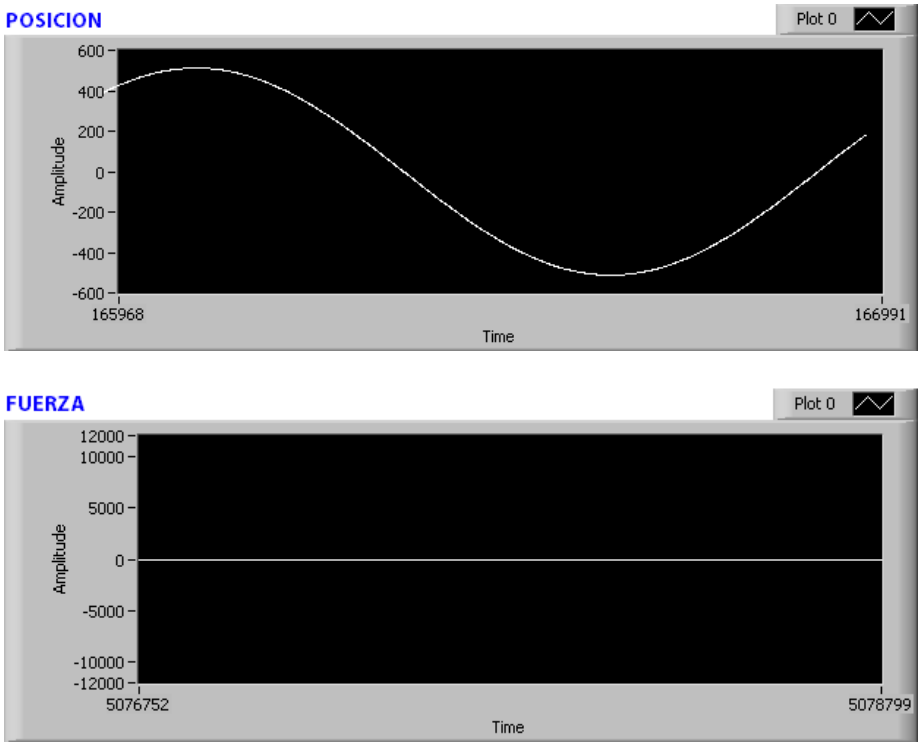


Figura 5.3 Posición y fuerza sin tope ficticio

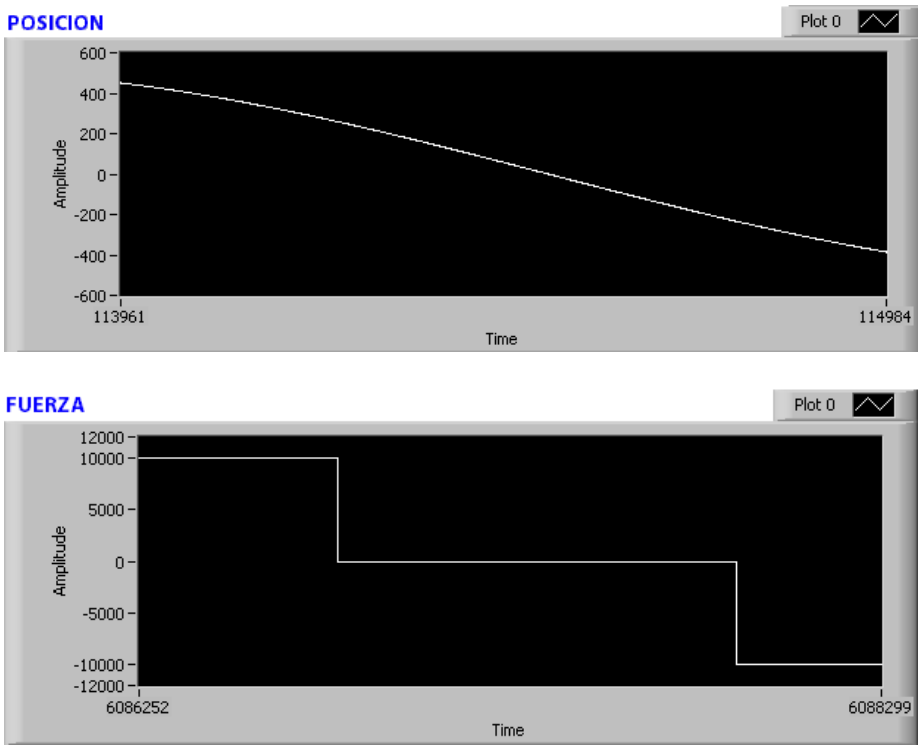


Figura 5.4 Posición y fuerza con tope ficticio para un recorrido de 180°

5.3 Cambio relación entre giro del volante y giro de ruedas

El cambio de relación entre el giro del volante y el giro de las ruedas es otra característica única de los sistemas Steer By Wire. Los sistemas convencionales de dirección tienen una relación fija entre el giro del volante y el giro de las ruedas. Los sistemas Steer By Wire, al especificar dicha relación dentro de su software de control, abren la posibilidad de que el sistema de dirección cuente con un sinnúmero de relaciones. De esta forma el vehículo con Steer By Wire puede adaptar el sistema de dirección a diversas situaciones y siempre priorizar la comodidad y seguridad del conductor. Por ejemplo, con los sensores adecuados, se puede detectar si el conductor se encuentra estacionando el vehículo, y alterar la relación de giro de tal forma que un pequeño giro del volante signifique un giro mucho más significativo en las ruedas. Esto reduce al mínimo el esfuerzo normalmente asociado a las maniobras de parqueo.

En la plataforma de investigación, la posibilidad de cambiar la relación de giro permite investigar la dinámica de diversos sistemas de dirección sin la necesidad de realizar ningún cambio en la estructura física del sistema. Con un rápido cambio en el software se puede conseguir que el sistema dirección pase de comportarse como el de un vehículo normal a comportarse como el de un camión.

Como todas las pruebas, este cambio de relación puede ser realizado en el panel frontal de la aplicación ejecutada en el computador portátil. Como se mencionó brevemente en la sección 4.2, existen 4 relaciones de giro predeterminadas entre las cuales puede escoger el usuario. Las relaciones fueron seleccionadas, de tal forma que las diferencias entre una y otra puedan ser notadas con facilidad. La figura 5.5 muestra una imagen tomada del panel frontal de la aplicación donde se muestra el cuadro con las cuatro relaciones de giro predeterminadas.

Figura 5.5 Cuadro de selección de la relación el giro del volante y giro de las ruedas

Las siguientes gráficas, presentadas en las figura 5.6 muestran la posición de los motores respecto al tiempo para cada una de las diferentes relaciones de giro. Para todos los casos el volante se movió el recorrido mínimo necesario para iniciar el movimiento de los motores. A pesar de que la velocidad de muestreo de la tarjeta de adquisición de datos no permite establecer con exactitud el número de vueltas dadas por los motores, es claro que en cada gráfica el número de vueltas dadas por recorrido mínimo del volante es diferente. Las gráficas son útiles para demostrar que la relación de giro de los motores respecto al movimiento del volante se modifica correctamente según la orden enviada por el usuario, sin embargo, el microcontrolador adicional implementado únicamente para contar el número de vueltas dadas por los motores, junto a la pantalla LCD, permite observar con mayor precisión el cambio en la relación de giro de los motores, y concluir que el mismo es tan preciso como se esperaba para esta prueba.

