ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Control de velocidad por cambio de frecuencia de motor trifásico sincrónico usando microcontroladores avanzados e interfaz serial para la visualización de resultados."

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRICIDAD

ESPECIALIDAD EN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

Presentado por:

Jorge Gonzalo Espinoza Mendoza

Freddy Andy Rosero Vera

GUAYAQUIL – ECUADOR

AÑO 2010

AGRADECIMIENTO

Este seminario de graduación, si bien ha requerido de esfuerzo y mucha dedicación por parte de los autores y su director, no hubiese sido posible su finalización sin la cooperación desinteresada de todas y cada una de las personas que aportaron su granito de arena y muchas de las cuales han sido un soporte muy fuerte en momentos de angustia y desesperación.

Primero y antes que nada, dar gracias a **Dios**, por estar con nosotros en cada paso que damos, por fortalecer nuestros corazones e iluminar nuestras mentes y por haber puesto en nuestro camino a aquellas personas que han sido nuestro soporte y compañía durante todo el periodo de estudio.

DEDICATORIA

A Dios que siempre está con nosotros, siendo su amor la fuente de energía para alcanzar nuestras metas.

A nuestros padres, agradecerles por el apoyo brindado en todo sentido durante estos años de estudio. Gracias por su paciencia y amor, quienes siempre nos inculcaron perseverancia con valores éticos, permitiéndonos iniciar nuestra vida profesional.

Una dedicatoria muy especial a nuestros hijos Mathias Rosero y Angie Espinoza.

TRIBUNAL DE SUSTENTACIÓN

_____

MSc. Carlos Valdivieso

Director de Seminario de Graduación

_____

MSc. Hugo Villavicencio

Delegado del Decano

DECLARACIÓN EXPRESA

“La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesina, nos corresponde exclusivamente, y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL.”

(Reglamentos y exámenes y títulos profesionales de la ESPOL)

_____

JORGE ESPINOZA MENDOZA

_____

FREDDY ROSERO VERA

RESUMEN

El presente documento corresponde al seminario de graduación “Microcontroladores Avanzado”, este proyecto consiste en un “Control de velocidad por cambio de frecuencia de un motor trifásico sincrónico usando microcontroladores avanzados e interfaz serial para la visualización de resultados”.

Este informe describe el funcionamiento, desarrollo e implementación de un prototipo de controlador para motores BLDC con sensores de efecto Hall y sin sensor, desarrollado por Microchip en las notas de aplicación AN857A y AN957. En dichas notas se desarrollan dos tipos de programas, uno que permite manejar motores BLDC sin sensor de efecto Hall y el otro programa, para motores BLDC con sensor de efecto Hall. En este proyecto se implementó los dos programas para controlar motores BLDC.

Además se expresa en este informe un marco teórico que describe:

1) El microcontrolador a utilizarse en este proyecto, el PIC16F877A, que va a servir para la etapa de control y en donde se programará el software seleccionado para esta implementación.

2) El funcionamiento y operación del motor BLDC y del sensor de efecto Hall.

3) La etapa de potencia compuesta por un puente trifásico compuesto de Mosfet’s tipo N.

ÍNDICE GENERAL

RESUMEN……..…… I

INDICE GENERAL……..…… II

ÍNDICE DE GRAFICOS……..….. III

ÍNDICE DE TABLAS….…… IV

ABREVIATURAS…..…… V

INTRODUCCIÓN……..…… VI

Capítulo 1: GENERALIDADES…….. 1

1.1 Arquitectura de los controladores de velocidad para motores…. 1
1.2 Motivos para emplear controladores de velocidad…… 4
1.3 Fomentar el ahorro de energía mediante el uso de controladores de velocidad……. 5
1.4. Tipos de controladores de velocidad……. 5
1.4.1 Controladores para motores de CC……...……. 6
1.4.2 Controladores por corrientes de Eddy.……...……. 6
1.4.3 Controladores de deslizamiento…… 7
1.4.4 Controladores para motores de CA…….. 7

1.5 Control de motores de bajo costo y de múltiples funciones……. 8

1.5.1 Un sistema de bajo costo……... 9

1.5.2 Un sistema con múltiples funciones……..... 11

Capítulo 2: MARCO TEÓRICO……. 14

2.1 Uso del microcontrolador…… 14

2.1.1 Descripción general del pic18f877a……. 16

3.1 Motor Brushless (sin escobillas)……. 17

2.2.1 Características del los motores brushless…….. 19

4.1 Sensor de efecto hall y su funcionamiento…….. 20

5.1 Puentes H con Mosfet's…… 21

6.1 Modulación de ancho de pulso PWM……... 23

7.1 Inversor Trifásico……. 24

Capítulo 3: DESARROLLO Y DESCRIPCIÓN DEL CONTROLADOR

DE VELOCIDAD…… 26

3.1 Descripción del proyecto……... 26

3.2 Diagrama de bloques del controlador de velocidad……. 27

3.3 Etapa de control……. 28

3.4 Etapa de acoplamiento……. 29

3.5 Etapa de potencia…… 30

3.6 Esquemáticos del controlador de velocidad……... 32

3.6.1 Esquema del circuito de control para el motor BLDC con

sensor…… 32

3.6.2 Esquema del circuito de control para el motor BLDC sin sensor…… 33
3.6.3 Esquema del circuito de acoplamiento y potencia para el motor BLDC con sensor y sin sensor…… 34
3.7. Diagramas de flujo de los programas…….. 36
3.7.1 Diagrama de flujo para el motor BLDC con sensor de efecto hall…….. 36
3.7.2 Diagrama de flujo para el motor BLDC sin sensor de efecto hall…….. 40

Capítulo 4: SIMULACIÓN, IMPLEMENTACIÓN Y PRUEBAS……..... 43

4.1 Simulación del controlador de velocidad del motor BLDC con sensor de efecto hall y sin sensor……..… 43
4.2. Implementación del hardware en el protoboard……..... 52
4.3. Pruebas en el osciloscopio de las etapas de control y de potencia del proyecto……..…… 53

CONCLUSIONES

RECOMENDACIONES

ANEXO A

ANEXO B

ANEXO C

ANEXO D

BIBLIOGRAFÍA

ÍNDICE DE GRÁFICOS

Figura 1. TMCM-160 1-axis BLDC controller/driver module 5A/40V…… 3

Figura 2. Plush 60A Brushless Speed Controller……. 3

Figura 3. Round Mercury es un controlador de motor en lazo cerrado para motores DC y brushless…… 3

Figura 4. Diagrama de bloque de un controlador con MCU…….. 10

Figura 5. Ejemplo de los componentes de un controlador con MCU…… 12

Figura 6. PIC16F877A…….. 16

Figura 7. Motor BLDC con sensor…….. 17

Figura 8. Bobinado del motor BLDC en conexión estrella…… 19

Figura 9. Sensor de efecto Hall……. 20

Figura 10. Puente H con Mosfet's tipo N……. 22

Figura 11. Puente H con Mosfet's tipo P y N…….. 22

Figura 12. PWM……... 23

Figura 13. Conexión de un inversor trifásico…….. 25

Figura 14. Diagrama de bloques…….. 27

Figura 15. Circuito de control del controlador de velocidad…….…… 28

Figura 16. Etapa de acoplamiento…… 29

Figura 17. Disparos en el puente inversor…….. 31

Figura 18. Esquema de la etapa de control para el motor BLDC con sensor de efecto Hall…….…… 33

Figura 19. Esquema de la etapa de control para el motor BLDC sin sensor de efecto Hall…….…… 34

Figura 20. Esquema de la etapa de acoplamiento y etapa de potencia para los dos circuitos de control…….…….. 35

Figura 21. Diagrama de flujo para el motor BLDC con sensor de efecto Hall…….…….. 37

Figura 22. Diagrama de flujo motor BLDC sin sensor-lazo principal …. 41

Figura 23. Señales en el puerto C del microcontrolador entre el par de disparos de la fase A y B…….…….. 44

Figura 24. Señales en el puerto C del microcontrolador entre el par de disparos de la fase B y C…….…….. 45

Figura 25. Señales en el puerto C del microcontrolador entre el par de disparos de la fase A y C.…… 45

Figura 26. Señales del sensor de Efecto Hall…….…….…….…….. 46

Figura 27. Formas de ondas producidas en las bobinas del motor BLDC... 47

Figura 28. Gráfica producida en la fase A del motor BLDC al 50% de la velocidad…….…….…….…….…….…….…….…….…….…… 48

Figura 29. Gráfica producida en la fase A del motor BLDC al 75% de la velocidad…….…….…….…….…….…….…….…….…….…… 48

Figura 30. Gráfica producida en la fase A del motor BLDC al 100% de la velocidad…….…….…….…….…….…….…….…….…….…… 49

Figura 31. Gráficas de ondas producidas en las bobinas del motor BLDC sin sensor ……..…….. 50

Figura 32. Velocidad del motor BLDC sin sensor al 50%..... 50

Figura 33. Velocidad del motor BLDC sin sensor al 75% …… 51

Figura 34. Velocidad del motor BLDC sin sensor al 100%.....…. 51

Figura 35. Circuito del proyecto armado en protoboard……..…….. 52

Figura 36. Forma de onda entre el sensor de efecto Hall A y B…….…. 53

Figura 37. Forma de onda entre el sensor de efecto Hall A y C…….…….. 54

Figura 38. Forma de onda PWM al 27% de la velocidad del motor…… 54

Figura 39. Forma de onda PWM al 50% de la velocidad del motor…… 55

Figura 40. Forma de onda PWM al 100% de la velocidad del motor…….. 55

Figura 41. Forma de onda entre las bobinas A y B…….…….…….. 56

Figura 42. Forma de onda entre las bobinas A y C…….…….…….… 56

Figura A1. Baquelita Circuito de Control con Sensor.....…….…… Anexo D

Figura A2. Baquelita del Circuito de Control sin Sensor.…….…… Anexo D

Figura A3. Baquelita del Circuito de Potencia……..…….. Anexo D

Figura A4. Controlador de Velocidad…….…….…….Anexo D

Figura A5. Perspectiva superior del controlador de velocidad…….. Anexo D

Figura A6. Circuito de control implementado……..... Anexo D

Figura A7. Circuito de potencia implementado……..... Anexo D

ÍNDICE DE TABLAS

Tabla 1. Características del Pic16F887A……. 17

Tabla 2. Códigos según orden de fases de conmutación…….. 38

Tabla 3. Códigos según orden del sensor de efecto Hall en sentido de las manecillas del reloj……. 39

Tabla 4. Códigos según orden del sensor de efecto Hall en sentido contrario de las manecillas del reloj……. 39

ÍNDICE DE ABREVIATURAS

µC, MCU	Microcontrolador
CC	Corriente Continua
CA	Corriente Alterna
PWM	Modulación de ancho de pulso
Vref	Voltaje de referencia
Vdd	Voltaje de alimentación +5V del microcontrolador
GND	Tierra
Vss	Voltaje de alimentación +0V del microcontrolador
BLDC	Motor sin escobillas de corriente continúa
LCD	Display de cristal líquido
TMR0	Módulo Timer 0
V	Voltios
Amp	Amperios
I^2C	Inter-Integrated Circuit (Circuitos Inter-Integrados)
CAN	Controller Area Network
RAM	Random Access Memory - Memoria de acceso aleatorio
ROM	Read Only Memory - Memoria de sólo lectura
EEPROM	(Electrically Erasable Progammable Read Only Memory) Memoria de sólo lectura programable y borrable eléctricamente

INTRODUCCIÓN

En los siguientes capítulos de este informe, se explicará sobre la programación e implementación del controlador de velocidad de la nota de aplicación AN857A de Microchip, que estará basado en la obtención de un valor analógico proporcionado por un potenciómetro, el cual va a ser convertido en un valor digital por el PIC16F877A, y este a su vez será almacenado en el módulo TMR0 con el cual se generará el PWM que serán entregados en el puerto C correspondiente del microcontrolador, dicha señal se necesitará en la parte alta de la etapa de potencia del puente H conformado por los Mosfet's. Y por programación proveeremos también de señales mantenidas en nivel alto a la parte baja de la etapa de potencia.

La etapa de control y la etapa de potencia serán acopladas con la ayuda de tres drivers de potencia, usando los integrados IR2101s que proporcionarán las mismas señales de control a las compuertas de los Mosfet's. Y con la debida programación y los debidos disparos de conducción de los Mosfet's entregaremos una señal DC convertida en AC con la cual nuestro motor BLDC comenzará a girar.

Observaremos también gráficos obtenidos del osciloscopio, el cual nos

mostrará el comportamiento del motor, de las etapas de control y la etapa de potencia. Y la visualización en un LCD el sentido de giro del motor BLDC y una analogía de la velocidad que desarrollará el motor.

CAPÍTULO 1

GENERALIDADES

1.1 ARQUITECTURA DE LOS CONTROLADORES DE VELOCIDAD PARA MOTORES

El Controlador de Velocidad es en un sentido amplio, un dispositivo eléctrico o electrónico empleado para controlar la velocidad giratoria de maquinaria, especialmente de motores.

La maquinaria industrial generalmente es accionada a través de motores eléctricos, a velocidades constantes o variables, pero con valores precisos. No obstante, los motores eléctricos generalmente operan a velocidad constante o cuasi-constante, y con valores que dependen de la alimentación y de las características propias del motor, los cuales no se pueden modificar fácilmente. Para lograr regular la velocidad de los motores, se emplea un controlador especial que recibe el nombre de variador de velocidad.

Hoy en día existe una variedad de controladores de velocidad para la diversidad de motores existentes en el mercado. Existen controladores de velocidad para motores de corriente alterna y para motores de corriente continua, los cuales varían en sus diseños y usos.

Pero la construcción de controladores de velocidad para los motores sin escobillas, o más conocidos como motores BLDC ha acaparado las aplicaciones más automatizadas en el área industrial, como en ventiladores y equipo de aire acondicionado, equipo de bombeo, bandas y transportadores industriales, elevadores, llenadoras, tornos y fresadoras, etc., también en el área automotriz. Y ahora en el mercado del aeromodelismo, donde es muy útil el uso de estos motores son construidos en una escala muy conveniente, en un tamaño a veces muy reducido como para los CD ROM de las computadoras.

Cabe decir que el tamaño de estos controladores de velocidad también se los desarrolla en escalas muy pequeñas, que servirían para aplicaciones como el de uso en juguetes, carros a control remoto o robots. Su uso es amplio y sus precios varían según las aplicaciones.

La arquitectura y programación de los controladores de velocidad se basa de acuerdo a las necesidades de uso, por ejemplo un controlador de velocidad para motores BLDC con sensor de efecto Hall o sin dicho sensor. Controladores de velocidad con conexión de comunicación para visualización que muestren las revoluciones del motor, etc.

Las siguientes gráficas muestran algunos de los controladores de velocidad que existen en el mercado, de los cuales podremos adquirir según nuestras necesidades.

Figura 1. TMCM-160 1-axis BLDC controller/driver module 5A/40V

Figura 2. Plush 60A Brushless Speed Controller

Figura 3. Round Mercury es un controlador de motor en lazo cerrado para motores DC y brushless de 400W

1.2 MOTIVOS PARA EMPLEAR CONTROLADORES DE VELOCIDAD

El control de procesos y el ahorro de la energía son las dos de las principales razones para el empleo de variadores de velocidad. Históricamente, los controladores de velocidad fueron desarrollados originalmente para el control de procesos, pero el ahorro energético ha surgido como un objetivo tan importante como el primero.

Entre las diversas ventajas en el control del proceso proporcionadas por el empleo de controladores de velocidad destacan:

• Operaciones más suaves.

• Control de la aceleración.

• Distintas velocidades de operación para cada fase del proceso.

• Compensación de variables en procesos variables.

• Permitir operaciones lentas para fines de ajuste o prueba.

• Ajuste de la tasa de producción.

• Permitir el posicionamiento de alta precisión.

• Control del Par motor (torque).

1.3 FOMENTAR EL AHORRO DE ENERGÍA MEDIANTE EL USO DE CONTROLADORES DE VELOCIDAD

Un equipo accionado mediante un controlador de velocidad emplea generalmente menor energía que si dicho equipo fuera activado a una velocidad fija constante. Los ventiladores y bombas representan las aplicaciones más llamativas. Por ejemplo, cuando una bomba es impulsada por un motor que opera a velocidad fija, el flujo producido puede ser mayor al necesario. Para ello, el flujo podría regularse mediante una válvula de control dejando estable la velocidad de la bomba, pero resulta mucho más eficiente regular dicho flujo controlando la velocidad del motor, en lugar de restringirlo por medio de la válvula, ya que el motor no tendrá que consumir una energía no aprovechada.

1.4 TIPOS DE CONTROLADORES DE VELOCIDAD

Existen cuatro categorías de controladores de velocidad eléctrico-electrónicos:

- Controladores para motores de CC.
- Controladores de velocidad por corrientes de Eddy.
- Controladores de deslizamiento.
- Controladores para motores de CA conocidos como variadores de frecuencia.

1.4.1 Controladores para Motores de CC

Estos controladores permiten variar la velocidad de motores de corriente continua serie, derivación, compuesto y de imanes permanentes. Aprovechando esta situación se puede controlar la velocidad de un motor de CC: controlando su voltaje terminal, o bien, manipulando el valor de la corriente de campo.

1.4.2 Controladores por Corrientes de Eddy

Un controlador de velocidad por corrientes de Eddy consta de un motor de velocidad fija y un embrague de corrientes de Eddy. El embrague contiene un rotor de velocidad fija (acoplado al motor) y un rotor de velocidad variable, separados por un pequeño entrehierro. Se cuenta, además, con una bobina de campo, cuya corriente puede ser regulada, la cual produce un campo magnético que determinará el par mecánico transmitido del rotor de entrada al rotor de salida. De esta forma, a mayor intensidad de campo magnético, mayor par y velocidad transmitidos, y a menor campo magnético menores serán el par y la velocidad en el rotor de salida. El control de la velocidad de salida de este tipo de controladores generalmente se realiza por medio de lazo cerrado, utilizando como elemento de retroalimentación un tacómetro de CA.

1.4.3 Controladores de Deslizamiento

Este tipo de variadores se aplica únicamente para los motores de inducción de rotor devanado. De esta forma es que puede conseguirse el control de la velocidad en los motores de inducción de rotor devanado. Sin embargo, este tipo de variadores es de menor eficiencia que otros, razón por la cual en la actualidad tiene muy poca aplicación.

1.4.4 Controladores para Motores de CA

Los controladores de frecuencia permiten controlar la velocidad tanto de motores de inducción (asíncronos de jaula de ardilla o de rotor devanado), como de los motores síncronos mediante el ajuste de la frecuencia de alimentación al motor.
Por ello es que este tipo de controladores manipula la frecuencia de alimentación al motor a fin de obtener el control de la velocidad de la máquina. Estos variadores mantienen la razón Voltaje/ Frecuencia (V/Hz) constante entre los valores mínimo y máximos de la frecuencia de operación, con la finalidad de evitar la saturación magnética del núcleo del motor y además porque el hecho de operar el motor a un voltaje constante por encima de una frecuencia dada disminuye el par del motor y la capacidad del mismo para proporcionar potencia constante de salida.

1.5 CONTROL DE MOTORES DE BAJO COSTO Y DE MÚLTIPLES FUNCIONES

Las soluciones de bajo costo basadas en microcontroladores (MCU), que utilizan relés electromecánicos o control de ángulo de desfasamiento con tiristores, fueron utilizadas de manera exitosa por los fabricantes de equipos originales para los controles de motor de velocidad variable. Estas soluciones son adecuadas para determinados tipos de motores y aplicaciones que necesitan control mínimo de velocidad o en los casos donde el rendimiento del motor y las fluctuaciones torsionales no son los requerimientos principales.

Las regulaciones de consumo de energías actuales y futuras demandan artefactos con mayor ahorro energético. Con el fin de cumplir con los requisitos energéticos, se utilizan nuevas tecnologías de motores y topologías de control alternativas. Una topología de control comúnmente utilizada es el inversor, que consiste en extraer voltaje de la línea de corriente continua (CC) y luego convertirla para generar un voltaje de corriente alterna (CA). Finalmente, se generan voltajes de determinadas amplitudes y frecuencias mediante el uso de una técnica de modulación especial, denominada Modulación de la Magnitud del Impulso (PWM). Los voltajes de salida del inversor se generan a través de transistores de energía por conmutación. Gracias a la topología del inversor, es posible implementar nuevos algoritmos de control para distintos tipos de motores, es decir, control de motores de inducción voltio por Hertz o control de velocidad

de motores de corriente continua sin escobilla (BLDC).

La solución planteada para las aplicaciones de control de motor apunta a motores de velocidad variable que requieren electrónica de energía por conmutación, tales como los inversores. Los fabricantes de equipos originales que busquen mejor rendimiento y funcionalidad del motor, además de mantener el bajo costo, deberían tener en cuenta esta solución basada en microcontroladores (MCU) con módulos especiales para controlar el motor, tales como módulos PWM.

1.5.1 Un sistema de bajo costo

Con el objetivo de lograr la funcionalidad del motor en arquitecturas MCU, hoy en día algunas operaciones que llevan demasiado tiempo se realizan mediante el hardware interno MCU con módulos integrados especiales. Integrar un módulo PWM en un MCU reduce considerablemente el número de componentes externos, reduciendo de esta manera el costo del sistema. Analizaremos algunas ventajas de los módulos PWM integrados para tratar de explicar cómo el costo total del sistema se mantiene bajo mediante la eliminación de circuitos externos.

La mayor ventaja del módulo PWM para el control del motor es que este tiempo muerto se inserta automáticamente a través del MCU, eliminando de esta manera la necesidad de circuitos de sincronización externos en el impulsor de la compuerta.

Los módulos PWM cuentan con varios canales para accionar las topologías del motor. Como muestra la Figura 1, se implementó una topología de inversor

trifásico para accionar un motor de corriente continua sin escobilla. Como tiene la ventaja de generar seis salidas PWM junto con el módulo dedicado para dicho fin, el MCU se conecta directamente a cada impulsor de compuerta de hasta seis transistores de potencia, eliminando de esta manera cualquier componente externo que realice enmascaramiento y multiplexión.

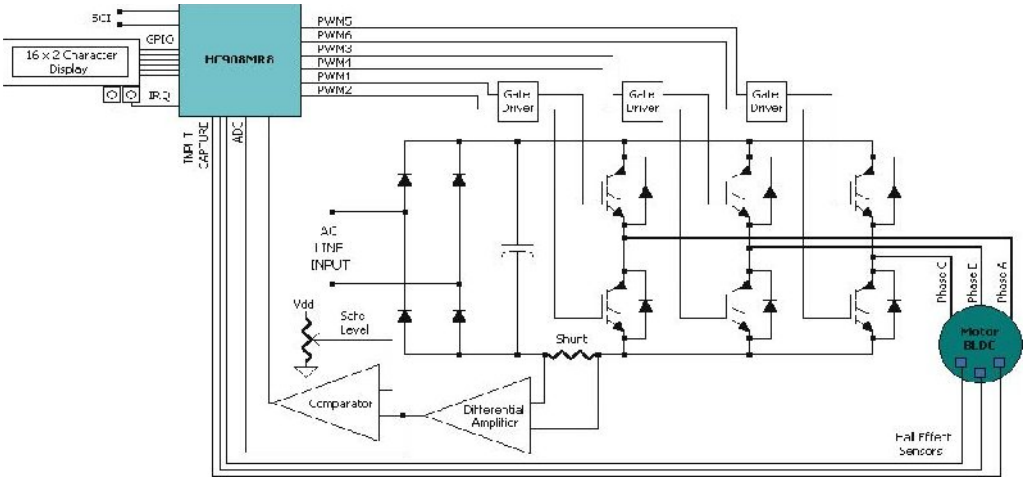


Figura 4. Diagrama de bloque de un controlador con MCU

Un requerimiento común para las aplicaciones de alta potencia es contar con una barrera de aislamiento entre la parte de control y los dispositivos electrónicos de energía. Por lo general, la solución consiste en tener aisladores ópticos en el circuito de accionamiento de compuerta, pero la parte del controlador del aislador óptico necesita una corriente para accionar los diodos emisores de luz (LED). Además, el módulo PWM ayuda a accionar los LED directamente desde las salidas del MCU, eliminando de esta manera cualquier transistor o circuito intermedio que realice la misma tarea.

Otro requerimiento para las aplicaciones de control del motor es la tolerancia a fallos del sistema. Estos fallos pueden ocurrir debido a sobrecorrientes y sobrevoltajes, que obligan al MCU a apagar todos los dispositivos de potencia para proteger el motor y los dispositivos de potencia. Las funciones de fallo se incluyen en el módulo PWM, donde el sistema se protege mediante el uso de clavijas de entrada especiales para condiciones de fallo. Esta capacidad reduce la complejidad del accionamiento de compuerta, ya que el impulsor de compuerta no necesita tener funcionalidad de desconexión para proteger el sistema contra los fallos.

Otras funciones del MCU como por ejemplo la tecnología Flash también representan una reducción de costos para algunas aplicaciones que requieren almacenamiento de datos no volátil, ya que se puede emular el flash interno como una memoria EEPROM para almacenar información.

