Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

Sistema embebido para hardware in the loop de un generador en esquema motor de CD - Generador Síncrono vía UDP/IP

Tesis para obtener el grado de Licenciatura en

Ingeniería Electrónica

Presenta:

Miguel Angel Pimentel Vallejo

Asesorado por:

Dr. Jaime Cerda Jacobo

Morelia Michoacán a Julio de 2019

Agradecimientos

Al laboratorio de máquinas Eléctricas de posgrado, por el material e instrumentos otorgados durante la elaboración de esta tesis.

A mi familia, novia y amigos por el apoyo económico y moral.

A los maestros de la Facultad de Ingeniería Eléctrica, por los conocimientos que me brindaron, en especial a la Maestra Rosalía, que con sus apoyos extraordinarios logró impulsar en mí la creatividad para realizar cosas nuevas en este trabajo; al maestro Feliche, por todas las enseñanzas, consejos y sobre todo por los momentos divertidos; al maestro Chava, por sus orientación siempre tan certera; al maestro Zamora, por su crítica objetiva y al maestro Cerda, por el apoyo para terminar esta tesis.

Dedicatoria

Dedico mi tesis a las personas más importantes durante este proceso, a mi madre, por obligarme a estudiar cuando no quería, cuánta razón tuvo, también por tu apoyo en todos los sentidos, por el esfuerzo tan grande que hiciste para que esto fuera realidad; a mi tita, que con su cariño, sabias palabras y consejos, me guiaron durante la carrera; a mi hermana, que me aguantó durante todo este tiempo; a mi papá, por su apoyo y ejemplo a seguir; a mi novia que se a convertido en un gran pilar, que me ha motivado, apoyado, acompañado, querido y amado durante este tiempo; a mis amigos Frank, Raúl y Homie por distraerme de todas las ocupaciones de la carrera, los buenos momentos, las risas, el apoyo y la compresión; a mis compañeros, por ayudarme a lo largo de la carrera y a mis profesores, por tenerme paciencia para enseñarme todo lo que ahora se.

Una dedicación especial al Maestro Ávalos, sin usted nada de esto hubiera sido posible, gracias por las enseñanzas y confiar en mí para llevar a cabo este proyecto, formó una gran parte de mi formación cómo ingeniero en la última parte de mi carrera, le recordaré siempre como la gran persona que era y nunca olvidaré lo aprendido de usted.

Resumen

Una simulación en tiempo real se ejecuta en el mismo tiempo que lo hace la planta física, esto para hacer el comportamiento más parecido, este tipo de simulación se realiza en sistemas operativos en tiempo real. Hardware in the loop es una técnica que consiste en un modelo virtual que se está simulando e interactúa con una planta física. Esta técnica sirve para probar de manera más segura y rápida, tanto modelos como controladores. Se realizó un sistema embebido, el cual realiza pruebas en HIL, la planta la cual controla el sistema embebido es un generador conformado por una máquina síncrona acoplado con una máquina de CD. La máquina síncrona da la amplitud del voltaje generado de corriente alterna, mientras que el motor de CD da la frecuencia. El control de las máquinas es con electrónica de potencia. Con esto se puede simular la generación convencional o eólica, con la finalidad de simular microredes.

La programación del sistema embebido es realizada con funciones de un sistema operativo en tiempo real, para garantizar que el tiempo de simulación no se vea afectado.

El sistema embebido mantendrá comunicación con el simulador en tiempo real mediante una conexión de tipo UDP/IP, con esto se mantiene una comunicación rápida y estable, dado que el sistema virtual es ejecutado en el simulador en tiempo real y la planta física es controlada por el sistema embebido, que constantemente envía y recibe información. [1]

[1] Sistemas, Electrónica, Eléctrica, Generación, Alterna

Abstract

A simulation in real time is executed in the same time as the physical plant does, this to make the behavior more similar, this type of simulation is done in operating systems in real time. Hardware in the loop is a technique that consists of a virtual model that is simulating and interacting with a physical plant. This technique is used to test more safely and quickly, both models and controllers. An embedded system was made, which performs tests in HIL, the plant which controls the embedded system is a generator formed by a synchronous machine coupled with a CD machine. The synchronous machine gives the amplitude of the generated voltage of alternating current, while the CD motor gives the frequency. The control of the machines is with power electronics. With this you can simulate conventional or wind generation, in order to simulate micro networks.

The programming of the embedded system is performed with functions of an operating system in real time, to ensure that the simulation time is not affected.

The embedded system will maintain communication with the simulator in real time through a UDP / IP type connection, with this a fast and stable communication is maintained, since the virtual system is executed in the simulator in real time and the physical plant is controlled by the embedded system, which constantly sends and receives information.

Contenido

Agradecimientos I
Dedicatoria II
Resumen III
Abstract IV
Contenido IV
Lista de Figuras IX
Símbolos y Abreviaciones XI

1. Introducción 1
1.1. Antecedentes 1
1.2. Introducción 1
1.3. Objetivo 2
1.4. Justificación 2
1.5. Metodología 3
1.6. Contenido de los capítulos 4

2. Propulsores de CD 5
2.1. Introducción 5
2.2. Tipos de Propulsores 6
2.2.1. Propulsores Monofásicos 6
2.2.2. Propulsores Trifásicos 6
2.2.3. Propulsores de Pulsador 7
2.2.4. Control de aceleración 7

3. Sistemas Operativos en Tiempo Real 9
3.1. Introducción 9
3.2. Interrupciones 10
3.3. Gestión del tiempo 10
3.4. Tareas 10
3.5. Planificador 11
3.6. Semáforos 11
3.7. Colas 11
3.8. FreeRTOS 12

4. Simulación en tiempo real 13
4.1. Introducción . . . 13
4.2. Hardware in the Loop . . . 13
4.3. Rapid Control Prototyping . . . 14
4.4. OPAL-RT 5600 . . . 14
4.5. Módulo OP8660 HIL Controller . . . 15

5. Microcontrolador ESP32 17
5.1. Introducción . . . 17
5.2. Modulación por Ancho de Pulso . . . 17
5.3. Convertidor Analógico a Digital . . . 18
5.4. Módulo WI-Fi . . . 18

6. Comunicación 19
6.1. Introducción . . . 19
6.2. Protocolo IP . . . 19
6.3. Protocolo TCP . . . 19
6.4. Protocolo UDP . . . 20
6.5. Tecnología Websocket . . . 20

7. Desarrollo 21
7.1. Esquema motor de CD - Generador Síncrono . . . 21
7.2. Diseño de Propulsor . . . 22
7.3. Retroalimentación . . . 25
7.4. Programación Controlador Digital . . . 26
7.4.1. Programación del Controlador . . . 26
7.4.2. Comunicación . . . 28
7.5. Pruebas del Propulsor Controlado . . . 29
7.6. Comunicación con el simulador en Tiempo Real . . . 33
7.7. Protecciones de los IGBTs . . . 35
7.8. Sistema operativo en tiempo real del sistema embebido . . . 36
7.8.1. Tareas del RTOS . . . 36
7.8.2. Interrupción de comunicación . . . 37
7.8.3. Colas para el envió de información entre tareas . . . 39
7.8.4. Semáforo para la sincronización . . . 39
7.9. Diagrama del RTOS . . . 40
7.9.1. Prueba del RTOS . . . 41
7.10. Diseño del controlador de voltaje generado . . . 42
7.10.1. Topología del control . . . 42
7.10.2. Medición de voltaje generado . . . 42
7.10.3. Prueba del controlador de voltaje generado . . . 43
7.11. Construcción del sistema embebido . . . 46
7.12. Cambio de topología . . . 46
7.12.1. Cambios en el sensor de voltaje generado . . . 47
7.12.2. Cambios en la alimentación . . . 48

7.13. Pruebas finales 48
7.14. Conclusiones 51

8. Conclusiones 53
8.1. Conclusiones generales 53
8.2. Trabajo a futuro 55

A. Programas utilizados 57
A.A. Programa Control Digital 57
A.B. Programa Control con RTOS para Hardware in the loop 60
A.C. Programa Control con RTOS para Rapid Control 65

B. Hojas de datos 71
B.A. IGBT SGTW90V60F 72
B.B. Optoacoplador 4N25 73
B.C. L293d 74
B.D. Diodo VS-ETH3007-M3 75
B.E. LM324 76

Referencias 77

Lista de Figuras

2.1. Pulsador de aceleración 8

4.1. Esquema de un HIL 13
4.2. Esquema de un RCP 14
4.3. Diagrama de bloques del OP5600 15
4.4. Simulador OPAL-RT OP5600 15
4.5. Módulo OP8660 16

7.1. Esquema motor de CD maquina síncrona 21
7.2. Esquema de generación 22
7.3. Diagrama del primer propulsor 22
7.4. Diagrama optoacoplador 23
7.5. Diagramas de los cambios en la señal PWM 24
7.6. Esquema de sensor de velocidad 25
7.7. Diagrama de flujo 27
7.8. Modelo de comunicación de prueba del propulsor controlado 28
7.9. Primer Modelo para la prueba del Propulsor Controlado 29
7.10. Diagrama a bloques de las conexiones de prueba 30
7.11. Primera prueba del propulsor controlado 31
7.12. Prueba de arranque del propulsor con un control proporcional 31
7.13. Prueba de arranque del propulsor con un control PI 32
7.14. Segundo modelo para la prueba del propulsor 32
7.15. Prueba simulando la velocidad de un generador 33
7.16. Diagrama del proceso de comunicación vía UDP/IP 33
7.17. Esquema de protección electrónica 35
7.18. Histograma de tiempo del algoritmo de control 37
7.19. Histograma del tiempo de la comunicación 37
7.20. Gráfica del tiempo que tarda el dato en llegar por muestra 38
7.21. Diagrama de espera de la cola 39
7.22. Diagrama del RTOS 40
7.23. Histograma del tiempo de simulación del RTOS 41
7.24. Esquema de control de voltaje generado 42
7.25. Esquema de medidor de voltaje generado 43

7.26. Bloque maestro de la prueba del control de voltaje generado 44
7.27. Bloque de consola de la prueba del control de voltaje generado 44
7.28. Prueba de arranque del control 45
7.29. Prueba del control con diferentes referencias 45
7.30. PCB 46
7.31. Nueva topología de potencia 47
7.32. Modificación al sensor de voltaje 47
7.33. Bloque maestro de la prueba final 48
7.34. Bloque de consola de la prueba final 49
7.35. Prueba final 49
7.36. Arranque de la prueba final 50
7.37. Cambio de frecuencia 51
7.38. Apagado del generador 52

Lista de Acrónimos

Símbolos	Significado
CD	Corriente Directa
CA	Corriente Alterna
RTOS	Sistema Operativo en Tiempo Real
PWM	Modulación por ancho de pulso
IP	Protocolo de Internet
TCP	Protocolo de control de transmisión
UDP	Protocolo de datagrama de usuario
API	Interfaz de programación de aplicaciones
SPI	Interfaz periférica serial
I2C	Circuito inter-integrado
PCB	Tarjeta de circuito impreso

Capítulo 1

Introducción

1.1. Antecedentes

Las investigaciones que se consultaron son de energía renovable, control, protecciones eléctricas y microrredes, con el uso de HIL. En estas se usan los módulos del fabricante, que son para uso general, con algunas adaptaciones.

En de los documentos de investigación se conecto el módulo de HIL a un protección eléctrica, se simulaba una red eléctrica la cual tendría un fallo, esto provoca que la protección la cual fisica se activará.[osc18]

Otro de documentos consultados habla acerca del control de un inversor, destaca el hecho de que para este solo se usaron los módulos producidos por el fabricante como se acostumbra en este tipo de simulaciones.[OPAL-RT].

Todos los documentos aunque usan módulos generales logran su objetivo, un nuevo aspecto a investigar es construir los propios módulos para hacer las investigaciones con HIL.

1.2. Introducción

Hoy en día, cada vez más se modelan sistemas complejos, para la comprensión de cómo es que estos se comportan, es por eso que ha surgido una necesidad enorme hacia técnicas de simulación más parecidas a la realidad. En este marco es cuando surge como una alternativa, la simulación en tiempo real, esta no solo reduce el tiempo de elaboración y validación de los modelos, si no que hace mucho más barata la implementación de estos, con un correcto uso de la simulación en tiempo real es posible tener cualquier tipo de planta

o controlador virtual.

Unas de las técnicas que surgen para complementar la simulación en tiempo real, es la de Hardware In the Loop o "HIL", esta consiste en que un sistema simulado en tiempo real, interactúa con un sistema físico. Para que esto sea posible debe haber un acoplamiento de señales entre lo virtual y lo real.

Si lo que se simula es el control y la planta es real, se le suele llamar Rapid Control Prototyping o RCP", esta técnica sirve para validar controladores.

Son muchas las ventajas de esta forma de simular, pero aún hay mucho desarrollo por hacer para que este campo siga creciendo, debido a que hoy las plataformas para realizar esto son bastantes caras y muy pocas.

1.3. Objetivo

Realización de un sistema embebido capaz de controlar en tiempo real la amplitud y frecuencia del voltaje generado de una máquina síncrona acoplada con una máquina de CD, que mantiene una comunicación con el simulador en tiempo real por medio de UDP/IP.

La finalidad es que este módulo es para pruebas en Hardware in the Loop, donde se desee probar sistemas relacionados son la generación como lo son microrredes, generacion de energia renovable o convencional.

Con esto se comprueba que se pueden diseñar sistemas embebidos, para cada una de las necesidades que se tengan en el laboratorio. El hecho de que la comunicación por UDP/IP es adecuada para este tipo de aplicaciones.

1.4. Justificación

La realización de este sistema embebido, es motivada por la necesidad de contar con herramientas que complementen, el uso del simulador en tiempo real con el que cuenta la universidad, debido a que este es muy versátil para hacer pruebas de sistemas modelados. Este sistema embebido valida el hecho de que se pueden fabricar los módulos externos según la aplicación que se quiera realizar. Una de estas necesidades es la de simular sistemas de energía aislados, con una mezcla de generación convencional y energías renovables, es por eso que se seleccionó un generador, como planta de este sistema embebido.

1.5. Metodología

Se planteó el problema de cómo se debe de controlar máquinas de grandes potencias, ya que de esto no se tenía referencia práctica alguna. Analizando cuál sería la topología más conveniente a utilizar en libros e internet, se seleccionó la más conveniente.
Se procedió a buscar los componentes necesarios para la topología seleccionada, éstos deben de ser adecuados para el manejo de la potencia nominal de las máquinas.
Realizando distintas pruebas para saber si el diseño es correcto. Los fallos presentados, se corrigieron con la adición de nuevos elementos o cambios en la topología.

Para completar el control, fue necesario añadir etapas de medición tanto de voltaje como de velocidad, esto se logró con diversos sensores y etapas de instrumentación, que se diseñaron. Esto fueron basados en lo visto durante toda la carrera.

Investigar la manera en que se realizaría el control en el microcontrolador. Se consultaron maestros, libros e internet, sobre el tema, la información estudiada, ayudó a conformar un código para el control, una vez fue terminado su funcionamiento era correcto, con esto la etapa de potencia y control serían terminadas.

Muchos componentes de potencia, se dañan durante las pruebas por los picos de corriente de los devanados, debido a esto se diseñaron una protecciones electrónicas. Estas fueron diseñadas con el conocimiento que se tenía de electrónica analógica, su funcionamiento se probaría con una simulación.

De una variedad de protocolos de comunicación con los que cuenta el simulador en tiempo real, se investigó el que fuera el más adecuado para la necesidades del módulo, el cual es UDP/IP. Se realizaron las configuraciones adecuadas para la correcta transferencia de información, lo cual se validó con distintas pruebas.

El uso de un sistema en tiempo real, para administrar de manera más eficaz la forma en que se utilizan los recursos del sistema embebido, así como para la sincronización del simulador en tiempo real con el módulo.

Finalmente se integró todo lo realizado para validar el correcto funcionamiento de todo el módulo.

1.6. Contenido de los capítulos

Capítulo 1 Trata acerca de lo que se debe considerar al iniciar a leer este documento, contiene las motivaciones del trabajo, la forma en que se desarrolló, lo que aspectos importantes y lo que se ha hecho antes de este documento.

Capítulo 2 Se explican las diferentes opciones con las cuales se puede controlar la velocidad de un motor de CD, sus topologías, como se clasifican y ventajas de cada una de ellas.

Capítulo 3 Es la descripción de todas las partes que conforman un sistema operativo en tiempo real, cada uno de los aspectos en los que influyen, sus características y se da a conocer la herramienta con la cual se desarrolla el sistema operativo del sistema embebido.

Capítulo 4 Se habla acerca de la simulación en tiempo, de diversas formas en las que estas se puede usar, las características para que se lleve de manera adecuada y los modelos de cada una de la técnicas que se utilizan en este documento.

Capítulo 5 Se describe el microcontrolador a utilizar, sus características, funcionalidad y aplicaciones, acompañada de las definiciones generales de los sistemas con los que generalmente cuentan los microcontroladores.

Capítulo 6 Es acerca de los principales protocolos de Internet, IP como capa de Internet, UDP o TCP como capas de transporte, como es que estos funcionan y sus características.

Capítulo 7 Se describe el proceso de desarrollo e investigación del sistema embebido, se dan detalles de los errores y las soluciones seleccionadas que surgen del desarrollo del módulo.

Capítulo 8 En este se exponen todas las conclusiones a las que se llegó después de terminar el sistema embebido y el trabajo futuro que se realizará.

Capítulo 2

Propulsores de CD

2.1. Introducción

Las máquinas de CD son ampliamente usadas en diversos ámbitos, esto debido a que su control de velocidad se puede hacer en un gran rango, también posee un alto par de arranque. Los propulsores para control de velocidad de máquinas de CD son generalmente más fáciles, simples, rápidos y baratos de hacer que los de máquinas de CA, también el análisis es sencillo por eso es que el uso de estos se ha esparcido. Usando las configuraciones de excitación en serie, usada para aplicaciones de tracción o excitación independiente, esta excitación es la que se usa normalmente en controles de velocidad.

