IMPLEMENTACIÓN DE UN ALGORITMO PARA EL CONTROL DE VELOCIDAD DE UN MOTOR DE INDUCCIÓN MEDIANTE TMS320F28069M

HENRY ALFONSO SEPULVEDA PACAGUI

UNIVERSIDAD DE PAMPLONA
PROGRAMA DE INGENIERÍA ELÉCTRICA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PAMPLONA
2019

IMPLEMENTACIÓN DE UN ALGORITMO PARA EL CONTROL DE VELOCIDAD DE UN MOTOR DE INDUCCIÓN MEDIANTE TMS320F28069M

HENRY ALFONSO SEPULVEDA PACAGUI

Tesis o trabajo de investigación presentado como requisito parcial para optar al título de:
INGENIERO ELECTRÓNICO

Director
EDISON CAICEDO PEÑARANDA
M.Sc. Ingeniero Eléctrico

Codirector (a):
LUIS D. PABÓN FERNÁNDEZ
M.Sc. Ingeniero Eléctrico

Línea de Investigación:
Calidad de la energía
Grupo de Investigación:
Sistemas Energéticos

UNIVERSIDAD DE PAMPLONA
PROGRAMA DE INGENIERÍA ELÉCTRICA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PAMPLONA
2019

DEDICATORIA

Dedico este logro a las personas que me han brindado su confianza, apoyo incondicional y motivación durante todo el proceso de elaboración de mi proyecto de grado.

Este trabajo está dedicado a mi familia, en especial a mi madre BERTHA DILIA PACAGUI MALDONADO y mis hermanos porque ellos han sido mi apoyo incondicional y me han proporcionado todo y cada cosa que he necesitado.

A mi novia quien me apoyo y motivó en los momentos más turbulentos en la realización de mi proyecto de grado.

A mi director de tesis por haberme confiado este proyecto, por su paciencia ante mi inconsistencia, por su valiosa dirección y apoyo para culminar este trabajo.

A todas las personas que de una u otra manera estuvieron a mi lado, que me enseñaron y me dieron ánimos.

AGRADECIMIENTOS

En primer lugar, quiero agradecer a Dios por haberme guiado durante este proceso de arduo esfuerzo y por colocarme personas que han sido apoyo incondicional durante este tiempo de una etapa más de vida.

A mi madre que a lo largo de la vida me ha apoyado y motivado, creyendo en mi todo momento.
A cada uno de los docentes que hicieron parte de mi formación, en especial a mi director de tesis EDISON CAICEDO PEÑARANDA y codirector LUIS DAVID PABÓN.

A mis compañeros porque de cada uno aprendí cosas nuevas, mil gracias por todo su apoyo y enseñanzas.

Por último, a la universidad de Pamplona por permitirme hacer parte de esta gran familia, brindándome la oportunidad de formarme como profesional.

RESUMEN

Este libro presenta el desarrollo un algoritmo para el control de velocidad para el motor de inducción mediante el microcontrolador TMS320F28069M, el cual recibe la variable velocidad proveniente un tacogenerador, la cual es filtrada y normalizado por un circuito de diseño y Autor, donde la magnitud eléctrica adquirida será transformada a su valor digital para ser manipulada por la lógica de programación elaborada. Los algoritmos se efectuaron en base a los límites de operación de la técnica de control escalar, y las características del DSP, la técnica de control junto con el algoritmo de optimización empleado proveerán las modulaciones con optimización del contenido armónico, basado en el método de variación de frecuencia adoptando un convertidor multinivel, el cual recibe las señales de control provenientes de los GPIO del dispositivo actuando como elemento final de control del lazo planteado. El convertidor multinivel adoptado convierte el voltaje proveniente del BUS de CD a un voltaje cuasi sinusoidal AC gracias a las etapas antes planteadas, proporcionando la fuente de alimentación al motor de inducción que intrínsecamente lleva la técnica de control escalar.

Por último se implementó un controlador PI desarrollado para el sistema, donde la respuesta del sistema se supervisa con la ayuda de una interfaz la cual ayudo a observar el comportamiento del prototipo planteado.

Palabras clave: Distorsión Armónica, Inversor de Potencia, Optimización, TMS320F28069M, motor de inducción, control velocidad, sistemas embebidos.

ABSTRACT

This book presents the development of an algorithm for the speed control for the induction motor by means of the TMS320F28069M microcontroller, which receives the variable speed from a taco-generator, which is filtered and normalized by a circuit of its own design and elaboration, where the acquired electrical magnitude will be transformed to its digital value to be manipulated by the elaborated programming logic. The algorithms are detected based on the operating limits of the scalar control technique, and the characteristics of the DSP, the control technique together with the optimization algorithm used will provide the modifications with the harmonic content optimization, based on the method of frequency variation adopting a multilevel converter, which receives the control signals from the GPIO of the device acting as the final control element of the raised loop. The adopted multilevel converter converts the voltage coming from the CD BUS to a quasi-sinusoidal AC voltage thanks to the steps outlined above, the power supply to the induction motor that intrinsically carries the scalar control technique.

Finally, a PI controller developed for the system is implemented, where the response of the system is monitored with the help of an interface which helped to observe the behavior of the proposed prototype.

Keywords: Harmonic Distortion, Power Inverter, Optimization, TMS320F28069M, induction motor, speed control, embedded systems.

Tabla De Contenido

Tabla De Ilustraciones

Tabla De Tablas

INTRODUCCIÓN

Los microcontroladores piccolo ™ C2000, MCU surgieron cuando la tecnología permitió su fabricación y las características de las aplicaciones lo necesitaron. La telefónica móvil, la electro medicina, la robótica, las comunicaciones, la reproducción y el procesamiento del sonido y la imagen, el internet, el control de motores, la instrumentación, son algunas de las áreas típicas de los MCU Cuando los microprocesadores y microcontroladores de 8, 16 y 32 bits no fueron capaces de resolver eficientemente las tareas que el procesamiento digital de señales exigía se reforzaron sus arquitecturas, se amplió el repertorio de instrucciones y se le arropó con numerosos periféricos complementarios para dar lugar a los MCU tipo DSP's.

Los MCU son microcontroladores cuyo diseño ha sido enfocado para soportar las especificaciones del tratamiento de señales, que, por su implicación en los campos tecnológicos más demandados recientemente, supone disponer de un dispositivo programable con los recursos físicos y lógicos precisos para la exigencia de dicha área.

En el capítulo 1 se describe a nivel general las características de la tarjeta de desarrollo LAUNCHXL-F28069M adoptada para la ejecución de los objetivos a nivel de algoritmos planteados, se describe completamente las características del control a implementar destacando sus ventajas, desventajas y limitaciones propias de acuerdo a las características intrínsecas de las máquinas de inducción; en el capítulo 2 se realiza la descripción de la instrumentación sensórica de acuerdo al lazo de control planteado.

En el capítulo 3 se describe la mayoría de componentes a usarse para la implementación de la técnica de control, se realiza una descripción a detalle del inversor multinivel adoptado para el control del motor junto con la metodología del cálculo para hallar los tiempos y estados de activación las modulaciones optimizadas.

En el capítulo 4 se realiza una completa descripción de cada módulo empleado para el desarrollo de los algoritmos en el TMS320F28069, resaltando al algoritmo del controlador PI el cual se implementará para el desarrollo del proyecto.
El capítulo 5 se describe a detalle la metodología con la cual se realizaron las pruebas y validaciones de todas las etapas planteadas en el proyecto.

JUSTIFICACIÓN

Los motores de inducción trifásicos son ampliamente utilizados en la industria[1], y generalmente estos necesitan de condiciones de velocidad especificadas por la aplicación, para lograr este objetivo se requiere de un elemento encargado de modificar las variables eléctricas que se reflejen en la modificación de la velocidad de operación [1]. Los elementos encargados de modificar las magnitudes eléctricas han evolucionado gracias a las investigaciones realizadas en las últimas décadas, motivadas por los avances tecnológicos en los semiconductores, que han provocado el desarrollo de nuevas topologías de convertidores y métodos de control los cuales no son elementos industriales actuales debido a que se encuentran en etapa de desarrollo[2].

El cambio en los elementos finales de control, implican ventajas en términos de calidad de la energía, eficiencia energética, conservación de los equipos, causando que el control de la máquina de inducción desarrolle metodologías complejas de perfeccionamiento, por lo cual la reducción del contenido armónico es un tema de investigación en los sistemas de control del motor de inducción[3].

Otra perspectiva del problema es el aumento del consumo energético y el agotamiento de combustibles fósiles que causan el aumento del valor de la energía eléctrica, en este campo existen alternativas de utilización de fuentes de generación no convencionales como los sistemas de generación fotovoltaicos, pero para aplicaciones de motores los sistemas necesitan de un sistema de arranque por las limitaciones di/dt, un inversor, un variador de frecuencia, y opcionalmente equipos que mejoren la calidad de la energía, los cuales se pueden compactar en un sistema de control de velocidad controlado por sistemas embebidos, adoptando un convertidor que fue diseñado para la aplicación descrita incorporando las nuevas topologías desarrolladas y una técnica de control de velocidad aplicada [3].

Por ello esta investigación busca realizar un control de velocidad adoptando un convertidor multinivel con contenido armónico optimizado donde se aplicará la técnica de control optima incorporada un sistema embebido[4], considerando la calidad de la energía que alimentará la máquina brindando así condiciones que permitan preservar la vida útil y obtener ventajas de la utilización de los sistemas de generación de energía no convencional con tecnología en desarrollo que ha tiene imperfectos como la regulación de los niveles de voltaje[2]. Al utilizar técnicas de control implementadas en un TMS320F28069M se logran obtener grandes ventajas en cuanto al procesamiento de datos de entrada dado que cada dato debe ser multiplicado,

sumado y además de eso transformado de acuerdo a fórmulas complejas, esto se logra ya que están diseñados para tareas de altas prestaciones repetitivas y numéricamente intensas, las grandes velocidades que nos brindan estos dispositivos permiten que se realicen de forma muy eficiente el tratamiento, además que nos permiten tener control de muchas variables de nuestro sistema a controlar limitado únicamente por nuestro dispositivo a usar[5], el bajo costo y grades prestaciones de estos dispositivos han aumentado su uso en los últimos años por las prestaciones antes descritas[6].

DELIMITACIONES

Objetivo General

Implementar un algoritmo para el control de velocidad de un motor de inducción mediante TMS320F28069M.

Objetivos Específicos

- Determinar la metodología de control y las características del TMS320F28069M.
- Diseñar el circuito de acondicionamiento de los sensores que registraran las variables a medir en el motor y la carga.
- Implementar la técnica de control para el motor de inducción adoptando el convertidor multinivel trifásico y las modulaciones optimizadas.
- Desarrollar los algoritmos en el TMS320F28069M para el control, y la implementación de modulaciones con contenido armónico optimizado y la adquisición de datos.
- Validar el funcionamiento del prototipo desarrollado, visualizándolo en una interfaz para el monitoreo on-line

ACOTACIONES

- Se adoptará un convertidor multinivel trifásico con contenido armónico optimizado.
- Los sensores que registraran las variables a medir tanto eléctricas como físicas, serán sensores que se encuentren en los laboratorios y grupos de investigación.
- El algoritmo de control de velocidad que se implementará se definirá de acuerdo a las características del TMS320F28069M.
- Los ángulos de disparo optimizados para las modulaciones serán previamente encontrados con una herramienta computación la cual utilizo un algoritmo genético.
- Se aplicará una estrategia de control óptima ya desarrollada en la literatura implementándola en el TMS320F28069M.
- La validación del funcionamiento del control de velocidad se realizará con instrumentos de medición presentes en la universidad.

1. GENERALIDADES DEL DISPOSITIVO TMS320F28069

La tarjeta dispuesta para la ejecución de este proyecto es el elemento fundamental en el sistema de control, ya que es el encargado de las tareas más importantes para el control del sistema. Dichas tareas son la adquisición de datos para su posterior procesamiento control y la generación de señales optimizadas para obtener el menor contenido armónico en la salida del inverso poder llevar a cabo un algoritmo de control de velocidad implementado por el usuario para la máquina trifásica [7]. La norma IEC 61000-3-2 define los componentes armónicos que una carga electrónica puede inyectar en la línea de suministro, lo que obliga a los fabricantes a minimizar los armónicos de entrada de bajo orden (h<50). A continuación, se explica cómo el microcontrolador Piccolo ™ ilustración 1.1 de bajo costo de 32 bits de Texas Instruments ofrece una solución para controlar la velocidad de un motor. El sistema general está pensado para mejorar la eficiencia, minimizar el contenido de armónicos de la corriente de entrada, regular con precisión el bus de CC y ahorrar tiempo de desarrollo[8].

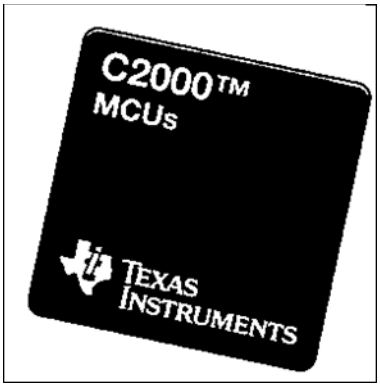


Ilustración 1. 1 Microcontrolador Piccolo ™ Fuente: [8]

1.1 C2000™ Piccolo™ 32-bit MCU

La familia C2000 de microcontroladores de alto rendimiento diseñados específicamente para controlar la electrónica de potencia y proporcionar procesamiento avanzado de señales digitales en aplicaciones industriales y automotrices. Una gama de microcontroladores y dispositivos procesadores de señales líderes en control analógico y digital, los MCU C2000 han evolucionado para

proporcionar una mejor preciso, procesamiento potente y actuación Premium para permitir crear los sistemas de control de energía más eficientes del mundo, con diferentes aplicaciones las cuales se observan en la ilustración 1.2 [9].

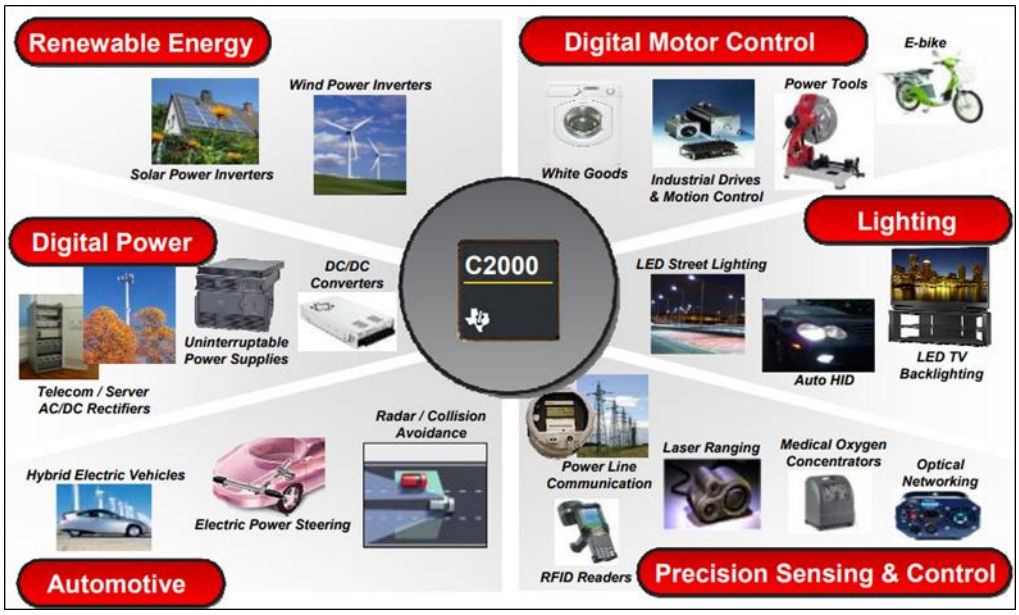


Ilustración 1. 2 Aplicación de los C2000 Fuente: [9]

En esencia, las MCU C2000 se basan en el núcleo DSP TMS320C28x de 32 bits, que presenta multiplicaciones de hardware de 32 × 32 bits de un solo ciclo y ejecución de instrucción atómica de ciclo único. El núcleo C28x también incluye aceleradores de hardware como la unidad matemática compleja Viterbi para algoritmos de comunicación de líneas de alimentación y la unidad de matemática trigonométrica para acelerar funciones trigonométricas comunes en muchos algoritmos de control de motores. El coprocesador en tiempo real, también conocido como CLA, proporciona una CPU independiente capaz de manejar tareas independientemente del núcleo C28x principal. Aumente el ancho de banda del núcleo C28x descargando tareas intensivas en matemática al CLA, en la ilustración 1.3 se puede apreciar en la de organización del núcleo de C28x[9].

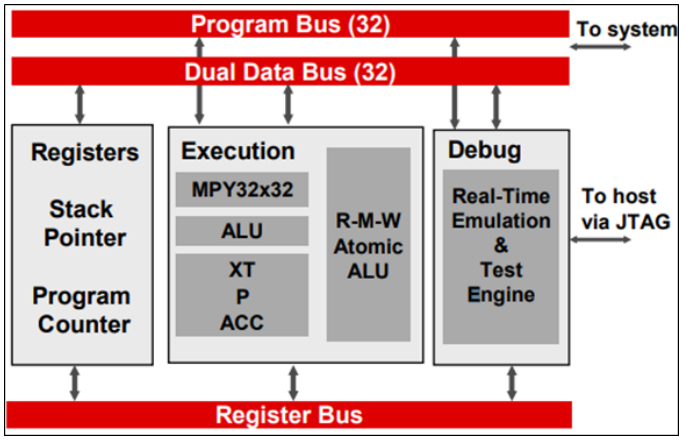


Ilustración 1. 3 Core C28x Fuente: [9].

1.1.1 Control en tiempo real C2000

Los microcontroladores de control en tiempo real C2000 ilustración 1.4, utilizan un núcleo propietario de 32 bits, la CPU C28x, que ofrece operaciones de un solo ciclo y hasta 300 MIPS junto con un bus de periféricos y gestión de interrupciones altamente optimizadas. Con potentes periféricos integrados como se aprecia en la siguiente ilustración, estas soluciones de control de chip único en tiempo real están diseñadas para una variedad de aplicaciones de control [9].

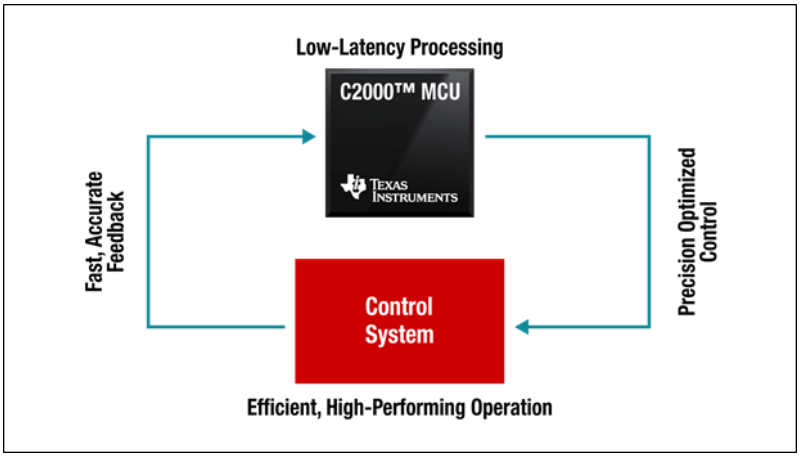


Ilustración 1. 4 Aplicación para control en tiempo real Fuente: [9]

El núcleo C28x optimizado para matemáticas brinda a los diseñadores la flexibilidad para mejorar la eficiencia y la confiabilidad del sistema. Como un cruce entre un

microcontrolador y un procesador de señal digital, los controladores en tiempo real C2000 traen la densidad de código y la velocidad de ejecución de un DSP con la facilidad de uso y la accesibilidad de un microcontrolador [9].

1.1.2 Aceleradores

La combinación de núcleo de la CPU C28x de alto rendimiento, se puede realizar una potencia de procesamiento rápida y eficiente para sistemas de control complejos en tiempo real como se puede apreciar en la ilustración 1.5. Hay cuatro aceleradores de hardware integrados en el chip disponibles que aumentan considerablemente el rendimiento de la MCU C2000 [9]:

- Acelerador de punto flotante
- Unidad de matemática compleja y unidad CRC (VCU)
- Unidad de matemáticas trigonométricas (TMU) acelerador
- Acelerador de la ley de control (CLA)

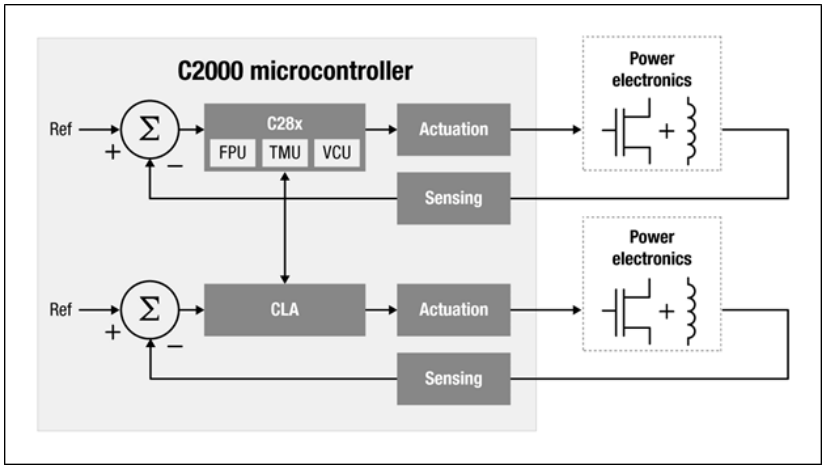


Ilustración 1. 5 Aceleradores de leyes de control CLA. Fuente: [9]

Los microcontroladores TMS320F28069M cuentan con un núcleo C28x y el CLA (Acelerador de leyes de control), junto con periféricos de control altamente integrados en dispositivos de bajo número de pines. Es compatible con el código con el código basado en C28x, contando también con un alto nivel de integración analógica [10].

Entre los módulos destacables cuenta con HRPWM para permitir el control de borde doble (modulación de frecuencia), comparadores analógicos con referencias internas

de 10 bits y se pueden enrutar directamente para controlar las salidas de ePWM. El ADC convierte de 0 a 3.3 V rango fijo de escala completa y admite referencias V_{REFLA} / V_{REFLO}. El módulo ADC se ha optimizado para bajos gastos generales y latencia [10].

1.2 Nomenclatura TMS320F28069

Los dispositivos TMS y las herramientas de apoyo al desarrollo de TMDS se han caracterizado completamente, y la calidad y confiabilidad del dispositivo se han demostrado completamente. En la siguiente Ilustración 1.6 muestra la caracterización del dispositivo F28069M definiendo el significado de cada marca. La revisión del dispositivo puede determinarse mediante los símbolos marcados en la parte superior del paquete [11].

Ilustración 1. 6 Microcontrolador TMS320F28069M

- C=Código de fabrica
- A= Esta revisión de Silicon está disponible como TMS.
- 39A78JW = Código de seguimiento del lote
- 39= Código de año / mes de 2 dígitos
- A78J= lote de ensamblado
- W=código de sitio de ensamble
- G4=Verde (bajo en halógenos y compatible con RoHS).

Tabla 1. 1 Nomenclatura TMS320F28069M fuente: [1]

Prefijo	*Familia De Dispositivo*	*Tecnología*	*Dispositivo*	*Tipo De Paquete*	*Rango De Temperatura*
Dispositivo Comprobado y calificado	Familia de MCU TMS320	Flash	28069M	Paquete plano cuádruple de perfil bajo PZ de 100 pines	−40°C a 105°C
TMS	320	F	28069M	PZ	T

1.2.1 Descripción y características generales del MCU

En la tabla 1.2 se realiza una descripción grandes rasgos acerca de las cualidades principales que comprende el MCU TMS320F28069M.

Tabla 1. 2 Parámetros Y Características del TMS320f28069M

CARACTERÍSTICA	**28069M (90 MHz)**	**Descripción general del dispositivo**	
Tipo de paquete	**100-Pin PZ**	**Características**	
Ciclo de instrucción	11.11 ns	***• CPU de 32 bits de alta eficiencia (TMS320C28x)*** -90 MHz (11.11-ns Cycle Time) -16 × 16 y 32 × 32 Operaciones de multiplicación y acumulación (MAC) -16 × 16 Dual MAC -Bus de Arquitectura Harvard -Operaciones atómicas -Rápida respuesta de interrupción y procesamiento -Modelo de Programación de Memoria Unificada -Código Eficiente (en C / C ++ y Assembly) ***• Unidad de punto flotante (FPU)*** -Operaciones de punto flotante de precisión única nativa ***• Acelerador de Ley de Control Programable (CLA)*** -Acelerador matemático de punto flotante de 32 bits	***• Bloque de expansión de interrupción periférica (PIE) que admite todas las interrupciones periféricas*** ***• Tres temporizadores de CPU de 32 bits*** ***• Periféricos de control avanzado*** ***• Hasta 8 módulos de modulador de ancho de pulso mejorado (ePWM)*** -Total de 16 canales PWM (8 HRPWM-Capable) -Temporizador independiente de 16 bits en cada módulo ***• Tres módulos de captura mejorada de entrada (eCAP)*** ***• Hasta 4 módulos de captura de alta resolución (HRCAP)*** ***• Hasta 2 módulos de pulsos de codificador de cuadratura mejorados (eQEP)*** ***• Convertidor de analógico a digital (ADC) de 12 bits, doble muestreo y retención (S / H)*** -Hasta 3.46 MSPS -Hasta 16 canales. ***• Sensor de temperatura en chip***
Unidad de punto flotante (FPU)	Si		
VCU	Si		
CLA	Si		
6-canales DMA	Si		
On-chip Flash (16-bit word)	128K		
On-chip SARAM (16-bit word)	50K		
Código de seguridad para on-chip Flash, SARAM, and OTP blocks	Si		
Boot ROM (32K × 16)	Si		
One-time programable (OTP) ROM (16-bit word)	1K		
Canales ePWM	16		
Canales ePWM de alta resolución	8		
Entradas eCAP	3		
HRCAP	4		

Módulos eQEP		2	-Ejecuta código independientemente de la CPU principal • ***Viterbi, Matemáticas complejas, Unidad CRC (VCU)*** -Extiende el conjunto de instrucciones C28x para soportar la multiplicación compleja, las operaciones de Viterbi y la verificación de redundancia cíclica (CRC) • ***Memoria integrada*** Hasta 256 KB de Flash - Hasta 100 KB de RAM -2 KB de ROM programable de una sola vez (OTP) • ***Acceso directo a la memoria de 6 canales (DMA)*** • ***Bajo costo del dispositivo y del sistema*** -Suministro individual de 3.3 V -Sin requisito de secuencia de poder -Reinicio de encendido integrado y reinicio de reducción de tensión -Modos de operación de bajo consumo -Sin pin de soporte analógico • ***Soporte de exploración de límites JTAG*** -IEEE Standard 1149.1-1990 Puerto de acceso de prueba estándar y arquitectura de exploración de límites • ***Reloj*** -Dos osciladores internos de cero pin	• ***Llave de seguridad de 128 bits y bloqueo*** -Protege los bloques de memoria segura -Previene la ingeniería inversa de firmware • ***Periféricos de puerto serie*** -Dos módulos de interfaz de comunicaciones serie (SCI) [UART] -Dos Módulos de Interfaz Periférica Serial (SPI) -Un bus de circuito integrado (I2C) -Un bus de puerto serie con almacenamiento intermedio multicanal (McBSP) -Una red de área de controlador mejorada (eCAN) -Bus serie universal (USB) 2.0 (consulte la Tabla de comparación de dispositivos para conocer la disponibilidad) -Modo de dispositivo de velocidad completa - Modo host de velocidad completa o velocidad baja • ***Hasta 54 pines de entrada / salida de uso general (GPIO) multiplexados y individualmente programables con filtrado de entrada*** • ***Funciones avanzadas de emulación*** -Análisis y funciones de punto de interrupción. -Depuración en tiempo real a través de hardware • ***Opciones de paquete*** -PZ PowerPAD ™ de 100 pines -Mochilas planas de bajo perfil (LQFP) de 100 pines PZ • ***Temperatura*** -T: –40 ° C a 105 ° C
12-Bit ADC	MSPS	3.46		
	Tiempo de Conversión	289 ns		
	Canales	16		
	Sensor de temperatura	Si		
	Dos canales de muestreo y retención	Si		
32-Bit CPU Timers		3		
Comparadores con DACs integrados		3		
I2C		1		
McBSP		1		
eCAN		1		
SPI		2		
SCI		2		
USB		1		
2-pin Oscilador		1		
0-pin Oscilador		2		
Pines de E / S (compartidos)	GPIO	54		
	AIO	6		
Interrupciones externas		3		
Tensión de alimentación (nominal)		3.3 V		
Opciones de temperatura	T: –40°C-105°C	PZ		

1.2.2 Diagrama en bloques TMS320F28069M

En la ilustración 1.7 se muestra un diagrama de bloques funcional del dispositivo donde se puede observar la integración de cada una de las partes antes descritas y su interrelación con el resto de sistemas que comprende el dispositivo.

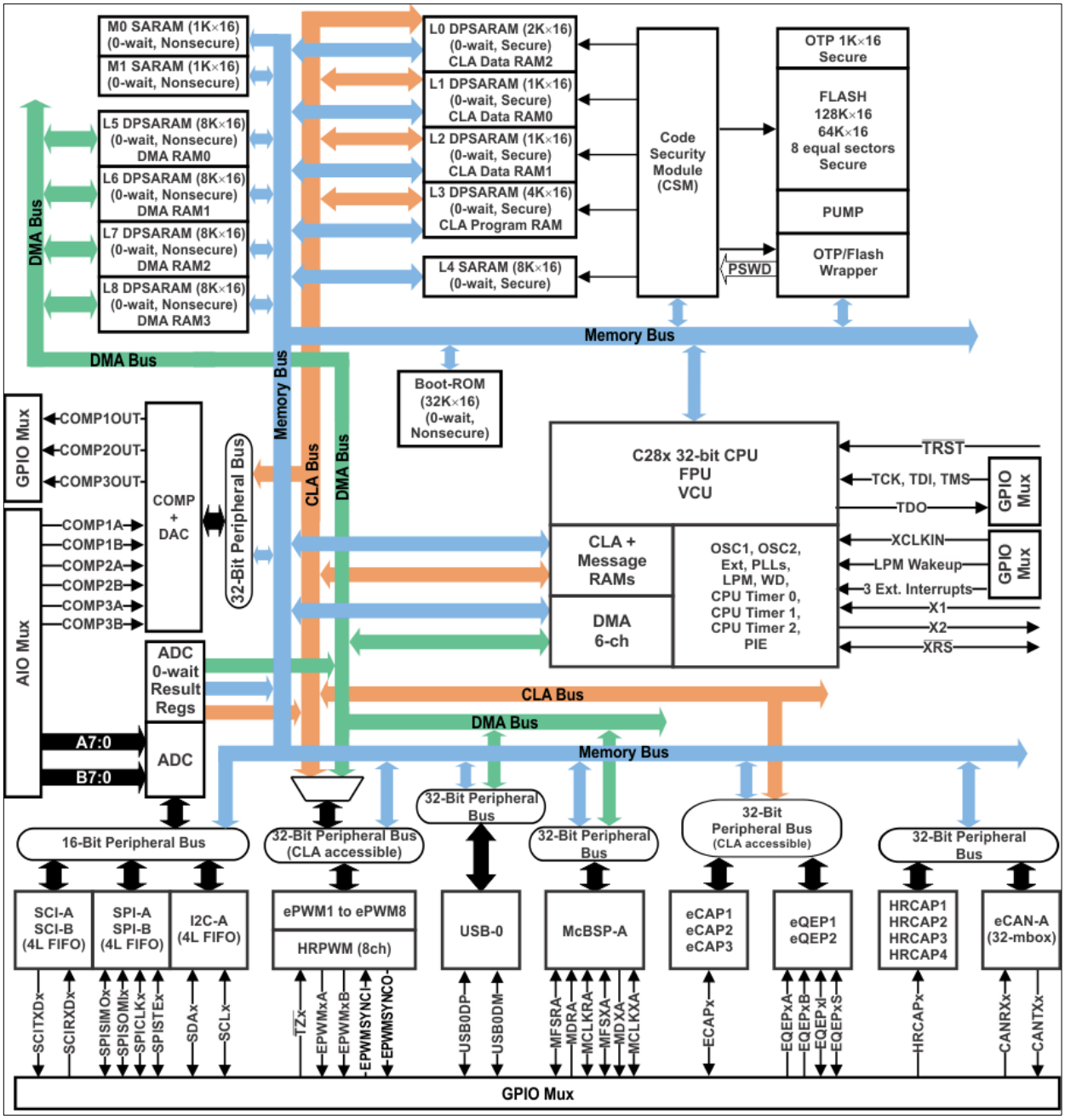


Ilustración 1. 7 Diagrama de bloques funcionales TMS320F28069M. Fuente: [12]

1.2.3 Vista superior TMS320F28069M

En la ilustra 1.9 se puede identificar la organización y ubicación de los pines que tiene el microcontrolador seleccionado para la aplicación donde se pueden observar los distintos periféricos ya descritos en la tabla anterior.