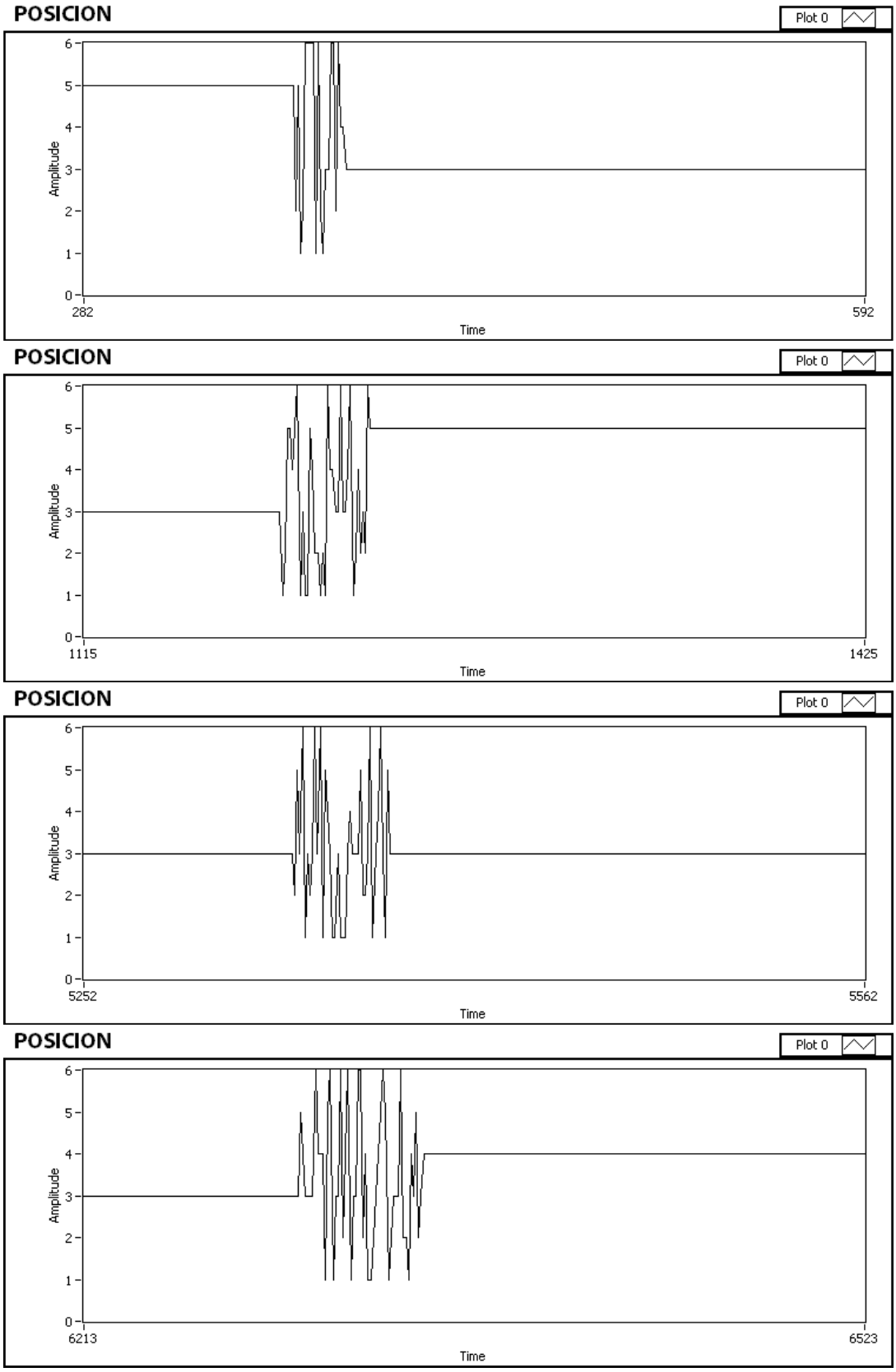


Figura 5.6 Número de vueltas para relaciones de giro de 1:20, 1:25, 1:30 y 1:35 respectivamente

5.4 Orientación independiente de las ruedas

La orientación independiente de la ruedas, es una característica ciertamente impresionante de la tecnología Steer By Wire. Los sistemas mecánicos capaces de orientar independientemente las ruedas de un vehículo, existen únicamente como parte de automóviles conceptuales, y requieren de complicados sistemas de distribución de torque. (HowStuffWorks, Jeep Hurricane). La tecnología By Wire, al utilizar actuadores eléctricos, permite ubicar un actuador en cada rueda del vehículo, sin interferir significativamente con la estructura mecánica del mismo. El único nivel de complejidad para proveer de orientación independiente a las ruedas, entra en las especificaciones que deben ser tomadas en cuenta en el desarrollo del software de control.

La orientación independiente de las ruedas, abre campo para proveer y fabricar vehículos en el futuro con nuevas características. El hecho de que los actuadores posibiliten orientar las ruedas traseras del vehículo, ya es una mejora y diferencia indiscutible ante un sistema convencional de dirección que accede únicamente a la orientación de las ruedas frontales. Con la capacidad de orientar las ruedas independientemente, el vehículo, puede realizar giros sobre su propio eje, es decir con un radio de giro de 0º. Algo imposible en los vehículos convencionales. De igual manera, durante una maniobra de parqueo las ruedas pueden ser orientadas de forma que el vehículo se desplace lateralmente, como un cangrejo (HowStuffWorks, Jeep Hurricane). Como ejemplo de lo que se espera alcanzar con la orientación independiente de las ruedas en un sistema Steer By Wire, la figura 5.7, muestra una imagen del Jeep Hurricane, un concepto de Chrysler que implementa un sistema de dirección mecánico que cuenta con la capacidad de giro de 0º. Ciertamente proveer de esta capacidad a un vehículo mediante

actuadores eléctricos resultaría mucho más barato que hacerlo mediante medios mecánicos como es el caso de este Jeep.

Figura 5.7 Jeep Hurricane en curva con 0º de radio de giro (Dymler Chrysler).

La plataforma desarrollada en este proyecto cuenta con una prueba que permite demostrar el funcionamiento de un sistema Steer By Wire con la capacidad de orientación independiente de sus ruedas. Los dos motores con los que cuenta el sistema, son suficientes para mostrar la capacidad de independizar su orientación. En el panel frontal de la aplicación principal de control, se encuentra un switch de control para activar o desactivar la orientación independiente de los motores. Con fines demostrativos, si la orientación independiente se activa, los motores girarán en sentido contrario; mientras que si dicha característica se encuentra desactivada los dos motores giran hacia el mismo lado, como en cualquier vehículo convencional. Debido a que el muestreo de las señales de sensores no es suficiente para poder determinar el sentido de giro de los motores, los resultados de esta prueba pueden ser confirmados observando el movimiento de los motores con la plataforma en funcionamiento.

La figura 5.8 muestra una imagen del control para activar/desactivar la orientación independiente de los motores.

Figura 5.8 Control para selección de orientación independiente

Capítulo 6

Conclusiones y Recomendaciones Finales

En este proyecto final, se han revisado la teoría, diseño y resultados para la implementación de una plataforma de investigación de un sistema Steer By-Wire. Se describieron en profundidad las etapas de control de posición, retroalimentación de fuerza y desarrollo de la aplicación de control que, en conjunto, permiten el funcionamiento de la plataforma como un sistema de dirección completamente electrónico.

