ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"DISEÑO E IMPLEMENTACIÓN DE UN ROBOT VELOCISTA DE COMPETENCIA SOBRE PLATAFORMA FPGA"

INFORME DE PROYECTO DE GRADUACIÓN

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Herman Isaac Veriñaz Jadán

Caril Ronnie Martínez Vera

GUAYAQUIL – ECUADOR

AÑO 2015

AGRADECIMIENTO

Agradezco a Dios, a mis padres y a mi hermana por su apoyo incondicional.

A mis compañeros y profesores que son parte del Club de Robótica de la ESPOL “Robota” por facilitar el desarrollo del presente proyecto de grado.

Al director de este proyecto, Msc Ronald Ponguillo.

Herman Isaac Veriñaz Jadán.

En primer lugar agradezco a Dios, por todas las bendiciones que me ha brindado a lo largo de mi vida.

A mis padres, por el apoyo que siempre me han dado y todo el esfuerzo realizado para que pueda cumplir mis metas.

Agradezco a mis amigos y demás familiares, por motivarme día a día para seguir adelante.

Caril Ronnie Martínez Vera.

DEDICATORIA

A mis padres y a mi hermana que siempre me han apoyado durante mi vida académica.

Herman Isaac Veriñaz Jadán.

A Dios, por darme la fuerza necesaria para sobrellevar los obstáculos presentados en el camino. A mis padres, porque su apoyo ha sido un factor incondicional en mi formación, por sus enseñanzas y todo lo que han hecho por mi bien. A todos mis familiares por su apoyo y confianza que siempre me han brindado.

Caril Ronnie Martínez Vera.

TRIBUNAL DE SUSTENTACIÓN

MSc. Sara Ríos Orellana

DECANA SUBROGANTE

MSc. Ronald Ponguillo Intriago

DIRECTOR DEL PROYECTO DE GRADUACIÓN

MSc. Carlo Valdivieso Armendariz

MIEMBRO DEL TRIBUNAL

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este informe, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”.

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Herman Isaac Veriñaz Jadán.

_____

Caril Ronnie Martínez Vera.

RESUMEN

Este trabajo describe el desarrollo de un robot velocista para competencias en el aspecto electrónico, mecánico y de software.

El robot diseñado puede alcanzar una velocidad máxima de 2.23 m/s en el circuito de competición, este es un camino cerrado formado por rectas y curvas, las curvas pueden tener un radio mínimo de 10 cm. La velocidad mencionada se alcanza sin tener un conocimiento previo de la pista a recorrer.

El computador que controla el robot fue desarrollado sobre la FPGA EP4CE22F17C6N de la familia Cyclone IV de Altera. Para esto se usó la tarjeta de desarrollo DE0-Nano.

El sistema usa el procesador definido en software Nios II de Altera y posee periféricos específicos para el manejo del robot, estos se agregaron desde la biblioteca de Altera y en ciertos casos se crearon usando lenguaje VHDL, para esto se utilizó la herramienta Qsys de QUARTUS II.

El algoritmo que controla el robot es un sistema de control en cascada. Un sistema de realimentación controla la velocidad de los motores y este a su vez es controlado para fijar la dirección del robot y velocidad del centro de masa.

Además del algoritmo de control se implementó en software un sistema de filtrado, para esto se usó el algoritmo de Filtrado de Kalman.

En el primer capítulo, se detalla el problema a resolver, se exponen los objetivos generales y específicos, así como también los alcances y limitaciones del proyecto.

En el segundo capítulo, se describe los elementos usados en el proyecto, módulos electrónicos de control y sensores. Además se presenta aspectos teóricos sobre control y filtrado digital.

En el tercer capítulo, se explica sobre el diseño e implementación del sistema electrónico, mecánico y de software del robot.

En el cuarto capítulo se muestran los resultados experimentales obtenidos al realizar pruebas en pista del robot, rendimiento de los algoritmos de control y filtrado, además se muestra resultados relacionados a la rapidez de procesamiento del computador.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ADC	Analog-to-Digital Conversion
ANSI	American National Standards Institute
BJT	Bipolar Junction Transistor
DC	Direct Current
DIP	Dual in-line package
DMIPS	Dhrystone MIPS (Million Instructions Per Second)
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPCS	Erasable Programmable Configurable Serial
FFC	Flexible Flat Cable
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
HAL	Hardware Abstraction Layer
HDL	Hardware Description Language
ICC	Instantaneous Center of Curvature
IDE	Integrated Development Environment

IMU	Inertial Measurement Unit
ISA	Instruction Set Architecture
I2C	Inter-Integrated Circuit
I/O	Input Output
JTAG	Joint Test Action Group
Ksps	Kilo samples per second
LED	Light-Emitting Diode LiPo
LTI	Linear Time-Invariant
mAh	Milliampere-hour
MATLAB	MATrix LABoratory
MB	Megabyte
MHz	Megahertz
mNm	Millinewton meter
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
PC	Personal Computer
PCB	Printed Circuit Board
PID	Proportional Integral Derivative

PLL	Phase locked loop
PWM	Pulse-Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
ROM	Read-Only Memory
RPM	Revolutions per minute
RS-232	Recommended Standard 232
SBT	Software Build Tools
SDRAM	Synchronous Dynamic Random-Access Memory
SMD	Surface Mount Device
SOPC	System on Programmable Chip
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

INTRODUCCIÓN

Tanto a nivel nacional como a nivel internacional existe una gran cantidad de torneos de robótica. A nivel nacional destaca el Concurso Ecuatoriano de Robótica (CER) y Robot Games Zero Latitud. Entre los principales a nivel internacional se puede nombrar: Robocup, RobotChallenge, All Japan Micromouse Contest, All Japan Robot SUMO Tournament.

Una de las categorías de competencia es la de robots velocistas. Los robots deben recorrer de forma autónoma en el menor tiempo posible un camino cerrado marcado con una línea. Existen además reglas adicionales que pueden diferir según la competencia.

Los robots más rápidos, dentro de su categoría, están en Japón, España, Polonia y Letonia. En Latinoamérica cada vez se está mejorando más en este tipo de competiciones, México es el país que destaca realizando buenas participaciones en torneos internacionales.

Estos eventos tienen como objetivo impulsar el desarrollo de diversas tecnologías relacionadas a la robótica. Además plantean un reto importante al enfrentar a los competidores a problemas reales que deben resolver usando ingeniería.

En el ámbito nacional el nivel de complejidad de los robots ha venido aumentando, aunque existe una monotonía en las soluciones que se

presentan y aún se puede observar robots muy básicos. En general todavía no es común usar técnicas más profesionales.

En este proyecto no se busca presentar una solución óptima o concluyente para los problemas que se presentan en el desarrollo de un robot velocista. De hecho siempre se puede realizar mejoras en cada campo, sea electrónica, mecánica o software, de eso tratan las competencias.

En este trabajo se expone una solución específica, enfocada en el área de software y de electrónica, una implementación diferente a lo que se presenta actualmente en competencias nacionales, que permita desarrollar soluciones nuevas para robots velocistas y en general para robots de competencia ecuatorianos, de forma que se incremente el nivel de dificultad de torneos locales y la competitividad a nivel internacional.

CAPÍTULO 1

1 GENERALIDADES

1.1 Descripción del Proyecto

En este trabajo se presenta el diseño de un robot velocista para competencias. El robot es autónomo y su objetivo es recorrer un camino cerrado en el menor tiempo posible, el camino puede ser marcado con una línea de color negro o blanco sobre una superficie del color opuesto.

Para la detección de la pista se utiliza un arreglo de sensores de luz infrarroja que está ubicado en la parte delantera del robot. Además de

estos sensores se emplea otro tipo de sensores como los son: codificadores rotatorios, giroscopio, acelerómetro y magnetómetro, ya sea para mejorar el algoritmo de control, como para la adquisición de datos.

La solución que se presenta utiliza técnicas de ingeniería, adquisición y análisis de datos, diseño y prueba de algoritmos usando el software MATLAB, una estructura de control usando un sistema de realimentación en cascada y la implementación del Filtro de Kalman para mejorar las mediciones de los sensores.

El sistema computacional que controla el robot se lo implementa en un solo chip, una FPGA. El sistema es controlado por el procesador definido en software NIOSII de Altera. Para este trabajo se usó la tarjeta de desarrollo DE0-Nano.

El proyecto está orientado a la robótica en su aspecto competitivo, así como para aplicaciones de control y procesamiento de señales usando FPGA.

1.2 Identificación del Problema

Con el avance de la tecnología, las competencias de robótica han tenido un gran desarrollo en los últimos años, varios países alrededor

del mundo cuentan con eventos a nivel nacional, además existen torneos regionales y a nivel mundial, siendo los países de Asia Oriental los que más destacan. Entre tantas categorías existentes en este tipo de competencias, la de robots velocistas está entre las principales.

En competencias nacionales se presentan básicamente dos versiones de robot velocistas. Hay una versión muy básica del robot en la cual se usa un control de posición tipo “on-off”, este control puede ser implementado usando electrónica analógica, o a su vez usando un microcontrolador con una programación por casos, es decir una estructura tipo “case” en la cual se especifica cómo debe moverse el robot de acuerdo a que sensor infrarrojo se activa. Este tipo de robots cumplen con el objetivo de seguir el camino, pero el problema de controlar de esta forma al robot, es que la respuesta del sistema es oscilatoria y esto no permite recorrer la pista con rapidez y de una manera suave.

En la otra versión del robot se usa una forma diferente de controlador, esta es la más común en competencias y un poco más compleja, consiste en implementar en un microcontrolador un algoritmo de control de posición tipo PID. Este algoritmo mejora considerablemente el desempeño del robot, pero debido a diferentes factores, como la forma de la pista a recorrer o parámetros físicos asociados al diseño del robot,

principalmente la tracción, el momento de inercia y la masa, existe una máxima velocidad que se puede alcanzar antes de que el robot empiece a perder la pista y salirse del recorrido, generalmente la curva más cerrada de la pista limita la velocidad.

Mejorar el desempeño del robot a partir de este punto, se vuelve complicado, las mejoras que se suelen presentar se enfocan en el diseño físico del robot. La dificultad principal en software es que se necesita mayor cantidad de procesamiento y los microcontroladores usados comúnmente para pequeñas aplicaciones de robótica se vuelven obsoletos, por lo que el algoritmo necesario no puede ser implementado.

1.3 Justificación

El desarrollo de este proyecto permitirá que las competencias nacionales de robótica, en esta categoría, aumenten su nivel de dificultad.

El conocimiento generado en la construcción de este robot será de utilidad para futuras aplicaciones de control y procesamiento de señales sobre FPGA.

Se construirá por primera vez en Ecuador un robot velocista de nivel de competencias internacionales, que servirá como motivación para que jóvenes estudiantes ingresen al mundo de la robótica.

1.4 Objetivos

1.2.1 Objetivos Generales

La construcción de un robot de competición que sea capaz de representar a nuestra institución en competencias de robótica a nivel nacional y a nuestro país a nivel internacional.

Diseñar e implementar un robot velocista que alcance una velocidad máxima de al menos 2 m/s y presente una mejora respecto a las versiones utilizadas actualmente a nivel nacional, haciendo énfasis en el software y en la electrónica.

1.2.2 Objetivos Específicos

Diseñar software y componentes en HDL sobre FPGA, necesarios para el procesamiento de señales y control, con el fin de obtener el mejor rendimiento posible del robot.

Diseñar el circuito electrónico más simplificado posible que permita: llevar la información obtenida por los diferentes sensores hacia la tarjeta DE0-Nano, ya sea señales analógicas o digitales e información que se envía mediante comunicación serial; así como, transmitir señales de control desde la FPGA hacia los motores, y datos hacia un módulo de comunicación inalámbrica.

Construir el hardware tratando de que las características mecánicas del robot beneficien el desempeño de los algoritmos de control y siempre que el hardware esté dentro de las dimensiones físicas que requieren este tipo de robots.

1.5 Metodología

Revisión del estado de desarrollo actual de robots velocistas en competencias nacionales, comparación con modelos usados internacionalmente.

Diseño y construcción de prototipo físico del robot con sensores adicionales que permitan obtener mayor información sobre el desempeño del robot, los sensores son: giroscopio, acelerómetro, magnetómetro y codificadores rotatorios.

Creación de la computadora sobre FPGA que contenga los componentes de Qsys necesarios para el manejo de sensores, dispositivos de transmisión y actuadores.

Diseño e implementación de algoritmo de control de velocidad para cada motor, para esto se usará codificadores rotatorios en cada motor.

Diseño e Implementación de algoritmo de control de posición para centrar el robot en la pista.

Implementación de algoritmo de filtrado digital y fusión de sensores para mejorar la calidad de las señales, complementar la información proporcionada por los sensores infrarrojos y crear una visión más completa de lo que sucede.

Adquisición y análisis de datos de posición, aceleración y velocidad angular. Según se necesite, la información podrá ser guardada en la memoria RAM incluida en la tarjeta o transmitida inalámbricamente usando un módulo de transmisión.

Pruebas en pista para evaluar el desempeño, mejorar o rediseñar los algoritmos de control y de filtrado, para esto se usará el software MATLAB y se analizará ángulo de giro, aceleración lineal, velocidad lineal, velocidad angular y distancia del robot a la pista.

Diseño final del robot, agregando el circuito impreso que permita minimizar el uso de cables.

Medición de datos inerciales para el diseño final.

1.6 Alcances y Limitaciones del Proyecto

El presente proyecto tiene los siguientes alcances:

- El robot construido es competitivo a nivel nacional y de Latinoamérica, las competencias en Europa y Asia tienen un nivel superior principalmente debido a la facilidad que tienen para el diseño y la construcción del hardware, en lo electrónico y mecánico.
- En este proyecto se hace mayor énfasis en la parte electrónica y de software, y no en la mecánica, debido a los conocimientos adquiridos a lo largo de nuestra carrera.
- El robot diseñado está limitado principalmente a competencias con pistas planas o pistas con pendientes que no excedan los 20 grados de inclinación.
- El robot tendrá un correcto desempeño únicamente en pistas del material adecuado. Las pistas suelen ser planchas de formica o

melamina pintadas de color blanco o negro, en pistas de diferente material su correcto desempeño no está garantizado.

Además de las siguientes limitaciones:

- No existe mucha información relacionada a la construcción de robots velocistas profesionales. Existen pocos sitios web donde se explica sólo aspectos básicos sobre los robots, aunque sí se puede encontrar información sobre robots similares, como los Micromouse[1].
- Para el sistema de reducción mecánico del motor se necesita rodamientos, engranes, arandelas, diferentes tipos de tornillos y tuercas de tamaño muy pequeño. En Ecuador es difícil encontrar a la venta este tipo de materiales debido a que no es común trabajar en proyectos que utilicen piezas mecánicas de ese tamaño.
- Se necesita precisión para la construcción de piezas mecánicas de robots de competición, estas piezas es posible realizarlas en Ecuador pero su precio puede ser elevado. Se encontró una alternativa en las impresiones 3D, las cuales dan una precisión aceptable y a un buen precio.

[1] Competencia donde un pequeño robot debe encontrar la salida de un Laberinto.

- En Ecuador se puede encontrar sitios donde se realiza PCB de buena calidad, pero es difícil conseguir elementos electrónicos pequeños y de soldadura superficial que permitan optimizar el diseño del circuito, esto limita la forma y el tamaño del robot.

CAPÍTULO 2

2 MARCO TEÓRICO

En este capítulo se presentan los principales conceptos teóricos relacionados con el desarrollo del proyecto.

2.1 Sistemas Embebidos Configurables

Un sistema embebido es un sistema de computación sujeto a elementos físicos [1]. El sistema cuenta con la electrónica necesaria para realizar un control computacional dedicado a un propósito específico, lo cual lo diferencia a un ordenador de propósito general, cómo por ejemplo una PC [2]. Debido a esto un sistema embebido

posee solo el hardware necesario para realizar una o varias funciones encaminadas a cumplir con dicho propósito, por lo que su costo y tamaño se ven reducidos.

Un circuito embebido está formado por circuitos integrados programables, memoria flash o ROM, el circuito impreso y el software embebido que se ejecuta sobre la respectiva arquitectura de control.

2.1.1 Tarjeta de Desarrollo DE0-Nano de ALTERA

La tarjeta DE0-Nano presenta una plataforma de desarrollo basado en FPGA, adecuado para una gran variedad de prototipos electrónicos, cómo robots o proyectos móviles. La placa está diseñada cómo una implementación de CYCLONE IV y es ideal para diseños con sistemas embebidos [3].

La DE0-Nano tiene una colección de interfaces, entre ellas dos cabeceras GPIO externas, con el fin de poder extender los diseños más allá de la tarjeta; además cuenta con elementos de memoria como SDRAM y EEPROM, con el fin de tener un almacenamiento de datos más grandes, así como periféricos de uso general como LEDs y pulsadores.

Entre las ventajas de la tarjeta DE0-Nano se incluyen su tamaño y peso, así como su capacidad para reconfigurarse. Además, en

diseños móviles donde la energía es crucial, la DE0-nano proporciona a los diseñadores opciones para proveer energía, las cuales son: un mini puerto USB, cabecera de alimentación externa de dos pines, y dos pines de 5V DC.

Las características principales de la tarjeta DE-0 Nano se enlistan a continuación [3]:

- Dispositivos destacados
  - FPGA Altera Cyclone® IV EP4CE22F17C6N
  - 153 pines máximos de E/S
- Elementos de configuración
  - Circuito para programación USB-Blaster
  - Dispositivo para configuración serial EPCS64
- Cabeceras de expansión
  - Dos puertos de 40 pines (GPIOs), proveen 72 pines de E/S, 2 pines de alimentación de 5V, 2 pines de alimentación de 3.3 V, 4 pines de tierra.
- Dispositivos de memoria
  - 32Mb SDRAM
  - 2Kb I2C EEPROM
- Interfaces de entrada/salida
  - 8 LEDs verdes

  - 2 pulsadores sin rebote
  - 4 DIP switch
- G-Sensor
  - ADI ADXL345, acelerómetro de 3 ejes con alta resolución (13-bits)
- Convertidor A/D
  - NS ADC128S022, 8-Canales, Convertidor A/D de12-bits
  - 50 Ksps a 200 Ksps
- Reloj del Sistema
  - Oscilador de 50 MHz
- Fuentes de alimentación
  - Mini puerto USB tipo AB (5V)
  - Pin de 5 V DC para cada cabecera GPIO (2 pines 5 VDC)
  - 2 pines de alimentación externa (3.6 – 5.7 V)

Diseño y Componentes

En la Figura 2.1 y Figura 2.2 se representa el diseño de la tarjeta y se indica la ubicación de los conectores y componentes principales.