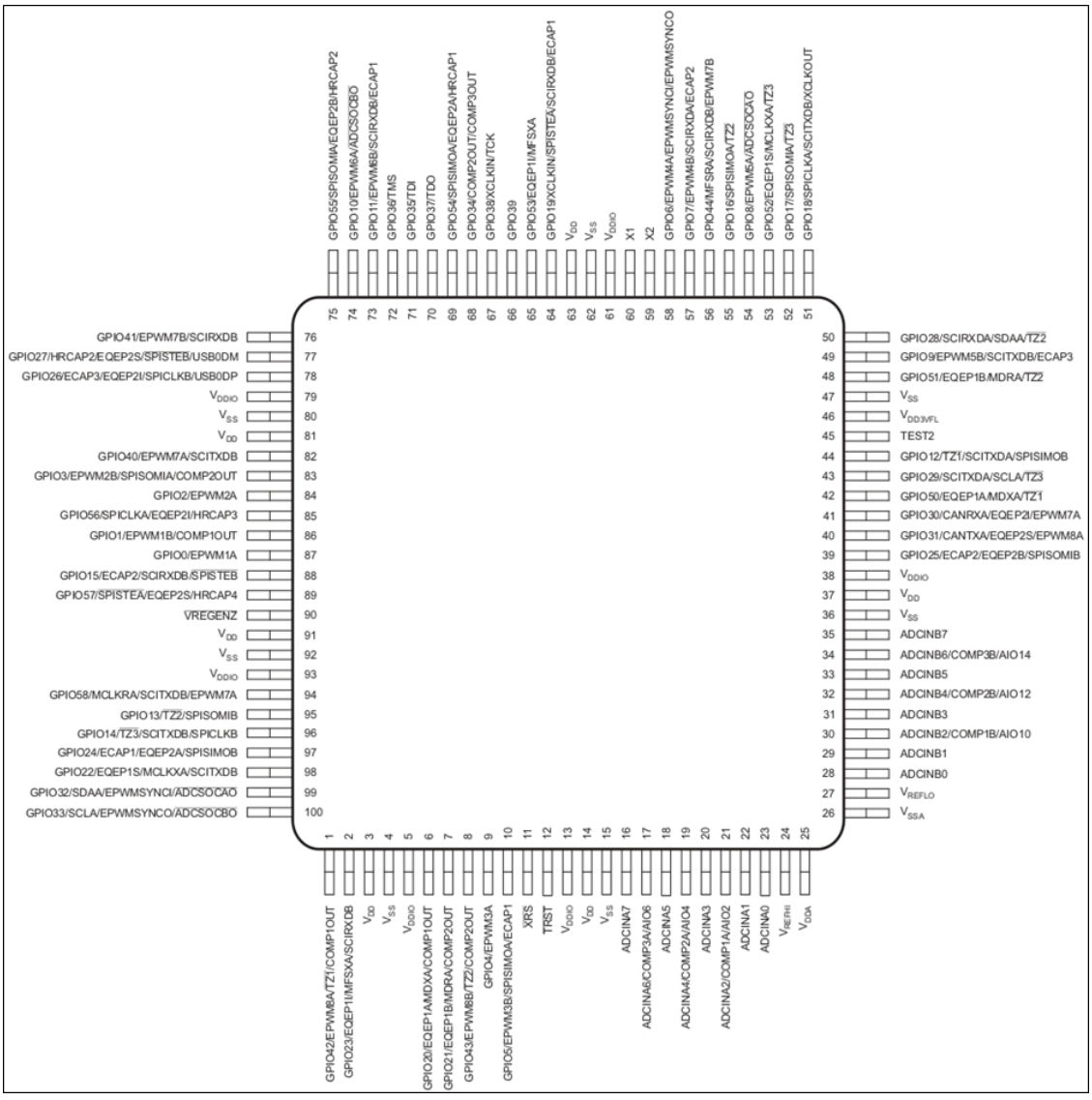


Ilustración 1. 8 Parámetros Y Características del TMS320f28069M

1.3 Placa de desarrollo

1.3.1 LAUNCHXL-F28069M

El C2000 ™ Piccolo ™ LaunchPad ™, LAUNCHXL-F28069M, es una placa de desarrollo completa de bajo costo para los dispositivos Texas Instruments Piccolo F2806x. El kit LAUNCHXL-28069M seleccionado presenta alta eficiencia de hardware y software necesarios para desarrollar aplicaciones en el control de motores donde se pueden evaluar desarrollo a nivel académico u investigativo. El LaunchPad se basa en el dispositivo F28069M el cual permite la migración fácilmente a dispositivos F2806x de menor costo una vez que se termine la aplicación para la cual fue seleccionada en este caso el control del motor de inducción. Ofrece una herramienta de emulación JTAG incorporada que permite la interfaz directa a una PC para una fácil programación, depuración y evaluación. Además de la emulación JTAG, la interfaz USB proporciona una conexión serie receptor / transmisor asíncrono universal (UART) desde el dispositivo F2806x al equipo principal[12].

Como se muestra en la ilustración 1.19 se muestran las funciones de LAUNCHXL-F28069M C2000 entra las cuales incluye[12] :

- Interfaz de programación y depuración USB a través de una sonda de depuración XDS100v2 aislada galvánicamente de alta velocidad con conexión USB / UART.
- Dispositivo conjunto F28069M que permite que las aplicaciones migren fácilmente a dispositivos de menor costo.
- Dos LED’s de usuario.
- Botón de reinicio del dispositivo.
- Pines de dispositivo de fácil acceso para propósitos de depuración o como enchufes para agregar tableros de extensión personalizados.
- La biblioteca InstaSPIN en ROM, que permite la implementación de las soluciones InstaSPIN-MOTION e InstaSPIN-FOC.
- Interfaces de codificador dual de 5 V en cuadratura.
- Interfaz CAN con transceptor integrado.
- Botones de selección de arranque

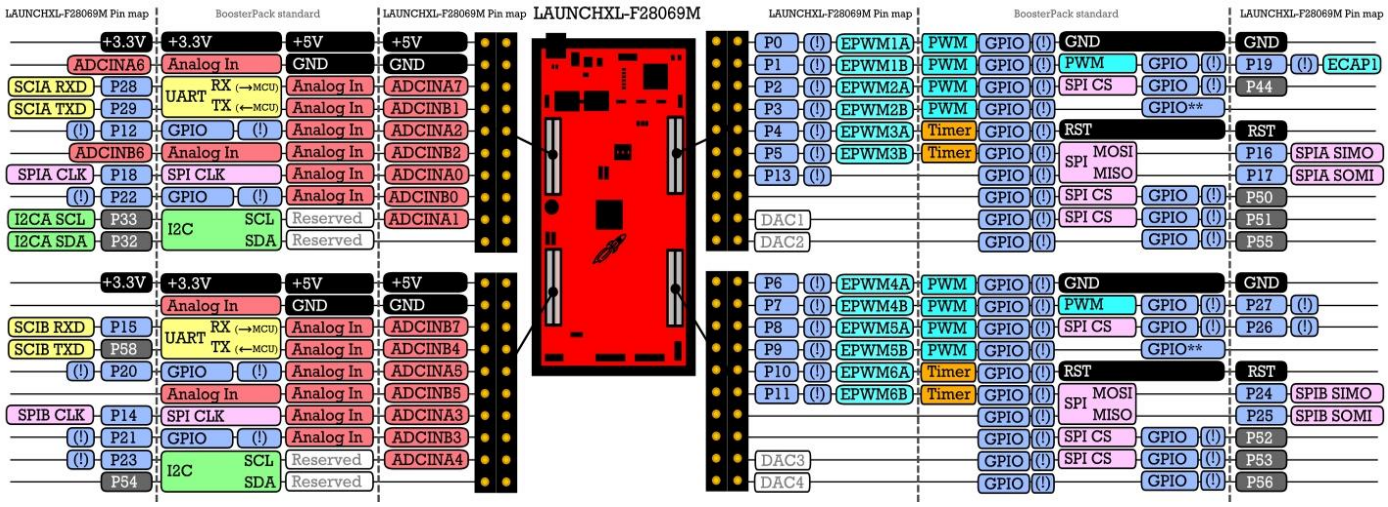


Ilustración 1. 9 Vista general LAUNCHXL-F28069M. Fuente: [12]

1.3.2 Disposición

Para Launchpad F28069M obtenga su mayor rendimiento y mejor eficiencia a nivel software se programa con Code Composer Studio debido a que las modulaciones optimizada que se explicaran más adelante necesitan una eficiencia alta de código. Texas instruments ofrece también una serie de herramienta para desarrollar aplicaciones en la herramienta computacional Matlab, pero no se aplicó por las razones antes expuestas.

1.3.3 Code compositor studio

Code Composer Studio es un entorno de desarrollo integrado (IDE) que admite la cartera de Microcontroladores y procesadores integrados de TI ilustración 1.10. Code Composer Studio comprende un conjunto de herramientas que se utilizan para desarrollar y depurar aplicaciones integradas. Incluye un compilador C / C ++ optimizador, un editor de código fuente, un entorno de compilación de proyectos, un depurador, un generador de perfiles y muchas otras características. El IDE intuitivo proporciona una única interfaz de usuario que lo lleva a través de cada paso del flujo de desarrollo de la aplicación [13].

Ilustración 1. 10 Code composer studio. Fuente: [13]

1.3.4 Configuración del hardware.

Para configurar la placa F28069M LaunchPad de acuerdo a la aplicación a desarrollar se realiza de acuerdo a la siguiente tabla[12].

- **Dominio de la energía.**

El LaunchPad F28069M tiene varios dominios de alimentación diferentes para habilitar el aislamiento de JTAG tabla 1.3. Los puentes JP1, JP2, JP3, JP4 y JP5 configuran dónde se pasa la alimentación[12].

Tabla 1. 3 Descripción del jumper JPx

Jumper	Dominio de poder
JP1	Habilitar 3.3 V desde USB (desactiva el aislamiento).
JP2	Habilitar GND desde USB (deshabilita el aislamiento).
JP3	Habilitar el conmutador de 5 V (apagado del suministro de 3,3 V del dispositivo de destino).
JP4	Conecta el objetivo MCU 3.3 V al segundo conjunto de encabezados Booster Pack.
JP5	Conecta la MCU de destino 5V al segundo conjunto de encabezados de Booster Pack.

- **Conectividad en serie.**

El LAUNCHXL-F28069M tiene un adaptador USB a UART incorporado. Esto facilita la impresión de la información de depuración en la PC host incluso en entornos aislados. El dispositivo F28069M en este LaunchPad contiene dos periféricos SCI

(UART), mientras que el LaunchPad tiene tres lugares donde se deben enrutar estos periféricos. El enrutamiento se configura a través de dos puentes (JP6 y JP7) tabla 1.4. Configure los puentes como se muestra en la Tabla para la conectividad en serie que desea[12].

Tabla 1. 4 Serial Connectivity fuente

MUX_SEL (JP7)	CH_SEL (JP 6)	Función
ON	ON	USB / UART desactivado; J1.3 y J1.4g –GPIO 28 n GPIO 29; J7.3 y J7.4 –GPIO15 y GPIO 58.
ON	OFF	USB / UART - GPIO 28 n GPIO 29, J1.3 y J1.4 - Hi-Z; J7.3 y J7.4 - GPIO15 y GPIO 58.
OFF	ON	USB / UART - GPIO15 y GPIO58; FALLA / OCTW - GPIO28 y GPIO29; J7.3 y J7.4 - Hi-Z.
OFF	OFF	USB / UART - GPIO15 y GPIO58; FALLA / OCTW - GPIO28 y GPIO29; J7.3 y J7.4 - Hi-Z.

- **Selección del modo de arranque.**

Para la selección del modo de arranque del F28069M LaunchPad el cual incluye una ROM de inicio que realiza algunas verificaciones básicas de inicio y permite que el dispositivo arranque de muchas maneras diferentes, se ha proporcionado S1 apreciable en la ilustración 1.11, para permitir a los usuarios configurar fácilmente los pines que la ROM de inicio verifica para tomar esta decisión[12].

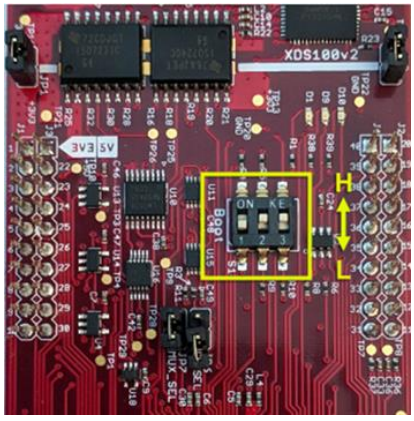


Ilustración 1. 11 Orientación del interruptor de arranque. Fuente: [12]

Los modos de arranque que se muestran en la siguiente Tabla 1.5 donde se pueden seleccionar usando S1:

Tabla 1. 5 Configuración del modo de arranque S1

El modo de inicio.	S1-Switch 1 (GPIO34) H = Pulled to 1 L = Pulled to 0	S1-Switch 2 (GPIO37 / DO) H = Pulled to 1 L = Pulled to 0	S1-Switch 3 (TRSTn) H = XDS100v2 (1) L = Tied to 0
Emulación de arranque.	L	H	H
IO paralelo.	L	L	L
SCI	H	L	L
Espera	L	H	L
Obtener modo	H	H	L

1.3.5 Hardware LAUNCHXL-F28069M

Pines dispuestos físicamente.

La organización física de los pines de conexión del hardware se puede observar en las tablas 1.6, 1.7, 1.8, y 1.9. Del anexo 3 al anexo 9 se muestran los todos los circuitos que comprende el hardware launchxl-F28069M[12].

Tabla 1. 6 J1 Pin y J3 Pin

Mux Value				J1 Pin	J3 Pin	Mux Value			
3	2	1	0			0	1	2	3
			+3.3V	1	21	+5V			
			ADCINA6	2	22	GND			
			J1.3	3	23	ADCINA7			
			J1.4	4	24	ADCINB1			
SPISIMOB	SCITXDA	TZ1	GPIO12	5	25	ADCINA2			
			ADCINB6	6	26	ADCINB2			
XCLKOUT	SCITXDB	SPICLKA	GPIO18	7	27	ADCINA0			
SCITXDB	MCLKXA	EQEP1S	GPIO22	8	28	ADCINB0			
ADCSOCBO	EPWMSYNCO	SCLA	GPIO33	9	29	ADCINA1			
ADCSOCAO	EWPMSYNCI	SDAA	GPIO32	10	30	NC			

Tabla 1. 7 J4 Pin y J2 Pin.

Mux Value				J4 Pin	J2 Pin	Mux Value			
3	2	1	0			0	1	2	3
Rsvd	Rsvd	EPWM1A	GPIO0	40	20	GND			
COMP1OUT	Rsvd	EPWM1B	GPIO1	39	19	GPIO19	SPISTEA	SCIRXDB	ECAP1
Rsvd	Rsvd	EPWM2A	GPIO2	38	18	GPIO44	MFSRA	SCIRXDB	EPWM7B
COMP2OUT	SPISOMIA	EPWM2B	GPIO3	37	17	NC			
Rsvd	Rsvd	EPWM3A	GPIO4	36	16	RESET#			
ECAP1	SPISIMOA	EPWM3B	GPIO5	35	15	GPIO16	SPISIMOA	Rsvd	TZ2
SPISOMIB	Rsvd	TZ2	GPIO13	34	14	GPIO17	SPISOMIA	Rsvd	TZ3
			NC	33	13	GPIO50	EQEP1A	MDXA	TZ1
			DAC1	32	12	GPIO51	EQEP1B	MDRA	TZ2
			DAC2	31	11	GPIO55	SPISOMIA	EQEP2A	HRCAP1

Tabla 1. 8 J5 Pin y J7 Pin

Mux Value				J5 Pin	J7 Pin	Mux Value			
3	2	1	0			0	1	2	3
			+3.3V	41	61	+5V			
			NC	42	62	GND			
			J7.3	43	63	ADCINB7			
			J7.4	44	64	ADCINB4			
COMP1OUT	MDXA	EQEP1A	GPIO20	45	65	ADCINA5			
			NC	46	66	ADCINB5			
SPICLKB	SCITXDB	TZ3	GPIO14	47	67	ADCINA3			
COMP2OUT	MDRA	EQEP1B	GPIO21	48	68	ADCINB3			
SCIRXDB	MFSXA	EQEP1I	GPIO23	49	69	ADCINA4			
HRCAP1	EQEP2A	SPISIMOA	GPIO54	50	70	NC			

Tabla 1. 9 J8 Pin y J6 Pin

Mux Value				J8 Pin	J6 Pin	Mux Value			
3	2	1	0			0	1	2	3
Rsvd	Rsvd	EPWM4A	GPIO6	80	60	GND			
COMP1OUT	Rsvd	EPWM4B	GPIO7	79	59	GPIO27	HRCAP2	EQEP2S	SPISTEB
Rsvd	Rsvd	EPWM5A	GPIO8	78	58	GPIO26	ECAP3	EQEP2I	SPICLKB
COMP2OUT	Rsvd	EPWM5B	GPIO9	77	57	NC			
Rsvd	Rsvd	EPWM6A	GPIO10	76	56	RESET#			
ECAP1	Rsvd	EPWM6B	GPIO11	75	55	GPIO24	ECAP1	EQEP2A	SPISIMOB
			NC	74	54	GPIO25	ECAP2	EQEP2B	SPISOMIB
			NC	73	53	GPIO52	EQEP1S	MCLKXA	TZ3
			DAC3	72	52	GPIO53	EQEP1I	MFSXA	Rsvd
			DAC4	71	51	GPIO56	SPICLKA	EQEP2I	HRCAP3

1.4 Metodología del Control escalar a implementar

El control de un motor de inducción es complejo y exigen el uso de algoritmos de control de alto rendimiento como el "control vectorial" y un potente microcontrolador para ejecutar este algoritmo. Control escalar es el término usado para describir una forma más simple de control de motor, utilizando esquemas de control no controlados por vector. Un motor ACI puede ser conducido a un estado estable mediante simples esquemas alimentados por voltaje, controlados por corriente o controlados por velocidad[14].

La variable escalar se puede manipular después de obtener su valor, ya sea mediante medición directa o cálculo, y se puede utilizar en formatos de retroalimentación de bucle abierto y cerrado. Aunque su comportamiento transitorio no es ideal, un sistema escalar conduce a una respuesta satisfactoria en estado estable[14].

1.4.1 Desarrollo básico control de la velocidad por variación de la frecuencia ley escalar

Para una maquina asincrónica como la descrita anteriormente se sabe que velocidad sincrónica, en radianes por segundo, del campo magnético giratorio de un motor de inducción trifásico está dada por[14]:

$$Ws = \frac{2.\pi . f1}{\left(\frac{P}{2}\right)} \quad P = numero\ de\ polos \qquad (1.1)$$

La velocidad de la maquina asíncrona que presenta en el rotor viene dada por la siguiente ecuación teniendo en cuenta el deslizamiento presente en este tipo de máquinas viene dada por las siguientes ecuaciones.

$$wm = WS.(1 - s) \qquad (1.2)$$

$$wm = \frac{2.\pi . f1}{\left(\frac{P}{2}\right)}.(1 - s) \qquad (1.3)$$

$$nm = \frac{120.f1}{P}.(1 - s)\ [Rpm] \qquad (1.4)$$

Donde s es el deslizamiento del motor el cual, en condiciones estables de trabajo, varía generalmente entre 0.005 y 0.1. Quiere decir que si se varía la frecuencia de trabajo del motor de inducción puede variarse su velocidad de rotación. Para controlar por debajo de la velocidad base debe reducirse la frecuencia $f1$. Ahora bien, recuérdese que la *FEM* inducida en el devanado del estator depende de la frecuencia según la relación siguiente[14]:

$$ES = 4{,}44.f1.wS\ .Kdevs\ .\phi m \qquad (1.5)$$

Por otra parte, del circuito equivalente del motor se obtiene que el fasor del voltaje aplicado en el estator por fase de acuerdo al modelos simplificado de la máquina de inducción es igual a[14]:

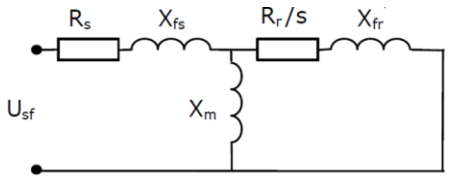


Ilustración 1. 12 . Circuito equivalente del motor de inducción. Fuente: [15]

$$US = ES + jI\ S\ .(RS + jX\ S) \qquad (1.6)$$

A frecuencias cercanas a la nominal, la caída en la impedancia interna del estator es despreciable y puede suponerse aproximadamente que es igual a la FEM. [14]:

$$US \approx ES \qquad (1.7)$$

Quiere esto decir que, si la frecuencia disminuye y el voltaje no varía, el flujo ϕm aumenta provocando la saturación de la máquina y el consiguiente aumento de la corriente magnetizaste y las pérdidas del motor a valores inadmisibles. Por lo tanto, es recomendable operar de forma tal que, al reducirse la frecuencia, el voltaje se reduzca proporcionalmente, o sea con la ley de Mando Voltaje / Frecuencia Constante:

$$\frac{U_s}{f_1} = K_{U.f} \qquad (1.8)$$

En la práctica, la relación de tensión a frecuencia del estator se basa generalmente en los valores nominales de estas variables. El perfil típico de $\frac{V}{Hz}$ se puede mostrar en la siguiente figura. Básicamente, hay tres rangos de velocidad en el perfil de $\frac{V}{Hz}$ de la siguiente manera[15]:

- A $0 - fc$ Hz, se requiere un voltaje, por lo que la caída de voltaje a través de la resistencia del estator no puede descuidarse y debe compensarse aumentando los Vs. Entonces, el perfil $\frac{V}{Hz}$ no es lineal. La frecuencia de corte (fc) y los voltajes de estator adecuados se pueden calcular analíticamente a partir del circuito equivalente de estado estable con $Rs \neq 0$.
- En $fc - frated\ Hz$, sigue la relación constante $\frac{V}{Hz}$. La pendiente en realidad representa la cantidad de flujo de espacio de aire como se ve en la Ecuación.
- Con Hz de frecuencia más alta, la relación $\frac{V}{Hz}$ constante no se puede satisfacer porque los voltajes del estator se limitarían al valor nominal para evitar la ruptura del aislamiento en los devanados del estator. Por lo tanto, el flujo de espacio de aire resultante se reduciría, y esto provocará inevitablemente la disminución del par desarrollado en consecuencia. Esta región es usualmente llamada "región de debilitamiento de campo".

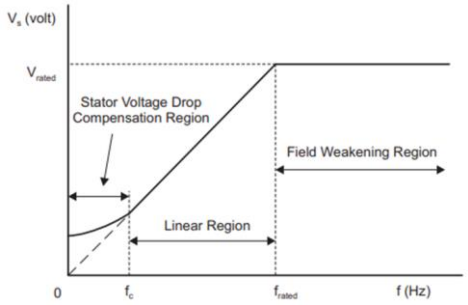


Ilustración 1. 13 Ley de control escalar. Fuente: [16].

2. MEDICIÓN VELOCIDAD Y ACONDICONAMIENTO

2.1 Medida de velocidad del motor de induccion

La medida de la velocidad del motor de induccion se realiza a traves de un motor de CC presenten en los laboratorios Ilustración 2.1, el cual es un motor de iman permanete que genera voltage DC dependiendo de la velocicdad con que gire su rotor, actuando como transdutor de nuestro sistema convirtiendo velocidad de Rpm a su correspondiente valor en voltios.

Los dos motores se acoplan por el eje del rotor para asi poder realizar la correcta medida de velocidad rotórica de la maquina de induccion.Sus características principales se pueden apreciar a continución.

Ilustración 2. 1 Máquina de inducción. Fuente: Autor

- **Datos técnicos de la máquina de derivación de cc**
  - Tensión y corriente de armadura: 205 V / 2 A
  - Potencia: 0,3 kW, a 2000 rpm.
- **DC-Tacómetro**
  - Salida 1: 10V, a 1500rpm
  - Salida 2: 10V, a 3000rpm

Al realizar una primera captura de los datos de velocidad del motor de inducción mediante una tarjeta de adquisición de datos DAQ6022 en el software Matlab con una tasa de muestro de $20kHz$ funcionando a una velocidad de $1573\ Rpm$, se obtuvieron las muestras a continuación mostradas en la ilustración 2.2:

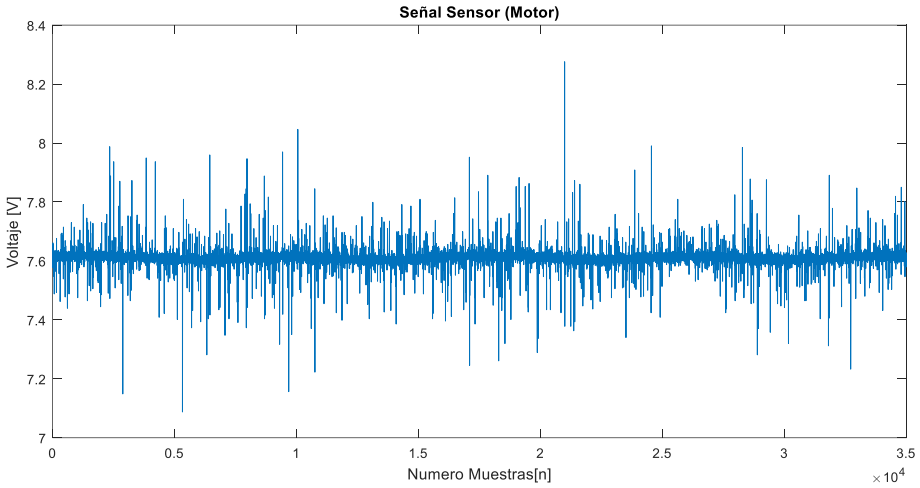


Ilustración 2. 2 Frecuencia de muestreo de 20KHz para una velocidad de 1573Rpm. Fuente: Autor

Como se observa en la ilustración 2.2 los datos de velocidad capturados presentan mucho ruido, con componentes de alta frecuencia pero de baja magnitud además de que la magnitud electrica se encuentra muy elevada al rededor de $7.6V$ impidiendo ser adquirida por la tarjeta F28069M, la cual solo admite señales de $0V\ a\ 3.3\ V$ de acuerdo a esto se hace necesario el diseño e implementación de un circuito analógico que filtre y acondicione las señales.

Al aplicar la transformada discreta del coseno sobre la señal con el fin de tener una base de diseño para el filtro análogo, se puede observar en la Ilustración 2.3 que la frecuencia de corte es más o menos $20Hz$ considerando los limites de operación de los dispositivos electrónicos de tipo pasivo y activo involucrados en el diseño.

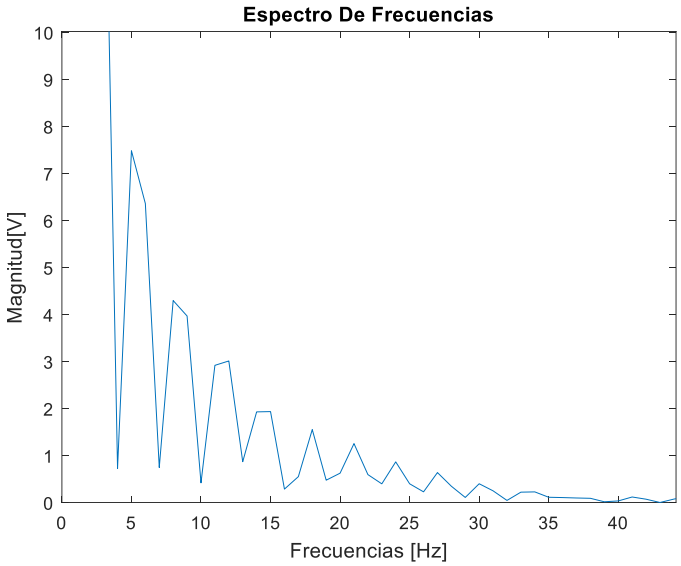


Ilustración 2. 3 Espectro de frecuencias. Fuente: Autor-Matlab

2.2 Circuito de filtrado y acondicionamiento de la señal de entrada

2.2.1 Diseño del filtro analógico pasa baja

Para la correcta captura y análisis de los datos de entrada se hizo necesario el diseño e implementación de un filtro digital pasa bajas con el fin de obtener los datos de entrada al ADCINT0 lo más continuos posible para el correcto funcionamiento del sistema.

El filtro pasa baja se diseñó en base al integrado de amplificadores operacionales TL074CN mostrado en la ilustración 2.4, ya que son operacionales que utiliza tecnología JFET, con la polarización de entrada baja con compensación de corrientes con velocidad de respuesta rápida con tiempo de establecimiento rápido hasta 0.01%: 2 µs. La baja distorsión armónica y el bajo nivel de ruido hacen que el TL074 sea ideal para aplicaciones de filtrado se señales. Cada amplificador cuenta con entradas JFET (para alta impedancia de entrada) acopladas con etapas de salida bipolares integradas en un solo chip monolítico, que al analizar la función de transferencia se deduce una ecuación de frecuencia de corte ecuación 2.1 donde el filtro dejará de atenuar las señales por debajo de la frecuencia de corte[16].

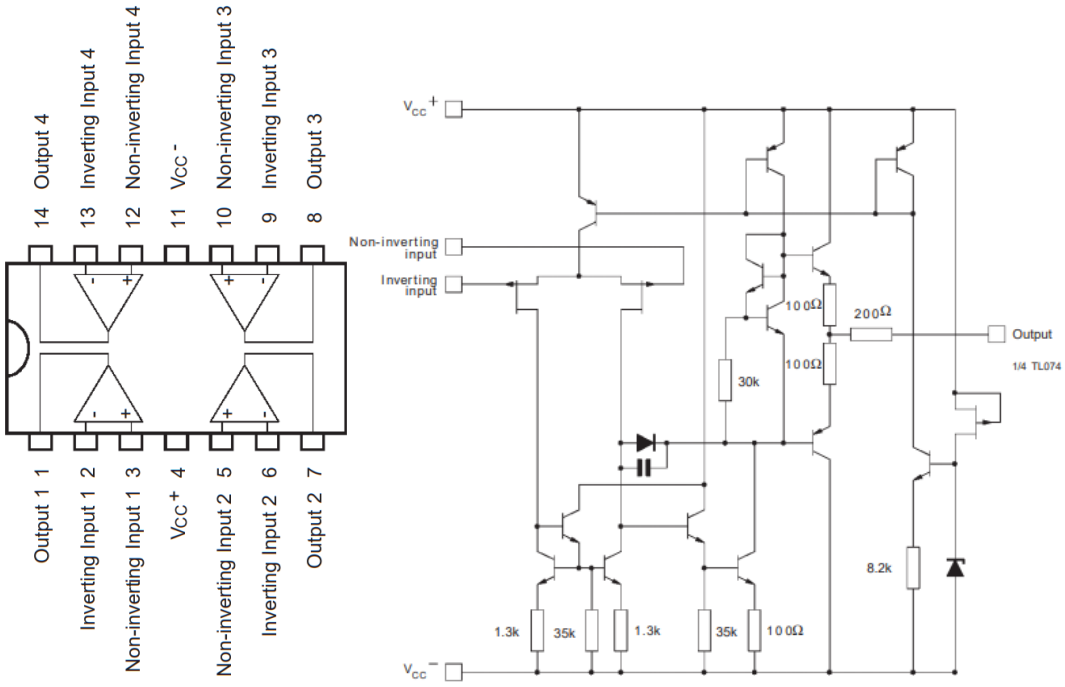


Ilustración 2. 4 Pines de conexiones y diagrama esquemático. Fuente: [17]

$$Fc = \frac{1}{2\pi R_2 C} \qquad (2.1)$$

Donde Fc es la frecuencia mínima en la que el filtro fundamentada en Ilustración 2.3, el filtro activo atenuaran las señales de orden mayor descritas propiamente como ruido, R_2 y C son la resistencia y condensador conectada a la entrada inversora y salida del amplificado operacional, si $C = 2\mu F$ y $Fc = 20Hz$ entonces:

$$Fc = 20Hz \qquad (2.2)$$

$$C = 2\mu F \qquad (2.3)$$

$$R_2 = \frac{1}{2\pi * 20\text{Hz} * 2\mu} \approx 4\text{k}\,\Omega \qquad (2.4)$$

La Ilustración 2.5 se observan las conexiones la cuales se realizaron y simularon en el software Multisim ya que presenta mejor comportamiento en cuanto a simulación de circuitos digitales[17]:

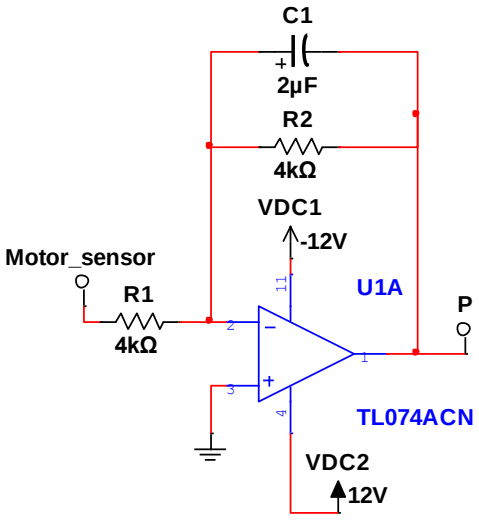


Ilustración 2. 5 OA Filtro pasa baja. Fuente: Autor [18]

2.2.2 Acondicionamiento de la señal

Como ya se describió en el capítulo anterior las entradas analógicas del TMS320F28069M tiene un voltaje máximo de 3.3 V esto hace necesario la normalización de los valores de entrada al puerto ADCINT0, a valores dentro del rango de voltaje comprendido entre $0\,V <= Vin <= 3\,V$ como se observa en la Ilustración 2.2 el voltaje de salida ronda los $7\,V$.