Si bien el desarrollo a nivel comercial de un sistema como el tratado en este proyecto no se da en nuestro país; la realización de este proyecto, deja claro que el desarrollo de sistemas By Wire a nivel de investigación académica es posible, y con presupuestos aceptables para las posibilidades económicas en nuestro medio. También es importante notar, que este proyecto abre las puertas para que la USFQ inicie investigaciones en temas actuales relacionados con los sistemas de dirección y su influencia sobre la dinámica de los vehículos. La plataforma del sistema Steer By Wire, permitirá investigar y desarrollar algoritmos de control para sistemas de dirección, así como reproducir investigaciones que se realizan en importantes centros de investigación alrededor del mundo, como es el caso de las investigaciones de la Universidad de Standford y su Laboratorio de Diseño Dinámico. Retroalimentación de fuerza al conductor, estabilización de un vehículo en sus límites de maniobrabilidad, y asistencia electrónica para evitar que el vehículo colisione con otros vehículos o abandone el camino, son algunos de los temas en los que se enfocan las investigaciones que se realizan en la actualidad en dicho laboratorio, y que podrían ser utilizadas como ejemplo para mantener a la USFQ en un campo actual de investigación (ddl.stanford.edu).

Por otro lado, la plataforma construida constituye un proyecto innovador en lo que se refiere a las soluciones presentadas ante las limitaciones económicas que se han enfrentado durante su desarrollo. El controlador generado para proveer acceso total sobre el volante de juegos Logitech, junto a los circuitos de control y potencia construidos con componentes electrónicos disponibles en el mercado local, representan desafíos superados con éxito y que demuestran la capacidad de desarrollar este tipo de proyectos en el país.

Si bien, la plataforma desarrollada implementa un sistema bastante simplificado, ha permitido analizar y reproducir las principales características de un sistema Steer By Wire, e identificar las ventajas que ubican este tipo de sistemas como el futuro de los sistemas de dirección en los automóviles. De igual manera, el desarrollo de la plataforma ha sido suficiente para poder identificar las características claves de estos tipos de sistemas que deben tratarse en futuras investigaciones para que en algún momento reemplacen los sistemas de dirección convencionales. Ha quedado claro que es necesario desarrollar soluciones ante temas como la seguridad del sistema, costos, y retroalimentación de fuerza, antes de que los sistemas Steer By Wire ganen espacio dentro del mercado automotriz mundial.

Ya que, limitaciones de tiempo y recursos llevaron a que el desarrollo de la plataforma de investigación presentada en este proyecto se enfoque en responder únicamente las necesidades básicas que involucran el diseño e implementación de un sistema Steer By Wire, muchos otros temas involucrados en el desarrollo de un sistema Steer By Wire no pudieron ser tratados con profundidad. Sería importante que se apoyen nuevos proyectos en los que temas como la selección de los actuadores para la implementación del sistema en un vehículo, sistemas eléctricos, sistemas de comunicación, métodos de diagnósticos de fallas, sistemas de cuatro ruedas direccionales, retroalimentación de fuerza en un volante convencional, entre otros puedan ser tratados

realizados con la dedicación y presupuesto que cada uno de ellos merece. El objetivo, y a la vez recomendación, es que las investigaciones futuras a realizarse a partir de este proyecto tengan como meta integrar un sistema Steer By Wire, con todas sus capacidades, a un vehículo convencional.

Basándonos en los resultados obtenidos para las pruebas realizadas en el capítulo 5, y tomando en cuentas las limitaciones ya mencionadas, el sistema desarrollado responde dentro de lo esperado a las necesidades de un sistema de dirección.

En resumen, el proyecto ha resultado un éxito en cuanto a marcar un punto de partida y generar los recursos necesarios para el desarrollo de importantes investigaciones relacionadas con la tecnología By Wire.

Bibliografía

Amberkar S., Bolourchi F., Demerly J., y Millsap Scott. *A Control System Methodology for Steer by Wire Systems*, Steering and Suspension Technology Symposium (2004): SP-1868.

Areya, Anand, Cattle Bryan, Collins Brendan, Franken Gordon, y Saxe Andrew. *DARPA Grand Challenge 2005*, Princeton University, 2005.

Autocosmos, Mazda Washu: un auto del otro mundo. < http://www.autocosmos.com.ar/noticias/expand.asp?id=5871>

Bertoluzzo M., Bolognesi P., Buja B. G., Land A., y Zucollo A. *Drive-by-Wire Systems for Ground Vehicles*, IEEE (2004):0-7803-8304-4

Brown, Ward. *Brushless DC Motor Control Made Easy,* Microchip, AN857, Technology Inc., 2002.

Chou, Eric. *Steer by Wire System – Hardware in the Loop Simulation*, Department of Information of Technology and Electrical Engineering, University of Queensland, 2002.

Ciberhábitat, De lo analógico a lo digital, Agosto 2001. <http://www.ciberhabitat.org.mx/museo/historia/texto/texto_guerra.htm>

DirectX SDK Samples: FFConst Sample. Microsoft Corporation, 2007. <http://msdn2.microsoft.com/en-us/library/bb173383(VS.85).aspx>

e-mas, Generaciones de computadoras. <http://www.e-mas.co.cl/categorias/informatica/gencomp1.htm>

elmundo.es, Gráficos Interactivos, Cómo funciona un avión no tripulado, 15 Septiembre 2004. <http://www.elmundo.es/elmundo/2004/graficos/oct/s2/piloto_automatico.html>

Eyewitness to History*, Henry Ford Changes de World, 1908,* 2005. <http://www.eyewitnesstohistory.com/ford.htm>

How Stuff Works, *Jeep Hurricane*, 2005. < http://auto.howstuffworks.com/jeep-hurricane.htm >

Hubert, Charles., *Electric Machines:Theory Operation, Applications, Adjustment and Control,* Macmillian Publishing Company, New York, 1991.

Johannessen, Per. *GAST Demonstrators*, Volvo Cars, 2003

LabView 8.2 Help/Input Device Control VIs, National Instruments Corporation, 2006. <http://zone.ni.com/reference/en-XX/help/371361D-01/glang/input_device_control_vis/>

Leppin, Leiann. *The Conversion of a General Motors Cadillac SRX to Drive-By-Wire Status*, Virginia Polytechnic Institute and State University, 2005.

Manual CEAC Del Automóvil, Grupo Editorial CEAC S.A, 2003.

Microsoft Help: DirectInput, Microsoft Corporation, 2008. <http://msdn2.microsoft.com/en-us/library/bb219802(VS.85).aspx>

Microsoft Help: The Window Procedure, Microsoft Corporation, 2008. <http://msdn2.microsoft.com/en-us/library/aa383738.aspx>

Sistema de Dirección. Atiko Estudio, Diseño Industrial, 2004-2007. <http://www.atikoestudio.com/disenador/industrial/automovil/direccion.htm>

Yedamale, Padmaraja. *Brushless DC Motor Fundamentals,* Microchip, AN885, Technology Inc., 2003.

Yih, Paul. *Steer-By-Wire: Implications for vehicle handling and safety*, Department of Mechanical Engineering, Standford, 2005.