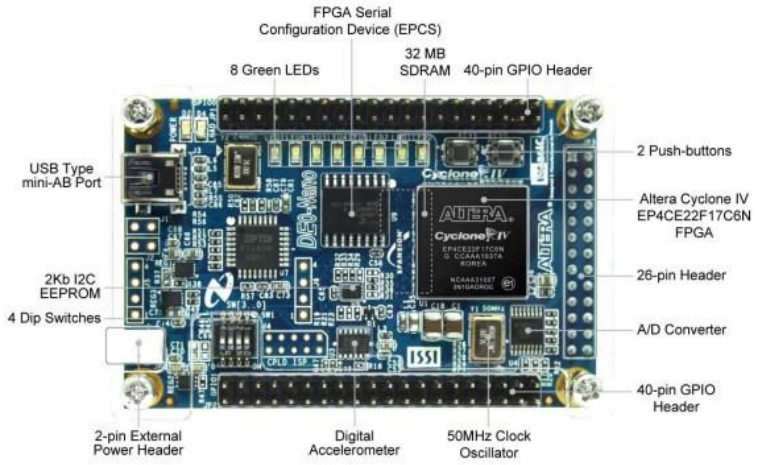


Figura 2.1 Tarjeta PCB y diagrama de componentes de la DE0-Nano (vista superior) [3]

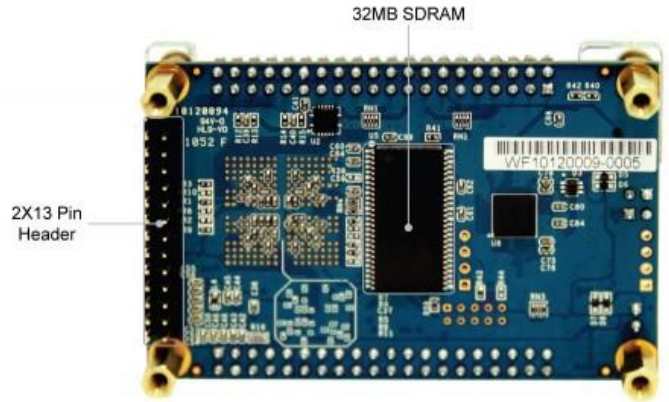


Figura 2.2 Tarjeta PCB y diagrama de componentes de la DE0-Nano (vista inferior) [3].

Diagrama de bloques de la tarjeta DE0-Nano

La Figura 2.3 muestra el diagrama de bloques de la tarjeta DE0-Nano. Todas las conexiones se realizan a través de la FPGA

Cyclone IV. Por lo tanto, el usuario puede configurar la FPGA para implementar cualquier diseño del sistema.

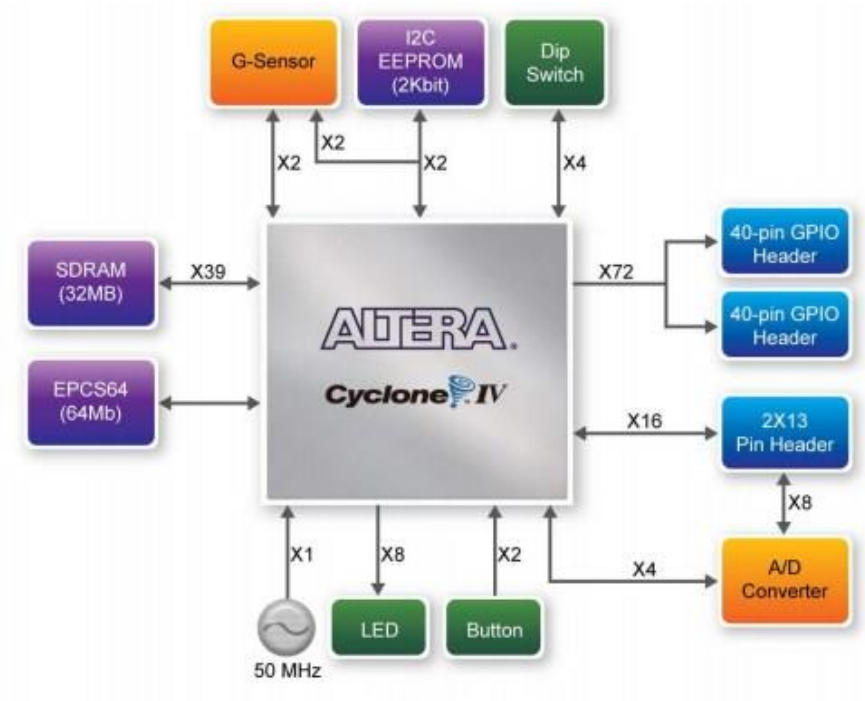


Figura 2.3 Diagrama de bloques de la tarjeta DE0-Nano [3].

2.1.2 Procesador NIOS II

Nios II es una arquitectura de procesador de 32 bits diseñado específicamente para la familia de FPGAs de Altera, está basado en la arquitectura tipo Harvard. Nios II incorpora muchas mejoras con respecto a la arquitectura original Nios de 16 bits, por lo que es más adecuado para una gama más amplia de aplicaciones de sistemas embebidos.

Nios II está definido en un lenguaje de descripción de hardware, el cual puede ser implementado sobre dispositivos FPGA de Altera usando Quartus II.

Se puede encontrar Nios II en tres configuraciones diferentes: Nios II/f, Nios II/s, Nios II/e [4].

Nios II/f (fast): Está diseñado para un máximo rendimiento a costa del tamaño del núcleo.

Nios II/s (standard): Está diseñado para tener un equilibrio ente el rendimiento y el costo.

Nios II/e (economy): Está diseñado para la utilización lógica más pequeña posible, especialmente eficiente para aplicaciones de Cyclone II de bajo coste.

Características principales del procesador NIOS II

El NIOS II es un procesador de propósito general con las siguientes características principales [5]:

- Juego de instrucciones, datos y direcciones de 32 bits.
- 32 registros de propósito general.
- Conjunto de registros “shadow” opcionales.
- 32 fuentes de interrupción.

- Interfaz de control de interrupciones externas para más fuentes de interrupción.
- Instrucción de multiplicar y dividir de 32x32 con resultado de 32 bits.
- Instrucciones dedicadas para procesar multiplicaciones de 64 y 128 bits.
- Instrucciones opcionales para operaciones de punto flotante.
- Acceso a una variedad de periféricos integrados, e interfaces a memorias y periféricos externos.
- Módulo de depuración asistido por hardware que permite que el procesador inicie, se detenga y compile bajo las herramientas de desarrollo de software de NIOS II.
- Unidad de gestión de memoria (MMU) opcional como apoyo para sistemas operativos que la requieran.
- Unidad de protección de memoria (MPU) opcional.
- Entorno de desarrollo de software basado en la gama de herramientas de GNU C/C++ y NIOS II Software Build Tools (SBT) para Eclipse.
- Integración con Altera's SignalTap® II Embedded Logic Analyzer que permite el análisis en tiempo real de instrucciones y datos, junto con otras señales en la FPGA.

- Arquitectura de conjunto de instrucciones (ISA) compatibles con todos los sistemas de procesador NIOS II.
- Rendimiento de hasta 250 DMIPS.
- Soporte de código corrector de errores (ECC) opcional, para un subconjunto de bloques de RAM internos del procesador NIOS II.

Un sistema de procesador NIOS II es equivalente a un microcontrolador o “computador en un chip”, que incluye un procesador y una combinación de periféricos y memoria en un mismo encapsulado. Consta de un núcleo de procesador NIOS II, un conjunto de periféricos integrados, memoria integrada y las interfaces a las memorias externas, todo implementado en un único dispositivo Altera.

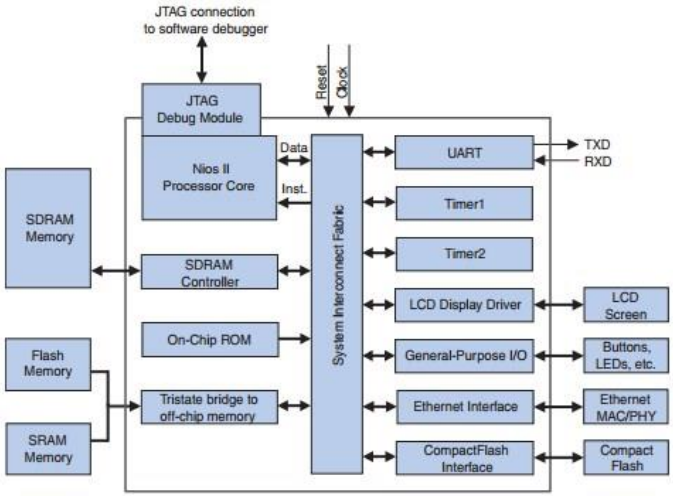


Figura 2.4 Ejemplo de un sistema de procesador NIOS II [5].

Arquitectura del procesador

NIOS II describe una arquitectura de conjunto de instrucciones (ISA). ISA a su vez requiere de un conjunto de unidades funcionales que implementen las instrucciones. Un núcleo de procesador NIOS II es un diseño de hardware que implementa las instrucciones NIOS II y sirve como apoyo para las unidades funcionales. El núcleo del procesador no incluye periféricos ni la lógica de conexión con el mundo exterior. Incluye solo los circuitos necesarios para implementar la arquitectura NIOS II [5].

La arquitectura NIOS II define las siguientes unidades funcionales:

- Unidad lógica aritmética (ALU).
- Interfaz a la lógica de instrucción personalizada.
- Controlador de excepción.
- Controlador de interrupción interna o externa.
- Bus de instrucción.
- Bus de datos.
- Unidad de gestión de memoria (MMU).
- Unidad de protección de memoria (MPU).
- Memoria caché de instrucción y datos.
- Interfaces de memoria para datos e instrucciones.
- Módulo de depuración JTAG.

- Archivo de registro.

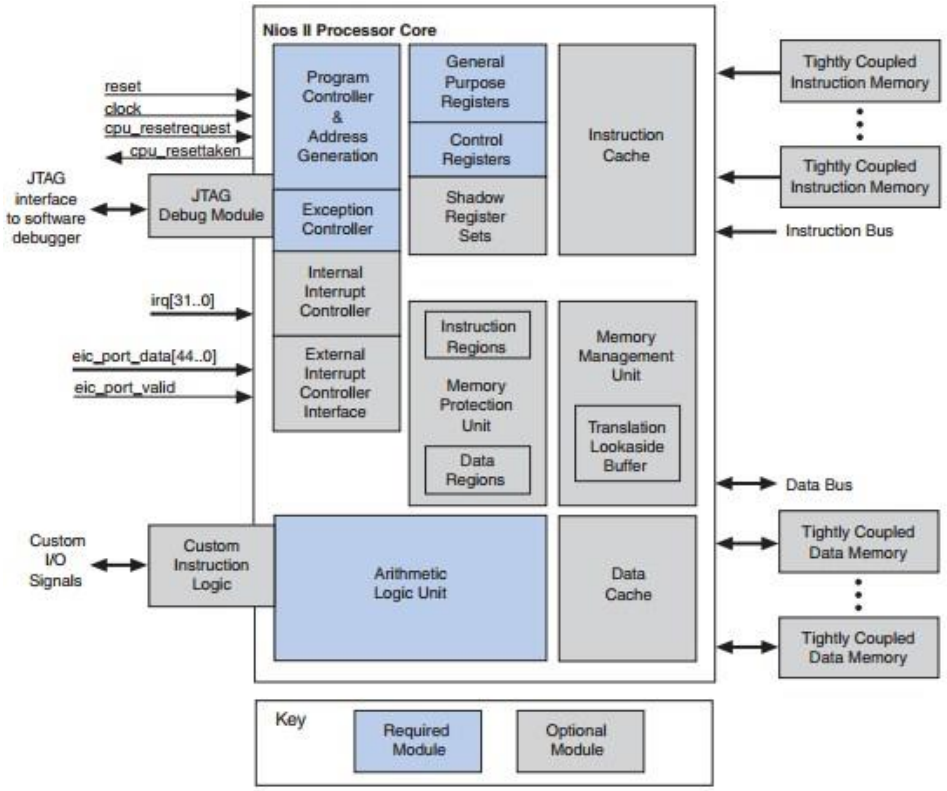


Figura 2.5 Diagrama de bloques del núcleo del procesador NIOS II [5].

2.2 Robótica de Competencia: Robot Velocista

Así como en muchas otras áreas, en la robótica existen competencias. En éstas los participantes deben diseñar robots que realicen una determinada tarea, ya sea de manera autónoma o controlada por un

humano. Estos eventos tienen como objetivo impulsar el desarrollo de diversas tecnologías para el diseño de robots.

En las competencias de robot existen varias categorías. Entre las principales categorías encontramos: robot de batalla, robot velocista, robot laberinto, robot sumo, robot soccer.

En nuestro país existen varios eventos de robótica de los cuáles los que más destacan son: Concurso Ecuatoriano de Robótica y Robot Games Zero Latitud. A nivel internacional existen muchas competencias de robótica como Robocup, RoboGames, RobotChallenge, All Japan Micromouse Contest, All Japan Robot SUMO Tournament etc.

Figura 2.6 Diferentes robots participantes del RoboGames.

En la categoría de robot velocista los robots deben ser autónomos y deben ser capaces de recorrer un camino cerrado sin salirse del mismo y lo deben hacer en el menor tiempo posible. Los lineamientos acerca de las características de los robots y de la pista lo determina la entidad organizadora de cada competencia.

En la competencia básica el robot debe ser capaz de completar una vuelta al circuito y lo debe hacer en el menor tiempo posible. Pero además existen algunas variaciones, dos de las más interesantes debido a la velocidad que se puede alcanzar son:

- Persecución: Compiten dos robots en la misma pista, estos se colocan de forma que recorran el circuito en el mismo sentido. La pista es simétrica para que ambos recorran la misma distancia, los robots deben arrancar al mismo tiempo y la competencia termina cuando un robot alcance al otro. Debido a que los robots pueden recorrer varias veces el circuito hasta que uno alcance al otro, es posible usar algoritmos de reconocimiento de pista que permitan incrementar la velocidad del robot en tramos rectos.
- NatCar: Su principal característica es que cuenta con marcas de señalización. A la izquierda de la pista hay marcas de inicio y fin de cada curva, a la derecha habrán dos marcas las cuales indican el inicio y fin del recorrido, el robot tendrá que detenerse de

manera autónoma entre ellas. El robot tienen tres intentos, la idea es que el robot memorice el recorrido en el primer intento y optimice su tiempo a partir del segundo.

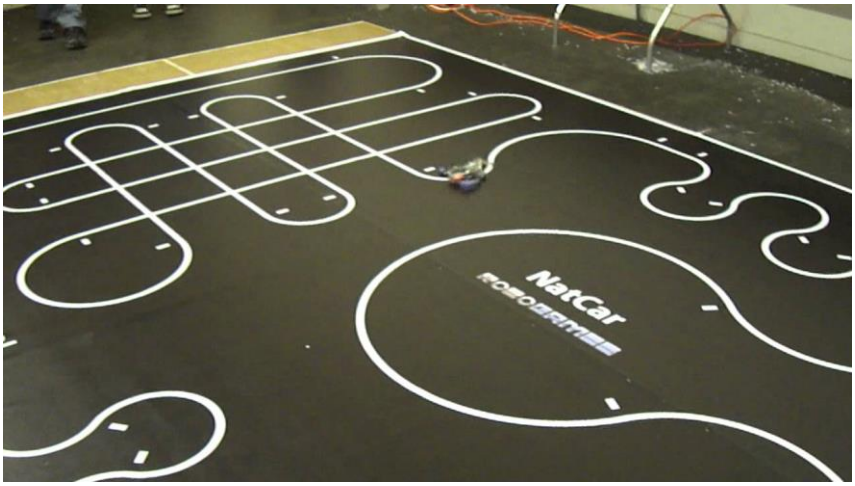


Figura 2.7 Parte de la trayectoria de una competencia de NatCar.

Robots velocistas nacionales e internacionales

A nivel nacional, los robots velocistas han tenido un avance en los últimos años, desde el año 2012 se comenzó a utilizar un algoritmo de control tipo PID y a partir de ahí se ha venido mejorando los prototipos tanto en mecánica, electrónica y software.

Los robots ecuatorianos por lo general usan motores de marca Pololu, los cuales se caracterizan por un precio económico, peso ligero, gran velocidad, pero con poco torque. Entre los microcontroladores más

usados están los PIC de Microchip y los AVR de ATMEL. Para detectar la línea es común usar arreglos de 8 sensores infrarrojos de Pololu.

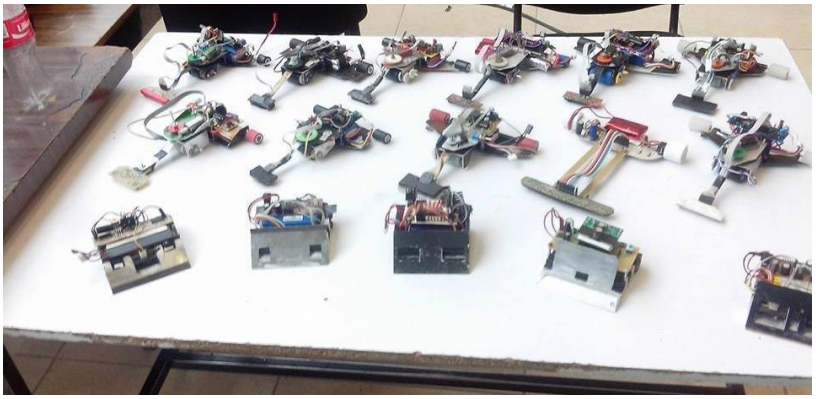


Figura 2.8 Robots participantes del UMEBOT 2014.

En la Figura 2.8 se muestran varios robots velocistas, participantes del UMEBOT 2014, concurso realizado en Quito.

A nivel internacional existe mucha variedad de diseños para los robots velocistas. En Latinoamérica y Europa existe un gran predominio de velocistas con configuración diferencial, destacándose los diseños polacos, letones y españoles. El robot de la Figura 2.9 es el campeón del RobotChallenge 2014.

Figura 2.9 Robot Letón BridgeRacer

En Polonia y Letonia se usa arreglos de entre 12 y 18 sensores infrarrojos, los motores usados son de alto torque y velocidad lo que implica también un peso elevado. La estructura principal de estos suele ser el mismo PCB, o es construida aparte de un material ligero y la electrónica se monta en un circuito electrónico flexible.

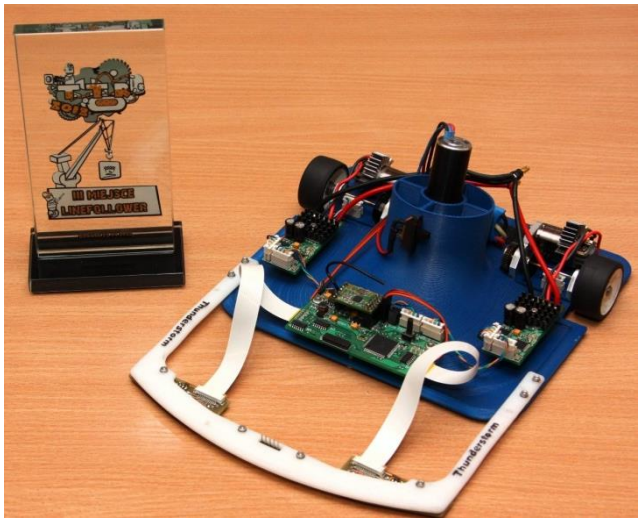


Figura 2.10 Robot Polaco Thunderstorm

El robot de la Figura 2.10 es el campeón del RobotChallenge 2013. Usa un motor con una hélice como un mecanismo de succión para ganar más adherencia al suelo, su estructura fue impresa en 3D y la electrónica es montada aparte.

En España los robots se caracterizan por ser muy ligeros, se usa un arreglo de máximo 8 sensores infrarrojos, se usa motores de marca Pololu, el PCB es también el chasis y se suele usar algoritmos de reconocimiento de pista para aumentar la velocidad en rectas. Pero hay que tomar en cuenta que el reglamento español no permite curvas muy pronunciadas, el radio mínimo es por lo general 30 cm.

Las velocidades más altas registradas se registran en robots asiáticos, especialmente en los diseños japoneses. Allí son populares las competencias de Natcar. En la Figura 2.9 se muestra el Cartisx04, un robot japonés ganador de las principales competencias de robots velocistas en Japón en los últimos años.

Figura 2.11 Robot velocista Cartisx04.