Para la normalización de la señal de entrada se ha dispuesto del circuito integrado TL074CN[16] de amplificadores operacionales en su configuración de ganancia, de acuerdo a la siguiente ecuación.

$$\frac{Vout}{Vint} = -\frac{R2}{R1} \qquad (2.5)$$

Donde se quiere que para el voltaje de $7V$ el cual corresponde a la frecuencia mayor atenué a $1.5\,V$ para ser conservadores en cuanto a los límites de entrada del dispositivo.

$$\frac{1.5\,V}{7\,V} = \frac{R2}{R1} = 0.22 \qquad (2.6)$$

Sí:

$$R_1 = 1k\Omega$$

$$R_2 = 1k\Omega * 0.22 = 220\ \Omega \qquad (2.7)$$

De acuerdo a los cálculos realizados se obtiene una ganancia de 0.22 la cual me permite normalizar el valor máximo de voltaje entregado por el motor-taco generador el cual es de $10\ V$ a $3000\ Rpm$; la simulación se realizó en el software Multisim[17] donde el circuito esquemático se puede apreciar en la ilustración 2.6.

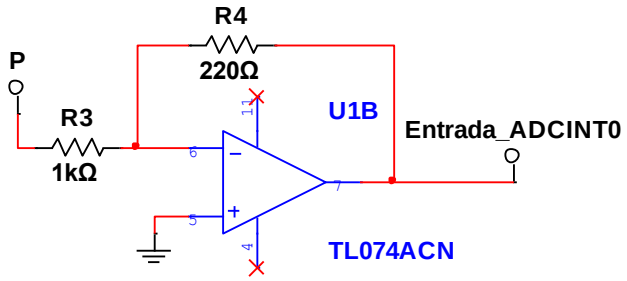


Ilustración 2. 6 OA Configuración de ganancia. Fuente: Autor

El sistema completo de filtrado y normalización se puede apreciar en la ilustración 2.7 realizada en el software Multisim[17].

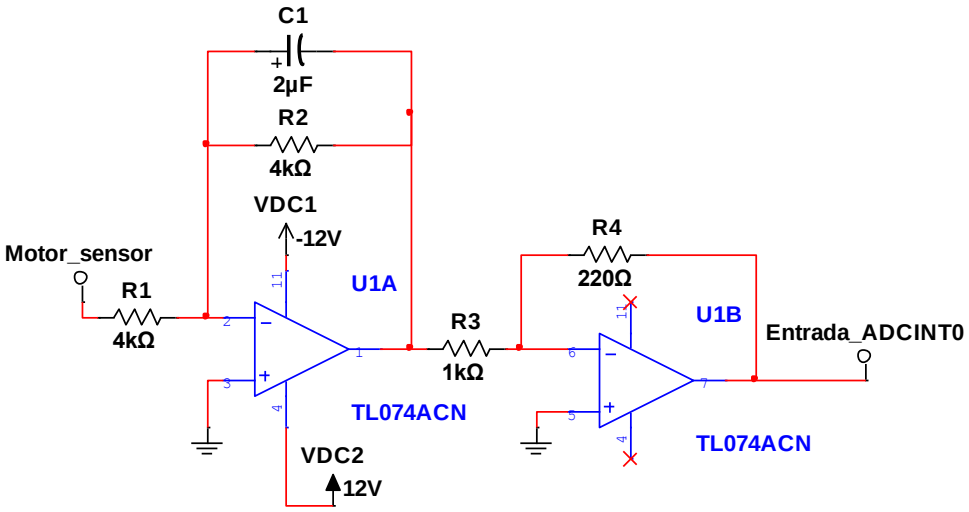


Ilustración 2. 7 Circuito esquemático completo. Fuente: Autor

Como se mencionó anteriormente la simulación se efectuó en Multisim™ el cual es un software estándar en la industria para el diseño de circuitos y simulación SPICE

para electrónica de potencia, analógica y digital en la educación y la investigación de alta confiabilidad para ver el comportamiento de circuitos análogos a altas frecuencias[17].

A continuación, en la ilustración 2.8 se analizan las simulaciones en que al generador de onda se le agrega un valor de offset de $6.6V$ a una frecuencia de $100Hz$ con un $Vp = 2V$ con el fin de simular una señal con una componente de ruido de $100Hz$ [17].

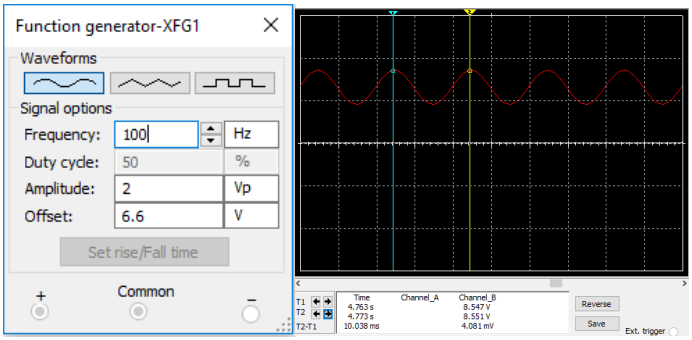


Ilustración 2. 8 Señal de entrada sin filtrar. Fuente: [18]

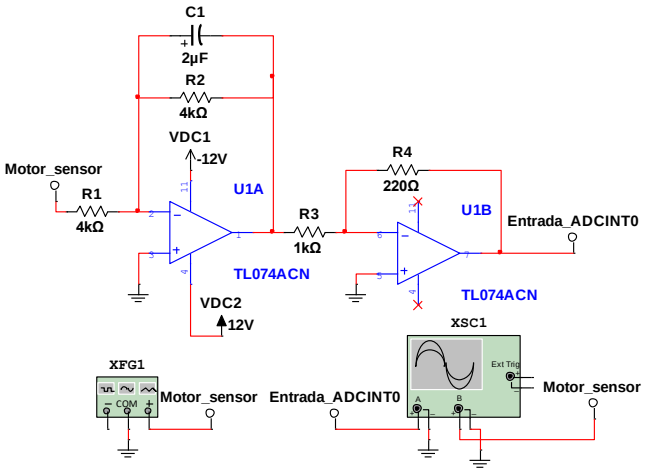


Ilustración 2. 9 Esquema de simulación. Fuente: [18]

Los resultados obtenidos se visualizaron en el osciloscopio XSC1 presentados a continuación en la ilustración 2.10 donde se presenta una señal normalizada y filtrada correctamente según datos de la simulación donde según la ecuación 2.5 $Vout =$

$Vin * 0.22$ donde con un voltaje de entrada de 6.6 V se obtiene $Vout = 6.6V * 0.22 = 1.452V$ con un %error esperado de 1.23%.

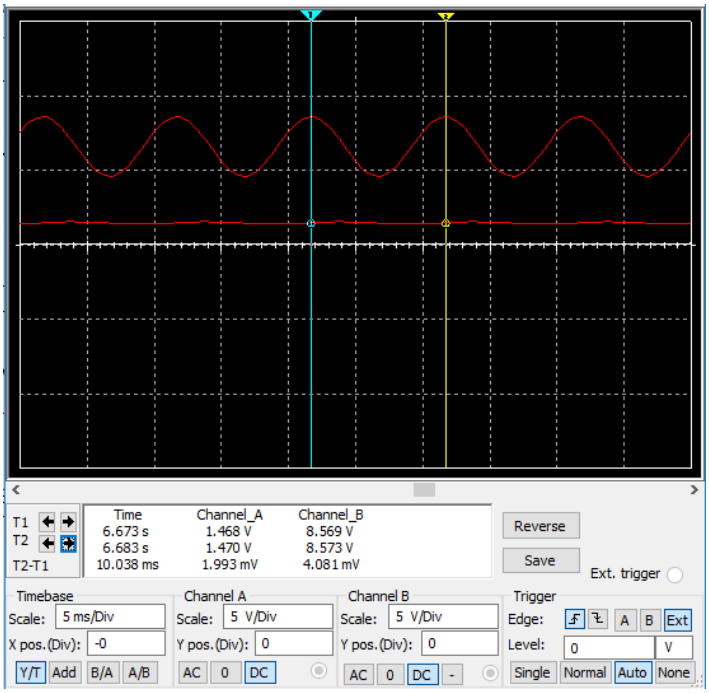


Ilustración 2. 10 Señal de salida

2.3 Implementación de circuito de filtrado y acondicionamiento de la señal

En las secciones anteriores describió la metodología de diseño, a continuación se mostrarán los resultados de la implementación y acople de este circuito propuesto con el fin de obtener una un entrada del lazo de retroalimentación lo más estable posible para un mejor funcionamiento del controlador que se describirá en posteriores secciones en la ilustración 2.11 se puede observar el circuito en PCV implementado, realizado en el software Proteus pues este posee componentes más variados gracias a sus diversas librerías, y por sus funciones al momento de crear una PCV de cualquier circuito.

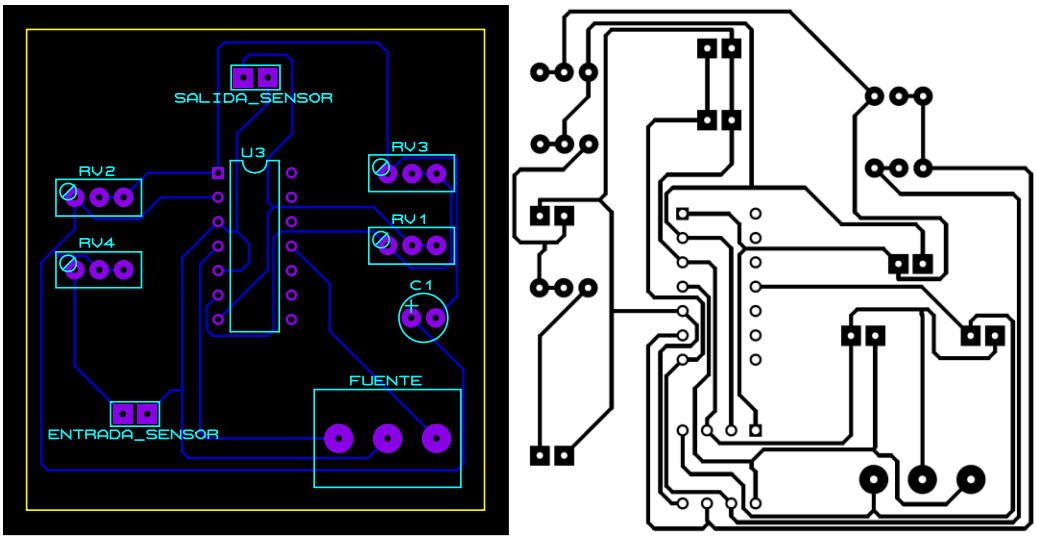


Ilustración 2. 11 PCV circuito planteado

Donde RV1, RV2, RV3 y RV4 son resistencia variable de alta precisión con el fin de poder modificar la ganancia del filtro junto con la frecuencia de corte con el fin de mejorar las características del filtro en la ilustración 2.12 se aprecia el diseño en 3D proporcionado por Proteus.

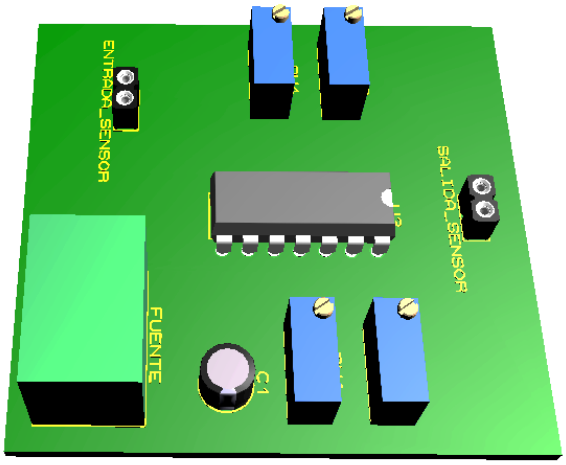


Ilustración 2. 12 Circuito 3D. Fuente: Autor

Después de realizar el procedimiento del esquema del circuito y soldadura se obtiene el circuito mostrado en la ilustración 2.13 donde que para evitar posibles interferencias de ruidos provocados por la fuente de alimentación del circuito final se le añadieron unas entradas de condensadores paralelos a la fuente de alimentación, junto con una entrada de integrado en caso de emergencia.

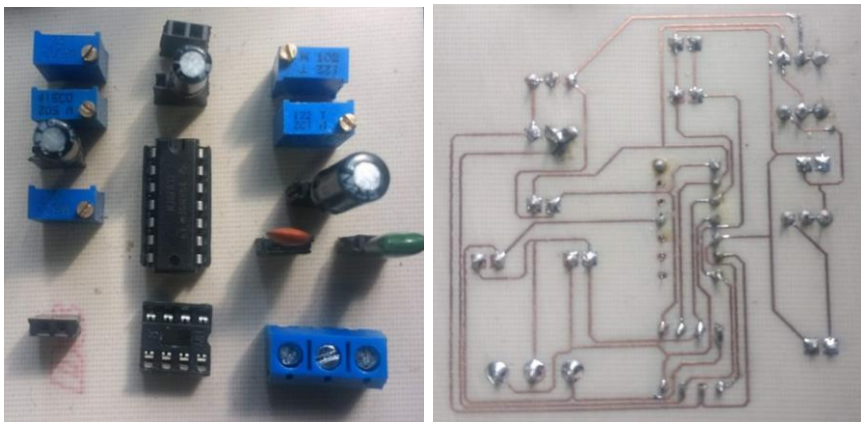


Ilustración 2. 13 Circuito Obtenido

Al realizar la validación del sistema de acondicionamiento donde se acopla el sensor de velocidad (Motor) al circuito ya descrito de filtrado y normalización descrito en esta misma sección se obtienen los siguientes resultados ilustración 2.14 medida en el DAQ 6022 con una tasa de muestro de 20kHz.

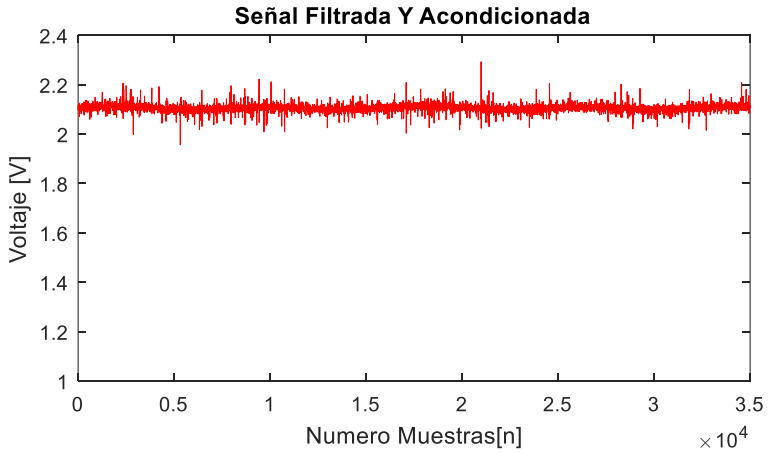


Ilustración 2. 14 Señal de salida del circuito

Como se puede apreciar en la ilustración 2.14 los datos efectivamente se encuentran normalizados aunque con un ruido aun presente, esto puede deberse a que la tarjeta TMS320F28069M se encuentra cerca a la etapa de electrónica de potencia y los transformadores del inversor lo cual hace que la señal se presente con un ruido apreciable; en este punto del desarrollo se plantea la aplicación de un filtrado digital de la señal a partir de una técnica conocida como la integral proyectiva la cual consiste en sacarle la media a una serie de datos finitos, la cantidad de datos se define por un Δt conocido la descripción matemática se define de acuerdo a la siguiente ecuación .

$$\frac{\sum_{n=0}^{\Delta t} ADCINT0_n}{n} \qquad (2.8)$$

Donde Δt es el tiempo que hay entre una acción de control y otra, calculado como 4 ciclos de la frecuencia mayor lo que aproxima a $72727.2727\mu S$. En la ilustración 2.15 se observa la aplicación del filtrado digital de la señal donde la gráfica inferior derecha muestra una señal llena de ruido, la cual es mejorada en las dos graficas de la izquierda de la ilustración, la gráfica inferior izquierda representa el valor digital filtrado de a señal y la gráfica superior izquierda representa el valor digital convertido a revoluciones por minuto, todo esto se observa gracias a la interfaz que se describirá en el la sección 5.

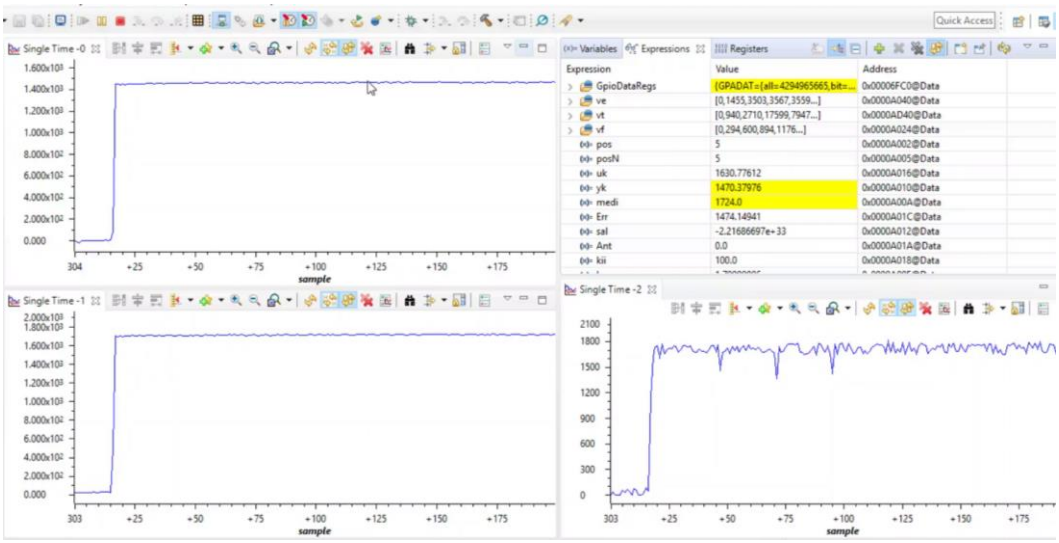


Ilustración 2. 15 Filtrado Digital

3. IMPLEMENTACION DE LA TECNICA DE CONTROL

3.1 Inversor multinivel

El Inversor de potencia multinivel es aquel que produce un voltaje alterno a partir de diferentes niveles de voltaje continuo ilustración 3.1. Estos inversores pueden provenir de una conexión en serie (con fuentes DC flotantes galvánicamente aisladas) o en paralelo (con fuente DC común y galvánicamente aislados con transformadores de potencia)[3].Los accionamientos de motores de inducción de alta potencia que utilizan convertidores trifásicos clásicos tienen las desventajas de una mala tensión y calidades de corriente. Para mejorar estos valores, se debe aumentar la frecuencia de conmutación, lo que provoca pérdidas de conmutación adicionales[5].

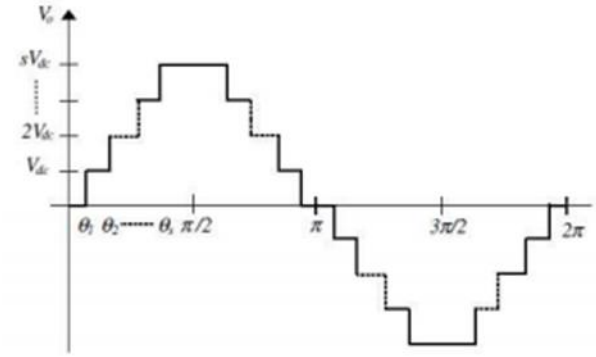


Ilustración 3. 1 Onda de salida del inversor multinivel. *Fuente:* [3].

La topología de inversor multinivel el cual fue adoptado como se puede apreciar en la ilustración 3.2 para el desarrollo del proyecto es presentada en la siguiente figura la cual corresponde a un convertidor de puente H en cascada asimétrico de fuente común con relación 1:3 y dos 2 etapas en cada fase proporcionando 9 niveles de voltaje y 15 niveles en el voltaje de línea el cual fue descrito e implementado en trabajos [3], [4] y [18] donde se adopta este inversor por las siguientes rezones:.

- Esta topología es la que necesita menos componentes y la que mayor número de niveles obtiene.
- Ya que el convertidor del proyecto debe contar con el mayor número de niveles en la tensión de salida; esta topología permite obtener el máximo número de niveles de tensión solo utilizando dos etapas de puentes de H.
- Esta topología permite adoptar modulaciones PWM optimizadas.

- Esta topología presenta la ventaja, que solo necesita una fuente de alimentación y puede ser fácilmente aplicada en sistemas de control.
- No presenta problemas de balanceo de tensión.
- El control es sencillo, lo cual facilita la labor de implementar una modulación optimizada.
- La inclusión de los transformadores añade regulación de tensión.

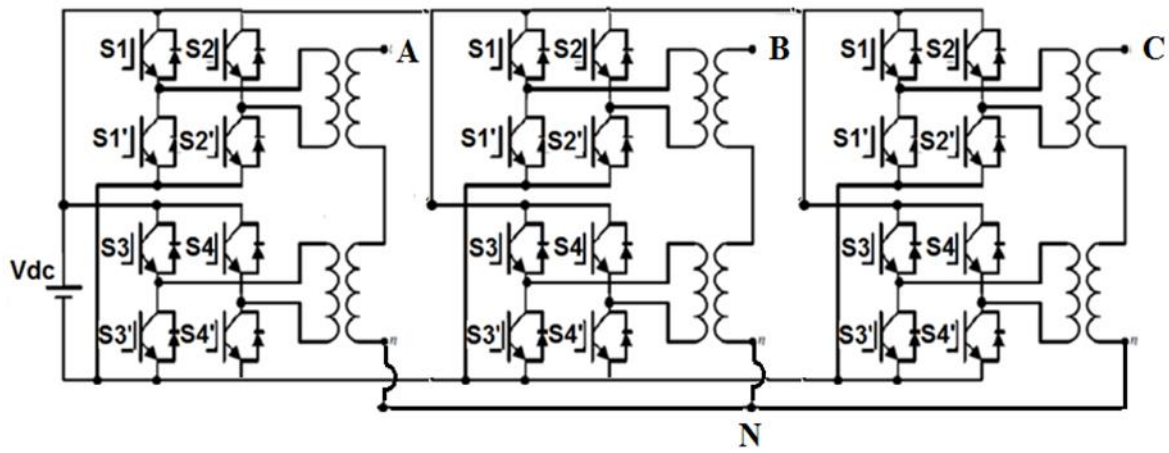


Ilustración 3. 2 Inversos multinivel adoptado para el control del motor de inducción. Fuente: [18]

Las celdas de subnivel multinivel en cascada son para elegir una serie de ángulos de conmutación para combinar elementos separados en una sola unidad según la forma de onda de la tensión sinusoidal deseada. Para la forma de onda de paso uniforme, la expansión de la serie de Fourier de la forma de onda del voltaje de salida que usa el esquema de conmutación de frecuencia fundamental se va describir a continuación.[18]

3.2 Estrategias de modulación

Para generar la señal de salida deseada del inversor, hay que ejercer un adecuado control sobre el apagado/encendido de los interruptores de éste. Las estrategias de conmutación son formas de crear los patrones de conmutación adecuados para generar la salida deseada[4].

Se han propuesto varias estrategias de modulación para inversores multinivel. Los métodos de modulación utilizados en los inversores multinivel pueden clasificarse según la frecuencia de conmutación. En este documento, se ha utilizado la técnica fundamental de cambio de frecuencia con modulaciones optimizadas[19].

3.2.1 Modelado matemático de la modulación

A continuación, se describe paso a paso el cálculo del THD y el valor eficaz verdadero de la tensión de línea de la modulación multinivel PWM de nueve escalones por fase. Los cálculos se realizan con una cota 50 armónicos, según lo establece la IEEE 519 de 1992, en cuanto al límite para le evaluación del contenido armónico.

3.2.2 Serie de Fourier de la fase a PWM de nueve escalones

En el trabajo [2] se describe de una forma general todos los pasos para el cálculo de las modulaciones optimizados con su respectivo estado de tiempo aplicado a inversor multinivel adoptado; generalizando, refiriendo ecuación utilizada a un caso particular, que muestra una modulación multinivel PWM de nueve escalones, la cual en el primer escalón tiene cinco ángulos 39 de disparo, como el último ángulo coincide con el primer ángulo de activación del segundo escalón esto hace que el escalón presente dos pulsos de activación , existiendo cinco ángulos de disparo; el segundo escalón tiene 3 ángulos de disparo, de los cuales el primero coincide con el último del primer nivel, este escalón tendrá un pulso y medio, el tercer nivel tendrá los mismos pulsos que el tercero, por último el cuarto escalón tendrá un pulso y medio en el primer cuarto de onda es decir, tres ángulos de disparo[2].

Como la forma de la modulación tiene simetría de ¼ de onda, el cálculo del contenido armónico se puede reducir al cálculo de la mitad de la modulación y multiplicar este resultado por dos. La ilustración 3.3 muestra la forma de onda de medio ciclo de la modulación. En los cálculos descritos siempre se presentará un dos precediendo, debido a que solo se evalúa medio ciclo[2].

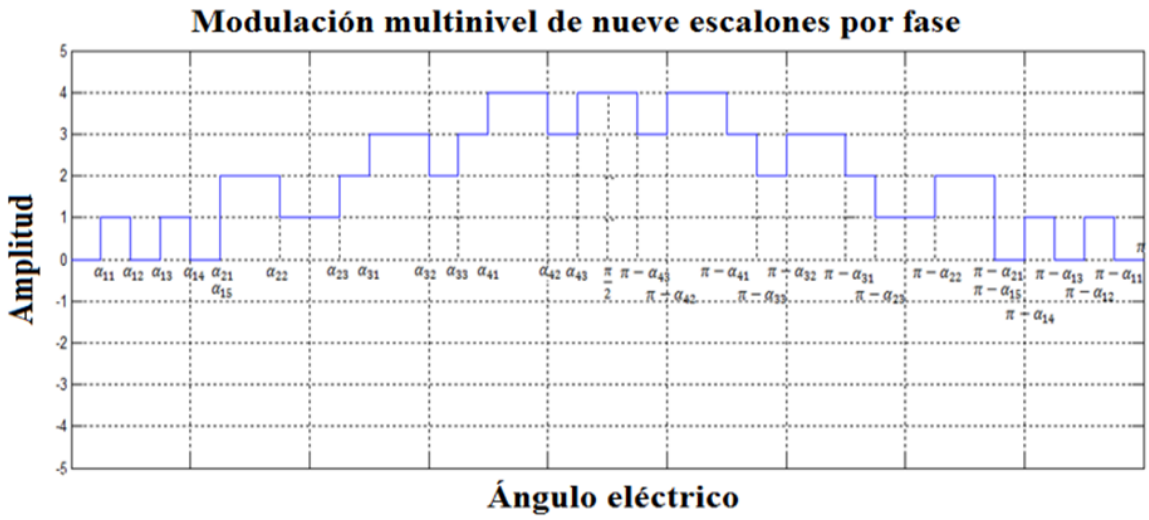


Ilustración 3. 3 Medio ciclo de la modulación multinivel PWM de siete escalones. Fuente: [18]

Los ángulos de disparo tienen la notación αij en donde la letra i denota el número del escalón al que pertenece y j indica el número del ángulo de disparo dentro del escalón. La serie de Fourier para ondas periódicas presenta la siguiente forma [2].:

$$v(t) = \frac{a_0}{2} + \sum_{n=1}^{a} (a_n cosnw_0 t + b_n sennw_0 t) \qquad (3.1)$$

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} v(wt) d(wt) \qquad (3.2)$$

a_n Coeficiente de la serie de Fourier, se calcula según la expresión:

$$a_n = \frac{1}{\pi} \int_0^{2\pi} v(wt) \cos n\,(wt) d(wt) \qquad (3.3)$$

b_n Coeficiente de la serie de Fourier, calculada mediante la ecuación:

$$b_n = \frac{1}{\pi} \int_0^{2\pi} v(wt) \operatorname{senn} n\,(wt) d(wt) \qquad (3.4)$$

El algoritmo que permite variar la frecuencia y el voltaje con en menor THD posible donde la ley de control escalar esta intrínseca en los resultados utiliza las ecuaciones del voltaje de línea RMS y el THD, en términos de los ángulos de disparo por fase[18].

De esta manera se define un vector *L*= [*a b c d*] que representa el número total de ángulos de encendido y apagado en cada nivel y con la diferencia de las series de fase se determina la serie para voltajes de línea[18].

La serie de Fourier para la Fase A se define por:

$$h_n = \left\{ \frac{4v_{cd}}{\pi n} \sum_{i=1}^{4} \sum_{j=1}^{L} (-1)^{j-1} \cos\left(n\alpha_{\tilde{y}}\right) para\ n\ impar \right. \qquad (3.5)$$

Donde v_{dc} es el valor del voltaje de cada escalón, y α_{ij} es el ángulo *j (a,b,c,d)* del escalón *i (1 2 3 4)*. Para la fase B, que se encuentra desplazada 120° eléctricos con respecto a la fase A, la serie de Fourier estará definida por[18]:

$$h_n = \begin{cases} 0 & para\ n\ par \\ \frac{4V_{dc}}{\pi n} \sum_{i=1}^{4} \sum_{j=1}^{L} (-1)^{j-1} cosn\alpha_{\tilde{y}} & para\ n\ impar\ múltiplo\ de\ tres \qquad (3.6) \\ \frac{4V_{dc}}{\pi n} \sum_{i=1}^{4} \sum_{j=1}^{L} (-1)^{j-1} cosn\alpha_{\tilde{y}} & para\ n\ impar\ no\ múltiplo\ de\ tres \qquad (3.7) \end{cases}$$

Realizando las respectivas diferencias de en términos de las series de Fourier de las fases *A* y *B*, se obtiene la serie de Fourier para la el voltaje de línea v_{AB}[18]:

$$h_n = \begin{cases} 0 & para\ n\ par \\ 0 & para\ n\ múltiplo\ de\ tres \\ \frac{4\sqrt{3V_{dc}}}{\pi n} \sum_{i=1}^{4} \sum_{j=1}^{L} (-1)^{j-1} cosn\alpha_{\tilde{y}} & para\ n\ no\ múltiplo\ de\ tres \qquad (3.8) \end{cases}$$

- **Modelo voltajes de línea y THD**

El estándar IEEE 519 de 1992, define la distorsión armónica total calculada hasta el armónico 50 [35], donde el armónico h_1 es la componente fundamental y h_n el pico de la armónica $_n$[18]

$$THD = \frac{\sqrt{\sum_{n=2}^{50} \left\{ \frac{1}{n} \left[\sum_{i=1}^{4} \sum_{j=1}^{Li} (-1)^{j-1} cosn\alpha_{ij} \right] \right\}^2}}{\left[\sum_{i=1}^{4} \sum_{j=1}^{Li} (-1)^{j-1} cosn\alpha_{ij} \right]} \qquad (3.9)$$

Donde *n* toma valores impares múltiplos diferentes de tres debido a que los armónicos múltiplos de 3 se suprimen en la conexión de los trasformadores, es decir 5, 7, 11, 13, 17, … y L_i son los componentes del vector *L*= [*a b c d*].

De igual forma, el valor eficaz se puede definir en términos de los ángulos de Conmutación y los armónicos según la ecuación ():

$$V_{line_{RMS}} = \sqrt{\sum_{n=1}^{50} \frac{\left\{\frac{4\sqrt{3V_{dc}}}{\pi n}\left[\sum_{i=1}^{4}\sum_{j=1}^{Li}(-1)^{j-1}cos n\alpha_{ij}\ \right]\right\}^2}{2}} \qquad (3.10)$$

$$n = 5,7,\dots, múltiplos\ difernetes\ de\ tres$$

- **Algoritmo para la determinación de voltajes de línea**

Para evitar la saturación de las máquinas el convertidor multinivel reduce el voltaje de línea en términos proporcionales a la frecuencia, mediante una ley escalar obtenida con la programación de un algoritmo de búsqueda de una modulación con un determinado valor RMS mediante las ecuaciones antes descritas con la restricción de mínimo THD. El algoritmo de búsqueda de las modulaciones en todo el rango de frecuencia emplea un algoritmo genético [6].
Cabe mencionar que las soluciones y el planteamiento de dichas ecuaciones y metodología de solución se encuentran descritas a detalle en las referencias plasmadas a lo largo del documento.

3.3 Algoritmo de optimización

El propósito de la aplicación de Algoritmos Genéticos es la optimización de la modulación PWM, esto se logra a través de la creación de un conjunto de individuos (diferentes valores de frecuencias, número de pulsos y posición del pulso) que representan las posibles soluciones al problema en particular que se quiere optimizar y buscar la menor distorsión armónica del sistema en la Ilustración 3.4 se puede apreciar el diagrama del algoritmo de optimización adoptado[4].