Whitfield, Kermit. *Solve for X: X-By-Wire technologies,* Diciembre 2001. <http://findarticles.com/p/articles/mi_m0KJI/is_12_113/ai_81147817>

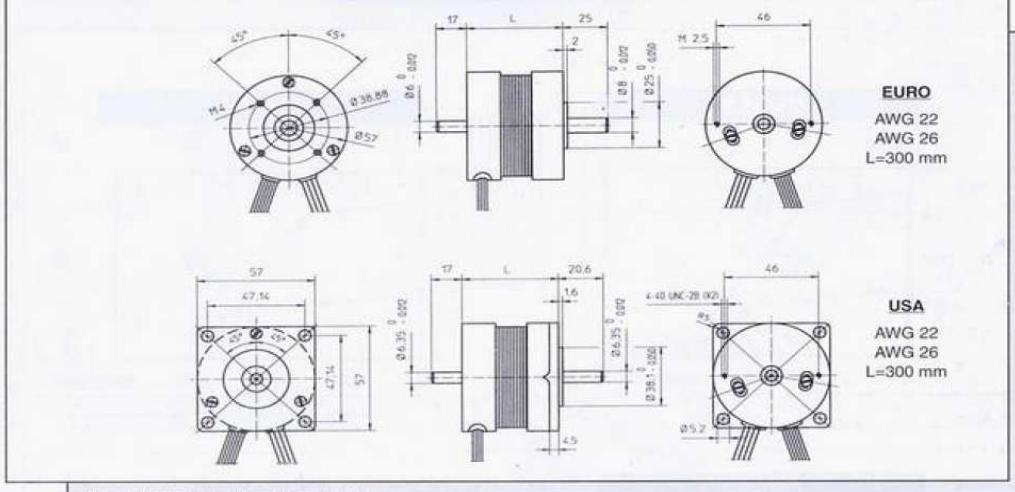
Wikipedia, General Motors Hy-Wire, 22 Mayo 2007. <http://en.wikipedia.org/wiki/General_Motors_Hy-wire>

Anexo A

Hoja Técnica Motores BLDC

SERVOMOTORI BRUSHLESS
BRUSHLESS SERVOMOTORS

SERIE BM05



Dimensioni in mm - *Dimensions in mm*

Specifiche / *Specifications*		BM05 / 1	BM05 / 2	BM05 / 3
- coppia continuativa allo stallo (Δtcu 80°C) - *continuous stall torque (Δtcu 80°C)*	Nm	0,14	0,27	0,39
- coppia continuativa alla velocità nominale - *continuous torque at rated speed*	Nm	0,11	0,22	0,32
- coppia max di sovraccarico (S.I.R. 10%) - *maximum peak torque (duty cycle 10%)*	Nm	0,35	0,68	0,98
- velocità nominale - *rated speed*	1/min. *RPM*	4000	4000	4000
- potenza alla velocità nominale - *output power at rated speed*	Watt	46	92	133
- momento d'inerzia rotorico - *rotor inertia*	Kgm^2	$7,5 \cdot 10^{-6}$	$11,9 \cdot 10^{-6}$	$17,3 \cdot 10^{-6}$
- costante di coppia - *torque constant*	Nm/Adc	0,08	0,084	0,084
- resistenza ai morsetti a 20°C - *terminal resistance at 20°C*	ohm	2,8	1,3	0,78
- induttanza concatenata - *line to line inductance*	mH	8,6	4,2	2,7
- F.C.E.M. alla vel. nominale - *B.E.M.F. at nominal speed*	Vrms	27	28	28
- corrente nominale a rotore bloccato - *locked rotor rated current*	Adc	1,75	3,2	4,7
- corrente di picco massima - *max peak current*	Adc	5	9	13
- lunghezza (L) - *lenght (L)*	mm	55	75	95
- massa - *mass*	Kg	0,5	0,75	1

3 Sensori di Hall a 120° elettrici - Caratteristiche elettriche: Iin=30 mA - Vin = 4÷24V
3 Hall sensors at electrical 120° - Electrical specs.: Iin - 30 mA - Vin = 4÷24V