Aunque estos no son muy recomendados para aplicaciones de muy alta velocidad. Otra desventaja es que las máquinas de CD ocupan mucho más mantenimiento que una de CA.

Con los grandes avances tecnológicos en microcontroladores, técnicas de control, conmutadores y convertidor de energía, las máquinas de CA cada vez se vuelven más competitivas en este ámbito, no es de sorprenderse que por estas razones el futuro parece apuntar a los propulsores de máquinas de CA, aunque en el panorama actual los propulsores de CD son los que son más usados en la industria. Pero en un par de años podríamos estar ante la inminente sustitución de todas las máquinas de CD por las nuevas máquinas CA con sofisticados controles de velocidad.[Rashid95]

2.2. Tipos de Propulsores

Se pueden clasificar en tres grandes ramas los propulsores, las cuáles serían monofásicos, trifásicos y de pulsador.

2.2.1. Propulsores Monofásicos

Estos se consiguen conectando el circuito de armadura a un rectificador monofásico controlado, con el cual se puede variar el voltaje de armadura, variando el ángulo de retraso del convertidor. Estos rectificadores de conmutación forzada, también pueden ser usados para mejorar los armónicos y el factor de potencia de los rectificadores convencionales.

Puede que la eficiencia del motor al usar esta técnica baje, esto debido a que la corriente puede verse interrumpida debido al ángulo de retraso del convertidor, por lo que se recomienda la conexión de un inductor suavizador en serie con la armadura, esto con la finalidad de filtrar la corriente.

No solo debe controlar el voltaje en la armadura, si no también el de campo, para tener un control más amplio. También si se quiere controlar el sentido del giro se pueden agregar contactores en los devanados armadura o campo para cambiar la polaridad.[Rashid95]

Según el rectificador monofásico que se elija y la cantidad que se conecten se puede clasificar esto propulsores en:

- Propulsores de convertidor de media onda monofásicos
- Propulsores de semiconvertidor monofásico
- Propulsores de convertidor completo monofásico
- Propulsores de convertidor dual monofásico

2.2.2. Propulsores Trifásicos

La armadura del motor de CD se conecta a la salida de un rectificador trifásico de conmutación forzada, estos son comúnmente usados en aplicaciones de grandes potencias, en megawatts. Debido a que este es un rectificador trifásico la frecuencias de voltaje de salida son mayores a la de uno monofásico, esto hace que la inductancia de suavizado sea menor en magnitud, debido a que la corriente de armadura se ve interrumpida por menor

tiempo que en el propulsor monofásico, por esta misma razón la eficiencia de la máquina se incrementa.[Rashid95]

Al igual que los propulsores monofásico se dividen en:

- Propulsores de convertidor trifásico de media onda
- Propulsores de semiconvertidor trifásico
- Propulsores de convertidor completo trifásico
- Propulsores de convertidor dual trifásico

2.2.3. Propulsores de Pulsador

Este tipo de propulsor es el más usado en aplicaciones de tracción, este es conectado entre una fuente de CD con voltaje fijo y la armadura del motor de CD, con el fin de tener control sobre el voltaje en el devanado de armadura.

Con este propulsor es posible usar el freno regenerativo para devolver energía a la fuente de energía, para que funcione es necesario que la fuente de energía sea receptiva de lo contrario no funcionara. Esto suele ser muy atractivo para algunos áreas como la de los autos eléctricos.

Los motores de CD pueden ser utilizados en cuatro cuadrantes, controlando su voltaje, corriente de armadura o campo, para esto también es necesario poder invertir el voltaje en las terminales.[Rashid95]

Los posibles modos de control de un propulsor pulsador de CD son:

- Control de potencia o de aceleración
- Control de freno regenerativo
- Control de freno reostático
- Control combinado de freno regenerativo y reostático

2.2.4. Control de aceleración

El pulsador controla el voltaje que entra a la armadura de un motor de CD, este es regulado por un interruptor, el cual puede ser un transistor, MOSFET, IGBT o

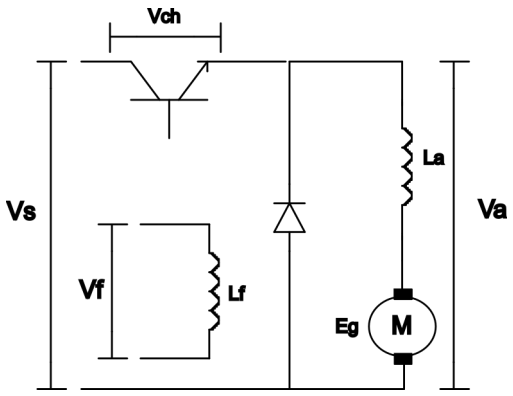


Figura 2.1: Pulsador de aceleración

cualquier otro dispositivo de potencia que permita la conmutación forzada, se puede ver la composición de estos propulsores en la figura 2.1.

El voltaje promedio de la armadura será

$$V_a = K * V_s$$

donde K, es el tiempo de alto del motor durante un periodo de tiempo, el valor de K es de entre cero y uno, donde uno sería que se deja todo el voltaje durante todo el periodo.

Para lo corriente de armadura pasa lo mismo

$$I_a = K * I_s$$

la ventaja de este propulsor, es que la corriente nunca se ve interrumpida, ya que el diodo ayuda a la armadura a conservarla, esto hace que no tenga componentes armónicas la corriente y hace mayor la eficiencia del motor de CD.[Rashid95]

Capítulo 3

Sistemas Operativos en Tiempo Real

3.1. Introducción

Un sistema operativo en tiempo real, es un sistema operativo que al recibir estímulos físicos externos reacciona a estos de una manera esperada, todo esto en un periodo de tiempo establecido según las necesidades del sistema, esta propiedad de que se cumpla en un determinado tiempo, es lo que lo hace diferente a los demás sistemas operativos, este generalmente puede ir sobre algún computador o microcontrolador.

Su uso se ha esparcido a muchos ámbitos de la ingeniería debido a su sencillez, escalabilidad y su robustez ante los errores. Uno de los usos más comunes es el control de procesos, donde un sistema embebido es el encargado de controlador todo los procesos que se están realizando. El núcleo de este sistema es un computador con un RTOS, él cuál tiene que responder en un intervalo de tiempo, para que los procesos no se salgan de control.

Este modelo empezó a ser usado desde los años 60 donde por medio de un computador se garantiza el flujo estable de un fluido. Para esto se tenía un sensor con el cual se medía la cantidad de fluido que estaba pasando y para controlar, se contaba con una válvula, en la cual el ángulo de apertura era controlada por el computador, este sistema era muy efectivo para controlar el fluido[Burns A.03].

3.2. Interrupciones

Es la pausa asíncrona del proceso del sistema, para realizar una función, la cual se debe de haberse establecido previamente. Generalmente estas acciones son provocados por procesos los cuáles no se sabe en qué momento van a pasar, son de vital importancia para el RTOS, ya que este debe de ser programado para atender estas interrupciones.

Se debe considerar que una interrupción en un RTOS no debe de llamar a funciones que puedan bloquearse, ni conmutar tareas, para que esto no llegue a pasar, es necesario configurar al RTOS, para iniciar las rutinas de atención a interrupciones con funciones especiales desde una rutina para servicio de interrupciones.[González]

3.3. Gestión del tiempo

En la mayoría de los RTOS esta se realiza por medio de un reloj en tiempo real, el cual es un reloj que mantiene la hora actual, con ayuda de este y un contador se provoca una interrupción periódica cada determinado tiempo, que se conoce como tick de reloj, el cual es usado como unidad de tiempo en diversos procesos que lleva el RTOS, el tiempo que transcurre entre cada tick puede ser modificado.[González]

3.4. Tareas

Es la unidad de subprograma de un sistema operativo en tiempo real, se ejecutan de manera repetida e indefinida, están asociadas a una función donde se declaran sus parámetros.

Las tareas pueden ser pausadas, iniciadas o finalizadas, desde el manejador de estas.Otra de las características que se le modifican son el tiempo de ejecución, el acceso a recursos y datos de entrada y salida. Pueden ser ejecutadas en paralelo, esto si se cuenta con múltiples núcleos, ya que cada núcleo realizará una tarea.

Se les coloca una prioridad, para que así el RTOS tome la decisión de a cuál tarea darle el acceso al recurso. [González]

3.5. Planificador

Encargado de gestionar las prioridades de las interrupciones, dependiendo de cuáles son de mayor importancia para el RTOS, también de manejar los tiempos que tardan en ejecutarse.

Se pueden clasificar como:

- Cooperativos, donde el planificador llama a las tareas con mayor prioridad para ser ejecutadas, una vez que ésta termine se llama a la siguiente tarea con más prioridad, esto se cumple siempre que las tareas no tarden en responder más de un tiempo límite.
- Expropiativo, en éste tipo el planificador es llamado automáticamente cada cierto tiempo, para ejecutar la tarea que este crea necesario, pero tiene el problema de que si se tienen dos tareas al mismo tiempo no sabe cuál ejecutar.

[González]

3.6. Semáforos

Los semáforos se pueden ver cómo banderas, que indican si se debe o no iniciar una tarea en específico. Estas pueden ser activadas por cualquier elemento dentro del software ya sean interrupciones o otras tareas, puede haber varios semáforos en un RTOS inclusive puede haber uno para varias tareas o un semáforo por cada tarea.

Sirven para gestionar el inicio de las tareas, esto para que la coordinación sea mejor, también se puede usar para sincronizar las tareas, por ejemplo, no ejecutar una tarea hasta que se realice una anterior o ejecutar una tarea después de una interrupción en específico.[González]

3.7. Colas

Estas son arreglos que ocupan una parte de la memoria del microcontrolador o computador, por los cuáles se puede enviar información entre las tareas. Se pueden ver cómo sistemas FIFO donde el primer dato en entrar es el primero en salir, estas pueden ser del tamaño que se necesitan y pueden ser para varios tipos de variables. Son usadas cuando

se requiere retener por un corto periodo de tiempo alguna información o para cuando una tarea necesita recibir datos de muchas más tareas. El orden en el cuál se van almacenando es decidido por el RTOS. Otra propiedad importante es que cuando la cola se llena o vacía por completo el RTOS puede bloquear ya sea la escritura o lectura de la cola.[González]

3.8. FreeRTOS

Es un sistema operativo en tiempo real de grado profesional, el cuál funciona para dispositivos embebidos entre los que están 35 microcontroladores y pequeños microprocesadores, para el desarrollo de este kernel ha sido necesario la contribución de diferentes empresas a lo largo de 15 años. Cuenta con funciones para el uso de RTOS como lo son, creación de tareas, semáforos y colas. Es un proyecto de software libre esto lo hace totalmente gratis, también cuenta con una gran documentación sobre su uso en su página oficial de Internet.[AWS]

Capítulo 4

Simulación en tiempo real

4.1. Introducción

Es una simulación que se calcula en tiempo real, esto quiere decir que todos los cálculos necesarios para hacer la simulación se deben realizar al tiempo al que se está realizando la simulación, en comparación a una simulación normal en la cual el tiempo es diferente. Es muy común que para este tipo de simulaciones se utilicen un RTOS.[rts]

4.2. Hardware in the Loop

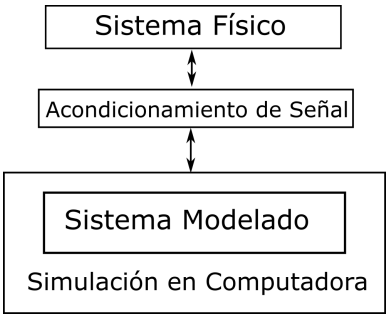


Figura 4.1: Esquema de un HIL

Técnica mediante la cual un modelo simulado en tiempo real interactúa con una planta física. Este tipo de técnicas son muy utilizadas para validar el funcionamiento de un controlador o un modelo, esto debido a su bajo coste y flexibilidad, en la figura 4.1 se

muestra un diagrama de un HIL.[rts]

4.3. Rapid Control Prototyping

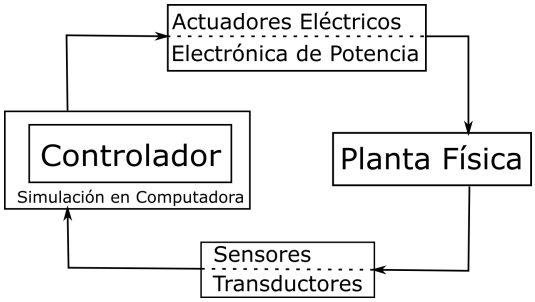


Figura 4.2: Esquema de un RCP

Proceso para probar controladores, basado en un HIL, donde el modelo del controlador es simulado en tiempo real, mientras controla la planta física. Dado que las señales de la planta física son de magnitudes mayores a las del simulador en tiempo real se requieren módulos para acoplar estas señales, en la figura 4.2 se muestra un ejemplo.[rts]

4.4. OPAL-RT 5600

Plataforma creada para la simulación en tiempo real, construida como una computadora de alto desempeño, contando con varios procesadores, memoria RAM, disco de estado sólido, puertos PCI en la parte posterior y frontal. Cuenta con FPGAs las cuáles se configuran según el modelo a simular, dando una gran capacidad y rapidez de procesamiento. Como cualquier computadora cuenta con salidas de puerto serie, PS/2, VGA, USB y Ethernet. En cuanto al software cuenta con un RTOS basado en una versión modificada de Red Hat, distribución de linux. En la figura 4.3 se muestra el diagrama a bloques del OPAL-RT 5600.

El computador donde se realiza el modelo, denominado computadora host, construye un archivo con extensión c el cual contiene la compilación del modelo, posteriormente se envía este archivo al OPAL-RT 5600 con el cual realiza la simulación en tiempo real.

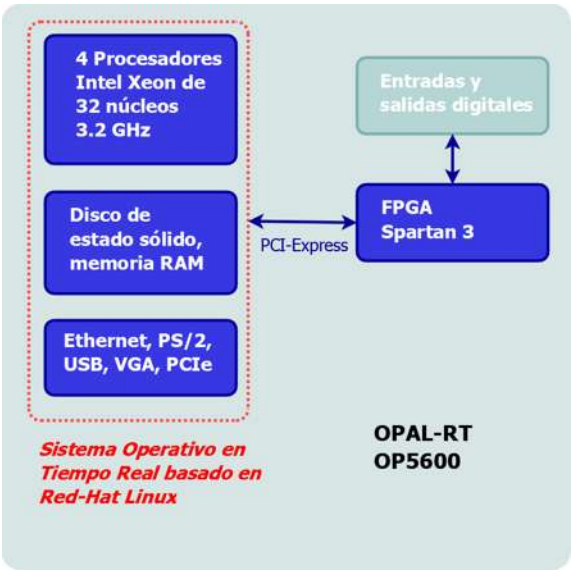


Figura 4.3: Diagrama de bloques del OP5600

La empresa también vende módulos para ampliar el rango de posibilidades que se tiene con el simulador en tiempo real, módulos que generalmente pueden manejar potencias grandes, se muestra una imagen tomada del OPAL-RT OP5600 en la figura 4.4.[OPAL-RT19]

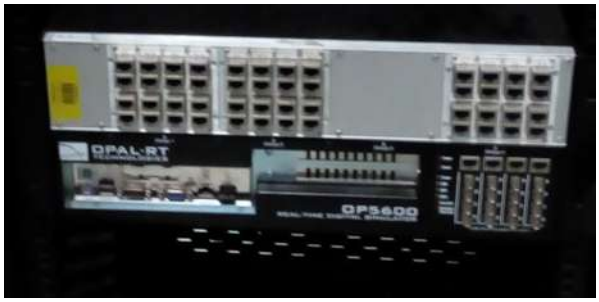


Figura 4.4: Simulador OPAL-RT OP5600

4.5. Módulo OP8660 HIL Controller

Es un módulo para hardware in the loop, fabricado por OPAL-RT, el cuál contiene dos salida para inversores, dos entradas para encoder, 16 salidas analógicas las cuales tienen

un rango de +-15 volts, 8 salidas digitales, 8 entradas digitales,16 entradas análogas de +-15 volts, 16 entradas analógicas de alto voltaje de hasta 600 voltios y 16 medidores de corriente capaces de medir hasta 15 amperes. Para el uso de este módulo el software RT-LAB incluye módulos en donde se configuran par su uso, en la figura 4.5 se muestra el OP8660.[OPAL-RT19]

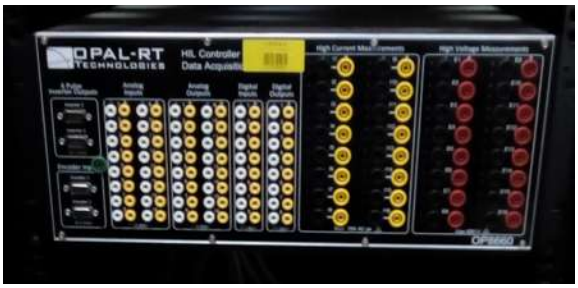


Figura 4.5: Módulo OP8660

Capítulo 5

Microcontrolador ESP32

5.1. Introducción

El ESP32 es un microcontrolador de ultra bajo consumo, con integraciones de alto nivel como lo son antenas, amplificadores, amplificadores de bajo ruido para recepción, filtros, módulos administradores de potencia, PWM, puertos de propósito general. Aparte cuenta con múltiples puertos de comunicación como lo son wifi, SPDI, SDIO, I2C, I2S, IR, UART, CAN, Bluetooth; tiene una arquitectura de 32 bits, con un o dos núcleos; memoria SRAM; memoria ROM; timers; convertidor analógico digital; un sensor de campo magnético y sensor táctil. Es ideal para cualquier tipo de aplicación de sistemas embebidos, debido a sus muchas especificaciones.[ESPRESSIFa]

5.2. Modulación por Ancho de Pulso

Es una modulación muy usada en microcontroladores, por medio de la cuál a una señal con un periodo constante se le varía el ciclo de servicio.