1.5.2 Un sistema con múltiples funciones

Con la utilización del módulo interno MCU para generar señales PWM de tolerancia a fallos sin consumir demasiados recursos de la CPU le permitirá al sistema incluir otras funciones en el producto final. Es posible utilizar los módulos de comunicación MCU como puertos seriales para monitorear variables globales del sistema, por lo que el proceso de depuración es mucho más fácil. La capacidad de observar las variables mientras se controla el motor permite diseñar

y validar algoritmos nuevos para mejorar el rendimiento del artefacto. Otra función es la implementación de controladores de circuito cerrado en el software, tales como los controles de velocidad Proporcional-Integral (PI). La Figura 2 muestra un ejemplo de una lavadora con un motor de corriente continua sin escobilla (BLDC). La figura muestra los módulos internos MCU, así como también algunos módulos de software para controlar la velocidad del motor. En este ejemplo, la velocidad de referencia del motor BLDC se toma de una tabla interna, lo que permite que diferentes perfiles de movimiento del agitador evalúen y validen nuevos ciclos de lavado. Por ejemplo, el requerimiento para un nuevo ciclo de lavado puede ser que se dañen menos prendas fabricadas con telas especiales.

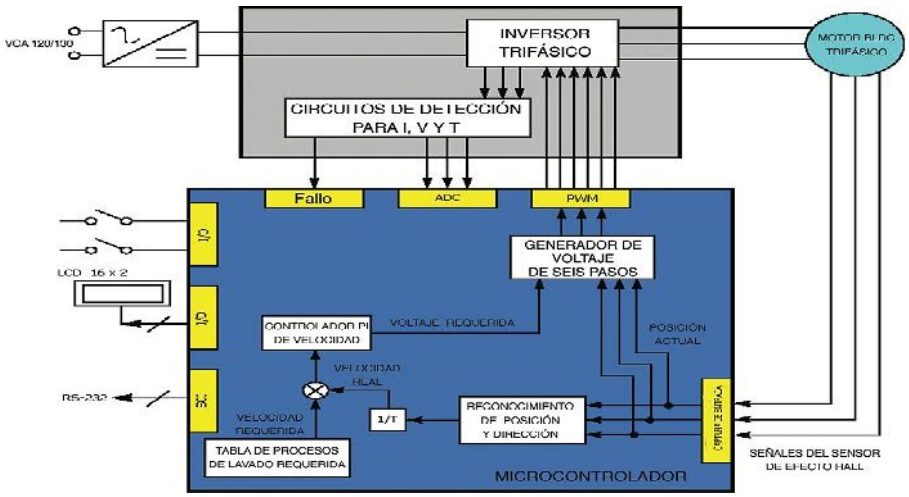


Figura 5. Ejemplo de los componentes de un controlador con MCU

La nueva tecnología Flash puede ofrecer varias funciones adicionales para los artefactos. Permite la programación en circuito del MCU una vez que se haya

instalado la placa electrónica en el artefacto, haciendo que se pueda ampliar la capacidad del software y que el fabricante también pueda corregir fallos de software en el momento del testeo. A los artefactos basados en esta tecnología podrá introducirse mejoras y ampliaciones como un servicio de postventa al cliente.

Los diseñadores de los artefactos, que agregaron valor al sistema total, incluyendo indicadores LED, pantallas con caracteres y botones, utilizaron otros módulos MCU, tales como entradas y salidas (I/O) de uso general e interrupciones externas.

CAPÍTULO 2

MARCO TEÓRICO

En este capítulo se describe el funcionamiento y características teóricas de cada una de las partes que estarán implementadas en este proyecto como ciertas teorías de necesidad para la programación, mostrando las ventajas y desventajas de sus componentes.

2.1 USO DEL MICROCONTROLADOR

Explicado mediante términos sencillos, podemos definir a un microcontrolador como un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.

Un microcontrolador puede disponer de un generador de reloj integrado y una pequeña cantidad de memoria RAM, ROM/EPROM/EEPROM, significando que para hacerlo funcionar, todo lo que se necesita son unos pocos

programas de control y un cristal de sincronización. Los microcontroladores disponen generalmente también de una variedad de dispositivos de entrada/salida, como convertidores de analógico a digital, temporizadores y buses de interfaz serie especializados, como I^2C y CAN, entre otros. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados.

Microchip es la empresa que fabrica los microcontroladores PIC. En los últimos tiempos esta familia de microcontroladores ha revolucionado el mundo de las aplicaciones electrónicas. Tienen una facilidad de uso y programación, que junto a las inmensas posibilidades de E/S que brindan han conquistado a programadores y desarrolladores. Su principal ventaja (y según sus detractores la principal desventaja) es su carácter general, la flexibilidad que les permite ser empleados en casi cualquier aplicación. Otras familias de microcontroladores son más eficaces en aplicaciones específicas.

La familia PIC se divide en cuatro gamas, gamas que podemos llamar mini, baja, media y alta. Las principales diferencias entre estas gamas radica en el número de instrucciones y su longitud, el número de puertos y funciones, lo cual se refleja en el encapsulado, la complejidad interna y de programación, y en el número de aplicaciones.

2.1.1 Descripción General del PIC18F877A

Pertenece a la denominada gama media es la más variada y completa de los PIC. Abarca modelos con encapsulado desde 18 hasta 68 pines (ver figura 6 Pic16F877A), cubriendo varias opciones que integran abundantes periféricos. En esta gama sus componentes añaden nuevas prestaciones a las que poseían los de la gama baja, haciéndoles más adecuados en las aplicaciones complejas. Poseen comparadores de magnitudes analógicas, convertidores A/D, puertos serie y diversos temporizadores.

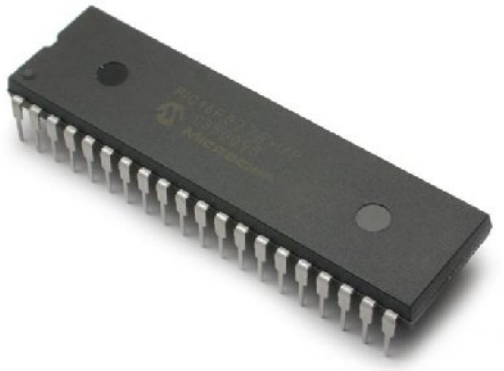


Figura 6. PIC16F877A

El repertorio de instrucciones es de 35, de 14 bits cada una y compatible con el de la gama baja. También dispone de interrupciones y una pila de 8 niveles que permite el anidamiento de subrutinas. A continuación se detalla en la siguiente tabla características del Pic16F887A.

Nombre del parametro	Valor
Program Memory Type	Flash
Program Memory (KB)	14
CPU Speed (MIPS)	5
RAM Bytes	368
Data EEPROM (bytes)	256
Digital Communication Peripherals	1-A/E/USART, 1-MSSP(SPI/I2C)
Capture/Compare/PWM Peripherals	2 CCP
Timers	2 x 8-bit, 1 x 16-bit
ADC	8 ch, 10-bit
Comparators	2
Temperature Range (C)	-40 to 125
Operating Voltage Range (V)	2 to 5.5
Pin Count	40

Caracteristicas
2 PWM 10-bit
256 Bytes EEPROM data memory
ICD
25mA sink/source per I/O
Self Programming
Parallel Slave Port

Tabla 1. Características del Pic16F887A

2.2 **MOTOR BRUSHLESS (SIN ESCOBILLAS)**

Los motores de corriente continua sin escobillas (BLDC) son uno de los tipos de motores que más popularidad ha ganado en los últimos años. Actualmente, los motores BLDC se emplean en sectores industriales tales como: Automóvil, Aeroespacial, Consumo Médico, Equipos de Automatización e Instrumentación.

Figura 7. Motor BLDC con sensor

La palabra brushless se puede traducir como "sin escobillas", las escobillas son los elementos que hacen contacto en el colector de un motor común. En los motores de DC más pequeños, son de una aleación de cobre y en motores más grandes son de un compuesto a base de carbón.
Estos motores carecen de colector y escobillas o carbones. Entonces ¿cómo funcionan?. Es simple, en vez de funcionar en DC funcionan en AC, la mayoría se alimentan con una señal trifásica, esta señal idealmente debería ser sinusoidal, pero en la práctica son pulsos, haciendo que la señal sea un continua pulsante o bien una continua con mucho componente de AC sin embargo se los clasifica como de DC porque al igual que los motores comunes tienen imanes permanentes.
Estos imanes son atraídos por la polaridad de un campo magnético generado en las bobinas, las cuales como decíamos reciben pulsos en un patrón especifico. Si queremos que el motor gire más rápido, simplemente hacemos girar el campo magnético secuencial a mayor velocidad. O lo que sería lo mismo a aumentar la frecuencia de los pulsos.

En el motor existen tres circuitos electromagnéticos conectados en un punto común. Cada circuito electromagnético se divide en el centro, permitiendo así el imán permanente del rotor a moverse en el medio del campo magnético inducido. La mayoría de los motores BLDC tienen un bobinado trifásico con topología de conexión en estrella.

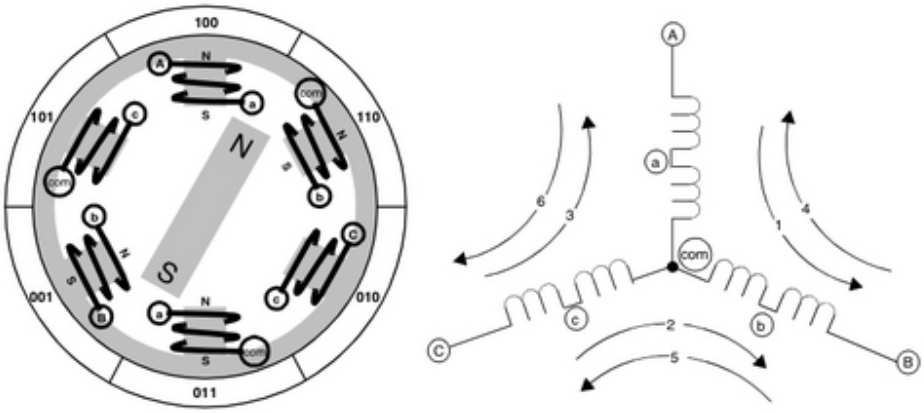


Figura 8. Bobinado del motor BLDC en conexión estrella

2.2.1 Características del los Motores Brushless

Los motores BLDC tienen la característica de que no emplean escobillas en la conmutación para la transferencia de energía; en este caso, la conmutación se realiza electrónicamente. Esta propiedad elimina el gran problema que poseen los motores eléctricos convencionales con escobillas, los cuales producen rozamiento, disminuyen el rendimiento, desprenden calor, son ruidosos y requieren una sustitución periódica y, por tanto, un mayor mantenimiento. Los motores BLDC tienen muchas ventajas frente a los motores DC con escobillas y frente a los motores de inducción. Algunas de estas ventajas son:

- Mejor relación velocidad-par motor.
- Mayor respuesta dinámica.
- Mayor eficiencia.
- Mayor vida útil.

- Menor ruido.
- Mayor rango de velocidad.

Además, la relación par motor-tamaño es mucho mayor, lo que implica que se puedan emplear en aplicaciones donde se trabaje con un espacio reducido. Por otra parte, los motores BLDC tienen dos desventajas, que son las siguientes:

- Tienen un mayor coste.
- Requieren un control bastante más complejo.

2.3 **SENSOR DE EFECTO HALL Y SU FUNCIONAMIENTO**

Los sensores de efecto Hall se utilizan para medir velocidades de rotación o detectar la posición de un determinado elemento. Su principal ventaja es que pueden ofrecer datos fiables a cualquier velocidad de rotación. Y sus inconvenientes son la mayor complejidad y precio con respecto a un sensor inductivo.

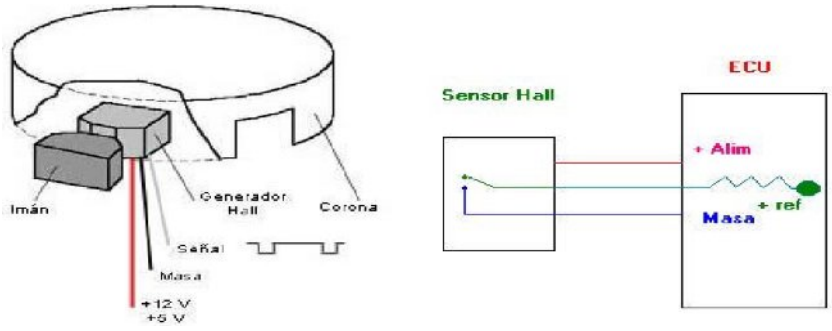


Figura 9. Sensor de efecto Hall

El sensor de efecto Hall se basa en la tensión transversal de un conductor que está sometido a un campo magnético. Colocando un voltímetro entre dos puntos transversales de un cable se puede medir esa tensión. Para ello hay que hacer circular por el cable una intensidad fija y acercar un imán. Los electrones que pasan por el cable se verán desplazados hacia un lado. Entonces aparece una diferencia de tensión entre los dos puntos transversales del cable. Al separar el imán del cable, la tensión transversal desaparece. Para poder utilizar la tensión transversal es necesario amplificarla, porque su valor es muy reducido.

Para comprobar el funcionamiento de un sensor Hall basta verificar el valor de la tensión de alimentación y la variación de la tensión en la señal de salida cuando alguna ventana de la corona permite el flujo del campo magnético.

2.4 **PUENTES H CON MOSFET'S**

Aquí una breve explicación del funcionamiento de los puentes H con Mosfet's, explicando rápidamente cada una de sus partes. A pesar de la gran ventaja de los Mosfet's sobre los demás tipos de transistor, existen algunas desventajas en su uso. Como saben el MOSFET es un dispositivo que controla corriente con una entrada de voltaje. La mayoría de los Mosfet's (salvo Mosfet's especiales) usan voltajes base- surtidor de entre 10 a 12 o 15V para poder lograr su saturación, si se aplica una diferencia de voltaje menor pues el MOSFET se comportara como un transductor lineal, es decir la salida

de corriente será proporcional al voltaje aplicado y esto no es lo que queremos, lo que necesitamos es el MOSFET como interruptor de potencia.

Existen 2 tipos de arquitecturas de puentes H con Mosfet's.

- Todos los Mosfet's son canal N (ver figura 10).

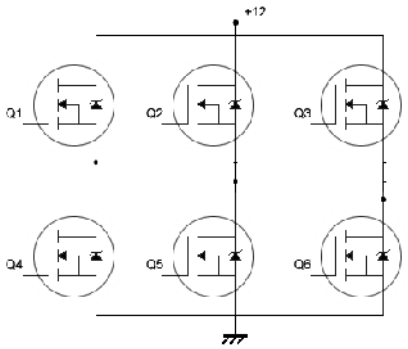


Figura 10. Puente H con Mosfet's tipo N

- Los Mosfet's superiores son canal P y los inferiores canal N (ver figura 11).

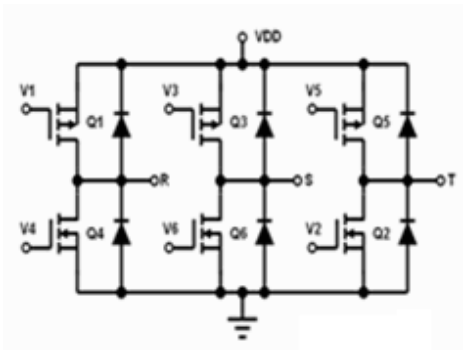


Figura 11. Puente H con Mosfet's tipo P y N

Cada una de estas tiene ciertas ventajas y desventajas. Cuando se tiene todos los Mosfet's canal N se consigue un mejor control del motor al tener tiempos de propagación similares. Pero el encendido de los disparadores de base se complica.

En cambio cuando se tiene un circuito que haga uso de Mosfet's de diferente tipo (canal N y canal P) tienen un tiempo de propagación el cual no necesariamente es el mismo para Mosfet's de canal N y canal P, a decir verdad los canal P son mucho más lentos que los canal N) otra desventaja de esta topología es el hecho de que no podremos efectuar un control de 4 cuadrantes del motor (Torque vs Velocidad).

2.5 MODULACIÓN DE ANCHO DE PULSO PWM

PWM son las siglas de "Pulse Width Modulation" o "Modulación por Anchura de Pulsos", el PIC incorpora esta función hardware que se puede utilizar para conseguir una salida analógica a partir de un señal digital a través de la variación del valor eficaz una señal, esto se consigue modulando la anchura de pulso de un tren de onda cuadrada.

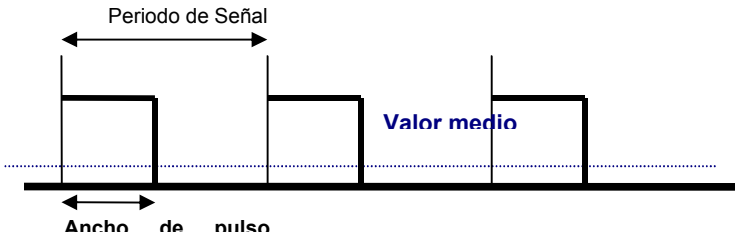


Figura 12. PWM

Para un pulso más ancho el valor eficaz de la señal es mayor que para un pulso más estrecho, por lo tanto variando la anchura del tren de pulsos de la señal digital se puede conseguir una señal cuyo valor eficaz (valor medio) varíe de forma deseada.

El movimiento de motor eléctrico se consigue mediante la variación continua de un campo eléctrico o magnético, dependiendo del tipo de motor, para obtener así un campo rotatorio, esta variación continua se consigue alimentando adecuadamente los embobinados del rotor o del estator, según el tipo de motor, y variando dicha corriente de alimentación se consigue variar la velocidad del motor, un método utilizado para lograr este propósito es mediante PWM.

Mediante PWM se controla la corriente de alimentación del motor. Variando el porcentaje de tiempo de la señal rectangular en estado alto o bajo, es decir variando la anchura de pulso de la señal varía la potencia entregada al motor. Controlando esta relación se logra variar la velocidad del motor con mucha precisión.

2.6 **INVERSOR TRIFÁSICO**

La función de un Inversor trifásico es generar energía eléctrica trifásica de corriente alterna a partir de una fuente de energía de corriente continua, con magnitudes y frecuencias deseadas. Se constituye principalmente por

dispositivos electrónicos de potencia, que trabajan como interruptores operando en corte y saturación con una secuencia apropiada para obtener tres tensiones de salida simétricas y balanceadas. El controlador es otro componente fundamental en la constitución del convertidor, es el que genera las señales de encendido y apagado de los dispositivos semiconductores y garantiza su buen comportamiento.

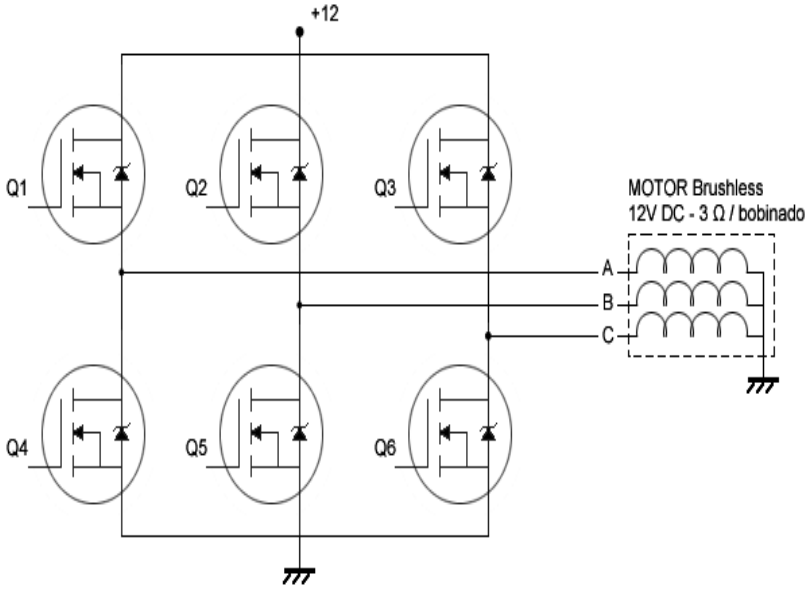


Figura 13. Conexión de un inversor trifásico

Básicamente, se trata de 3 inversores monofásico en puente combinado en un solo sistema que usa 6 transistores, la numeración de los cuales está acorde a la secuencia de activación de los mismos (ver figura 13).

CAPÍTULO 3

DESARROLLO Y DESCRIPCIÓN DEL CONTROLADOR DE VELOCIDAD

3.1 DESCRIPCIÓN DEL PROYECTO

Consiste en un controlador PWM con microcontrolador, en este caso se usa el PIC16F877A, en donde vamos a variar su velocidad con la ayuda de un potenciómetro y vamos a visualizar el valor de la velocidad del motor en un LCD. Para controlar el motor necesitamos hacer la interacción entre el motor y el controlador para ello utilizamos un inversor trifásico.

Cada transistor es controlado mediante la técnica PWM. De esta manera se obtiene un voltaje alterno trifásico, desfasado 120 grado entre fases, con un frecuencia y tensión variables que dependerán de los valores establecidos.

Este trabajo se divide en tres etapas fundamentales: la primera es la forma de generar las señales de control para cada uno de los transistores del puente inversor, la segunda es establecer conexión entre las señales de control y el circuito inversor, por último la tercera consiste en la etapa de potencia en el cual estará conectado el motor BLDC.

3.2 DIAGRAMA DE BLOQUES DEL CONTROLADOR DE VELOCIDAD

El siguiente diagrama de bloques muestra la disposición de los componentes a utilizar en el controlador de velocidad para el motor BLDC con sensor de efecto Hall y sin sensor, estructurando así los esquemáticos de los circuitos a implementarse en este proyecto.

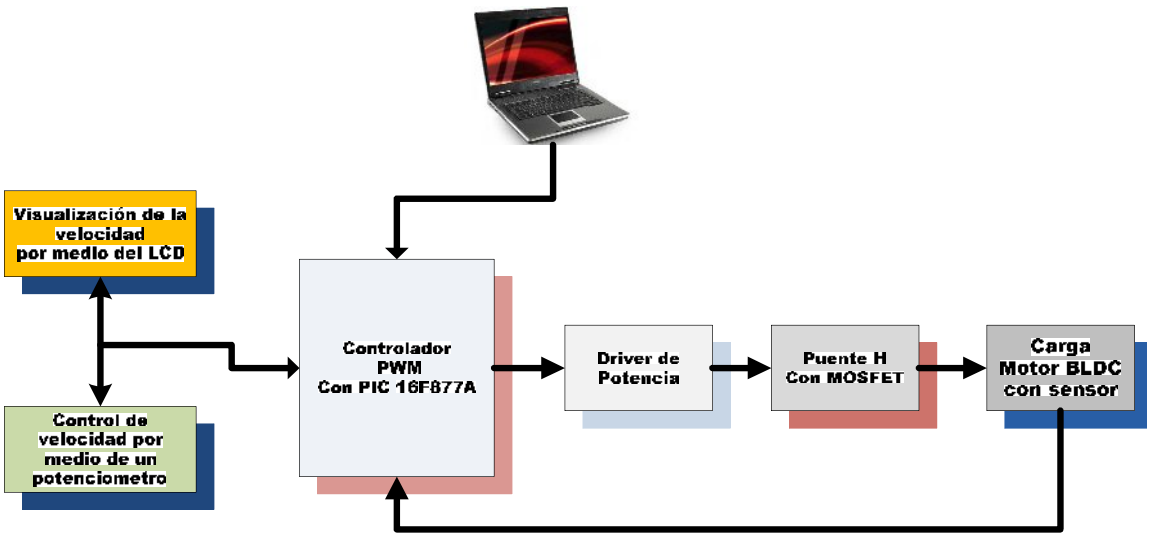


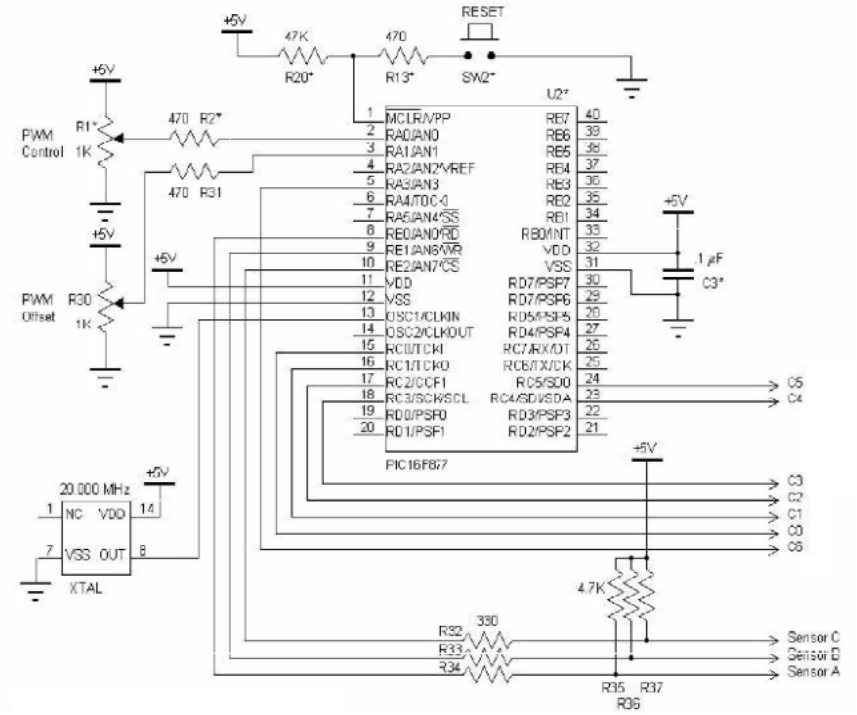
Figura 14. Diagrama de bloques

3.3 ETAPA DE CONTROL

Esta etapa es la encargada de generar las señales digitales PWM, las inserciones de tiempos muertos para evitar cortos circuitos entre pares complementarios, el aislamiento mediante los drivers IRF2101s (figura 16) entre la etapa digital y la etapa de potencia, así como el manejo de los interruptores Mosfet's. Para generar los pulsos de control de los interruptores de potencia se utiliza PIC16F877A (figura 15) el cual es alimentando a 5 Vdc, mediante la programación que se describe más adelante.