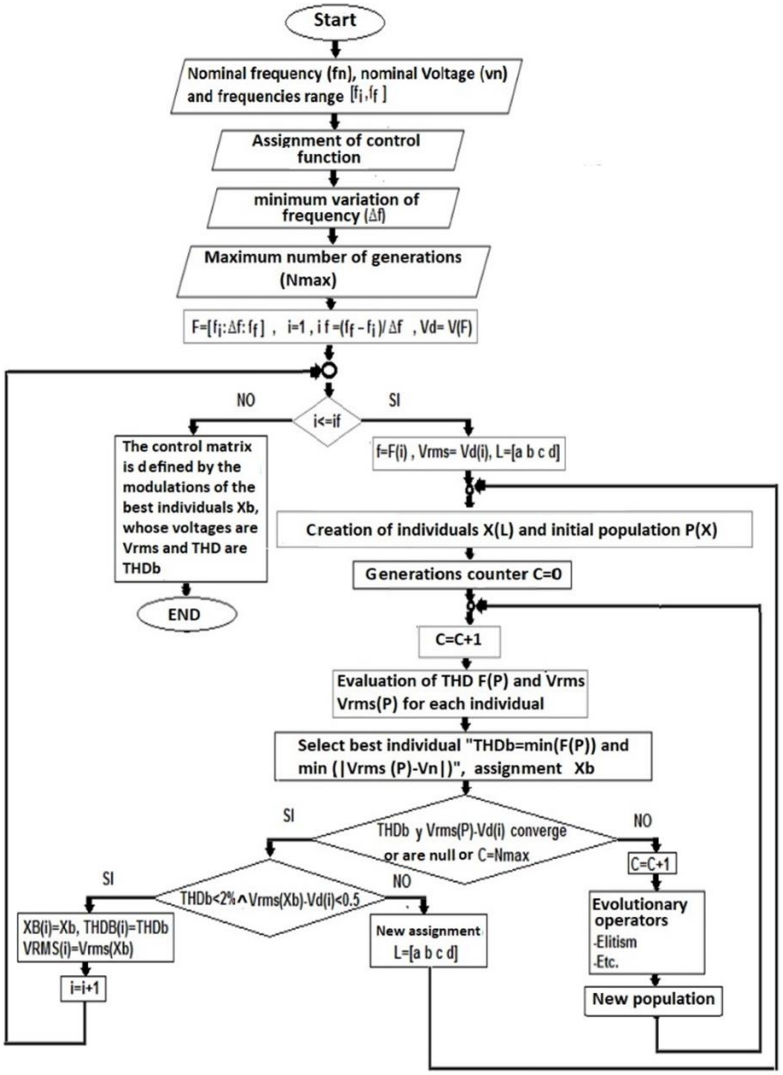


Ilustración 3. 4 Diagrama de algoritmo genético. Fuente: [4]

3.3.1 Resultados Del Algoritmo De Optimización

Se asignó el voltaje nominal de la maquina en $220\ Vrms$, la frecuencia nominal $50\ Hz$, el rango de frecuencias será de $[45\ Hz, 55\ Hz]$ con pasos de $\Delta f = 1\ Hz$, el nivel Vboost se establece en $30\ Vrms$ de línea. Con estos datos se corrió el algoritmo en Matlab y se obtuvieron como resultado los ángulos de activación de 11 modulaciones

con contenido armónico óptimo y con valores RMS que siguen la ley V vs f con los datos anteriores[2]. Otorgando los ángulos exactos de conmutación o de cambio de estados de los pines GPIO de la tarjeta descrita para el rango de frecuencia preestablecido en el algoritmo genético que se obtiene como una matriz[2].

El algoritmo de creación de datos de tiempos y de estados de modulaciones de la ley de mando escalar donde principalmente se obtienen un vector de tiempos que representan el tiempo de apertura o cierre de un dispositivo de control del inversor, y una matriz de estados binarios de longitud de 12 bit que representa las salidas físicas de control de la tarjeta a los semiconductores donde para fines prácticos se convirtieron todos a valores de tipo entero, el tercer vector obtenido del algoritmo antes mostrado es el vector de finales donde cada posición contenida representa el final de una modulación correspondiente a una frecuencia especifica.

$$Ve[E145Hz, E245Hz, \dots\dots, Efinal45Hz, E146Hz, \dots, Efinal46H, \dots\dots\dots, E146Hz, \dots\dots, Efinal55Hz] \quad (3.18)$$

$$Vt[t145Hz, t245Hz, \dots\dots, tfinal45Hz, t146Hz, \dots, tfinal46Hz, \dots\dots\dots\dots, t146Hz, \dots\dots, tfinal55Hz] \quad (3.19)$$

$$Vf[F145Hz, F246Hz, \dots\dots\dots\dots, Ffinal55Hz \quad (3.20)$$

De esta forma solo estaríamos trabajando con tres vectores relacionados entre sí de nuestra tarjeta descrita para la investigación, de esta forma se mejoraría la carga de datos a la tarjeta de adquisición; Donde las modulaciones son codificadas en Matlab y almacenadas en 2 archivos tipo txt denominados **Ve.txt** y **Vt.txt** los cuales se cargan a la tarjeta mediante líneas de código descritas en la sección 3.4, a continuación en la tabla 3.1 se presentan una parte de estos datos.

Tabla 3. 1 Extracto de datos obtenidos de Modulación 50Hz

Vt	Ve											
2782	1	1	1	1	0	1	1	1	1	0	1	1
24058	1	1	1	1	0	1	0	1	1	0	1	1
924	1	1	1	1	0	1	0	0	1	0	1	1
6483	1	1	1	1	0	1	0	0	1	0	1	0
338	1	1	1	0	0	1	0	0	1	0	1	0
5131	1	1	1	0	0	1	0	0	1	0	0	0
1900	1	1	1	0	0	1	0	1	1	0	0	0

3.4 Implementación de las modulaciones optimizadas en el TMS320F28069

A partir de este punto del libro cada algoritmo y código descrito se desarrolló en el entorno de programación CODE COMPOSER STUDIO con lenguaje CCS el cual está basado en C, C++ y assembler donde los registros se organizan en estructuras de campos de bits, ya que brinda una gran flexibilidad en cuanto optimización de código y tiempos de ejecución. En el **anexo 1** se muestra una guía de inicio rápido para empezar a codificar en este entorno.

Para la obtención de las modulaciones se parte de la adquisición de los datos decodificados por Matlab en los archivos $\boldsymbol{Vt.txt}$ y $\boldsymbol{Ve.txt}$, dicha adquisición se realiza en base a las librerías presentes en la tarjeta como file.h con las cuales es posible abrir un archivo tipo fichero en este caso .txt con el fin de leer o escribir datos en él. En la ilustración 3.5 se presenta un diagrama en bloques para comprender mejor el funcionamiento de este algoritmo primordial.

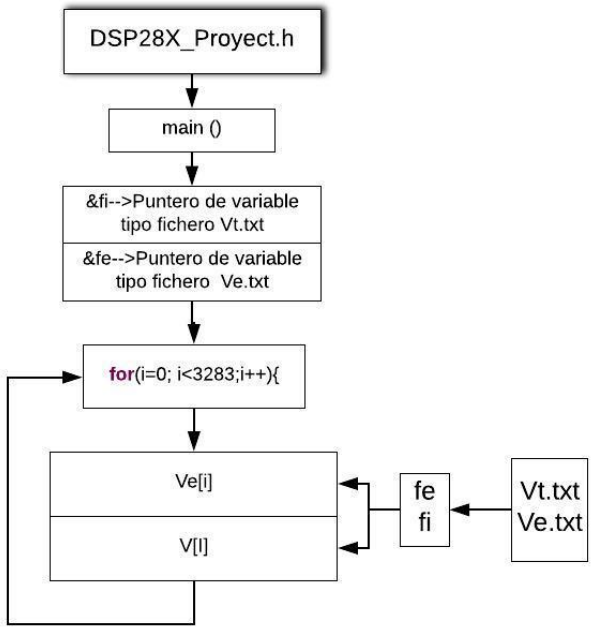


Ilustración 3. 5 Algoritmo de adquisición de datos de las modulaciones

Algoritmos de adquisición de datos

```
#include "DSP28x_Project.h" //archivo encabezado del dispositivo donde se
encuentran funciones ejemplo del dispositivo
#include "string.h"
#include <stdio.h>

unsigned int vt[3283];//vector de tiempos
unsigned int ve[3283];//vector de estados
unsigned long acu_M=0;//acumulador de muestras ADCINT0
int vf[12]={0, 294, 600, 894, 1176, 1446, 1752, 2058, 2364, 2670, 2976,
3282};//vector de finales
int i=0;//variable de control

void main(void){


// Adquisición de los datos desde un archivo .txt-------------------------
    FILE* fe; //puntero archivo Ve.txt
    FILE* fi; //puntero archivo Vt.txt
    fe = fopen("ve.txt","r");//open archivo Ve.txt
    fi = fopen("vt.txt","r");//open archivo Vt.txt
    for(i=0; i<3283;i++){
    fscanf(fe,"%u",&ve[i]); //almacenamiento vector Ve
    fscanf(fi,"%u",&vt[i]); //almacenamiento vector Vt
       }fclose(fe);
       fclose(fi);

//-----------------------------------------------------------------
    while(1)
    {// Ciclo infinito
    }
}
```

3.5 Descripción general del algoritmo de la técnica control de velocidad

El algoritmo general de la técnica de control de velocidad, se implementó una vez se adquieran los vectores de tiempo y estados en sus respectivas variables tipo vector dentro de la función principal de ejecución del programa. En este algoritmo se realiza la respectiva elaboración del código que controla la técnica de control ya descrita con anterioridad, con el fin de que cada modulación generada por pines de control de la tarjeta corresponda con su respectiva frecuencia y así mismo cada posición del vector de tiempos corresponda con su respectivo estado del vector de estados. Para la

inclusión de una nueva etapa al código principal de ejecución se debe tener presente los tiempos de ejecución de cada línea de código pues le suman un pequeño error a la modulación lo que hace que se deban ajustar los valores de los tiempos de conmutación para compensar la sobrecarga causada por el ejecución del código en general, el código principal comprende además de los algoritmos propuestos, archivos encabezado, archivos fuente, librerías, librerías de desarrollo especifico y demás configuraciones y/o enrutamientos realizados para comprender mejor estas etapas ver **Anexo1**.

La descripción simplificada del algoritmo de la técnica de control.

- Primero que todo se realizan las configuraciones necesarias de todos los módulos usados en el desarrollo TIMER0 y GPIO que se describen en la sección 3.52, 3.5.1 junto con las líneas de código de configuración de y programación de la memoria Flash.

- El código en general va a realizar una interrupción en cada valor del vector de tiempo $Vt[tn]$, el cual su posición coincide con su respectivo valor en el vector de estados Ve $[En]$ donde la posición "n" coincide con el vector de estados y vector de tiempos, la cual corresponde a una modulación de una frecuencia especifica en el rango de frecuencias de $45Hz\ y\ 55Hz$.

- En este punto del algoritmo, el vector de finales definido como $Vf[Nf]$ almacena en cada una de sus posiciones "Nf" la posición final de cada modulación correspondiente a una frecuencia específica, convirtiéndose en un punto de control para asignación de condicionales dentro el código e ejecución.

- Como se explica anteriormente el código de ejecución se ejecutará cíclicamente en una frecuencia establecida por usuario "Fn" deste $Vt[Vf[Nfinal_{Fn} - 1] + 1]$ hasta $Vt[Vf[Nfinal_{Fn}]]$.

En Ilustración 3.6 en modo de diagrama bloque se esquematiza el algoritmo general de control de velocidad con modulaciones optimizadas donde el XDS100V2 el programador la tarjeta de desarrollo.

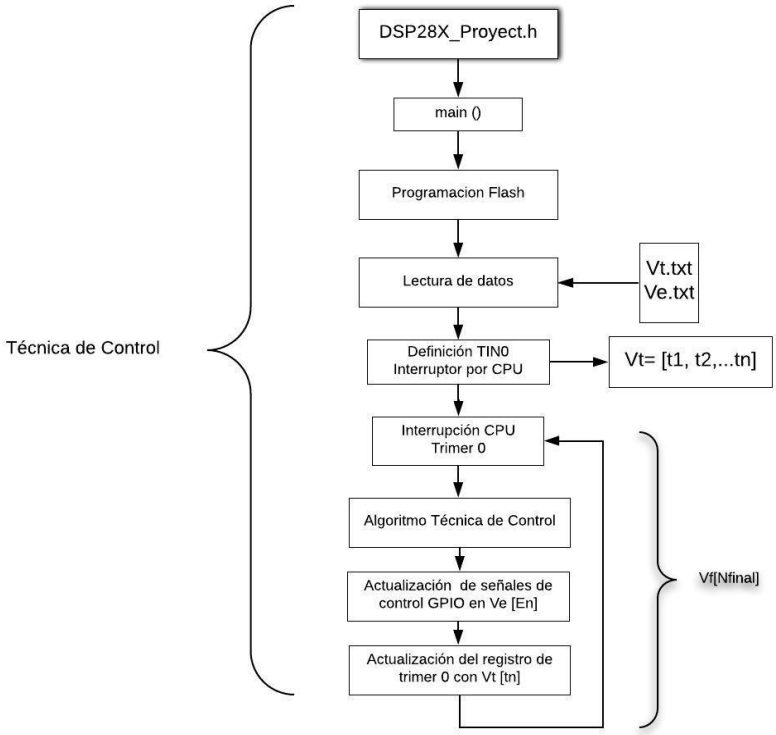


Ilustración 3. 6 Diagrama algoritmo general. Fuente: Autor

Donde el algoritmo de la técnica de control consta de tres etapas las cuales son:

- Configuración y utilización de la memoria flash
- Configuración y utilización del Temporizador Timer0
- Asignación de las salidas GPIOx en dependencia del temporizador

3.5.1 Configuración y utilización de la memoria flash

Para aumentar la memoria donde del programa maneja los datos generados al ejecutar la lógica de programación, fue necesario migrar el desarrollo a la memoria flash del dispositivo ya que por defecto cuando se crea un archivo nuevo en el code composer studio este se guarda al momento del debugger en la memoria RAM la cual es volátil y muy reducida. A continuación se muestran la líneas de código que se usaron para realízarlo, es de suma importancia agregar el archivo F28069.cmd realizando el mismo procedimiento mostrado en el anexo 1.

Algoritmos de Memoria flash

```
#include "DSP28x_Project.h" //archivo encabezado del dispositivo donde se
encuentran funciones ejemplo del dispositivo
#include "string.h"
#include <stdio.h>

//definiciones por el enlazador F2808.cmd----------------------
extern Uint16 RamfuncsLoadStart;
extern Uint16 RamfuncsLoadEnd;
extern Uint16 RamfuncsRunStart;
extern Uint16 RamfuncsLoadSize;
//---------------------------------------------------------------
void main(void){
    //Copia el código crítico de tiempo y el código de configuración de
Flash a la RAM
    memcpy(&RamfuncsRunStart, &RamfuncsLoadStart,
    (Uint32)&RamfuncsLoadSize);
    InitFlash();//inicialización de flash ser grabada función de la memoria
RAM
    while(1)
    {// Ciclo infinito
    }
}
```

3.5.2 Control de los tiempos de modulación con el Timer-0

Para el TMS320F28069M el CPU Timer-0 de 32 Bits y el CPU-Timer 1 se pueden usar en aplicaciones de usuario. En este caso se usa el Timer-0 que genera una interrupción del temporizador de la CPU (TINT0) la cual hay que tener claro la forma como se encuentra organizado y como se muestra en la Ilustración 3.7 se analizó que el timer-0, en primer lugar depende de un grupo de vectores de interrupción denominados PIE para luego sea asignado la prioridad o jerarquía de interrupción de acuerdo a unos vectores que van desde INT1 a INT12 [20].

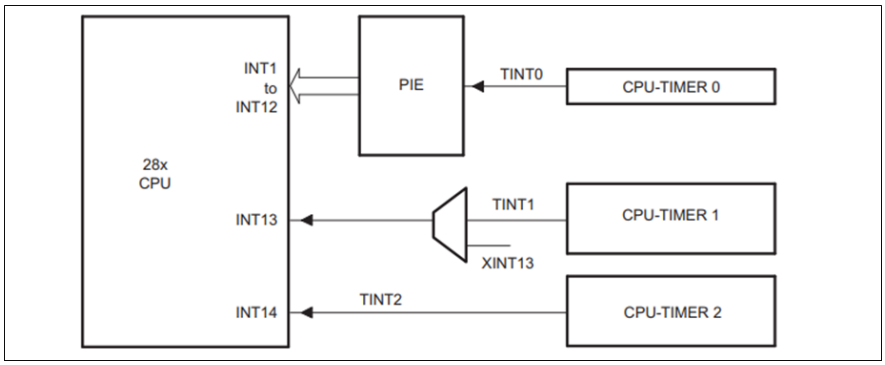


Ilustración 3. 7 Señales de salida del CPU-Timer. Fuente: [21]

El funcionamiento general del temporizador de la CPU es el siguiente: El registro de contador de 32 bits TIMH: TIM se carga con el valor en el registro de período PRDH: PRD. El contador disminuye una vez cada (SYSCLKOUT) (TPR [TDDRH: TDDR] +1), donde TDDRH: TDDR es el divisor del temporizador. Cuando el contador llega a 0, una señal de salida de interrupción del temporizador genera un impulso de interrupción para el desarrollo de la aplicación se tiene una función la cual me traduce tiempo en microsegundos a ciclos de reloj[20].

Interrupciones de CPU

Cuando se produce la interrupción generada por el evento el cual sucede cuando el temporizador ha terminado de contar el tiempo $Vt[tn]$. En el C28x, las interrupciones se activan por software (con la instrucción EINT y ERTM) y se atiende de acuerdo con una clasificación de prioridad establecida que para el desarrollo del algoritmo se estableció en INT1 con prioridad de hardware de 5 ver Tabla 3.2 .El módulo PIE proporciona un control adicional antes de que una interrupción llegue a la CPU C28x ya que en términos simples es un grupo de vectores de interrupción $PIEIER_{1\text{-}12}$ donde cada uno posee grupos de 8 vectores de interrupción INTx7 lo que me da una flexibilidad de 96 posibles interrupciones. [6]:

Tabla 3. 2 Prioridad de vectores de interrupción extracto. Fuente: [6]

Vector	Dirección absoluta (hexadecimal)		Prioridad de Hardware	Descripción
	VMAP=0	VMAP=1[1]		
RESET	00 0000	3F FFCO	1(elevado)	Reset
INT1	00 0002	3F FFC2	5	Maskable Interrupt 1

Para ejemplificar mejor este funcionamiento se puede observar la Ilustración 3.8 donde se muestra cómo se multiplexan las diversas fuentes de interrupción, para el desarrollo del código y como solo existe una interrupción general que controla los algoritmos se configuro de manera simple como se explica a continuación[6].

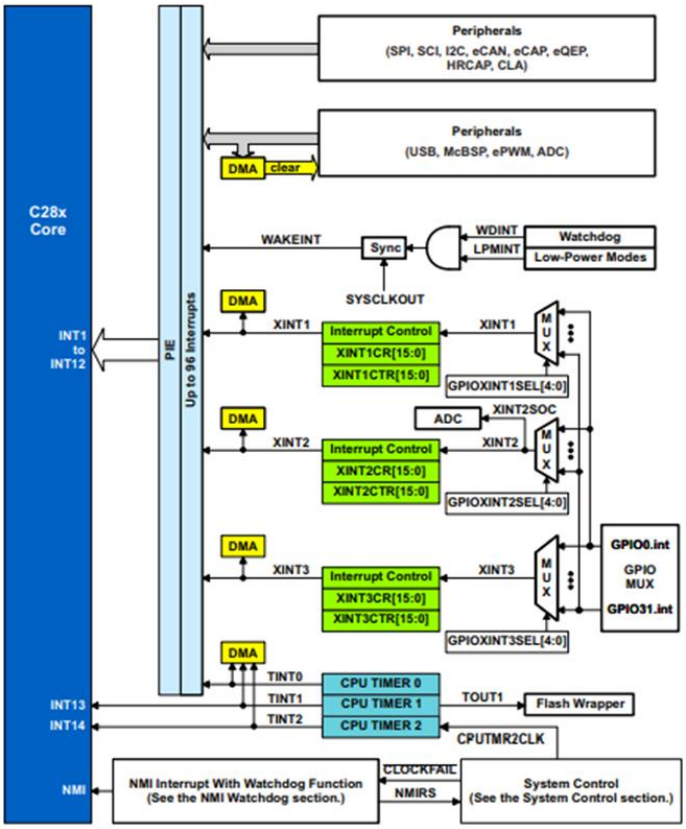


Ilustración 3. 8 Diagrama de bloques PIE Vector. Fuente: [6]

Para la configuración de timer específicamente el TIMER0 junto con la interrupción de CPU que este genera cuando termina de contar el tiempo establecido en su registro, el cual es primordial para el correcto funcionamiento del sistema pues de él dependen los módulos GPIO (salida de las modulaciones optimizadas $Ve[En]$ vector de estados) se realizaron los siguientes pasos:

- Se realiza una limpieza de las interrupciones que están activas, llevando a valores conocidos todos los bits de sus registros, esto se realiza con los registros IER e IFR.

- Después de este paso se debe enviar la dirección de la rutina de interrupción a la tabla de vectores PIE la cual tiene almacenado todos los periféricos que generan interrupción, la cual se ejecute cuando el temporizador termine de contar el tiempo del vector $Vt[tn]$ en una posición "tn". En conclusión, se le asigna la dirección de la subrutina a la interrupción de genera el Timer0.
- Con una función que me traduce tiempo en microsegundos a señales de reloj se le asigna el valor en binario en los registros de tiempo TIMERxPRD el cual lo carga al registro de conteo TIMERxTIM para que ocasionen una interrupción de CPU ISR cuando el contador llegue a 0.
- Cuando el tiempo en la posición "tn" se cumple se presenta la primera interrupción de CPU provocada por el Timer0, en donde se va a dar un nuevo estado en los GPIO $Ve[En+1]$y actualización del valor de conteo de timer por $Vt[tn+1]$.

En la Ilustración 3.9, se describe esquematiza el algoritmo implementado para la configuración del timer.

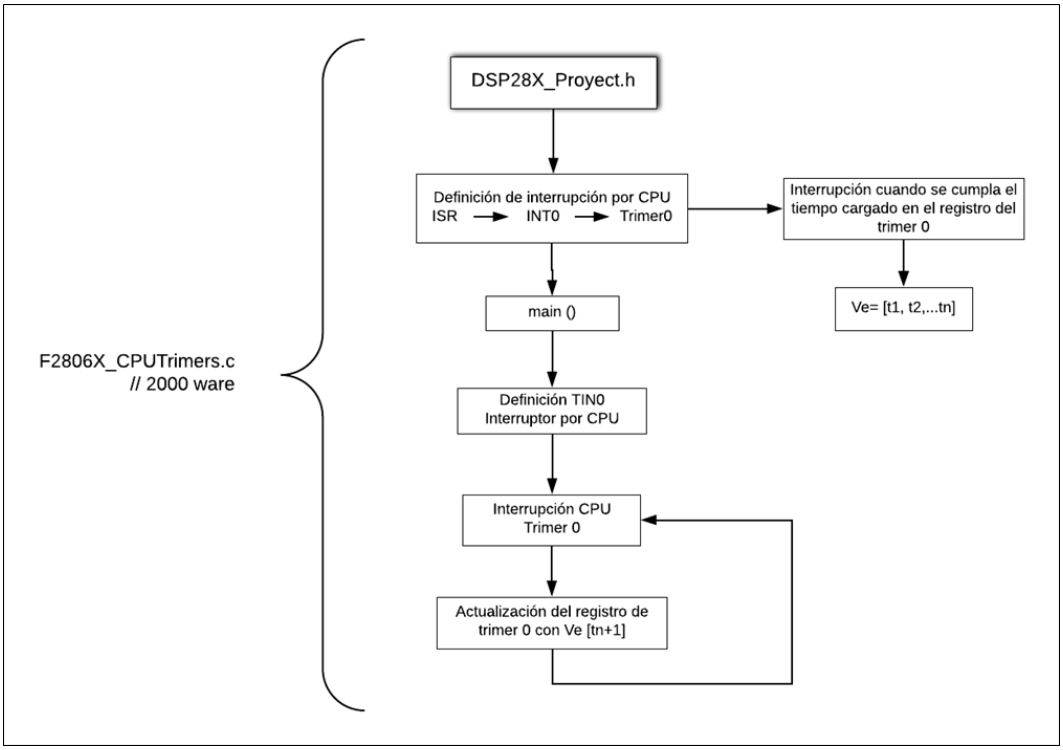


Ilustración 3. 9 Algoritmo del Timer. Fuente: Autor

Algoritmo en CCS desarrollado en code composer studio para interrupción por TIMER-0.

```
#include "DSP28x_Project.h"//archivo encabezado del dispositivo donde se encuentran funciones ejemplo del dispositivo
__interrupt void cpu_timer0_isr(void); //interrupción externa, inicio de rutina de servicio de interrupción
float ope;
int f=90;
int i;
unsigned int vt[3283];
int vf[12]={0, 294, 600, 894, 1176, 1446, 1752, 2058, 2364, 2670, 2976, 3282};//vector de finales

void main(void){
    InitSysCtrl();//Inicializar el control del sistema: PLL, WatchDog, habilita los relojes a los periféricos F2806x_SysCtrl.c
    DINT; //Borrar todas las interrupciones e inicializar la tabla de vectores PIE: Desactivar las interrupciones de la CPU
    InitPieCtrl();//Inicializar los registros de control PIE a su estado predeterminado interrupciones deshabilitadas archivo F2806x_PieCtrl.c
    IER = 0x0000; // Desactivar las interrupciones de la CPU y borrar todos los indicadores de interrupción de la CPU
    IFR = 0x0000; // Desactivar las interrupciones de la CPU y borrar todos los indicadores de interrupción de la CPU

    InitPieVectTable();// Inicializar la tabla de vectores PIE con punteros al shell Interrupción

    EALLOW;//habilita escritura sobre registros de protegidos
    PieVectTable.TINT0 = &cpu_timer0_isr; //se le asigna la dirección de función de interrupción a TINT0
    EDIS;//deshabilita escritura sobre registros de protegidos

    InitCpuTimers();//inicialice los temporizadores de CPU
     ope=vt[i];// Convierte Vt[i] en float

     ConfigCpuTimer(&CpuTimer0, f,(ope/100));// Frecuencia de CPU de 90MHz, ope/100  (en uSegundos)
     CpuTimer0Regs.TCR.all = 0x4001;//habilita la interrupción del temporizador
     IER |= M_INT1; // Asigna interrupción a nivel INT1 la interrupción
     PieCtrlRegs.PIEIER1.bit.INTx7 = 1; //Grupo 1 del pie habitado INTx7 individualmente
     EINT; //Habilitar la interrupción global INTM
```

```
     ERTM; //Habilitar DBGM de interrupción global en tiempo real
    while(1){// Ciclo infinito
    }
}
__interrupt void
cpu_timer0_isr(void)
{    ConfigCpuTimer(&CpuTimer0,f,(ope/100) );// Frecuencia de CPU de 90MHz,
ope/100  (en uSegundos)
    CpuTimer0Regs.TCR.all = 0x4001; //habilita la interrupción del
temporizador
}
```

3.5.3 Configuración de salidas de control módulo GPIO$_{0\text{-}11}$

Para la configuración de señales de control GPIO$_{0\text{-}11}$ primero se debe conocer que estas salidas periféricas independientes las cuales se multiplexan según tablas 1.6, 1.7, 1.8, y 1.9. mostradas en este libro, cada GPIO se puede usar con hasta tres propósitos: periféricos como ePWM, comunicación, entrada y ADC entre otras; en la ilustración 3.10 se observa la forma organizacional de los registros que involucran el módulo GPIO los cuales se configuraran para ser las salidas de control[20].

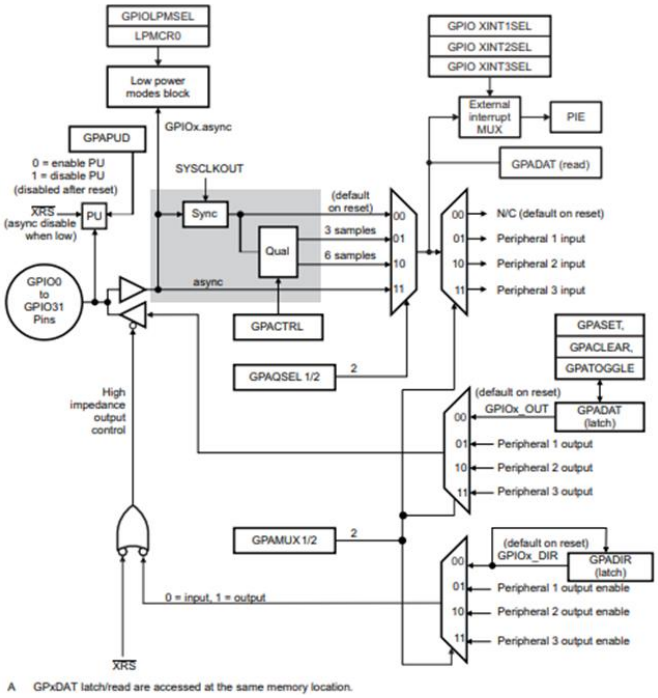


Ilustración 3. 10 Diagrama GIO0 a GPIO31, GPIO34, GPIO040-GPIO58. Fuente: [21]

Para la configuración de los GPIO utilizados en el control de las modulaciones se partió del siguiente procedimiento

- Identificación de pines organizados de forma secuencial con el fin de optimizar la asignación de los datos, activar los pines y asignarlos como salida en su respectivo registro.
- Habilitar o deshabilitar resistencias de pull-up internas. Pero para propósitos del desarrollo, se van a seleccionar como salida donde la funcionalidad de resistencias de pull-up se encuentran deshabilitadas.
- Seleccione la función de pin: configure los registros $GPAMUX_{0\text{-}11}$ de modo que al enviar un bit 0 al registro específico del GPIOx se selecciona la funcionalidad de GPIO.
- Tanto para entrada como para salida digital de propósito general, se configura en el registro GPADIR. Por defecto, todos los pines GPIO son entradas digitales. Para cambiar el pin de entrada a salida, primero se carga un bit de 1 al registro correspondiente de los pines a configurar, que para este desarrollo específico fueron los GPIO0-GPIO11 del registro GPADIR.

En la ilustración 3.11, se describe por medio de diagrama de bloques el algoritmo implementado para la configuración los GPIO como salida del GPIO0-GPIO11.