Anexo B

Librería de Enlace Dinámico (DLL) desarrollada para retroalimentación de fuerza al volante MOMO Logitech

```
#using <System.Drawing.dll>
#using <System.Windows.Forms.dll>
#using <System.dll>
#using <user32.dll>

using namespace System;
using namespace System::Runtime::InteropServices;
using namespace System::Drawing;
using namespace System::Windows::Forms;

#define STRICT
#define DIRECTINPUT_VERSION 0x0800



//INCLUDES
#include "stdafx.h"
#include "dinput.h"
#include "math.h"
#include "mmsystem.h"
#include "windows.h"
#include <tchar.h>
#include <windowsx.h>
#include <commctrl.h>
#include <basetsd.h>
#pragma warning( disable : 4996 )
#include <strsafe.h>
#pragma warning( default : 4996 )
#include "resource.h"
#if defined(DEBUG) | defined(_DEBUG)
#include <crtdbg.h>
#endif



//DEFINES Y VARIABLES
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
#define SAFE_DELETE(p)  { if(p) { delete (p);     (p)=NULL; } }
#define SAFE_RELEASE(p) { if(p) { (p)->Release(); (p)=NULL; } }

LPDIRECTINPUT8        g_pDI       = NULL;
BOOL                  g_bActive   = TRUE;
DWORD                 g_dwNumForceFeedbackAxis = 0;
LPDIRECTINPUTEFFECT   g_pEffect   = NULL;
LPDIRECTINPUTDEVICE8  g_pDevice   = NULL;
unsigned short num_joypads=0;
GUID joyid[1];
DIPROPDWORD dipdw;
HRESULT     hr;
```

```
HINSTANCE hinst;
HWND hwndp;
WNDPROC OldProc;


//PROTOTIPOS DE LAS FUNCIONES UTILIZADAS
BOOL CALLBACK EnumFFDevicesCallback( const DIDEVICEINSTANCE* pInst, VOID*
pContext );
BOOL CALLBACK EnumAxesCallback( const DIDEVICEOBJECTINSTANCE* pdidoi,
VOID* pContext );
VOID    FreeDirectInput();
HRESULT InitDirectInput(HWND hwnd);
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM
lParam);



//PROTOTIPOS FUNCIONES EXPORTADAS POR EL DLL
extern "C"_declspec (dllexport) VOID terminar(HWND hwnd);
extern "C"_declspec (dllexport) HWND crearventana();
extern "C"_declspec(dllexport) HRESULT info(HWND hDlg,long *but,long
*axis);
extern "C"_declspec (dllexport) HRESULT iniciarefectorawforce(HWND hwnd,
long *force);

//DLLMAIN: Manejo de la instancia del DLL
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID
lpvReserved)
{
      InitCommonControls();

      switch(fdwReason)
  {
            case DLL_PROCESS_ATTACH:
      {
      hinst=hinstDLL;
      }
      break;

      case DLL_PROCESS_DETACH:
            {

            }
            break;
    }
  return TRUE;
}



//WNDPROC:Manejo de los mensajes a la ventana de la aplicación en LabView
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM
lParam)
{
      switch(msg)
            {
```

```
          case WM_ACTIVATE:
                  {
                      g_pDevice->Acquire();
                           g_pEffect->Start(1,0);
                  }
            break;
            default:
            return CallWindowProc(OldProc,hwnd, msg, wParam, lParam);
            }
      return 0;
}



//ENUMFFDEVICESCALLBACK: Encuentra dispositivo con soporte para
ForceFeedback
BOOL CALLBACK EnumFFDevicesCallback( const DIDEVICEINSTANCE* pInst, VOID*
pContext )
{
    LPDIRECTINPUTDEVICE8 pDevice;
    HRESULT              hr1;

    // OBTENCION DE INTERFASE PARA EL DISPOSITIVO CON FORCE FEEDBACK
ENCONTRADO
    hr1 = g_pDI->CreateDevice( pInst->guidInstance, &pDevice, NULL );

    if( FAILED(hr1) )
        return DIENUM_CONTINUE;

   //EL PROGRAMA TRABAJA ÚNCAMENTE CON EL PRIMER DirectInputDevice8
ENCONTRADO.
      g_pDevice = pDevice;

    return DIENUM_STOP;
}



//ENUMAXESCALLBACK:Enumera el número de ejes disponibles en el
//dispositivo y configura el rango de sus valores
BOOL CALLBACK EnumAxesCallback( const DIDEVICEOBJECTINSTANCE* pdidoi,
                                VOID* pContext )
{
    DWORD* pdwNumForceFeedbackAxis = (DWORD*) pContext;

    if( (pdidoi->dwFlags & DIDOI_FFACTUATOR) != 0 )
        (*pdwNumForceFeedbackAxis)++;

      if( pdidoi->dwType & DIDFT_AXIS )
    {
        DIPROPRANGE diprg;
        diprg.diph.dwSize       = sizeof(DIPROPRANGE);
        diprg.diph.dwHeaderSize = sizeof(DIPROPHEADER);
        diprg.diph.dwHow        = DIPH_BYID;
        diprg.diph.dwObj        = pdidoi->dwType;
        diprg.lMin              = -512;
        diprg.lMax              = +511;

        //CONFIGURACION DEL RANGO EN EL QUE TRABAJA CADA EJE
```

```
        if( FAILED( g_pDevice->SetProperty( DIPROP_RANGE, &diprg.diph ) )
)
            return DIENUM_STOP;
      }

    return DIENUM_CONTINUE;
}



//FREEDIRECTINPUT: Libera el dispositivo para que pueda ser utilizado por
otras aplicaciones
VOID FreeDirectInput()
{
    if( g_pDevice )
        g_pDevice->Unacquire();

    SAFE_RELEASE( g_pEffect );
    SAFE_RELEASE( g_pDevice );
    SAFE_RELEASE( g_pDI );
}



//TERMINAR: Funcion exportada por el DLL para liberar el dispositivo
desde LabView
extern "C"_declspec (dllexport) VOID terminarDirectInput(HWND hwnd)
{
      FreeDirectInput();
      SetWindowLong(hwnd, GWL_WNDPROC, (DWORD) OldProc);
}



//CREARVENTANA: Encuentra la ventana de LabView, reemplaza su proceso con
//uno propio del DLL que se encarga de manejar los mensajes de interes
para el sistema By Wire.

extern "C"_declspec (dllexport) HWND crearventana()
{
      hwndp = FindWindow("LVDChild","dos.vi");

      if(hwndp == NULL)
            {
                  MessageBox(NULL, "NO WINDOW!", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
            return FALSE;
            }

      SetForegroundWindow(hwndp);
    OldProc = (WNDPROC) GetWindowLongPtr(hwndp, GWL_WNDPROC);
      SetWindowLongPtr(hwndp, GWL_WNDPROC, (LONG_PTR)WndProc);
    InitDirectInput(hwndp);
              MessageBox(NULL, "TODO OK!", "ATENCIÓN!",
                  MB_ICONEXCLAMATION | MB_OK);

      return hwndp;

}
```

```
//INFO: Envia el estado actualizado del joystick a LabView.
extern "C"_declspec(dllexport) HRESULT info(HWND hDlg,long *but,long
*axis)
{
      HRESULT     hr;
      DIJOYSTATE js;
      int i;


      //POLL PARA LEER EL ESTADO ACTUAL DEL VOLANTE
    hr = g_pDevice->Poll();
    if( FAILED(hr) )
    {
        hr = g_pDevice->Acquire();
        while( hr == DIERR_INPUTLOST )
            hr = g_pDevice->Acquire();

        return 7;
   }
//ADQUIRIR DATOS ACTUALIZADOS DEL ESTADO DEL VOLANTE
     if( FAILED( hr = g_pDevice->GetDeviceState( sizeof(DIJOYSTATE), &js
) ) )
        return 8;

//ENVIAR INFORMACION DEL ESTADO DEL VOLANTE
      for(i = 0; i < 8; i++)
  {
        but[i] = js.rgbButtons[i];
  }
  axis[0]=js.lX;
  axis[1]=js.lY;

  return 10;
}

//INICIAREFECTORAWFORCE: Cambia la magnitud de la fuerza para un efecto
//de Fuerza Constante en el joystick. La Magnitud es recibida desde
//LabView.

extern "C"_declspec (dllexport) HRESULT iniciarefectorawforce(HWND hwnd,
long *force)
{
      LONG rglDirection[2] = { 0, 0 };

    DICONSTANTFORCE cf;

    if( g_dwNumForceFeedbackAxis == 1 )
    {
        cf.lMagnitude = force[0];
        rglDirection[0] = 0;
    }
    else
    {
        rglDirection[0] = force[0];
        rglDirection[1] = force[1];
        cf.lMagnitude = (DWORD)sqrt( (double)force[0]* (double)force[0] +
```

```
                                        (double)force[1] * (double)force[1]
);
    }

    DIEFFECT eff;
    ZeroMemory( &eff, sizeof(eff) );
    eff.dwSize                = sizeof(DIEFFECT);
    eff.dwFlags               = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
    eff.cAxes                 = g_dwNumForceFeedbackAxis;
    eff.rglDirection          = rglDirection;
    eff.lpEnvelope            = 0;
    eff.cbTypeSpecificParams  = sizeof(DICONSTANTFORCE);
    eff.lpvTypeSpecificParams = &cf;
    eff.dwStartDelay          = 0;

    if( FAILED( hr = g_pEffect->SetParameters( &eff, DIEP_DIRECTION
|DIEP_TYPESPECIFICPARAMS |
                                               DIEP_START |
DIEP_NODOWNLOAD) ) )
    {
        return hr;
    }
      PostMessage(hwnd,WM_ACTIVATE,0,0);

      return S_OK;
}

//INITDIRECTINPUT: Inicializa la interfase DirectInput, y un efecto de
//FuerzaConstante
HRESULT InitDirectInput(HWND hwnd)
{

      if(FAILED(hr=DirectInput8Create(hinst, DIRECTINPUT_VERSION,
IID_IDirectInput8, (VOID**)&g_pDI, NULL)))
      {
            MessageBox(NULL, "DirectInput8Create FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }
      if(FAILED(hr=g_pDI->Initialize(hinst, DIRECTINPUT_VERSION)))
      {
            MessageBox(NULL, "INICIALIZAR DirectInput FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }

      if(FAILED(hr=g_pDI->EnumDevices( DI8DEVCLASS_GAMECTRL,
EnumFFDevicesCallback, NULL,
                                       DIEDFL_ATTACHEDONLY |
DIEDFL_FORCEFEEDBACK )))
                                                                  {
            MessageBox(NULL, "ENUM DEVICES FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }


      if(FAILED(hr=g_pDevice->SetDataFormat( &c_dfDIJoystick )))
      {
            MessageBox(NULL, "SET DATA FORMAT FALLÓ!", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }
```

```
      if(FAILED(hr=g_pDevice->SetCooperativeLevel( hwnd,DISCL_EXCLUSIVE |
DISCL_FOREGROUND)))
      {
            MessageBox(NULL, "SET COOPERATIVE LEVEL FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }
      dipdw.diph.dwSize       = sizeof(DIPROPDWORD);
    dipdw.diph.dwHeaderSize = sizeof(DIPROPHEADER);
    dipdw.diph.dwObj        = 0;
    dipdw.diph.dwHow        = DIPH_DEVICE;
    dipdw.dwData            = TRUE;

    if(FAILED(hr=g_pDevice->SetProperty( DIPROP_AUTOCENTER, &dipdw.diph
)))
      {
            MessageBox(NULL, "SET PROPERTY FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }
      if(FAILED(hr=g_pDevice->EnumObjects(
EnumAxesCallback,(VOID*)&g_dwNumForceFeedbackAxis, DIDFT_AXIS )))
      {
            MessageBox(NULL, "ENUM OBJETCS FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }

      DWORD           rgdwAxes[2]     = { DIJOFS_X, DIJOFS_Y};
    LONG            rglDirection[2] = { 0, 0 };
    DICONSTANTFORCE cf              = {0};

      cf.lMagnitude=0;

      DIEFFECT eff;
    ZeroMemory( &eff, sizeof(eff) );
    eff.dwSize                  = sizeof(DIEFFECT);
    eff.dwFlags                 = DIEFF_CARTESIAN | DIEFF_OBJECTOFFSETS;
    eff.dwDuration              = INFINITE;
    eff.dwSamplePeriod          = 0;
    eff.dwGain                  = DI_FFNOMINALMAX;
    eff.dwTriggerButton         = DIEB_NOTRIGGER;
    eff.dwTriggerRepeatInterval = 0;
    eff.cAxes                   = g_dwNumForceFeedbackAxis;
    eff.rgdwAxes                = rgdwAxes;
    eff.rglDirection            = rglDirection;
    eff.lpEnvelope              = 0;
    eff.cbTypeSpecificParams    = sizeof(DICONSTANTFORCE);
    eff.lpvTypeSpecificParams   = &cf;
    eff.dwStartDelay            = 0;

      if(FAILED(hr=g_pDevice->CreateEffect( GUID_ConstantForce, NULL,
&g_pEffect, NULL )))
      {
            MessageBox(NULL, "CREAR EFECTO RawForce FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }

      if(FAILED(hr=g_pEffect->SetParameters(&eff,DIEP_AXES |
DIEP_DIRECTION | DIEP_DURATION | DIEP_ENVELOPE
                                                | DIEP_GAIN | DIEP_START |
DIEP_TYPESPECIFICPARAMS | DIEP_NODOWNLOAD)))
    {
```