Además ofrecerá la visualización del sentido de giro del motor BLDC y también de la velocidad máxima y mínima mediante un display de cristal liquido.

Figura 15. Circuito de control del controlador de velocidad



3.4 ETAPA DE ACOPLAMIENTO

El manejador es un sistema capaz de gobernar eficazmente la conducción y no conducción del interruptor de potencia (MOSFET IRF530A) partiendo de las órdenes que llegan del circuito de control (PIC16F877A).

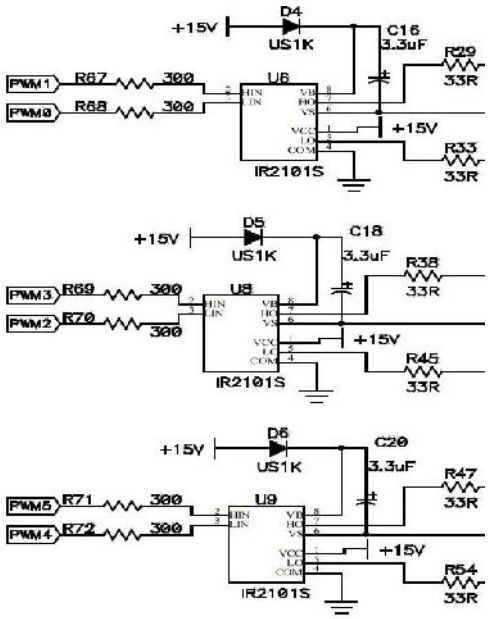


Figura 16. Etapa de acoplamiento

Los circuitos de potencia y los circuitos de control no manejan los mismos niveles de tensión y corriente, será necesario utilizar una etapa de

acoplamiento de voltajes, corrientes e impedancias, para ello se utilizó los drivers IRF2101 (figura 16) para que el PIC16F877A y los Mosfet's puedan interactuar.

3.5 ETAPA DE POTENCIA

El inversor diseñado se alimenta con 5 voltios. A la salida se tendrá una línea trifásica con amplitud y frecuencia variables. La magnitud de voltaje y frecuencia dependerá de la velocidad que se quiera para el motor trifásico, la cual estará totalmente con el usuario.

El puente inversor está construido básicamente por seis transistores Mosfet's. De acuerdo a la figura 17. Cada transistor puede conducir durante 120 grado, con la respectiva señal PWM. Cuando se enciende el transistor Q1, la terminal "R" está conectada con la terminal positiva del voltaje DC de entrada. Cuando se enciende el transistor Q5, la terminal "R" se lleva a la terminal negativa de la fuente de CC. Los interruptores de cualquier rama del inversor, no se pueden encender en forma simultánea, porque se produciría un corto a través del enlace con la fuente de voltaje DC de alimentación. De igual modo, para evitar estados indefinidos y en consecuencia voltajes indefinidos de corriente alterna de salida, los interruptores de cualquier rama del inversor no pueden apagarse en forma

simultánea, porque se producirían voltajes que dependen de la polaridad de la corriente de línea correspondiente.

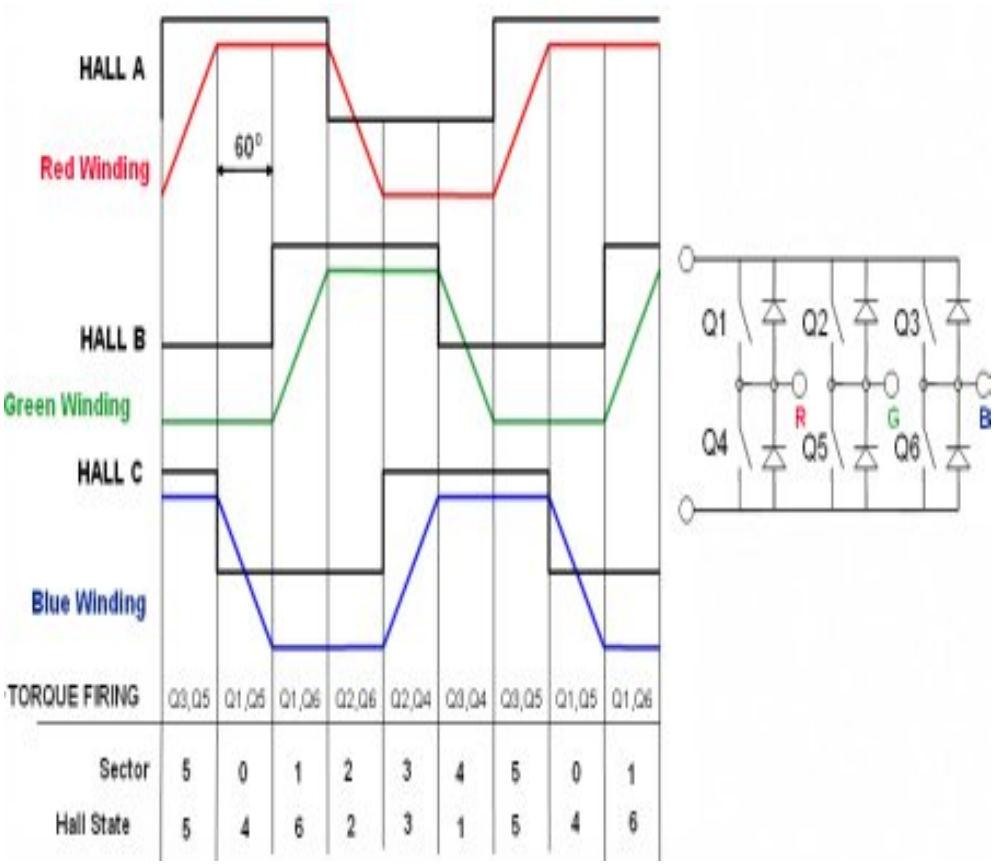


Figura 17. Disparos en el puente inversor

El interruptor de potencia usado en la construcción de este inversor trifásico es el transistor MOSFET (IRF530A) el cual por su característica cumple nuestra expectativa este transistor tienen tres terminales identificadas como: compuerta,

drenador y surtidor, son dispositivos controlados por voltaje aplicado entre compuerta y surtidor, la corriente que circula por la compuerta es casi nula debido a que esta unida mediante un aislante por lo que tendrá una impedancia muy grande, sin embargo durante las conmutaciones si existirá circulación de corriente por la compuerta. La corriente que entrega el transistor MOSFET depende del voltaje aplicado entre compuerta y el surtidor, y no de la corriente que circula por la compuerta como ocurre con los transistores bipolares. Debido a sus características de construcción los MOSFET nos permiten que las conmutaciones se realicen en tiempos muy cortos.

3.6 ESQUEMÁTICOS DEL CONTROLADOR DE VELOCIDAD

Los siguientes gráficos muestran las conexiones de los diferentes elementos electrónicos del controlador de velocidad para motores trifásicos BLDC con sensor de efecto Hall y sin sensor.

3.6.1 Esquema del circuito de control para el motor BLDC con sensor

Este esquemático está conformado por los componentes electrónicos de la parte de control de este proyecto, donde se encuentran los dispositivos como el PIC16F877A que es el elemento que va ejecutar el programa instalado para el control del motor BLDC con sensor, un LCD donde visualizaremos la velocidad del motor un

potenciometro para ajustar la velocidad, una botón para la dirección del motor y otros elementos pasivos de necesidad para la implementación del controlador.

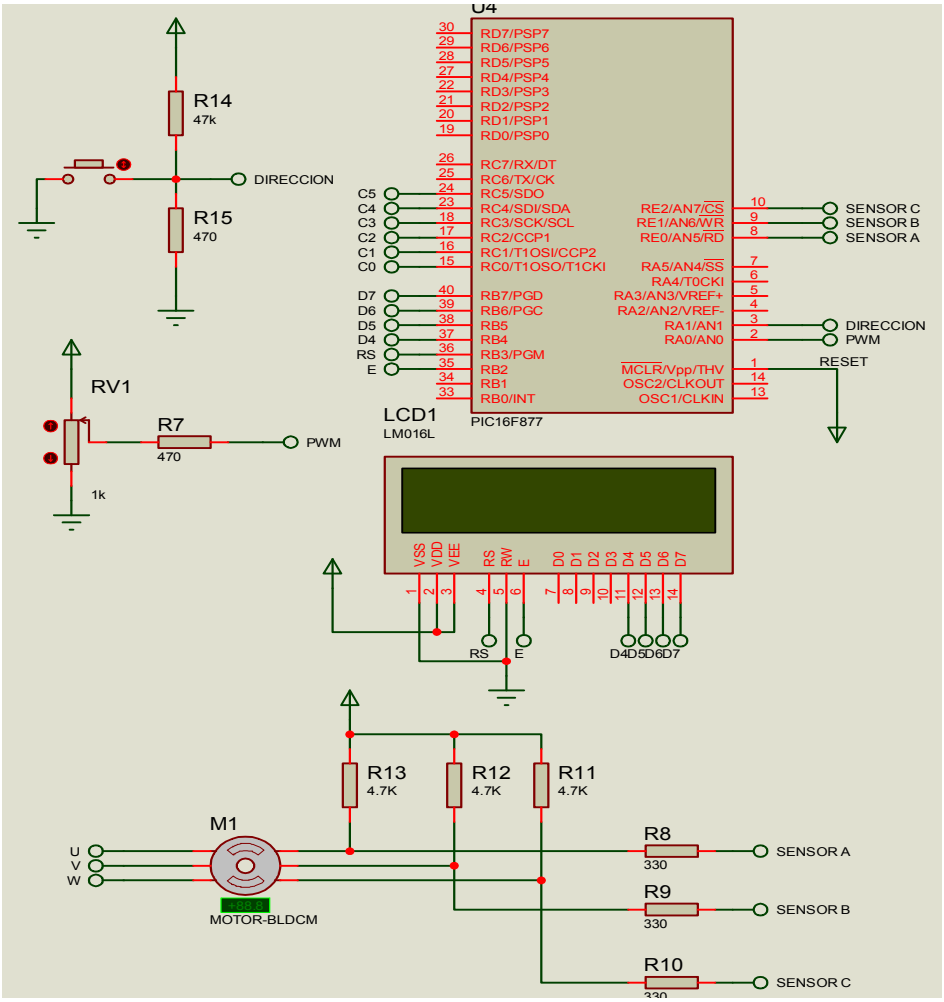


Figura 18. Esquema de la etapa de control para el motor BLDC con sensor de efecto Hall

3.6.2 Esquema del circuito de control para el motor BLDC sin sensor

Los componentes de este circuito tienen las mismas funcionabilidad que del esquemático anterior, el caso es que este control va a ser utilizado para operar el motor BLDC sin sensor.

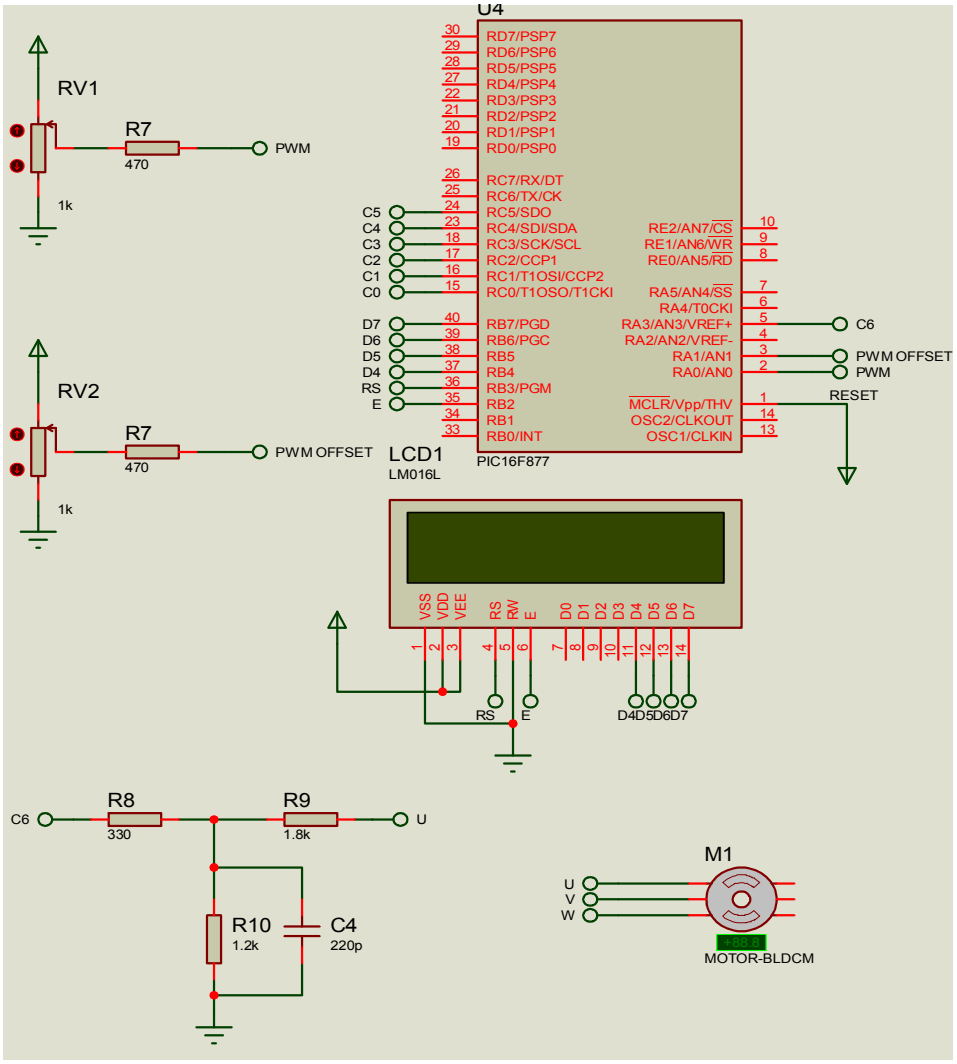


Figura 19. Esquema de la etapa de control para el motor BLDC sin sensor de efecto Hall

3.6.3 Esquema del circuito de acoplamiento y potencia para el motor BLDC con sensor y sin sensor

El siguiente esquemático corresponde al circuito de acoplamiento y al circuito de potencia de este controlador de velocidad, donde constan las conexiones de los drivers IR2101 y del puente inversor trifásico tipo puente H, además de los terminales donde se va a conectar el motor BLDC. Cabe recalcar que este circuito de potencia será utilizado para los dos circuitos de control mencionados anteriormente.

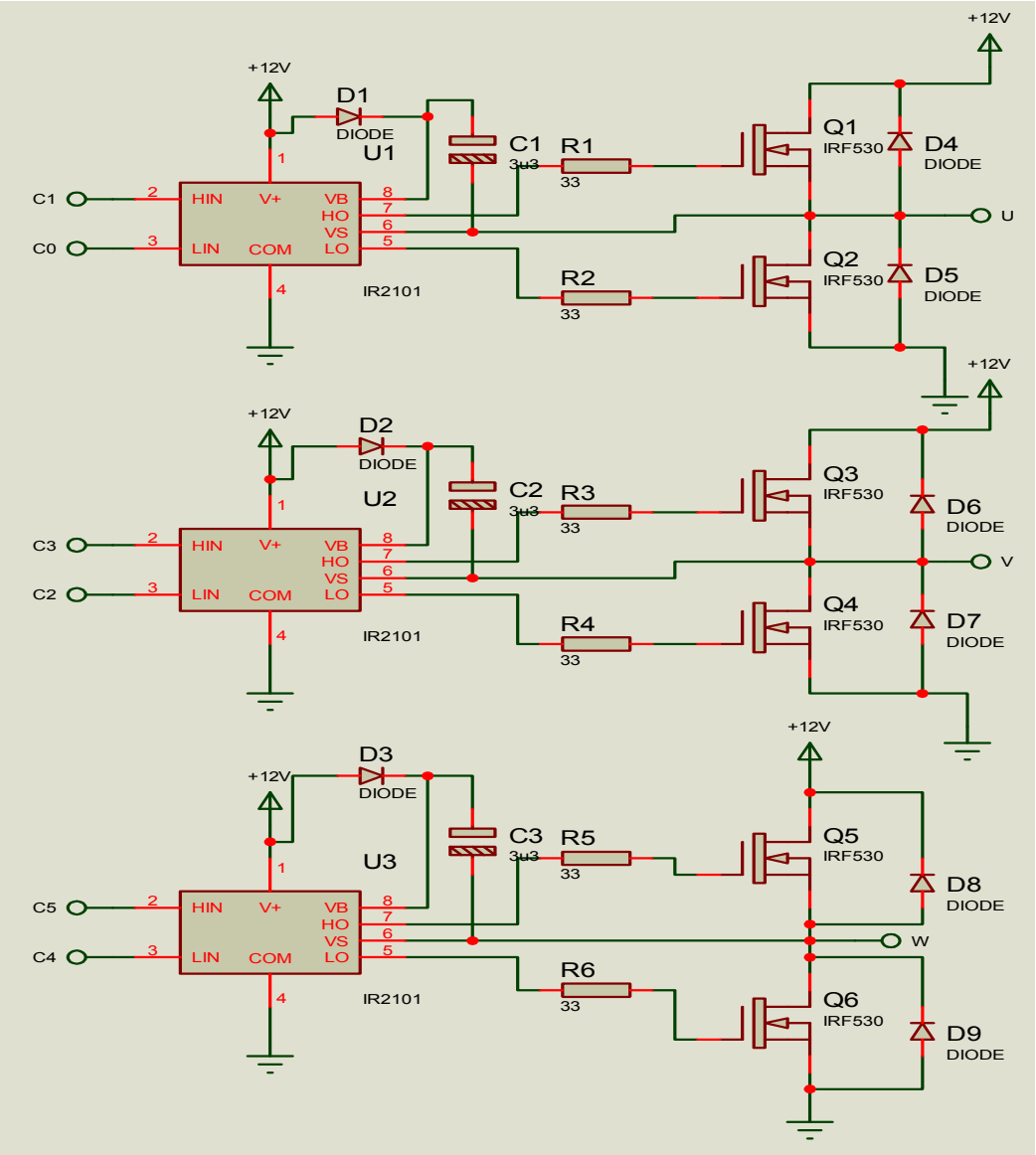


Figura 20. Esquema de la etapa de acoplamiento y etapa de potencia para los dos circuitos de control

3.7. DIAGRAMAS DE FLUJO DE LOS PROGRAMAS

3.7.1 Diagrama de flujo para el motor BLDC con sensor de efecto Hall

El siguiente diagrama de flujo muestra los pasos de la programación del controlador de velocidad. Usaremos el convertidor analógico digital (ADC) del PIC16F877 para leer un potenciómetro y el uso de la lectura de voltaje como la velocidad relativa de control de entrada. Tan solo 8 bits del ADC se utilizan, por lo que nuestro control de velocidad tendrá 256 niveles, el cual será almacenado en un registro llamado ADC. Deseamos que la velocidad relativa corresponda a la relativa posición del potenciómetro. La velocidad del motor es directamente proporcional a la tensión aplicada, de forma lineal desde 0% a 100% se traducirá en un control lineal de velocidad de 0% a 100% del máximo de RPM. El ancho del impulso se determina de forma continua al añadir el resultado ADC en el módulo Timer0, para determinar cuando los conductores deben estar encendidos o apagados. Para obtener una frecuencia PWM de 10 kHz del Timer0 debe estar en ejecución a 256 veces mayor que la tasa, o 2,56 MHz El valor mínimo para pre escalar Timer0 es de 1:2, por lo que necesita una frecuencia de entrada de 5,12 MHz La frecuencia de entrada al Timer0 es FOSC / 4. Esto requiere una FOSC de 20,48 MHz Un cristal de 20 MHz está lo suficientemente cerca, así que el resultado de la frecuencia PWM será de 9,77 kHz.

El PIC16F877A leerá la posición del sensor de efecto Hall para la correcta conmutación en las salidas del PIC, el cual será almacenado, además de comprobar el bit de dirección de giro del motor accionado por una botonera.

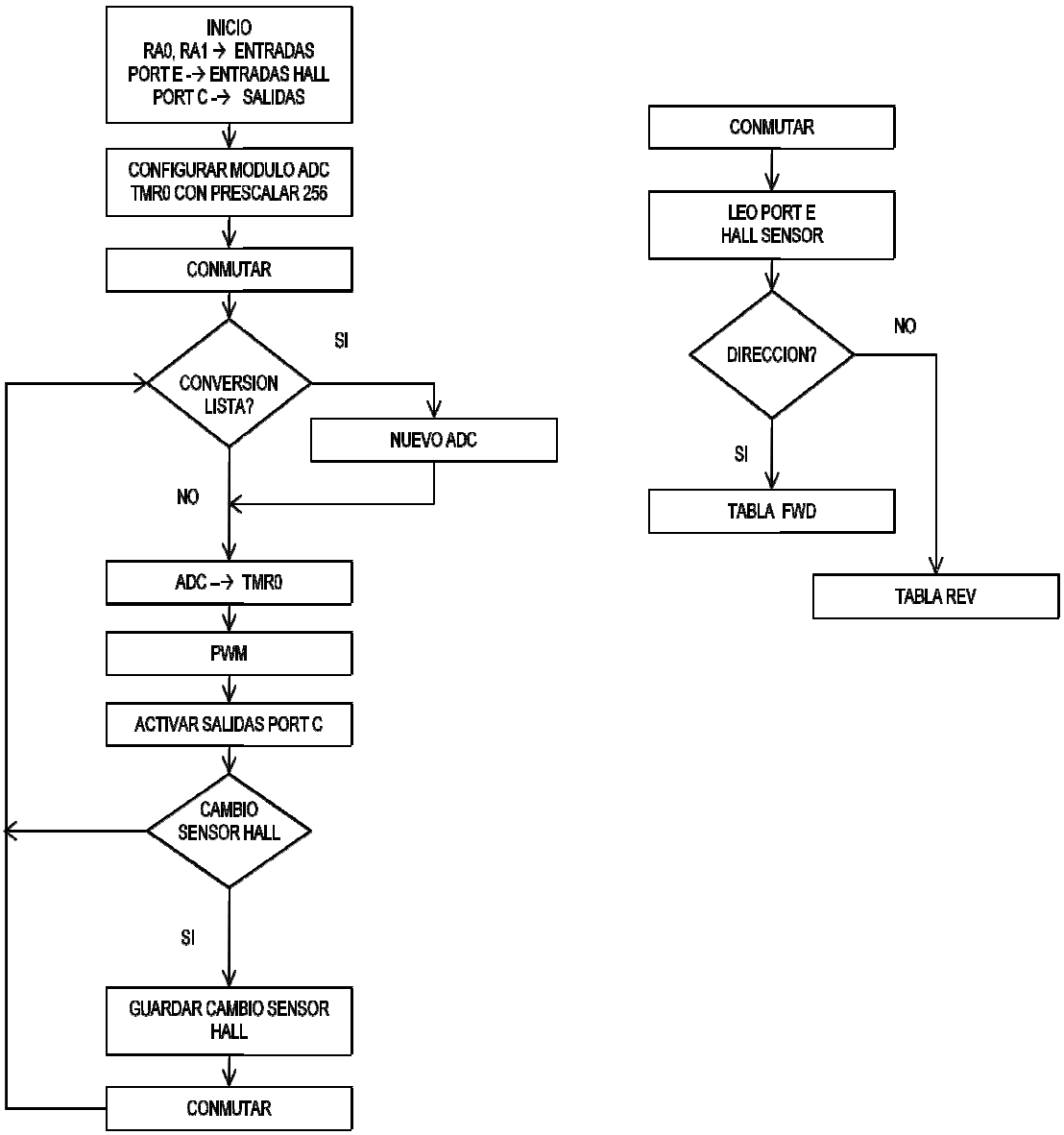


Figura 21. Diagrama de flujo para el motor BLDC con sensor de efecto Hall

La conmutación consiste en el estado de entrada del sensor con la correspondiente unidad de estado de salida en PORTC. Esto se logra con una tabla de estado. Las entradas de sensor formarán un puntero de la tabla de desplazamiento, y la lista de posibles códigos de la unidad de salida será la tabla de estado. El código de desarrollo se realizará en el PIC16F877A. El PORTC asignado arbitrariamente como el puerto de la unidad de motor y PORTE como el puerto de entrada del sensor. Cada fase requiere de dos pines, uno para el impulso en alta y otro para el impulso en baja, es decir, seis pines de PORTC se utilizarán para control de los seis MOSFETS. Cada sensor requiere un pin, así tres pines de PORTE se utilizará para leer el estado actual del motor. Las siguientes tablas muestran los códigos respectivos de los estados del sensor de efecto Hall y el código de los pines de PORTC que deberán ser accionados.

Pin	RE2	RE1	RE0	RC5	RC4	RC3	RC2	RC1	RC0
Phase	Sensor C	Sensor B	Sensor A	C High Drive	C Low Drive	B High Drive	B Low Drive	A High Drive	A Low Drive
1	1	0	1	0	0	0	1	1	0
2	1	0	0	1	0	0	1	0	0
3	1	1	0	1	0	0	0	0	1
4	0	1	0	0	0	1	0	0	1
5	0	1	1	0	1	1	0	0	0
6	0	0	1	0	1	0	0	1	0

Tabla 2. Códigos según orden de fases de conmutación

La tabla 3 muestra los códigos de disparos para los drivers del motor, en dirección de rotación de acuerdo con el sentido de las manecillas del reloj, esta tabla será la utilizada para la programación en el movimiento del rotor en el sentido mencionado.