Su diseño usa una configuración diferencial de cuatro ruedas. Tiene tres motores, dos a los lados para mover las ruedas y otro para girar el arreglo de sensores. Esto último permite que el arreglo de sensores sea pequeño. Los motores que se usa tienen un alto costo debido a que no solo tienen buena velocidad final, sino también un buen torque.

2.3 Sistemas de Control Discreto

Un sistema de control es un conjunto de señales que interactúan con el objeto de que la salida de un proceso se comporte como se desee, utilizando acciones de control.

Un sistema de control en tiempo de discreto es aquel en que conocemos el valor de sus variables solamente en ciertos instantes, a

diferencia de los sistemas de control continuo en los cuales conocemos estos valores en todo momento.

Muchas veces nos encontramos que la variación discreta del sistema proviene de señales continuas sometidas a un proceso de muestro digital.

Debido que en la práctica los controladores de hoy en día están implementados digitalmente, los sistemas de control discreto son de gran importancia para cualquier sistema de control.

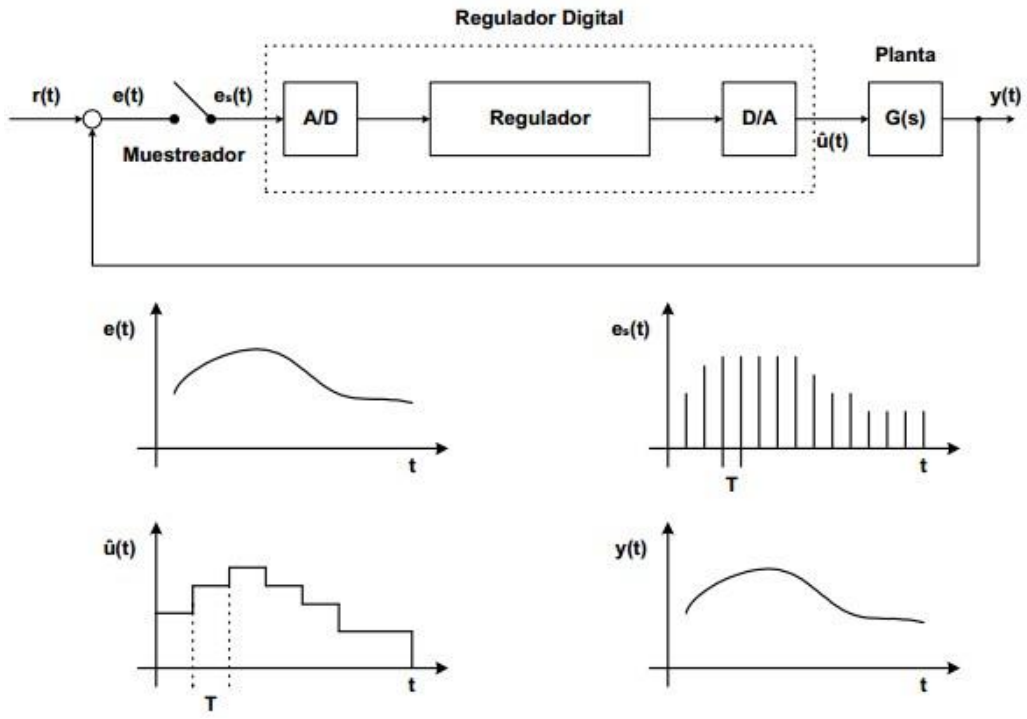
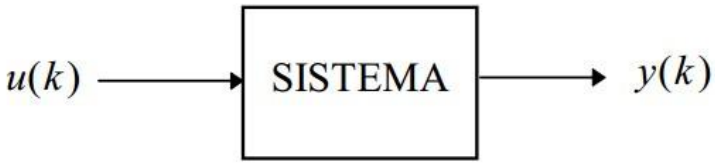


Figura 2.12 Sistema de control en tiempo discreto.

Aunque un sistema de control puede ser estable implementado de manera continua, podría ser inestable implementado de manera discreto debido a un intervalo de tiempo de muestreo grande.

Ecuaciones de diferencia

Los sistemas de control en tiempo discreto, son descritos mediante ecuaciones de diferencia, las cuales relacionan la señal de salida con la señal de entrada.



$$u(kT) = u(k) = \{u(0), u(T), u(2T), \ldots, u(kT), \ldots\}$$

$$y(kT) = y(k) = \{y(0), y(T), y(2T), \ldots, y(kT), \ldots\}$$

Figura 2.13 Señales de entrada y salida de un sistema de control discreto [6].

La ecuación general sería:

f(y(k),y(k−1), ..., y(k−n), u (k), u (k−1), ..., u (k−n))=0

Los sistemas discretos lineales e invariantes en el tiempo están definidos por:

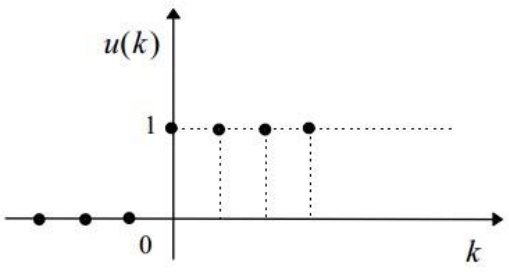
$$a_n y(k+n) + \ \ldots + \ a_1 y(k+1) + \ a_0 y(k)$$
$$= \ b_n u(k+n) + \ldots + b_1 u(k+1) + \ b_0 u(k)$$

Transformada Z

Debido a que una herramienta que juega un papel importante en los sistemas de tiempo continuo es la transformada de Laplace, en los sistemas discretos se puede describir una señal x(k) por su transformada X(z), la cual es llamada Transformada Z. La Transformada Z para una señal x(k), con x(k)=0, para valores de k menores a 0 se define como:

$\mathrm{X(z)} = \mathrm{Z\{x(k)\}} = \sum_{k=0}^{\infty} \mathrm{x(k)z^{-k}}$; z ϵ ℂ

$\mathrm{X(z)} = \mathrm{x(0)} + \mathrm{x(1)z^{-1}} + \mathrm{x(2)z^{-2}}$



$$X(z) = \sum_{k=0}^{\infty} 1 \cdot z^{-k} = \sum_{k=0}^{\infty} (z^{-1})^k$$

$$X(z) = \sum_{k=0}^{\infty} (z^{-1})^k = \frac{1}{1 - z^{-1}}$$

Figura 2.14 Transformada Z de señales [6].

Transformada Inversa de Z

La transformada Z^{-1} nos permite obtener la expresión de x(k) a partir de X(z). Para ellos se descompone en fracciones simples $\frac{X(z)}{z}$ y mediante el método de los residuos se obtienen los valores k_{ij}.

$$\frac{X(z)}{z} = \frac{k_{11}}{z+p_1} + \frac{k_{12}}{(z+p_1)^2} + \cdots + \frac{k_{1r_1}}{(z+p_1)^{r_1}} + \cdots + \frac{k_{21}}{z+p_2} + \frac{k_{22}}{(z+p_2)^2} + \cdots + \frac{k_{2r_2}}{(z+p_2)^{r_2}} + \cdots$$

Luego se aplican métodos que varían dependiendo de raíces simples y repetidas y aplicando las relaciones x(k) – X(z) de acuerdo a la transformada Z.

Existe otro método más sencillo es mediante la división directa X(z) = $\frac{N(z)}{D(z)}$, donde el grado de N(z) es menor o igual que el grado de D(z).

Con lo cual se obtendrá que

$$X(z) = c_0 + c_1 z^{-1} + c_2 z^{-2} + \cdots$$

x(k) = $\{c_0 , c_1 , c_2 , \ldots \}$

Función de Transferencia Discreta

Para un sistema de control discreta LTI, la función de transferencia es la relación entre la Transformada Z de la salida y la Transformada Z de la entrada, con sus condiciones iniciales igualadas a cero.

$$G(z) = \frac{Y(z)}{R(z)}$$

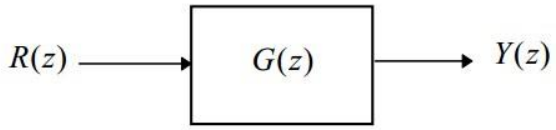


Figura 2.15 Función de transferencia de un sistema discreto [6].

En un sistema discreto LTI tomando las condiciones iniciales nulas, tenemos que:

$$G(z) = \frac{Y(z)}{R(z)} = \frac{b_m z^m + \cdots + b_1 z + b_0}{a_n z^n + \cdots + a_1 z + a_0}$$

Dado G(z) y la entrada R(z) podemos obtener los valores Y(z)

Y(z)=R(z)G(z)

Y aplicar Z^{-1} para obtener los valores de y(k).

2.4 Fusión de Sensores

Con la finalidad de tener una percepción de ciertas características físicas del entorno, en muchos tipos de sistemas usamos transductores de entradas a los que llamamos sensores. Un transductor es un dispositivo que convierte una señal tipo de energía en otra, por lo general eléctrica. Por lo que un sensor transforma señales del medio, las cuáles nos sirven como información de entrada para un sistema.

La fusión de sensores es la combinación de datos provenientes de distintos sensores con la finalidad de que la información resultante sea más precisa que cuando se usan estos sensores de manera individual.

La base teórica de la fusión multisensor se encuentra en investigación de operaciones, estadísticas y probabilidades, y teoría de estimación.

La fusión de datos, ya sea de múltiples sensores o de un solo sensor puede tener diferentes niveles de representación, los cuáles se podrían denominar: nivel de señal, de pixel, de características y de símbolos [7]; existiendo diferentes métodos de fusión para cada nivel. La mayoría de los sensores típicos usados en la práctica proveen datos que pueden ser fusionados en uno o más de esos niveles.

Los diferentes niveles de la fusión multisensor pueden ser usados para proveer información al sistema, que puede ser usada para una variedad

de propósitos: El nivel de señal puede ser usado en aplicaciones de tiempo real y se puede considerar simplemente como un paso adicional en el procesamiento general de las señales; el nivel de pixel puede ser usado para mejorar el rendimiento de muchas tareas de procesamiento de imágenes tal como una segmentación; y los niveles de característica y símbolo pueden ser utilizados para mejorar un sistema que realice la tarea de reconocimiento de objetos con características adicionales que se pueden utilizar para aumentar sus capacidades de reconocimiento. Los diferentes niveles pueden ser reconocidos por el tipo de información que proveen al sistema, cómo se modeliza la información sensorial, el grado de registro de sensor requerido para la fusión, los métodos usados y los medios por los que el proceso de fusión mejora la calidad de la información proporcionada. Los algoritmos de fusión son clasificados en dos tipos generales, distinguidos por la manera en que la información se combina para obtener una solución global al problema de procesamiento multisensorial: en los algoritmos débilmente acoplados, la operación de los diferentes módulos de procesamiento sensorial no es afectada por el proceso de fusión; en los algoritmos fuertemente acoplados la salida de un módulo interactúa con otros módulos y afecta su operación [7].

Sistema de control en cascada

Los sistemas de control en cascada son casos especiales de los sistemas de control con variables auxiliares controladas y aplicables a lo que respecta fusión de sensores. Como se puede apreciar en la Figura 2.16, el controlador principal con la función de transferencia G_{C_2} no afecta directamente al actuador, pero proporciona el valor de referencia para el controlador auxiliar con la función de transferencia G_{C_1}. Este controlador auxiliar junto con la primera sección de la planta G_{P_1}, forman el bucle de control auxiliar que está dentro del bucle de control principal. Las perturbaciones en la primera sección de la planta serán controladas por el controlador auxiliar con el fin de que tengan menos influencia en la segunda sección, por lo que el controlador principal actuaría de manera más leve.

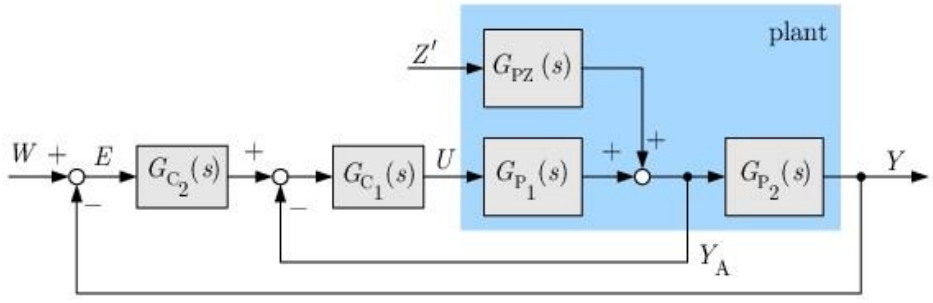


Figura 2.16 Diagrama de bloques de un sistema de control en cascada [8].

Cuando se miden múltiples variables auxiliares, se pueden construir múltiples sistemas de control en cascada. Para el sistema de control en cascada de la Figura 2.16, la variable controlada *y* está dada por:

$$Y = \left\{\left[(W - Y)G_{C_2} - \frac{Y}{G_{C_2}}\right]G_{C_1}G_{P_1} + Z'G_{PZ}\right\}G_{P_2},$$

con lo cual tenemos

$$Y = \frac{G_{PZ}G_{P_2}}{1 + G_{C_1}G_{P_1}\left(1 + G_{C_2}G_{P_2}\right)}Z' + \frac{G_{C_1}G_{C_2}G_{P_1}G_{P_2}}{1 + G_{C_1}G_{P_1}\left(1 + G_{C_2}G_{P_2}\right)}W$$

2.5 Filtrado Digital

El filtrado digital es una de las herramientas más poderosas de un procesador digital de señales (DSP). Los filtros tienen dos usos: separación de la señal y de restauración de la señal.

La separación de la señal es necesaria cuando una señal ha sido afectada con interferencia, ruido u otras señales. Por ejemplo, dado un dispositivo para medir la actividad eléctrica del corazón de un bebé mientras está en el vientre materno. Es probable que la señal se corrompa por la respiración y el latido del corazón de la madre. Un filtro puede ser utilizado para separar estas señales de manera que puedan ser analizadas individualmente [9].

La restauración de la señal se utiliza cuando una señal ha sido distorsionada de alguna manera. Por ejemplo, una grabación de audio hecha con un equipo de mala calidad puede ser filtrada para representar mejor el sonido, ya que en realidad ocurrió.

Los filtros digitales pueden alcanzar miles de veces mejor rendimiento que los filtros analógicos [9]. Esto hace una gran diferencia en cómo se abordan los problemas de filtrado. Con filtros analógicos, el énfasis está en el manejo de las limitaciones de los componentes electrónicos, tales como la precisión y la estabilidad de las resistencias y condensadores. En comparación, los filtros digitales son tan buenos que el rendimiento del filtro se ignora frecuentemente. El énfasis se desplaza a las limitaciones de las señales.

Como se muestra en la Figura 2.17, cada filtro lineal tiene una respuesta de impulsos, una respuesta de paso y una respuesta de frecuencia. Cada una de estas respuestas contiene información completa sobre el filtro, pero en una forma diferente. Si se especifica una de los tres, las otras dos son fijas y se pueden calcular directamente.

La forma más sencilla de implementar un filtro digital es mediante la convolución de la señal de entrada con la respuesta al impulso del filtro

digital. Cuando se utiliza la respuesta de impulso de esta manera, los diseñadores le dan un nombre especial: el núcleo del filtro.

También hay otra manera de diseñar filtros digitales, llamada recursividad. Cuando un filtro se implementa mediante convolución, cada señal en la salida se calcula mediante la ponderación de las señales en la entrada. Los filtros recursivos son una extensión de esto, utilizando valores calculados previamente de la salida, además de puntos de la entrada. En lugar de utilizar un núcleo de filtro, filtros recursivos son definidos por un conjunto de coeficientes de recursión.

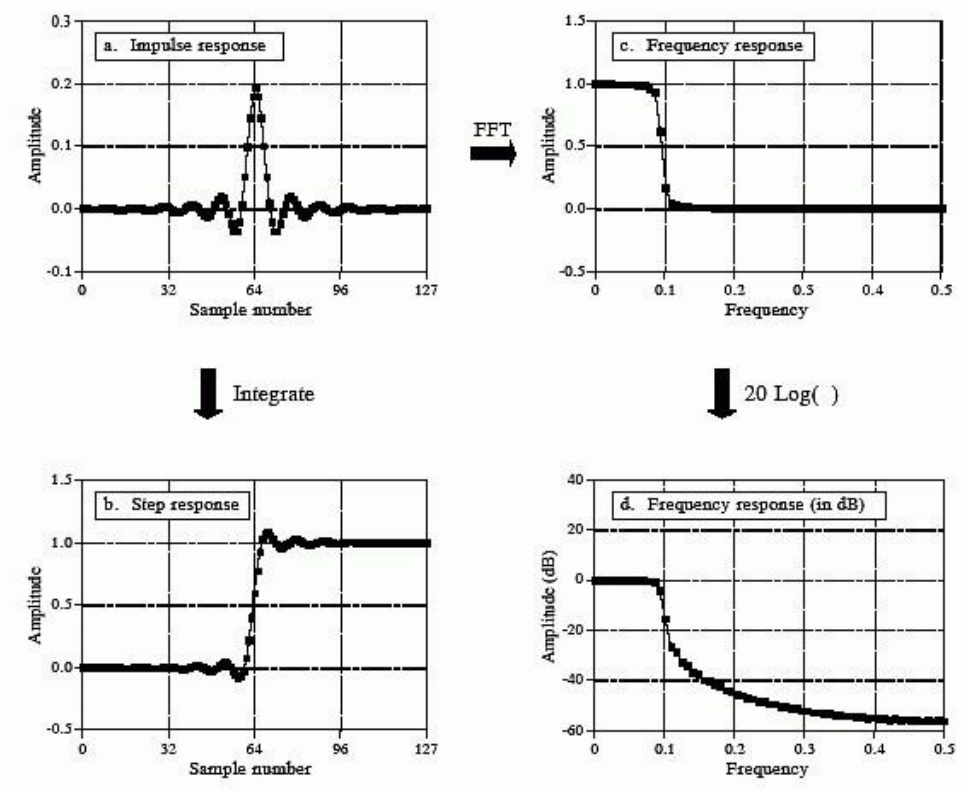


Figura 2.17 Parámetros de un filtro digital [9].

Tipos de filtros digitales

Muchos de los filtros digitales se basan en la transformada rápida de Fourier, un algoritmo matemático que extrae rápidamente el espectro de frecuencia de una señal, permitiendo que este espectro sea manipulado (por ejemplo, para crear filtros pasa banda) antes de convertir el espectro modificado en una serie en el tiempo [9].

Otra forma de un filtro digital es la de un modelo de espacio de estado. Un filtro de espacio de estado bien utilizado es el filtro de Kalman publicado por Rudolf Kalman en 1960.

Filtro de Kalman

El filtro de Kalman se utiliza en una gran cantidad de sistemas multisensor cuando es necesario fusionar datos en tiempo real [7]. El filtro utiliza las características estadísticas de un modelo de medición para determinar de forma recursiva estimaciones para los datos fusionados de una manera óptima. Si el sistema se puede describir con un modelo, y el error del sistema como el error del sensor pueden ser modelados como ruido blanco gaussiano, el filtro de Kalman proporcionará estimaciones estadísticamente óptimas para los datos fusionados. La naturaleza recursiva del filtro lo hace apropiado para sistemas que no tengan un gran almacenamiento de datos. Ejemplos del uso del filtro para la fusión multisensor incluyen el reconocimiento

de objetos usando secuencias de imágenes a partir de un sensor, navegación de un robot,y la teledetección.