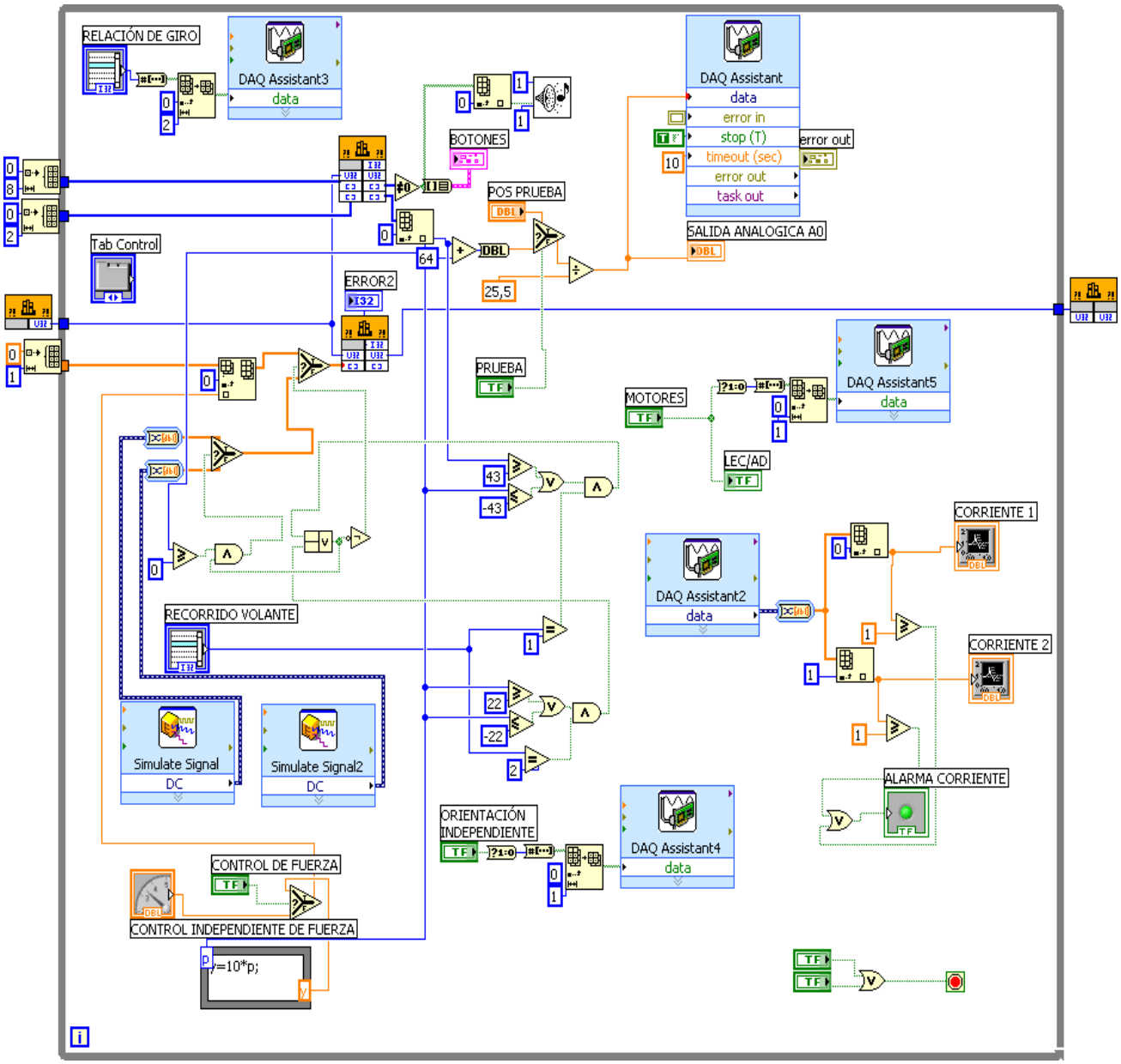
```
            MessageBox(NULL, "SETTING PARAMETERS FALLÓ", "Error!",
                  MB_ICONEXCLAMATION | MB_OK);
      }

      return hr;
}


//FIN DEL DLL
```