Esta puede servir para comunicaciones o para controlar la energía media suministrada. La forma típica de generar esta señal se realiza por medio de un comparador, donde entra una señal diente de sierra y una constante. La frecuencia de la señal va estar dada por la señal de diente de sierra y el ciclo de servicio por la señal constante.

En los microcontroladores la manera de hacerlo es igual, solo que en vez de una señal diente de sierra, se usa un contador y la constante es un registro. Esta técnica es muy usada para controlar motores, elementos termoeléctricos y otras aplicaciones.

El ESP32 cuenta con dos módulos dedicados a PWM. Uno se recomienda usar para control de motores y luces inteligentes, debido a que cuenta con submódulos que permiten reaccionar de manera automática a estímulos externos sin necesidad de pasar por el procesador. El otro módulo PWM es más sencillo y solo se puede generar la señal PWM cómo normalmente se hace, este cuenta con 16 canales independientes que pueden operar a diferente frecuencia y ciclo de servicio.[ESPRESSIFb][Hirzel]

5.3. Convertidor Analógico a Digital

Es un dispositivo capaz de pasar una señal analógica a una valor digital, tomando muestras analogicas cada cierto tiempo, a este periodo se le denomina tiempo de muestreo.

Generalmente se puede clasificar los convertidores es por su velocidad a la que pueden tomar muestras y por cuantos bits de resolución convierten los valores analógicos, entre mayor sean estos dos parámetros mejor calidad tendrá la señal digital y más parecida a la analógica.

El método conversión más usado es el de aproximaciones sucesivas, las ventajas de este convertidor es que es bastante rápido.

El ESP32 integra dos convertidores con una resolución de 12 bits, capaces de tomar a hasta 200,000 muestras en un segundo.[ESPRESSIFb]

5.4. Módulo WI-Fi

El ESP32 implementa un módulo con los protocolos TCP/IP y 802.11 b/g/n con MAC asignada. Dependiendo de la tarjeta del desarrollo puede contar con entradas adicionales para diferentes tipos de antenas compatibles con las velocidades de transmisión que maneja.

Hay funciones que el microcontrolador hace de manera automática cómo crear hasta 4 interfaces virtuales, modo promiscuo, desfragmentación, monitoreo beacon entre otras.[ESPRESSIFb][ESPRESSIFa]

Capítulo 6

Comunicación

6.1. Introducción

Las comunicación por Internet cada día toma más relevancia en el desarrollo de la actividades del ser humano, se ha arraigado en todos los ámbitos que nos concierne, desde un simple mensaje hasta controlar a distancia, su popularidad y uso se ha extendido. Vemos hoy el impacto en diversas áreas la ingeniería como lo son seguridad informática, encriptación y Internet de las cosas.

6.2. Protocolo IP

El protocolo de Internet o IP, es un protocolo para la comunicación de datos digitales, este se ubica sobre la capa de red, donde realiza varias funciones las cuáles incluyen fragmentar paquetes si es necesario, direccionamiento mediante las direcciones lógicas y distribución de paquetes. La unidad de información con la que trabaja este protocolo es el datagrama.[ipt][León81]

6.3. Protocolo TCP

Es un protocolo que ofrece un flujo de bytes orientado a conexión, que garantiza la entrega de datos. Es usado en aplicaciones de red que requieren una garantía en la entrega y no pueden hacer tiempos de espera o retransmisiones. Las conexiones realizadas mediante este protocolo se conectan a puertos, estos están bien definidos y se dedican a aplicaciones

específicas. Al ser un protocolo de ventana deslizante, tiene un tamaño de ventana definido, el cual determina la cantidad de datos que pueden ser transmitidos.[T. Socolofsky91]

6.4. Protocolo UDP

Este es un protocolo para el envío de datagramas, el cual permite enviar datagramas sin una conexión directa entre el transmisor y receptor, ya que incluye suficiente información en la cabecera del datagrama para que este llegue a su destino. No contiene ninguna forma de control de flujo, por lo que los paquetes pueden adelantarse unos con otros, tampoco cuentan con alguna forma de comprobar si han sido recibidos de manera correcta. Este tipo de protocolo son usados cuando se quiere transmitir una gran cantidad de información, por ejemplo transmisión de vídeo, audio o información en tiempo real, el envió de esta información puede ser de un elemento a varios.[ipt][Postel80]

6.5. Tecnología Websocket

Es una tecnología que permite conexión entre el cliente y un servidor de larga duración, permite una comunicación bidireccional o full-duplex. Los mensajes son distribuidos de manera inmediata dando como resultado una conexion con muy poca latencia, esto permite conexiones más dinámicas que las tradicionales. Facilita la comunicación en tiempo real, esto debido a que permite al servidor enviar información al cliente sin que este la solicite, esto siempre que la comunicación esté abierta.[Alexey Melnikov]

Capítulo 7

Desarrollo

7.1. Esquema motor de CD - Generador Síncrono

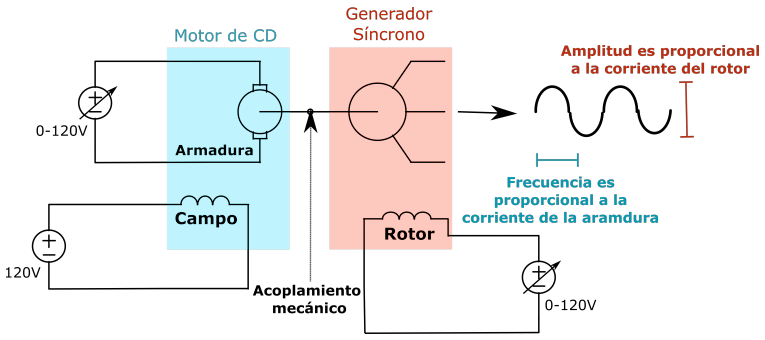


Figura 7.1: Esquema motor de CD maquina síncrona

Se elige el esquema motor CD - Generador síncrono, mostrado en la figura 7.1, debido a que con esta se puede controlar tanto la amplitud del voltaje generado como la frecuencia. El motor de CD se encarga de otorgar el par necesario para la generación, su velocidad es proporcional a la frecuencia del voltaje generado, con esto controla la frecuencia.

$$I_{armadura} * K = f_{generación} \tag{7.1}$$

$$Sí\, I_{campo}\, es\, constante$$

La corriente del rotor de la máquina síncrona es proporcional a la amplitud de voltaje generado,con este se controla la amplitud del voltaje generado.

$$I_{Rotor} * K = V_{generación} \tag{7.2}$$

Con esta configuración podremos controlar todos los aspectos de la generación como se observa en la figura 7.2.

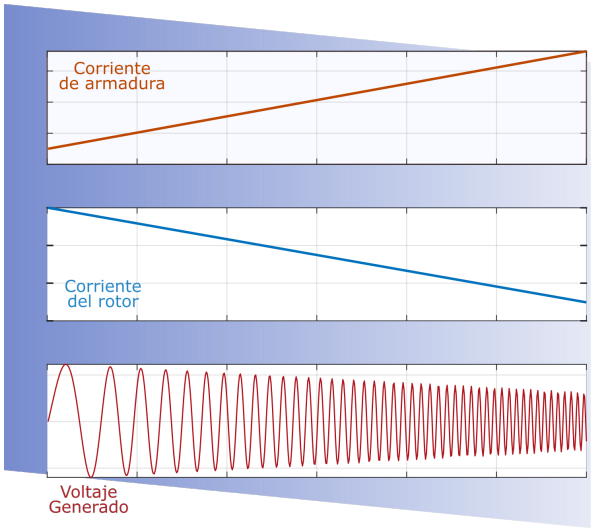


Figura 7.2: Esquema de generación

7.2. Diseño de Propulsor

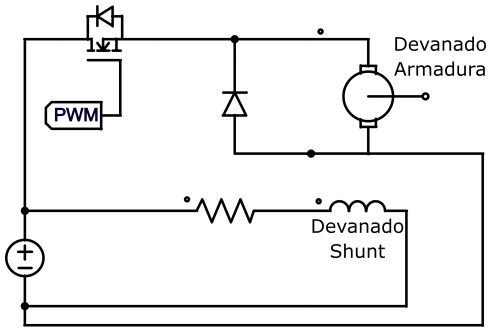


Figura 7.3: Diagrama del primer propulsor

La primera parte desarrollada, es la de potencia para la cual se seleccionó la topo-

logía mostrada en la figura 7.3.

La topología de potencia seleccionada es la de un propulsor, debido a su fácil elaboración. El elemento de conmutación activa es un IGBT, para la selección de este se toma en cuenta de la corriente, voltaje y frecuencia a utilizar.

Para el caso de la corriente se consideran los picos de corriente de 50 amperes. La velocidad nominal es de 1800 revoluciones por minuto. Con un voltaje de 120 volts constantes. La frecuencia de conmutación es de 30 KHz, con esta el par no se ve afectado.

Se optó por el IGBT SGTW90V60F el soporta hasta 600 volts, tiene una corriente nominal de 80 amperes, soporta picos de 240 amperios y la frecuencia de conmutación son cercanas a 500 KHz. Las características del IGBT sobrepasan el nivel de operación y permiten realizar pruebas con parámetros mayores a los valores nominales.

El diodo se selecciona tomando en cuenta el voltaje, corriente y potencia, seleccionando un diodo de alta potencia.

Se generar el PWM que controla la corriente media del devanado. Se usó un programa en el sistema Arduino en el cual se configuraron los registros para la frecuencia deseada. Un elemento que se adiciono para proteger al microcontrolador fue un optoacoplador 2N25 configurando como se muestra en la figura 7.4.

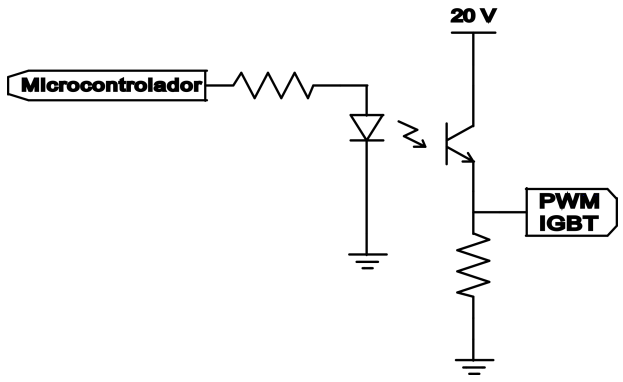


Figura 7.4: Diagrama optoacoplador

Para la prueba del circuito de la figura 7.4, se cargó el programa en el microcontrolador, este recibe un valor de entre 0 y 255 por medio del puerto serie, así varía el ancho de pulso de la salida PWM, para controlar el motor de CD en lazo abierto. Los resultados no fueron los esperados debido a varios problemas los cuáles fueron:

- Calentamiento excesivo del dispositivo de conmutación
- El rango en que se varia la velocidad es más pequeño de lo esperado

Analizando el circuito con el osciloscopio, se observó que la salida del optoacoplador era una señal deformada (figura 7.5 b)). Al consultar la hoja de datos más a detalle se encuentra, que a altas frecuencias la señal de salida tiene este problema, debido a que el cambio de voltaje no es rápido. Investigando sobre la forma correcta de conmutar IGBTs, se concluyó que el uso de drivers con salida totem pole es la solución al problema.

Se eligió el driver L293d que soporta las frecuencias usadas. Este cuenta con salida totem pole (figura 7.5 c)), soporta más de los 20 voltios necesarios para la conmutación del IGBT.

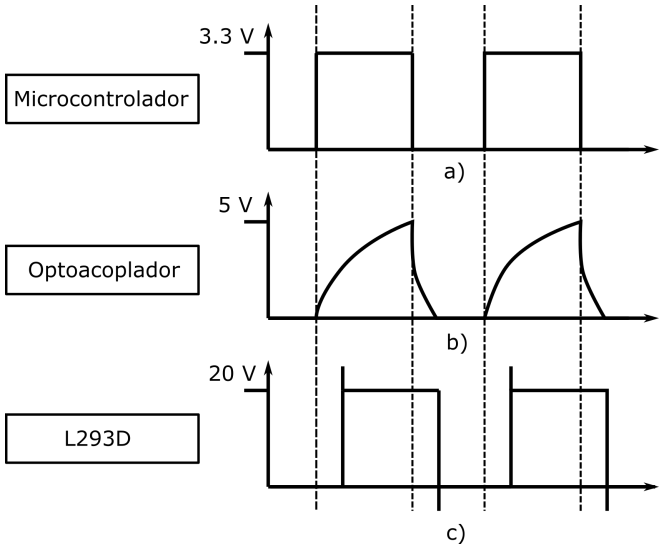


Figura 7.5: Diagramas de los cambios en la señal PWM

Está conectado entre el optoacoplador y el IGBT. Con esto la señal PWM mejora y el microcontrolador continúa aislado.

Hay pérdida en la resolución del PWM, debido a la tardanza del optoacoplador. Con las pruebas se comprobó que los cambios fueron los correctos dado a que ahora varía la velocidad en lazo abierto y el IGBT no mostraba calentamiento.

El diodo comenzó calentarse, esto debido a que el diodo conmuta a la frecuencia de 30 KHz de forma pasiva, investigando diodos que cumplan esta característica se optó por

el diodo VS-ETH3007-M3. Es un diodo de recuperación rápida capaz soportar frecuencias en el orden de los kilohertzios, soporta una corriente nominal de 30 amperes y un voltaje de 650 voltios. Realizando pruebas con este diodo, el funcionamiento del motor de CD fue correcto, ninguno de los componentes mostró calentamiento y el control en lazo abierto fue en todo el rango de velocidad.

7.3. Retroalimentación

Para el control es necesario medir la velocidad del motor de CD. El laboratorio cuenta con tacómetros, que son un transductores de velocidad a voltaje, donde por cada 1000 revoluciones por minuto el tacómetro entrega a la salida un volt, este se comporta de una manera lineal, esto lo hace adecuado para el control dado que no afecta la medición.

Para acoplar esta señal al microcontrolador, se colocó un seguidor de voltaje implementado con un amplificador operacional, para aislar el tacómetro del circuito electrónico, más un ganancia que mejora el rango de voltaje.

El tacómetro presentaba un oscilación a la salida, para filtrar la oscilación se adiciono un capacitor. Realizando las pruebas, se encontró que aún presentaba oscilaciones la señal del tacómetro, por lo que sólo se ajustó el valor del filtro, en la figura 7.6 muestra el diagrama del acoplamiento de señal del tacómetro.

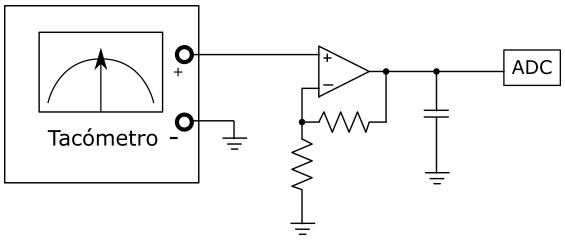


Figura 7.6: Esquema de sensor de velocidad

7.4. Programación Controlador Digital

7.4.1. Programación del Controlador

El controlador digital se hace a través de un microcontrolador, en el cual se programa las operaciones que definen a un controlador.

La referencia se recibe por el puerto serie del microcontrolador, la retroalimentación es medida por medio del ADC que está conectado al tacómetro.

Se calcula el error con la diferencia entre la referencia y la retroalimentación.

(7.3)

La acción de control proporcional es igual al error multiplicado por la constante proporcional.

(7.4)

Los errores se va sumando en una variable donde se van acumulando, esto es para el cálculo del integrador.

(7.5)

En los controladores se coloca un valor máximo en la sumatoria de error acumulado para acotar la acción de control, dado que puede causar problemas, esta técnica es conocida con el nombre de anti-windup. Está acotamiento debe ser tanto como para el límite inferior y superior.[win13]

La parte derivativa, se calcula con diferencia entre el error actual y el anterior.

(7.6)

Una vez calculadas cada una de la partes del controlador se suman. Con esto se tiene la acción de control para aplicar a la planta.

Las constantes de los controladores se sintonizaron de manera manual. Empezando con las todas las constantes en cero, se incrementa la constante proporcional hasta que se se acerca a la respuesta deseada, el controlador presenta oscilaciones debido a ser únicamente proporcional, para reducir estas oscilaciones se incrementa la constante integral hasta que se obtenga un comportamiento apropiado. [**?**]

La respuesta del controlador al arranque saldría del límite de operación seguro, debido a que el controlador inicia en cero he intenta alcanzar la referencia, esto no es bueno para los elementos de potencia, debido a el pico de corriente. Por lo que se programó un algoritmo de arranque suave que por medio de una rampa se establece en el valor de la referencia en lazo abierto. Este se realizó con un ciclo que incrementa el ciclo de servicio del PWM. El código del controlador se encuentra en el apéndice A.A, y el diagrama de flujo se muestra en la figura 7.7.