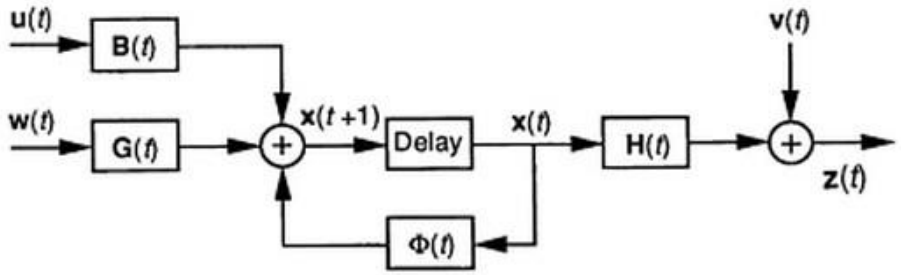


Figura 2.18 Diagrama de bloques del filtro de Kalman [7].

Las mediciones de un grupo de *n* sensores se pueden fusionar usando el filtro de Kalman para proporcionar tanto una estimación del estado actual de un sistema como una predicción del estado futuro del sistema. El estado que se estima puede ser, por ejemplo, la ubicación actual de un robot móvil, la posición y velocidad de un objeto en el entorno, características extraídas de los datos sensoriales, o las propias mediciones reales.

Dado un sistema representado por un proceso lineal discreto Markov, el modelo de estado

$$x(t+1) = \Phi(t)x(t) + B(t)u(t) + G(t)w(t)$$

y el modelo de medición

$$z(t) = H(t)x(t) + v(t)$$

pueden ser usados para describir el sistema (Figura 2.18), dónde

x : vector de estado de dimensión *m*
Φ: matriz de transición de estado de *m* x *m*
B: matriz de transición de entrada de *m* x *p*
u: vector de entrada de dimensión *p*
G: matriz de transmisión del ruido del proceso de *m* x *q*
w: vector de ruido del proceso de dimensión *q*
z: vector de medición de dimensión *n*
H: matriz de medición de *n* x *m*
v: vector de medición de ruido de dimensión *n*

Las secuencias w y v son secuencias de ruido blanco gaussiano, con media cero y correlacionadas en el tiempo discreto, con covarianzas

$$E\{ \mathrm{w}(t_i)\mathrm{w}^T(t_j)\} = \mathrm{Q}(t_i)\delta_{ij}$$

$$E\{ \mathrm{v}(t_i)\mathrm{v}^T(t_j)\} = \mathrm{R}(t_i)\delta_{ij},$$

Dónde $E\{\ldots\}$ denota el valor esperado y δ_{ij} la función delta de Kronecker.

Cuando todos los parámetros (las matrices Φ, B, G, H, Q y R) del modelo son conocidos, las ecuaciones de filtrado óptimo son:

$$\dot{\mathrm{X}}(t|t) = \dot{\mathrm{x}}(t|t\text{-}1) + \mathrm{K}(t)[\mathrm{z}(t) - \mathrm{H}(t)\, \dot{\mathrm{x}}(t|t\text{-}1)]$$

$$\dot{\mathrm{x}}(t\text{+}1|t) = \Phi(t)\, \dot{\mathrm{x}}(t|t) + \mathrm{B}(t)\mathrm{u}(t)$$

dónde $\dot{\mathrm{x}}(t|t)$ es el valor estimado de x(*t*) basado en las mediciones {z(0), z(1), … , z(*t*)}, y $\dot{\mathrm{x}}(t\text{+}1|t)$ es la predicción de x(*t*+1) basado en las

mediciones {z(0), z(1), … , z(*t*)}. La matriz K de *m* x *n* es la “ganancia del filtro de Kalman” y se define como

$$K(t) = P(t|t\text{-}1)H^{T}(t)\,[H(t)P(t|t-1)\,H^{T}(t) + r(t)]^{-1}$$

Dónde $P(t|t-1) = E\{(x(t) - \dot{x}(t|t\text{-}1))\,(x(t) - \dot{x}(t|t-1))^{T}\}$ es la matriz *m* x *n* de covarianza condicional del error en la predicción de x(*t*) y es determinada usando

$$P(t\text{+}1|t) = \Phi(t)P(t|t)\,\Phi^{T}(t) + G(t)Q(t)G^{T}(t),$$

Dónde

$$P(t|t) = P(t|t\text{-}1) – K(t)H(t)\,P(t|t\text{-}1).$$

Las condiciones iniciales para la recursión están dadas por $\dot{x}(0|0) = \dot{x}_0$ y $P(0|0) = P_0$.

2.6 Sensores Inerciales

Se conocen como sensores inerciales a aquellos que nos dan información acerca de la velocidad, orientación y fuerzas gravitacionales, para lo cual requieren de acelerómetros, giroscopios, y en algunos casos magnetómetros; estos sensores también son conocidos como IMU (unidad de medición inercial).

Debido a que ciertos datos de medición inercial son de suma importancia para el desarrollo del presente proyecto, se ha optado por el uso del MinIMU-9 v3 de Pololu.

El MinIMU-9 v3 de Pololu es una unidad de medición inercial compacta que viene equipada con un L3GD20H, que es un giroscopio de 3 ejes; y un LSM303D que contiene un acelerómetro de tres ejes y un magnetómetro de 3 ejes, en una pequeña placa de 0.8” x 0.5”. Se accede a nueve mediciones diferentes mediante una interfaz I2C, dichas mediciones pueden ser usadas para calcular la orientación del sensor. El MinIMU-9 v3 incluye un regulador de voltaje y un circuito que permite operar con un voltaje entre 2.5 y 5.5 V.

Figura 2.19 MinIMU-9 v3 de Pololu.

El L3GD20H y LSM303D tienen muchas opciones configurables, incluyendo sensibilidades seleccionables para el giroscopio,

acelerómetro y magnetómetro. Los dos integrados se puede acceder a través de una interfaz I²C / TWI compartida, lo que permite que los sensores puedan abordarse de forma individual.

Con un algoritmo apropiado, un microcontrolador o un ordenador pueden utilizar los datos para calcular la orientación del MinIMU. El giroscopio puede ser usado para realizar un seguimiento de manera muy precisa de la rotación en un tiempo corto. Los ejes respectivos de los dos integrados están alineados al tablero para facilitar los cálculos de fusión de sensores.

El MiniIMU-9 v3 cuenta con 6 pines para su uso, que se encuentran descritos en la Tabla I:

Tabla I: Pines del MinIMU-9 v3.

PIN	Descripción
SCL	Línea de reloj de la comunicación I2C.
SDA	Línea de datos para comunicación I2C.
GND	Conexión a tierra.
VIN	Fuente de voltaje de 2.5 a 5.5V
VDD	Salida del regulador de voltaje de 3.3V, solo cuando VIN es mayor a 3.3V
SA0	Entrada de nivel lógico de 3.3V para determinar las direcciones de esclavo del I2C.

Para utilizar el MinIMU-9 v3 son necesarios mínimo cuatro pines: SCL, SDA, GND, VIN.

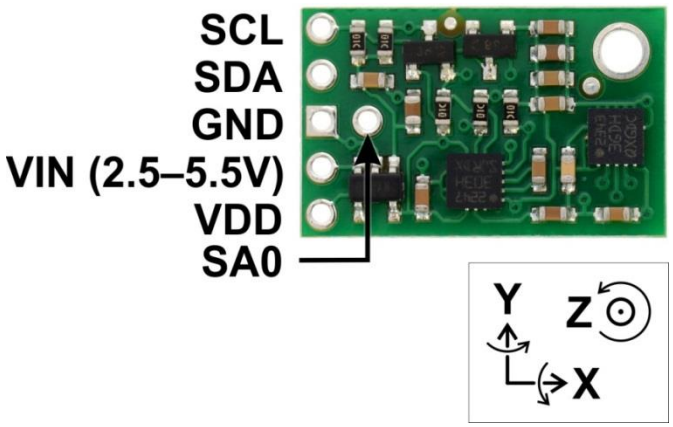


Figura 2.20 Pines del MinIMU-9 v3 de Pololu.

2.6.1 Giroscopio

Un giroscopio es un dispositivo que se utiliza generalmente para la medición de la velocidad angular, los giroscopios pueden medir la velocidad de giro en una, dos o tres dimensiones. Los giroscopios de tres ejes suelen utilizarse con un acelerómetro de tres ejes para proporcionar un sistema de seguimiento de movimiento completo de 6 grados de libertad.

Hay tres tipos básicos de giroscopios:

- Giroscopios rotantes.
- Giroscopios con estructura de vibración.
- Giroscopio óptico.

El sensor inercial MinIMU-9 v3 utiliza un giroscopio de tres ejes, el cual se encuentra encapsulado en el circuito integrado. L3GD20H.

El L3GD20H es un sensor de velocidad de angular de bajo consumo de energía, contiene un elemento de detección basado en una estructura de vibración, y un circuito para proporcionar la velocidad angular a través de una interfaz digital (I2C/SPI).

Entre las principales características del L3GD20H tenemos [10]:

- Alimentación de 2.2 a 3.6V
- Bajo consumo de energía
- Salida de datos de 16 bits
- Interfaz de salida digital I2C/SPI
- Sensor de temperatura incorporado
- Salida de datos de temperatura de 8 bits

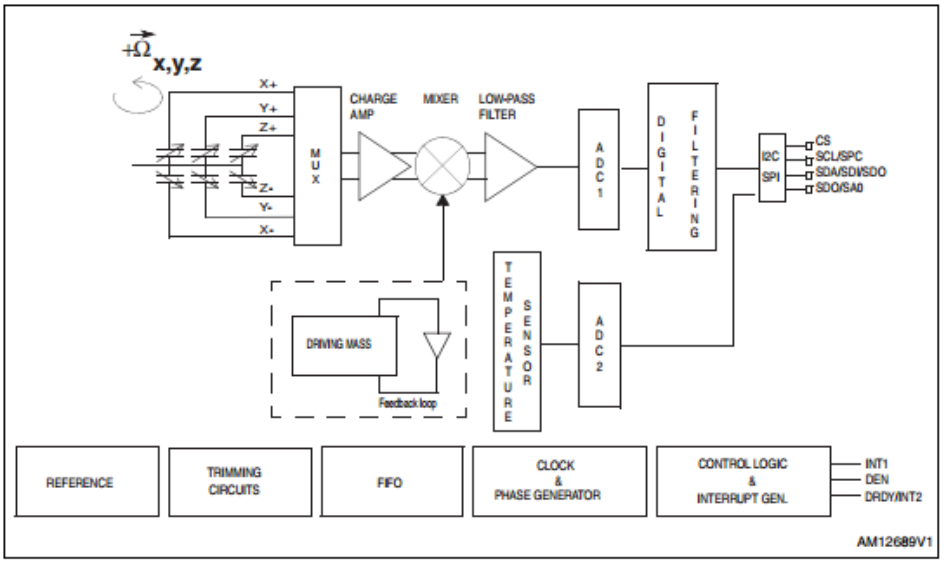


Figura 2.21 Diagrama de bloques del L3GD20H [10].

2.6.2 Magnetómetro

Los magnetómetros son dispositivos que miden la fuerza y/o dirección de un campo magnético. Los magnetómetros que solo miden la fuerza o la dirección son llamados magnetómetros escalares, mientras los que miden ambos son llamados magnetómetros vectoriales. Hoy en día tanto magnetómetros escalares como vectoriales se encuentran en aparatos de consumo, como en tablets y dispositivos móviles. En la mayoría de los casos, los magnetómetros se utilizan para obtener información direccional en tres dimensiones al ser emparejados con acelerómetros y giroscopios.

Por lo general los magnetómetros tienen como elemento sensorial una brújula, que es sensible al campo magnético de la tierra. Además de la brújula, pueden existir magnetómetros escalares en varias otras formas, usando una variedad de técnicas para obtener información sobre un campo magnético. Entre estos métodos alternativos se incluye al magnetómetro de protones, que mide la frecuencia de resonancia de protones en un campo magnético dado.

Es posible encontrar el ángulo del sensor con respecto al campo magnético de la tierra en un magnetómetro escalar de tres ejes, debido a que las tensiones de salida V_x y V_y varían como funciones del seno y coseno de dicho ángulo. El valor instantáneo del ángulo se obtiene con la operación arcotangente:

$$\Theta = \text{arctg}(\frac{V_x}{V_y})$$

El sensor MinIMU tiene incorporado un magnetómetro de tres ejes, el cual junto a un acelerómetro se encuentra encapsulado en un circuito integrado LSM303D, del cual se hablará con más detalles en la siguiente sección.

2.6.3 Acelerómetro

El acelerómetro es uno de los sensores inerciales más comunes, pueden medir la aceleración en uno, dos o tres ejes ortogonales. Se utilizan típicamente en uno de estos tres modos:

- Como una medición inercial de la velocidad y la posición.
- Cómo un sensor de inclinación en dos o tres dimensiones, a partir de la aceleración de la gravedad.
- Como un sensor de vibración o impacto.

La mayoría de acelerómetros son sensores micro-electro-mecánicos (MEMS). El principio básico de funcionamiento es el desplazamiento de una pequeña masa de prueba, sobre la superficie de silicio de un circuito integrado y suspendido por soportes pequeños. De acuerdo con la segunda ley de Newton (F=ma), como una aceleración se aplica al dispositivo, se desarrolla una fuerza que desplaza la masa. Los soportes actúan como un resorte, y el fluido (generalmente de aire) dentro del integrado actúa como amortiguador, resultando en un sistema físico de segundo orden. Esta es la fuente del ancho de banda operativo y la respuesta de frecuencia de los acelerómetros.

Tipos de acelerómetros:

Existen varios principios útiles para la construcción de un acelerómetro. Los dos tipos más comunes utilizan detección capacitiva y el efecto piezoeléctrico, para detectar el desplazamiento de la masa de prueba, proporcional a la aceleración aplicada. Hay otros tipos de acelerómetros como: piezoresistivos, de efecto Hall y de transferencia de calor.

El MinIMU-9 v3 tiene incorporado un acelerómetro de tres ejes, que se encuentra encapsulado en el circuito integrado LSM303D.

El LSM303D es un sistema empaquetado que contiene un sensor de aceleración lineal de tres ejes y un sensor magnético digital de 3 ejes.

Entre las principales características del LSM303D tenemos [11]:

- Escala magnética seleccionable de ±2/ ±4/ ±8/ ±12 gauus.
- Escala de aceleración lineal seleccionable de ±2/ ±4/ ± 6/ ±8 ±16 g.
- Salida de datos de 16 bits.
- Interfaz serial I2C/SPI
- Voltaje de alimentación de 2.6 a 3.6 V.
- Generadores de interrupción programables para: caída libre, detección de movimiento y detección de campo magnético.

- Sensor de temperatura integrado.

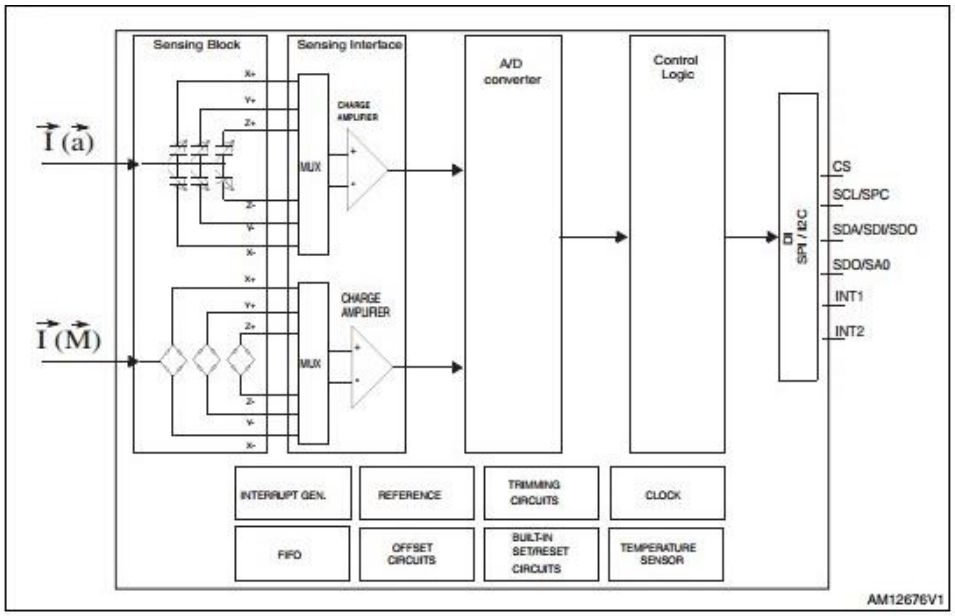


Figura 2.22 Diagrama de bloques del LSM303D [11].

2.7 Sensores de Posición

Los sensores de posición son aquellos que miden la posición lineal o angular con respecto a un punto de referencia. Pueden ser utilizados para medir distancia, desplazamiento, o simplemente detectar la presencia o ausencia de un objeto.

Existen muchos tipos de sensores de posición, pero el presente proyecto hará uso de los siguientes: codificadores rotatorios y arreglo de sensores de luz infrarroja.

2.7.1 Codificadores Rotatorios

Un codificador rotatorio, comúnmente llamado encoder, es un interruptor electromecánico utilizado como un sensor de posición angular. Su salida es generalmente una codificación digital de la posición relativa o absoluta. Típicamente, el codificador rotatorio está acoplado con un microprocesador y se puede encontrar en aplicaciones industriales (control de motores).

Existen dos tipos de encoders: absolutos e incrementales.

Codificador Rotatorio Absoluto: Determina la posición angular absoluta. Cada posición del codificador es única, por lo tanto la posición no se perderá si el microcontrolador falla. La resolución del tipo codificador rotatorio absoluto está determinada por el número de bits en la salida, por ejemplo, un encoder de 8 bits tendrá 256 posiciones únicas por vuelta. Para evitar problemas en la transición, utiliza codificación Gray, estos codificadores rotatorios son más caros que los incrementales y requieren más ancho de banda para su comunicación, pero no requieren reinicialización si ocurre un corte de energía, ya que no se pierde la posición.

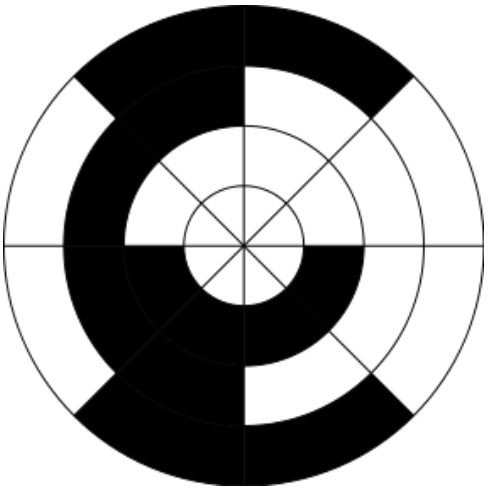


Figura 2.23 Configuración Gray de codificador rotatorio absoluto de 3 bits.

Codificador Rotatorio Incremental: Se utiliza para medir el cambio en la posición angular. Los requisitos de ancho de banda son mucho menores que el codificador rotatorio absoluto, debido a que solo determina la dirección y el cambio en el ángulo. Esto se logra con dos salidas digitales en cuadratura.

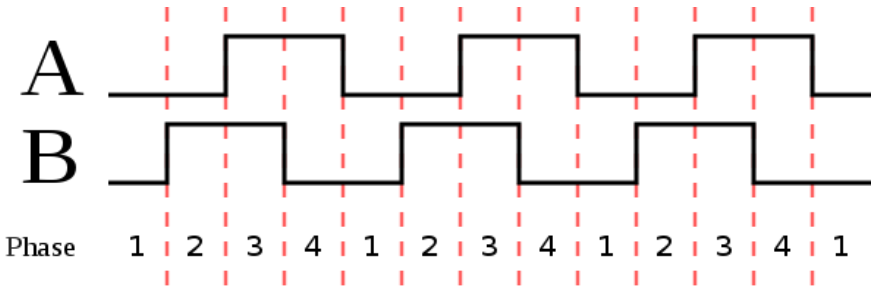


Figura 2.24 Diagrama de cuadratura de un codificador rotatorio incremental.