Anexo C

Código de la aplicación de control ejecutada en LabView



Anexo D

Código de la aplicación de control ejecutada en el microcontrolador PIC16F877A

```
'*****************************************************************
'*  TESIS PAULO GUERRA: Programa PIC16f877A                      *
'*****************************************************************
DEFINE OSC 20
DEFINE ADC_BITS 8
@ DEVICE pic16F877A
@ DEVICE pic16F877A, WDT_on
@ DEVICE pic16F877A, PWRT_ON
@ DEVICE pic16F877A, PROTECT_OFF
@ DEVICE pic16F877A, HS_OSC

'configuracion A/D
ADCON1 = 2 'usar puertoA como analogo y puertoE como digital
'---------------------------
TRISB.7=1'relacion de giro
TRISB.6=1'relacion de giro
TRISB.5=1'orientacion independiente
TRISB.4=0'DIR 2
TRISB.3=0'DIR 2
TRISB.2=0'DIR 1
TRISB.1=0'DIR 1
TRISB.0=1'activar lectura AD
'---------------------------
TRISC.7=1'HALL MOT2 = DAQ 25
TRISC.6=1'HALL MOT2 = DAQ 26
TRISC.5=1'HALL MOT2 = DAQ 27
'---------------------------
TRISD.0=0'LED1
TRISC.0=0'LED2
TRISC.3=0'LED3

'variables
PREND2 VAR portb.4
DIR2 VAR portb.3
PREND1 VAR portb.2
DIR1 var portb.1
INDP var portb.5
HALL11 var portc.7
HALL12 var portc.6
HALL13 var portc.5
lecAD var portb.0
REL1 var portb.7
REL2 var portb.6

pos var byte        'posicion
contador var byte  'contador vueltas
contador2 var byte
vueltas var byte 'numero de vueltas que se deben dar
posact var byte  'variable que guada ultima posicion del volante
posIN1 var byte  'posicion inicial motor1
posACT1 var byte  'posicion actual motor1
```

```
N var byte 'relacion de giro

low prend1:low prend2


'inicio del programa
high portc.0
prein:
while lecAD=0
low prend1:low prend2
high portc.0
wend
low portc.0
Gosub getpos
posact=pos
low prend1:low prend2


'obtener posicion del volante
inicio:

while lecAD=1

    if REL1=0 and REL2=0 then
    N=20
    endif
    if REL1=0 and REL2=1 then
    N=25
    endif
    if REL1=1 and REL2=0 then
    N=30
    endif
    if REL1=1 and REL2=1 then
    N=35
    endif


    Gosub getpos
    if pos>(posact+3) then
    vueltas=N
    posact=pos
    goto motoresDER
    endif

    if pos<(posact-3) then
    vueltas=N
    posact=pos
    goto motoresIZQ
    endif

    if (posact-3)<pos<(posact+3) then
    goto inicio
    endif

wend
    goto prein
goto inicio
```

```
'----------------------------
motoresDER:
contador=0
posIN1=4*hall11+2*hall12+1*hall13
posact1=posin1

high portd.0
while posact1==posin1
posact1=4*hall11+2*hall12+1*hall13
    IF INDP=0 then
    high Dir1:high dir2:high prend1:high prend2
    else
    LOW Dir1:high dir2:high prend1:high prend2
    endif
wend

contador2=contador
while contador<vueltas and lecad=1
    IF INDP=0 then
    high Dir1:high dir2:high prend1:high prend2
    else
    LOW Dir1:high dir2:high prend1:high prend2
    endif
    posact1=4*hall11+2*hall12+1*hall13
    if posact1==posIN1 and contador2==0 then
    contador=contador+1
        gosub getpos
        if pos>(posact+3) then
        vueltas=vueltas-contador+N
        posact=pos
        endif
        if pos<(posact-3) then
        vueltas=contador+N
        posact=pos
        low prend1:low prend2
        goto motoresIZQ
        endif
    contador2=1
    endif
    if posact1==posIN1 and contador2==1 then
    contador=contador
    endif
    if posact1!=posIN1 then
    contador=contador
    contador2=0
    endif
wend
vueltas=0
low portd.0
low prend1:low prend2
goto inicio

'----------------------------
motoresIZQ:
high portc.3
contador=0
posIN1=4*hall11+2*hall12+1*hall13
posact1=posin1

while posact1==posin1
```

```
posact1=4*hall11+2*hall12+1*hall13
    IF INDP=0 then
    low Dir1:low dir2:high prend1:high prend2
    else
    high Dir1:low dir2:high prend1:high prend2
    endif
wend

contador2=contador
while contador<vueltas and lecad=1
    IF INDP=0 then
    low Dir1:low dir2:high prend1:high prend2
    else
    high Dir1:low dir2:high prend1:high prend2
    endif
    posact1=4*hall11+2*hall12+1*hall13
    if posact1==posIN1 and contador2==0 then
    contador=contador+1
        gosub getpos
        if pos<(posact-3) then
        vueltas=vueltas-contador+N
        posact=pos
        endif
        if pos>(posact+3) then
        vueltas=contador+N
        posact=pos
        low prend1:low prend2
        goto motoresDER
        endif
    contador2=1
    endif
    if posact1==posIN1 and contador2==1 then
    contador=contador
    endif
    if posact1!=posIN1 then
    contador=contador
    contador2=0
    endif
wend
low portc.3
vueltas=0
low prend1:low prend2
goto inicio
'----------------------------

'lecturaAD
getpos:
        ADCON0=$41            ' leer AN0
        Gosub getad
        pos = ADRESH
Return

getad:
        Pauseus 30              ' esperar setup del canal
        ADCON0.2 = 1          ' iniciar conversion
        pauseus 25            ' esperar conversion
Return
```