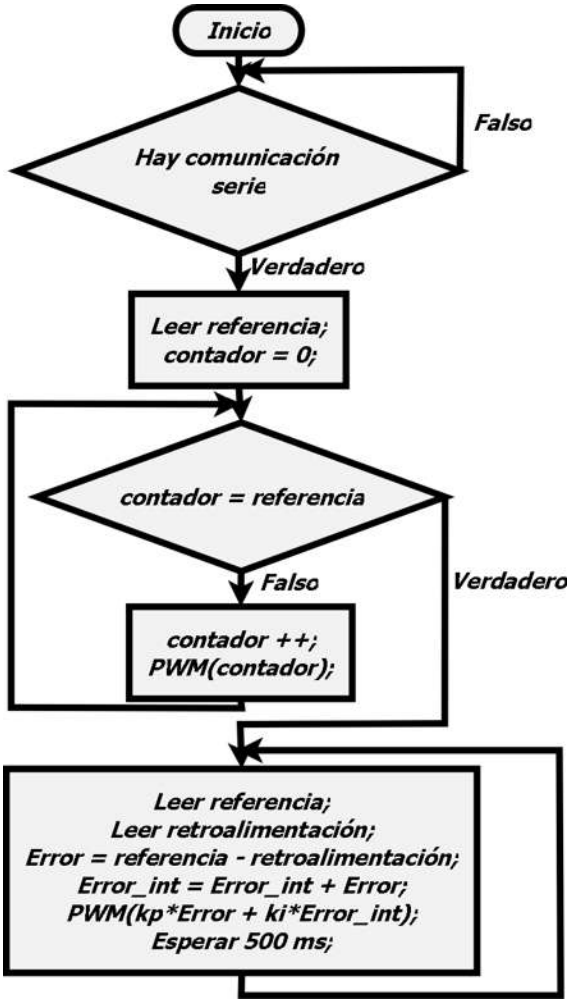


Figura 7.7: Diagrama de flujo

El modelo del controlador digital quedaría de la siguiente manera

$$G(z) = (k_P + \frac{k_I * Ts}{z^{-1}}) * E(z) \tag{7.7}$$

donde

$$E(z) = Ref(z) - Y(z) \tag{7.8}$$

Los valores de k_I y k_P son los que se sintonizaron, el valor de Ts es constante y se seleccionó $0.5s$ por ser un tiempo en el cual el sistema embebido puede hacer todas las tareas del controlador, este modelo de controlador es basado en un modelo continuo el cual fue discretizado para su uso en un control digital, el controlador es multiplicado por la función de transferencia del error, la cual es igual a la referencia menos el lazo de retroalimentación.[**?**]

7.4.2. Comunicación

Al hacer la pruebas del controlador, el puerto serie no funcionaba debido al ruido de la conmutación en los elementos de potencia. Por este fallo la comunicación serie, de donde llegaba la referencia se cambió por el convertidor analógico a digital. Una justificación para este cambio en la comunicación es conectar el sistema embebido con el OPAL-RT por medio módulo OP8660, el cual convierte las señales digitales provenientes de la simulación a analógicas. Al conectar el módulo OP8660 al ADC se tiene el control completo, el diagrama de comunicación entre el OPAL-RT y el módulo de control se muestra en la figura 7.8.

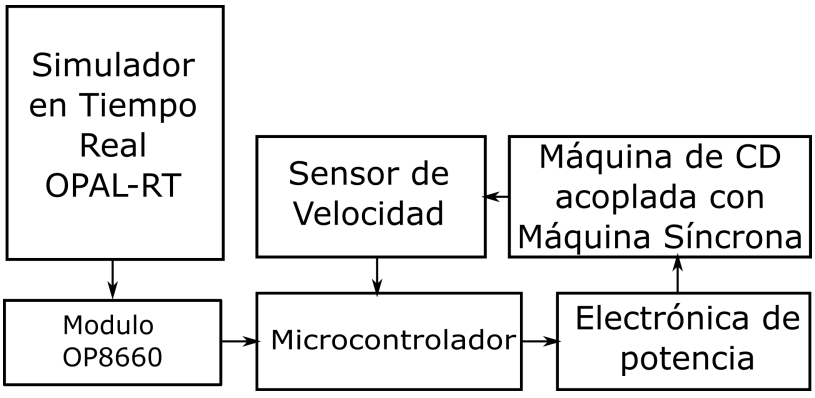


Figura 7.8: Modelo de comunicación de prueba del propulsor controlado

7.5. Pruebas del Propulsor Controlado

Se realizaron pruebas con el controlador, la primera se cargó un modelo en el simulador en tiempo real, el cual se observa en la figura 7.9
En este modelo hay dos bloques principales, el maestro que directamente se carga al simulador en tiempo real y la consola con el cual nosotros podemos controlar la simulación desde la computadora host.

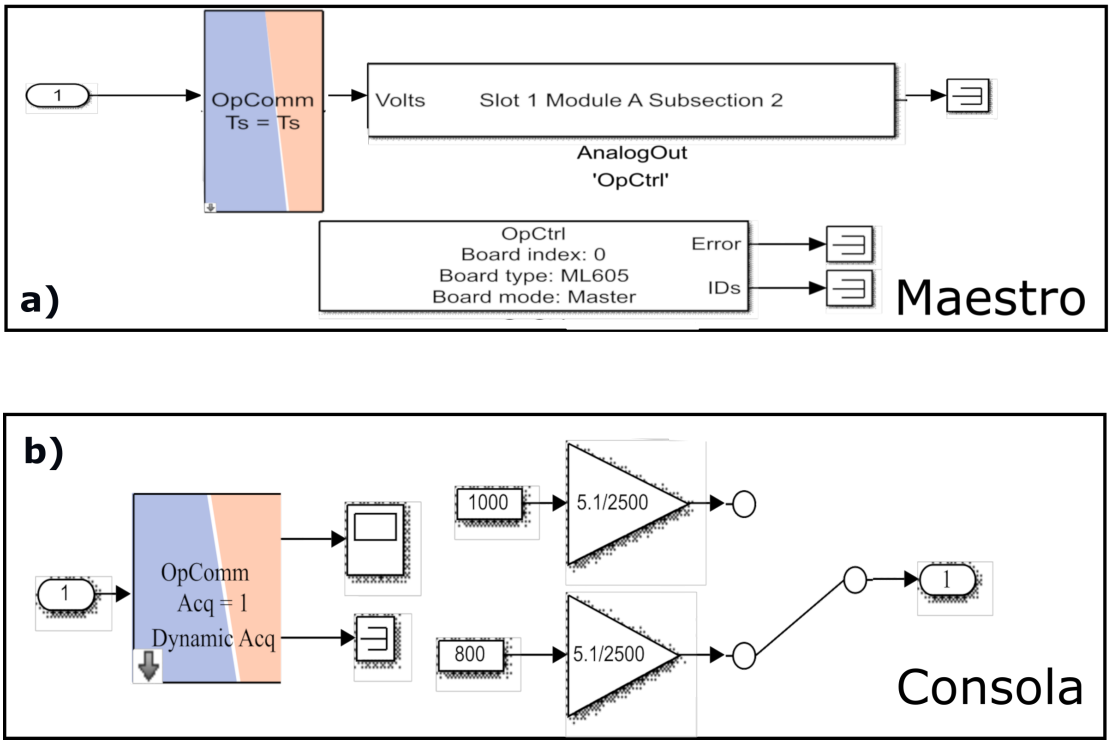


Figura 7.9: Primer Modelo para la prueba del Propulsor Controlado

En el bloque maestro está el bloque OpCtrl, en el va la información del módulo que se desea usar, el bloque OpComm es el encargado de la conexión entre el módulo de la consola y el maestro y el bloque de AnalogOut este es el encargado de convertir la señal a analógico por medio del módulo OP8660.

En el bloque de Consola se tiene el bloque OpComm, necesario para la comunicación con el módulo maestro. Un selector que tiene las referencias de velocidad, en este

caso 800 o 1000 revoluciones por minuto con un factor de conversión necesario para transformarlas a voltaje. Dado a que este módulo se puede controlar desde la computadora host, es posible modificar la referencia en cualquier momento.

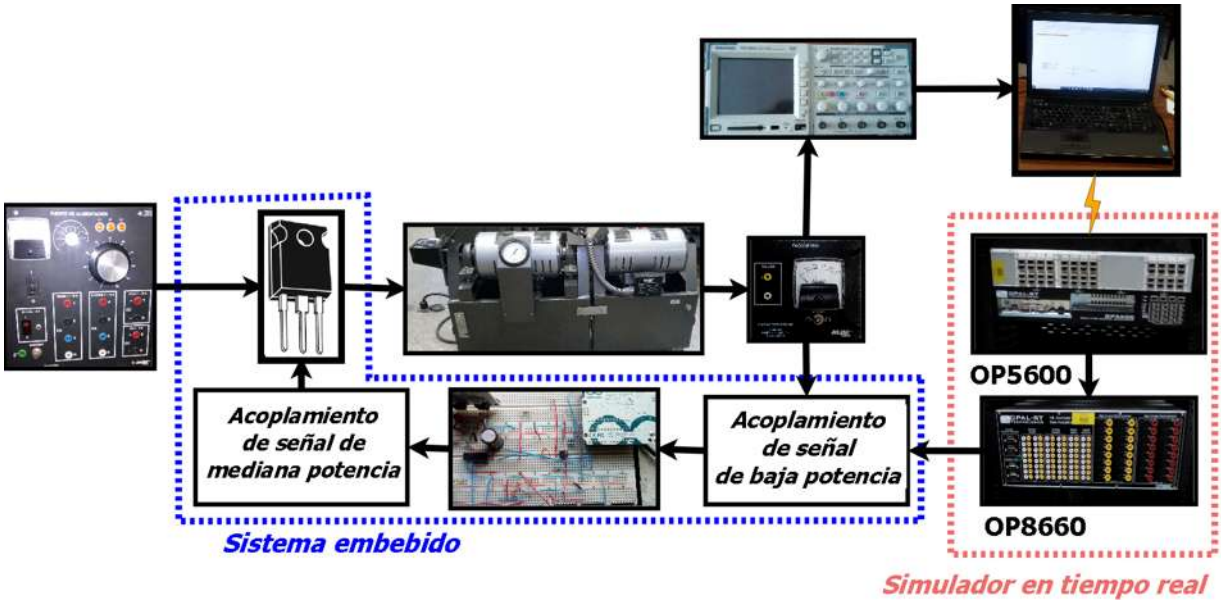


Figura 7.10: Diagrama a bloques de las conexiones de prueba

En la figura 7.10, se observa cómo fueron la conexiones para este experimento, el sistema embebido que se diseñó en la parte marcada en azul.

El simulador en tiempo real está conectado de manera física al módulo OP8660 y de manera inalámbrica a la computadora host, desde la cual se cambia la referencia.

El voltaje del tacómetro se mide con el osciloscopio, en el cuál se guardan los datos de las pruebas para después ser procesados en una computadora. El osciloscopio es de la marca Tektronix, modelo TPS 2024, el cual cuenta con cuatro canales independientes, con una capacidad de muestreo de hasta 2 mil millones de muestras en un segundo.

Los resultados de la prueba se muestran en la figura 7.11, donde se observa que el controlador es capaz de mantener la referencia. Se ajustaron las constantes del controlador para esta prueba con una Kp de 0.1 y un Ki de 0.1, esto es parte del proceso de sintonizar el controlador. Se observa cómo en controlador tiene un error en estado estable de cero y cómo alcanza la referencia en pocos segundos.

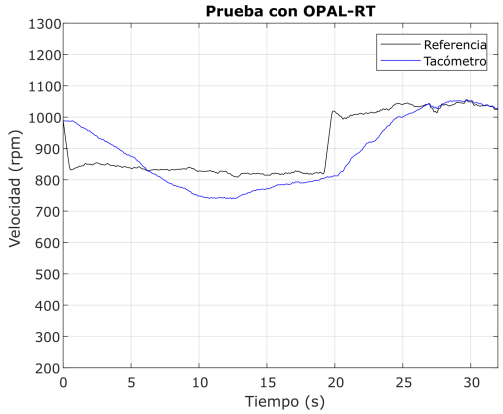


Figura 7.11: Primera prueba del propulsor controlado

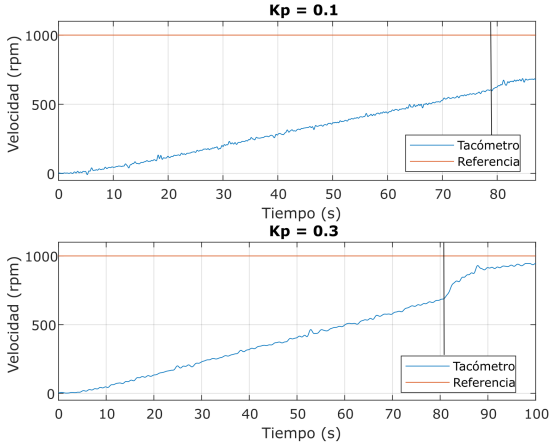


Figura 7.12: Prueba de arranque del propulsor con un control proporcional

En la figura 7.12 se observa cómo el controlador arranca el motor de manera suave, para unos segundos después entrar el control, el cual tiene un sobreimpulso, pero el error en estado estable no es cero, debido a que es un control puramente proporcional en este caso se probaron dos constantes Kp.

Probando el arranque con un controlador PI, con Ki = 0.4 y Kp = 0.4, se observa en la figura 7.13, el arranque es suave debido a la rampa para encenderlo, justo después tiene un sobreimpulso de velocidad, para establecerse en la referencia en pocos segundos. Este controlador no tiene error en estado estable, también es mucho veloz que el proporcional.

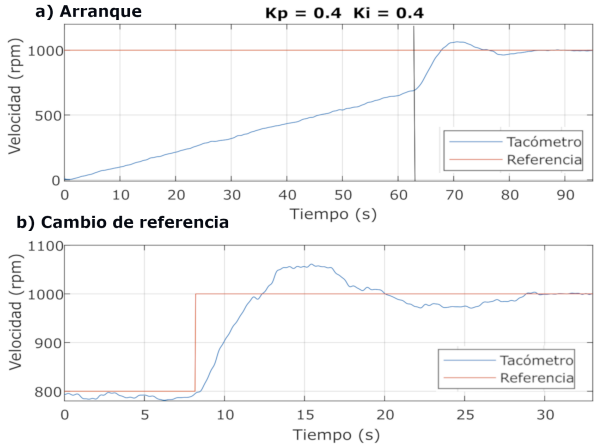


Figura 7.13: Prueba de arranque del propulsor con un control PI

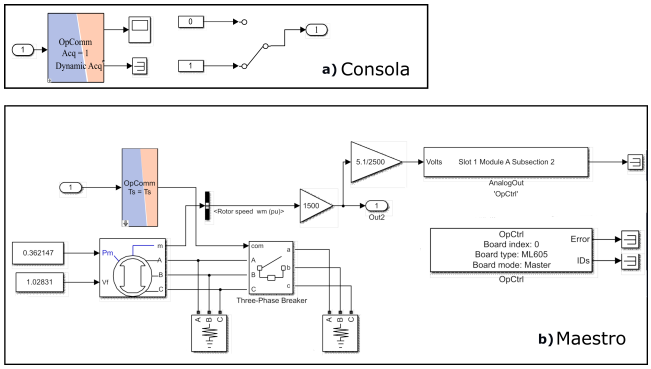


Figura 7.14: Segundo modelo para la prueba del propulsor

En la figura 7.13 se incluye un cambio de referencia de 800 a 1000 revoluciones por minuto, donde se ve que el sobreimpulso es menor.

El modelo para la segunda prueba es el que se muestra en la figura 7.14.

El modelo que se carga al simulador en tiempo real, es el de un generador conectado a su carga nominal. Por medio de un interruptor se adiciona más carga, lo que hace que el motor baje la frecuencia a la que está generando y por lo tanto la velocidad del rotor. Esta velocidad del motor simulado es la referencia para el controlador del motor de CD.

Los resultados obtenidos se muestran en la figura 7.15, se ve cómo la referencia va bajando de una manera no lineal cuando se le conecta una carga mayor a la nominal.

El controlador trata de seguir esta referencia, lo cual logra en los valores estables, pero el transitorio varía. Esto es debido a que el controlador no puede frenar el motor de CD, si no solo quitarle la energía, lo cual hace imposible que el motor baje la velocidad en ese tiempo.

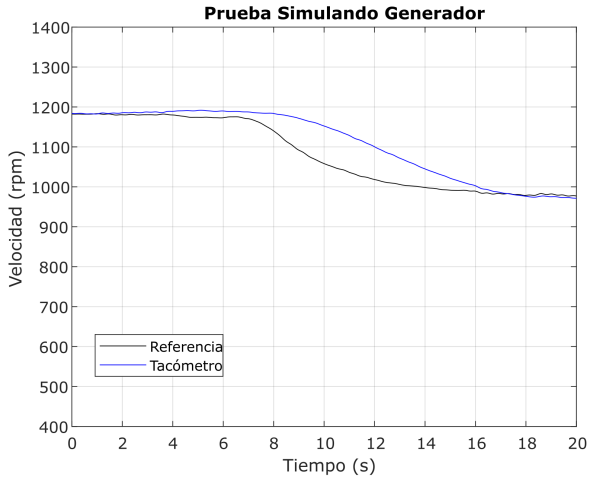


Figura 7.15: Prueba simulando la velocidad de un generador

7.6. Comunicación con el simulador en Tiempo Real

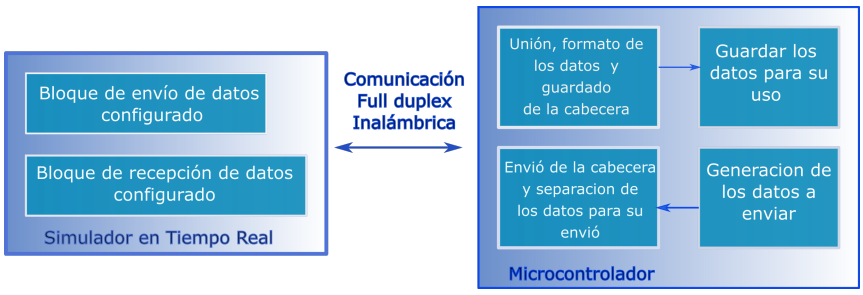


Figura 7.16: Diagrama del proceso de comunicación vía UDP/IP

En las pruebas anteriores fue necesario utilizar el módulo OP8660, esto no es lo adecuado, debido a que se busca un sistema embebido que sea totalmente independiente de cualquier otro módulo por lo que se debe de cambiar la comunicación. Se investigaron las opciones para la comunicación con las que cuenta el OPAL-RT, entre las cuáles están

unos bloques en RT-LAB, que realizan una conexión como por TCP/IP o UDP/IP, esta fue la comunicación que se seleccionó. La configuración necesaria es bastante sencilla solo se selecciona el puerto y la IP.