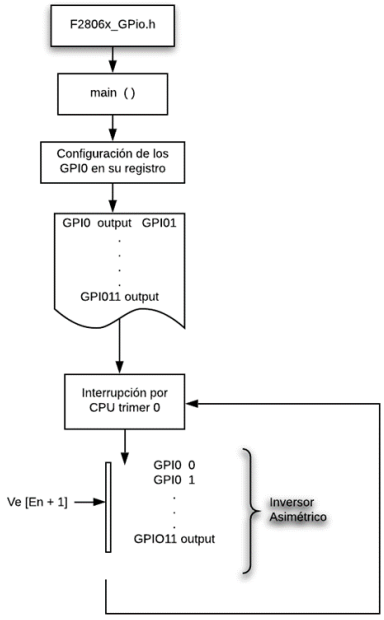


Ilustración 3. 11 Modulaciones del convertidor del GPIO0-GPIO11. Fuente: Autor

Algoritmo en CCS desarrollado en Code Composer Studio para configuración del GPIO0-11 señales de control

```
#include "DSP28x_Project.h" //archivo encabezado del dispositivo donde se
encuentran funciones ejemplo del dispositivo
#include "string.h"
#include <stdio.h>


__interrupt void cpu_timer0_isr(void);//interrupción externa, inicio de
rutina de servicio de interrupción
unsigned int ve[3283];//vector de estados
int vf[12]={0, 294, 600, 894, 1176, 1446, 1752, 2058, 2364, 2670, 2976,
3282};//vector de finales

int i=0;//variable de control

void main(void){
    EALLOW; //habilita escritura sobre registros de protegidos
//Activar pines Como GPIO0-11--------------------------
GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO7 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO8 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO9 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO11 = 0;
//Activar GPIO0-11 como salidas-------------------------
       GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO4 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO5 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO6 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO7 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO8 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO10 = 1;
       GpioCtrlRegs.GPADIR.bit.GPIO11 = 1;
```

```
        EDIS; //deshabilita escritura sobre registros de protegidos

    GpioDataRegs.GPADAT.all =ve[i]; //Salida Modulación de control
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //Bloquea demás interrupciones

//Habilitar DBGM de interrupción global en tiempo real
    while(1)
    {// Ciclo infinito

    }
}
__interrupt void
cpu_timer0_isr(void)
{
    i=i+1; //cambio de posición
    GpioDataRegs.GPADAT.all =ve[i]; //Salida Modulación de control
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //Bloquea demás interrupciones
//configuración de timer a nuevo tiempo de Vt[i]
}
```

Algoritmo en CCS de Implementación de la técnica de control

A continuación se presentan el algoritmo general desarrollado el cual incluye todas los algoritmos antes planteados, además de incluir las líneas de código que controlan que el cambio de modulación a otra se realice de una en una, ya sea hacia arriba o hacia abajo esto con el fin de cuidar la integridad física de todos los elementos y/o dispositivos involucrados en el desarrollo.

```
#include "DSP28x_Project.h" //archivo encabezado del dispositivo donde se
encuentran funciones ejemplo del dispositivo
#include "string.h"
#include <stdio.h>

__interrupt void cpu_timer0_isr(void);//interrupción externa, inicio de
rutina de servicio de interrupción

unsigned int vt[3283];//vector de tiempos
unsigned int ve[3283];//vector de estados
unsigned long acu_M=0;//acumulador de muestras ADCINT0
int vf[12]={0, 294, 600, 894, 1176, 1446, 1752, 2058, 2364, 2670, 2976,
3282};//vector de finales
int f=90;//frecuencia reloj 90MHz
int i=0;//variable de control
int pos=5;//modulacion actual
```

```
int posN=5;//modulacion objetivo

//definiciones por el enlazador F2808.cmd----------------------
extern Uint16 RamfuncsLoadStart;
extern Uint16 RamfuncsLoadEnd;
extern Uint16 RamfuncsRunStart;
extern Uint16 RamfuncsLoadSize;
//-------------------------------------------------------------

void main(void){


    InitSysCtrl();//Inicializar el control del sistema: PLL, WatchDog,
habilita los relojes a los periféricos F2806x_SysCtrl.c

    DINT;//Borrar todas las interrupciones e inicializar la tabla de
vectores PIE: Desactivar las interrupciones de la CPU

    InitPieCtrl();//Inicializar los registros de control PIE a su estado
predeterminado interrupciones deshabilitadas archivo F2806x_PieCtrl.c

    memcpy(&RamfuncsRunStart, &RamfuncsLoadStart,
(Uint32)&RamfuncsLoadSize); //Copia el código crítico de tiempo y el código
de configuración de Flash a la RAM
    InitFlash();//inicialización de flash ser grabada función de la memoria
RAM

    IER = 0x0000;// Desactivar las interrupciones de la CPU y borrar todos
los indicadores de interrupción de la CPU
    IFR = 0x0000;// Desactivar las interrupciones de la CPU y borrar todos
los indicadores de interrupción de la CPU

    InitPieVectTable();// Inicializar la tabla de vectores PIE con punteros
al shell Interrupción

    EALLOW;    //habilita escritura sobre registros de protegidos
    PieVectTable.TINT0 = &cpu_timer0_isr;//se le asigna la dirección de
función de interrupción a TINT0
    EDIS;     //deshabilita escritura sobre registros de protegidos

    EALLOW; //habilita escritura sobre registros de protegidos
    //Activar pines Como GPIO0-11--------------------------
       GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;
       GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;
```

```
        GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO7 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO8 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO9 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 0;
        GpioCtrlRegs.GPAMUX1.bit.GPIO11 = 0;
        //Activar GPIO0-11 como salidas-------------------------
        GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO4 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO5 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO6 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO7 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO8 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO10 = 1;
        GpioCtrlRegs.GPADIR.bit.GPIO11 = 1;
        EDIS;//deshabilita escritura sobre registros de protegidos
      kpp = 1.7;//constante proporcional
      kii = 100.0;//constante integral

    // Adquisición de los datos desde un archivo .txt----------------------
---
    FILE* fe;//puntero archivo Ve.txt
    FILE* fi;//puntero archivo Vt.txt
    fe = fopen("ve.txt","r");//open archivo Ve.txt
    fi = fopen("vt.txt","r");//open archivo Vt.txt
    for(i=0; i<3283;i++){
    fscanf(fe,"%u",&ve[i]);//almacenamiento vector Ve
    fscanf(fi,"%u",&vt[i]);//almacenamiento vector Vt
       }fclose(fe);
       fclose(fi);
    //-------------------------------------------------------------------
-----

    i=vf[pos]+1;//asignacion de pocision
    GpioDataRegs.GPADAT.all =ve[i];//Salida Modulación de control
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;//Bloquea demás interrupciones

        //-----------------------------------
```

```
    InitCpuTimers();//inicialice los temporizadores de CPU
     ope=vt[i];// Convierte Vt[i] en float
     ConfigCpuTimer(&CpuTimer0, 90,(ope/100));// Frecuencia de CPU de
90MHz, ope/100  (en uSegundos)

     CpuTimer0Regs.TCR.all = 0x4001;//habilita la interrupción del
temporizador

     IER |= M_INT1;// Asigna interrupción a nivel INT1 la interrupción

     PieCtrlRegs.PIEIER1.bit.INTx7 = 1;//Grupo 1 del pie habitado INTx7
individualmente
     EINT;//Habilitar la interrupción global INTM
     ERTM;//Habilitar DBGM de interrupción global en tiempo real
    while(1)
    {// Ciclo infinito

    }
}


__interrupt void
cpu_timer0_isr(void)
{   //Técnica de control-----------------------------------------------------
    i=i+1;
    if(i>vf[pos+1]){
       if(posN<pos){
           pos=pos-1;
           i=vf[pos]+1;}
        else{
            i=vf[pos]+1;
            pos=posN;}}
    GpioDataRegs.GPADAT.all =ve[i];//Salida Modulación de control

    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;//Bloquea demás interrupciones

    ope=vt[i];//Convierte Vt[i] en float

    ConfigCpuTimer(&CpuTimer0,f,(ope/100) );// Frecuencia de CPU de 90MHz,
ope/100  (en uSegundos)
    CpuTimer0Regs.TCR.all = 0x4001;//habilita la interrupción del
temporizador
}
```

4. ALGORITMOS DEL SISTEMA DE CONTROL EN EL TMS320F28069

4.1 Descripción general del algoritmo de control de velocidad del motor de inducción

El algoritmo general del control de velocidad del motor de inducción comprende la inclusión de cada una de las etapas antes propuestas en un mismo algoritmo. Con la correcta implementación de la técnica de control a nivel de código se procede a crear el algoritmo más general de control que comprende un controlador PI, junto con su entrada de retroalimentación provista por un puerto analógico; la descripción simplificada del procedimiento de desarrollo del algoritmo general de control se describe a continuación:

- Primero que todo se realizan las configuraciones necesarias de todos los módulos usados en el desarrollo PI, ADC, TIMER0 y GPIO
- El código en general va a realizar una interrupción en cada valor del vector de tiempo $Vt[tn]$ el cual coincide en su posición con su respectivo valor el vector de estados $Ve\ [En]$ en la misma posición "n" de acuerdo al límite puesto por el vector de finales $Vf[NfinalFn]$ en una modulación correspondiente a una frecuencia especifica.
- Cuando se ejecuta la rutina de interrupción primero que todo se captura el dato del sensor de velocidad externo AA0, el cual se muestrea en un Δt conocido luego se le saca la media dependiendo de la cantidad de datos medidos en ese tiempo, obteniendo una medida más exacta para ser pasada a su correspondiente valor en Rpm como $\boldsymbol{yk}$, se le envía al controlador PI que realiza su rutina de ejecución y devuelve una respuesta $\boldsymbol{uk}$ la cual realiza o no un cambio en la frecuencia para aumentar o disminuir la velocidad del motor.
- Después de saber la velocidad objetivo nueva proporcionada por el PI se realiza el cambio respectivo de acuerdo a unas condiciones y restricciones programadas en la técnica de control implementada en el capítulo anterior, cambiando los niveles lógicos de las salidas que van hacia los semiconductores de potencia del inversor trifásico.
- Por último, se actualiza el valor de tiempo en un $Vt[tn+1]$ en el registro para una nueva interrupción.

En ilustración 4.1 en modo de diagrama bloque se esquematiza el algoritmo general de control de velocidad con modulaciones optimizadas donde el XDS100V2 el programador la tarjeta de desarrollo.

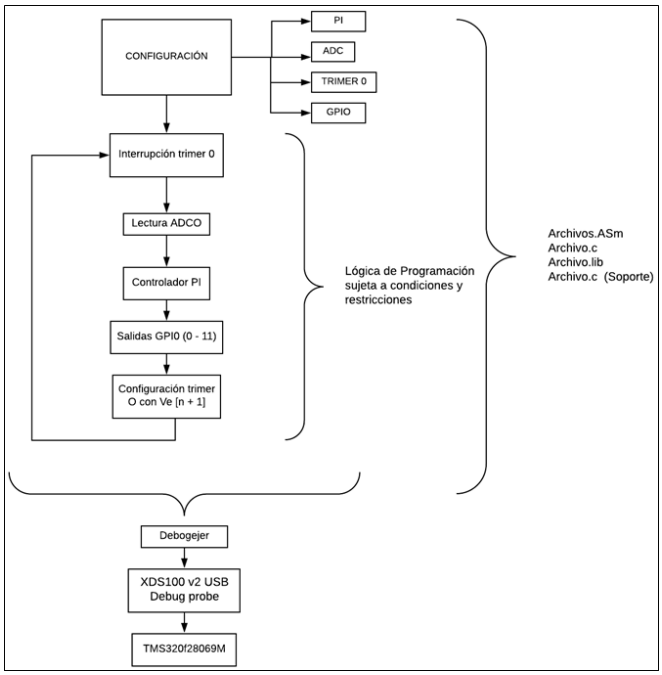


Ilustración 4. 1 Diagrama algoritmo general. Fuente: Autor

Donde el algoritmo de la técnica de control consta de 2 etapas las cuales son:

- Muestreo y adquisición de la velocidad a través del circuito de acondicionamiento
- Controlador PI de velocidad del motor

4.1.1 Muestreo y adquisición de la velocidad del circuito de acondicionamiento a través del ADC0

El módulo ADC programado es de 12 bits. Con 2 circuitos de muestreo y retención (S / H), un único núcleo de conversión vistos en la ilustración 4.2 junto con los registros correspondientes a su configuración. Los circuitos de muestreo y retención se pueden muestrear de forma simultánea o secuencial en este caso solo se muestrea de forma secuencial el módulo ADC cuenta con 16 canales de estrada analógica del cual solo se usa el ADC0. El principio básico de para configurar la operación de ADC se centra

en las configuraciones de conversiones individuales, denominadas ***SOC*** o inicio de conversiones. A continuación se enumeran las características destacables para su correcta configuración[20]:

- Núcleo ADC de 12 bits con doble muestreo y retención incorporados (S / H)
- Muestreo simultáneo o modos de muestreo secuencial.
- Entrada analógica de rango completo: 0 V a 3,3 V fija
- 16 canales.
- 16 SOC, configurables para disparador, ventana de muestra y canal
- 16 registros de resultados para almacenar valores de conversión.
- Múltiples fuentes de disparo
  - S / W - inicio inmediato del software
  - ePWM 1-8 GPIO
  - XINT2
  - **CPU Timers 0/1/2 (usada como fuente de disparo el TIMER-0)**
  - ADCINT1 / 2

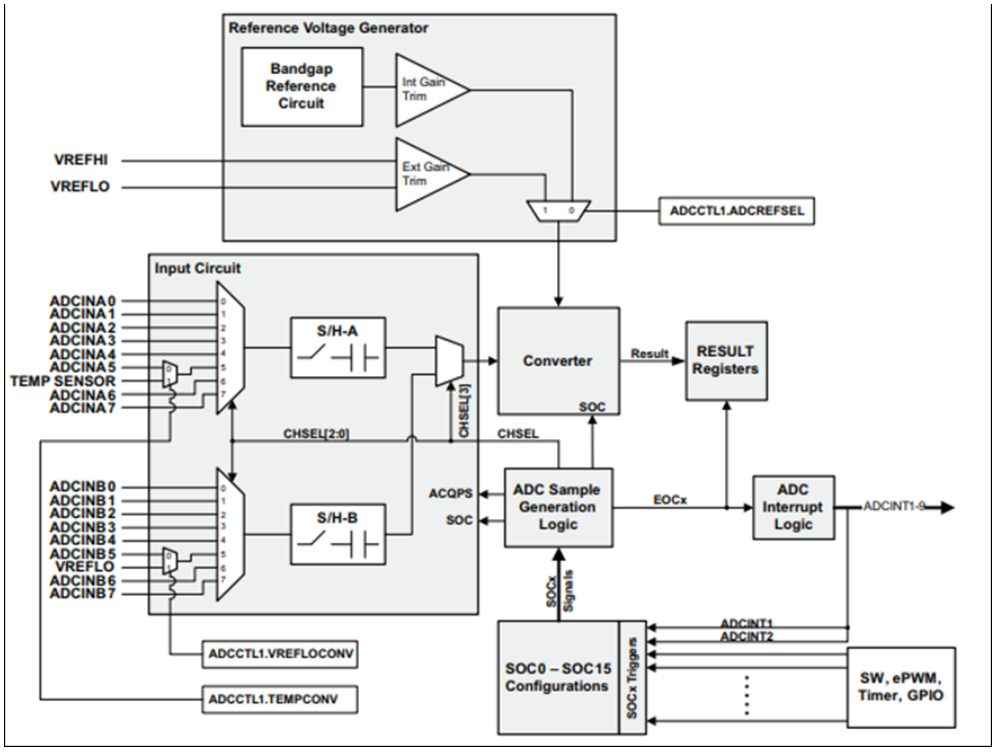


Ilustración 4. 2 Diagrama de bloques ADC. Fuente: [21]

- **Principio de Operación SOC**

En general hay que configurar en el SOC tres configuraciones generales: la fuente de activación que inicia la conversión, el canal a convertir y el tamaño de la ventana de adquisición (muestra). La fuente de activación para SOC0 se configura mediante una combinación del campo TRIGSEL en el registro ADCSOC0CTL y los bits apropiados en el registro ADCINTSOCSEL1. El canal y el tamaño de la ventana de muestra para SOC0 se configuran con los campos CHSEL y ACQPS del registro ADCSOC0CTL en la ilustración 4.3 se ejemplifica lo antes descrito para analizar que registros son relevantes para el desarrollo[20].

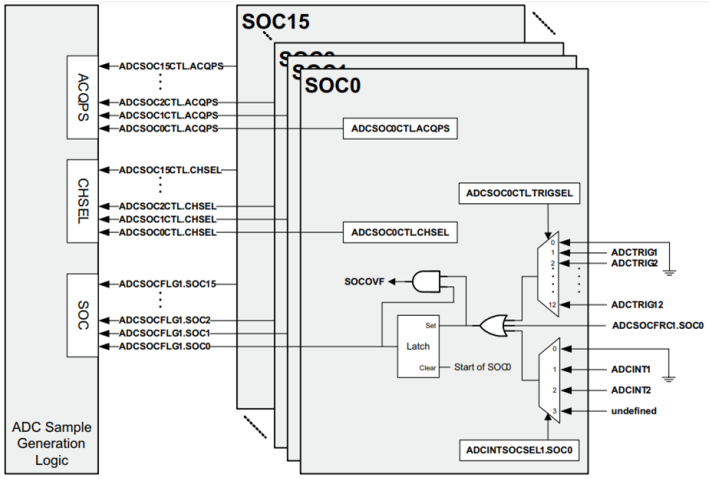


Ilustración 4. 3 Diagrama SOC. Fuente: [21]

Las principales configuraciones a nivel general que se realizan al ADCINT0 son las descritas anteriormente, seguido de estas configuraciones se procede a realizar la configuración del SOC descrito con anterioridad resaltando las tres configuraciones generales.

- La selección del canal esto se realiza en el registro SCO0 en el bit de registro CHSEL el cual como ya se menciono va a ser el ADCINT0 cargando un bit de 0 en el registro.
- La selección de la fuente de inicio de conversión la cual se configura para que sea cuando el TIMER-0 cumpla con el tiempo cargado en sus registros de conteo, se realiza cargado un bit de 1 en el registro SCO0 en el bit de registro TRIGSEL.

- Por último, se selecciona la ventana de conversión antes descrita en esta sección la cual va a tener el tiempo más rápido posible, lo cual se realiza cargando el valor decimal de 6 para indicarle que la ventana S/H a 7 ciclos del reloj.
- Par obtener el valor de la lectura del ADC se puede obtener en el registro de solo lectura ADCRESULT0 el cual está directamente relacionado con SCO0, el dato va a ser la entrada AA0 la cual va a ser la variable de adquisición de datos que se almacenan todas las muestras de velocidad en el acumulador acu_M, acumulando una cantidad de datos "j" definida por el tiempo de la Δt acción de control que es 4 ciclos de la frecuencia mayor lo que me garantiza una acumulación de datos mayor a 200 datos, la cual a través de lo descrito en el numeral 2 se le realiza un filtrado digital a los datos obteniendo así el valor que se le asigna a la entrada del controlador PI descrito en a continuación.

En la ilustración 4.4, se describe por medio de diagrama de bloques el algoritmo implementado para la configuración los ADCINA0 como entrada análoga para la lectura de la velocidad proveniente del sensor de velocidad el cual tiene un tiempo de muestreo dependiente del vector de tiempos Vt obtenidos, descritos en secciones anteriores de este trabajo.

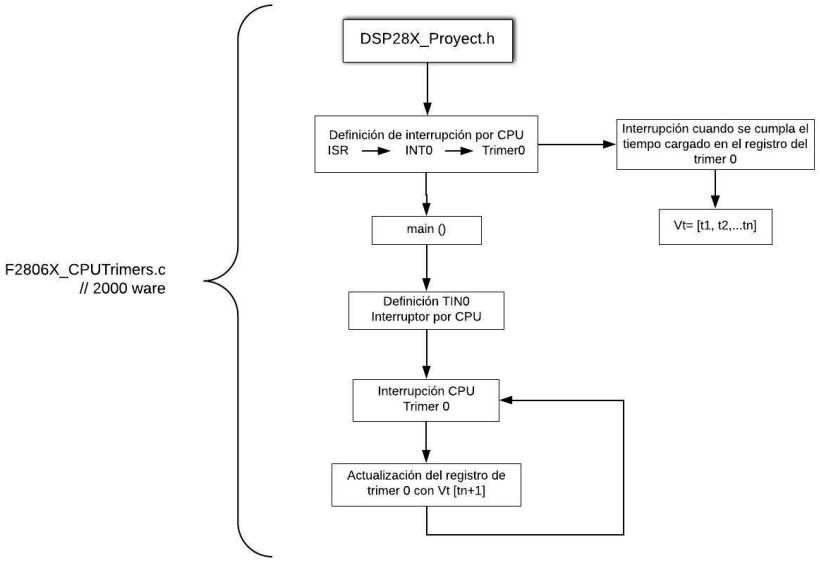


Ilustración 4. 4 Algoritmo del ADC. Fuente: [21]

Algoritmo en CCS desarrollado en code composer studio para adquirir la señal de velocidad en el ADCINT0

```
#include "DSP28x_Project.h" //archivo encabezado del dispositivo donde se
encuentran funciones ejemplo del dispositivo
#include "string.h"
#include <stdio.h>

__interrupt void cpu_timer0_isr(void);//interrupción externa, inicio de
rutina de servicio de interrupción

float yk = 0; // Entrada del sensor en Rpm
float acu_t;//Tiempo de controlador
float media;//resultado de la integral proyectiva
unsigned long acu_M=0;//acumulador de muestras ADCINT0
int i=0;//variable de control
int j=0;//contador numero de muestras capturadas
int AA0;//Muestras ADCINT0

void main(void){
  InitSysCtrl();//Inicializar el control del sistema: PLL, WatchDog,
habilita los relojes a los periféricos F2806x_SysCtrl.c
    EALLOW; //habilita escritura sobre registros de protegidos
    AdcRegs.ADCCTL1.bit.INTPULSEPOS = 0;//borde negativo del pulso
    AdcRegs.INTSEL1N2.bit.INT1E = 1; // activar ADCINT1
    AdcRegs.INTSEL1N2.bit.INT1CONT = 0; //deshabilitar el modo continuo
    AdcRegs.INTSEL1N2.bit.INT1SEL = 1; //fuente de interrupción
    AdcRegs.INTSEL1N2.bit.INT2E = 0; // activar ADCINT2
    AdcRegs.INTSEL1N2.bit.INT2CONT = 0; //deshabilitar el modo continuo
    AdcRegs.INTSEL1N2.bit.INT2SEL = 0; //fuente de interrupción
    AdcRegs.ADCSOC0CTL.bit.CHSEL = 0; // selección del canal ADCINA0
    AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 1;//sincronizador de captura TIMER0
    AdcRegs.ADCSOC0CTL.bit.ACQPS = 6;//Ventana de muestreo de 7 ciclos de
reloj
    EDIS;//deshabilita escritura sobre registros de protegidos
    while(1)
    {// Ciclo infinito

        AA0 = AdcResult.ADCRESULT0; // Captura de datos
 }
}

__interrupt void
cpu_timer0_isr(void)
{
```

```
    acu_M=acu_M+AA0; //Acumulación de los datos
    acu_t=(ope/100)+acu_t; //Contador de tiempo
     j++;//Contador de datos
     if(acu_t>=7272.2727){ //Condición de Tiempo de controlador
          media=acu_M/j; //Media
          j=0;
          yk = media*0.86424 - 19.57; //transformación a Rpm
          acu_M=0;}

}
```

4.1.2 Descripción del controlador del sistema de control de velocidad

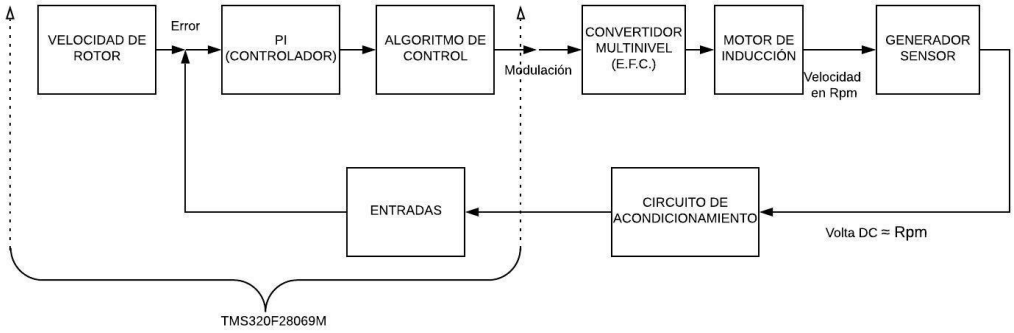


Ilustración 4. 5 Lazo de control del sistema. Fuente: [21]

El control se fundamenta en el principio del lazo mostrado en figura, donde el valor de la velocidad del rotor depende de un motor acoplado al eje que funciona como generador produciendo un nivel de voltaje DC que es proporcional a la velocidad en un instante de tiempo ilustración 4.5.

- **Controladores PI lineales**

Las aplicaciones como la de control de un motor de inducción que no requieren acción derivada, o que son más sensibles a la eficiencia del ciclo, pueden ser mejor atendidas por la estructura más simple del controlador PI descrita en la siguiente sección por su simplicidad en implementación y su poco tiempo de respuesta[22].

$$u(t) = k_p e(t) + k_i \int_{-\infty}^{t} e(\tau) d\tau \qquad (4.2)$$

Su diagrama en bloque se muestra en la siguiente ilustración 4.6 la cual solo cuenta con la suma de su canal proporcional e integral.

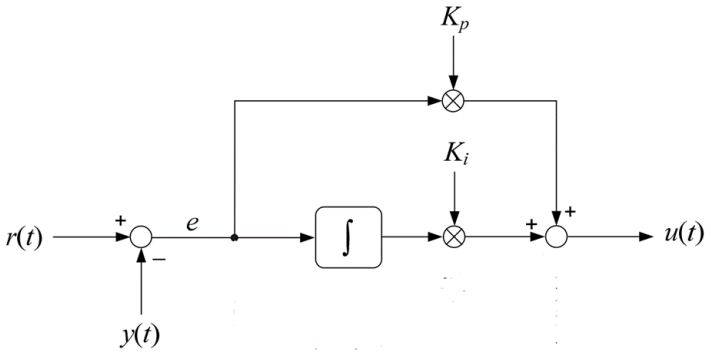


Ilustración 4. 6 Controlador PI. Fuente: [20]

- **Implementación**

Las arquitecturas del controlador PI se muestran en los anteriores diagramas en la aplicación de código el PI, que es un tanto diferente al diagrama simple antes expuesto, en la siguiente ilustración se puedes apreciar que el PI cuenta con un saturador. El cual actúa conforme los límites máximos de nuestro control a implementa.

El código se implementó teniendo en cuenta los alcances y limitaciones de todas las etapas del algoritmo que se explicaron; el controlador a implementar es el PI pues no es muy necesario su acción derivativa por la naturaleza de la planta, la metodología seguida para el cálculo de las constantes proporcional e integral se describe en un capítulo más adelante. Para la elaboración de los algoritmos del controlador PI en primera instancia se debe garantizar que el tiempo entre una acción de control y la siguiente sea constante, para el correcto funcionamiento de la acción de control y asegurar que la planta tenga un tiempo de respuesta, una breve descripción del algoritmo efectuado se muestra a continuación.

- Se nombra e inicializan las variables relevantes del lazo de control en diagrama de bloques antes expuesto declarando Kp, Ki, rk e yk.
- Se inician las variables con valores de entrada Kp, Ki, rk, Umax, Umin entre otros para el conecto funcionamiento de todos los lazos del controlador.

- Para determinar el tiempo entre una acción de control con el fin de que se garantice un tiempo en que la planta reaccione ante la acción de control de forma correcta se calculó como 4 ciclos de la frecuencia mayor $72727.2727\mu S$.
- Una vez se comprueba el tiempo de control se calcula el valor medio de los datos muestreados durante el Δt con el fin de aplicar un filtrado digital, en base a la definición de integral proyectiva que se utilizan en tratamiento de señales finitas; a este valor se calcula su respectiva valor de yk(media) en Rpm pues el controlador está calculado en velocidad de Rpm .
- Se calcula el valor de cada lazo del controlador PI en base al error calculado a partir yk y rk, obteniendo a respuesta del controlador uk, este valor se traduce a una modulación objetivo nombrado en código como posN.

En base a todo lo descrito anteriormente se desarrolló la correcta configuración e implementación del controlador PI en el lazo de control de velocidad el cual se describe por medio de diagrama de bloques en la ilustración 4.7.

Ilustración 4. 7 Algoritmo PI. Fuente: Autor.

Algoritmo en CCS desarrollado en code composer studio para controlador PI del sistema de control de velocidad

```
#include "DSP28x_Project.h" //archivo encabezado del dispositivo donde se
encuentran funciones ejemplo del dispositivo
#include "string.h"
#include <stdio.h>
__interrupt void cpu_timer0_isr(void);//interrupción externa, inicio de
rutina de servicio de interrupción

float uk; // salida controlador
float rk = 1471; // referencia de velocidad en RPM
float E_ant = 0; // Error-1
float yk = 0; // Entrada del sensor en Rpm
float kpp, kii,l_pro,l_inte,Err;// constantes, lazo proporcional, lazo
integral, error
float acu_t;//Tiempo de controlador
float ope;//variable de almacenamiento de tiempos
float media;//resultado de la integral proyectiva
unsigned long acu_M=0;//acumulador de muestras ADCINT0
int posN=5;//modulacion objetivo
int j=0;//contador número de muestras capturadas
int offset=1467;//suma a la salida
int AA0;//Muestras ADCINT0
void main(void){
      kpp = 1.7;//constante proporcional
      kii = 100.0;//constante integral

    while(1)
    {// Ciclo infinito
        AA0 = AdcResult.ADCRESULT0;// Captura de datos
    }
}
__interrupt void
cpu_timer0_isr(void)
{    acu_M=acu_M+AA0;//Acumulación de los datos
    acu_t=(ope/100)+acu_t;//Contador de tiempo
     j++;//Contador de datos
     if(acu_t>=72727.2727){//Condición de Tiempo de controlador

          media=acu_M/j;//Media o integral proyectiva
          j=0;
          yk = media*0.86424 - 19.57; //transformación a Rpm
          Err=(rk-yk); //cálculo del error
          l_pro=Err*kpp; //lazo proporcional
```

```
            l_inte=((E_ant+Err)*(acu_t/1000000))*kii;//lazo integral
            acu_t=0;
            E_ant=Err;
            uk=l_pro+l_inte+offset; //respuesta del controlador
            posN=(0.034021*uk-44.86)+0.4; //cálculo de modulación objetivo
            if(uk>1615){posN=10;} //saturador
            if(uk<1320){posN=0;} //saturador
            acu_M=0;}
}
```

En el **anexo 2**. Se puede observar el algoritmo completo desarrollado el cual comprende cada parte de código descrita a lo largo de estos capítulos.

5. RESULTADOS DE FUNCIONAIENTO DEL SISTEMA DE CONTROL

5.1 Pruebas de funcionamientos y validación

Para las pruebas de funcionamiento y validación de todas las etapas propuestas en los capítulos anteriores se utilizaron diversos elementos de medida y extracción de datos los cuales se encuentran en los laboratorios y/o grupos de investigación de la universidad de pamplona con el fin de verificar el correcto funcionamiento de cada etapa antes planteada para realizar una validación final del funcionamiento, los equipos continuación mencionados están certificados por estándares y normas que los rigen.

Equipos de medición y registro de datos

Osciloscopio digital portátil de 20MHz - Fluke 123

Descripción

Osciloscopio de Almacenamiento Digital ScopeMeter® de 20 MHz con Multímetro y Registrador de Datos; Pantalla Monócroma. Modelo Fluke 123 Codigo: FLUKE-123/008[23].

Características

- Osciloscopio digital de dos canales de 20 MHz
- TrendPlot™: registrador de dos canales
- Función disparo automatico Connect-and-View™ para funcionamiento sin manos
- Cables simples de prueba con blindaje para mediciones de osciloscopio, resistencia y continuidad
- Duración de las baterías: hasta 7 horas
- Seguridad eléctrica: CAT III 600 V
- Interfaz RS-232 ópticamente aislada
- Equipo compacto y muy robusto
- 2 memorias de pantalla y 10 configuraciones

Ilustración 5. 1 Fluke 123. Fuente: [23].

Analizador de calidad de la energía eléctrica - Fluke 434-II

Descripción

Analizador de calidad de la energía eléctrica (PQ) trifásico, cuenta con la nueva función de cálculo de pérdida de energía, cuenta también con la medición de potencia eléctrica unificada (UPM), además podrá detectar y solucionar problemas en primer nivel de actuación, realizar un mantenimiento predictivo, hacer análisis a largo plazo y estudios de carga; incluye las nuevas pinzas i430flex-TF. Modelo Fluke 434-II Codigo: FLUKE-434-II[23].

Características

Calculadora de Pérdida de Energía.

- La clasificación de seguridad más alta de la industria: Clasificación de **CAT IV** a 600V y **CAT III** a 1.000V para su uso en la entrada de servicio.
- Permite medir las tres fases y el neutro.
- Todas las mediciones se registran siempre automáticamente, sin necesidad de configuración alguna.
- Diez parámetros de calidad de potencia en una sola pantalla, de acuerdo con la norma de calidad de potencia eléctrica **EN50160.**

Ilustración 5. 2 Fluke 434-II
Fuente: [23].

- Función de registrador: Configurado para cualquier condición de prueba con memoria de hasta 600 parámetros a intervalos definidos por el usuario.
- Visualización de gráficos y generación de informes.
- Medición de Potencia Eléctrica Unificada.

5.1.1 Funcionamiento de modulaciones optimizadas (Salida de los $GPIO_0$-$GPIO_{11}$)

En la ilustración 5.3 se puede observar montaje realizado con el fin de validar las salidas de las modulaciones en modo de diagrama de bloques.

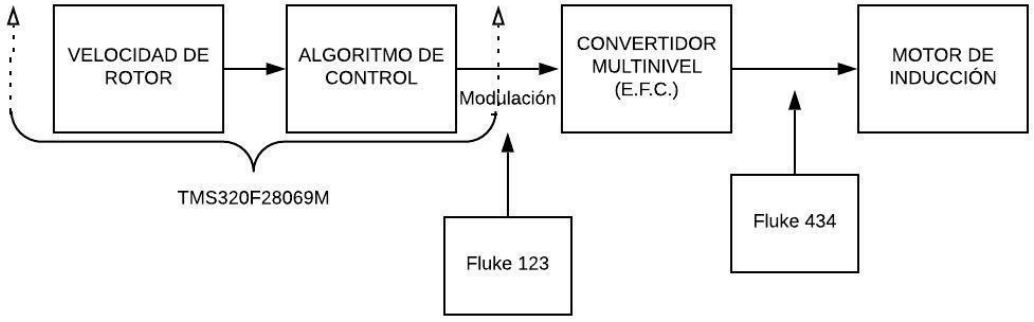


Ilustración 5. 3 Diagrama del montaje Fuente: Autor

Las modulaciones provenientes del microcontrolador hacia la entrada de los Mosfet de control del inversor trifásico, deben ser validadas con el fin de que cumplan las restricciones mismas del diseño del inversor trifásico adoptado. En las ilustraciones 5.4 - 5.8, se presentan las medidas registradas de cada frecuencia del control establecido, en la medida solo se tomó la suma de todos los tiempos de modulación del pin GPIO0 el dual tiene unos tiempos de conmutación de baja frecuencia registradas en el osciloscopio ***Fluke 123***.

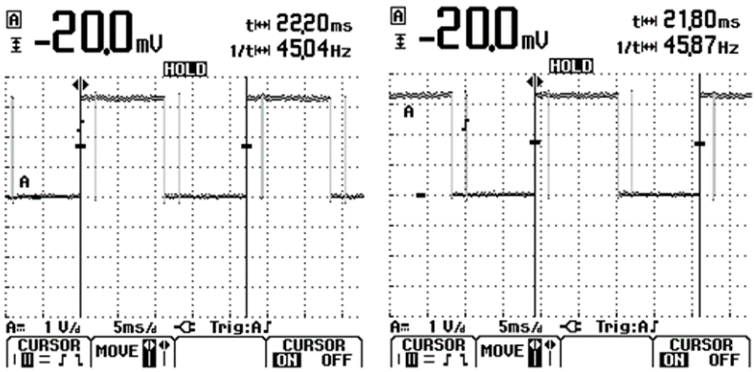


Ilustración 5. 4 Modulaciones de salida a) GPIO 0 a 45 Hz. B) GPIO 0 A 46 Hz Fuente: Autor

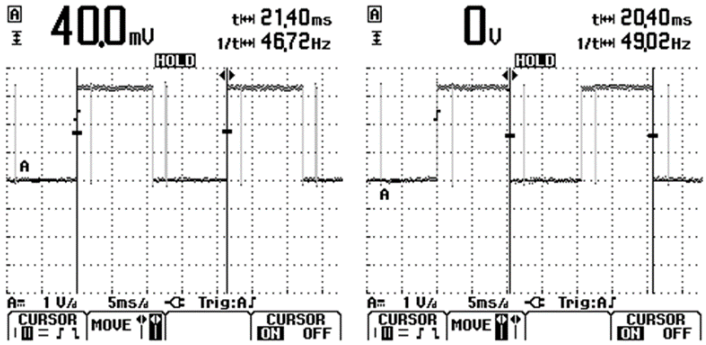


Ilustración 5. 5 Modulaciones de salida a) GPIO 0 a 47 Hz. B) GPIO 0 A 49 Hz Fuente: Autor

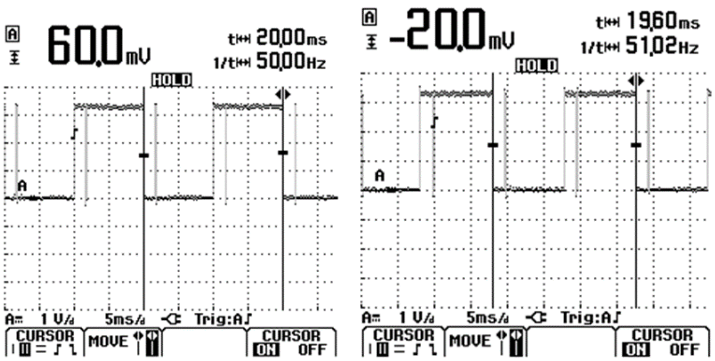


Ilustración 5. 6 Modulaciones de salida a) GPIO 0 a 50 Hz. B) GPIO 0 A 51 Hz Fuente: Autor

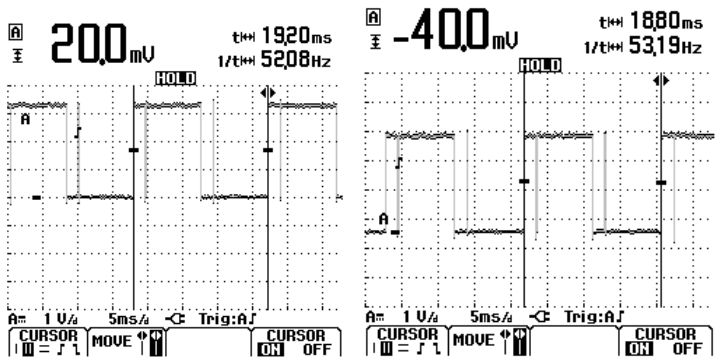


Ilustración 5. 7 Modulaciones de salida a) GPIO 0 a 52 Hz. B) GPIO 0 A 53 Hz Fuente: Autor

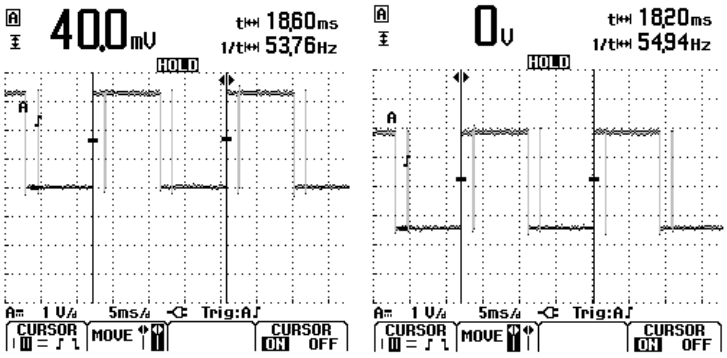


Ilustración 5. 8 Modulaciones de salida a) GPIO 0 a 52 Hz. B) GPIO 0 A 53 Hz Fuente: Autor

Como se puede observar en las anteriores medidas las frecuencias de cada modulación optimizada se encuentran con un pequeño porcentaje de error, lo que comprueba el correcto funcionamiento de la lógica del algoritmo planteado para las modulaciones entre la interrupción de la CPU y el cambio de estados de los GPO's.