Existen varias tecnologías de implementar un codificador rotatorio:

- **Conductora.**
- **Óptica.**
- **Magnética.**

Este proyecto utiliza codificadores rotatorios incrementales para obtener datos de la velocidad en que giran los motores, dichos codificadores son los IE2-512 de Faulhaber y se encuentran integrados en el motor.

2.7.2 Arreglo de Sensores de Luz Infrarroja

En un robot velocista es importante mantener la posición del robot tal que la pista se encuentre en el medio del eje transversal del mismo. Para detectar la posición de la pista con respecto al robot se utiliza un arreglo de sensores de luz infrarroja.

Este arreglo nos permite saber la posición exacta de la pista en determinado instante de tiempo, haciendo uso de los sensores infrarrojos que distinguen los negro de lo blanco.

En la actualidad, debido a que las competencias de robot velocistas van en aumento, es posible encontrar algunos de estos arreglos en el mercado.

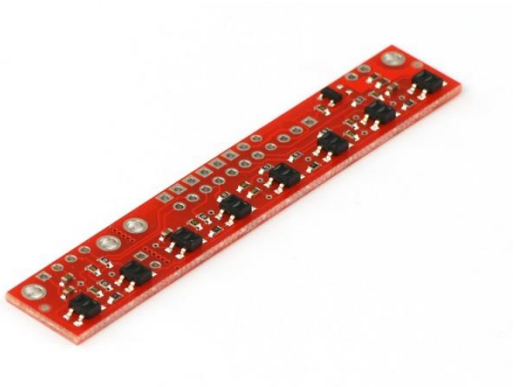


Figura 2.25 Arreglo de sensores de luz infrarroja de Pololu.

Este proyecto hace uso de un arreglo de sensores infrarrojos, el cual usa 14 sensores infrarrojos individuales, los cuales se ubican en una placa. Los sensores usados para aquello son los QTR-1A de Pololu que tienen las siguientes características:

- Voltaje de operación: 3.3V – 5V
- Salida de voltaje analógica
- Distancia óptima para el sensado: 3mm

2.8 Módulo de Comunicación Inalámbrica

Los módulos de comunicación inalámbrica son dispositivos electrónicos pequeños utilizados para transmitir y recibir señales. Por lo general las tecnologías usadas son comunicación mediante señales infrarrojas o comunicación mediante señales de radiofrecuencia, especialmente estas últimas, ya que no requieren línea de vista.

Los módulos de comunicación inalámbrica comúnmente utilizan protocolos definidos dentro de las comunicaciones, como Zigbee, Bluetooth o Wi-Fi.

Módulo Bluetooth HC-06

El módulo de comunicación inalámbrica HC-06 es un dispositivo de comunicación que usa la tecnología Bluetooth. Es un módulo práctico, ya que es fácil de usar y barato, además que opera de manera eficaz.

Entre las principales características del módulo inalámbrico HC-06 tenemos [12]:

- Protocolo Bluetooth: Bluetooth Specification v2.0+EDR
- Frecuencia: 2.4 GHz
- Modulación: GFSK
- Potencia de emisión: 4dBm
- Sensitividad: -84dBm

- Seguridad: Autenticación y cifrado
- Perfil: Puerto Serial Bluetooth
- Fuente de alimentación: 3.3V, 50mA
- Dimensiones: 26.9mm x 13mm x 2,2 mm

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN

Para la construcción del robot es necesario trabajar en tres aspectos, el diseño físico o mecánico del robot, el diseño electrónico y el diseño de software. Primero se tratará sobre el hardware, esto incluye el aspecto mecánico y electrónico del robot, luego se detalla los aspectos relacionados al software, empezando desde la creación de la computadora sobre la plataforma FPGA, el Soft-Core Processor junto a sus periféricos, hasta el diseño de los algoritmos de control y filtrado usando MATLAB y su implementación sobre el procesador.

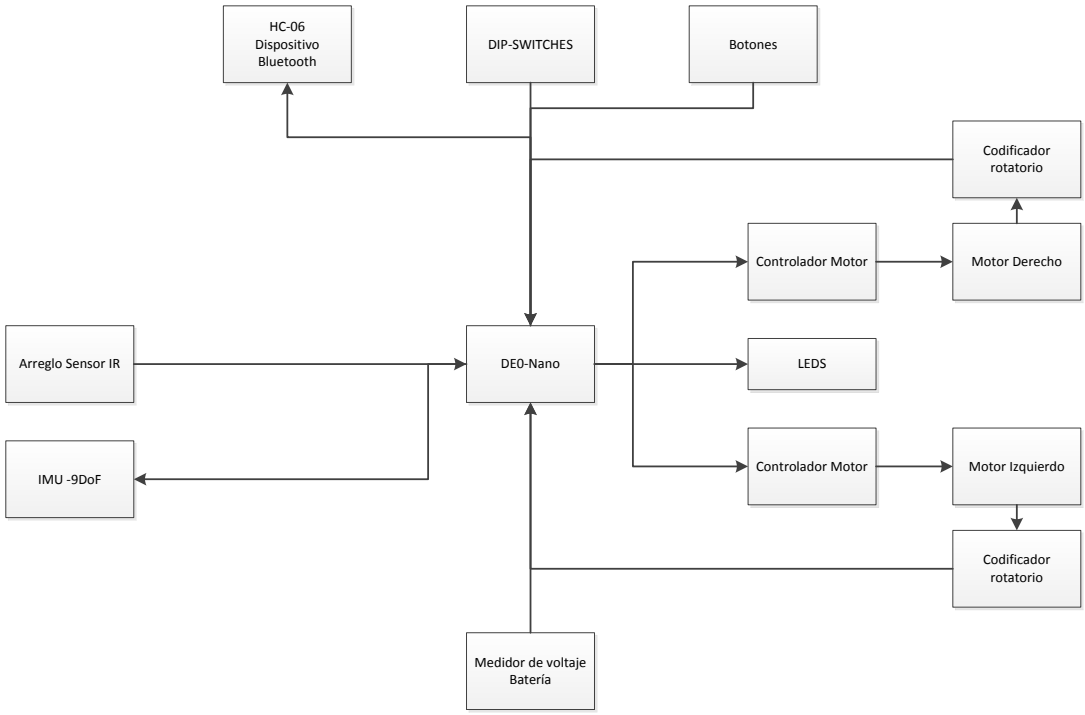


Figura 3.1 Diagrama de Bloques del Robot.

En la Figura 3.1 se muestra el diagrama de bloques del sistema que se describe en este capítulo.

3.1 Hardware

En esta sección se detalla los aspectos relacionados al hardware del robot. Primero se escribe sobre la mecánica del robot, se explica sobre las magnitudes físicas que influyen en el comportamiento del robot y el comportamiento de los motores.

Luego se describe el circuito electrónico adicional a la tarjeta DE0-Nano diseñado para interconectar los sensores y actuadores y para energizar todo el sistema.

3.1.1 Mecánica del Robot

Este proyecto no se enfoca en el área de diseño mecánico, pero es necesario tener noción de la física que rige el comportamiento del robot de tal manera que el diseño del robot beneficie en lo posible el desempeño de los algoritmos de control y por consiguiente del robot.

Para esto se necesita conocer sobre la cinemática, la dinámica del robot y el comportamiento de los motores de corriente continua, estos últimos son sistemas parte mecánicos y parte eléctricos, pero se los describe en esta sección.

Existen varias formas de locomoción para robots móviles, en el caso de los robots velocistas, las formas más usadas son la de tipo triciclo y la locomoción diferencial de dos y cuatro ruedas.

En este proyecto se usó la locomoción de tipo diferencial de dos ruedas, principalmente porque es eficaz y presenta la menor dificultad de implementación en el aspecto mecánico.

Cinemática Diferencial

La locomoción diferencial que se describe es la de dos ruedas. En este tipo de locomoción se tiene dos ruedas fijas controladas, una a cada lado del robot, cada rueda es manejada de forma independiente, de esta manera el robot puede cambiar su dirección variando la velocidad de giro de cada rueda. Para balancear el robot se suele añadir ruedas adicionales, esféricas o ruedas orientables que giran en cualquier dirección.

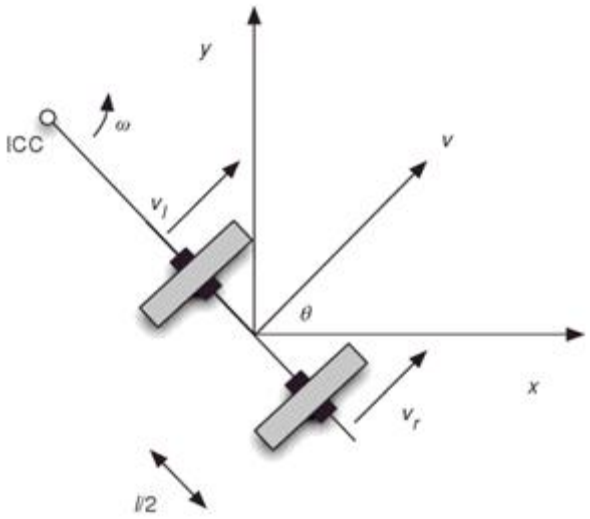


Figura 3.2 Cinemática diferencial [13].

Dadas las velocidades lineales en el punto de contacto de las ruedas del robot con la pista, se puede definir la velocidad angular del robot respecto a un punto llamado centro instantáneo de

curvatura, el cual se encuentra en la línea formada por la unión de los dos centros de las ruedas, como se observa en la Figura 3.2. De las ecuaciones de cinemática se tiene que la velocidad lineal en el punto de contacto de cada rueda es:

Ecuación 3. 1

$$v_l = \omega\left(R - \frac{l}{2}\right)$$

Ecuación 3. 2

$$v_r = \omega\left(R + \frac{l}{2}\right)$$

v_l: Velocidad lineal en el punto de contacto de la rueda izquierda.
v_r: Velocidad lineal en el punto de contacto de la rueda derecha.
ω: Velocidad angular del robot.
R: Radio de curvatura instantáneo.
l: Distancia entre las ruedas del robot.

Entonces el radio de curvatura instantáneo y la velocidad angular del robot son:

Ecuación 3. 3

$$\mathbf{R} = \left(\frac{\mathbf{l}}{\mathbf{2}}\right)\frac{\mathbf{v_r} + \mathbf{v_l}}{\mathbf{v_r} - \mathbf{v_l}}$$

Ecuación 3. 4

$$\boldsymbol{\omega} = \frac{\mathbf{v_r} - \mathbf{v_l}}{\mathbf{l}}$$

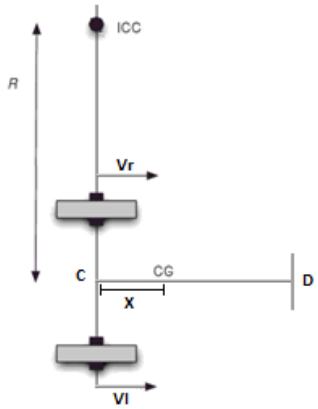


Figura 3.3 Esquema Robot Velocista

Además se puede encontrar la velocidad del centro de masa del robot. Debido a la simetría del robot el centro de masa se encuentra sobre la línea CD de la Figura 3.3, si la distancia del centro de masa al punto C es x, la velocidad v_x en este punto es:

Ecuación 3. 5

$$\mathbf{v_x = \omega\sqrt{R^2 + x^2}} = \sqrt{(\frac{\mathbf{v_r + v_l}}{\mathbf{2}})^2 + (\frac{\mathbf{v_r - v_l}}{\mathbf{l}}\mathbf{x})^2}$$

Para el caso en que X es cero:

Ecuación 3. 6

$$\mathbf{v_0 = \omega R} = \frac{\mathbf{v_r + v_l}}{\mathbf{2}}$$

La Ecuación 3. **4** y Ecuación 3. **6** son importantes para el control del robot, esto se detalla más adelante en la sección 3.2.4.

Dinámica

En esta sección se analiza las fuerzas que actúan sobre el robot e influyen en su movimiento. La descripción completa de la dinámica del robot es compleja, influyen factores como la deformación de la rueda, la forma de la rueda, la forma del robot, su momento de inercia, resistencia aerodinámica, resistencia a la rodadura, varios factores hacen que el análisis vaya más allá de lo que pretende este proyecto. Pero se presenta un análisis de situaciones específicas que ayudan a entender el comportamiento del robot y mejorar su diseño físico.

Lo primero que se analiza es cómo el robot acelera o empieza a moverse cuando avanza en línea recta. En esta situación, suponiendo que las ruedas sujetas a los motores no se deslizan sobre el suelo, las fuerzas que actúan sobre él son dos principalmente, la fuerza de fricción estática, fuerza que hace que el robot acelere y actúa en las dos ruedas fijas del robot, y la fuerza de fricción cinética provocada por un deslizamiento de la rueda esférica delante del robot, Figura 3.4, esta fuerza aparece cuando la rueda esférica no gira sino que se arrastra sobre la superficie.

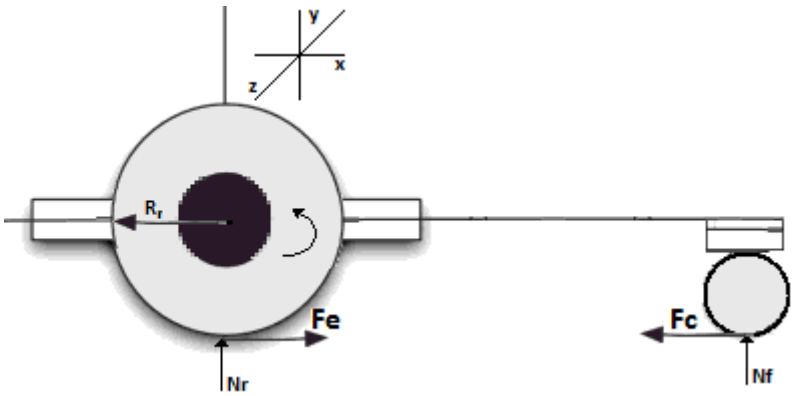


Figura 3.4 Diagrama de Cuerpo Libre, Eje X.

Del diagrama de cuerpo libre, Figura 3.4, sobre el eje X se tiene:

$$\sum \mathbf{F_x} = \mathbf{ma}$$

Ecuación 3. 7

$$\mathbf{2F_e - F_c = ma}$$

$$\mathbf{F_e \leq F_{e_{máx}}}$$

$$\mathbf{F_{e_{máx}} = \mu_e N_r}$$

$$\mathbf{F_c = \mu_c N_f}$$

Además para que se cumpla la condición de no rozamiento entre las ruedas fijas y el suelo:

Ecuación 3. 8

$$\mathbf{a = \alpha_r R_r}$$

$\mathrm{F_e}$: Fuerza de fricción estática
$\mathrm{F_c}$: Fuerza de fricción cinética
m: : Masa del robot
a : Aceleración linealdel robot
α_r: Aceleración angular de las ruedas fijas
$\mathrm{R_r}$: Radio de l*as* ruedas $fijas$
μ_e: Coeficiente de fricción estático
μ_e: Coeficiente de fricción cinético
$\mathrm{N_f}$: Fuerza normal sobre la rueda delantera.
$\mathrm{N_f}$: Fuerza normal sobre cada rueda trasera.

Si se cumple la condición de no rozamiento, el movimiento del robot depende exclusivamente del comportamiento del motor. Es decir la fricción estática toma el valor necesario para que la aceleración angular de las ruedas coincida con la aceleración lineal del robot, siempre que este valor que deba tomar no supere la cantidad $\mathrm{F_{e}}_{máx}$, que es la fricción estática máxima.

Entonces de esta primera situación lo que se debe tener en cuenta es que, en el modelo utilizado, la aceleración del robot en línea recta depende únicamente de parámetros relacionados únicamente con el comportamiento del motor, si no se supera la fricción estática máxima, por lo que según la Ecuación 3. **7** se debe

tratar de que el coeficiente de fricción estático sea alto[2] y que la fuerza normal N_r sea máxima

La siguiente situación que se analiza es el movimiento en dirección al eje Y. Si el robot acelera demasiado sucede que se puede romper el equilibrio y la parte delantera del robot se levantaría. En la Figura 3.5 se observa las fuerzas que intervienen en esta situación.

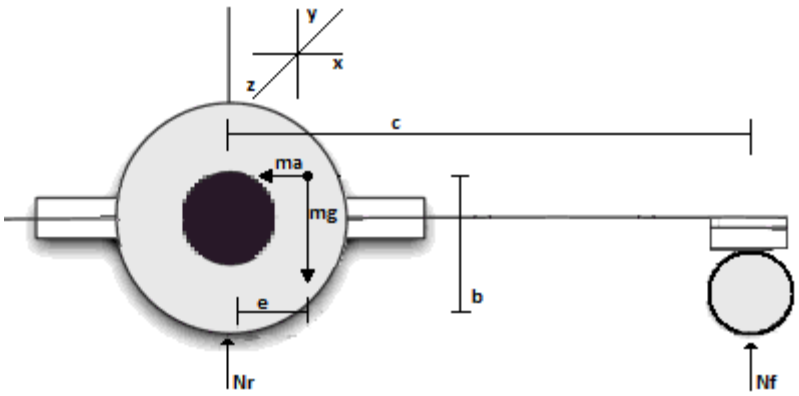


Figura 3.5 Diagrama de Cuerpo Libre, Eje Y.

Para que exista equilibrio de traslación en el eje Y y de rotación alrededor del eje Z, debe suceder que:

$$\sum F_y = 0$$

Ecuación 3. 9

$$2N_r + N_f - mg = 0$$

[2] Comercialmente existen llantas de diferente dureza fabricadas para el automodelismo RC.

$$\sum \tau_z = 0$$

Ecuación 3. 10

$$\mathrm{mge} - \mathrm{F}_a\mathrm{b} - \mathrm{N_f c} = 0$$

En la última ecuación F_a corresponde a una fuerza aparente, esta es igual al producto de la masa del robot por su aceleración y actúa sobre el robot en el centro de masa cuando se experimenta una aceleración, esta se observa en la Figura 3.5.

Como se mencionó, es deseable que el valor de $\mathrm{N_r}$ sea máximo. Según la Ecuación 3. **9** se tiene que para esto $\mathrm{N_f}$ debe ser mínimo, esto ocurre cuando $\mathrm{N_f} = 0$, de la Ecuación 3. **10** :

$$\mathrm{mge} - \mathrm{F}_a\mathrm{b} = 0$$
$$\mathrm{mge} - \mathrm{mab} = 0$$

Ecuación 3. 11

$$\mathrm{a} = \frac{\mathrm{ge}}{\mathrm{b}}$$

Esta aceleración es la que hace que $\mathrm{N_r}$ sea máxima, todo el peso esta soportado por las ruedas fijas sin que el robot se levante, es lo que se asumió en la Ecuación 3. **10** . Si se supera ligeramente esta aceleración el peso del robot aún será soportado totalmente por las ruedas traseras pero este empezará a girar alrededor del eje Z, algo que se debe evitar ya que los sensores infrarrojos fallarían y se tendría un comportamiento errático.