Es necesario cambiar la tarjeta de desarrollo por una adecuada que tenga la comunicación UDP/IP o TCP/IP, la que se seleccionó fue una tarjeta con un microcontrolador ESP32, el cual cuenta con todo lo necesario para este tipo de comunicación.

Para realizar la prueba de este protocolo se cargó en él OPAL-RT, el ejemplo de comunicación asíncrona vía UDP/IP, el cual envía datos de una señal sinusoidal junto a cuatro constantes.

Mientras que en el microcontrolador se cargó el código de un Websocket asíncrono. Para que este funcionara de manera adecuada se le asigno una IP fija al microcontrolador.

Debido a que en el laboratorio hay diversos módem con el mismo SSID y solo uno de estos es capaz de asignar la dirección IP fija, se usa el BSSID de este punto de acceso para su conexión, junto con esto se le habilitó la comunicación serie a el microcontrolador para ver lo que este recibe por medio de la comunicación por UDP/IP.

Una vez que se programó se se estableció la conexión, recibiendo datos los cuáles no parecían tener ningún sentido. Modificando el modelo del OPAL-RT para que solo enviará un dato constante y decodificar lo que este estaba enviando. Se observó que envía paquetes de un byte, que están codificados en un formato double de 32 bits, por lo tanto enviaba cuatro paquetes.

Los datos se guardan de manera automática en una sección de memoria, para darles formato se utilizan apuntadores de tipo byte.

Aparte del dato, se recibe información como el número de paquete y de qué número de bloque de comunicación se envía.

Para el envió de información, se hace el proceso contrario, se envían los paquetes con la información de a qué bloque van dirigidos y qué número de paquete es, más el paquete de información que se divide en 4 bytes. La forma de envió y recepción en el microcontrolador se hace usando la API que se tiene disponible para estas funciones.

7.7. Protecciones de los IGBTs

Para continuar con la pruebas del sistema embebido, se diseñaron unas protecciones para los picos de corriente, debido a una pérdida considerable de IGBTs, el esquema de la protecciones es el mostrado en la figura 7.17.

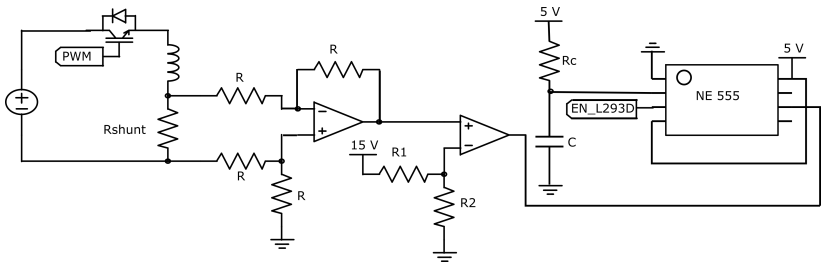


Figura 7.17: Esquema de protección electrónica

La corriente en el IGBT, medida por medio de una resistencia shunt, que soportar la potencia sea de la armadura del motor de CD o del rotor de la máquina síncrona. Para ambos casos se seleccionó resistencias menores a un ohm, el voltaje de estas resistencias va a un amplificador operacional, como se ve en el esquema de la figura 7.17, esto debido a que se desea aislar electrónicamente la resistencia shunt de el circuito de protección, las resistencias marcadas como R en el diagrama, son todas del mismo valor para que la ganancia del voltaje sea uno. Después pasa a un amplificador operacional configurado como un comparador, donde se compara el voltaje de la resistencia shunt con el de un divisor de voltaje, si el voltaje de la resistencia shunt, llega a superar el del divisor de voltaje la salida del operacional se saturara.

A la salida del comparador se encuentra un NE555, que está configurado como monoestable, cuando llega un voltaje alto en el pin de entrada, la salida se pone en alto, y solo puede volver a bajar si se da un pulso en alto en el pin de reset que es el 7.

El pin de inicio está conectado a un circuito RC, lo que significa que la protección se activa después que el capacitor se carga por completo. La salida de este circuito NE555 va conectada a el enable del driver del L293D, por lo tanto si llega a haber un pico de corriente desactiva el IGBT, protegiéndolo así de estos picos. El valor del pico de corriente máximo puede ser modificado por medio del divisor de voltaje.

7.8. Sistema operativo en tiempo real del sistema embebido

El sistema embebido debe responder en el mismo tiempo de simulación del simulador en tiempo real, es indispensable utilizar un sistema operativo en tiempo real en el microcontrolador. Otro de los puntos a favor del uso de un sistema operativo en tiempo real es la tolerancia a fallos, su fácil implementación y la gran escalabilidad que este tiene. El RTOS a usar es FreeRTOS el cuál es una API enfocada a microcontroladores.

7.8.1. Tareas del RTOS

Las tareas que realiza el RTOS son dos, hacer todas las acciones necesarias enfocadas al control de la frecuencia del generador es decir la velocidad de la máquina de CD y la otra controla la amplitud de voltaje generado, ósea del rotor de la máquina síncrona, estas dos tareas prácticamente aran lo mismo y se están ejecutando en conjunto.

Para configurar nuestro RTOS, se declaran las tareas junto a las características de ellas, la función de la tarea, el nombre, la cantidad que se le asigna de memoria, si tienes parámetros de entrada o salida, la prioridad de la tarea y el número del núcleo en el que se va realizar la tarea. Una vez terminada la configuración de las dos tareas se obtiene el periodo de ejecución de estas tareas, esto es importante, porque lo necesitamos para saber si es posible establecer el tiempo de simulación necesitado.

El resultado de la pruebas se muestra en el histograma de la figura 7.18, para el cual se tomaron 500 muestras. Donde se puede observar que la mayoría de veces el proceso tarda 14 microsegundos, esto da suficiente tiempo para los 500 milisegundos a los que se va a realizar la simulación, por lo tanto es posible.

Otra de las configuraciones que se les hicieron a la tareas como parte de la gestión de recursos, fue cambiarlas al núcleo que se tiene libre en microcontrolador, esto para distribuir las tareas de manera adecuada. Como resultado en un núcleo, se está ejecutando la parte de la comunicación, mientras en el otro núcleo se realizan las dos tareas que se crearon.

El objetivo es que la tarea se esté ejecutando en un periodo de tiempo constante, se usó una función para la administración de tareas, de la API de FreeRTOS, que es DelayUntilx, que nos permite que una tarea espere cierto tiempo para que se ejecute de nuevo.

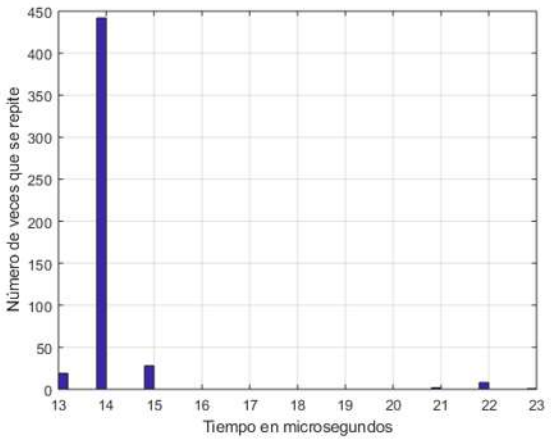


Figura 7.18: Histograma de tiempo del algoritmo de control

7.8.2. Interrupción de comunicación

La comunicación es asíncrona, por lo que la información puede transferirse en cualquier momento debido a la naturaleza del proyecto, la mejor manera de manejar estos eventos es utilizar una interrupción.

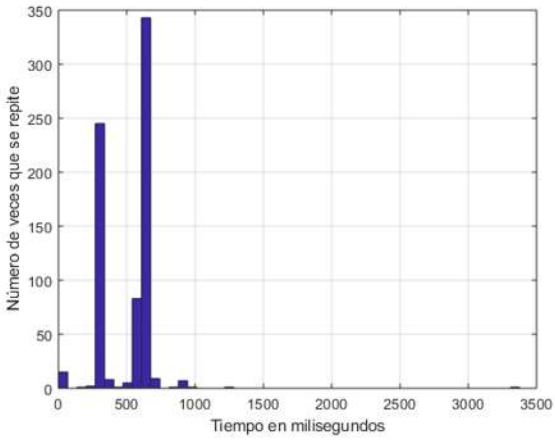


Figura 7.19: Histograma del tiempo de la comunicación

Esta interrupción, se ejecuta cada que el simulador en tiempo real envía un dato. Se necesita medir cada que esté envía un dato, para esto se tomaron 1000 muestras del

periodo de ejecución de la interrupción y determinar la viabilidad de la comunicación en tiempo real, si la latencia es mucha, no se podría. Los resultados de esta prueba se muestran en la figura 7.19, donde se observa que la mayoría de los mensajes llegan 100 milisegundos después y la otra parte llega unos 200 milisegundos antes.Los que llegan antes no representa un gran problema, ya que estos paquetes de información, están dentro del rango aceptable, pero los que no se pueden recuperar sin perder la estabilidad del RTOS, son esos paquetes que superan los 800 milisegundos, esto significa que se perderían el 1.4 % de los paquetes, que son una minoría, por lo tanto este medio de comunicación si es viable para el sistema embebido que se desea realizar.

En la figura 7.20 se observa cómo es que llegan los datos, se intercambia entre 300 y 600 milisegundos. Si esto fuera de una manera continua, es decir, si todos los datos de 600 o más milisegundos fueran seguidos, el tiempo de comunicación se volvería demasiado grande. La media de todos estos datos es de 497 milisegundos, está muy cerca de lo que debería ser la velocidad transferencia configurada.

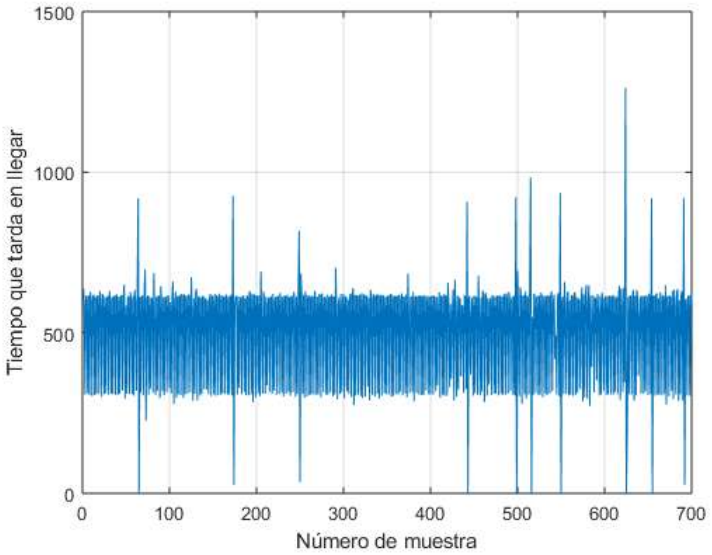


Figura 7.20: Gráfica del tiempo que tarda el dato en llegar por muestra

7.8.3. Colas para el envió de información entre tareas

Se usaron colas para el envió de información entre la interrupción de comunicación con las tareas de control. Para crear las colas, se selecciona el tipo de variable que se necesiten, en nuestro caso double. El tamaño de la cola, será de dos elementos, solo se necesita guardar el dato anterior, en caso de que llegara un dato adelantado, como se observó en la parte de comunicación. El tiempo que espera la cola el dato, es vital para que el RTOS no pierda el control del tiempo. La tarea de control tarda solo microsegundos en ejecutarse, nos dan un amplio tiempo de espera, se seleccionó 450 milisegundos, dando así el tiempo necesario para la llegada de los datos con un retraso mayor y guardando aquellos que llegaron antes. Se puede observar un diagrama de la espera del dato de en la figura 7.21.

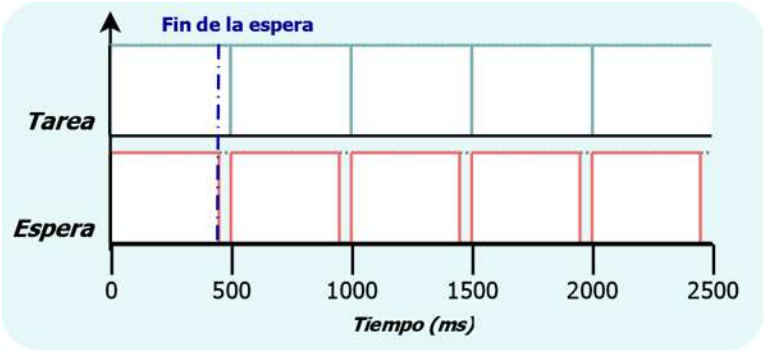


Figura 7.21: Diagrama de espera de la cola

7.8.4. Semáforo para la sincronización

Debido a que la comunicación puede iniciar en cualquier momento, es necesario un semáforo el cual mantiene detenidas las tareas, hasta que el primer dato del simulador en tiempo real llegue. Una vez que esto pase debe de iniciar las tareas, con esto se asegurará que el simulador en tiempo real y el sistema embebido empiezan en el mismo instante sin un retraso. Aunque este método puede que funcione solo en un inicio, debido a que las fluctuaciones en el envío de datos puede que hagan que se pierda esta sincronización.

7.9. Diagrama del RTOS

En la figura 7.22, se puede observar el diagrama de lo que sería el RTOS, donde se observa la distribución de los núcleos. En el núcleo uno, el cuál es el principal, se ejecutará la interrupción de comunicación junto con el procesado de los datos de la misma y el envió de datos. Se observa cómo las señales de interrupción alternan en el tiempo, esto debido a la forma en que el simulador en tiempo real envia los datos. En el núcleo dos, se realizan las dos tareas para los controles, estos por el contrario de las interrupciones de comunicación, si se debe de ejecutar periódicamente cada 500 milisegundos sin demora alguna.

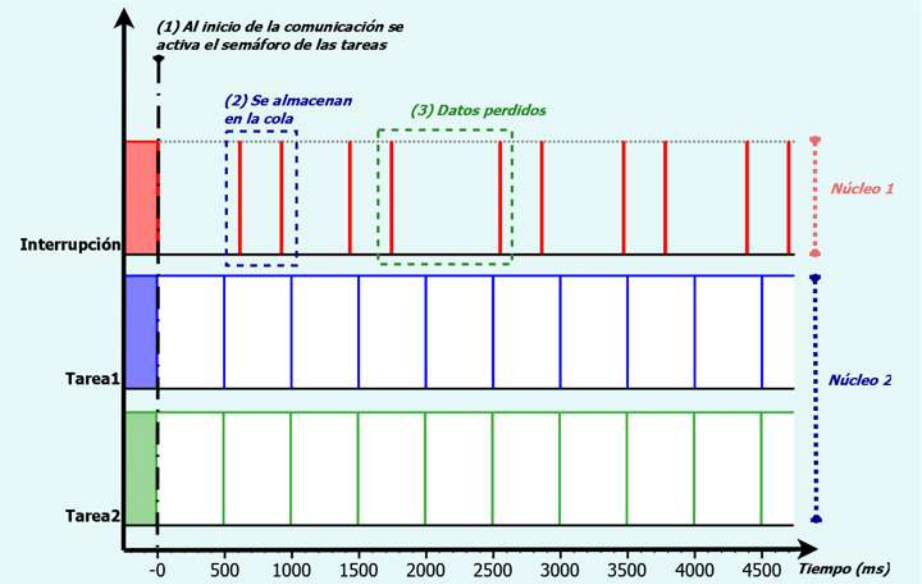


Figura 7.22: Diagrama del RTOS

En el punto (1) del diagrama, se observa que antes de que llegue el primer dato no se realiza ninguna acción por parte de las tareas, es hasta que inicia la comunicación, que inician las tareas, esto es gracias al semáforo el cual sincroniza estas. En el punto (2), se ve el caso en que dos datos llegan en un mismo periodo de ejecución de tarea, esto no es ningún problema, debido a que este dato extra que llegó, se almacena en la cola para que la siguiente periodo de ejecución de tarea lo utilice. En el punto (3) se puede observar el peor de los casos, este es ocasionado por un dato que tarda demasiado tiempo en llegar, el RTOS no tiene ninguna forma de solucionar esto, aunque las afectaciones son mínimas debido a que solo se perdería uno de los paquetes y el envió de datos no se ve afectado por

lo tanto el control tampoco.

7.9.1. Prueba del RTOS

Una vez terminado el codigo, quedo de la manera que se muestra en el anexo A.B.

En el sistema operativo en tiempo real se cargo el mismo modelo de prueba de comunicación que incluye el OPAL-RT solo se modificó, para dos constantes. Con esto el sistema embebido recibe las referencias de amplitud y frecuencia para el control en lazo cerrado. Aunque no se conectó la sección de potencia, se observó la salida PWM, es la salida del controlador. Se utiliza una resistencia variable como divisor de voltaje para simular la retroalimentación del sistema.

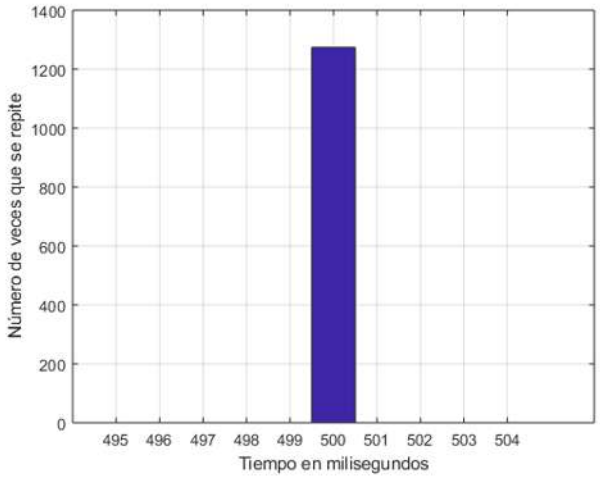


Figura 7.23: Histograma del tiempo de simulación del RTOS

El resultado es el esperado, se hicieron pruebas para verificar si se encuentra dentro de los parámetros de tiempo de simulación de 500 milisegundo. Se tomaron 1000 muestras del tiempo que tarda en ejecutarse la tarea, los resultado se muestra en el histograma de la figura 7.23, donde se aprecia claramente que la programación que se realiza es adecuada para conservar el tiempo de ejecución del RTOS. La pérdida de paquetes apenas se puede percibir por lo que el control no se ve afectado.