Pin	RE2	RE1	RE0	RC5	RC4	RC3	RC2	RC1	RC0
Phase	Sensor C	Sensor B	Sensor A	C High Drive	C Low Drive	B High Drive	B Low Drive	A High Drive	A Low Drive
6	0	0	1	0	1	0	0	1	0
4	0	1	0	0	0	1	0	0	1
5	0	1	1	0	1	1	0	0	0
2	1	0	0	1	0	0	1	0	0
1	1	0	1	0	0	0	1	1	0
3	1	1	0	1	0	0	0	0	1

Tabla 3. Códigos según orden del sensor de efecto Hall en sentido de las manecillas del reloj

La tabla 4 se construyó mediante el cambio de todos los estados altos y bajos de PORTC de la tabla 3. De acuerdo con la tabla de estado construida, hará que el motor gire en sentido contrario a las manecillas del reloj.

Pin	RE2	RE1	RE0	RC5	RC4	RC3	RC2	RC1	RC0
Phase	Sensor C	Sensor B	Sensor A	C High Drive	C Low Drive	B High Drive	B Low Drive	A High Drive	A Low Drive
/6	0	0	1	1	0	0	0	0	1
/4	0	1	0	0	0	0	1	1	0
/5	0	1	1	1	0	0	1	0	0
/2	1	0	0	0	1	1	0	0	0
/1	1	0	1	0	0	1	0	0	1
/3	1	1	0	0	1	0	0	1	0

Tabla 4. Códigos según orden del sensor de efecto Hall en sentido contrario de las manecillas del reloj

En el anexo B se encuentra el programa en assembler para el circuito de control para el motor BLDC con sensor, con el cual va hacer programado en el PIC 16F877A.

3.7.2 Diagrama de flujo para el motor BLDC sin sensor de efecto Hall

El programa utiliza dos potenciómetros como entradas de control de velocidad. Un potenciómetro, lo vamos a llamar el potenciómetro de PWM Offset, que está directamente relacionado con el PWM del ciclo de trabajo. El segundo potenciómetro es el PWM, se utiliza para proporcionar una compensación de PWM determinado por el potenciómetro de PWM Offset. Una conversión analógica a digital de los potenciómetros PWM produce un número entre 0 y 255. Este resultado, se convierte en el umbral del ciclo de trabajo PWM, y controla la unidad.

En la siguiente figura, mostramos el diagrama de flujo del lazo principal del programa del motor BLDC sin sensor que provee Microchip en la nota de aplicación AN857A. El resto del diagrama de flujo de este programa se encuentra en el Anexo A de este informe, así como se encontrará en el anexo C el programa en ASM de este diagrama de flujo, el cual será el programado en el PIC16F877A para el circuito de control sin sensor.

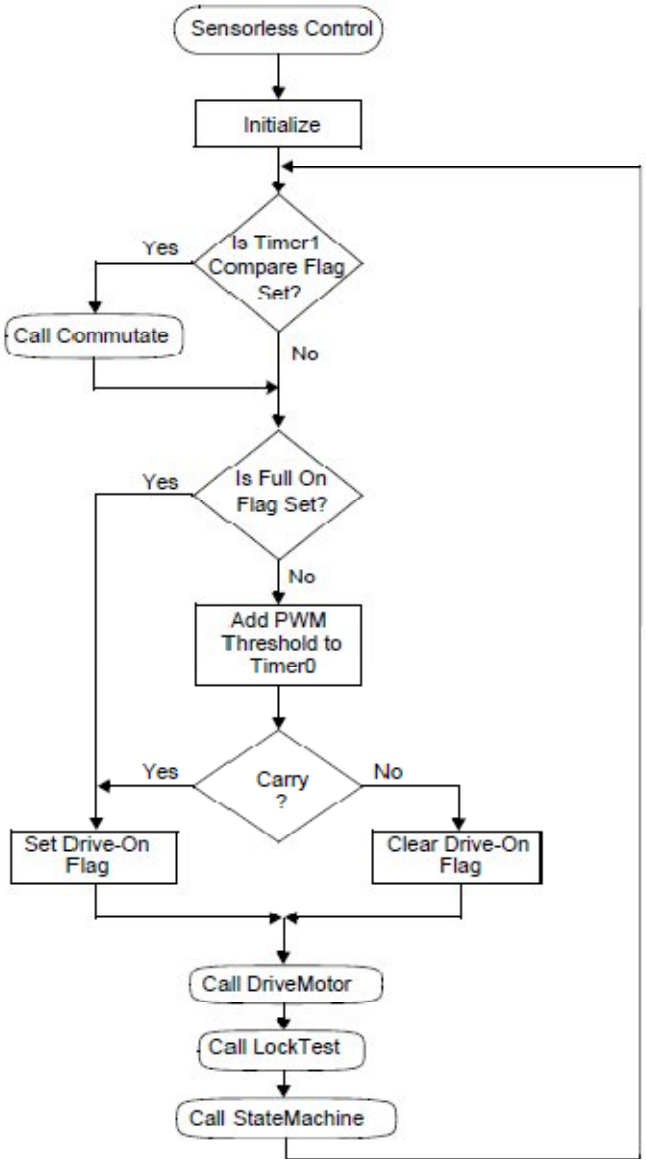


Figura 22. Diagrama de flujo motor BLDC sin sensor-lazo principal.

Podemos medir la BEMF y la tensión aplicada, luego compararlas entre sí para determinar la posición del rotor. Podemos variar el voltaje aplicado con el PWM y el control de la velocidad del motor. Algunas de las mediciones de los eventos

deben ser perfectamente sincronizadas. La conversión analógica a digital debe cambiar de un canal a otro y permitir el suficiente tiempo para la adquisición de datos. Algunos eventos deben suceder rápidamente con una latencia mínima, esto incluye el PWM y la conmutación.

Podemos lograr todo con un bucle principal, que llama a una tabla de estado. El bucle principal se encargará de PWM y la conmutación, y la tabla de estado leerá los dos potenciómetros, el pico de tensión aplicada y la BEMF en las dos ocasiones en que la fase A del motor este en estado flotante.

CAPÍTULO 4

SIMULACIÓN, IMPLEMENTACIÓN Y PRUEBAS

4.1 SIMULACIÓN DEL CONTROLADOR DE VELOCIDAD DEL MOTOR BLDC CON SENSOR DE EFECTO HALL Y SIN SENSOR

Las siguientes imágenes muestran la simulación del controlador de velocidad para el motor BLDC con sensor y sin sensor. Observaremos el comportamiento del PWM al manipular los potenciómetros de cada uno de los circuitos de control, así como también las formas de ondas que se generan en los drivers de dichos circuitos y en las fases del motor.

Las primeras figuras mostradas a continuación son las simuladas para el motor BLDC con sensor de efecto Hall. Observamos cómo opera su sistema de control y los disparos que ejecuta el programa en las salidas del puerto C del microcontrolador así como el comportamiento del sensor de efecto Hall.

Las figuras 23, 24 y 25 muestran las simulaciones de las señales que genera el microcontrolador en el puerto C, donde se puede observar el desfasamiento entre las señales C1, C3 y C5 la cual es de 120°, de igual manera entre C0, C2 y C4. Además vemos como se aplica la señal PWM solamente en las salidas C1, C3 y C5 y un pulso mantenido para las demás.

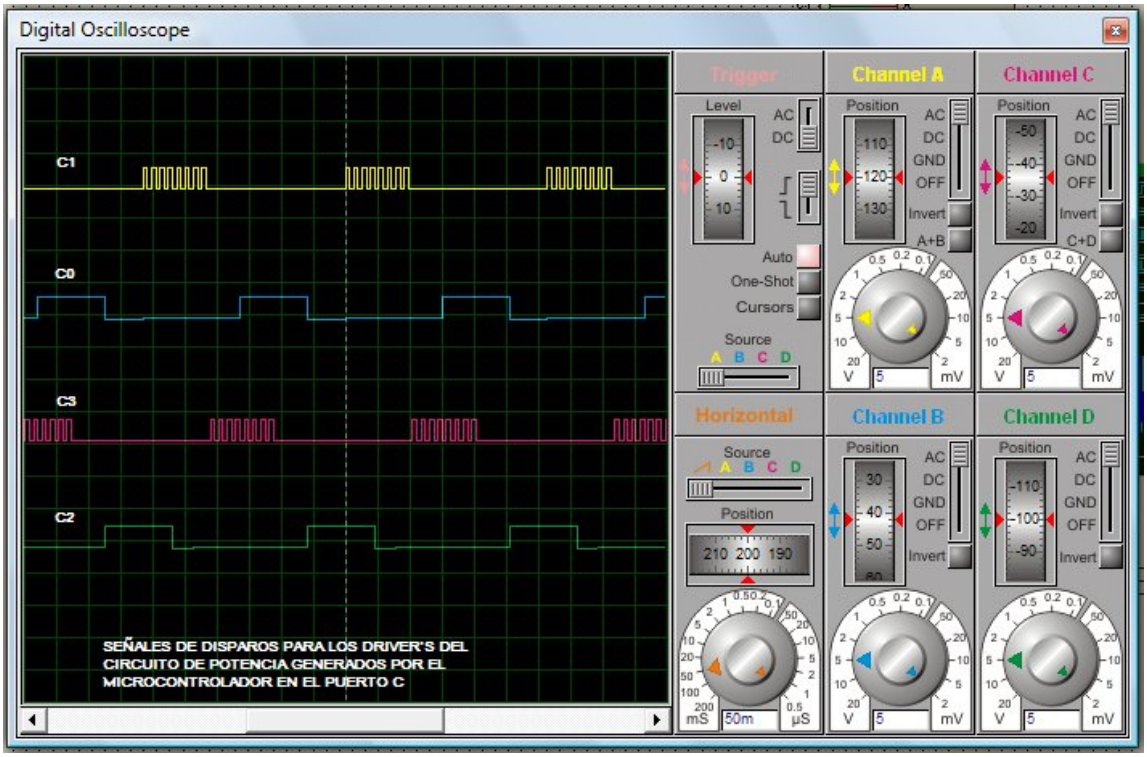


Figura 23. Señales en el puerto C del microcontrolador entre el par de disparos de la fase A y B

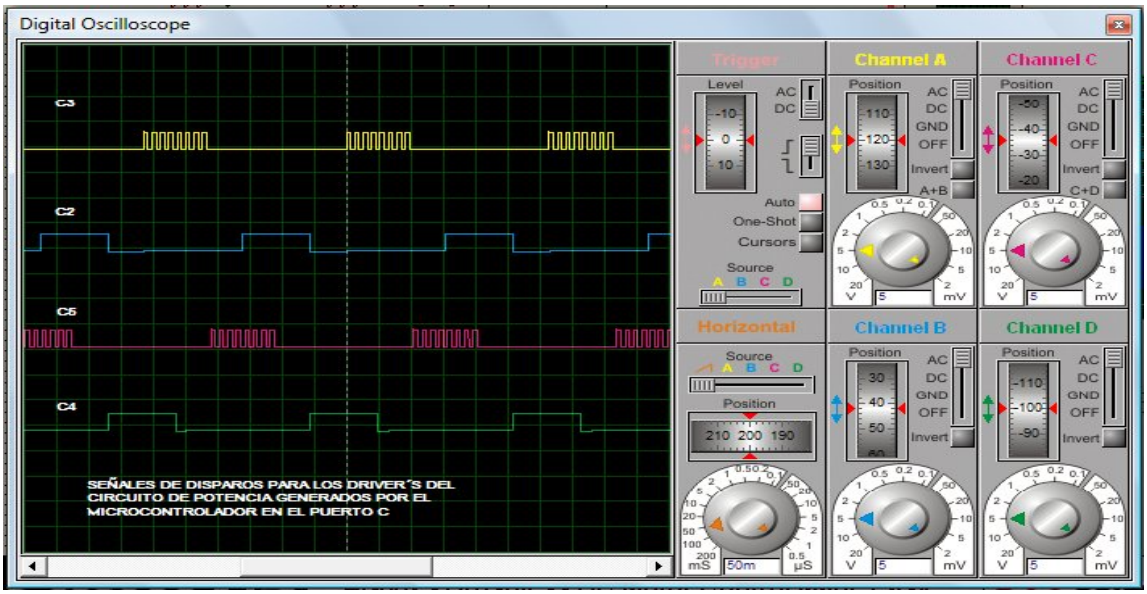


Figura 24. Señales en el puerto C del microcontrolador entre el par de disparos de la fase B y C

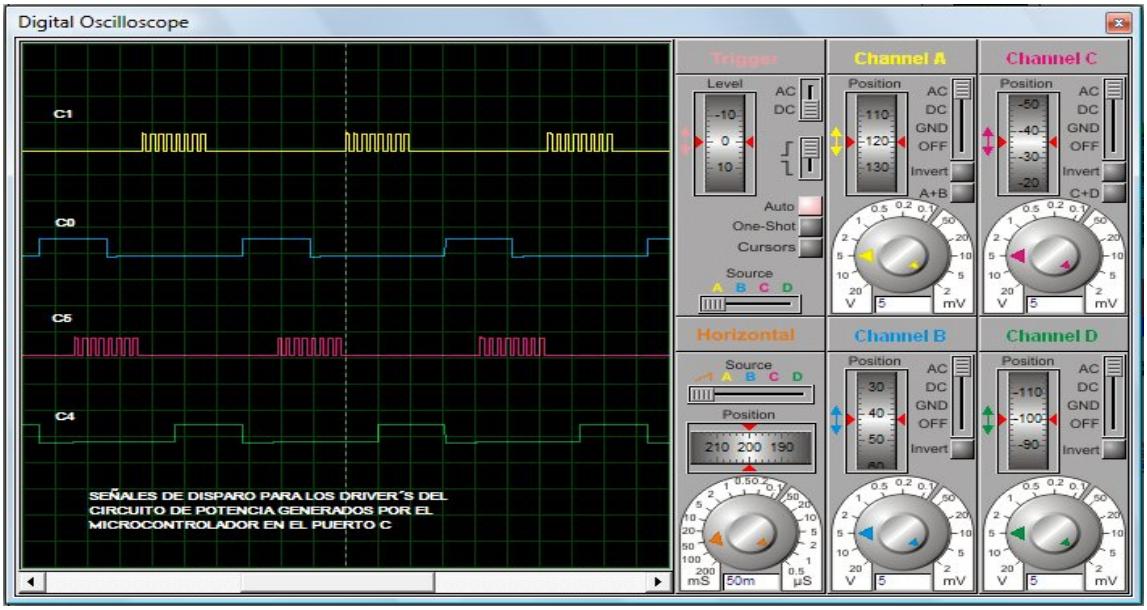


Figura 25. Señales en el puerto C del microcontrolador entre el par de disparos de la fase A y C

La presente figura es la generada por el sensor de efecto Hall del motor BLDC, en donde podemos observar que tiene un desfasamiento entre cada una de las señales de 120°.

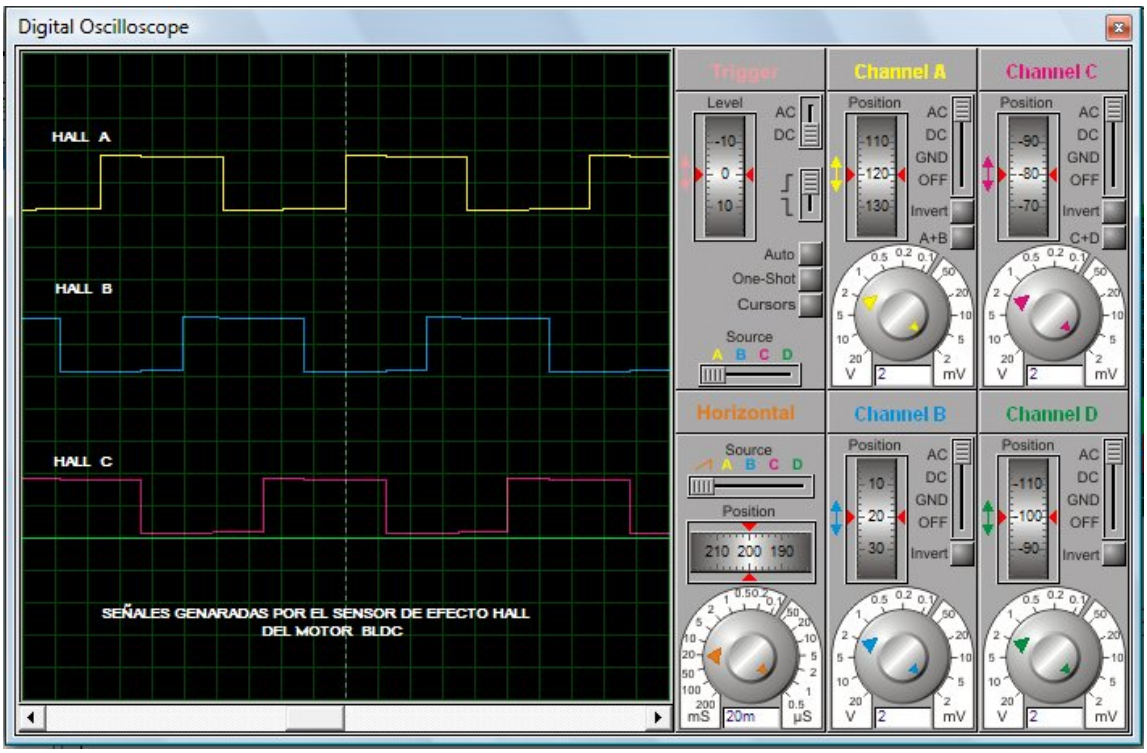


Figura 26. Señales del sensor de Efecto Hall

La figura 27 es el producto de las señales que se generan en las bobinas del motor BLDC, las cuales tienen un desfasamiento de 120° entre cada una de las fases.

Cabe recalcar que estas formas de ondas son las producidas al poner al potenciómetro en un 100%, dándole la máxima velocidad al motor.

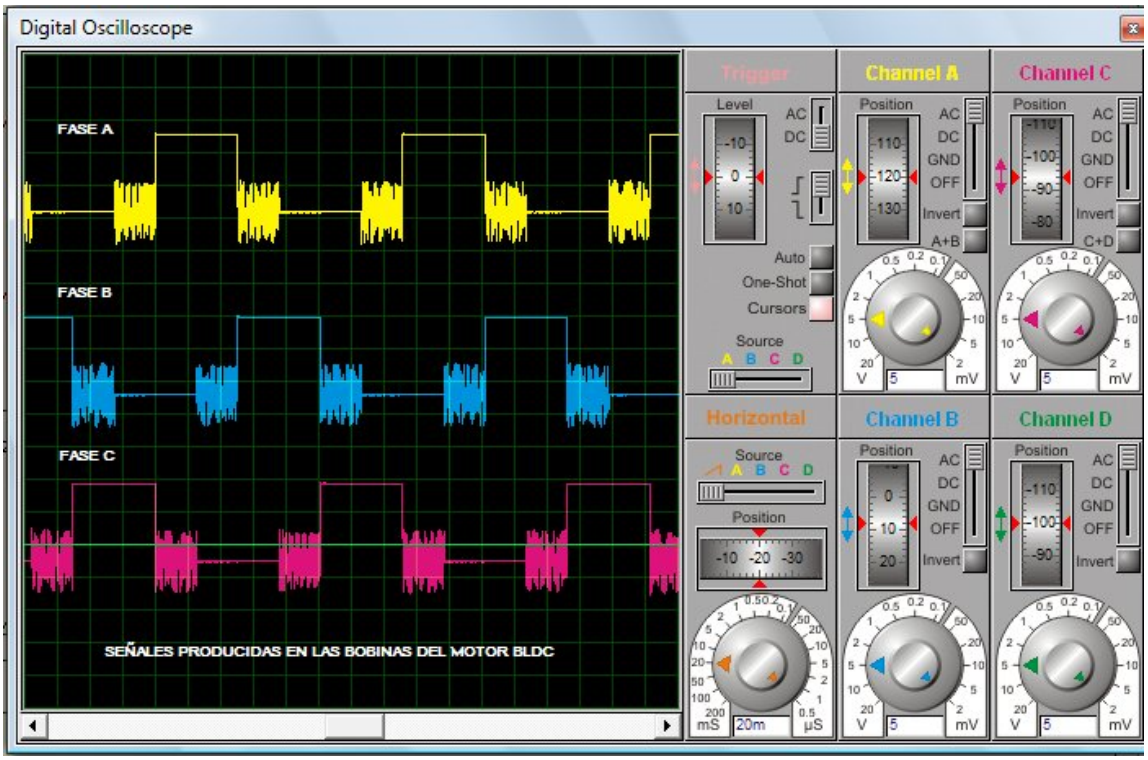


Figura 27. Formas de ondas producidas en las bobinas del motor BLDC

Las siguientes gráficas (figuras 28, 29 y 30) muestran el cambio de frecuencia del motor BLDC debido la variación de velocidad en el motor. La simulación está dada para una velocidad al 50%, 75% y 100% del PWM al variar el potenciometro del circuito de control.

La obtención de los gráficos en la simulación esta medida solamente en la bobina de la fase A del motor BLDC con sensor de efecto Hall.