Lo que se debe observar es que, según la Ecuación 3. 11 , para que esta aceleración sea máxima se necesita que el centro de masa esté lo más bajo posible y lo más lejos de las ruedas fijas. Aunque esto último no es recomendable, ya que se aumenta el momento de inercia del robot.

Además, de la Ecuación 3. **10** , se tiene que en general la fuerza normal en la rueda delantera está dada por:

$$N_f = \frac{mge - mab}{c}$$

Por lo que otro detalle importante es que para que N_f sea mínima y que la rueda delantera soporte el menor peso posible, se necesita que c sea máxima, es decir se necesita que la rueda esférica este lo más lejos posible de las otras dos ruedas.

Otra situación que se analiza es cuando el robot empieza a rotar alrededor del centro instantáneo de curvatura (ICC), como se muestra en la Figura 3.6:

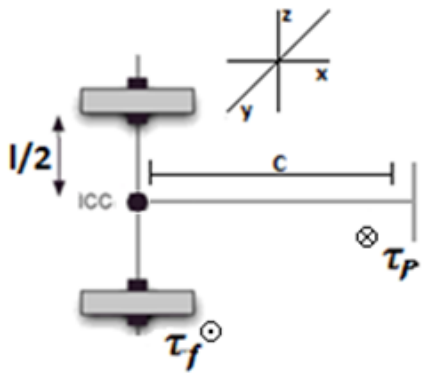


Figura 3.6 Torque Sobre El Eje Z.

Para que el robot empiece a girar se necesita que se ejerza un torque neto diferente de cero alrededor del eje Y, se tiene que:

$$\sum \tau = I_{ICC}\alpha$$
$$\tau_f - \tau_p = I_{ICC}\alpha$$

Ecuación 3. 12

$$\tau_f = I_{ICC}\alpha + \tau_p$$
$$\tau_{femáx} = 2\mu_e N_r r$$

Donde I_{ICC} corresponde al momento de inercia del robot respecto al centro de curvatura, τ_f corresponde al torque de la fricción estática y τ_p es el torque debido a la fricción cinética entre la rueda esférica y el suelo.

Una vez más la fuerza de fricción estática toma el valor necesario para que se cumpla la condición de no rozamiento en las ruedas, siempre que no se supere el valor máximo de fricción.

Entonces aquí lo importante de notar es que para evitar superar el valor máximo, según la Ecuación 3. **12**, lo más conveniente es que el valor del momento de inercia sea el menor posible, es decir se debe evitar que la masa este distribuida lejos al centro de curvatura.

Pero sucede que el centro de curvatura no es punto fijo, sino que varía según el radio de giro, por lo que el momento de inercia varía también. Usando el teorema de Steiner el momento de inercia alrededor del centro de curvatura es:

Ecuación 3. 13

$$I_{ICC} = I_{CG} + m\sqrt{R^2 + x^2}$$

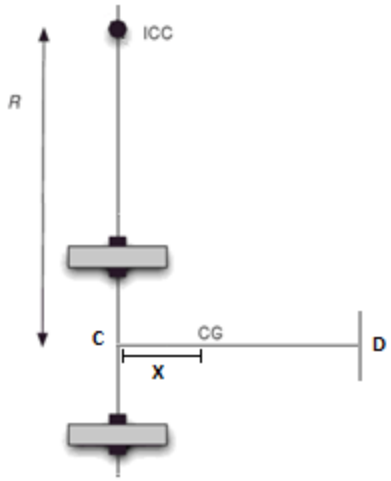


Figura 3.7 Centro de Curvatura y Centro de gravedad.

El valor de I_{ICC} es el momento de inercia respecto al centro de curvatura, I_{CG} es el momento de inercia respecto al centro de

gravedad y m corresponde a la masa del robot. Las distancias R y x se muestran en la Figura 3.7.

La Ecuación 3. **12** relaciona el momento de inercia de un punto variable como el centro de curvatura con el de un punto fijo como el centro de gravedad.

Se puede concluir que para que el momento de inercia respecto al centro instantáneo de curvatura sea mínimo se necesita que el momento de inercia respecto al centro de gravedad sea mínimo y que la distancia x sea lo más cercana a cero posible, lo mejor sería que el punto C de la Figura 3.7 coincida con el centro de gravedad. No se puede hacer nada con respecto al valor de R, ya que depende del radio de giro.

Motores DC de imán permanente

Las leyes que rigen el comportamiento de los motores de corriente continua son parte mecánicas y parte eléctricas, las ecuaciones son las siguientes:

Ecuación 3. 14

$$\frac{di_a}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a} \omega_r + \frac{1}{L_a} u_a$$

Ecuación 3. 15

$$\frac{d\omega_r}{dt} = \frac{k_p}{J} i_a - \frac{B_m}{J} \omega_r - \frac{1}{J} T_L$$

u_a es la tensión de alimentación del rotor.

i_a la corriente que va a circular por el rotor o corriente de armadura.
r_a la resistencia del bobinado del rotor.
L_a la inductancia del bobinado del rotor.
k_a constante de fuerza contra-electromotriz.
k_p constante de torque electromagnético.
ω_r la velocidad angular de giro a la cual trabaja el rotor.
J el momento de inercia equivalente del eje rotor con la carga.
B_m el coeficiente de rozamiento viscoso.
$T_L$es un torque de carga, o torque externo al motor.

Un motor se puede analizar como un sistema dinámico de segundo orden, sus variables de entrada son el voltaje y el torque de carga externo, sus variables de estado son la corriente y la velocidad angular.

Además se observa que existen varios parámetros que modifican el comportamiento del sistema, propios de las características eléctricas y mecánicas del mismo.

En estado estable ω_r es constante, por lo que $\frac{d\omega_r}{dt} = 0$, además si T_L es constante, según Ecuación 3. **15**, i_a es también constante, por lo que la velocidad angular final con torque de carga constante es:

Ecuación 3. 16

$$\omega_r = \frac{k_p u_a - r_a T_L}{k_a k_p + B_m r_a}$$

De la Ecuación 3. **16** se tiene que, el voltaje de entrada y el torque externo influyen en la velocidad angular en estado estable. Los

parámetros k_a, k_p, B_m, r_a influyen también pero estos son propios del motor y no pueden ser modificados.

El momento de inercia es otro parámetro a considerar, ya que su valor incluye el de la carga mecánica. Este parámetro no influye en la velocidad final, pero sí en la aceleración angular, es decir mientras menor sea el momento de inercia equivalente más rápido se alcanza la velocidad final [14].

El motor debe poner en movimiento la rueda del motor junto con el robot, se puede intuir que no será lo mismo mover la rueda libremente, que mover la rueda sobre el suelo desplazando al robot. Esta carga es la carga equivalente que soporta el motor.

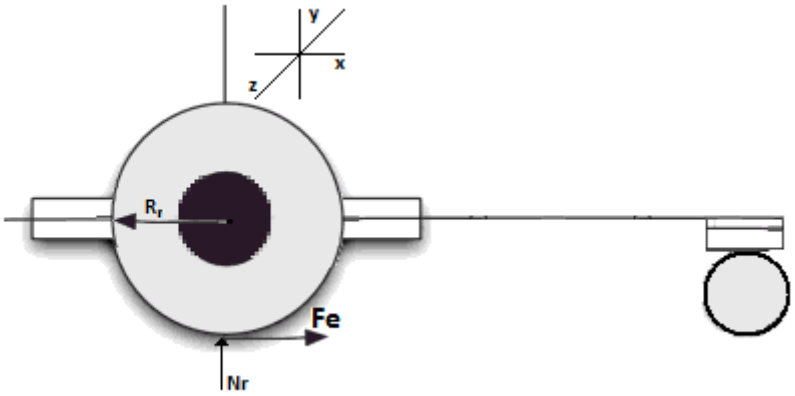


Figura 3.8 Carga equivalente al avanzar de frente.

Ahora se va a analizar la carga equivalente del motor cuando este se mueve de frente, en dirección al eje X, Figura 3.8.

De la Ecuación 3. **15** se tiene que:

$$J\frac{d\omega_r}{dt} = k_p i_a - B_m \omega_r - T_L$$

El valor de J corresponde al momento de inercia del eje del motor junto con la rueda.

Si se cumple la condición de no rozamiento entre la rueda y la superficie se tiene que el torque adicional que actúa sobre el motor es:

$$T_L = F_e R_r$$

Donde F_e es la fricción estática entre el suelo y la rueda y R_r es el radio de la rueda.

Además de la Ecuación 3. **7** se tiene:

$$2F_e - F_c = ma$$

Para simplificar el análisis se va a suponer que la fricción cinética provocada por la rueda esférica es pequeña ($F_c = 0$) entonces:

$$F_e = \frac{1}{2}ma$$

$$T_L = \frac{1}{2}maR_r$$

$$T_L = 0.5m\frac{d\omega_r}{dt}{R_r}^2$$

Reemplazando esta última ecuación en la Ecuación 3. **15** se tiene:

$$(J + 0.5mR_r{}^2)\frac{d\omega_r}{dt} = k_p i_a - B_m \omega_r$$

Por lo que el momento de inercia equivalente en cada motor es:

Ecuación 3. 17

$$J_{eq} = J + 0.5mR_r{}^2$$

Esta última ecuación dice que se debe mantener la masa del robot y el radio de la rueda lo más pequeña posible para disminuir el momento de inercia equivalente sobre el motor cuando se avanza de frente.

Se puede hacer un análisis similar cuando el robot tiene que rotar. La situación que se analiza es cuando el robot gira alrededor del centro de curvatura ubicado a la mitad de la línea que une las dos ruedas fijas, como se muestra en la Figura 3.9.

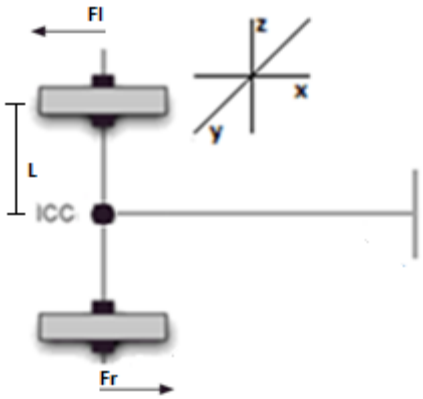


Figura 3.9 Carga equivalente al girar.

De la Ecuación 3. **12** se tiene:

$$\tau_f - \tau_p = I_{ICC}\alpha$$

Se va a suponer que el torque ocasionado por la fricción cinética en la rueda esférica es pequeño $\tau_p = 0$, además el torque de la fricción estática es dos veces el torque en cada rueda $\tau_f = 2\tau_r$, entonces:

$$\tau_r = \frac{1}{2} I_{ICC}\alpha$$
$$\tau_r = \frac{1}{2} I_{ICC}\frac{a}{L}$$

La aceleración lineal en el punto de contacto de la rueda es a, la distancia entre la rueda y el centro de curvatura es L, esta se especifica en la Figura 3.9. La aceleración a se puede relacionar con la aceleración angular de la rueda:

$$\tau_r = \frac{1}{2} I_{ICC}\frac{d\omega_r}{dt}\frac{R_r}{L}$$

Reemplazando esta última ecuación en la Ecuación 3. **15** se tiene:

$$(J + 0.5 I_{ICC}\frac{R_r}{L})\frac{d\omega_r}{dt} = k_p i_a - B_m\omega_r$$

Por lo que el momento de inercia equivalente en cada motor es:

Ecuación 3. 18

$$J_{eq} = J + 0.5 I_{ICC} \frac{R_r}{L}$$

De esta ecuación se puede concluir que, en esta situación, disminuir el radio de la rueda disminuye el momento de inercia equivalente sobre cada motor. Aumentar el valor de L no necesariamente disminuye la carga equivalente ya que el momento de inercia I_{ICC} aumenta cuando la distancia L aumenta.

Para finalizar esta sección se menciona que, los motores usados para el robot son FAULHABER 2224 SR06, estos motores tienen un voltaje nominal de 6V, alcanzan una velocidad angular sin carga de 8200 RPM y tienen un torque de parada de 21.2 mNm, además varios de los parámetros nombrados en el modelo matemático están detallados en la hoja de datos del motor [13].

3.1.2 Diseño electrónico del robot

Para conectar el módulo NIOS II a los diferentes actuadores del robot, como lo son sensores, motores, módulos de conexión inalámbrica, suministro de energía, etc.; es necesario el respectivo circuito electrónico, que sea capaz de que el sistema tenga el mejor rendimiento posible.

Fuente de Alimentación y Reguladores

Para proveer de energía al robot velocista, se hace uso de una batería de polímero de litio más conocidas como LiPo, las cuales son las más usadas en el mundo de la robótica debido a sus ventajas con respecto a otro tipo de baterías, como son: tamaño reducido, mayor capacidad de carga, buena tasa de descarga. Debido a los voltajes nominales requeridos por los diferentes componentes del robot, la batería usada fue de 7.4V, esto debido a que partes del circuito requieren voltajes de 5V y 3.3V, valores que es posible llegar de una manera sencilla con una batería de 7.4V usando los respectivos tipos de reguladores; además el motor de corriente continua usado , tiene como voltaje nominal 6V, un valor para el cual trabajando con la batería de 7.4V, no habría ningún problema en su funcionamiento al proveer señales PWM adecuadas.

Debido a que la mayoría de componentes requieren alimentación de 5V o 3.3V como valores nominales, se usan dos reguladores.

El primero de ellos es un regulador de 5V con una corriente de salida de 5A, antes de la selección de este regulador hubo pruebas con reguladores de 5V convencionales, pero con una corriente de salida de 1A, con resultados no favorables, ya que el

circuito demandaba una corriente mucho mayor. En este caso se optó por el regulador D24V50F5, con una salida de hasta 5A, es suficiente para abastecer cada componente del robot.

Entre las características principales de este regulador tenemos:

- Voltaje de entrada: 6V a 38V
- Voltaje de salida: 5V
- Corriente de salida máxima: 5A
- Tiene protección contra voltaje inverso, cortocircuito, sobretensión, además cuenta con un apagado por sobrecalentamiento.
- Eficiencia de 85% a 95%
- Un consumo de corriente de 700uA cuando no tiene carga.

Figura 3.10 Pines del D24V50F5.

En la Figura 3.10 se puede observar los diferentes pines de conexión del regulador, aparte de los pines que siempre forman parte de cualquier regulador, como lo son Vin, Vout y Gnd, se puede encontrar el pin Enable, que sirve para que en caso de que el sistema se encuentre en reposo, reduzca el consumo de corriente aun valor entre 10uA y 20uA.

La eficiencia de este regulador depende tanto del voltaje de entrada, como de la corriente de salida, a continuación se observa un gráfico de como varía la eficiencia.

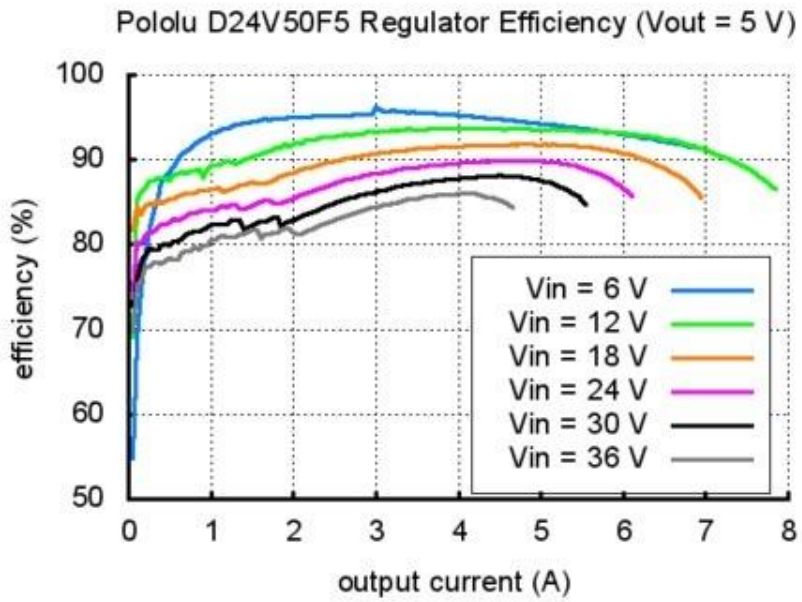


Figura 3.11 Eficiencia del regulador D24V50F5.

El segundo regulador corresponde a un LM 1117-33, el cual es en regulador de 3.3V con una salida máxima de 800mA, que alimenta a los componentes que requieren un voltaje de 3.3V, que por lo general son elementos sensoriales, en los cuales su salida debe sujetarse a los parámetros de entrada de la tarjeta del módulo NIOS II.

Este regulador viene en un encapsulado SMD, además requiere de dos capacitores de tantalio de 10uF cada uno, para su correcto funcionamiento.

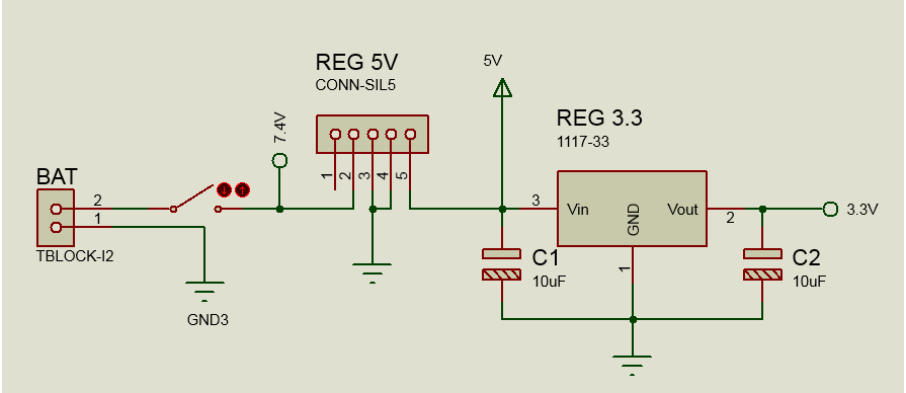


Figura 3.12 Conexión de la batería con los reguladores.

En la Figura 3.12 se pueden apreciar las conexiones de la batería y reguladores, la batería está separada por el resto del circuito a través de un switch, lo que permite conectar o desconectar la batería cuando plazca, el regulador de 5V se alimenta de los 7.4V

de la batería, y este a su vez sirve como entrada para el LM1117-33, debido a que si se alimenta con 7.4V, existe sobrecalentamiento, por lo que no es eficiente.

Conexiones del módulo NIOS II

Dado que la mayoría de los elementos electrónicos del robot van conectados al módulo NIOS II, se procedió a unir estos a los pines más adecuados, tomando como prioridad cercanía física, como ver la mejor manera de evitar cruces en la pista del circuito impreso, a continuación se muestran los pines del módulo que van conectados a los diferentes actuadores.

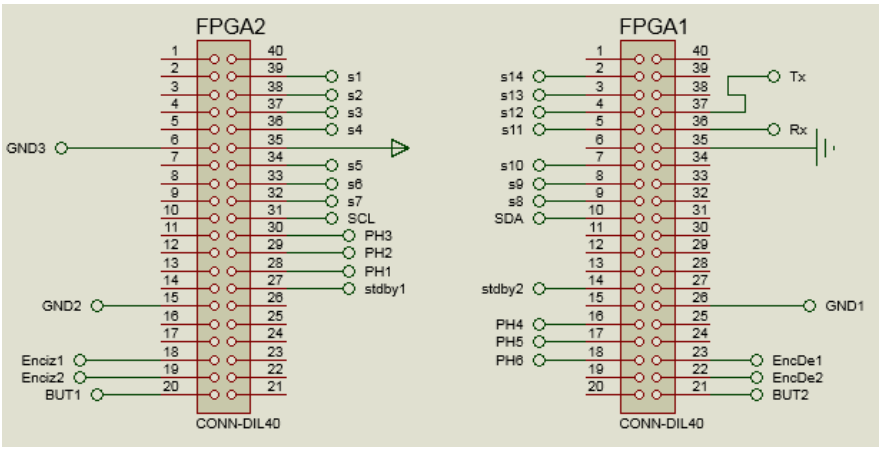


Figura 3.13 Conexión de los pines del módulo DE0-Nano con los diversos actuadores.