7.10. Diseño del controlador de voltaje generado

Como generador se utiliza una máquina síncrona, la cual está acoplada mecánicamente al motor de CD, del cual controlamos la velocidad que es proporcional a la frecuencia del voltaje generador.

7.10.1. Topología del control

La amplitud se controlará por medio del rotor de la máquina síncrona, se utiliza una topología similar a la del propulsor de CD, como se muestra en la figura 7.24. Donde se ve cómo el IGBT va conectado al rotor, mientras que se conecta en estrella los devanados del estator.

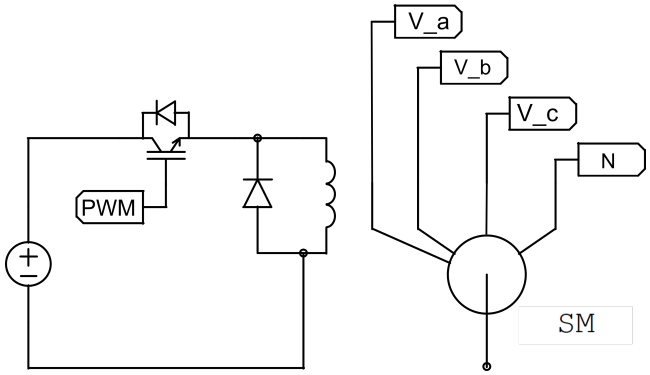


Figura 7.24: Esquema de control de voltaje generado

Se colocan las mismas protecciones para picos de corriente, con diferente resistencia shunt al del devanado de armadura y diferente pico de corriente ya que en el rotor tiene un pico de corriente menor.

7.10.2. Medición de voltaje generado

La medición del voltaje generado se realizó como se muestra en la figura 7.25. Por medio de un rectificador trifásico, que alimenta a un divisor de voltaje, formado por resistencias de gran magnitud para que el consumo de corriente sea el menor posible y no afecte el voltaje generado. También se seleccionó de manera adecuada el voltaje de salida del divisor, esto para que no supere el voltaje máximo que puede entrar al ADC del

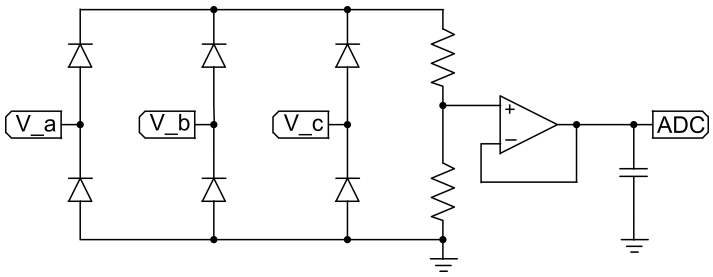


Figura 7.25: Esquema de medidor de voltaje generado

microcontrolador. Después del divisor, la señal pasa por un seguidor de voltaje y por último es filtrado por un capacitor para disminuir el rizado.

7.10.3. Prueba del controlador de voltaje generado

Para esta prueba se cargó el código que se encuentra en el anexo A.C, se envía el lazo de retroalimentación al simulador en tiempo real, este calcula la acción de control, que es enviada de nuevo al sistema embebido, para que este la aplique por medio del PWM a la planta. El control está cargado en el simulador en tiempo real, el sistema embebido es solo un vínculo entre el sistema físico y el simulador en tiempo real. Esta técnica de simulación se le conoce como Rapid Control.

El modelo que se cargó en el simulador en tiempo real para esta prueba fue el que se observa en la figura 7.26. Para el maestro, en el cuál se ven los bloques de RT-LAB necesarios para la comunicación, así como los de control, que en este caso se trata de un control PI, al cual se le sintonizaron las ganancias. Fue necesario usar bloques para poder cambiar el tiempo de transferencia de algunas señales, ya que la simulación tiene un menor tiempo de muestreo que el del modelo del control, esto afecta la manera en que se recibe la información, pero al agregar los bloques de transferencia esto se soluciona. En cuanto al bloque de la consola, se cargó el modelo que se muestra en la figura 7.27, en el cual se tienen varia variables de control, las cuales sirven para manejar la referencia del control o seleccionar entre en lazo abierto y cerrado.

Se cuenta con dos monitores uno para el voltaje generado y en el otro para la velocidad del motor de cd aunque esta no era necesaria para el control de voltaje generado.

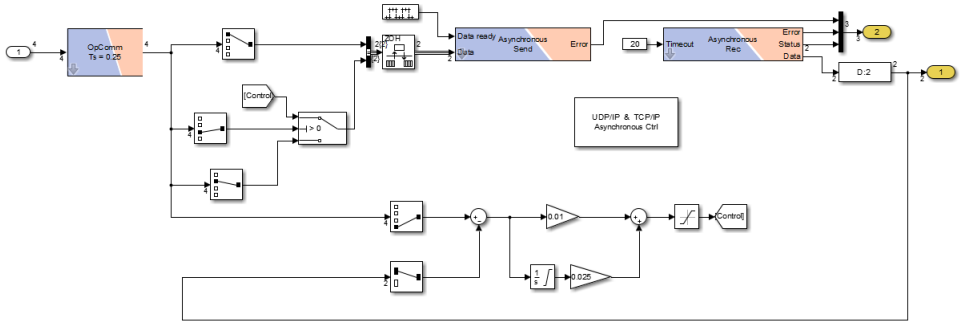


Figura 7.26: Bloque maestro de la prueba del control de voltaje generado

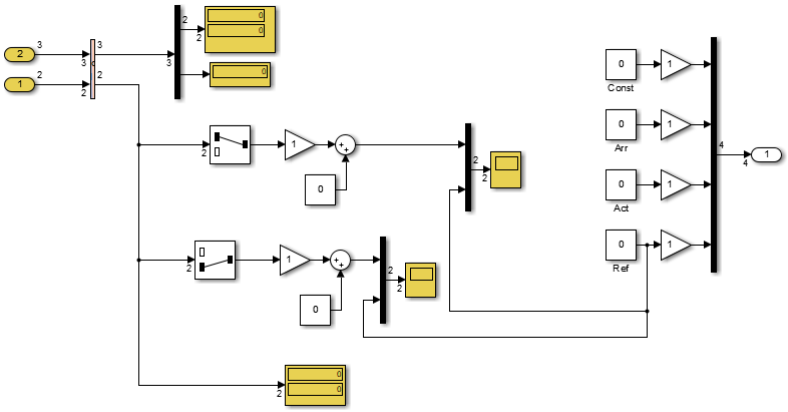


Figura 7.27: Bloque de consola de la prueba del control de voltaje generado

Para la prueba de este control se dejó constante la velocidad para hacer solo el control de voltaje, los resultados se muestran en la figura 7.28, donde primero se arrancó desde cero, a un voltaje fijo. Se observa que el sistema es bastante dinámico dado su poco tiempo de reacción.

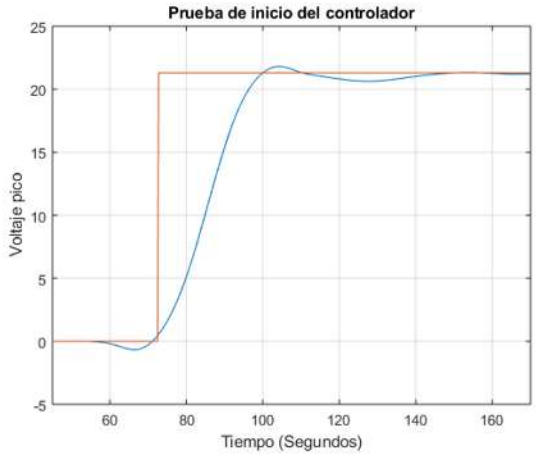


Figura 7.28: Prueba de arranque del control

Una vez comprobado el arranque del controlador, se pasó a probar los cambios de referencia que se observan en la figura 7.29. Se ve cómo el controlador sigue la referencia sin ningún problema.

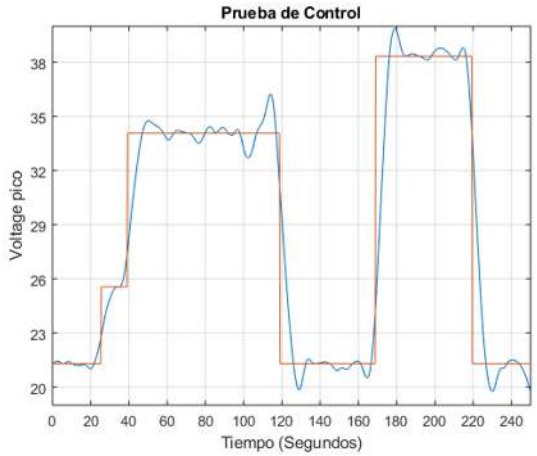


Figura 7.29: Prueba del control con diferentes referencias

En la figura7.29 se ve ciertos sobre impulsos en los cambios, también se observa que incluso alcanzando la referencia, se presentaba una cierta oscilación en el voltaje, estas variaciones también se notaron en el medidor de voltaje analógico, se deben a que la ganancia integral debe de ser pequeña.

Con estas pruebas se comprueba el funcionamiento del control de voltaje generado.

7.11. Construcción del sistema embebido

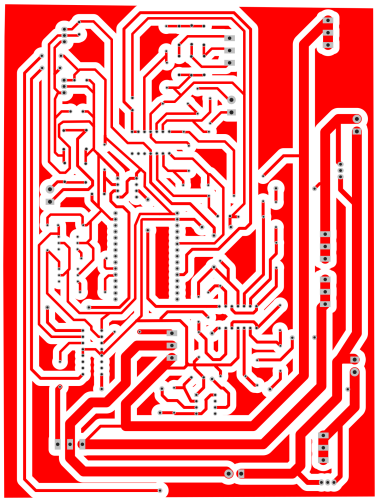


Figura 7.30: PCB

Una vez comprobado el funcionamiento de cada parte individual, se procedió a la realización del PCB para el sistema embebido, el cual dio como resultado el mostrado en la figura 7.30, donde se incluye una fuente lineal para alimentar el circuito, más unos reguladores para obtener los voltajes de 20, 15 y 5 voltios, que son necesarios para el correcto funcionamiento del sistema, en este diseño de PCB, en una placa de circuito impreso con un tamaño de 30 por 15 centímetros. Aunque se realizó con éxito el PCB, el gran tamaño era una desventaja, ya que es más difícil de realizar uno de ese tamaño.

7.12. Cambio de topología

Realizando las primera pruebas con el PCB diseñado, se noto un grave error en la topología de la etapa potencia que se había propuesto. Este error es impedir conmutar solo la armadura o el rotor, esto hacía imposible controlarlos independientemente. El cambio en la topología que se propuso se muestra en la figura 7.31.

En esta nueva topología, el devanado de armadura de la máquina de CD o el

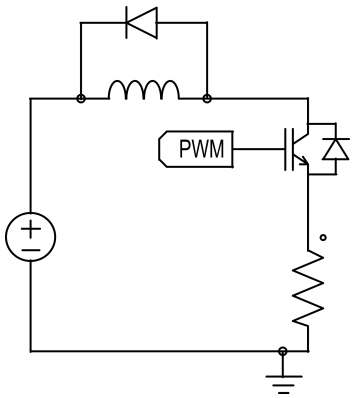


Figura 7.31: Nueva topología de potencia

rotor de la máquina síncrona, representados ambos con un inductor, ahora se encuentra conectado al colector del IGBT. Realizando pruebas a esta nueva topología, que sí funcionó correctamente. Se realizaron modificaciones al PCB, agregando una nueva placa con la nueva topología de potencia.

7.12.1. Cambios en el sensor de voltaje generado

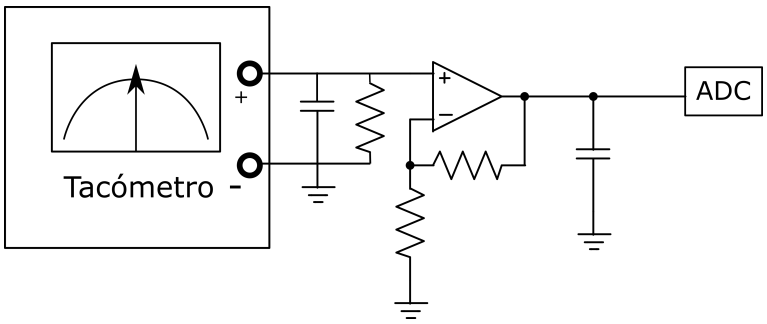


Figura 7.32: Modificación al sensor de voltaje

Con los cambios en la parte de potencia, se procedió a probar el sistema embebido, pero este no funcionó, debido a que el tacómetro se saturaba con las primeras conmutaciones de la armadura. Por lo tanto se agregó un filtro a la salida del tacómetro como se muestra en la figura 7.32. Esto fue suficiente para quitar ese ruido inicial, pero ocasiona una disminución en la velocidad de respuesta del sensor de voltaje, esto afectó la reacción del control a

variaciones de velocidad de la máquina de CD.

7.12.2. Cambios en la alimentación

En el diseño original del PCB se propone que la alimentación de todos los elementos proviene de una sola fuente lineal, si es así una falla puede causar daños en todo el circuito. Otro factor es el ruido que introduce al circuito la conmutación de los IGBTs. Así que se decidió cambiar la fuente de alimentación y poner una fuente para la parte de baja potencia y otra para la de mediana potencia, ambas lineales, reduciendo así el ruido y el riesgo de los componentes de baja potencia.

7.13. Pruebas finales

Una vez hechos los cambios, se realizaron las pruebas definitivas del sistema embebido. El modelo cargado en el OPAL-RT es el que se muestra en la figura 7.33 para el maestro y en la figura 7.34 para la consola. En el maestro, se colocaron más bloques para adaptar el tiempo de simulación, también se colocaron dos controladores PI uno para la frecuencia y otro para el voltaje.

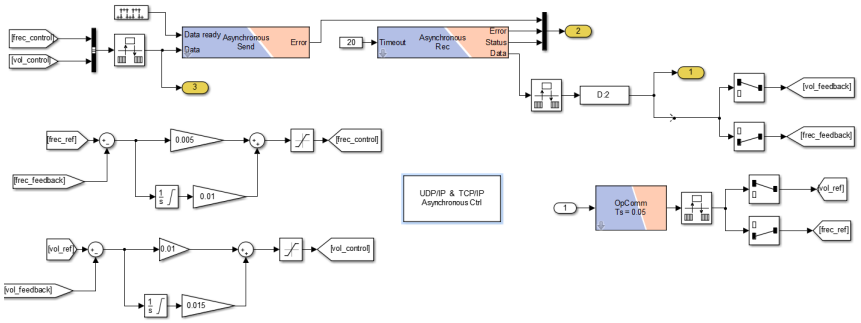


Figura 7.33: Bloque maestro de la prueba final

En cuanto el bloque de consola, ahora se coloca una referencia para la amplitud de voltaje como para la frecuencia. Se añadieron dos monitores con los cuáles se puede ver la acción de control, esta es necesaria para verificar que el control funciona de manera correcta.

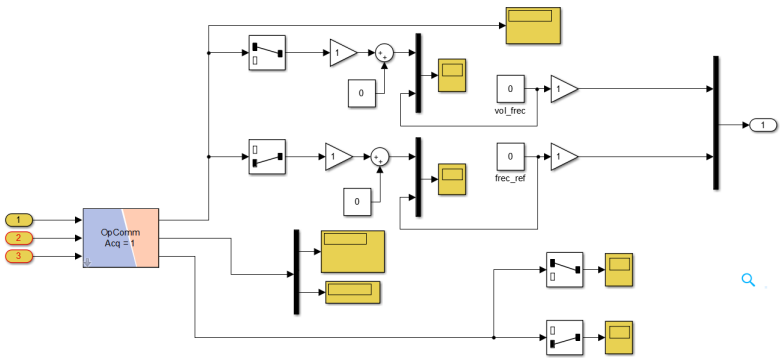


Figura 7.34: Bloque de consola de la prueba final

El modelo de consola cuenta con bloques de ganancia y suma en la entrada de los monitores, esto por si se quiere cambiar la escala en la que se está midiendo, así como unos bloques dedicados a mostrar los errores en la comunicación si es que suceden.

Una vez cargado el programa de Rapid Control A.C, en el sistema embebido, se procedió a probar, para así validar el funcionamiento, los resultados obtenidos se muestran en la figura 7.35, donde se aprecia el experimento con el cual se validó el funcionamiento.

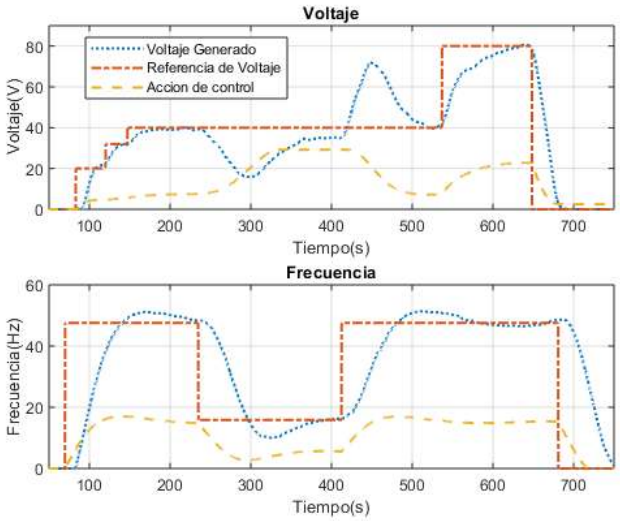


Figura 7.35: Prueba final

Esta prueba fue hecha para verificar los aspectos del controlador, desde el arranque

hasta generar voltaje a una cierta frecuencia y amplitud, pasando por varios cambios de referencias y finalmente el proceso de apagado del generador. En las figuras siguientes se muestra la referencia, la acción de control y la variable física medida.