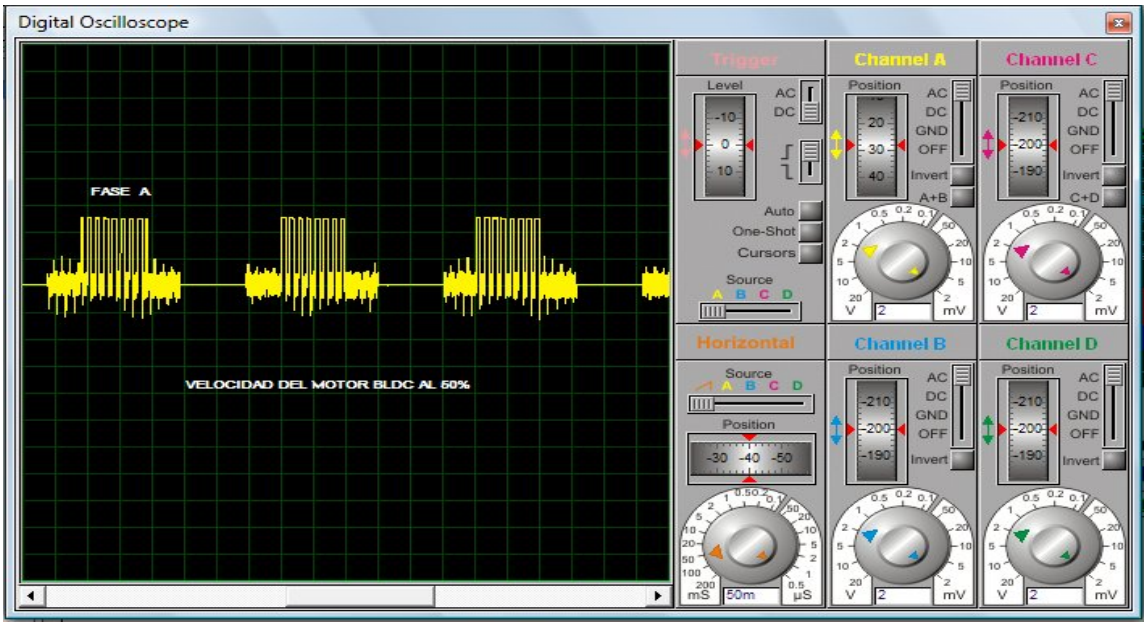


Figura 28. Gráfica producida en la fase A del motor BLDC al 50% de la velocidad

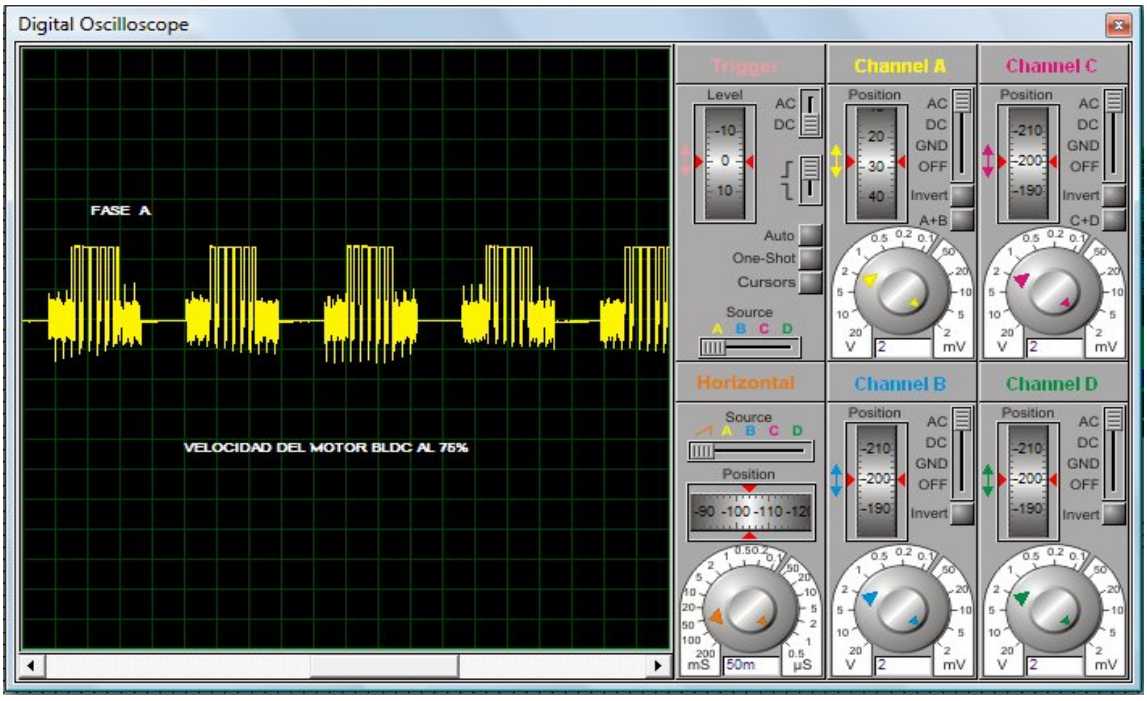


Figura 29. Gráfica producida en la fase A del motor BLDC al 75% de la velocidad

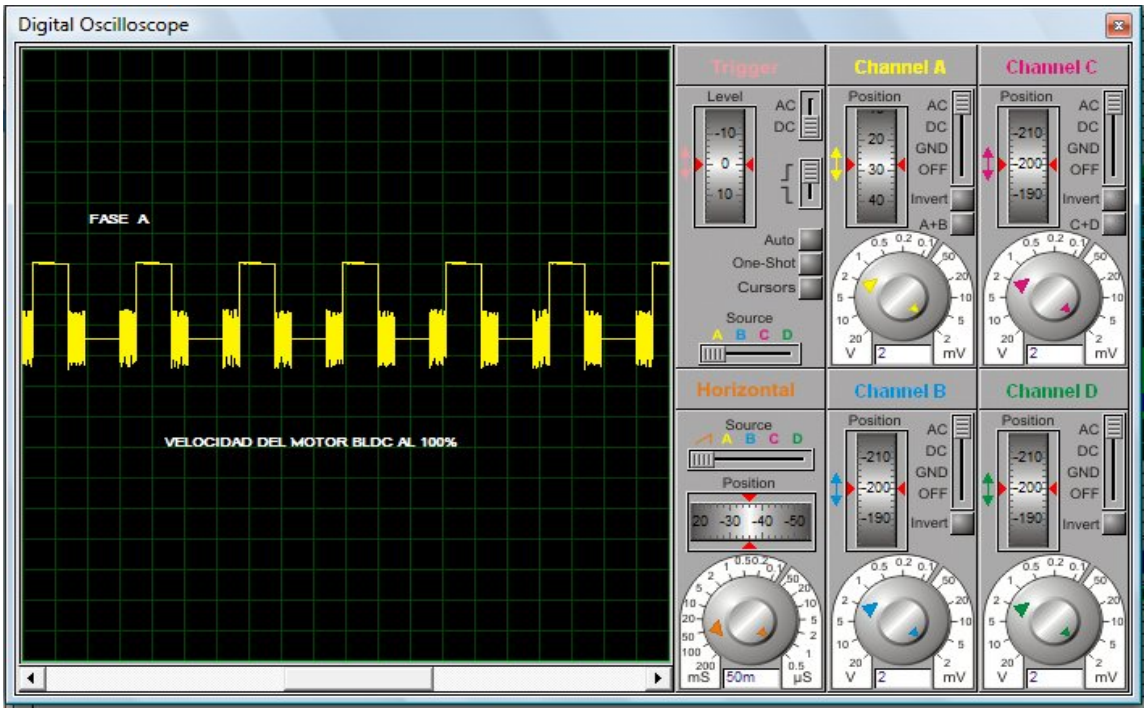


Figura 30. Gráfica producida en la fase A del motor BLDC al 100% de la velocidad

Las siguientes gráficas son las correspondientes a la simulación del motor BLDC sin sensor. Se muestra en la figura 31 las formas de onda obtenidas en las fases del motor BLDC las cuales están desplazadas 120° entre sí, además de la señal que recibe el microcontrolador en el pin 5 correspondiente a RA3 del puerto A.

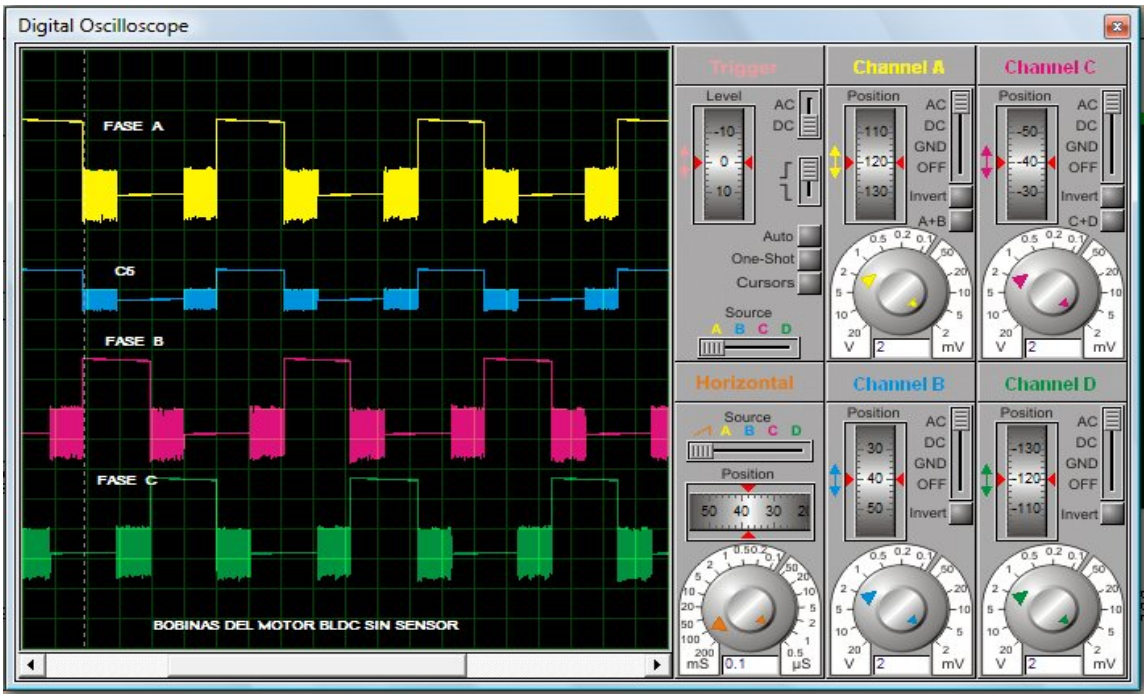


Figura 31. Gráficas de ondas producidas en las bobinas del motor BLDC sin sensor

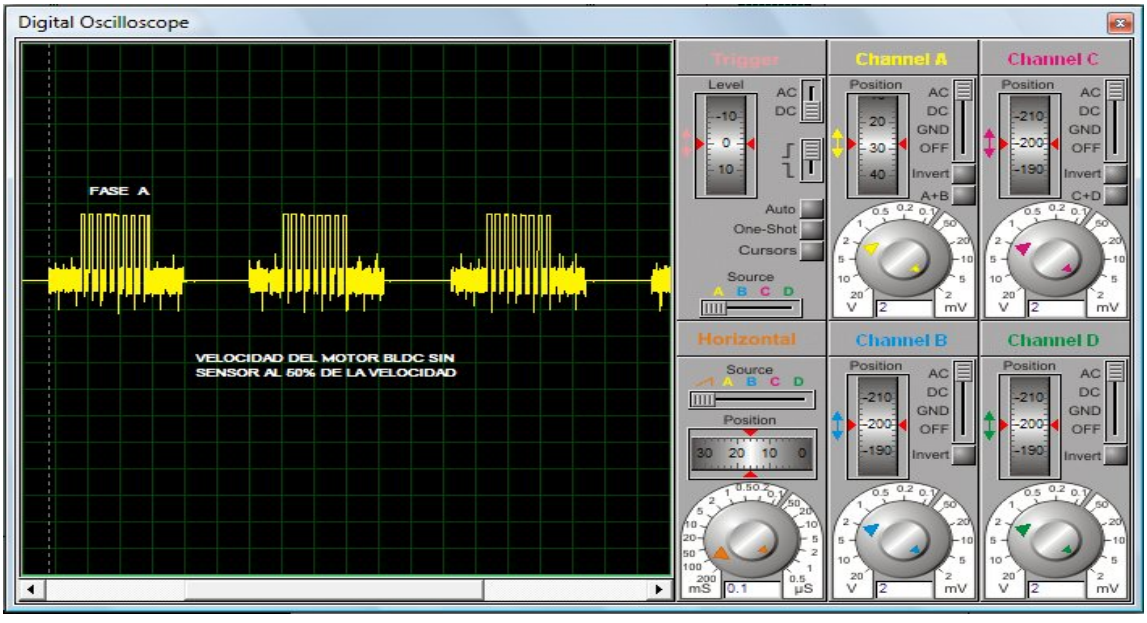


Figura 32. Velocidad del motor BLDC sin sensor al 50%

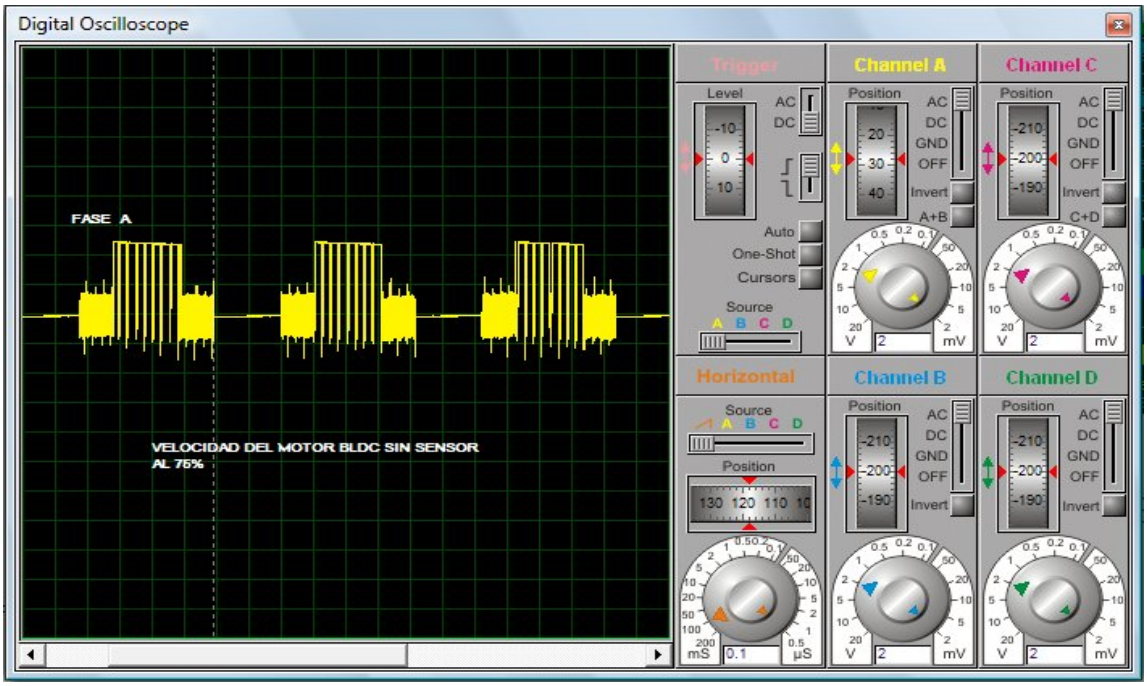


Figura 33. Velocidad del motor BLDC sin sensor al 75%

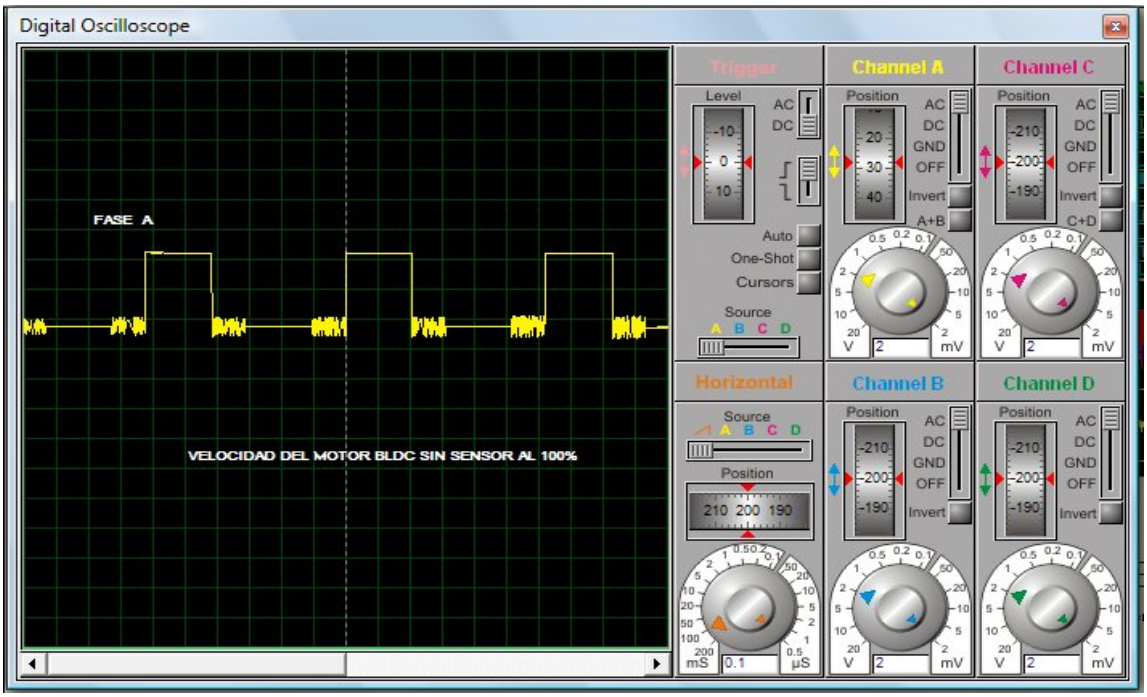


Figura 34. Velocidad del motor BLDC sin sensor al 100%

4.2. IMPLEMENTACIÓN DEL HARDWARE EN EL PROTOBOARD

El siguiente gráfico corresponde al circuito del proyecto armado en protoboard, el cual se implementó con los esquemáticos de las notas de aplicación AN857A y AN957. Utilizando el circuito de control y el circuito de potencia respectivamente de las notas de aplicación de Microchip mencionadas. Con el cual se hizo las pruebas de los dos programas que contiene la nota de aplicación AN857A tanto para el motor BLDC con sensor y sin sensor.

En el anexo D se encuentran los diseños en baquelita del controlador, tanto de los circuitos de control como el de potencia, además de mostrar el trabajo implementado y armado en una carcasa diseñada por nosotros.

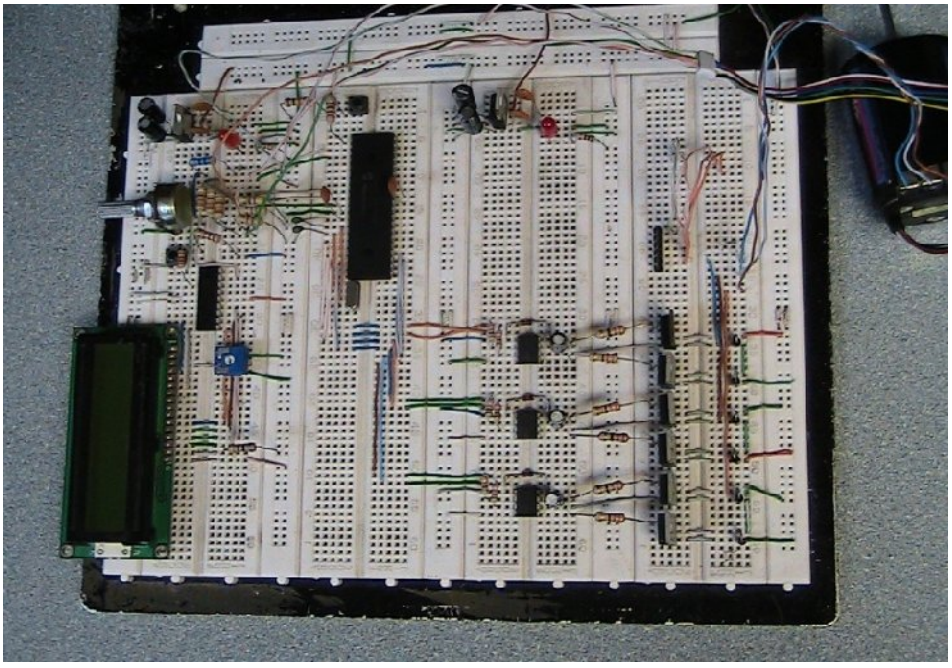


Figura 35. Circuito del proyecto armado en protoboard

4.3. PRUEBAS EN EL OSCILOSCOPIO DE LAS ETAPAS DE CONTROL Y DE POTENCIA DEL PROYECTO

Las siguientes figuras muestran las formas de onda que se producen en las salidas del sensor de efecto Hall, como el PWM que se produce en el puerto C del microcontrolador, y además las formas de ondas que producen en la salida del circuito de potencia para alimentar el motor BLDC.

Las siguientes dos figuras mostradas (figuras 36 y 37) son las formas de ondas entre los sensores de efecto Hall, en las cuales se puede observar que existe una conducción de 180° por cada sensor Hall y un desfase entre ellos de 120°

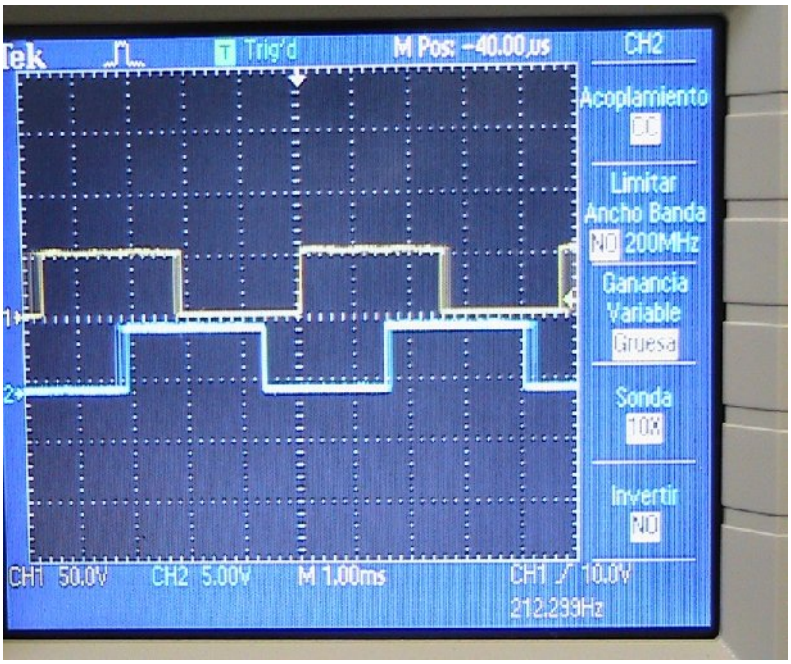


Figura 36. Forma de onda entre el sensor de efecto Hall A y B

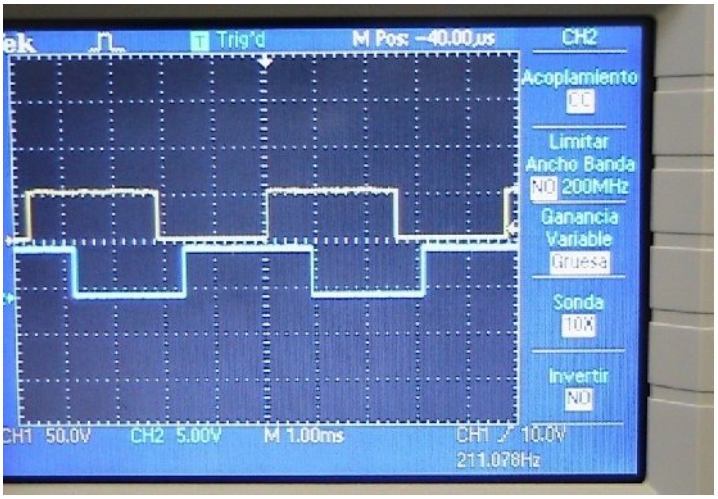


Figura 37. Forma de onda entre el sensor de efecto Hall A y C

El resultado que se muestra en el osciloscopio, corresponde a los PWM generados por el módulo de control del microcontrolador, se ven reflejados en las siguientes figuras (figuras 38, 39 y 40) muestran las formas de ondas respectivamente del 27%, 50% y 100% de la velocidad del motor BLDC.

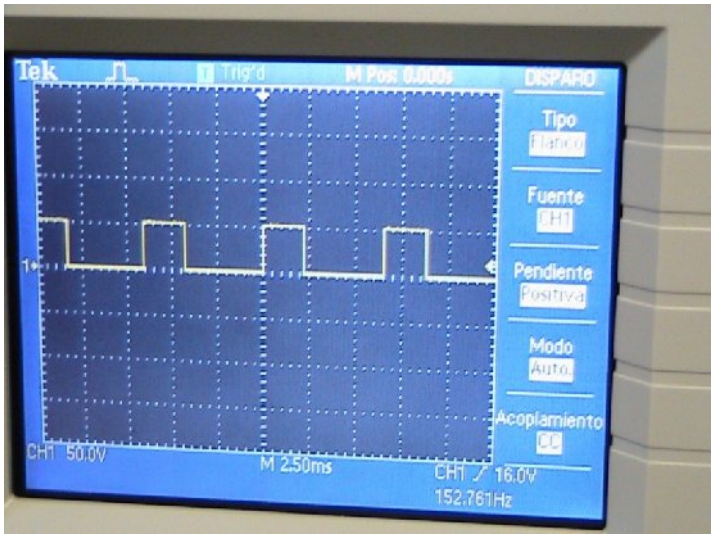


Figura 38. Forma de onda PWM al 27% de la velocidad del motor

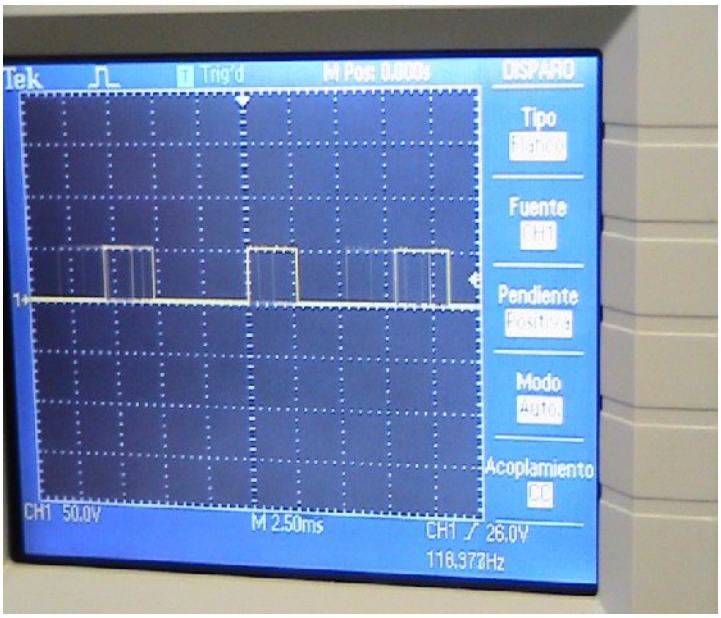


Figura 39. Forma de onda PWM al 50% de la velocidad del motor

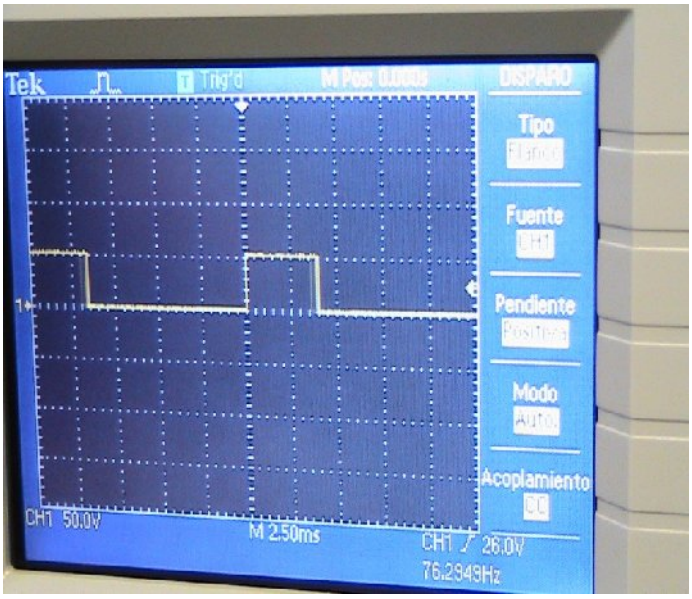


Figura 40. Forma de onda PWM al 100% de la velocidad del motor

Con las siguientes gráficas expuestas (figuras 41 y 42), son las formas de ondas que se producen en la salida del inversor trifásico, y se puede observar el desfase

entre ellas de 120°. Y estos resultados son los que reciben las fases A, B y C, bobinas del motor BLDC.