Se puede observar que existen varios puntos de GND, esto es porque todos ellos están unidos dentro del módulo, pero si se deja

solo uno de estos puntos, existe dificultad en la conexión de algunos elementos en el circuito impreso, por lo que estos puntos tienen mejor física para conectarlos.

Módulo Bluetooth

Como medida de comunicación entre el robot velocista, ya sea para el envío de datos para análisis del funcionamiento o para enviar órdenes al móvil de manera inalámbrica, se procede el uso del módulo Bluetooth Hc-06. Las conexiones al módulo Bluetooth desde la tarjeta NIOS II, son de manera directa y se pueden observar en la Figura 3.14.

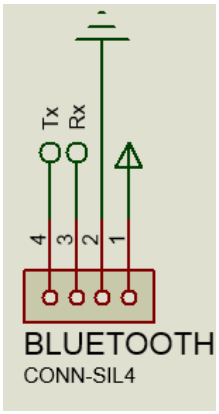


Figura 3.14 Conexión de los pines del módulo Bluetooth.

Los pines Tx y Rx van a los respectivos pines asignados en el módulo NIOS II.

Giroscopio

De la misma manera el giroscopio va se conecta de manera directa al módulo, alimentado por 3.3V, debido a que las señales SCL y SDA deben ser de ese valor. El pin correspondiente a VDD no es usado, ya que corresponde a una salida de 3.3V, pero en este caso se usa el regulador LM1117-33 anteriormente mencionado, la salida 3.3V del giroscopio no abastece con la corriente necesaria requerida por el circuito, a aquellos elementos alimentados con este voltaje.

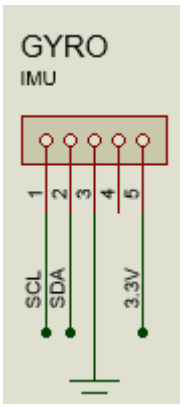


Figura 3.15 Conexión de los pines del módulo de sensores inerciales.

Divisor de voltaje

Para tener información del voltaje de la batería mientras se está descargando, se ha implementado un divisor de voltaje, cuya

salida se envía a un pin del módulo NIOS II, aquí mediante un ADC, se puede conocer el estado actual de la batería.

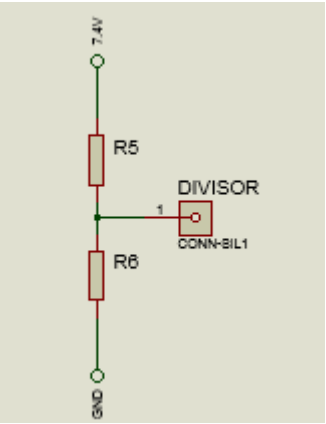


Figura 3.16 Divisor de voltaje de la batería.

En la Figura 3.16, se observa la configuración del divisor de voltaje, el voltaje que da el divisor está dado por:

$$V_{divisor} = \frac{R6}{(R5+R6)} V_{bat},$$

Se puede obtener una relación entre R5 y R6 de la siguiente manera:

$$V_{divisor}(R5 + R6) = V_{bat}\, R6$$

$$V_{divisor} R5 = (V_{bat} - V_{divisor}) R6$$

$$\frac{R5}{R6} = \frac{(V_{bat} - V_{divisor})}{V_{divisor}}$$

Las baterías LiPo con un valor nominal de 7.4V pueden alcanzar un valor de hasta 8.6V en carga completa, dado que el voltaje entregado por el divisor hacia el módulo es de 3.3V máximo. Se tiene que la relación entre R5 y R6 deberá ser de al menos 1.606. En el mercado los valores de resistencias existentes para que entre ellas tenga una relación aproximada a 1.606 son de 8.2k y 5.1k, por lo que a R5 se le asignó 8.2k y a R6 5.1k.

Sensores infrarrojos

Al igual que la mayoría de señales, la señal digital de los sensores infrarrojos va directamente a los pines del módulo FPGA. Estos sensores se encuentran físicamente en una placa distinta a la del módulo NIOS II, para ello se usa un bus de datos que une ambas placas. En la Figura 3.17, se observan los pines que conectan a ambas placas, además que estos van al módulo de control.

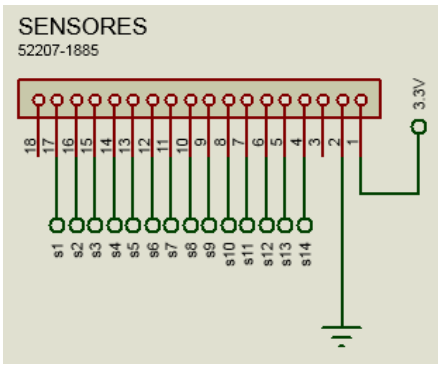


Figura 3.17 Pines de conexión de la placa del arreglo de sensores infrarrojos.

Los sensores tienen una alimentación de 3.3V.

Controlador para motores DC

Para controlar el sentido y velocidad de giro de los motores se necesita un circuito electrónico dedicado. El circuito que suele ser usado para realizar esta función es el llamado Puente H.

Para este proyecto se usó un módulo fabricado por la empresa Pololu. El módulo permite usar las funciones del circuito integrado TB6612FNG y además agrega capacitores de suministro de energía y protección para polarización reversa en la alimentación del motor.

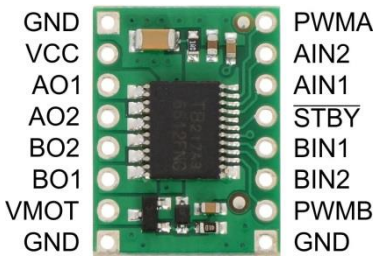


Figura 3.18 Módulo de Pololu con circuito integrado TB6612FNG.

El integrado TB6612FNG posee dos puentes H, sirve para controlar motores de corriente continua de baja corriente. Permite modificar el sentido de giro del motor, posee una entrada de PWM para controlar el voltaje promedio de entrada y permite trabajar en

cuatro modos: CW (Clockwise), CCW (Counterclockwise), Short brake, y Stop o Standby. Los modos de trabajo se muestran en la Tabla II.

Tabla II: Modos de trabajo del módulo puente H.

H-SW Control Function

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Los puentes H del integrado TB6612FNG son basados en MOSFET, son mucho más eficientes que los basados en BJT. Cada puente H permite manejar un motor con una corriente continua máxima por canal de 1A.

La corriente de parada que consume cada motor usado en el proyecto puede superar el valor de 1A, según su hoja de datos, por lo que se utilizó dos puentes H para cada motor.

Para esto se realizó una conexión en paralelo, en la Figura 3.19 se muestra las conexiones para usar en paralelo ambos canales de un solo módulo.

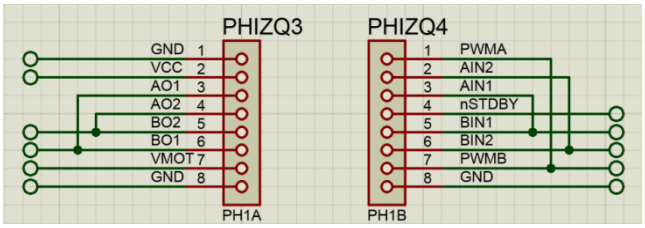


Figura 3.19 Conexión Paralelo del Módulo de Pololu.

Además un dato importante que se especifica en la hoja de datos es que para pasar de los Modos CCW o CW al modo Short Brake o viceversa es recomendable pasar de forma intermedia por el modo Stop o Standby durante un tiempo para evitar que corrientes entren al microcontrolador. Esto se muestra en la Figura 3.20.

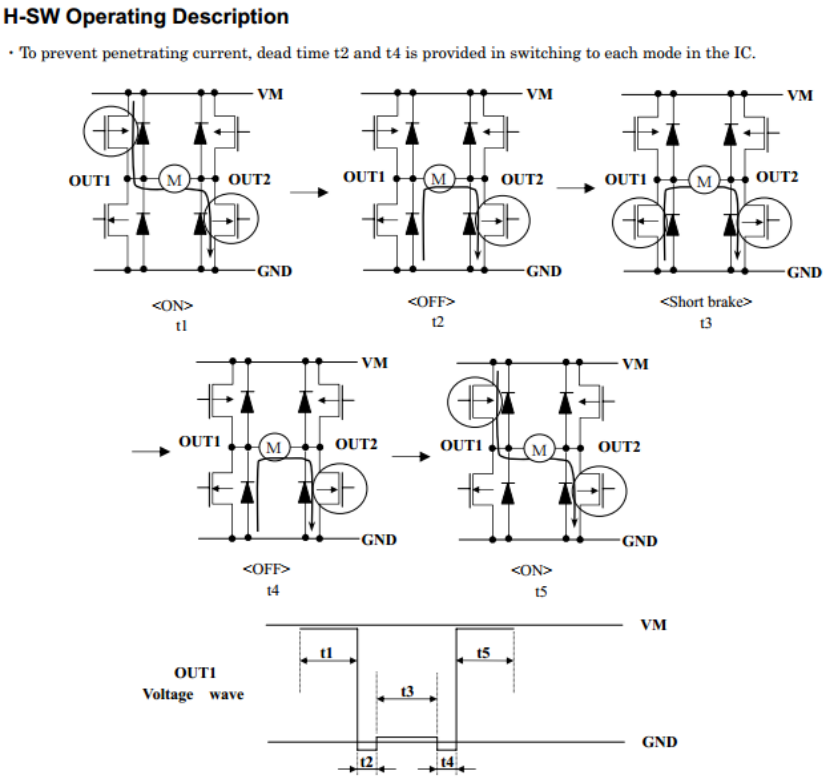


Figura 3.20 Especificaciones para el manejo del integrado TB6612FNG.

En la Figura 3.20 se muestra también la forma de onda de la señal de voltaje que alimenta a los motores, los tiempos t2 y t4 son los que debe estar el integrado en el modo STANDBY, son como mínimo 50ns y 230 ns respectivamente [15]. Para cumplir esta especificación se realizó un bloque en HDL que realiza esta tarea, esto se detalla en la sección 3.2.1.1.

Motores y codificadores rotatorios

Cada motor viene con 6 pines para conexión, los cuales son los siguientes: 2 pines que sirven como alimentación del motor y los 4 pines siguientes corresponden al encoder con que viene integrado el motor, de los cuales 2 sirven para su alimentación, y los otros 2 son las señales enviadas del encoder hacia el módulo NIOS II.

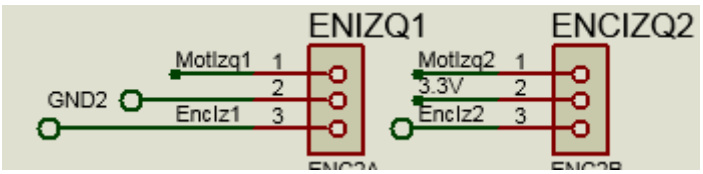


Figura 3.21 Conexiones de los diferentes pines del motor.

En la Figura 3.21, se puede observar la conexión de los seis pines, los pines numerados con el 1 corresponden al suministro de energía hacia estos, los cuales van conectados a las salidas del Puente H respectivo, aparte de los pines de alimentación como son 3.3V y GND2, los pines de la señal de encoder, que tienen

una salida de alto de 3.3V, van conectados a su respectivo punto del módulo NIOS II.

Elementos auxiliares

Como elementos auxiliares, el circuito electrónico cuenta con botoneras y pines de alimentación, las botoneras sirven para dar ciertas indicaciones al robot y los pines de alimentación sirven para que se pueda adaptar algún tipo de sensor que pueda servir para mejorar el desempeño del robot ante alguna tarea específica.

Circuito impreso

Una vez diseñado el circuito electrónico, con todos los elementos debidamente conectados, se procedió al diseño del circuito impreso mediante la herramienta ARES de Proteus.

El circuito impreso fue diseñado sobre dos placas, la placa principal contiene el módulo DE0-Nano, con la mayoría de los componentes del robot y la otra es la placa que contiene los sensores infrarrojos, ambas placas están unidas mediante un cable plano flexible (FFC), que es por donde pasan las señales de los sensores y la alimentación de estos.

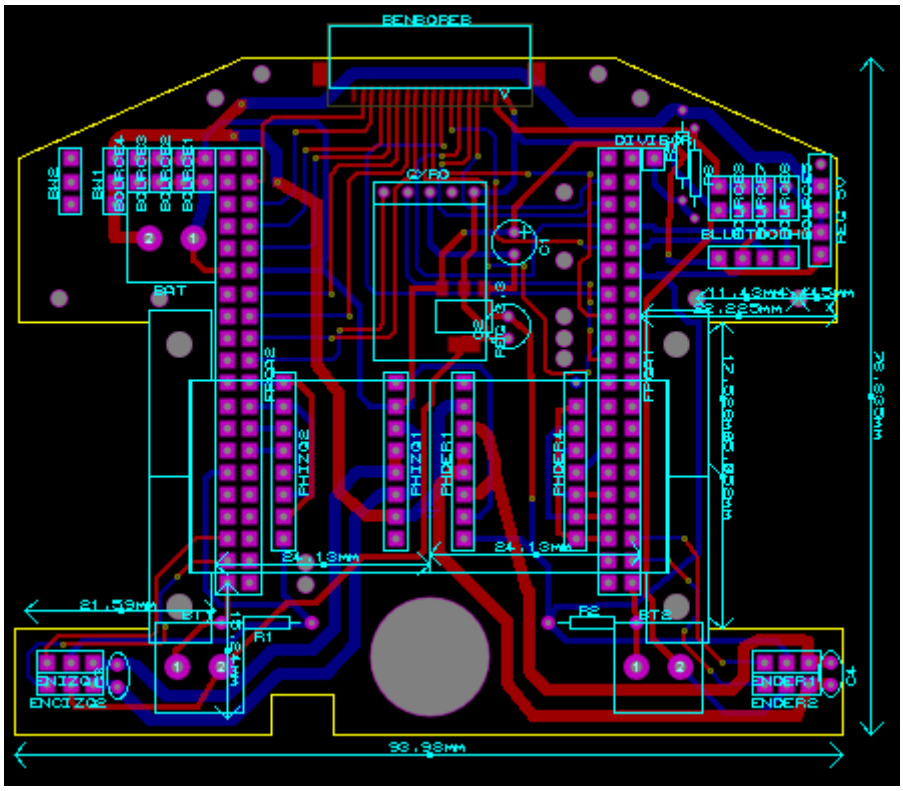


Figura 3.22 Diseño en ARES de la placa principal del robot.

La placa principal tiene dimensiones de 7.9cm x 9.4 cm, su diseño está basado en los requerimientos de un robot velocista, y su forma está pensado en ello, cuenta con dos entradas en las cuales se ubicará parte de las ruedas, además de agujeros para conectores, rueda loca y otros que sirven para ubicar la base del motor.

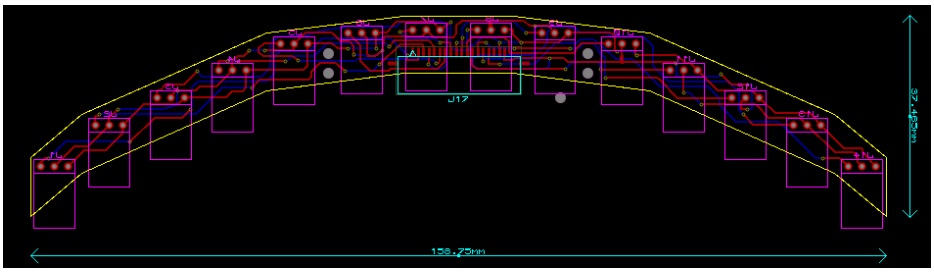


Figura 3.23 Diseño en ARES de la placa de sensores infrarrojos.

La placa que sirve como arreglo de sensores tiene una forma de arco de circunferencia, sus dimensiones son de 15.9cm x 3.7 cm.

3.2 Software

En esta sección se describe el software desarrollado para el manejo del robot. Primero se trata sobre la creación de la computadora usando la herramienta Qsys de Quartus II, se explica sobre el procesador y sus periféricos adicionales.

Además se detalla sobre el código VHDL adicional que fue necesario escribir para el manejo de codificadores rotatorios y del módulo puente H.

Luego se explica sobre los algoritmos de filtrado y de control diseñados usando MATLAB e implementados en software sobre la FPGA en lenguaje C.

En la última sección se da detalles sobre el tiempo en el que se debe ejecutar el algoritmo para un funcionamiento adecuado del sistema.

3.2.1 Creación de la computadora sobre FPGA

El sistema completo se presenta en diagrama de bloques en la Figura 3.24:

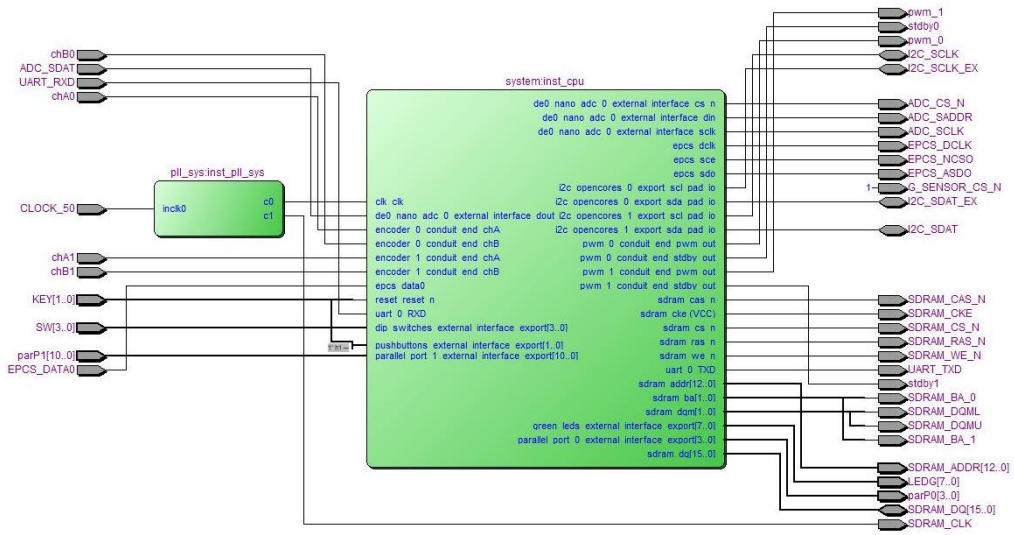


Figura 3.24 Diagrama de Bloques del Sistema Completo sobre la FPGA.

Se observan dos bloques:

El bloque inst_cpu es el computador, es decir el procesador junto a todos los periféricos esclavos, este fue creado usando la herramienta Qsys de Quartus II.

El bloque inst_pll_sys es el lazo de enganche de fase, este amplifica la frecuencia del oscilador de 50 MHz incluido en la DE0-Nano a 100 MHz, el bloque PLL se instanció usando la herramienta MegaWizard Plug-In Manager de Altera

Para construir la computadora se tomó como base la computadora DE0-nano Basic Computer [16] y se realizó algunas modificaciones para adaptarla al proyecto [17] [18].