Ahora no es necesario ninguna etapa de arranque, debido a las mejoras en la conmutación los IGBTs, son capaces de soportar los picos de corriente al arranque.

En la figura 7.36, inicia el control de frecuencia, debido a que este enciende el motor de CD, una vez que se ha iniciado este, se continúa con la generación de voltaje necesario.

Para la primera parte, se hicieron varios cambios de referencia de voltaje, se ve como estos no afectan de ninguna manera la frecuencia. El control de voltaje es capaz de seguir las referencias sin ningún error de estado estable, esto debido a las constantes integrales con las que cuentan los controladores.

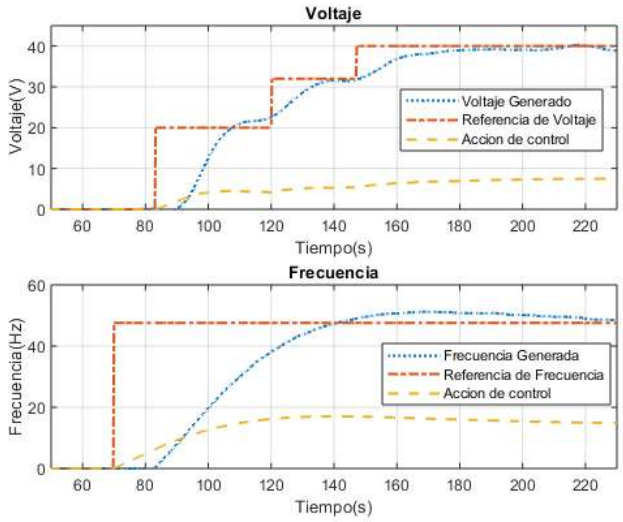


Figura 7.36: Arranque de la prueba final

El control de frecuencia tiene un ligero sobre impulso antes de establecerse en la referencia, esto se puede eliminar cambiando las constantes del controlador.

Se cambio la referencia de la frecuencia a un más baja, mientras que la referencia de amplitud de voltaje seguía en el mismo valor. Se observa en la figura 7.37, como la acción de control de la frecuencia baja inmediatamente, causando que la frecuencia baje hasta la referencia. La amplitud de voltaje generado baja de una manera rápida, la acción de control

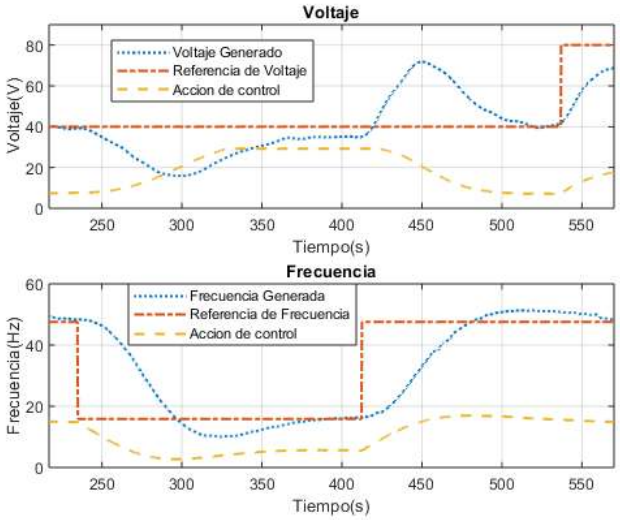


Figura 7.37: Cambio de frecuencia

de la amplitud de voltaje sube, aunque no logra evitar que la caída de la amplitud de voltaje.

Segundos después el voltaje empieza a subir, pero tiene un error en estado estable, debido a que la acción de control llega a su máximo.

La saturación se colocó para no superar la corriente máxima que soporta el devanado del rotor de la máquina síncrona.

De nuevo se cambia la referencia de la frecuencia a una mayor, lo que hace que aumente la frecuencia para alcanzar la referencia. Esto causa que se eleve el voltaje generado de una manera rápida, debido a esto la acción de control de amplitud del voltaje generado la cual está al máximo, esta empieza a bajar.

Finalmente para dejar de generar, se colocó la referencia en cero de ambos controladores como se muestra en la figura 7.38, el primer controlador que se detuvo fue el de voltaje y cuando este llegó a cero, se detuvo el controlador de frecuencia.

7.14. Conclusiones

Con las pruebas mostradas se concluye, que el sistema embebido funciona, ya que se pudo controlar por medio de un control simulado la planta, sin sin ser necesario el sistema embebido realice el control, sí no sólo siendo el vínculo entre el control y la planta.

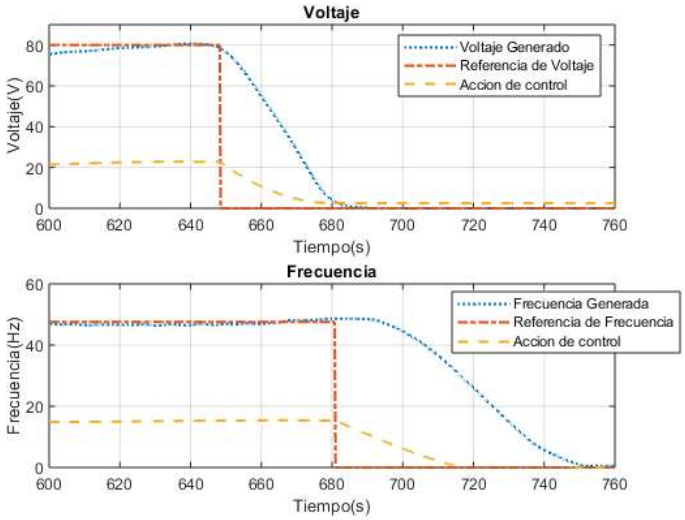


Figura 7.38: Apagado del generador

Se muestra cómo es que la saturación funcionan de manera correcta protegiendo así el rotor de la máquina síncrona de una corriente por parte del controlador.

Los controles fueron probados al seguir los cambios de referencias, inclusive cuando había perturbaciones en el sistema, comprobando que estos son válidos para el control del generador.

Capítulo 8

Conclusiones

8.1. Conclusiones generales

Con la realización de esta tesis se concluye, que es posible realizar módulos basados sistema embebidos para la realización de simulaciones en tiempo real.

Estos son más baratos y especializados que los vendidos por el fabricante.

Las ventajas de este tipo de módulos son una configuración personalizada, la cual se ajusta a las necesidades del proyecto, facilidad de uso, protecciones a la medida para la planta, menos fallos en la comunicación entre el simulador en tiempo real y reemplazos fáciles de conseguir además de económicos.

Aunque los módulos que ofrece el fabricante como principal ventaja tienen que la comunicación es mucho más rápida y con mayor seguridad.

Si nos enfocamos en este caso en particular, para el control del generador con el módulo del fabricante, el simulador en tiempo real tendría que generar varias señales de muy alta frecuencia a la vez, para esto se necesitaría un tiempo de simulación muy pequeño, esto no lo sería capaz de hacer el sistema embebido que se desarrolló, esto porque los IGBTS se deben de conmutar a altas frecuencias, estas señales se generan mediante cálculos por el simulador en tiempo real. Sin embargo al contar el microcontrolador con módulos PWM que generan las señales, no es necesario que el tiempo de simulación sea tan pequeño, ya que la señal se genera de manera automática sin estar la procesando tanta información, esto resulta ser una ventaja para el sistema embebido desarrollado.

La topología de potencia que se utilice es de suma importancia, esta puede afectar todo el funcionamiento, ya sea porque la señal de conmutación o la de los sensores se ve

contaminada por la conmutación, en ambos casos se le encontró solución cambiando la topología.

El uso de fuentes independientes para este tipo de sistemas, es de vital importancia ya que dan un aislamiento total entre los circuito de control y de potencia, haciendo que el ruido no se esparza en los sensores.

El protocolo de transporte UDP en este caso si lleva detención a fallos, la cual está establecida en los bloques del simulador tiempo real por defecto, aunque es uno muy sencillo, es posible que modificandolo admita la detección de problemas específicos, según el sistema embebido que se desarrolle, se podría hacer que el simulador en tiempo real tome acciones sobre estos errores o simplemente saber de que estos sucedieron.

Si bien, se decidió usar el tamaño double, se podrían usar tamaños de variable más pequeños, esto haría más rápido el proceso de recepción y envío de datos, sería mucho más sencillo de programar, aunque habría pérdida de información, el tamaño más adecuado es signed int, este permitiría usar el tipo de variable tanto el conversor analógico digital como el PWM.

La señales físicas deben ser filtradas, la instrumentación del sensor incrementar el tiempo de respuesta del sensor, debido a los capacitores o filtros, lo que tiene como consecuencia que el control se vuelve lento, una vez alcanzada la referencia el controlador tendrá oscilaciones, esto no es bueno si se desea tener un control preciso.

Al realizarse un modelo para el sistema, se debe tomar en cuenta las ganancias, la dinámica de los sensores y la instrumentación, si no se toman en cuenta los experimentos que se realicen con este módulo, tendrán errores.

La selección de los materiales que se utilizaron en el módulo, fueron cambiando. Esto hizo costoso y tardado el proceso, lo que se recomendable al hacer proyecto de tal tamaño, es hacer todas las simulaciones necesarias, con los modelos reales de los componentes que se piensan utilizar, si se toman los ideales no se observan todos los problemas que se tienen cuando se utilizan.

Ya que se tienen los cálculos adecuados de los valores nominales de los componentes, los reales deben de tener un valor mayor, esto para que el circuito tenga un mejor funcionamiento ante los imprevistos y su vida útil sea mucho mayor.

Si los componentes no conmutan a la frecuencia deseada, se pueden utilizar unos que se aproximen, como lo fue en el caso de los optoacopladores usados, pero se tiene que

corregir la señal con algún driver extra, esto ocasiona pérdida de la resolución de la señal. Por eso aunque conseguir los componentes lleve más tiempo y gasto económico, siempre es mejor hacer esto ya que la funcionalidad será bastante buena en comparación al circuito que se adaptó.

Un RTOS es de bastante utilidad para sistemas embebidos de este tipo, debido a que de otra manera el desarrollo del código hubiera tardado mucho más tiempo, este sería mucho más largo y engorroso para alguien que lo quisiera modificarlo según sus necesidades. La API del RTOS usado tiene muchas funciones para sincronizar y ejecutar en el tiempo debido las tareas. Otra de las cosas que nos permite el RTOS, es la gestión de los recursos de una manera sencilla, separar las tareas, asignarles los recursos necesarios para ellas.

Las ventajas de usar un RTOS a diferencia de usar solo interrupciones, es que este podemos administrar de manera sencilla las tareas que podemos hacer, aunque las interrupciones son de vital importancia para la comunicación con el simulador en tiempo real.

El desarrollo de este sistema embebido a servido para comprobar los numerosos conocimientos adquiridos durante la ingeniería, que tan capaz soy de hacer un proyecto donde todo lo he desarrollado desde cero. Adquirí nuevos conocimientos, ya que con los que se tenía no eran suficientes para concluir los objetivos del proyecto. He aprendido que, lo que veo en la teoría es diferente al aplicarlo. Que nunca se podrán tener en cuenta todos los aspectos que pueden influir y esto puede causar varios problemas, a los cuáles también se les tiene que buscar una solución, primero teorizando sobre el cómo es que sucedió y después investigando cómo solucionarlo.

8.2. Trabajo a futuro

Se planea duplicar el sistema embebido realizado, para implementar la simulación de una microred donde estos dos módulos simularán ser de generadores convencionales o generación eólica, conectados a un sistema de generación solar, para mediante experimentos conocer cómo es que estos tipos de generación se comporta.

Más adelante se podrían construir diversos módulos especializados para diferentes

tareas, los cuáles se pueden conectar al mismo simulador en tiempo real. Estos módulos no necesitan estar en la misma red del simulador, para medir parámetros o trabajar con datos de plantas físicas reales lejanas, que se deseen analizar. Esto abre un mundo de posibilidades, ya que los módulos serían capaces de hacer bastantes cosas, debido a la flexibilidad y a su bajo coste, se podría desarrollar todo un laboratorio con estos sistemas embebidos.

Apéndice A

Programas utilizados

A.A. Programa Control Digital

```
const int Sal_cntrl = 3;
const int Sal_cntrl_vol = 11;

const int Ent_cntrl = A2;
const int Ent_cntrl_vol = A5;

String Ref_Serial;
String Ref_Serial_vol;

long Ref = 127;
float Rtro = 0;
float Accion_cntrl=0;
float Error=0;
float Dif=0;
float Ref_ant=0;
float Error_Acm = 0;
float Ref_vol = 127;
float Rtro_vol = 0;
```

```
float Accion_cntrl_vol=0;
float Error_vol=0;
float Dif_vol=0;
float Ref_ant_vol=0;
float Error_Acm_vol = 0;
float Ki=0.4;
float Kp=0.4;
float Kd=0;
float Ki_vol=0;
float Kp_vol=0.1;
float Kd_vol=0;

void encendido(void);

void setup() {

 Serial.begin(9600);
 TCCR2B = 0b00000001;
 pinMode(Sal_cntrl,OUTPUT);
 pinMode(Sal_cntrl_vol,OUTPUT);
 analogWrite(Sal_cntrl_vol,0);
 analogWrite(Sal_cntrl,0);

 encendido();
}
void loop()
{

 Rtro = analogRead(Ent_cntrl);
 Rtro = Rtro/1023;
 Rtro = Rtro*532*5;
 Rtro = (Rtro*255)/2500;
```

```
 Error = Ref-Rtro;

Dif = Rtro-Ref_ant;

 Error_Acm = Error + Error_Acm;
 if(Error_Acm>500) {Error_Acm=500;}
 if(Error_Acm<-500) {Error_Acm=-500;}

 Accion_cntrl = Kp*Error + Ki*Error_Acm + Kd*Dif + 0.7*Ref;

 if(Accion_cntrl>255) {Accion_cntrl=255;}
 if(Accion_cntrl<0) {Accion_cntrl=0;}

 analogWrite(Sal_cntrl,(160*Accion_cntrl/255) + 20);

 Ref_ant=Rtro;

 delay(500);
}

void encendido(void){

 while(Serial.available()< 1);

 Ref_Serial = Serial.readString();
 Ref = Ref_Serial.toInt();
 Ref = Ref*255;
 Ref = Ref/2500;
 Serial.println(Ref)

 for(int i = 0; i<Ref;i++){
```

```
 analogWrite(Sal_cntrl,(160*i/255) + 20);
 delay(650);

 }
 analogWrite(Sal_cntrl,(160*Ref/255) + 20);
 loop();
}
```

A.B. Programa Control con RTOS para Hardware in the loop

```
#include "WiFi.h"
#include "AsyncUDP.h"

const char* ssid     = "WLPosgradoFIE";
const char* password = "dyZi9VeZ6AW7fL8";
int32_t chanel = 0;
const uint8_t bssid[] = {0xdc, 0xa4, 0xca, 0xbb, 0xa0, 0xc4};

byte recieved_data_vector[24];
byte send_data_vector[24];
byte * pointer[2];
double * recieved_data_1 = (double*) &recieved_data_vector[8];
double * recieved_data_2 = (double*) &recieved_data_vector[16];
double send_data_1 = 123;
double send_data_2 = 345;

QueueHandle_t queue_1;
QueueHandle_t queue_2;
int queueSize = 10;
```

```
const int armature_winding = 19;
const int rotor = 21;
const int tacometer = 32;
const int voltaje_generate = 35;
const int freq = 30000;
const int Channel_winding = 0;
const int Channel_rotor = 1;
const int resolution = 11;
AsyncUDP udp;
void setup() {

Serial.begin(115200);

Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password, chanel, bssid);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
 }

Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
Serial.println((int)&send_data_1);
Serial.println((int)&send_data_2);
Serial.println((int)&recieved_data_vector[8]);
Serial.println((int)&recieved_data_vector[23]);

queue_1 = xQueueCreate( queueSize, sizeof( double ) );
 if(queue_1 == NULL){
```

```
    Serial.println("Error creating the queue");
  }
  queue_2 = xQueueCreate( queueSize, sizeof( double ) );
  if(queue_2 == NULL){
    Serial.println("Error creating the queue");
  }
    ledcSetup(Channel_winding, freq, resolution);
    ledcAttachPin(armature_winding, Channel_winding);
    ledcSetup(Channel_rotor, freq, resolution);
    ledcAttachPin(rotor, Channel_rotor);
    ledcWrite(Channel_winding, 0);
    ledcWrite(Channel_rotor, 0);

xTaskCreate(
                    servidor,
                    "TaskOne",
                    10000,
                    NULL,
                    2,
                    NULL);

xTaskCreate(
                    control_1,
                    "TaskTwo",
                    10000,
                    NULL,
                    2,
                    NULL);

xTaskCreate(
                    control_2,
                    "TaskTree",
```

```
                    10000,
                    NULL,
                    2,
                    NULL);
}

void loop() {
delay(1000);
}
void servidor( void * parameter )
{
 if(udp.listen(50000)) {
 udp.onPacket([](AsyncUDPPacket packet) {
    for(int i=0; i<packet.length(); i++){
     recieved_data_vector[i] = packet.data()[i];
    }
    xQueueSend(queue_1, recieved_data_1 ,portMAX_DELAY);
    xQueueSend(queue_2, recieved_data_2 ,portMAX_DELAY);

    pointer[0] = &recieved_data_vector[8];
    pointer[1] = (byte*)(&send_data_2);
    for (int i = 0; i <=16; i++){
      *pointer[0] = *pointer[1];
      pointer[0]++;
      pointer[1]++;
    }
    packet.write( recieved_data_vector, packet.length());
 });
}
  vTaskDelete( NULL );
}
void control_1( void * parameter)
```

```
{
    TickType_t xLastWakeTime;
    const TickType_t xFrequency = 500;
    xLastWakeTime = xTaskGetTickCount();

    double element;
    for(;;){
      vTaskDelayUntil( &xLastWakeTime, xFrequency );
      xQueueReceive(queue_1, &element, portMAX_DELAY);
      send_data_1 = (double)analogRead(tacometer);
      ledcWrite(Channel_winding, element);
      Serial.println("Lo que recibe tarea 1");
      Serial.println((int)element);
    }
 }
void control_2( void * parameter)
{
    TickType_t xLastWakeTime;
    const TickType_t xFrequency = 500;
    xLastWakeTime = xTaskGetTickCount();

    double element;
    for(;;){
      vTaskDelayUntil( &xLastWakeTime, xFrequency );
      xQueueReceive(queue_2, &element, portMAX_DELAY);
      send_data_2 = (double)analogRead(voltaje_generate);
      ledcWrite(Channel_rotor, element);
      Serial.println("Lo que recibe tarea 2");
      Serial.println((int)element);
    }

}
```