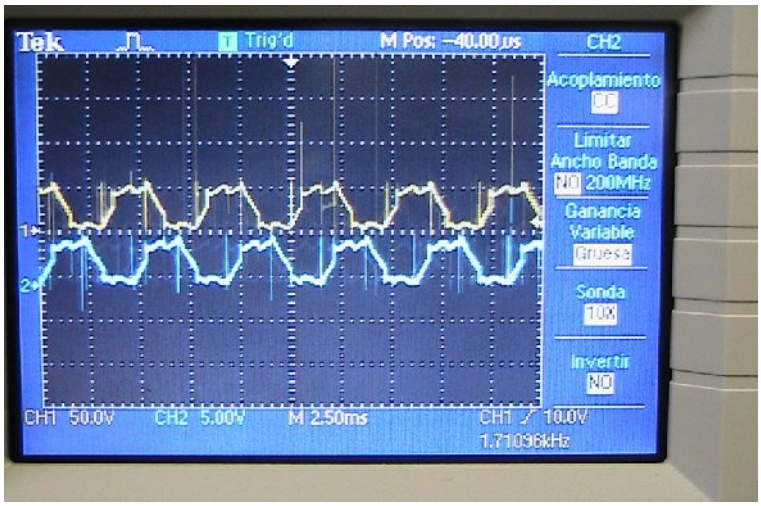


Figura 41. Forma de onda entre las bobinas A y B

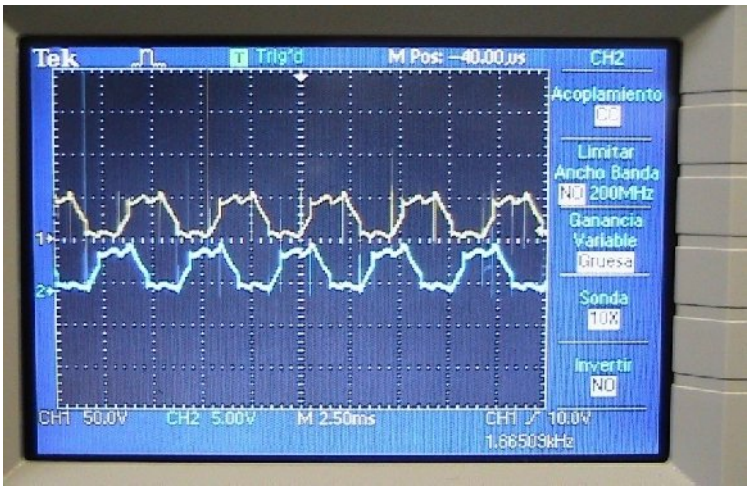


Figura 42. Forma de onda entre las bobinas A y C

CONCLUSIONES

1. Se consiguió la completa implementación del controlador de velocidad para motores BLDC con sensores de efecto Hall de Microchip, con el cual se implementó el hardware y el software, en una combinación entres las notas de aplicación dispuestas, estas son AN857A y AN957. Usando el programa para el control de velocidad de motores BLDC con sensor de la nota de aplicación AN857A en el PIC 16F877A obteniendo como resultado el arranque del motor BLDC trifásico.
2. El controlador de velocidad para motores BLDC con sensor de efecto Hall desarrollado en este proceso de graduación muestra la fácil implementación y aplicación para el uso didáctico, y experimental donde se puede comprobar el funcionamiento de sus etapas, tanto la etapa de control, acoplamiento y potencia. Además de cómo opera su programación y el funcionamiento del motor BLDC.
3. Se obtuvo el PWM por medio de la programación, simplemente con el PIC16F877A sin la necesidad de un microcontrolador de la gama alta los cuales tienen módulos PWM, solo utilizando el TMR0 en modo de temporizador y el producto de ese ejercicio mostrarlo en el puerto C del microcontrolador configurado como salida.
4. Se logró producir el adecuado disparo en la conmutación de los Mosfet's,

la programación en el microcontrolador ejecuta los correspondientes estados de conducción en las salidas del microcontrolador y observado en el osciloscopio cumple con los estados de conmutación expuestos anteriormente en la teoría. Lo cual satisface a la correcta operación del motor BLDC con el cual se está trabajando.

5. Se observó el funcionamiento del sensor de efecto Hall, gracias a la ayuda del osciloscopio se ve que cumple con la secuencia respectiva que se refirió en la parte teórica, mostrando su perfecto funcionamiento por la respuesta que se obtuvo en la adecuada operación del proyecto.
6. Se comprobó que el motor BLDC sin sensor, no tuvo un correcto funcionamiento con el programa de Microchip, debido a problemas en la implementación ya que los valores de los componentes electrónicos otorgados por la nota de aplicación AN857A no permitían se ejecute adecuadamente la rotación del motor. Otro problema adicional está en la programación, ya que el programa solo permite medir una sola fase del motor y no las tres, así no se obtiene un censo real de la BEMF de cada fase.

RECOMENDACIONES

1. Ser precavido en la conexión de la alimentación tanto en la etapa de control como en la etapa de potencia, ya que trabajan esos dos sectores a diferentes voltajes y podría una mala conexión dañar algún componente del circuito por una polarización inversa, especialmente en la etapa de control, ya que el microcontrolador es el dispositivo más sensible.
2. Las propiedades de los motores BLDC son diferentes, tener muy en cuenta las características de voltaje y de corriente en los motores a utilizar, este proyecto proporciona en la salida para la conexión de motores con una capacidad de conectar a una fuente de suministro de 100V y 14Amp. Ya que los Mosfet's soportan hasta esa cantidad de voltaje y amperaje respectivamente.
3. Al trabajar con motores que posean sensor de efecto Hall, tener muy en cuenta la correcta disposición del Hall A, Hall B y Hall C para una debida conexión en los pines del sensor Hall del proyecto. Ya que esto podría hacer parecer que el motor no funcionaría y pensar que podría ser otro problema que no existiría.
4. Tratar de mejorar el software para darle una mejor precisión en los resultados de velocidad, brindarle un sistema de lazo cerrado al programa y seguridades de sobrecorrientes y sobrevoltajes al circuito, de esta

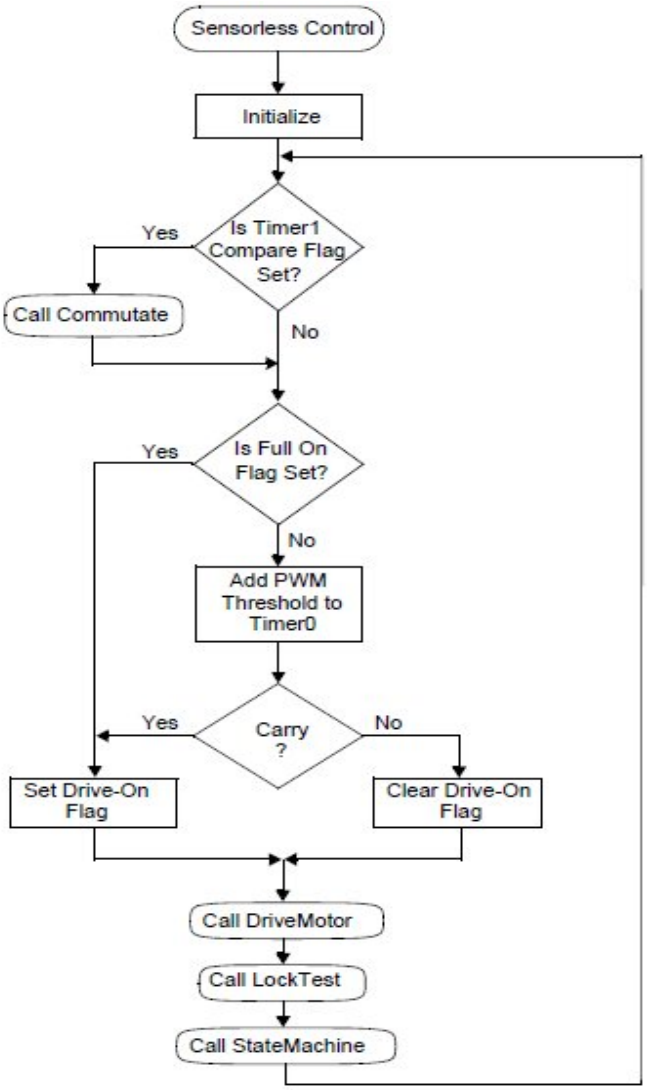
manera se pulirá las necesidades del proyecto si se desea utilizar para una actividad real donde se necesita precisión y seguridad del equipo.

5. Mejorar la implementación para el control del motor sin sensor, para eliminar problemas de ruidos en el circuito de control y tener un buen circuito adicional donde las tres fases del circuito de potencia se puedan censar y que la etapa de control las pueda medir, así como también mejorar la programación donde se pueda corregir problemas de las señales adquiridas y proporcionadas a las demás etapas del circuito.

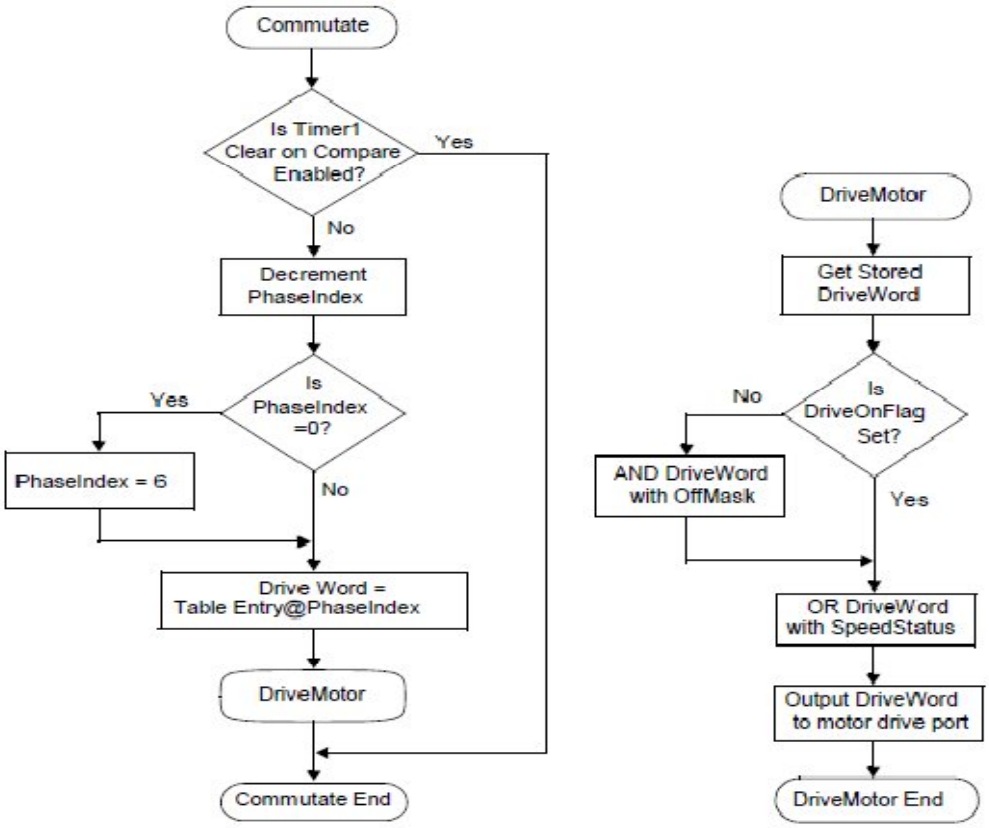
ANEXO A

Diagrama de flujo del programa para el motor BLDC sin sensor

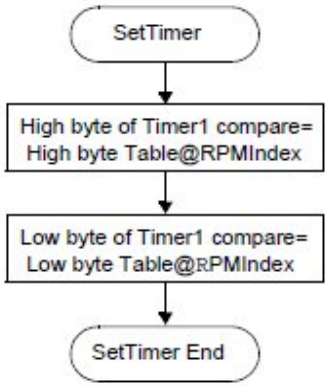
Bucle Principal



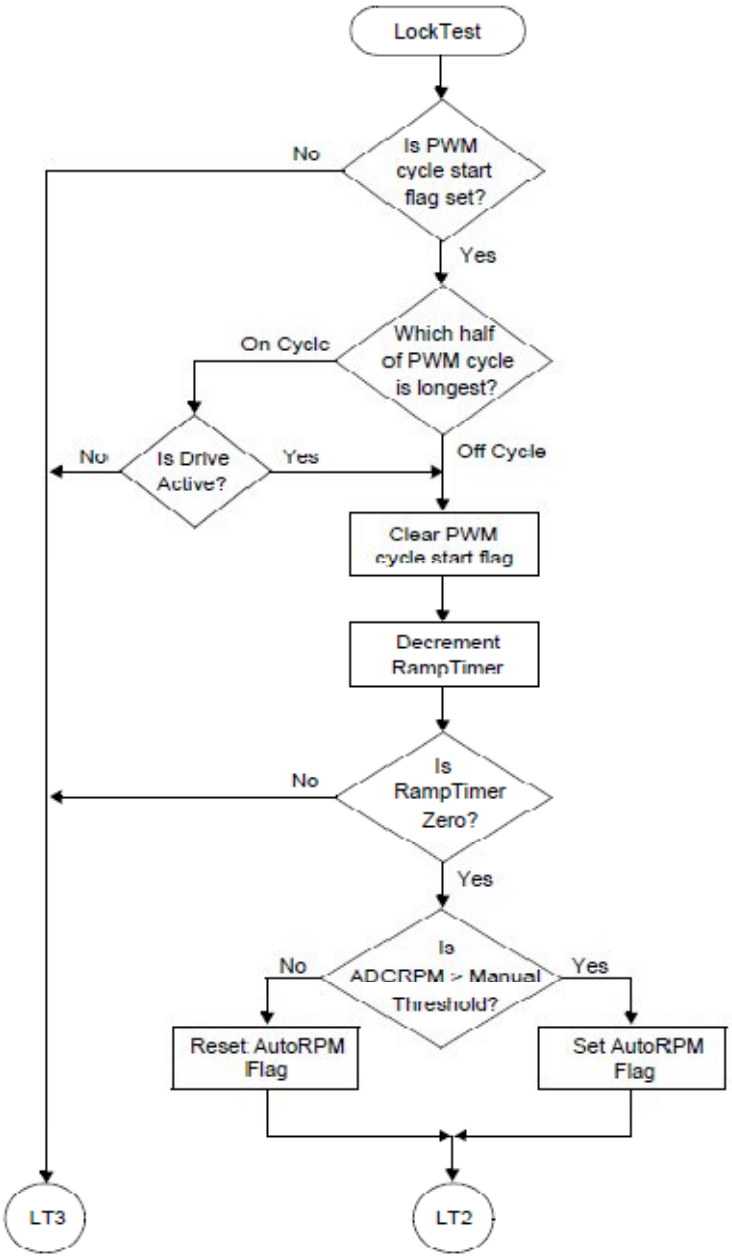
Conmutación de los drivers del motor



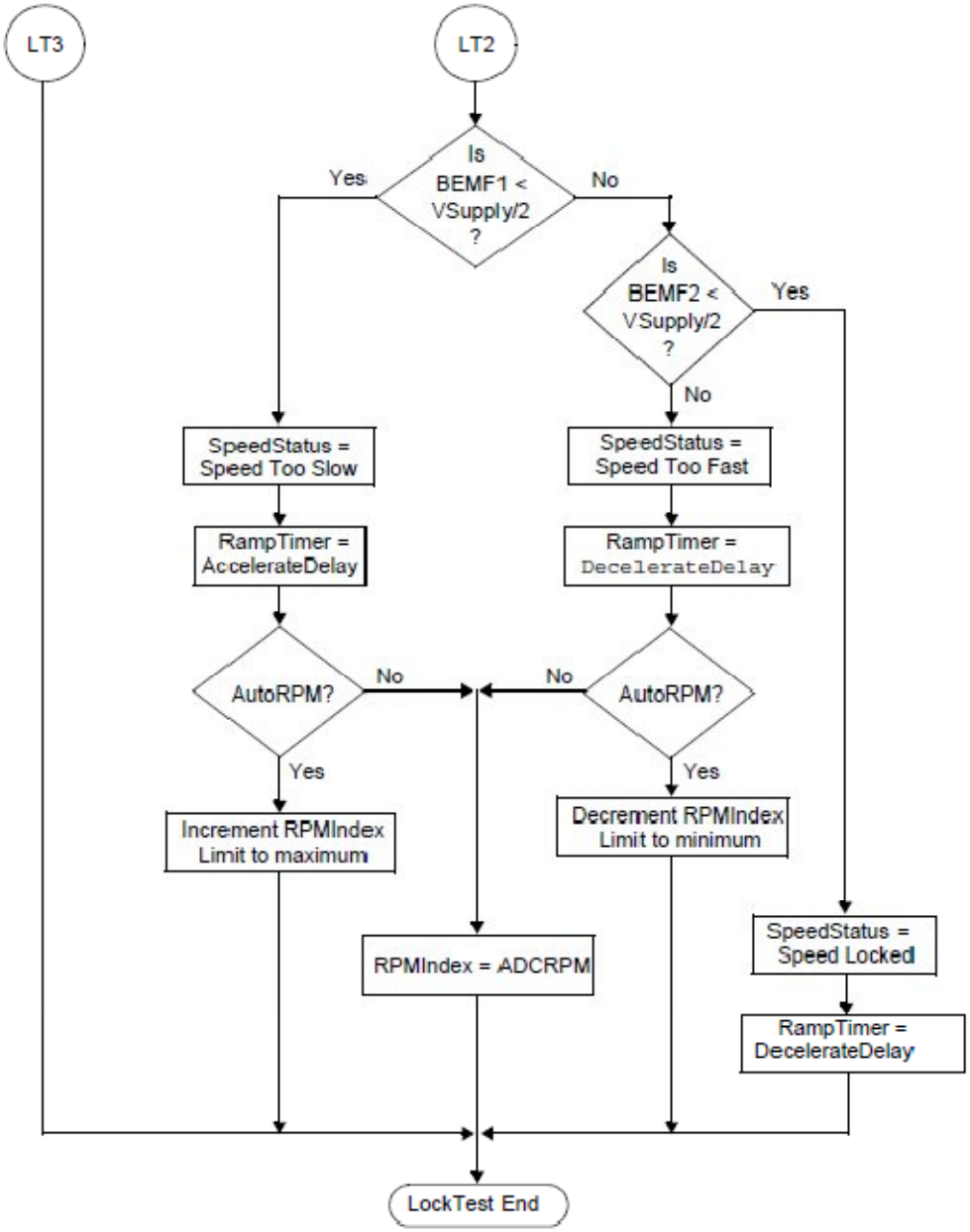
Fases de los drivers



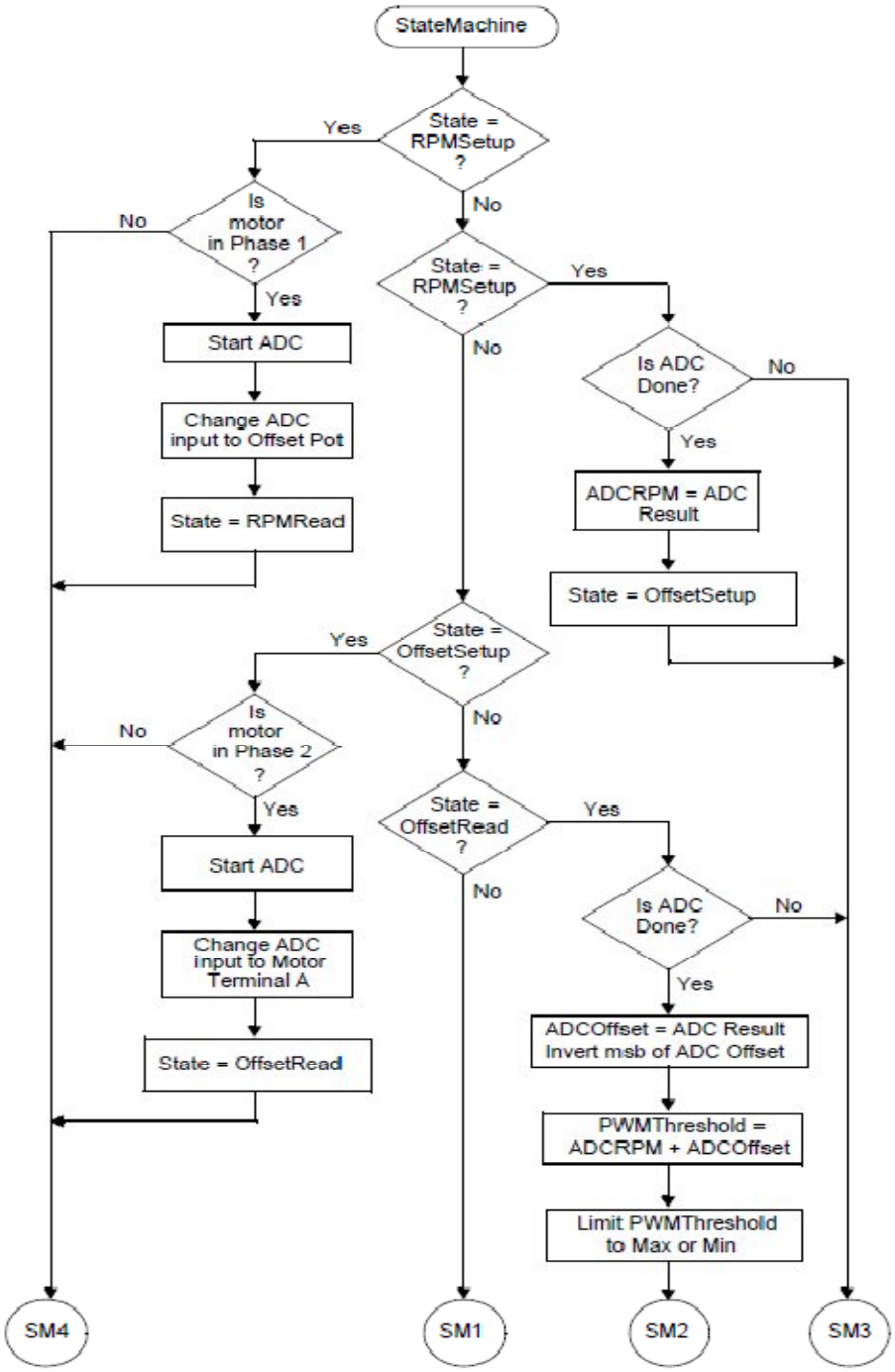
Bucle de la velocidad del motor y los rangos de conmutación



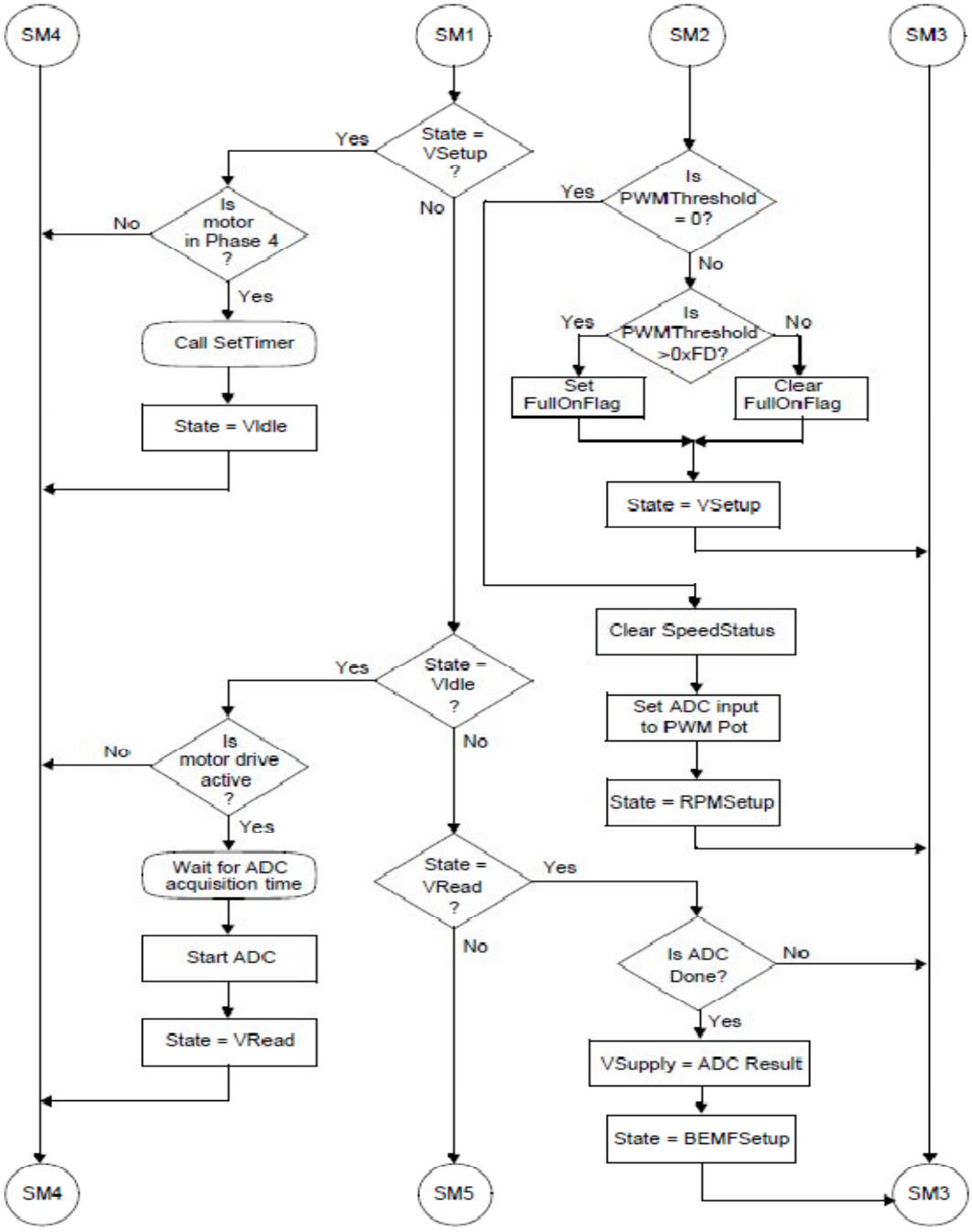
Bucle de la velocidad del motor y los rangos de conmutación (cont.)