5.1.2 Formas de onda salida del inversor trifásico hacia el motor de inducción pruebas a lazo abierto

En la ilustración 5.9 se puede observar montaje realizado con el fin de validar las salidas del sistema a lazo abierto en modo de diagrama de bloques, en la ilustración 4.10 se observa el montaje real realizado para validar el funcionamiento.

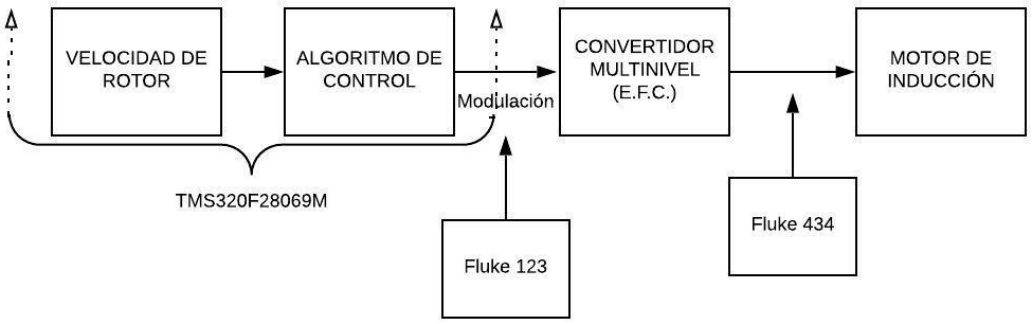


Ilustración 5. 9 Montaje prueba a lazo abierto Fuente: Autor

Ilustración 5. 10 Montaje. Fuente: Autor

Las formas de onda generadas por la conmutación de los semiconductores Mosffet en conjunto por el bus de CD del inversor de potencia trifásico deben tener un reducido porcentaje de THD para así comprobar el funcionamiento del prototipo planteado hasta este punto, las formas de onda cuasi sinusoidales obtenidas y mediadas para validar tanto la lógica de programación como la correcta aplicación del control escala del motor.

Para realizar la medida y validación de estos parámetros se usó el analizador de redes ***Fluke 435*** que permite obtener la medida de las tres fases de alimentación del motor

en las ilustraciones 4.11 – 4.21 se presenta las medidas obtenidas junto con el montaje empleado.

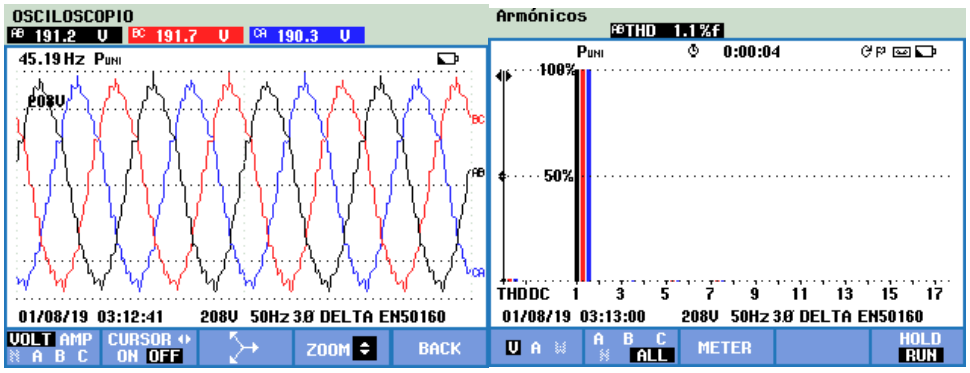


Ilustración 5. 11 Formas de onda para voltaje de línea y THD para una fn≈45Hz. Fuente: Autor

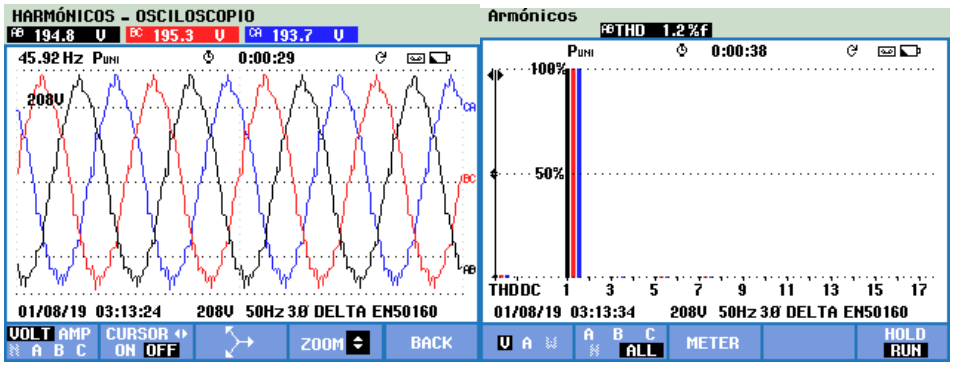


Ilustración 5. 12 Formas de onda para voltaje de línea y THD para una fn≈46Hz. Fuente: Autor.

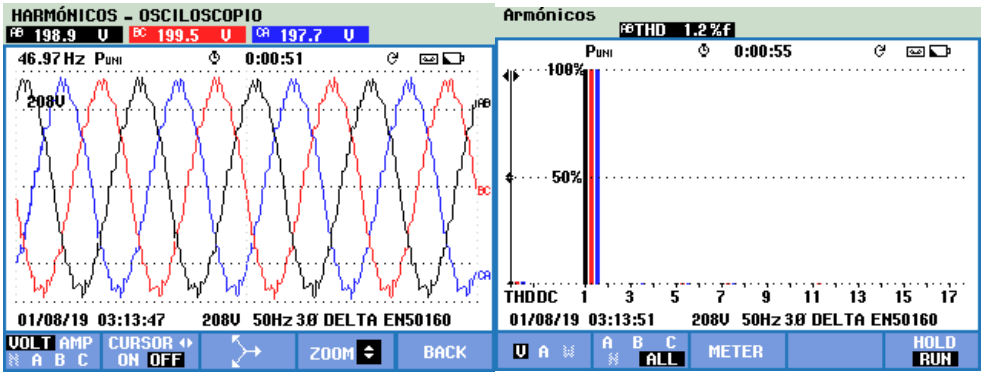


Ilustración 5. 13 Formas de onda para voltaje de línea y THD para una fn≈47Hz . Fuente: Autor

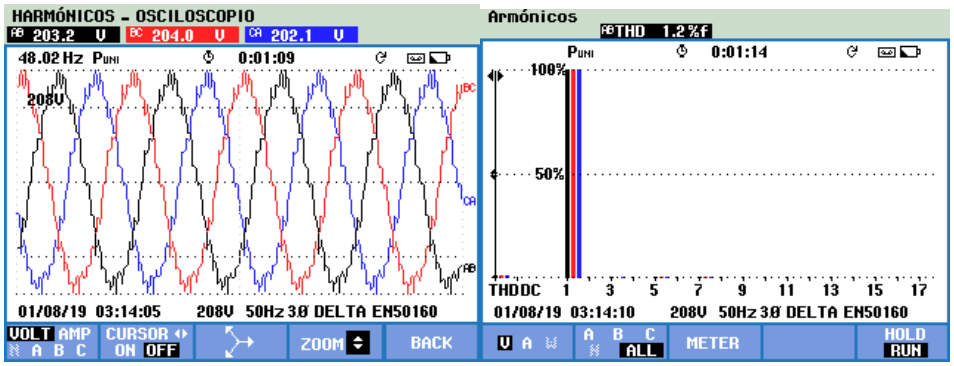


Ilustración 5. 14 Formas de onda para voltaje de línea y THD para una fn≈48Hz. Fuente: Autor.

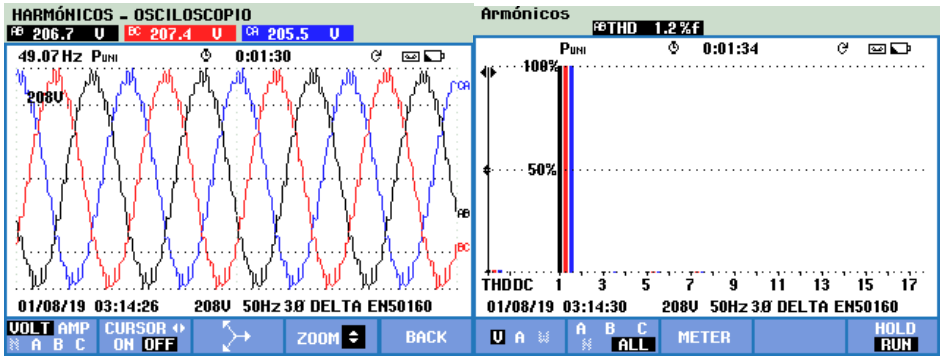


Ilustración 5. 15 Formas de onda para voltaje de línea y THD para una fn≈49Hz . Fuente: Autor.

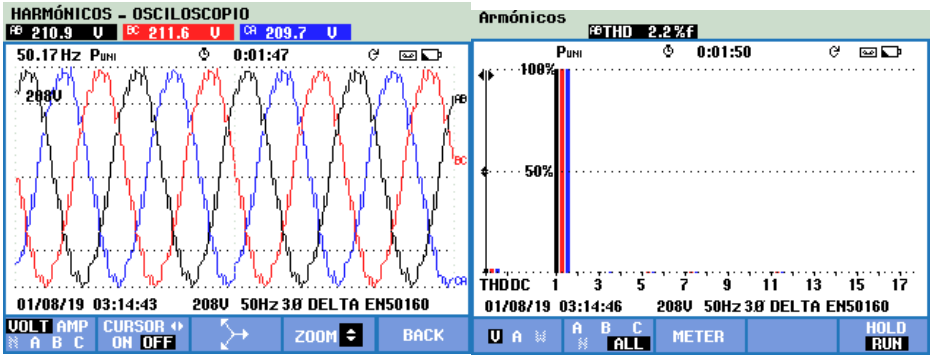


Ilustración 5. 16 Formas de onda para voltaje de línea y THD para una fn≈50Hz . Fuente: Autor.

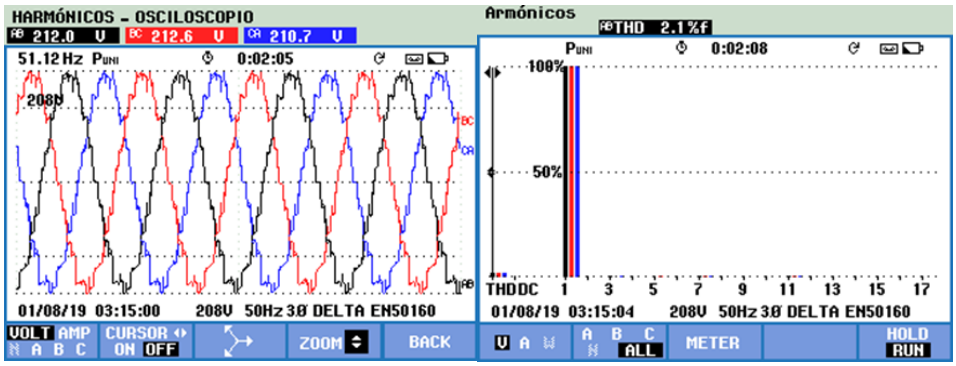


Ilustración 5. 17 Formas de onda para voltaje de línea y THD para una fn≈51Hz . Fuente: Autor.

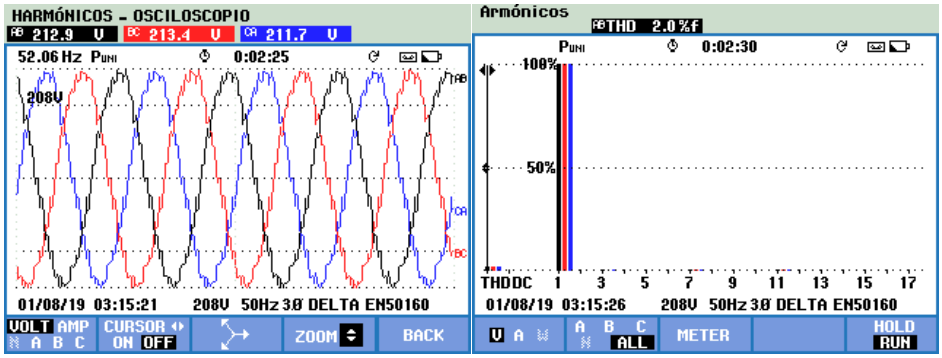


Ilustración 5. 18 Formas de onda para voltaje de línea y THD para una fn≈52Hz . Fuente: Autor.

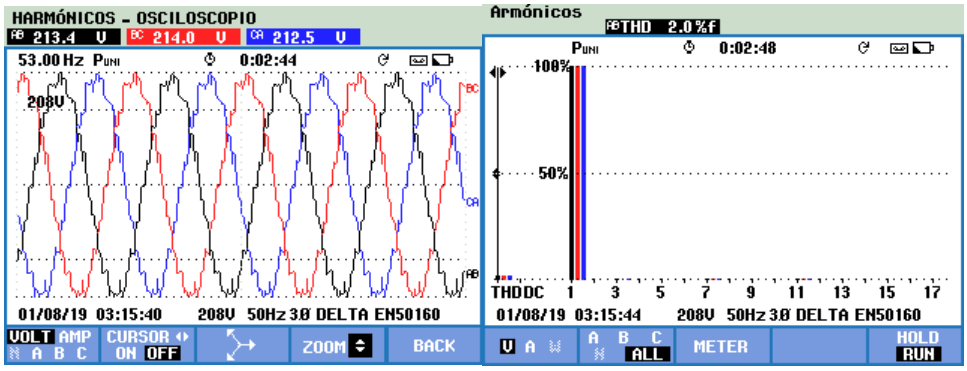


Ilustración 5. 19 Formas de onda para voltaje de línea y THD para una fn≈53Hz . Fuente: Autor.

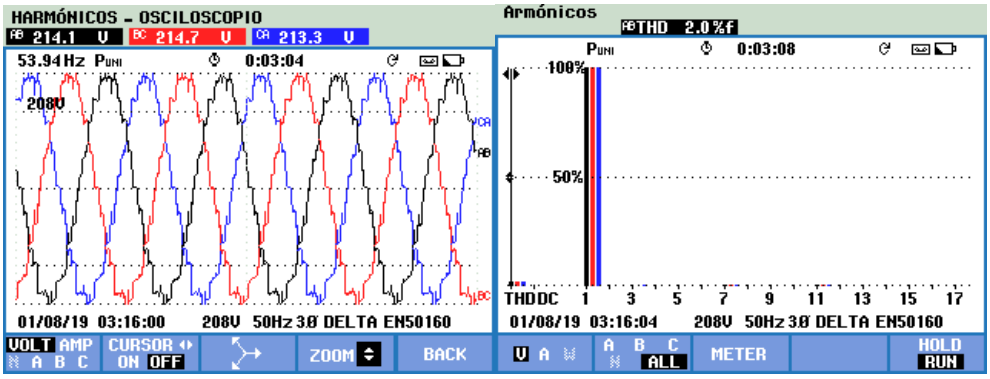


Ilustración 5. 20 Formas de onda para voltaje de línea y THD para una fn≈54Hz . Fuente: Autor.

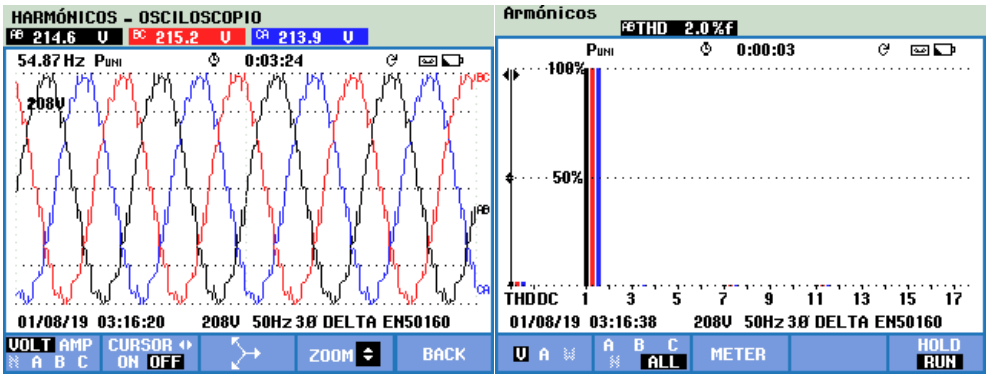


Ilustración 5. 21 Formas de onda para voltaje de línea y THD para una fn≈55Hz . Fuente: Autor.

Como se puede observar las formas de onda obtenidas tiene una similitud casi sinusoidal pura la cuales contienen una calidad de energía muy alta esto gracias al control ejercido por la tarjeta de desarrollo propuesta TMS320F2806, a continuación se presenta en la tabla 5.1 el comportamiento de la ley escalar descrita en capítulos anteriores.

Tabla 5. 1 Control escalarl.

Fn	VL	%THD	Rpm
45,19	190	1,1	1320
45,92	194	1,2	1344
46,97	198	1,2	1374
48,02	202	1,2	1407
49,07	205	1,2	1437
50,17	210	2,2	1471
51,12	211	2,1	1499
52,06	211	2	1526
53	212	2	1553
53,94	213	2	1581
54,87	213	2	1609

Con la medida y validación de los anteriores datos se construye la gráfica del control escala planteado y creado con las modulaciones optimizadas la cual se observa en la ilustración 5.22 donde se resalta el punto de trabajo del sistema de control.

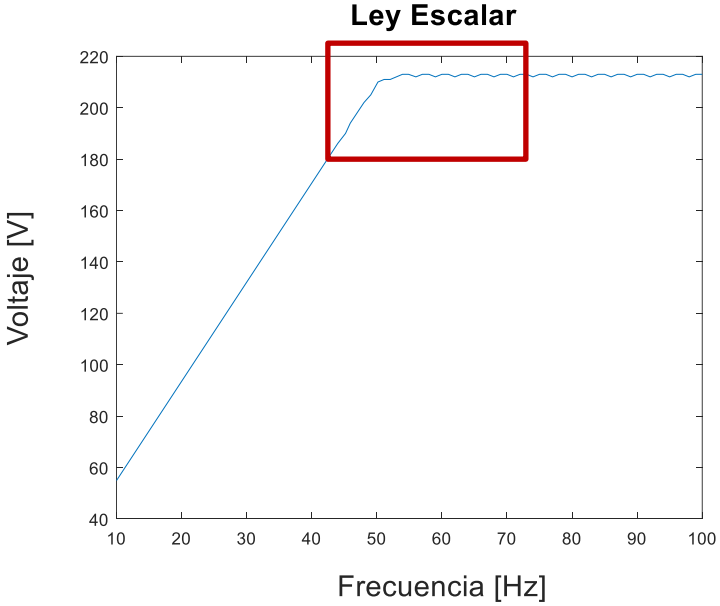


Ilustración 5. 22 Curva control escalar. Fuente: Autor

En la ilustración 5.23, se presenta el diagrama fasorial para corroborar el correcto desfase de los vectores de voltaje de línea junto con las formas de onda de la corriente en la cual se puede observar una las tres formas de onda esperadas casi sinusoidal con muy buena calidad de energía.

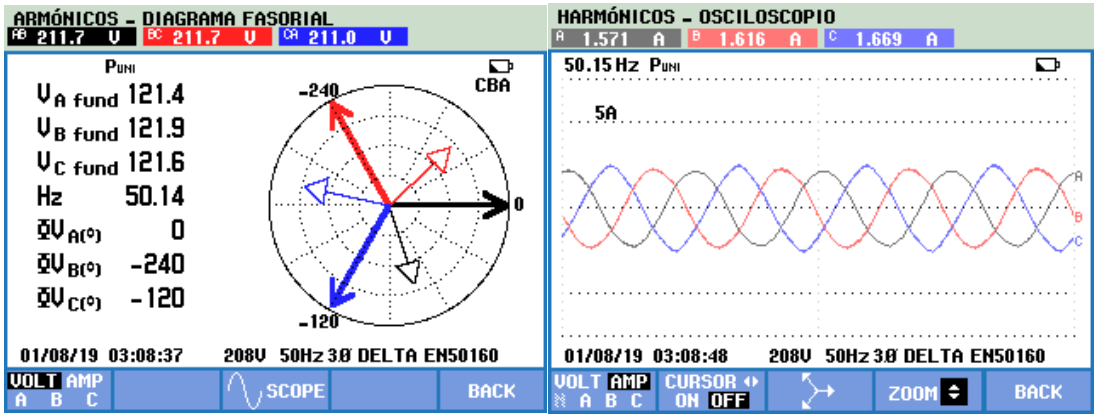


Ilustración 5. 23 Diagrama fasorial. Fuente: Autor.

5.1.3 Comparación con métodos tradicionales

Para corroborar los resultados obtenidos hasta este punto del trabajo, se realizó el montaje del mismo esquema , pero con un convertidor de frecuencia industrial el cual está presente en los laboratorios y/o grupos de investigación el cual cumple todos los estándares y normativas industriales[24] , se realizó el siguiente montaje ilustración 5.24.

Ilustración 5. 24 Montaje. Fuente: Autor.

Donde se obtuvieron las siguientes mediciones ilustración 5.25 – 5.26 y se compararon con las obtenidas por el desarrollo ya implementadas.

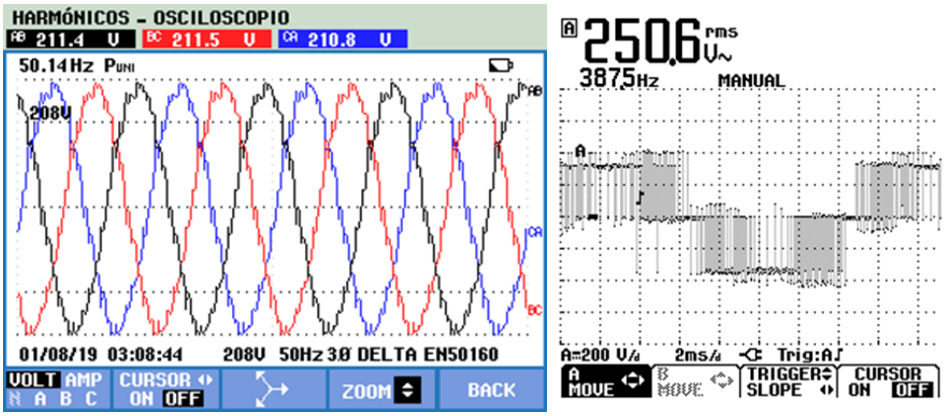


Ilustración 5. 25 Formas de onda a 50Hz. Fuente: Autor.

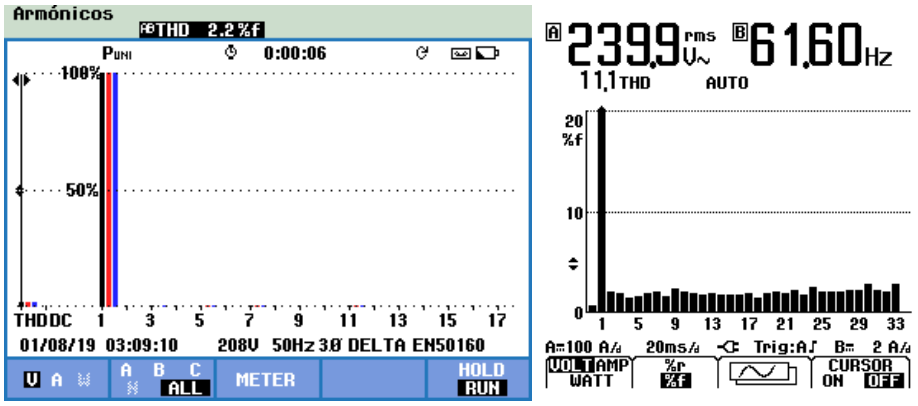


Ilustración 5. 26 %THD a 50Hz. Fuente: Autor.

En las anteriores ilustraciones se puede analizar a simple vista la gran diferencia que se presenta tanto en las formas de onda y %THD con lo cual obtenemos una base comparativa con la cual se puede validar un buen desarrollo metodológico del proyecto. Donde el convertidor de frecuencia usado tiene características elevadas en costos tanto económicos y operación ya que debe ser manipulado por personal calificado.

En la tabla se realiza una relación de los costos relacionados con los sistemas comparados anteriormente, donde el sistema desarrollado con la TMS320x se incluye el inversor multinivel adoptado junto con los costos relacionados a la investigación

expuesta en todo el libro y el sistema de control con un convertidor de frecuencia además del valor comercial se le coloco el valor de investigación ya que este dispositivo solo lo pueden operar personas capacitadas que realizaron alguna investigación previa para poder configurarlo correctamente.

Tabla 5. 2 Relación de costos

Sistema de control de control de velocidad desarrollado		**Método de control de velocidad convencional**	
Sistema	**Precio COP**	**Sistema**	**Precio COP**
TMS320F28069M	$55.288,00	Convertidos de frecuencia ABB, ACS800	$14.613.496,49
LAUNCHXL-F28069M	$86.352,94		
Inversor multinivel	$2.000.000,00		
Costos de investigación	$1.500.000,00	Costos de investigación asociados a su operación	$1.500.000,00
Totales	$3.586.352,94	Totales	$16.113.496,49

En base a los precios calculados se puede decir que el sistema de control desarrollado presenta una gran viabilidad en comparación a métodos tradicionales

5.1.4 Cálculo de las constantes Kp y Ki

Para realizar el cálculo de las constantes Kp y Ki necesaria para la implementación del algoritmo del controlador el cual se va a estar ejecutando en tiempo real definido por el valor contenido en un instante de tiempo en el vector de tiempos Vt, en primera instancia para el cálculo de parámetros las constantes se hace necesario realizar pruebas de funcionamiento a lazo abierto a la planta (Motor de inducción) de acuerdo al siguiente diagrama ilustración 5.27.

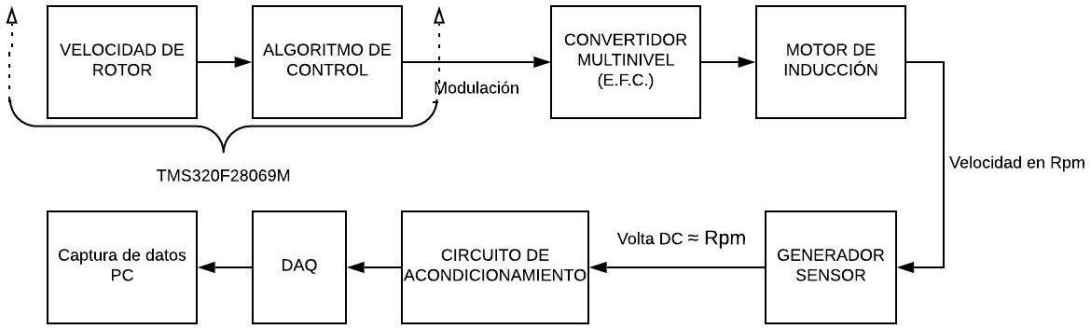


Ilustración 5. 27 Montaje en diagrama de bloques. Fuente: Autor

De acuerdo al montaje realizado se varió el sistema, empezó a funcionar en una velocidad de aproximada de $1320\,Rpm$ que correspondiente de a una frecuencia aproximada de 45Hz teniendo en cuenta el deslizamiento S, luego se realizó un cambio de velocidad con un step U registrado en el GPIO23 a una velocidad objetivo de $1581\,Rpm$, obteniendo el comportamiento del sistema ante un cambio de velocidad la cual es muestreada con un DAQ a una tasa de adquisición de $\frac{1}{20kHz}$, los resultados obtenidos se puedes apreciar en la ilustración 5.28.

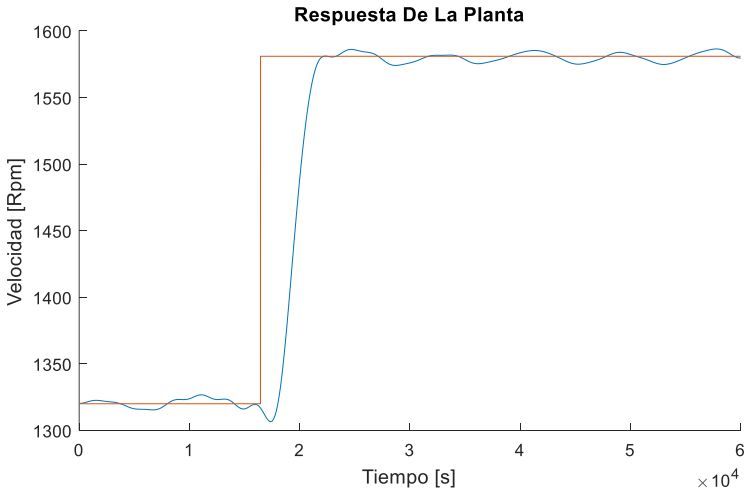


Ilustración 5. 28 Lectura de control a lazo abierto. Fuente: Autor

Que al graficarse junto con la acción de control tipo escalón se puede observar el tiempo que tarda en responder el sistema a la nueva velocidad la cual es la más alta posible que es aproximadamente $800ms$.

Para el cálculo de los parámetros del controlador implementado se sintonizó en Matlab, partiendo de la función de transferencia obtenida de la identificación completa del sistema en lazo abierto; donde el TMS320F28069M controla el convertidor multinivel de acuerdo a las etapas antes planteadas junto con el sensor de velocidad acoplado al eje del rotor; con la herramienta de identificación de sistemas IDENT de Matlab podemos obtener un modelo del comportamiento de nuestra planta a partir de unas variables de entrada como se puede ver en la ilustración 5.29.

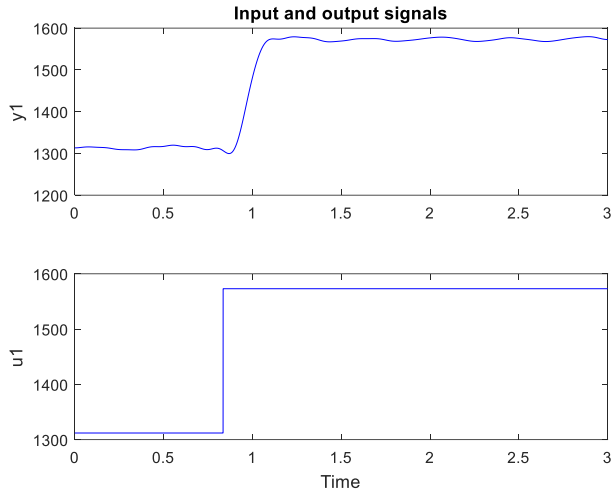


Ilustración 5. 29 Datos cargados a Ident Fuente: Autor Ident-Matlab

Se obtiene un modelo con el 97,2% de similitud ilustración 5.30 con la planta real para poder realizar la sintonización e las constantes del controlador a implementar donde tenemos el siguiente modelo para nuestra planta que se puede observar, cuenta con 3 polos y 2 ceros.

$$\frac{-0.4769 * S^2 - 236.1 * S + 7810}{S^3 + 45.64 * S^2 + 921.1 * S + 7808}$$

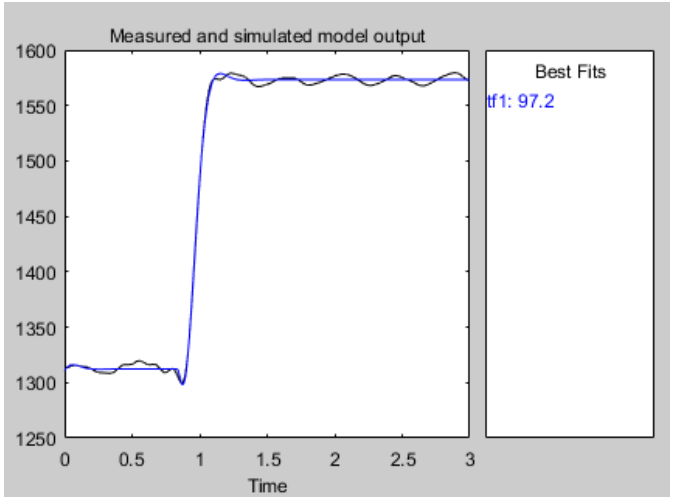


Ilustración 5. 30 Modelo obtenido. Fuente: Autor

Co la ayuda de la herramienta symulink se simula la planta junto con todo el lazo de control además de incluir el controlador proporcional integral en el lazo cerrado de control, que parte de una sintonización automática con la cual se puede obtener una aproximación de las constantes kp y ki.