A.C. Programa Control con RTOS para Rapid Control

```
#include "WiFi.h"
#include "AsyncUDP.h"

const char* ssid     = "WLPosgradoFIE";
const char* password = "dyZi9VeZ6AW7fL8";
int32_t chanel = 0;
const uint8_t bssid[] = {0xdc, 0xa4, 0xca, 0xbb, 0xa0, 0xc4};

byte recieved_data_vector[24];
byte send_data_vector[24];
byte * pointer[2];
double * recieved_data_1 = (double*) &recieved_data_vector[8];
double * recieved_data_2 = (double*) &recieved_data_vector[16];
double send_data_1 = 123;
double send_data_2 = 345;

QueueHandle_t queue_1;
QueueHandle_t queue_2;
int queueSize = 10;

const int armature_winding = 19;
const int rotor = 21; const int tacometer = 33;
const int voltaje_generate = 35;
const int freq = 30000;
const int Channel_winding = 0;
const int Channel_rotor = 1;
const int resolution = 11;

AsyncUDP udp;
void setup() {
```

```
 WiFi.begin(ssid, password, chanel, bssid);
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }

queue_1 = xQueueCreate( queueSize, sizeof( double ) );
 if(queue_1 == NULL){
 }
 queue_2 = xQueueCreate( queueSize, sizeof( double ) );
 if(queue_2 == NULL){
 }
   ledcSetup(Channel_winding, freq, resolution);
   ledcAttachPin(armature_winding, Channel_winding);
   ledcSetup(Channel_rotor, freq, resolution);
   ledcAttachPin(rotor, Channel_rotor);
   ledcWrite(Channel_winding, 0);
   ledcWrite(Channel_rotor, 0);

xTaskCreate(
                    servidor,
                    "TaskOne",
                    10000,
                    NULL,
                    2,
                    NULL);

xTaskCreate(
                    control_1,
                    "TaskTwo",
                    10000,
                    NULL,
```

```
                    2,
                    NULL);

xTaskCreate(
                    control_2,
                    "TaskTree",
                    10000,
                    NULL,
                    2,
                    NULL);
}

void loop() {
delay(1000);
}
void servidor( void * parameter )
{
 if(udp.listen(50000)) {
 udp.onPacket([](AsyncUDPPacket packet) {
   for(int i=0; i<packet.length(); i++){
    recieved_data_vector[i] = packet.data()[i];
   }
   xQueueSend(queue_1, recieved_data_1 ,portMAX_DELAY);
   xQueueSend(queue_2, recieved_data_2 ,portMAX_DELAY);

   pointer[0] = &recieved_data_vector[8];
   pointer[1] = (byte*)(&send_data_2);
   for (int i = 0; i <=16; i++){
     *pointer[0] = *pointer[1];
     pointer[0]++;
     pointer[1]++;
   }
```

```
    packet.write( recieved_data_vector, packet.length());
  });
}
  vTaskDelete( NULL );
}
void control_1( void * parameter)
{
  while(recieved_data_1 == 0);
   for(int i = 0; i<= *recieved_data_1; i++){
     ledcWrite(Channel_winding, i);
     delay(130);
  }

   TickType_t xLastWakeTime;
   const TickType_t xFrequency = 1000;
   xLastWakeTime = xTaskGetTickCount();

   double element;
   for(;;){
     vTaskDelayUntil( &xLastWakeTime, xFrequency );
     send_data_2 = (double)analogRead(tacometer);
     ledcWrite(Channel_rotor,*recieved_data_1 );

   }
  // vTaskDelete( NULL );
}
void control_2( void * parameter)
{
   TickType_t xLastWakeTime;
   const TickType_t xFrequency = 100;
   xLastWakeTime = xTaskGetTickCount();
   for(;;){
```

```
        vTaskDelayUntil( &xLastWakeTime, xFrequency );
        send_data_2 = (double)analogRead(voltaje_generate);
        ledcWrite(Channel_rotor,*recieved_data_2 );
    }

}
```

Apéndice B

Hojas de datos

B.A. IGBT SGTW90V60F

life.augmented

STGFW80V60F, STGW80V60F, STGWT80V60F

Trench gate field-stop IGBT, V series
600 V, 80 A very high speed

Datasheet - production data

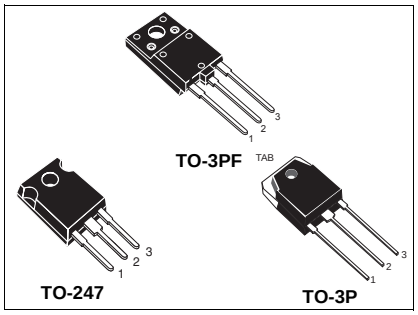
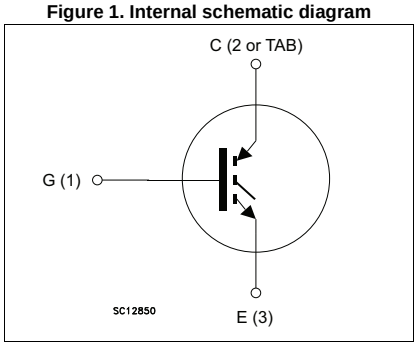


Figure 1. Internal schematic diagram



Features

- Maximum junction temperature: T_J = 175 °C
- Tail-less switching off
- $V_{CE(sat)}$ = 1.85 V (typ.) @ I_C = 80 A
- Tight parameters distribution
- Safe paralleling
- Low thermal resistance

Applications

- Photovoltaic inverters
- Uninterruptible power supply
- Welding
- Power factor correction
- Very high frequency converters

Description

This device is an IGBT developed using an advanced proprietary trench gate field stop structure. The device is part of the V series of IGBTs, which represent an optimum compromise between conduction and switching losses to maximize the efficiency of very high frequency converters. Furthermore, a positive $V_{CE(sat)}$ temperature coefficient and very tight parameter distribution result in safer paralleling operation.

Table 1. Device summary

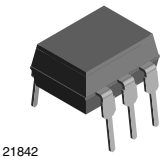
Order code	Marking	Package	Packaging
STGFW80V60F	GFW80V60F	TO-3PF	Tube
STGW80V60F	GW80V60F	TO-247	Tube
STGWT80V60F	GWT80V60F	TO-3P	Tube

B.B. Optoacoplador 4N25

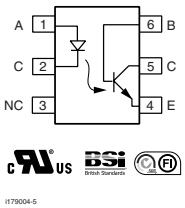
4N25, 4N26, 4N27, 4N28

Vishay Semiconductors

Optocoupler, Phototransistor Output, with Base Connection



21842



FEATURES

- Isolation test voltage 5000 V_{RMS}
- Interfaces with common logic families
- Input-output coupling capacitance < 0.5 pF
- Industry standard dual-in-line 6 pin package
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

Pb-free

e3

RoHS COMPLIANT

APPLICATIONS

- AC mains detection
- Reed relay driving
- Switch mode power supply feedback
- Telephone ring detection
- Logic ground isolation
- Logic coupling with high frequency noise rejection

AGENCY APPROVALS

- UL1577, file no. E52744
- BSI: EN 60065:2002, EN 60950:2000
- FIMKO: EN 60950, EN 60065, EN 60335

DESCRIPTION

The 4N25 family is an industry standard single channel phototransistor coupler. This family includes the 4N25, 4N26, 4N27, 4N28. Each optocoupler consists of gallium arsenide infrared LED and a silicon NPN phototransistor.

ORDER INFORMATION	
PART	**REMARKS**
4N25	CTR > 20 %, DIP-6
4N26	CTR > 20 %, DIP-6
4N27	CTR > 10 %, DIP-6
4N28	CTR > 10 %, DIP-6

ABSOLUTE MAXIMUM RATINGS (1)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
INPUT				
Reverse voltage		V_R	5	V
Forward current		I_F	60	mA
Surge current	t ≤ 10 µs	I_{FSM}	3	A
Power dissipation		P_{diss}	100	mW
OUTPUT				
Collector emitter breakdown voltage		V_{CEO}	70	V
Emitter base breakdown voltage		V_{EBO}	7	V
Collector current		I_C	50	mA
	t ≤ 1 ms	I_C	100	mA
Power dissipation		P_{diss}	150	mW

B.C. L293d

TEXAS INSTRUMENTS

L293, L293D

SLRS008D – SEPTEMBER 1986 – REVISED JANUARY 2016

L293x Quadruple Half-H Drivers

1 Features

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- High-Noise-Immunity Inputs
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

2 Applications

- Stepper Motor Drivers
- DC Motor Drivers
- Latching Relay Drivers

3 Description

The L293 and L293D devices are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.

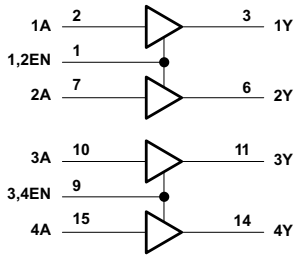
The L293 and L293D are characterized for operation from 0°C to 70°C.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
L293NE	PDIP (16)	19.80 mm × 6.35 mm
L293DNE	PDIP (16)	19.80 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

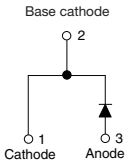
Logic Diagram



B.D. Diodo VS-ETH3007-M3

Hyperfast Rectifier, 30 A FRED Pt®

2L TO-220AC



FEATURES

- Hyper fast and soft recovery
- Low forward voltage drop
- 175 °C operating junction temperature
- Low leakage current
- True 2 pin package
- Designed and qualified according to JEDEC®-JESD 47
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

RoHS COMPLIANT HALOGEN FREE

PRIMARY CHARACTERISTICS	
$I_{F(AV)}$	30 A
V_R	650 V
V_F at I_F	1.4 V
t_{rr} typ.	33 ns
T_J max.	175 °C
Package	2L TO-220AC
Circuit configuration	Single

DESCRIPTION / APPLICATIONS

Ultra low V_F, soft-switching hyper fast rectifiers optimized for discontinuous (critical) mode (DCM) power factor correction (PFC).
The minimized conduction loss, optimized stored charge and low recovery current minimized the switching losses and reduce over dissipation in the switching element and snubbers.
The device is also intended for use as a freewheeling diode in power supplies and other power switching applications.

ABSOLUTE MAXIMUM RATINGS				
PARAMETER	SYMBOL	TEST CONDITIONS	VALUES	UNITS
Repetitive peak reverse voltage	V_{RRM}		650	V
Average rectified forward current	$I_{F(AV)}$	T_C = 120 °C	30	A
Non-repetitive peak surge current	I_{FSM}	T_J = 25 °C	210	
Operating junction and storage temperatures	T_J, T_{Stg}		-55 to +175	°C

ELECTRICAL SPECIFICATIONS (T_J = 25 °C unless otherwise specified)						
PARAMETER	SYMBOL	TEST CONDITIONS	MIN.	TYP.	MAX.	UNITS
Breakdown voltage, blocking voltage	V_{BR}, V_R	I_R = 100 µA	650	-	-	V
Forward voltage	V_F	I_F = 30 A	-	1.8	2.1	
		I_F = 30 A, T_J = 150 °C	-	1.4	1.6	
Reverse leakage current	I_R	V_R = V_R rated	-	0.02	30	µA
		T_J = 150 °C, V_R = V_R rated	-	50	300	
Junction capacitance	C_T	V_R = 650 V	-	22	-	pF
Series inductance	L_S	Measured lead to lead 5 mm from package body	-	8	-	nH

B.E. LM324

TEXAS INSTRUMENTS

LM124-N, LM224-N
LM2902-N, LM324-N

SNOSC16D – MARCH 2000 – REVISED JANUARY 2015

LMx24-N, LM2902-N Low-Power, Quad-Operational Amplifiers

1 Features

- Internally Frequency Compensated for Unity Gain
- Large DC Voltage Gain 100 dB
- Wide Bandwidth (Unity Gain) 1 MHz (Temperature Compensated)
- Wide Power Supply Range:
  - Single Supply 3 V to 32 V
  - or Dual Supplies ±1.5 V to ±16 V
- Very Low Supply Current Drain (700 µA) —Essentially Independent of Supply Voltage
- Low Input Biasing Current 45 nA (Temperature Compensated)
- Low Input Offset Voltage 2 mV and Offset Current: 5 nA
- Input Common-Mode Voltage Range Includes Ground
- Differential Input Voltage Range Equal to the Power Supply Voltage
- Large Output Voltage Swing 0 V to V^+ − 1.5 V
- **Advantages:**
  - Eliminates Need for Dual Supplies
  - Four Internally Compensated Op Amps in a Single Package
  - Allows Direct Sensing Near GND and V_{OUT} also Goes to GND
  - Compatible With All Forms of Logic
  - Power Drain Suitable for Battery Operation
  - In the Linear Mode the Input Common-Mode, Voltage Range Includes Ground and the Output Voltage
  - Can Swing to Ground, Even Though Operated from Only a Single Power Supply Voltage
  - Unity Gain Cross Frequency is Temperature Compensated
  - Input Bias Current is Also Temperature Compensated

2 Applications

- Transducer Amplifiers
- DC Gain Blocks
- Conventional Op Amp Circuits

3 Description

The LM124-N series consists of four independent, high-gain, internally frequency compensated operational amplifiers designed to operate from a single power supply over a wide range of voltages. Operation from split-power supplies is also possible and the low-power supply current drain is independent of the magnitude of the power supply voltage.

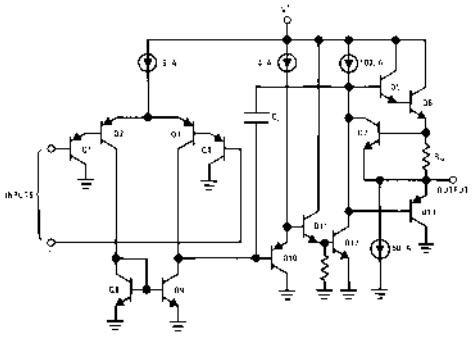
Application areas include transducer amplifiers, DC gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM124-N series can directly operate off of the standard 5-V power supply voltage which is used in digital systems and easily provides the required interface electronics without requiring the additional ±15 V power supplies.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM124-N	CDIP (14)	19.56 mm × 6.67 mm
LM224-N		
LM324-N	CDIP (14)	19.56 mm × 6.67 mm
	PDIP (14)	19.177 mm × 6.35 mm
	SOIC (14)	8.65 mm × 3.91 mm
	TSSOP (14)	5.00 mm × 4.40 mm
LM2902-N	PDIP (14)	19.177 mm × 6.35 mm
	SOIC (14)	8.65 mm × 3.91 mm
	TSSOP (14)	5.00 mm × 4.40 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Schematic Diagram



Referencias

[Alexey Melnikov] Alexey Melnikov, I. F. The websocket protocol. Https://tools.ietf.org/html/rfc6455.

[AWS] AWS. Freertos. Https://www.freertos.org.

[Burns A.03] Burns A., W. A. *Sistemas de Tiempo Real y Lenguajes de Programación*. 2003.

[cd] *CONTROL DIGITAL PID PARA SISTEMAS TÉRMICOS BASADO EN MICROCONTROLADOR PIC*.

[ESPRESSIFa] ESPRESSIF. Esp32. Https://www.espressif.com.

[ESPRESSIFb] ESPRESSIF. Esp32 datasheet.

[González] González, G. H. Página web personal. Www.guillehg.com.

[Hirzel] Hirzel, T. Pwm. Www.arduino.cc/en/Tutorial/PWMl.

[ipt]

[León81] León, P. J. P. Internet protocol, 1981. Rfc-es.org/rfc/rfc0791-es.txt.

[OPAL-RT] OPAL-RT. *Control of LAB-VOLT's 8540 dynamometer based on Mitsubishi's A-700 inverter*.

[OPAL-RT19] OPAL-RT. Real-time simulation real-time solutions opal-rt, 2019. Https://www.opal-rt.com/.

[osc18] *Real-time Hardware-in-the-loop Implementation for Power Systems Protection*, 2018.

[Postel80] Postel, J. Protocolo de datagramas de usuario, 1980. Www.rfc-es.org/rfc/rfc0768-es.txt.

[Rashid95] Rashid, M. H. *Eléctronica de potencia.* 2ª edón., 1995.

[rts] *The What, Where and Why of Real-Time Simulation.*

[T. Socolofsky91] T. Socolofsky, C. K. Un tutorial de tcp/ip, 1991. Rfc-es.org/rfc/rfc1180-es.txt.

[win13] *ANTI-WINDUP AND CONTROL OF SYSTEMS WITH MULTIPLE INPUT SATURATIONS*, 2013.