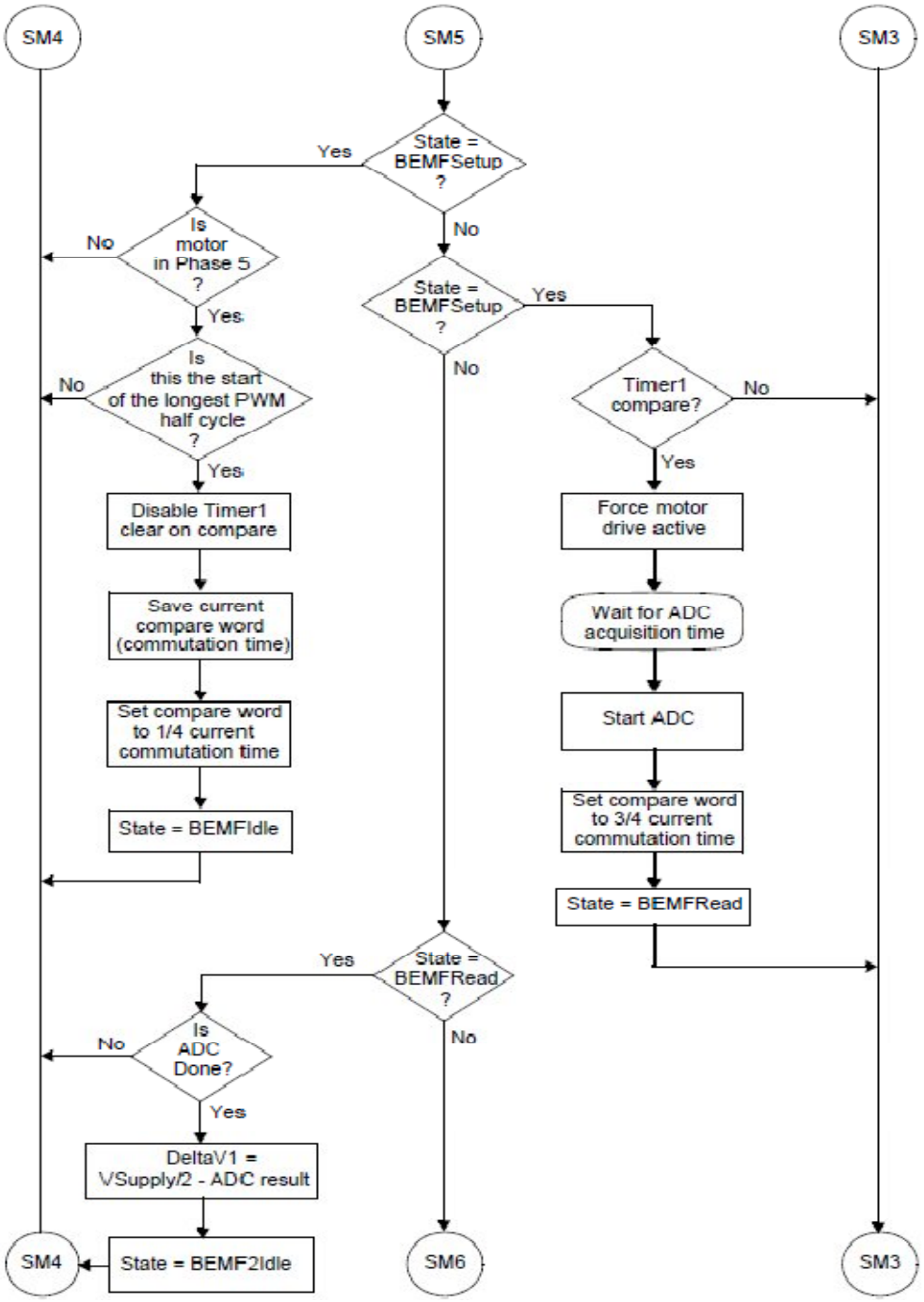
Estado de Maquina del control del motor



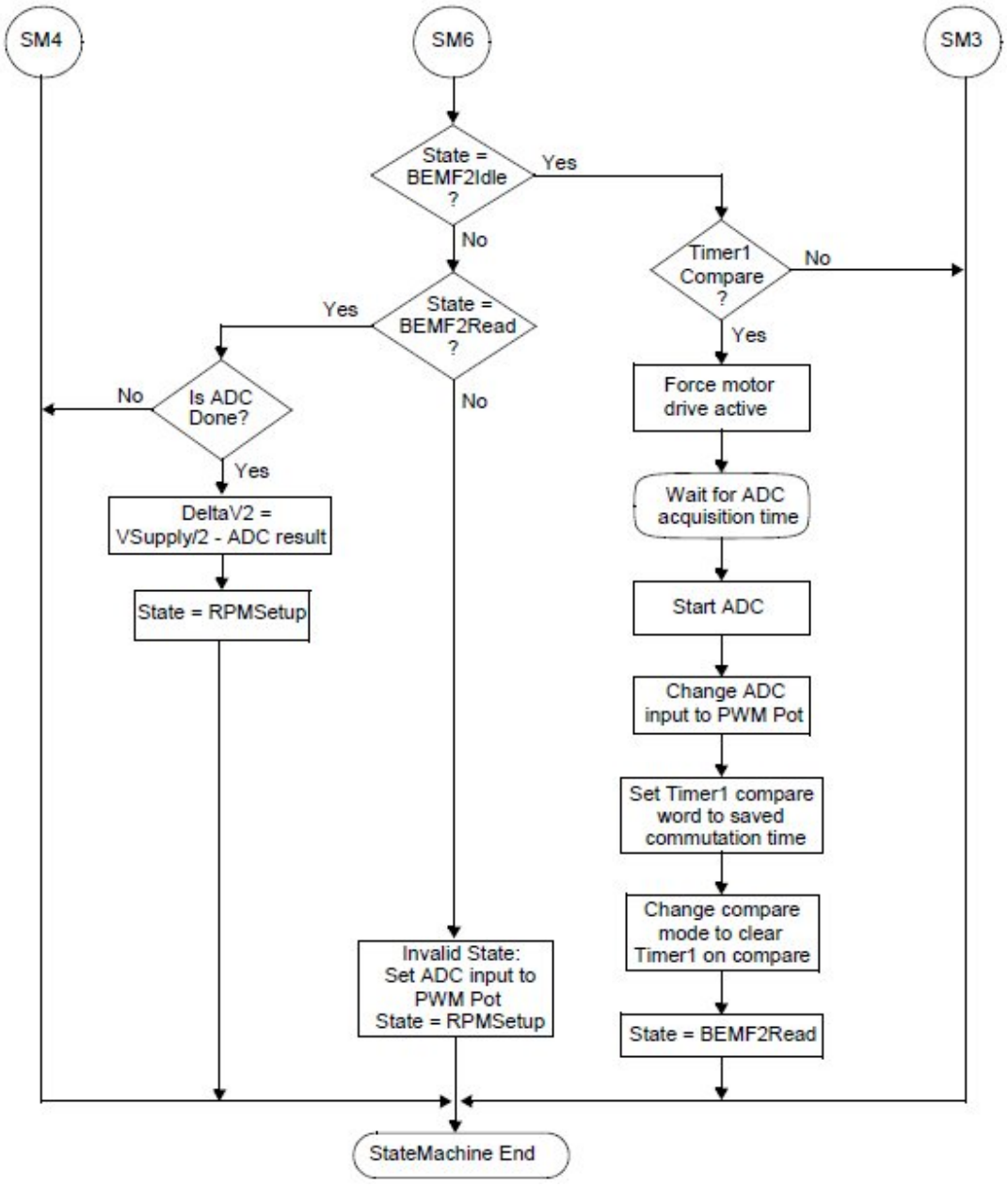
Estado de Maquina del control del motor (cont.)



Estado de Maquina del control del motor (cont.)



Estado de Maquina del control del motor (cont.)



ANEXO B

Programa del controlador de velocidad para control del motor BLDC con sensor

El siguiente programa ASM es el correspondiente a este proyecto, el cual es suministrado en la nota de aplicación de Microchip AN857A:

```
list      p=16f877                  ; list directive to define processor
#include <p16f877.inc>              ; processor specific variable definitions
__CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON & _HS_OSC &
_WRT_ENABLE_OFF & _LVP_ON & _DEBUG_OFF & _CPD_OFF
;*********************************************************************
;* Define variable storage
CBLOCK 0x20
        ADC                       ; PWM threshold is ADC result
        LastSensor                ; last read motor sensor data
        DriveWord                 ; six bit motor drive data
ENDC
;*********************************************************************
;* Define I/O
#define         OffMask         B'11010101'
#define         DrivePort       PORTC
#define         DrivePortTris   TRISC
#define         SensorMask      B'00000111'
#define         SensorPort      PORTE
#define          DirectionBit   PORTA,1
;*********************************************************************
org     0x000                     ; startup vector
nop                               ; required for ICD operation
clrf    PCLATH                    ; ensure page bits are cleared
 goto   Initialize                ; go to beginning of program
ORG     0x004                     ; interrupt vector location
retfie                             ; return from interrupt
;*********************************************************************
;* Initialize I/O ports and peripherals
org 0x10
Initialize
        clrf    DrivePort         ; all drivers off
        banksel TRISA
; setup I/O
        clrf    DrivePortTris     ; set motor drivers as outputs
        movlw   B'00000011'       ; A/D on RA0, Direction on RA1, Motor sensors on RE<2:0>
        movwf   TRISA             ;
```

```
; setup Timer0
;       movlw  B'11010000'    ; Timer0: Fosc, 1:2
        movlw  B'11010111'    ; Timer0: Fosc, 1:2
        movwf  OPTION_REG
; Setup ADC (bank1)
        movlw  B'00001110'    ; ADC left justified, AN0 only
        movwf  ADCON1
        banksel ADCON0
; setup ADC (bank0)
        movlw  B'11000001'    ; ADC clock from int RC, AN0, ADC on
        movwf  ADCON0
        bsf    ADCON0,GO      ; start ADC
        clrf   LastSensor     ; initialize last sensor reading
        call   Commutate      ; determine present motor positon
        clrf   ADC            ; start speed control threshold at zero until first ADC reading
;*****************************************************************
;* Main control loop
Loop
        call   ReadADC        ; get the speed control from the ADC
        incfsz ADC,w          ; if ADC is 0xFF we're at full speed - skip timer add
        goto   PWM            ; add timer 0 to ADC for PWM
        movf   DriveWord,w    ; force on condition
        goto   Drive          ; continue
PWM
        movf   ADC,w          ; restore ADC reading
        addwf  TMR0,w         ; add it to current timer0
        movf   DriveWord,w    ; restore commutation drive data
        btfss  STATUS,C       ; test if ADC + timer0 resulted in carry
        andlw  OffMask        ; no carry - supress high drivers
Drive
        movwf  DrivePort      ; enable motor drivers
        call   Commutate      ; test for commutation change
        goto   Loop           ; repeat loop
ReadADC
;*****************************************************************
;* If the ADC is ready then read the speed control potentiometer
;* and start the next reading
        btfsc  ADCON0,NOT_DONE  ; is ADC ready?
        return                ; no - return
        movf   ADRESH,w       ; get ADC result
        bsf    ADCON0,GO      ; restart ADC
        movwf  ADC            ; save result in speed control threshold
        return                ;
;*****************************************************************
;* Read the sensor inputs and if a change is sensed then get the
;* corresponding drive word from the drive table
Commutate
        movlw  SensorMask     ; retain only the sensor bits
        andwf  SensorPort,w   ; get sensor data
        xorwf  LastSensor,w   ; test if motion sensed
        btfsc  STATUS,Z       ; zero if no change
```

```
        return                  ; no change - back to the PWM loop
        xorwf   LastSensor,f    ; replace last sensor data with current
        btfss   DirectionBit    ; test direction bit
        goto    FwdCom          ; bit is zero - do forward commutation
                                        ; reverse commutation
        movlw   HIGH RevTable; get MS byte of table
        movwf   PCLATH          ; prepare for computed GOTO
        movlw   LOW RevTable ; get LS byte of table
        goto    Com2
FwdCom                                  ; forward commutation
        movlw   HIGH FwdTable           ; get MS byte of table
        movwf   PCLATH                  ; prepare for computed GOTO
        movlw   LOW FwdTable            ; get LS byte of table
Com2
        addwf   LastSensor,w    ; add sensor offset
        btfsc   STATUS,C        ; page change in table?
        incf    PCLATH,f                ; yes - adjust MS byte
        call    GetDrive                ; get drive word from table
        movwf   DriveWord       ; save as current drive word
        return
GetDrive
        movwf   PCL
;*********************************************************************
;* The drive tables are built based on the following assumptions:
;* 1) There are six drivers in three pairs of two
;* 2) Each driver pair consists of a high side (+V to motor) and low side (motor to ground) drive
;* 3) A 1 in the drive word will turn the corresponding driver on
;* 4) The three driver pairs correspond to the three motor windings: A, B and C
;* 5) Winding A is driven by bits <1> and <0> where <1> is A's high side drive
;* 6) Winding B is driven by bits <3> and <2> where <3> is B's high side drive
;* 7) Winding C is driven by bits <5> and <4> where <5> is C's high side drive
;* 8) Three sensor bits consitute the address offset to the drive table
;* 9) A sensor bit transitions from a 0 to 1 at the moment that the corresponding
;*    winding's high side forward drive begins.
;* 10) Sensor bit <0> corresponds to winding A
;* 11) Sensor bit <1> corresponds to winding B
;* 12) Sensor bit <2> corresponds to winding C
FwdTable
        retlw   B'00000000'     ; invalid
        retlw   B'00010010'     ; phase 6
        retlw   B'00001001'     ; phase 4
        retlw   B'00011000'     ; phase 5
        retlw   B'00100100'     ; phase 2
        retlw   B'00000110'     ; phase 1
        retlw   B'00100001'     ; phase 3
        retlw   B'00000000'     ; invalid
RevTable
        retlw   B'00000000'     ; invalid
        retlw   B'00100001'     ; phase /6
        retlw   B'00000110'     ; phase /4
        retlw   B'00100100'     ; phase /5
```

```
        retlw   B'00011000'     ; phase /2
        retlw   B'00001001'     ; phase /1
        retlw   B'00010010'     ; phase /3
        retlw   B'00000000'     ; invalid
END                             ; directive 'end of program'
```