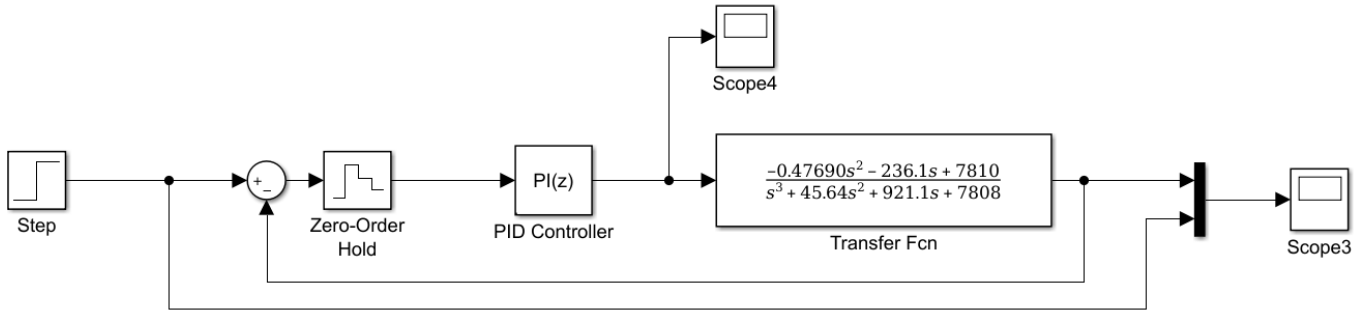


Ilustración 5. 31 Simulación de lazo de control. Fuente: Autor

Al realizar la simulación ilustración 5.31 y sintonización del controlador en tiempo discreto con un tiempo de discretización $72727.2727\mu S$ tiempo que el controlador da una acción de control, obteniendo las constantes:

$$Kp = 0.2965$$
$$Ki = 5.1545$$

Al simular cambio de velocidad ilustración 5.32, junto con un cambio de fallo obtenemos una satisfactoria respuesta en la simulación.

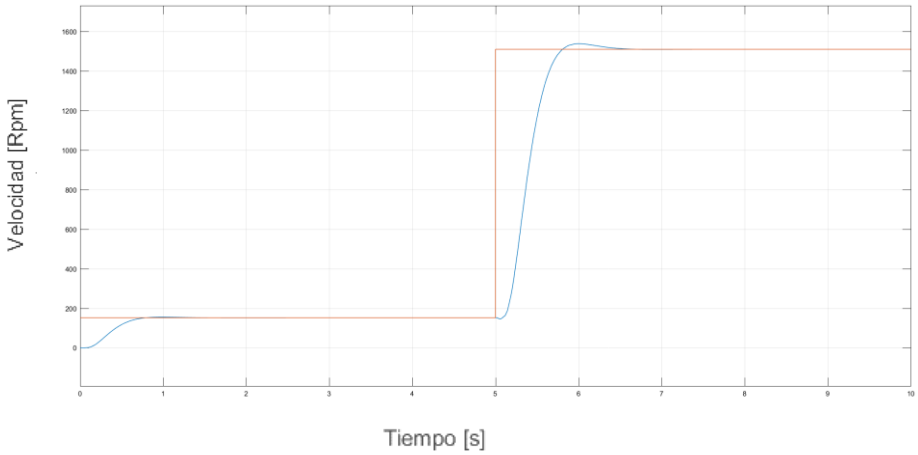


Ilustración 5. 32 Grafica del controlador Fuente: Autor

Vemos que el controlador simulado funciona correctamente actuando de una forma correcta ante posibles perturbaciones y/o cambios de velocidad, de acuerdo a esto se posee una base de partida para poder realizar variaciones en las constantes del controlador implementado en la tarjeta, en la próxima sección se procederá a realizar la prueba a lazo cerrado del sistema de control.

5.2 Funcionamiento de todo el sistema de control a lazo cerrado

En la ilustración 5.33 se puede observar el diagrama en bloques planteado para realizar las pruebas de validación del control de velocidad.

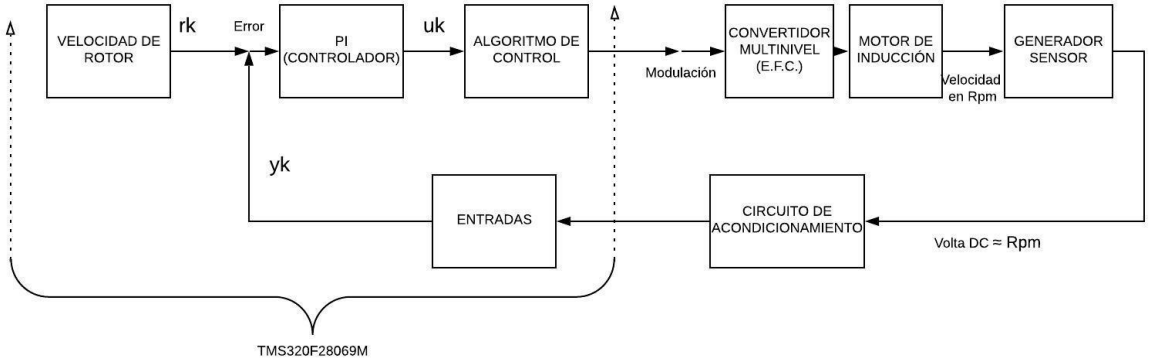


Ilustración 5. 33 Montaje prueba final lazo de control de velocidad Fuente: Autor

En la ilustracion 5.34 se puede apreciar el montaje real efectuado para la realizacion de las pruebas con todas las etapas ya descritas y fundamentadas en capitulos y secciones anteriores donde la interfaz de comunicaciones esta en conexión directa con la tarjeta TMS320F28069 a traver del enlace de conexión provisto por la JTAG de el LAUNCHXL empleado.

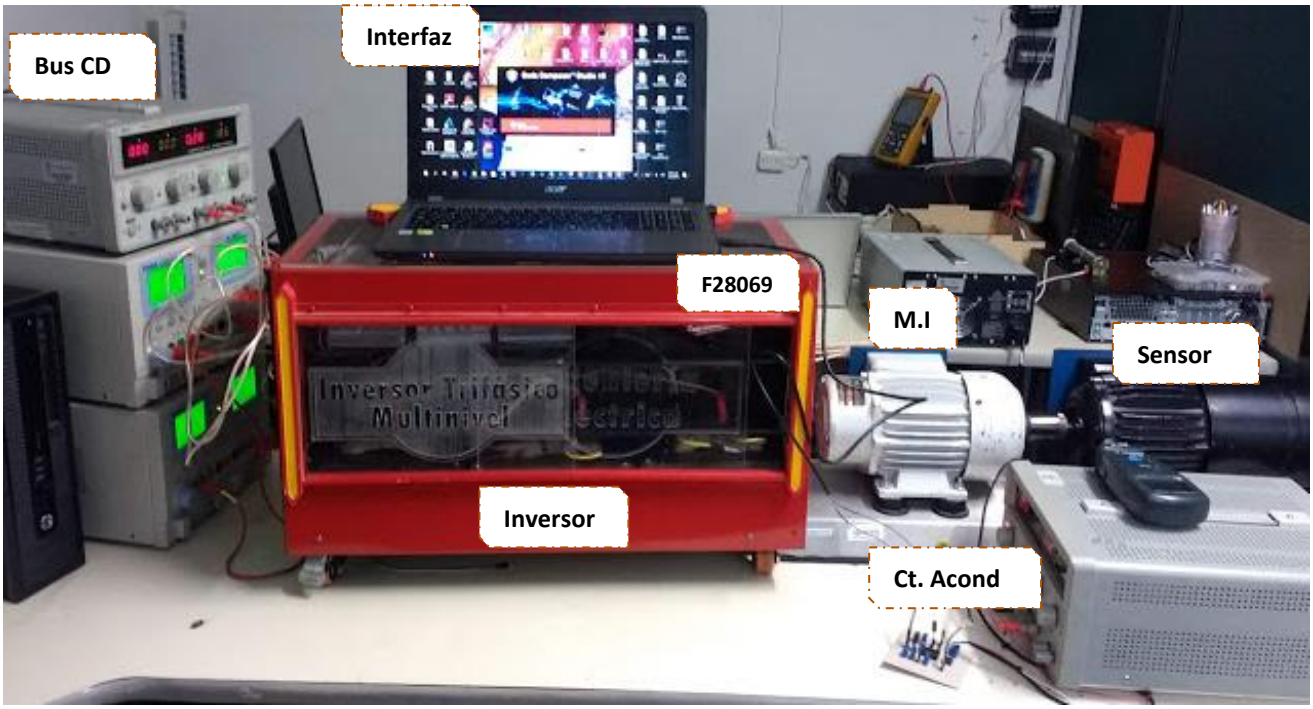


Ilustración 5. 34 Montaje Empleado. Fuente: Autor

5.2.1 Interfaz De monitoreo on line

Para realizar el control y monitoreo del sistema se hizo necesario el monitoreo de las variables importantes en el prototipo desarrollado como la lectura del sensor de velocidad, la respuesta el controlador y el error del lazo de control; también desde la misma interfaz se controla el punto de referencia rk del controlador entre otras variables de las cuales se puede obtener información en tiempo real en la ilustración 5.35 se presenta un prototipo de la interfaz donde se superviso el comportamiento del sistema de control planteado.

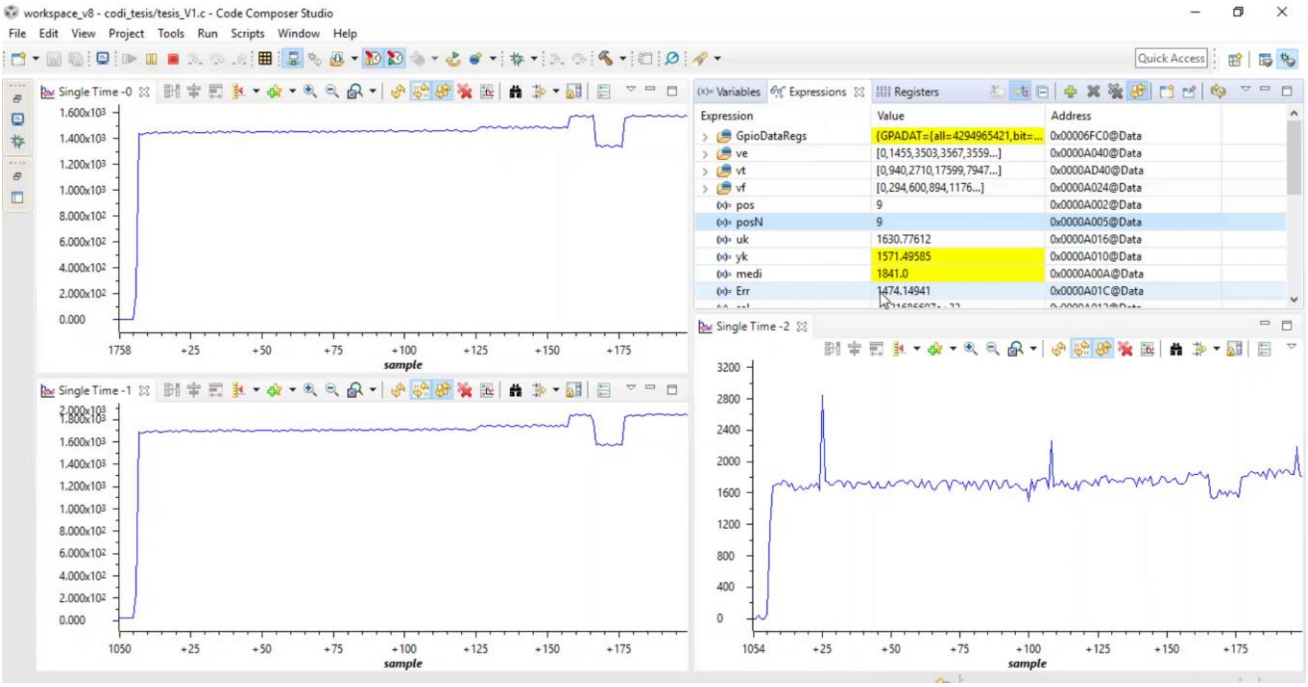


Ilustración 5. 35 Prototipo de la interfaz. Fuente: Autor

5.2.2 Prototipo final del sistema de control

De acuerdo a las constantes Kp y ki encontradas en la sección anterior se toma como referencia para empezar a realizar pruebas a lazo cerrado, con el controlador funcionando, que como es de esperarse las constantes kp y ki encontradas, solo nos dan una base de referencia para empezar a variar las constantes de acuerdo a las respuestas obtenidas.

A través de la interfaz de monitoreo planteada se establece una velocidad objetivo de $1500\ Rpm$ cabe resaltar que esta velocidad objetivo no se encuentra en ninguna modulación cargada en los vectores correspondiente a las modulaciones. En la ilustración 5.36 se puede apreciar una captura de pantalla de la interfaz donde se muestra $\boldsymbol{rk}$ el set point $\boldsymbol{uk}$ respuesta del controlador e $\boldsymbol{yk}$.

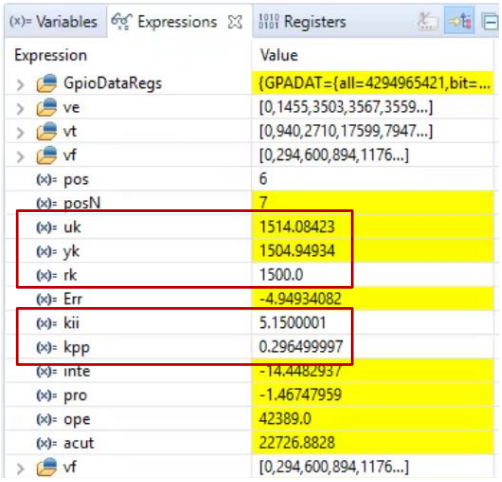


Ilustración 5. 36 Cambio de velocidad de referencia. Fuente: Autor

En esta parte de la interfaz es donde se puede cambiar la velocidad de referencia de la máquina de inducción también se puede supervisar el contenido en instante de tiempo de cada variable planteada para el desarrollo del sistema, en la ilustración 5.37 se observa la acción de control de la máquina de inducción donde el controlador cambia modulaciones con el objetivo de llegar a la velocidad objetivo.

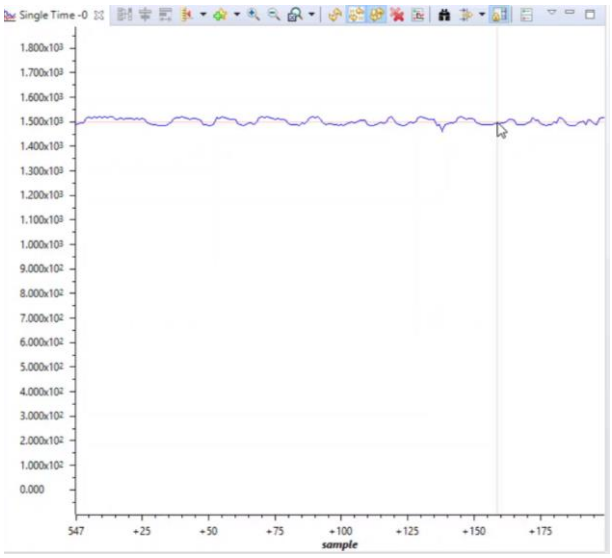


Ilustración 5. 37 Monitoreo de velocidad del motor. Fuente: Autor

Para validar que la maquina se encuentre en una velocidad de $1500\ Rpm$, en la ilustración se puede apreciar la velocidad real de la máquina de inducción medida a través de un tacómetro digital ilustración 5.38.

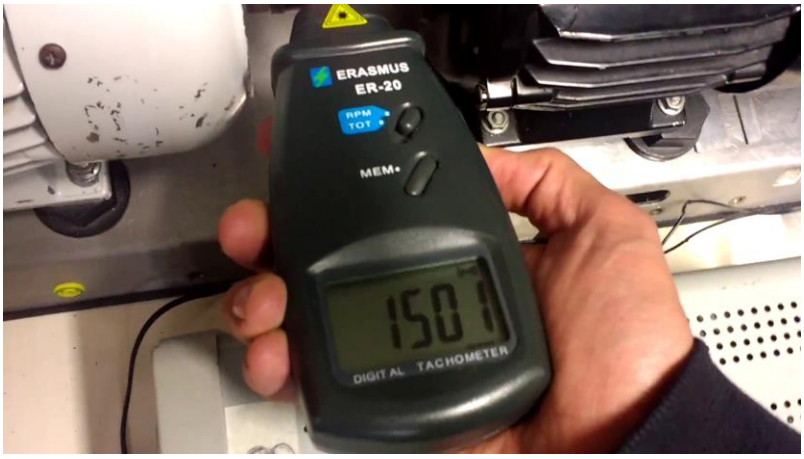


Ilustración 5. 38 Medida del set point Fuente: Autor

En la ilustración 5.39 se cambia el set point de $1500\ Rpm$ a $1470\ Rpm$ con el fin de visualizar el cambio de velocidad de un punto a otro, observando así el controlador en forma directa ante un cambio de velocidad.

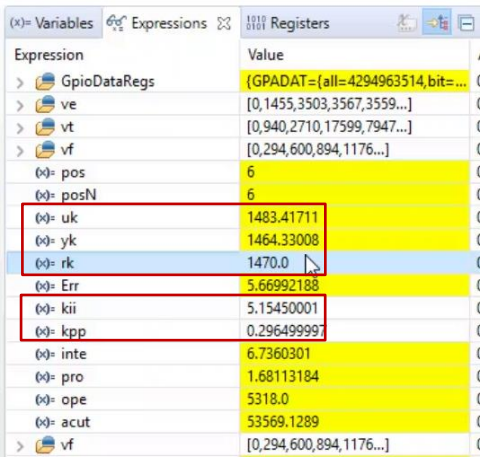


Ilustración 5. 39 Controlador Ante un cambio de velocidad. Fuente: Autor

En la ilustración 5. 40 se observa el cambio de modulaciones que realiza el controlador con el fin de llegar a $1470\ Rpm$. A través de la interfaz propuesta de monitoreo, se visualiza como los diferentes lazos de control cambian como yk y rk permanecen en un punto de relativo de equilibrio corroborando que el sistema de control funciona.

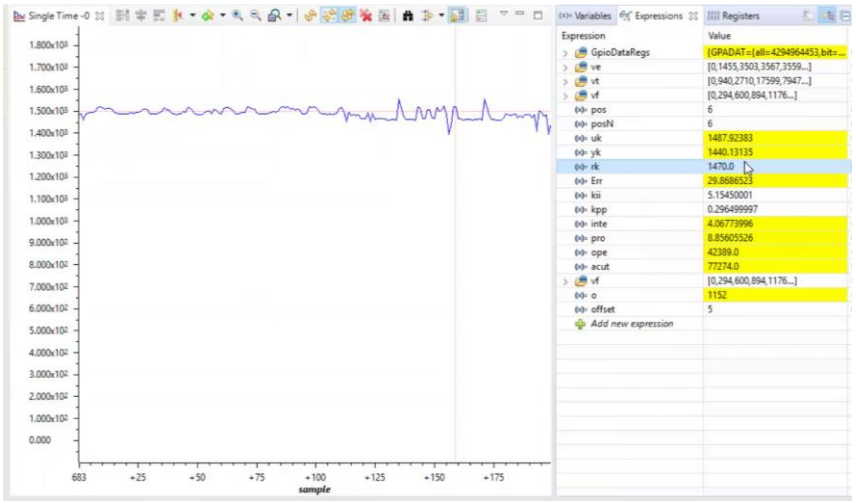


Ilustración 5. 40 Cambio de velocidad del controlador. Fuente: Autor

En la ilustración 5.41 se observa la velocidad real del motor de inducción corroborando así el funcionamiento del sistema de control realizado en la TMS320F28069 con satisfactorios resultados.

Ilustración 5. 41 Velocidad del Motor. Fuente: Autor

CONCLUSIONES

- La familia de microcontroladores y microprocesadores C200 de la Texas Instruments proporciona una amplia gama de microcontroladores para aplicaciones de control, procesamiento de datos, redes neuronales con muy bajo coste y relativa dificultad en el desarrollo el TMS320F28069 usado en este proyecto tiene grandes prestaciones, varias de las cuales se usaron en el desarrollo del proyecto permitiendo el agrupamiento de todos los módulos a un bajo consumo de memoria y alta velocidad en el procesamiento.
- El control escalar es una estrategia de control de las más conocidas para controlar la velocidad de un motor, su aplicación en este trabajo de grado fue crucial para el desarrollo de los objetivos planteados y para cuidar la integridad de todos los componentes involucrados en el desarrollo ya que su aplicación en controlar la velocidad de motores asíncronos está bien fundamentada.
- El acondicionamiento y filtrado de las señales provenientes del sensor, es vital para cualquier desarrollo donde se involucre un sistema de control a lazo cerrado.
- En la práctica hay que tener en cuenta los ruidos que se pueden ocasionar por agentes externos como las fuentes de alimentación, los circuitos de electrónica de potencia y/o transformadores presentes en el inversor multinivel que aunque no tengan contacto directo con la tarjeta inducen un ruido el cual fue necesario considerar en el tratamiento de las señales.
- La buena calidad de la energía de alimentación de un motor de inducción es clave fundamental para aplicar cualquier estrategia de control, esto se logró gracias a la tarjeta de desarrollo LAUNCHXL-F28069M ya que su gran velocidad y forma modular en el diseño permitió un sencillo acople a todas la etapas involucradas con la tarjeta.
- La programación CCS es compleja por lo que requiere tiempo para entenderla, la Texas Instruments nos da todas las herramientas tipo documentación, soporte, entrenamiento, soporte técnico por profesionales para lograr sacar adelante cualquier desarrollo propuesto, todo esto a base del software CODE COMPOSER STUDIO el cual fue la base de todos los algoritmos desarrollados.
- Para realizar desarrollos en el software code composer studio es necesario tener conocimiento sobre arquitectura de microcontroladores, su organización en los distintos bancos de registros, ya que es la principal característica de

este tipo de programación enfocada específicamente a la TMS320F28060 basada en estructuras de campos de bits, que optimizan la memoria y además permite que a cualquier desarrollo en especial al descrito en este proyecto tener un completo manejo en tiempos de ejecución.

- La metodología adoptada para encontrar los vectores de tiempo y estados de las señales de control está desarrollada, fundamentada y descrita en una gran cantidad de artículos científicos especificados en referencias plasmadas los cuales en conjunto con la tarjeta F28069M mejoran la calidad de energía en el control aplicado disminuyendo el %THD además de permitir al controlador desarrollado funcionar de una manera correcta en conjunto con todas las fases del desarrollo.
- El controlador PI planteado y desarrollado en lenguaje CCS fue una elección correcta ya que al tener nada más 2 lazos de respuesta en su diagrama de bloques disminuyo su duración en cuanto a tiempos de ejecución y sobrecarga de la memoria del programa optimizando y adaptándose a las demás etapas propuestas dentro de la tarjeta.
- En las pruebas de validación de funcionamiento se evidenció que al llevar un buen desarrollo metodológico de cada etapa, las medidas realizadas para validación muestran el correcto funcionamiento del código.
- Al realizar las comparaciones tanto de instrumentos industriales como trabajos desarrollados en otros dispositivos embebidos, es de destacar el reducido costo de la implementación planteada obteniendo los mismos resultados o mejores en cuanto a funcionamiento.

REFERENCIAS BIBLIOGRÁFICAS

[1] F. Hernán, M. Sarmiento, J. Jairo, and G. López, “Control escalar en motores de inducción monofásicos,” *Tecnura*, vol. 10, no. 19, pp. 29–37, 2006.

[2] L. D. P. FERNÁNDEZ, “CONTROL DE VELOCIDAD DEL MOTOR DE INDUCCIÓN MEDIANTE CONVERTIDOR DE POTENCIA MULTINIVEL CON OPTIMIZACIÓN DE ARMÓNICOS,” Pamplona, Norte santander, 2016.

[3] A. Fernando, L. Villamizar, J. Luis, and D. Rodríguez, “Minimización de la distorsión armónica de una modulación PWM con algoritmos genéticos,” vol. 8, pp. 79–86, 2011.

[4] J. Villalón and F. Salas Gómez, “Modelado, simulación y control de convertidores en cascada,” Sevilla, 2017.

[5] K. Preethi, G. Anil, and E. Vani, “Speed Control of Induction Motor Using Eleven Levels Multilevel Inverter,” *circuit.International J. Sci. Mod. Eng.*, no. 5, pp. 15–20, 2013.

[6] Texas Instruments Incorporated, “TMS320C28x CPU and Instruction Set Reference Guide,” 2015.

[7] J. Pérez and J. Barrero, “Diseño de un sistema empotrado basado en DSP C2000 de Texas Instruments para control de armario de potencia de 12 fases,” Sevilla, 2015.

[8] B. Larimore, “HVAC Dual-AC Motor Control with Active PFC Implementation Using Piccolo ™ MCUs,” 2010.

[9] Texas Instruments Incorporated, “C2000™ Piccolo™ 32-bit MCU Family,” Dallas, Texas, 2010.

[10] Texas Instruments Incorporated, “TMS320F2806x Piccolo™ Microcontrollers,” Dallas, texas, 2018.

[11] Texas Instruments Incorporated, “Silicon Errata TMS320F2806x Piccolo™ MCUs,” Dallas, Texas, 2018.

[12] Texas Instruments Incorporated, “User’s Guide LAUNCHXL-F28069M Overview,” Dallas, Texas, 2019.

[13] Texas Instruments Incorporated, “Code Composer Studio™ v8 Integrated

Development Environment (IDE) for Embedded Processors," Dallas, Texas, 2014.

[14] A. Pardo and J. L. Díaz, *Aplicaciones de los convertidores de frecuencia estrategias pwm en el control*. Colombia: 2004, 2004.

[15] B. Akin and N. Garg, "Scalar (V/f) Control of 3-Phase Induction Motors," no. July, pp. 1–25, 2013.

[16] STMicroelectronics, "WIDE BANDWIDTH DUAL J-FET OPERATIONAL AMPLIFIERS s HIGH INPUT IMPEDANCE J – FET INPUT TL074A - TL074B," *Datasheet*, no. March, pp. 1–10, 2001.

[17] multisim, "Multisim Live Online Circuit Simulator," 2019. [Online]. Available: https://www.multisim.com/. [Accessed: 28-Jul-2019].

[18] Peñaranda and A. E. Caicedo, "Sensorless control of an induction motor with common source multilevel converter," *WSEAS Trans. POWER Syst.*, pp. 1–2, 2018.

[19] M. R. Banaei and E. Salary, "Asymmetric Cascaded Multi-level Inverter: A Solution to Obtain High Number of Voltage Levels," *J Electr Eng Technol*, vol. 8, no. 2, pp. 316–325, 2013.

[20] Texas Instruments Incorporated, "TMS320x2806x Piccolo Technical Reference Manual." Dallas, Texas, 2017.

[21] P. Richard, "PID controller tuning," *Comput. Control Eng. J.*, vol. 10, no. 2, pp. 44–50, 2005.

[22] T. I. Incorporated, "C2000 Digital Control Library," no. November, pp. 1–32, 2017.

[23] Cedesa, "Fluke 123 - Osciloscopio Digital Portátil de 20MHz." [Online]. Available: https://www.cedesa.com.mx/fluke/osciloscopios/digitales/123/. [Accessed: 29-Jul-2019].

[24] ABB, "Convertidores de frecuencia industriales ABB - ACS800, 0,55 a 5600 kW, Catálogo."

ANEXOS.

Anexo 1. Guía para crear proyectos en code composer studio en tarjetas C2000.

Para realizar estos pasos se debe de tener descargado e instalado code composer studio, controlSUITE o C2000Ware las últimas son cuales softwares y herramientas de desarrollo para la familia de microcontroladores C2000 ™.

1. Se Inicie CCS IDE. Cuando abra CCS, se verá la ventana "Workspace Launcher". Escriba C: \ C2000Workspace.

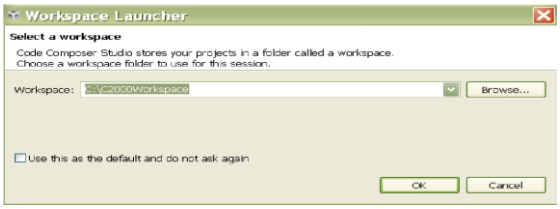


Ilustración 1 CCS

2. Se Cierra la ventana del explorador de recursos y se hace clic en Archivo> Nuevo> Nuevo proyecto CCS para crear un nuevo proyecto.
3. Ingrese el nombre del proyecto y seleccione la configuración del microcontrolador y el programador como se muestra a continuación:

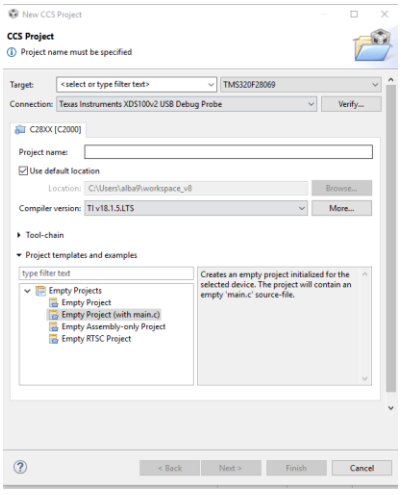


Ilustración 2 Configuración

4. Después de crear el proyecto, haga clic en Ventana> Preferencias para editar las preferencias del proyecto. Para crear la Buil Variable como se muestra a continuación direccionándolo donde se encuentra la carpeta C2000

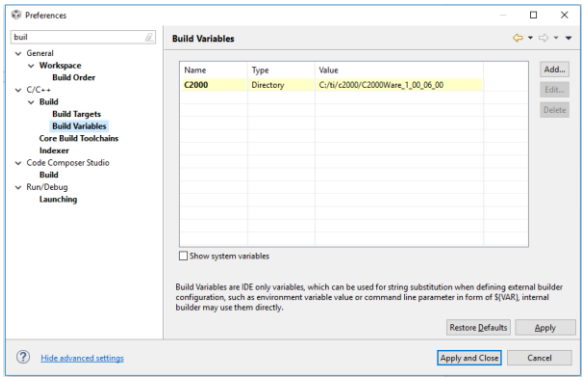


Ilustración 3 Buil Variable

5. Al completar este paso nos dirigimos a la dirección mostrada a continuación dependiendo donde se guardó el C200Ware se arrastran los siguientes archivos hacia el título del archivo ejemplo que se creó anteriormente.