Anexo E

Código de la aplicación de control ejecutada en los microcontroladores PIC16F628A

```
;*********************************************************************
;                                                                    *
;    TESIS: PAULO GUERRA PROGRAMA PIC16F628A                          *
;                                                                    *
;*********************************************************************

      list      p=16f628A           ; definicion procesador
      #include <p16F628A.inc>       ; archivo .inc para el procesador

      errorlevel  -302              ; eliminar msn 302

      __CONFIG   _CP_OFF & _DATA_CP_OFF & _LVP_OFF & _BOREN_OFF &
_MCLRE_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC
.

;***** VARIABLE DEFINITIONS
      w_temp        EQU     0x7E        ; variable guardar contexto
      status_temp   EQU     0x7F        ; variable guardar contexto

      CBLOCK 0X20
      REF
      ULTSensor
      NEWSensor
      DriveWord
      CONT
      ENDC

;*********************************************************************
;*
;* Define I/O
;*
#define     OffMask           B'10101011'
#define     DrivePort   PORTB
#define DrivePortTris   TRISB
#define Dir             PORTA,0
#define Prend           PORTA,1

;*********************************************************************
      ORG     0x000             ; vector de reset
      nop                               ;
      clrf    PCLATH            ; limpiar bits de pagina
      goto    Initialize        ; iniciar programa

      ORG     0x004             ; vector de interrupcion
      movwf   w_temp            ; guardar registro W
      movf  STATUS,w          ;
      movwf status_temp       ; guardar registro STATUS

      btfss Prend
      goto    Des
```

```
      incf  CONT,f
      movlw   0x04
      xorwf   CONT,w
      btfss STATUS,Z
      goto        IRef
      clrf  CONT
      movlw 0xFF
      xorwf REF,w
      btfss STATUS,Z
      incf  REF,f
      goto    IRef

Des
      incf  CONT,f
      movlw   0x04
      xorwf   CONT,w
      btfss STATUS,Z
      goto        IRef
      clrf  CONT
      movlw 0x00
      xorwf REF,w
      btfss STATUS,Z
      decf  REF,f

IRef
      bcf         INTCON,2
      movf    status_temp,w     ; regresar al estado antes de
interrupcion
      movwf STATUS            ;
      swapf   w_temp,f
      swapf   w_temp,w          ;
      retfie                    ;


;*********************************************************************
;*
;* Inicializar
;*

Initialize
      clrf  DrivePort   ; apagar salidas
      movlw       0x07
      movwf       CMCON

      banksel TRISA
; setup I/O
      clrf  DrivePortTris     ; PORTB como salida
      movlw B'10111111' ;
      movwf TRISA       ;
; setup Timer0
      movlw B'11010000' ; Timer0: Fosc, 1:2
      movwf OPTION_REG
; setup Interrupciones
      movlw B'10100000' ;
      movwf INTCON

      banksel     PORTA

      clrf  ULTSensor   ; inicializacion variables
      clrf  NEWSensor
```

```
      clrf  CONT
      clrf  REF
      call  Commutate   ; determinar posicion actual del motor
      goto  Loop

;*********************************************************************
;*
;* Main control loop
;*
Loop
      incfsz      REF,w       ; si REF es 0xFF obviar PWM
      goto  PWM             ; suma REF a timer0 para PWM
      movf  DriveWord,w ; encender drivers
      goto  Drive       ;
PWM
      movf  REF,w       ; leer REF
      addwf TMR0,w            ; sumarla a timer0
      movf  DriveWord,w ;
      btfss STATUS,C    ; pronar si REF + timer0 resulta en carry
      andlw OffMask           ; si no hay carry apagar drivers
Drive
      clrf  ULTSensor
      movwf DrivePort   ; habilitar drivers
      call  Commutate   ; determinar si ha habido cambio en los sensores
      goto  Loop        ; repetir



;*********************************************************************
;*    Lectura de la referencia de velocidad
;*;

;*********************************************************************
;*
;* Leer sensores y actualizar los drivers si existe cambio
;*
;*
Commutate
      movlw 0x00
      movwf NEWSensor

      btfsc  PORTA,5
      bsf    NEWSensor,0
      btfsc  PORTA,4
      bsf    NEWSensor,1
      btfsc  PORTA,3
      bsf    NEWSensor,2

      movf    NEWSensor,w
      xorwf ULTSensor,w ; probar si hay cambio
      btfsc STATUS,Z    ; cero si no hay cambio
      return                        ; no hay cambio regresar al lazo PWM

      xorwf ULTSensor,f ; reemplazar datos del ultimo sensor leido
      btfss   Dir         ; direccion de movimiento
      goto  FwdCom            ; determinar direccion de movimiento

                                    ; movimiento en reversa
      movlw HIGH RevTable     ; leer valor en tabla
      movwf PCLATH                  ;
```

```
      movlw LOW RevTable      ;
      goto  Com2

FwdCom                        ; movimiento hacia adelante
      movlw HIGH FwdTable     ; leer valor en tabla
      movwf PCLATH            ;
      movlw LOW FwdTable      ;
Com2
      addwf ULTSensor,w ; determinar valor correspondiente al sensor
      btfsc STATUS,C    ;
      incf  PCLATH,f    ;

      call  GetDrive    ; guardar orden para los drivers
      movwf DriveWord   ;
      return

GetDrive
      movwf PCL

;*********************************************************************
;*    Tablas para conmutación
;*
FwdTable
      retlw B'00000000' ; invalid
      retlw B'01000010' ;
      retlw B'00110000' ;
      retlw B'00010010' ;
      retlw B'00001100' ;
      retlw B'01001000' ;
      retlw B'00100100' ;
      retlw B'00000000' ; invalid

RevTable
      retlw B'00000000' ; invalid
      retlw B'00100100' ;
      retlw B'01001000' ;
      retlw B'00001100' ;
      retlw B'00010010' ;
      retlw B'00110000' ;
      retlw B'01000010' ;
      retlw B'00000000' ; invalid

      END                       ; fin de programa
```

Anexo F

Código de la aplicación para contar las revoluciones de los motores y mostrarlas en una pantalla LCD - PIC16F628A

```
'*****************************************************************
'*  TESIS: Programa PIC16f628A (CONTEO REVOLUCIONES LCD)         *
'*****************************************************************
DEFINE OSC 20
DEFINE INTHAND cont
@ DEVICE pic16F628A
@ DEVICE pic16F628A, WDT_ON
@ DEVICE pic16F628A, MCLR_OFF
@ DEVICE pic16F628A, PWRT_ON
@ DEVICE pic16F628A, PROTECT_OFF
@ DEVICE pic16F628A, HS_OSC

DEFINE LCD_DREG PORTB
DEFINE LCD_DBIT 4
DEFINE LCD_RSREG PORTB
DEFINE LCD_RSBIT 3
DEFINE LCD_EREG PORTB
DEFINE LCD_EBIT 2

CMCON=7
TRISA.0=1
TRISA.1=1
TRISA.2=1
TRISA.3=1
TRISA.4=1
TRISA.5=1
TRISA.6=0
TRISA.7=1

HALL1 VAR porta.5
HALL2 var porta.4
HALL3 VAR porta.3
posi  var byte
posa  var byte
cont  var byte
flag var BYTE
flag=1

wsave var   byte $20 system
ssave var   byte bank0 system
psave var   byte bank0 system

Symbol T0IE = INTCON.5 ' Habilitar Interrupción por Overflow del Timer0
Symbol T0IF = INTCON.2 ' Flag de interrupción de Timer0 por overflow.
Symbol GIE = INTCON.7 ' Habilitar Interrupción Global
Symbol PS0 = OPTION_REG.0 ' Prescaler bit-0
Symbol PS1 = OPTION_REG.1 ' Prescaler bit-1
Symbol PS2 = OPTION_REG.2 ' Prescaler bit-2
Symbol PSA = OPTION_REG.3
Symbol T0CS = OPTION_REG.5

GIE = 0 ' Deshabilitar interrupciones globales
While GIE = 1:GIE = 0:Wend
```

```
PSA = 1
PS0 = 0
PS1 = 0
PS2 = 0
T0CS = 0
TMR0 = 0
T0IE = 1
GIE = 1
LCDOUT  $FE,1,"CONTEO"
LCDOUT  $FE,$C0,"REVOLUCIONES"
posi=4*hall1+2*hall2+hall3
cont=0
goto inicio

ASM
cont
        movwf     wsave
            swapf STATUS, W
            clrf  STATUS
            movwf ssave
            movf  PCLATH, W
            movwf psave
endasm
T0IE=0
high portb.3
posa=4*hall1+2*hall2+1*hall3
if porta.2==1 then
   if posa==posI and flag=0 then
    cont=cont+1
    flag=1
   endif
   if posa!=posi then
   flag=0
   endif
endif
low portb.3
T0IE=1
T0IF=0
asm
End_Int
        movf      psave, W
            movwf PCLATH
            swapf ssave, W
            movwf STATUS
            swapf wsave, F
            swapf wsave, W
            retfie
endasm


inicio:

'contar revoluciones cuando se prenden los motores
if porta.2=0 then
    gosub show 'si los motores paran mostrar el número de vueltas
endif

goto inicio

show:
```

```
while porta.2=0 'mostrar numero de vueltas en pantalla LCD
  LCDOUT  $FE,1,"#REV=",DEC cont
wend
return
```