ANEXO C

Programa del controlador de velocidad para control del motor BLDC sin sensor

El siguiente programa ASM es el correspondiente para el circuito de control del controlador para manejar la velocidad del motor BLDC sin sensor, el cual es suministrado en la nota de aplicación de Microchip AN857A:

```
list P = PIC16F877
        include "p16f877.inc"
        __CONFIG _CP_OFF & _WRT_ENABLE_OFF & _HS_OSC & _WDT_OFF &
_PWRTE_ON & _BODEN_ON

; Acceleration/Deceleration Time = RampRate * 256 * 256 * Timer 0 prescale / Fosc

#define         AccelDelay      D'255'          ; determines full range acceleration time
#define         DecelDelay      D'200'          ; determines full range deceleration time

#define         ManThresh       0x3f            ; Manual threshold is the PWM potentiomenter
                                                ; reading above which RPM is adjusted
automatically
#define         AutoThresh      0x100-ManThresh

OffMask         equ     B'11010101'     ; PWM off kills the high drives
Invalid  equ    B'00000000'     ; invalid
Phase1 equ      B'00100001'     ; phase 1 C high, A low
Phase2 equ      B'00100100'     ; phase 2 C high, B low
Phase3 equ      B'00000110'     ; phase 3 A high, B low
Phase4 equ      B'00010010'     ; phase 4 A high, C low
Phase5 equ      B'00011000'     ; phase 5 B high, C low
Phase6 equ      B'00001001'     ; phase 6 B high, A low

#define CARRY STATUS,C
#define ZERO  STATUS,Z
#define subwl   sublw

;*************************************************************************
;*
;*      Define I/O Ports
;*
```

```
#define ReadIndicator   PORTB,0                 ; diagnostic scope trigger for BEMF readings
#define DrivePort       PORTC           ; motor drive and lock status

;*****************************************************************************
;
;*
;*      Define RAM variables
;*
;

        CBLOCK 0x20

        STATE           ; Machine state
        PWMThresh       ; PWM threshold
        PhaseIndx       ; Current motor phase index
        Drive           ; Motor drive word
        RPMIndex        ; RPM Index workspace
        ADCRPM                  ; ADC RPM value
        ADCOffset       ; Delta offset to ADC PWM threshold
        PresetHi        ; speed control timer compare MS byte
        PresetLo        ; speed control timer compare LS byte
        Flags           ; general purpose flags
        Vsupply                 ; Supply voltage ADC reading
        DeltaV1                 ; Difference between expected and actual BEMF at T/4
        DeltaV2                 ; Difference between expected and actual BEMF at T/2
        CCPSaveH        ; Storage for phase time when finding DeltaV
        CCPSaveL        ; Storage for phase time when finding DeltaV
        CCPT2H                  ; Workspace for determining T/2 and T/4
        CCPT2L                  ; Workspace for determining T/2 and T/4
        RampTimer       ; Timer 0 post scaler for accel/decel ramp rate
        xCount          ; general purpose counter workspace
        Status          ; relative speed indicator status

        ENDC

;*****************************************************************************
;
;*
;*      Define Flags
;*
;

#define DriveOnFlag     Flags,0         ; Flag for invoking drive disable mask when clear
#define AutoRPM                 Flags,1         ; RPM timer is adjusted automatically
;                       Flags,3         ; Undefined
#define FullOnFlag      Flags,4         ; PWM threshold is set to maximum drive
#define Tmr0Ovf                 Flags,5         ; Timer 0 overflow flag
#define Tmr0Sync        Flags,6         ; Second Timer 0 overflow flag
;                       Flags,7         ; undefined

#define BEMF1Low        DeltaV1,7       ; BEMF1 is low if DeltaV1 is negative
#define BEMF2Low        DeltaV2,7       ; BEMF2 is low if DeltaV2 is negative

;*****************************************************************************
;
;*
;
```

```
;*      Define State machine states and index numbers
;*
;

sRPMSetup     equ     D'0'              ; Wait for Phase1, Set ADC GO, RA1->ADC
sRPMRead      equ     sRPMSetup+1  ; Wait for ADC nDONE, Read ADC->RPM
sOffsetSetup  equ     sRPMRead+1   ; Wait for Phase2, Set ADC GO, RA3->ADC
sOffsetRead   equ     sOffsetSetup+1 ; Wait for ADC nDONE, Read ADC->ADCOffset
sVSetup               equ     sOffsetRead+1 ; Wait for Phase4, Drive On, wait 9 uSec, Set
ADC GO
sVIdle        equ     sVSetup+1       ; Wait for Drive On, wait Tacq, set ADC GO

sVRead                equ     sVIdle+1          ; Wait for ADC nDONE, Read ADC->Vsupply
sBEMFSetup    equ     sVRead+1        ; Wait for Phase5, set Timer1 compare to half phase time
sBEMFIdle     equ     sBEMFSetup+1; Wait for Timer1 compare, Force Drive on and wait 9
uSec,
                                      ; Set ADC GO, RA0->ADC
sBEMFRead     equ     sBEMFIdle+1   ; Wait for ADC nDONE, Read ADC->Vbemf
sBEMF2Idle    equ     sBEMFRead+1 ; Wait for Timer1 compare, Force Drive on and wait 9
uSec,
                                      ; Set ADC GO, RA0->ADC
sBEMF2Read    equ     sBEMF2Idle+1  ; Wait for ADC nDONE, Read ADC->Vbemf

;*****************************************************************************
;*
;*      The ADC input is changed depending on the STATE
;*      Each STATE assumes a previous input selection and changes the selection
;*      by XORing the control register with the appropriate ADC input change mask
;*      defined here:
;*
;

ADC0to1               equ     B'00001000'     ; changes ADCON0<5:3> from 000 to 001
ADC1to3               equ     B'00010000'     ; changes ADCON0<5:3> from 001 to 011
ADC3to0               equ     B'00011000'     ; changes ADCON0<5:3> from 011 to 000

;*****************************************************************************
;************************** PROGRAM STARTS HERE ******************************
;*****************************************************************************
;
   org   0x000
         nop
         goto    Initialize

         org     0x004
   goto   SVR_timer0

   org 0x08

SVR_timer0
         bsf     Tmr0Ovf                 ; Timer 0 overflow flag used by accel/decel timer
         bsf     Tmr0Sync        ; Timer 0 overflow flag used to synchronize code execution
         bcf     INTCON,T0IF
         retfie                  ;
```

```
Initialize
        clrf    PORTC           ; all drivers off
        clrf    PORTB

        banksel TRISA
; setup I/O
        clrf    TRISC           ; motor drivers on PORTC
        movlw   B'00001011'     ; A/D on RA0 (PWM), RA1 (Speed) and RA3 (BEMF)
        movwf   TRISA           ;
        movlw   B'11111110'     ; RB0 is locked indicator
        movwf   TRISB
; setup Timer0
        ;movlw  B'11010000'     ; Timer0: Fosc, 1:2
   movlw   B'11010111'     ; Timer0: Fosc, 1:2
        movwf   OPTION_REG
        bsf     INTCON,T0IE     ; enable timer 0 interrupts
; Setup ADC
        movlw   B'00000100'     ; ADC left justified, AN0, AN1
        movwf   ADCON1

        bankselPORTA
        movlw   B'10000001'     ; ADC clk = Fosc/32, AN0, ADC on
        movwf   ADCON0
; setup Timer 1
        movlw   B'00100001'     ; 1:4 prescale, internal clock, timer on
        movwf   T1CON
; setup Timer 1 compare
        movlw   0xFF            ; set compare to maximum count
        movwf   CCPR1L                  ; LS compare register
        movwf   CCPR1H                  ; MS compare register
        movlw   B'00001011'     ; Timer 1 compare mode, special event - clears timer1
        movwf   CCP1CON

; initialize RAM

        clrf    PWMThresh
        movlw   D'6'
        movwf   PhaseIndx
        clrf    Flags
        clrf    Status          ;
        clrf    STATE           ; LoopIdle->STATE
        bcf     INTCON,T0IF     ; ensure timer 0 overflow flag is cleared
        bsf     INTCON,GIE      ; enable interrupts

MainLoop
;*****************************************************************
;
;       PWM, Commutation, State machine loop
;
;*****************************************************************
```

```
        btfsc   PIR1,CCP1IF     ; time for phase change?
        call    Commutate       ; yes - change motor drive
PWM
        bsf     DriveOnFlag     ; pre-set flag
        btfsc   FullOnFlag      ; is PWM level at maximum?
        goto    PWM02                   ; yes - only commutation is necessary

        movf    PWMThresh,w     ; get PWM threshold
        addwf   TMR0,w                  ; compare to timer 0
        btfss   CARRY           ; drive is on if carry is set
        bcf     DriveOnFlag     ; timer has not reached threshold, disable drive

        call    DriveMotor      ; output drive word
PWM02
        call    LockTest
        call    StateMachine    ; service state machine
        goto    MainLoop        ; repeat loop

StateMachine
        movlw   SMTableEnd-SMTable-1            ; STATE table must have 2^n entries
        andwf   STATE,f                 ; limit STATE index to state table
        movlw   high SMTable    ; get high byte of table address
        movwf   PCLATH                  ; prepare for computed goto
        movlw   low SMTable     ; get low byte of table address
        addwf   STATE,w                 ; add STATE index to table root
        btfsc   CARRY           ; test for page change in table
        incf    PCLATH,f        ; page change adjust
        movwf   PCL             ; jump into table

SMTable                                 ; number of STATE table entries MUST be evenly
divisible by 2
        goto    RPMSetup        ; Wait for Phase1, Set ADC GO, RA1->ADC, clear timer0
overflow
        goto    RPMRead                 ; Wait for ADC nDONE, Read ADC->RPM
        goto    OffsetSetup     ; Wait for Phase2, Set ADC GO, RA3->ADC
        goto    OffsetRead      ; Wait for ADC nDONE, Read ADC->ADCOffset
        goto    VSetup          ; Wait for Phase4
        goto    VIdle           ; Wait for Drive On, wait Tacq, set ADC GO

        goto    VRead           ; Wait for ADC nDONE, Read ADC->Vsupply
        goto    BEMFSetup       ; Wait for Phase5, set Timer1 compare to half phase time
        goto    BEMFIdle        ; When Timer1 compares force Drive on, Set ADC GO after
Tacq, RA0->ADC
        goto    BEMFRead        ; Wait for ADC nDONE, Read ADC->Vbemf
        goto    BEMF2Idle       ; When Timer1 compares force Drive on, Set ADC GO after
Tacq, RA0->ADC
        goto    BEMF2Read       ; Wait for ADC nDONE, Read ADC->Vbemf

; fill out table with InvalidStates to make number of table entries evenly divisible by 2
```

```
        goto    InvalidState    ; invalid state - reset state machine
        goto    InvalidState    ; invalid state - reset state machine
        goto    InvalidState    ; invalid state - reset state machine
        goto    InvalidState    ; invalid state - reset state machine
SMTableEnd

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
RPMSetup                        ; Wait for Phase1, Set ADC GO, RA1->ADC, clear timer0
overflow

        movlw   Phase1          ; compare Phase1 word...
        xorwf   Drive,w         ; ...with current drive word
        btfss   ZERO            ; ZERO if equal
        return                  ; not Phase1 - remain in current STATE

        bsf     ADCON0,GO       ; start ADC
        movlw   ADC0to1                 ; prepare to change ADC input
        xorwf   ADCON0,f        ; change from AN0 to AN1
        incf    STATE,f                 ; next STATE
        bcf     Tmr0Sync        ; clear timer0 overflow
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
RPMRead                                 ; Wait for ADC nDONE, Read ADC->RPM

        btfsc   ADCON0,GO       ; is ADC conversion finished?
        return                  ; no - remain in current STATE

        movf    ADRESH,w        ; get ADC result
        movwf   ADCRPM                  ; save in RPM

        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
OffsetSetup                     ; Wait for Phase2, Set ADC GO, RA3->ADC

        movlw   Phase2          ; compare Phase2 word...
        xorwf   Drive,w         ; ...with current drive word
        btfss   ZERO            ; ZERO if equal
        return                  ; not Phase2 - remain in current STATE

        bsf     ADCON0,GO       ; start ADC
        movlw   ADC1to3                 ; prepare to change ADC input
        xorwf   ADCON0,f        ; change from AN1 to AN3
        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop
```

```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
OffsetRead                      ; Wait for ADC nDONE, Read ADC->ADCOffset

        btfsc   ADCON0,GO       ; is ADC conversion finished?
        return                  ; no - remain in current STATE

        movf    ADRESH,w        ; get ADC result
        xorlw   H'80'           ; complement MSB for +/- offset
        movwf   ADCOffset       ; save in offset
        addwf   ADCRPM,w        ; add offset to PWM result
        btfss   ADCOffset,7     ; is offset a negative number?
        goto    OverflowTest    ; no - test for overflow

        btfss   CARRY           ; underflow?
        andlw   H'00'           ; yes - force minimum
        goto    Threshold       ;

OverflowTest
        btfsc   CARRY           ; overflow?
        movlw   H'ff'           ; yes - force maximum

Threshold
        movwf   PWMThresh       ; PWM threshold is RPM result plus offset
        btfsc   ZERO            ; is drive off?
        goto    DriveOff        ; yes - skip voltage measurements

        bcf     FullOnFlag      ; pre-clear flag in preparation of compare
        sublw   0xFD            ; full on threshold
        btfss   CARRY           ; CY = 0 if PWMThresh > FullOn
        bsf     FullOnFlag      ; set full on flag
        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

DriveOff
        clrf    Status          ; clear speed indicators
        movlw   B'11000111'     ; reset ADC input to AN0
        andwf   ADCON0,f        ;
        clrf    STATE           ; reset state machine
        return

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
VSetup                          ; Wait for Phase4

        movlw   Phase4          ; compare Phase4 word...
        xorwf   Drive,w         ; ...with current Phase drive word
        btfss   ZERO            ; ZERO if equal
        return                  ; not Phase4 - remain in current STATE

        call    SetTimer        ; set timer value from RPM table
```

```
        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
VIdle                           ; Wait for Drive On, wait Tacq, set ADC GO


        btfss   DriveOnFlag     ; is Drive active?
        return                  ; no - remain in current STATE

        bsf     ReadIndicator   ; Diagnostic
        call    Tacq            ; motor Drive is active - wait ADC Tacq time
        bsf     ADCON0,GO       ; start ADC
        bcf     ReadIndicator   ; Diagnostic
        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
VRead                           ; Wait for ADC nDONE, Read ADC->Vsupply

        btfsc   ADCON0,GO       ; is ADC conversion finished?
        return                  ; no - remain in current STATE

        movf    ADRESH,w        ; get ADC result
        movwf   Vsupply                 ; save as supply voltage
        incf    STATE,f                 ; next STATE
        bcf     Tmr0Sync        ; clear timer 0 overflow
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
BEMFSetup                       ; Wait for Phase5, set Timer1 compare to half phase time

        movlw   Phase5          ; compare Phase5 word...
        xorwf   Drive,w         ; ...with current drive word
        btfss   ZERO            ; ZERO if equal
        return                  ; not Phase5 - remain in current STATE

        btfss   Tmr0Sync        ; synchronize with timer 0
        return                  ;

        btfss   PWMThresh,7     ; if PWMThresh > 0x80 then ON is longer than OFF
        goto    BEMFS1                  ; OFF is longer and motor is currently off - compute now

        btfss   DriveOnFlag     ; ON is longer - wait for drive cycle to start
        return                  ; not started - wait

BEMFS1
        bcf     CCP1CON,0       ; disable special event on compare
```

```
        movf    CCPR1H,w        ; save current capture compare state
        movwf   CCPSaveH        ;
        movwf   CCPT2H                  ; save copy in workspace
        movf    CCPR1L,w        ; low byte
        movwf   CCPSaveL        ; save
        movwf   CCPT2L                  ; and save copy
        bcf     CARRY           ; pre-clear carry for rotate
        rrf     CCPT2H,f        ; divide phase time by 2
        rrf     CCPT2L,f        ;
        bcf     CARRY           ; pre-clear carry
        rrf     CCPT2H,w        ; divide phase time by another 2
        movwf   CCPR1H                  ; first BEMF reading at phase T/4
        rrf     CCPT2L,w        ;
        movwf   CCPR1L                  ;

        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
BEMFIdle                        ; When Timer1 compares force Drive on, Set ADC GO after
Tacq, RA0->ADC

        btfss   PIR1,CCP1IF     ; timer compare?
        return                  ; no - remain in current STATE

        bsf     DriveOnFlag     ; force drive on for BEMF reading
        call    DriveMotor      ; activate motor drive
        bsf     ReadIndicator   ; Diagnostic
        call    Tacq            ; wait ADC acqisition time
        bsf     ADCON0,GO       ; start ADC
        bcf     ReadIndicator   ; Diagnostic

; setup to capture BEMF at phase 3/4 T

        movf    CCPT2H,w
        addwf   CCPR1H,f        ; next compare at phase 3/4 T
        movf    CCPT2L,w        ;
        addwf   CCPR1L,f        ; set T/2 lsb
        btfsc   CARRY           ; test for carry into MSb
        incf    CCPR1H,f        ; perform carry
        bcf     PIR1,CCP1IF     ; clear timer compare interrupt flag
        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
BEMFRead                        ; Wait for ADC nDONE, Read ADC->Vbemf

        btfsc   ADCON0,GO       ; is ADC conversion finished?
        return                  ; no - remain in current STATE
```

```
        rrf     Vsupply,w       ; divide supply voltage by 2
        subwf   ADRESH,w        ; Vbemf - Vsupply/2

        movwf   DeltaV1                 ; save error voltage
        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
BEMF2Idle                       ; When Timer1 compares force Drive on, Set ADC GO after
Tacq, RA0->ADC

        btfss   PIR1,CCP1IF     ; timer compare?
        return                  ; no - remain in current STATE

        bsf     DriveOnFlag     ; force drive on for BEMF reading
        call    DriveMotor      ; activate motor drive
        bsf     ReadIndicator   ; Diagnostic
        call    Tacq            ; wait ADC acqisition time
        bsf     ADCON0,GO       ; start ADC
        bcf     ReadIndicator   ; Diagnostic
        movlw   ADC3to0                 ; prepare to change ADC input
        xorwf   ADCON0,f        ; change from AN3 to AN0

; restore Timer1 phase time and special event compare mode

        movf    CCPSaveH,w
        movwf   CCPR1H                  ; next compare at phase T
        movf    CCPSaveL,w      ;
        movwf   CCPR1L                  ; set T lsb
        bcf     PIR1,CCP1IF     ; clear timer compare interrupt flag
        bsf     CCP1CON,0       ; enable special event on compare
        incf    STATE,f                 ; next STATE
        return                  ; back to Main Loop

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
BEMF2Read                       ; Wait for ADC nDONE, Read ADC->Vbemf

        btfsc   ADCON0,GO       ; is ADC conversion finished?
        return                  ; no - remain in current STATE

        rrf     Vsupply,w       ; divide supply voltage by 2
        subwf   ADRESH,w        ; Vbemf - Vsupply/2

        movwf   DeltaV2                 ; save error voltage

        clrf    STATE           ; reset state machine to beginning
        return                  ; back to Main Loop
```

```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
InvalidState                        ; trap for invalid STATE index
        movlw   B'11000111'         ; reset ADC input to AN0
        andwf   ADCON0,f            ;
        clrf    STATE
        return
;________________________________________________________________________________________________

Tacq
;*****************************************************************
;
;       Software delay for ADC acquisition time
;       Delay time = Tosc*(3+3*xCount)
;
;*****************************************************************

        movlw   D'14'               ; 14 equates to approx 9 uSec delay
        movwf   xCount ;
        decfsz  xCount,f            ;
        goto    $-1                 ; loop here until time complete
        return

LockTest
;*****************************************************************
;
;       T is the commutation phase period. Back EMF is measured on the
;       floating motor terminal at two times during T to determine
;       the approximate zero crossing of the BEMF. BEMF low means that
;       the measured BEMF is below (supply voltage)/2.
;       If BEMF is low at 1/4 T then accelerate.
;       If BEMF is high at 1/4 T and low at 3/4 T then speed is OK.
;       If BEMF is high at 1/4 T and 3/4 T then decelerate.
;
;       Lock test computation is synchronized to the PWM clock such
;       that the computation is performed during the PWM ON or OFF
;       time whichever is longer.
;
;*****************************************************************

; synchronize test with start of timer 0

        btfss   Tmr0Ovf                     ; has timer 0 wrapped around?
        return                      ; no - skip lock test

        btfss   PWMThresh,7         ; if PWMThresh > 0x80 then ON is longer than OFF
        goto    LT05                ; OFF is longer and motor is currently off - compute now

        btfss   DriveOnFlag         ; ON is longer - wait for drive cycle to start
        return                      ; not started - wait
```

```
LT05
        bcf     Tmr0Ovf                 ; clear synchronization flag
        decfsz  RampTimer,f     ; RampTimer controls the acceleration/deceleration rate
        return

; use lock results to control RPM only if not manual mode

        bsf     AutoRPM                 ; preset flag
        movf    ADCRPM,w        ; compare RPM potentiometer...
        addlw   AutoThresh      ; ...to the auto control threshold
        btfss   CARRY           ; CARRY is set if RPM is > auto threshold
        bcf     AutoRPM                 ; not in auto range - reset flag

        btfss   BEMF1Low        ; is first BEMF below Supply/2
        goto    LT20            ; no - test second BEMF

LT10
; accelerate if BEMF at 1/4 T is below Supply/2

        movlw   B'10000000'     ; indicate lock test results
        movwf   Status          ; status is OR'd with drive word later
        movlw   AccelDelay      ; set the timer for acceleration delay
        movwf   RampTimer       ;

        btfss   AutoRPM                 ; is RPM in auto range?
        goto    ManControl      ; no - skip RPM adjustment

        incfsz  RPMIndex,f      ; increment the RPM table index
        return                  ; return if Index didn't wrap around

        decf    RPMIndex,f      ; top limit is 0xFF
        return
LT20
        btfsc   BEMF2Low        ; BEMF1 was high...
        goto    ShowLocked      ; ... and BEMF2 is low - show locked

; decelerate if BEMF at 3/4 T is above Supply/2

        movlw   B'01000000'     ; indicate lock test results
        movwf   Status          ; status is OR'd with drive word later
        movlw   DecelDelay      ; set the timer for deceleration delay
        movwf   RampTimer       ;

        btfss   AutoRPM                 ; is RPM in auto range?
        goto    ManControl      ; no - skip RPM adjustment

        decfsz  RPMIndex,f      ; set next lower RPM table index
        return                  ; return if index didn't wrap around

        incf    RPMIndex,f      ; bottom limit is 0x01
```

```
        return

ShowLocked
        movlw   B'11000000'     ; indicate lock test results
        movwf   Status          ; status is OR'd with drive word later
        movlw   DecelDelay      ; set the timer for deceleration delay
        movwf   RampTimer       ;

        btfsc   AutoRPM                 ; was RPM set automatically?
        return                  ; yes - we're done

ManControl
        movf    ADCRPM,w        ; get RPM potentiometer reading...
        movwf   RPMIndex        ; ...and set table index directly
        return

Commutate
;*****************************************************************
;
;       Commutation is triggered by PIR1<CCP1IF> flag.
;       This flag is set when timer1 equals the compare register.
;       When BEMF measurement is active the compare time is not
;       cleared automatically (special event trigger is off).
;    Ignore the PIR1<CCP1IF> flag when special trigger is off
;       because the flag is for BEMF measurment.
;       If BEMF measurement is not active then decrement phase table
;       index and get the drive word from the table. Save the
;       drive word in a global variable and output to motor drivers.
;
;*****************************************************************
        btfss   CCP1CON,0       ; is special event on compare enabled?
        return                  ; no - this is a BEMF measurment, let state machine handle this
        bcf     PIR1,CCP1IF     ; clear interrupt flag
        movlw   high OnTable    ; set upper program counter bits
        movwf   PCLATH
        decfsz  PhaseIndx,w     ; decrement to next phase
        goto    $+2             ; skip reset if not zero
        movlw   D'6'            ; phase counts 6 to 1
        movwf   PhaseIndx       ; save the phase index
        addlw   LOW OnTable
        btfsc   CARRY           ; test for possible page boundry
        incf    PCLATH,f        ; page boundry adjust
        call    GetDrive
        movwf   Drive           ; save motor drive word
DriveMotor
        movf    Drive,w         ; restore motor drive word
        btfss   DriveOnFlag     ; test drive enable flag
        andlw   OffMask                 ; kill high drive if PWM is off
        iorwf   Status,w        ; show speed indicators
        movwf   DrivePort       ; output to motor drivers
        return
```

```
GetDrive
        movwf  PCL              ; computed goto
OnTable
        retlw   Invalid
        retlw   Phase6
        retlw   Phase5
        retlw   Phase4
        retlw   Phase3
        retlw   Phase2
        retlw   Phase1
        retlw   Invalid

SetTimer
;*****************************************************************
;
;       This sets the CCP module compare registers for timer 1.
;       The motor phase period is the time it takes timer 1
;       to count from 0 to the compare value. The CCP module
;       is configured to clear timer 1 when the compare occurs.
;       Get the timer1 compare variable from two lookup tables, one
;       for the compare high byte and the other for the low byte.
;
;*****************************************************************
        call    SetTimerHigh
        movwf  CCPR1H                   ; Timer1 High byte preset
        call    SetTimerLow
        movwf  CCPR1L                   ; Timer1 Low byte preset
        return

SetTimerHigh
        movlw  high T1HighTable     ; lookup preset values
        movwf  PCLATH                       ; high bytes first
        movlw  low T1HighTable              ;
        addwf  RPMIndex,w           ; add table index
        btfsc   STATUS,C            ; test for table page crossing
        incf    PCLATH,f            ;
        movwf  PCL                  ; lookup - result returned in W
SetTimerLow
        movlw  high T1LowTable              ; repeat for lower byte
        movwf  PCLATH                       ;
        movlw  low T1LowTable               ;
        addwf  RPMIndex,w           ; add table index
        btfsc   STATUS,C            ; test for table page crossing
        incf    PCLATH,f            ;
        movwf  PCL                  ; lookup - result returned in W

#include "BLDCspd4.inc"

        end
```

ANEXO D

Controlador de velocidad

Las siguientes gráficas corresponden a las placas de baquelita de este proyecto, tanto la de los dos circuitos de control, como la baquelita del circuito de potencia. Además la muestra gráfica del proyecto implementado y armado.

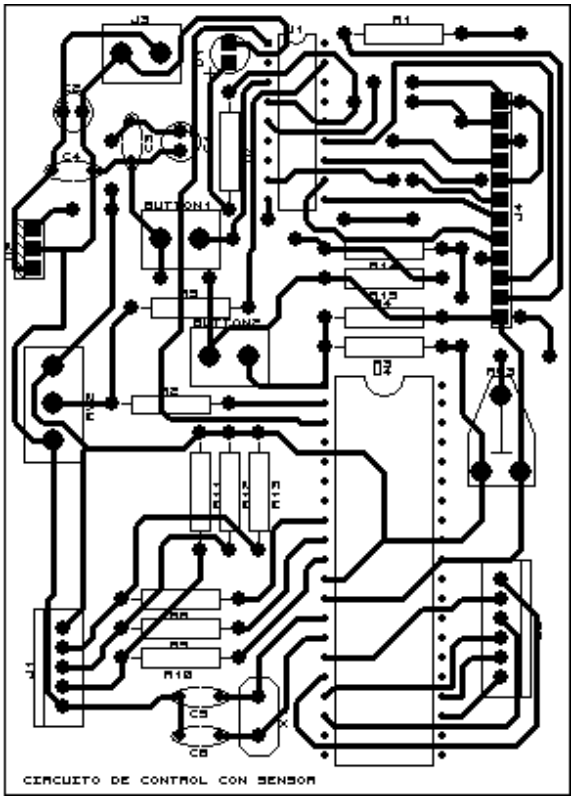


Figura A1. Baquelita del Circuito de Control con Sensor

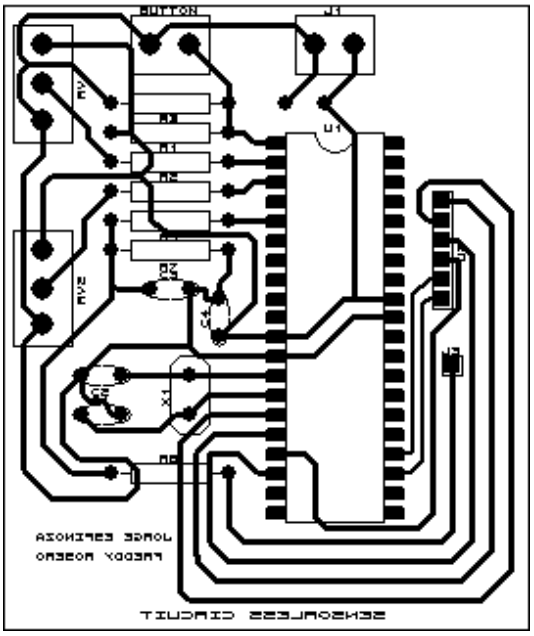


Figura A2. Baquelita del Circuito de Control sin Sensor

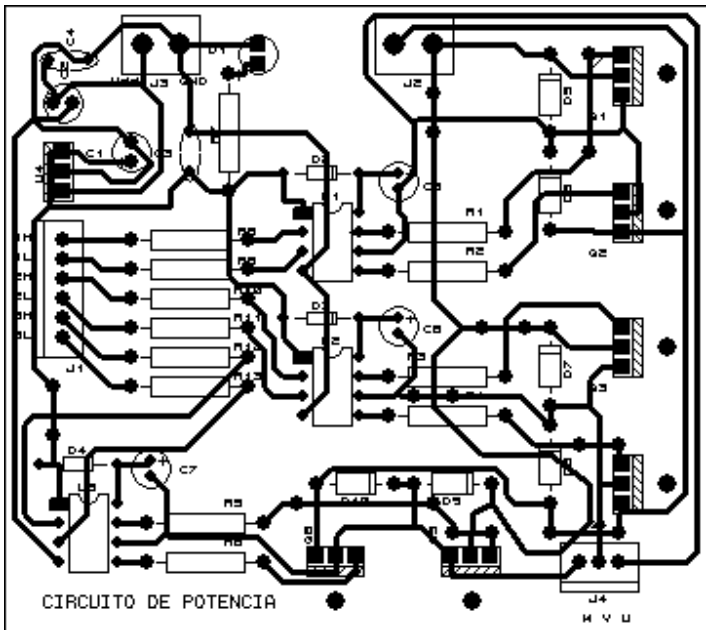


Figura A3. Baquelita del Circuito de Potencia

Las siguientes imágenes muestran el proyecto implementado y adaptado a una carcasa para un mejor manejo y presentación del controlador de velocidad para el motor BLDC.

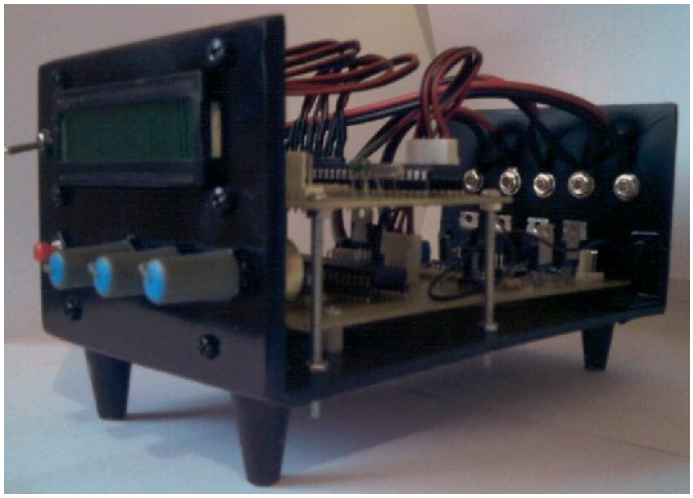


Figura A4. Controlador de Velocidad

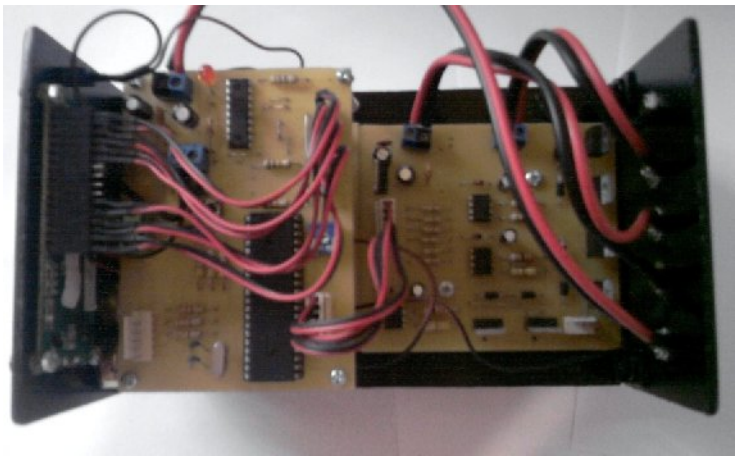


Figura A5. Perspectiva superior del controlador de velocidad

El circuito de control y el circuito de potencia se muestran a continuación, proporcionando una vista general de cómo se encuentran soldados los componentes y dispositivos en las placas de baquelita.

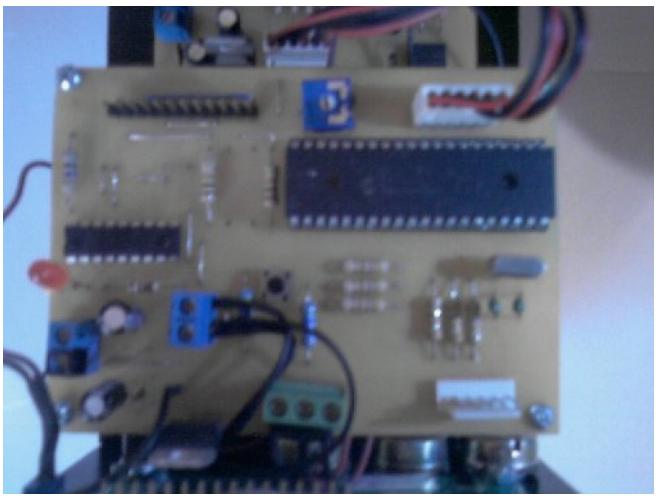


Figura A6. Circuito de control implementado

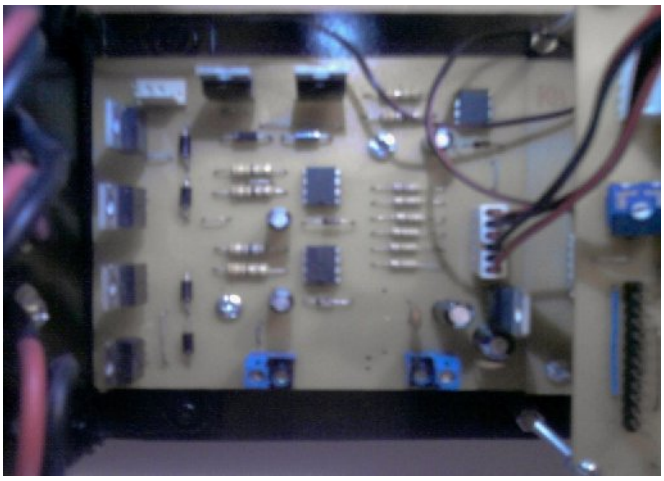


Figura A7. Circuito de potencia implementado

BIBLIOGRAFÍA

[1] Microchip, *AN885 – Brushless DC (BLDC) Motor Fundamentals*, http://ww1.microchip.com/downloads/en/AppNotes/00885a.pdf, Febrero 2010.

[2] Microchip, *AN857A – Brushless DC Motor Control Made Easy*, http://ww1.microchip.com/downloads/en/AppNotes/00857.pdf, Febrero 2010.

[3] Microchip, *AN957 - Sensored BLDC Motor Control Using dsPIC30F*, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en021551, Febrero 2010.

[4] Fernando Ramiro, Lucas J. López y Enrique Palacios, *Microcontrolador PIC16F84 – Desarrollo de proyectos*, Editorial Alfa Omega-RAMA, México, 2004.