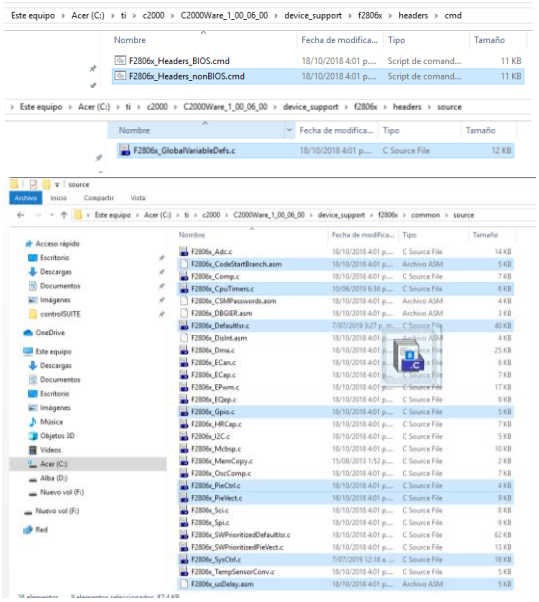


Ilustración 4 Archivos

6. Luego se selecciona la variable de control C200 para direccionar todo de una manera correcta

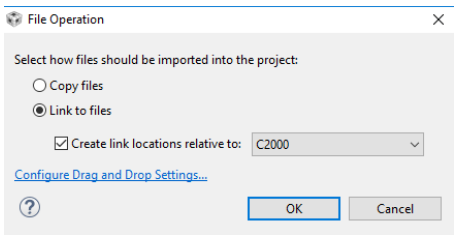


Ilustración 5 Enrutamiento

7. Luego se agregan una librerías de la siguiente manera

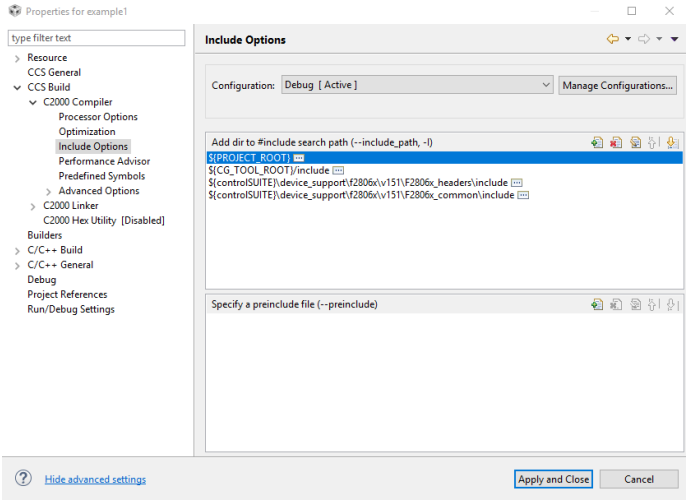


Ilustración 6 Librerías de soporte

8. A continuación se presenta un Código ejemplo donde la parte marcada es el ítem para ejecutar y cargar el código a la tarjeta : encender y apagar el led del GPIO34

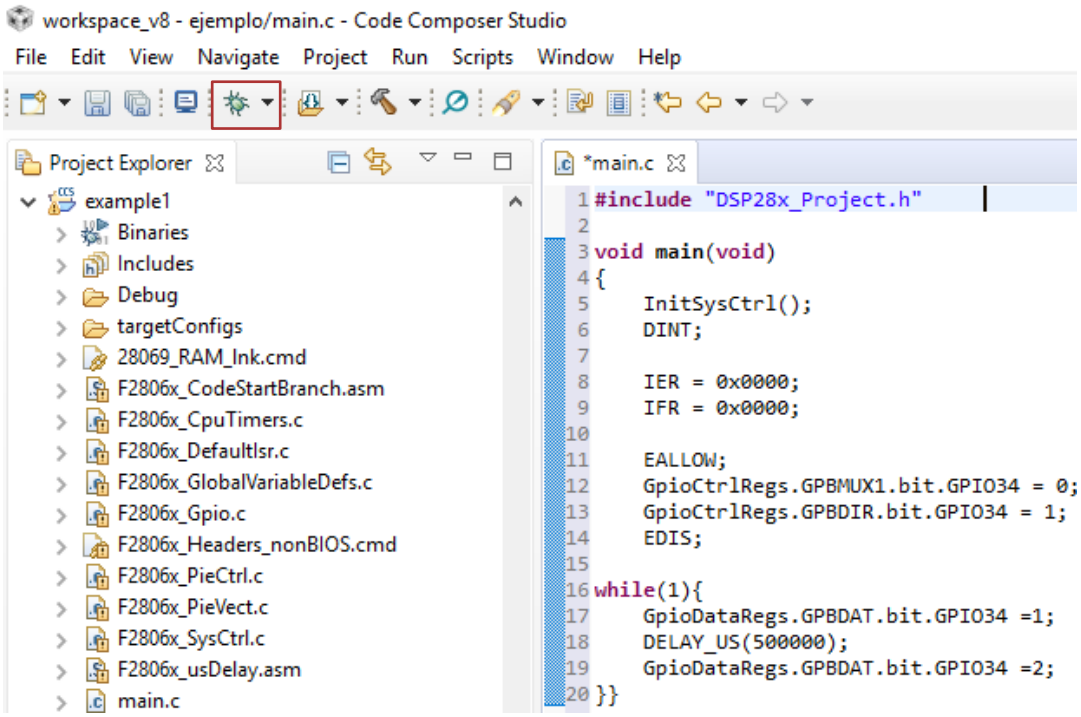


Ilustración 7 ejemplo

Anexo 2. Algoritmo en CCS desarrollado en code composer studio

```
#include "DSP28x_Project.h" //archivo encabezado del dispositivo donde se encuentran funciones ejemplo del dispositivo
#include "string.h"
#include <stdio.h>

__interrupt void cpu_timer0_isr(void);//interrupción externa, inicio de rutina de servicio de interrupción

float uk; // salida controlador
float rk = 1471; // referencia de velocidad en RPM
float E_ant = 0; // Error-1
float yk = 0; // Entrada del sensor en Rpm
float kpp, kii,I_pro,I_inte,Err;// constantes, lazo proporcional, lazo integral, error
float acu_t;//Tiempo de controlador
float ope;//variable de almacenamiento de tiempos
float media;//resultado de la integral proyectiva
unsigned int vt[3283];//vector de tiempos
unsigned int ve[3283];//vector de estados
unsigned long acu_M=0;//acumulador de muestras ADCINT0
int vf[12]={0, 294, 600, 894, 1176, 1446, 1752, 2058, 2364, 2670, 2976, 3282};//vector de finales
int f;//frecuencia reloj 90MHz
int i=0;//variable de control
int pos=5;//modulacion actual
int posN=5;//modulacion objetivo
int j=0;//contador número de muestras capturadas
int offset=1467;//suma a la salida
int AA0;//Muestras ADCINT0

//definiciones por el enlazador F2808.cmd----------------------
extern Uint16 RamfuncsLoadStart;
extern Uint16 RamfuncsLoadEnd;
extern Uint16 RamfuncsRunStart;
extern Uint16 RamfuncsLoadSize;
//-----------------------------------------------------------

void main(void){
```

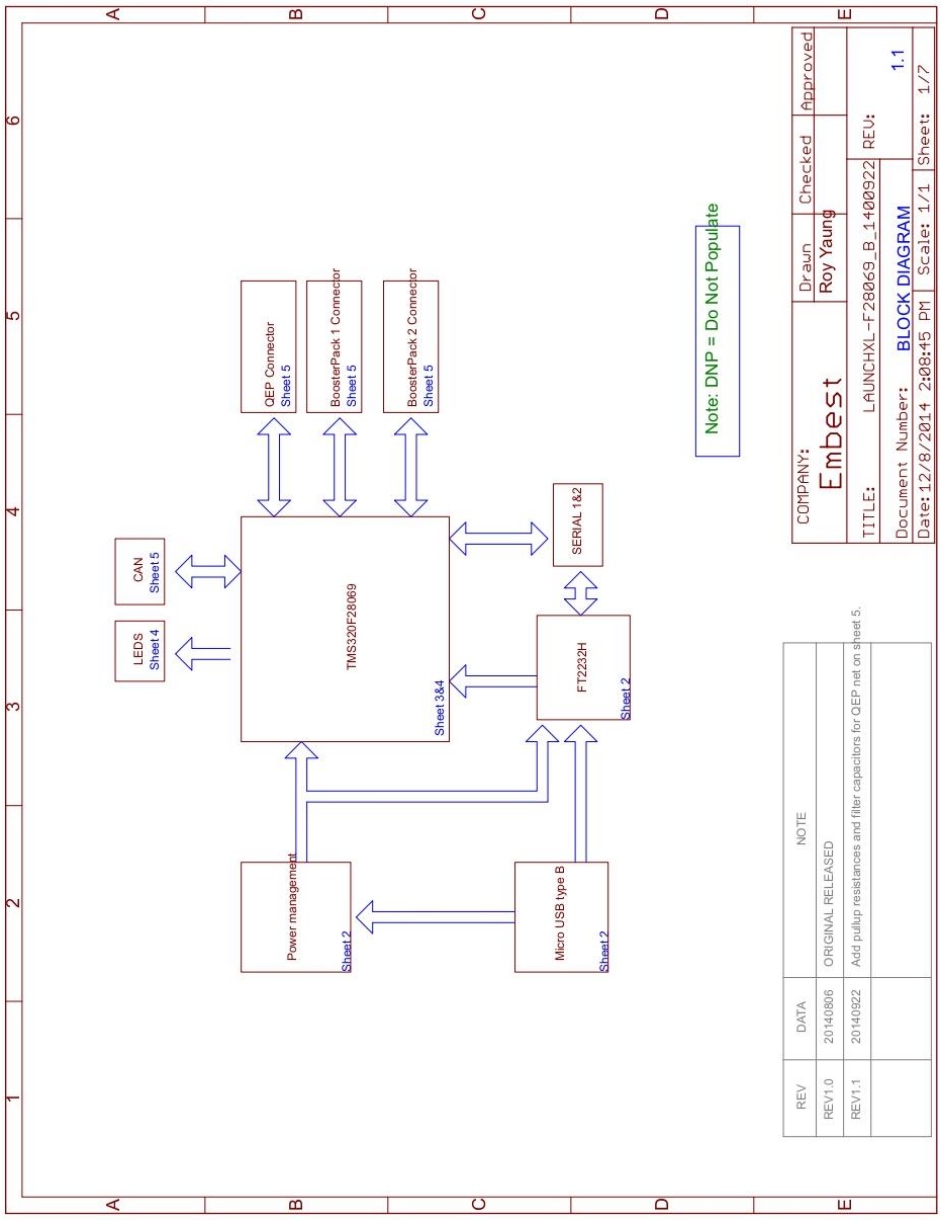
```
    InitSysCtrl();//Inicializar el control del sistema: PLL, WatchDog, habilita los relojes a los periféricos F2806x_SysCtrl.c

    DINT;//Borrar todas las interrupciones e inicializar la tabla de vectores PIE: Desactivar las interrupciones de la CPU

    InitPieCtrl();//Inicializar los registros de control PIE a su estado predeterminado interrupciones deshabilitadas archivo F2806x_PieCtrl.c

    memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (Uint32)&RamfuncsLoadSize); //Copia el código crítico de tiempo y el código de configuración de Flash a la RAM
    InitFlash();//inicialización de flash ser grabada función de la memoria RAM


    IER = 0x0000;// Desactivar las interrupciones de la CPU y borrar todos los indicadores de interrupción de la CPU
    IFR = 0x0000;// Desactivar las interrupciones de la CPU y borrar todos los indicadores de interrupción de la CPU


    InitPieVectTable();// Inicializar la tabla de vectores PIE con punteros al shell Interrupción


    EALLOW;    //habilita escritura sobre registros de protegidos
    PieVectTable.TINT0 = &cpu_timer0_isr;//se le asigna la dirección de función de interrupción a TINT0
    EDIS;     //deshabilita escritura sobre registros de protegidos

    EALLOW; //habilita escritura sobre registros de protegidos
    AdcRegs.ADCCTL1.bit.INTPULSEPOS = 0;    //borde negativo del pulso
    AdcRegs.INTSEL1N2.bit.INT1E = 1;        // activar ADCINT1
    AdcRegs.INTSEL1N2.bit.INT1CONT = 0;     //deshabilitar el modo continuo
    AdcRegs.INTSEL1N2.bit.INT1SEL = 1;      //fuente de interrupción
    AdcRegs.INTSEL1N2.bit.INT2E = 0;        //activar ADCINT2
    AdcRegs.INTSEL1N2.bit.INT2CONT = 0;     //deshabilitar el modo continuo
    AdcRegs.INTSEL1N2.bit.INT2SEL = 0;      //fuente de interrupción
    AdcRegs.ADCSOC0CTL.bit.CHSEL = 0;       // selección del canal ADCINA0
    AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 1;     //sincronizador de captura TIMER0
    AdcRegs.ADCSOC0CTL.bit.ACQPS = 6;       //Ventana de muestreo de 7 ciclos de reloj
```

```
EDIS;//deshabilita escritura sobre registros de protegidos


EALLOW; //habilita escritura sobre registros de protegidos
//Activar pines Como GPIO0-11-------------------------
   GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO7 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO8 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO9 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 0;
   GpioCtrlRegs.GPAMUX1.bit.GPIO11 = 0;
   //Activar GPIO0-11 como salidas-------------------------
   GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO4 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO5 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO6 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO7 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO8 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO10 = 1;
   GpioCtrlRegs.GPADIR.bit.GPIO11 = 1;
   EDIS;//deshabilita escritura sobre registros de protegidos
  kpp = 1.7;//constante proporcional
  kii = 100.0;//constante integral

// Adquisición de los datos desde un archivo .txt-------------------------
FILE* fe;//puntero archivo Ve.txt
FILE* fi;//puntero archivo Vt.txt
fe = fopen("ve.txt","r");//open archivo Ve.txt
fi = fopen("vt.txt","r");//open archivo Vt.txt
for(i=0; i<3283;i++){
fscanf(fe,"%u",&ve[i]);//almacenamiento vector Ve
```

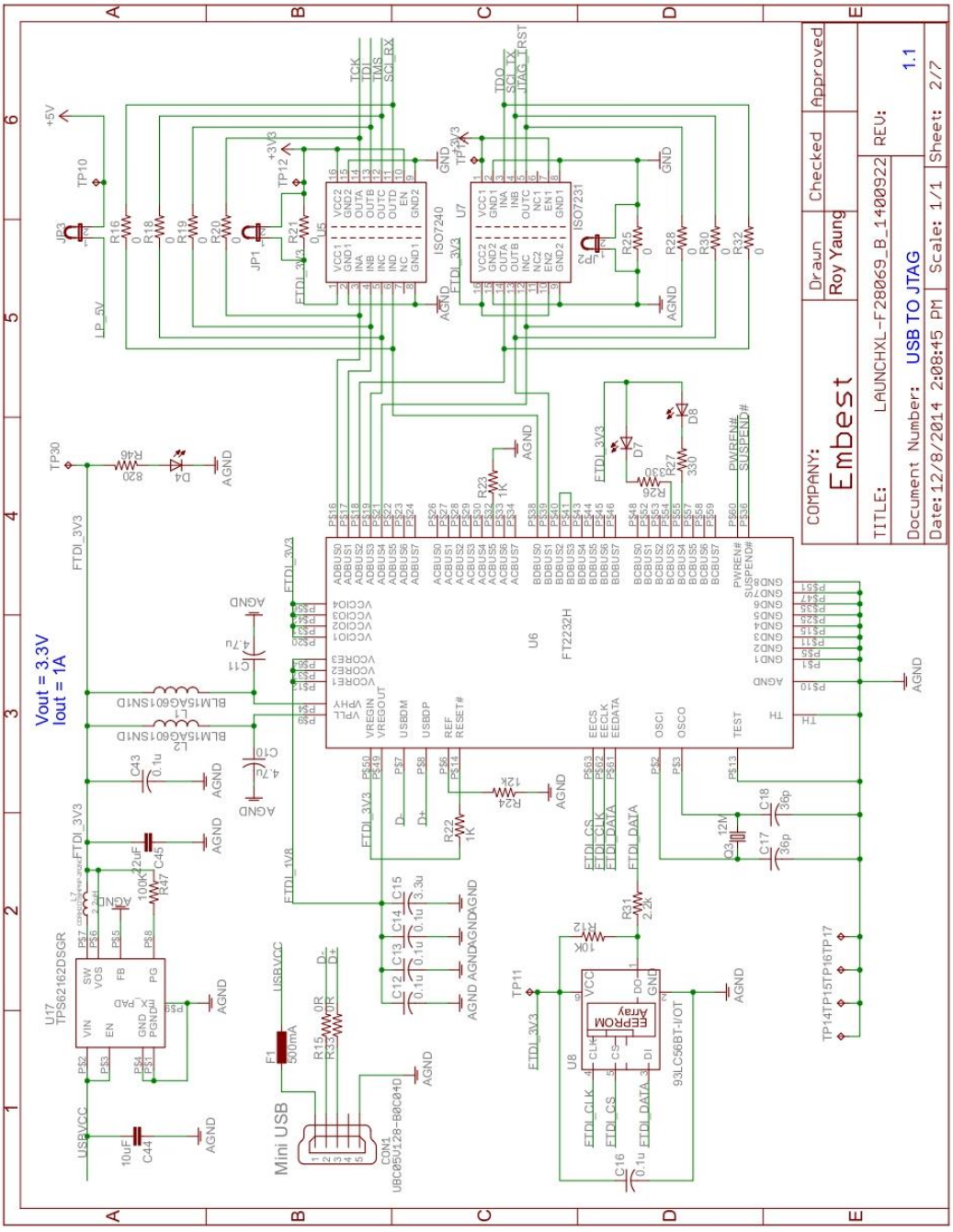
```
    fscanf(fi,"%u",&vt[i]);//almacenamiento vector Vt
      }fclose(fe);
      fclose(fi);
    //---------------------------------------------------------------------------

    i=vf[pos]+1;//asignacion de pocision
    GpioDataRegs.GPADAT.all =ve[i];//Salida Modulación de control
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;//Bloquea demás interrupciones

        //-----------------------------------
    InitCpuTimers();//inicialice los temporizadores de CPU
     ope=vt[i];// Convierte Vt[i] en float
     ConfigCpuTimer(&CpuTimer0, 90,(ope/100));// Frecuencia de CPU de 90MHz,
ope/100  (en uSegundos)

     CpuTimer0Regs.TCR.all = 0x4001;//habilita la interrupción del temporizador

     IER |= M_INT1;// Asigna interrupción a nivel INT1 la interrupción

     PieCtrlRegs.PIEIER1.bit.INTx7 = 1;//Grupo 1 del pie habitado INTx7
individualmente
     EINT;//Habilitar la interrupción global INTM
     ERTM;//Habilitar DBGM de interrupción global en tiempo real
    while(1)
    {// Ciclo infinito

        AA0 = AdcResult.ADCRESULT0;// Captura de datos

    }
}

__interrupt void
cpu_timer0_isr(void)
{
    acu_M=acu_M+AA0;//Acumulación de los datos
    acu_t=(ope/100)+acu_t;//Contador de tiempo
     j++;//Contador de datos
     if(acu_t>=7272.2727){//Condición de Tiempo de controlador

         media=acu_M/j;//Media
         j=0;
         yk = media*0.86424 - 19.57;//transformación a Rpm
```

```
        Err=(rk-yk);//cálculo del error
        l_pro=Err*kpp;//lazo proporcional
        l_inte=((E_ant+Err)*(acu_t/1000000))*kii;//lazo integral
        acu_t=0;
        E_ant=Err;
        uk=l_pro+l_inte+offset;//respuesta del controlador
        posN=(0.034021*uk-44.86)+0.4;//cálculo de modulación objetivo
        if(uk>1615){posN=10;}//saturador
        if(uk<1320){posN=0;}//saturador

        acu_M=0;}
  //Rampa de cambio de posición de frecuencias de acuerdo a restricciones----------
  i=i+1;
    if(i>vf[pos+1]){
      if(posN<pos){
         pos=pos-1;
         i=vf[pos]+1;}
       else{
          i=vf[pos]+1;
          pos=posN;}}
  //-------------------------------------------------------------------------
    GpioDataRegs.GPADAT.all =ve[i];//Salida Modulación de control
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;//Bloquea demás interrupciones

    ope=vt[i];//Convierte Vt[i] en float

    ConfigCpuTimer(&CpuTimer0,90,(ope/100) );// Frecuencia de CPU de 90MHz,
ope/100  (en uSegundos)
    CpuTimer0Regs.TCR.all = 0x4001;//habilita la interrupción del temporizador
}
```

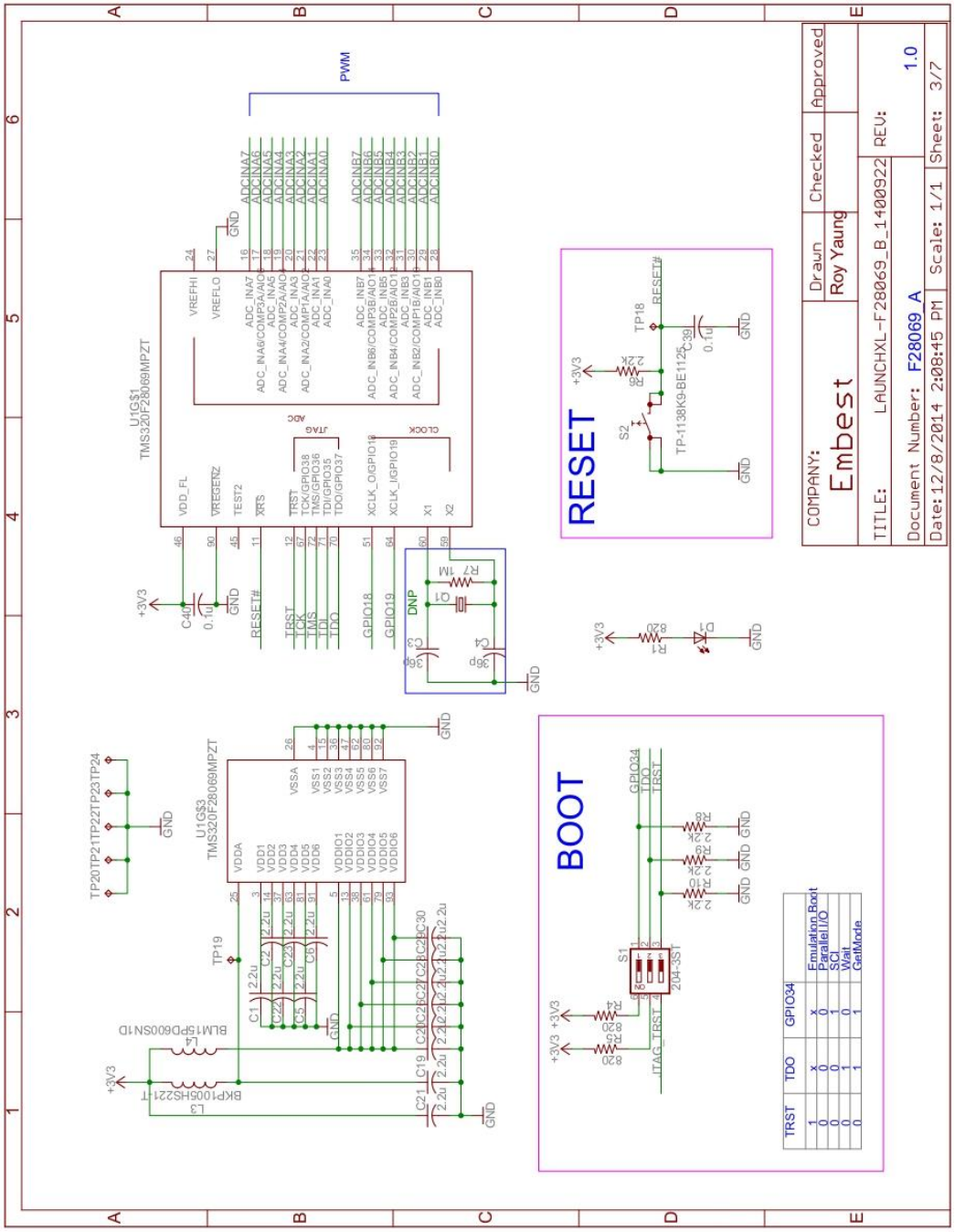
Anexo 3. Diagrama en bloques LAUNCHXL-F28069[12].



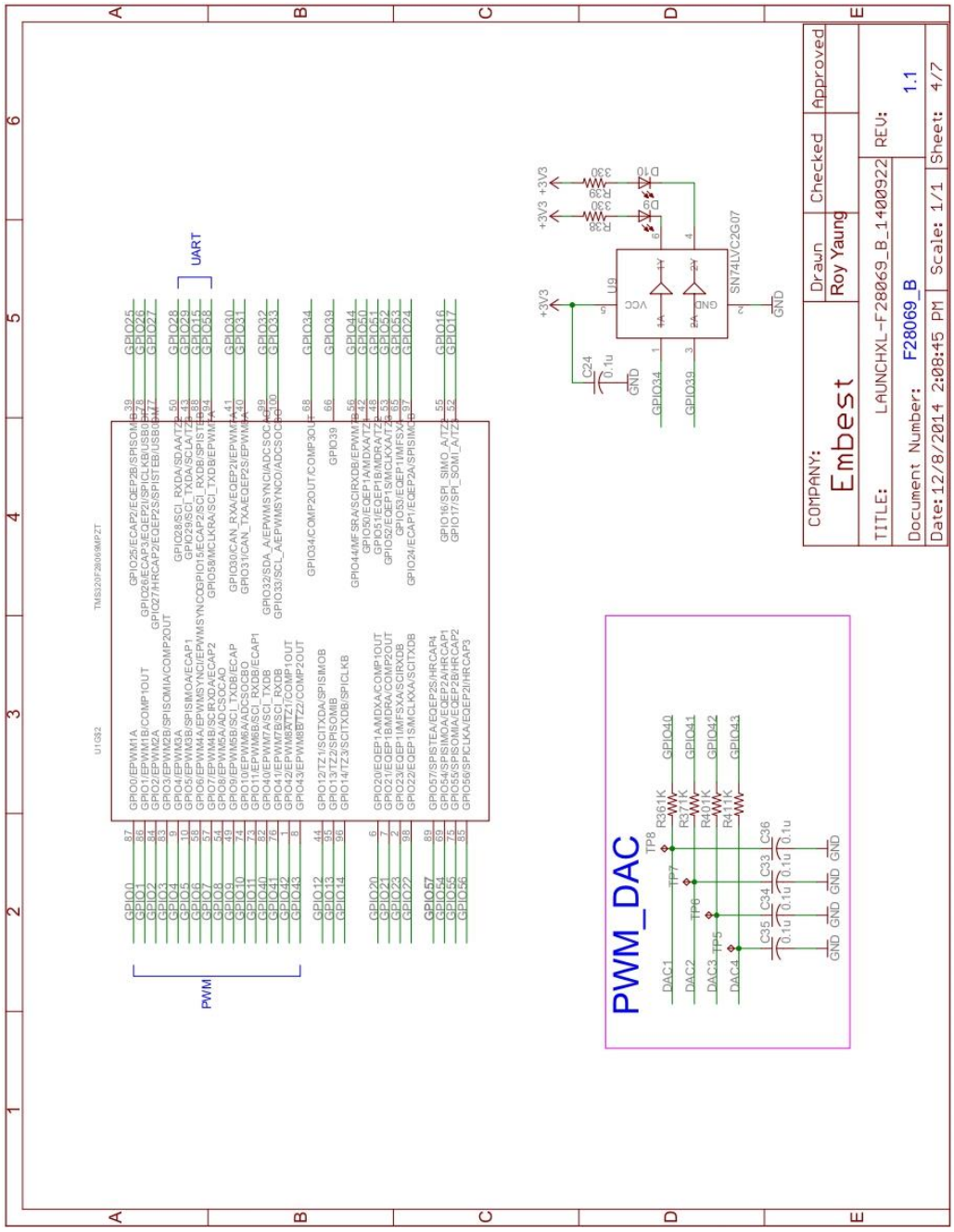
Anexo 4. USB JATAG LAUNCHXL-F28069[12].



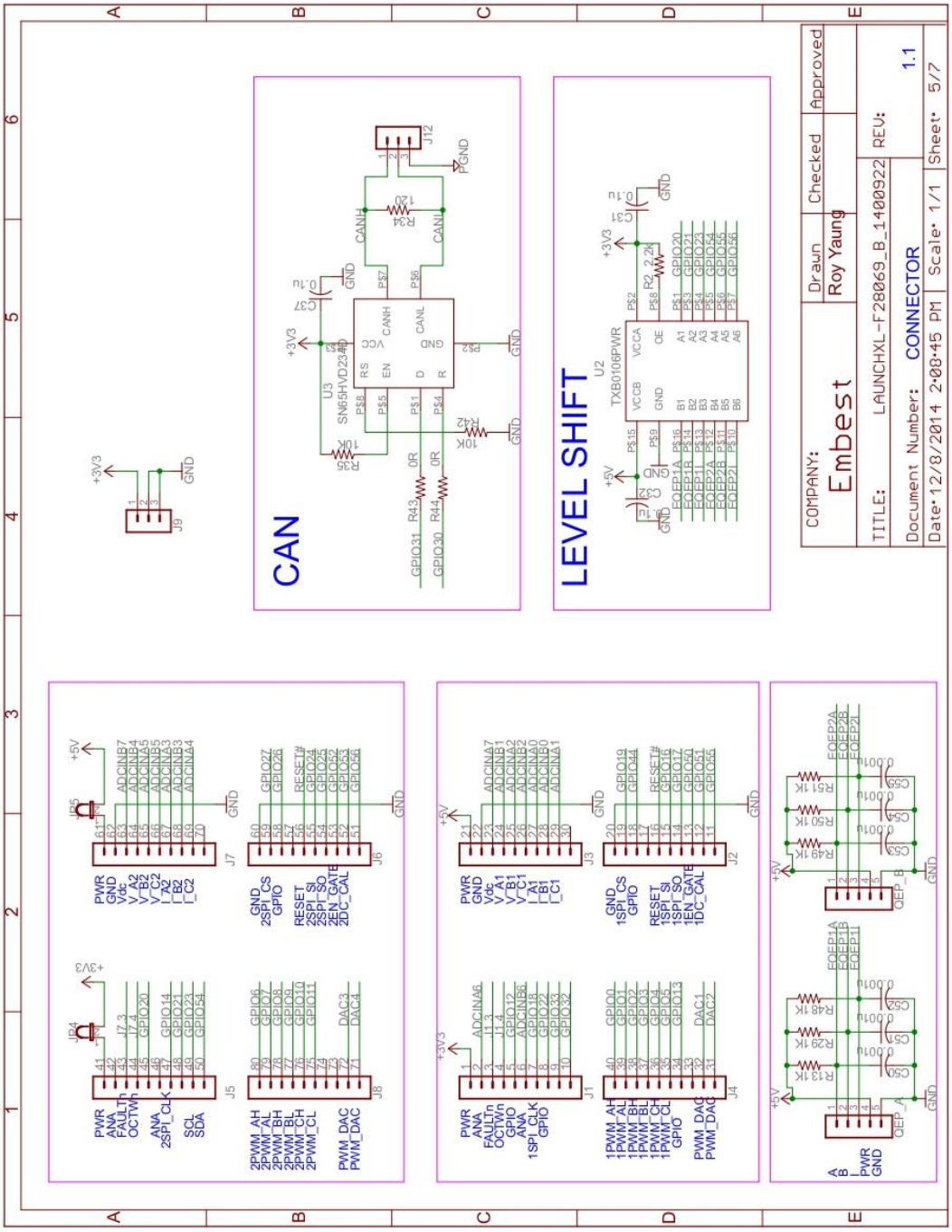
Anexo 5. TMS320F28069M PART_A[12].



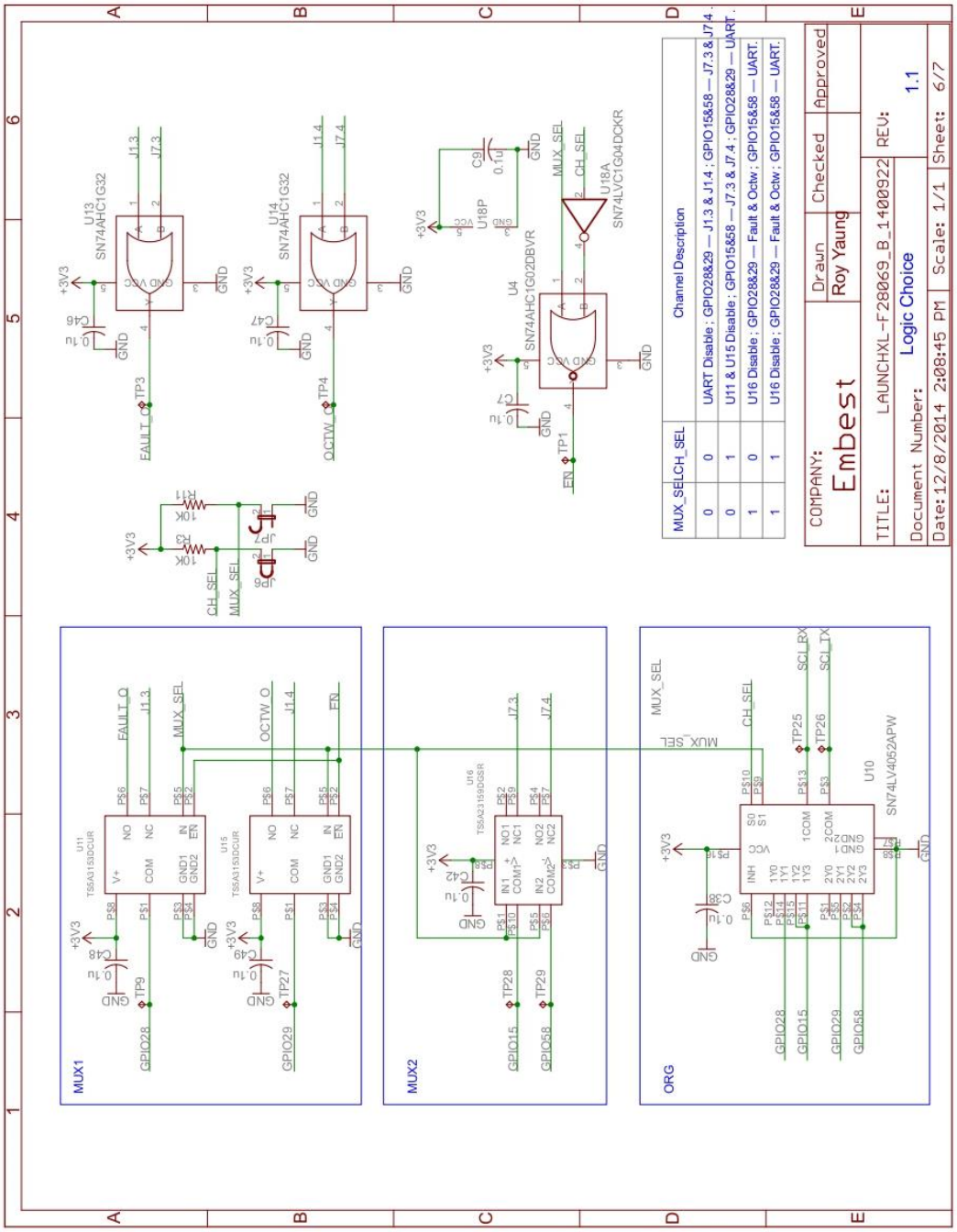
Anexo 6. TMS320F28069M PART_B[12].



Anexo 7. CONECTOR[12].



Anexo 8. LOGICA DE ELECCIÓN[12].



ANEXO 9. PODER[12].