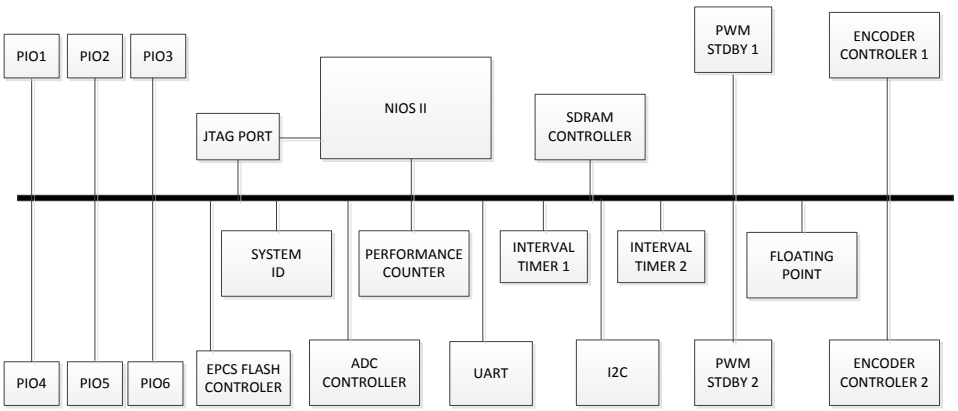


Figura 3.25 Diagrama de Bloques de la computadora sobre la FPGA .

En la Figura 3.25 se muestra el diagrama de la computadora. A continuación se describe cada elemento del sistema, en el caso de que se use más de un módulo del mismo será especificado.

- **Clock Source:**

Es simplemente una interfaz entre la señal de reloj externa, como un oscilador o un PLL, y la señal de reloj que va al sistema, no es

representado como bloque en la Figura 3.25 ya que no es propiamente un bloque, sólo un configuración del reloj. En Qsys, en la ventana de configuración sólo se fija la frecuencia de la fuente de reloj en 100 Mhz.

- **System ID Peripheral**:

System ID previene de tener una discordancia entra las configuraciones de hardware y de software [19].

En la ventana de configuración del módulo sólo se puede especificar un número ID, en este caso se usó el valor hexadecimal 55.

- **NIOS II Processor**:

En la sección 2.1.2 se detalló los aspectos técnicos del procesador NIOS II. El procesador que se escogió para el proyecto es NIOS II/f, se priorizó la rapidez de procesamiento ante el tamaño que ocupa el módulo en la FPGA.

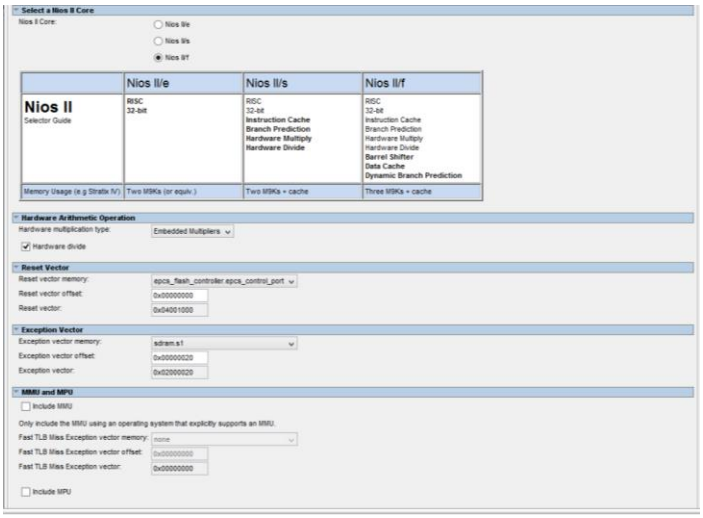


Figura 3.26 Configuración del Core NIOS II.

En la Figura 3.26 se observa parte de la configuración del procesador en Qsys, en esta ventana se especifica que se utilice hardware dedicado para realizar las multiplicaciones y divisiones. Además el vector de excepciones se configura en la memoria SDRAM externa de la DE0-nano y la dirección de reinicio se la coloca en la memoria EPCS, esto último es para que en el reinicio el procesador lea las instrucciones de programa desde la memoria no volátil del computador.

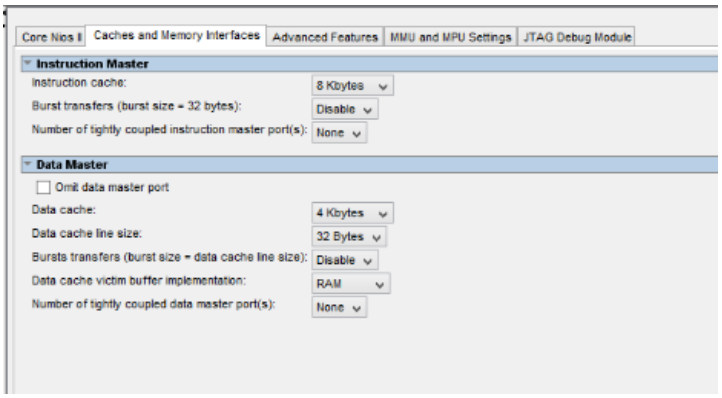


Figura 3.27 Configuración de la interfaces Cache y de Memoria del módulo NIOS II.

También en la Figura 3.27 se muestra la ventana donde se especifica el tamaño de la memoria cache de instrucciones, y el tamaño de la memoria cache de datos, estas se fijan en 8kB y 4 kB respectivamente.

- **EPCS flash controller**

El módulo controlador serial flash EPCS permite acceder a los dispositivos EPCS de Altera [19]. La DE0-Nano incluye una memoria EPCS.

En la ventana de configuración del módulo no se hizo ningún cambio adicional. Para almacenar el código de hardware y software en la memoria flash EPCS se utilizó la utilidad Flash Programmer del IDE de Nios II.

- **JTAG UART**:

Este módulo permite transmitir caracteres de manera serial entre el PC y un sistema Qsys en la FPGA. El PC se puede comunicar con la FPGA a través de cualquier cable USB de descarga [19].

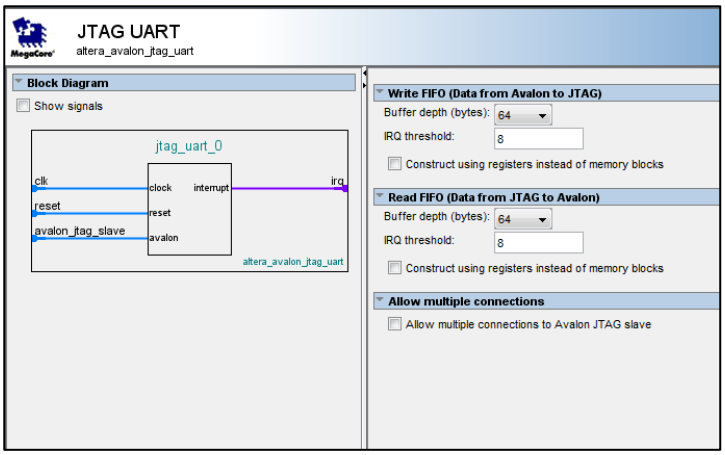


Figura 3.28 Configuración del módulo JTAG UART.

En la Figura 3.28 se muestra la ventana de configuración. En esta se especifica los tamaños de los buffer de entrada y salida de datos y se especifica el umbral en el que se notifica al procesador que el buffer está por vaciarse. Los valores que se especificaron se muestran en la Figura 3.28, estos son recomendados por Altera por ser óptimos considerando consumo de recursos y desempeño [19].

- **SDRAM Controller:**

El módulo controlador para SDRAM provee una interfaz que permite conectarse a uno o más chips SDRAM y se ocupa de todos los requisitos de protocolo [19].

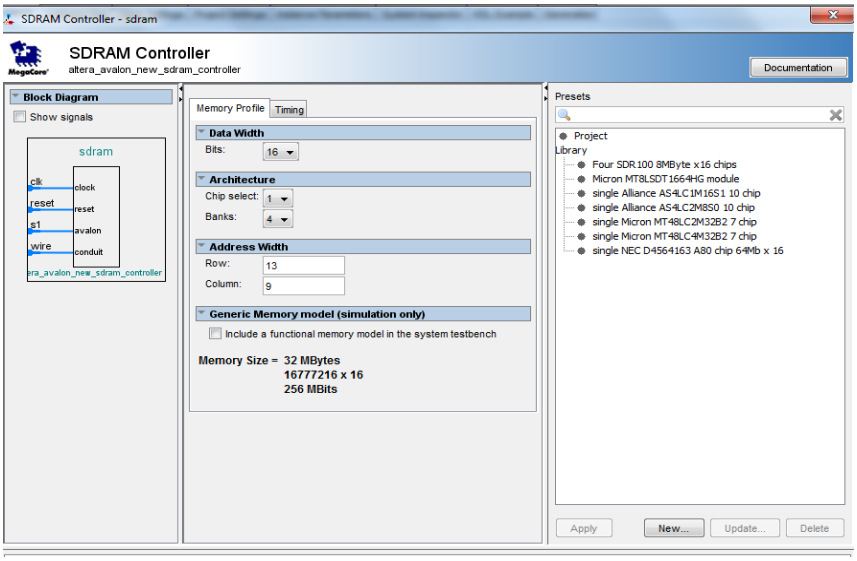


Figura 3.29 Configuración del módulo SDRAM Controller.

En la Figura 3.29 se muestra la ventana de configuración. La configuración que se muestra es para la memoria de 32MB de la tarjeta DE-Nano.

En la ventana se debe especificar el ancho de los datos, en este caso es 16 bits. Si se desea trabajar con varios chips de memoria y se debe cambiar el parámetro Chip select, en este caso el valor es 1. Se tiene que especificar además el número de bancos, el número de columnas y filas de la SDRAM, estos valores son 4,13 y 9 respectivamente. Estos datos definen la capacidad de la tarjeta a usar:

Tamaño de la memoria= $4x2^{13}x2^{9}x16=32MB$

No se realizó modificaciones adicionales a estas, Altera explica en detalle cada parámetro de la ventana [19].

- **Interval Timer:**

Este módulo es un contador configurable, entre susu características está que: puede tener un tamaño de 32 o 64 bits, permite generar interrupciones, puede trabajar como “watchdog timer” y permite opcionalmente generar pulsos cuando el contador llega a cero [19].

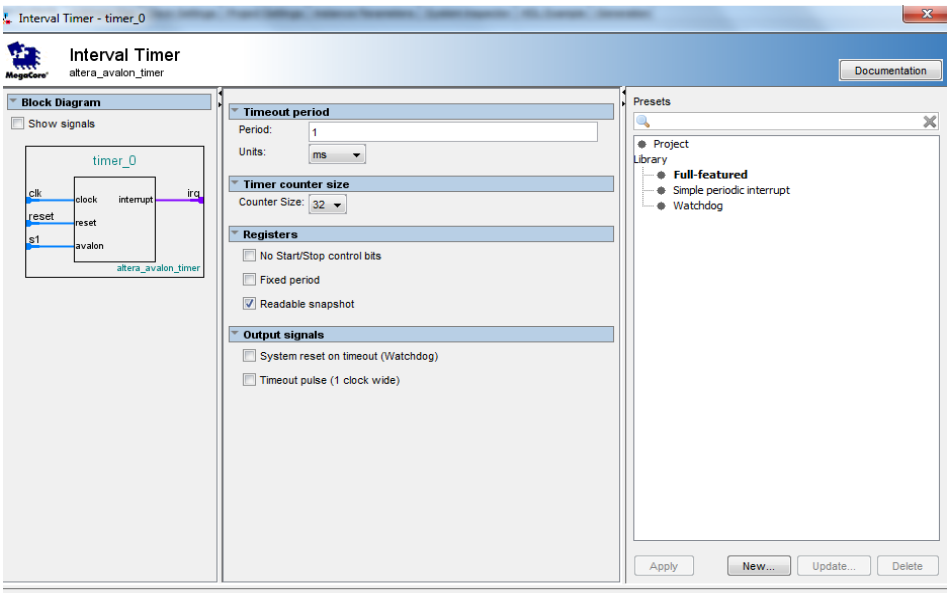


Figura 3.30 Configuración del módulo Interval Timer.

En la ventana de configuración del módulo, Figura 3.30, se fijó el tamaño del contador en 32 bits, se permite cambiar el periodo de

conteo en software en la casilla "*Fixed Period*". Además desmarcando "*No Start/Stop control bits*" se puede detener e iniciar el conteo y al marcar la casilla "*Readable snapshot*" se hace que el procesador pueda leer el valor del contador. En la sección Output Signals se mantienen desmarcadas ambas casillas.

En la computadora se usó dos módulos Interval Timer, llamados Timer1 y Timer0, cada módulo es encargado de generar una interrupción.

- **UART (RS-232 Serial Port):**

Permite la comunicación serial con un dispositivo externo. El módulo implementa el protocolo RS-232 y permite ajustar tasa de bits, paridad, bits parada y número de bits de datos [16].

El módulo UART es parte de la librería University Program de Qsys. Para el sistema se usó un solo bloque UART que permite la conexión con un módulo de comunicación Bluetooth.

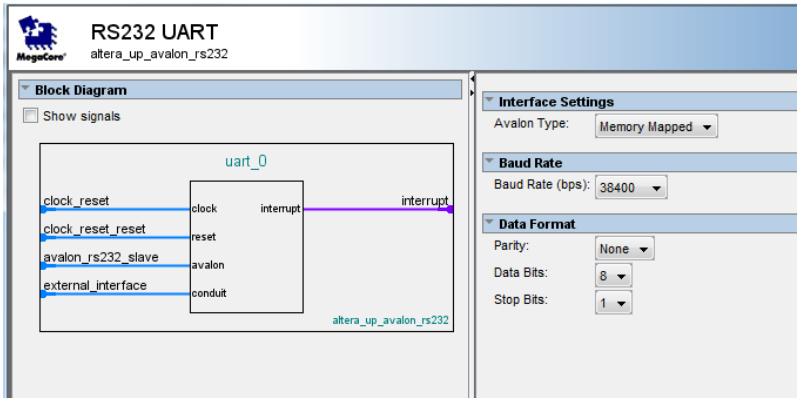


Figura 3.31 Configuración del módulo RS232 UART.

En la Figura 3.31 se observa la ventana de configuración del módulo en Qsys, aquí se especifica una tasa de transmisión de 38400 bps, no se usa bit de paridad, el tamaño del dato de fija en 8 bits y se usa un bit de parada.

- **PIO (Parallel I/O)**:

Permite comunicar a un periférico maestro con puertos de entrada o salida externos a la FPGA, por ejemplo botones o una pantalla led [20].

Los módulos PIO usados son los siguientes:

- PushButtons, esta configuración es hecha por Altera para manejar los botones de sus tarjeta de desarrollo, la DE0-

Nano posee 2 botones, uno es usado como botón de reinicio por lo que el módulo maneja un sólo botón con 1 bit.

- Green_LEDs, configuración de Altera para manejar el banco de LEDs de sus tarjetas de desarrollo, tienen 7 bits porque la DE0-nano posee siete LEDs.
- Dip_Switches, configuración de Altera para leer los interruptores de la tarjeta de desarrollo, en la DE0-nano se maneja cuatro bits, uno para cada interruptor tipo DIP.
- btn_Ex, este módulo es para manejar botones externos a la tarjeta, es de 2 bits. En el robot se agregaron 2 botones.
- sens_IR, con este módulo se lee de manera digital los valores de los sensores infrarrojos, el módulo posee 14 bits, uno para cada sensor.
- dir_PH con este módulo se envía las señales de dirección (IN1, IN2) para el Controlador de Motores de Pololu, posee 4 bits, 2 bits para cada módulo.

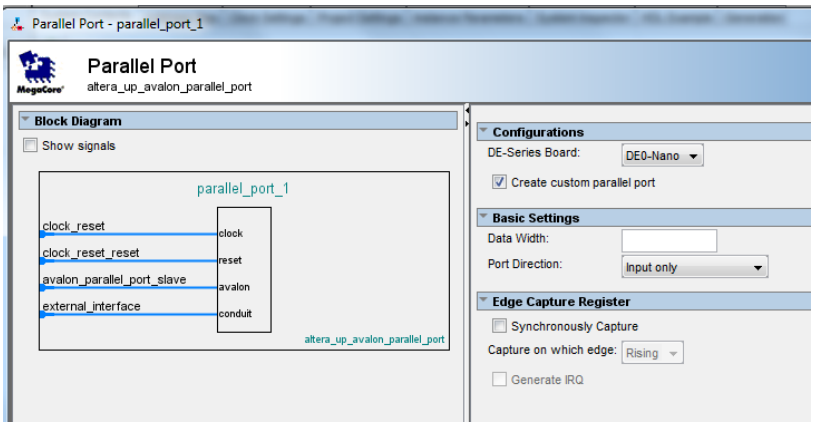


Figura 3.32 Configuración del módulo Parallel Port

En la Figura 3.32 se observa la ventana de configuración, en la primera casilla se escoge DE0-Nano y al desmarcar “Create custom parallel port” se puede escoger configuraciones previamente hecha, si no se crea una propia: se especifica el tamaño en bits del puerto y si el puerto es de entrada, salida o bidireccional y en la pestaña “Edge Capture Register” se activa la detección de flancos de subida, bajada o ambos y se puede activar una solicitud de interrupción en la detección.

El módulo Parallel Port se agrega a Qsys al instalar el software University Program.

- **ADC CONTROLLER**

El módulo ADC Controller es una interfaz para comunicarse con el convertidor analógico digital ADC128S022 de la DE0-nano [21].

Este módulo es parte del software University Program de Altera. El módulo maneja todas las señales de entrada y salida del ADC, provee acceso a los 8 canales de entrada del convertidor y permite seleccionar la frecuencia de trabajo del ADC.

En la ventana de configuración se puede especificar el número de canales a usar. Para este proyecto se necesita únicamente un bit del ADC con el que se mide el voltaje de la batería de alimentación, por lo que se seleccionó el mínimo que es 2 canales y además se escogió la mayor frecuencia que es 3.2 Mhz.

- **Floating point Hardware**

Este componente permite realizar operaciones de punto flotante en hardware. El conjunto básico de instrucciones incluye adición, resta y multiplicación, la división está disponible como una extensión. El compilador reconoce el código que puede tomar ventaja del módulo y se compila para hacer uso de este hardware

En la ventana de configuración se puede agregar o no la división, en este proyecto sí se usa esta opción.

- **Performance Counter**

Con este módulo se puede medir con precisión el tiempo de ejecución de varias secciones de código, sólo hay que añadir una instrucción al principio y al final de cada sección que se desea medir.

En la ventana de configuración del módulo se especifica el número de secciones simultáneas que se puede medir, las cuales son máximo siete, en este proyecto se usó tres.

- **I2C**

El módulo I2C implementa el protocolo de comunicación I2C. Permite programar por software la frecuencia de entrada del reloj, iniciar, detener y repetir la generación y el reconocimiento de los bits de confirmación y de inicio, y soporta direccionamiento de siete o diez bits.

En la ventana de configuración del módulo, no existe ninguna configuración adicional que se tenga que hacer.

Este módulo no es provisto por Altera, es un módulo de libre uso compartido en una comunidad de “cores” libres llamada Opencores [22].

- **Módulo PWM y modo de espera:**

Este módulo tiene dos funciones, la primera es generar una señal modulada por ancho de pulso para controlar el voltaje promedio de entrada de los motores y la segunda es la de manejar la señal de entrada Standby del controlador de Pololu para poner el integrado en modo de espera.

El módulo permite por software iniciar la señal PWM a cualquier frecuencia menor a la del sistema, detener la generación de la señal y cambiar el ciclo de trabajo de la misma.

En la ventana de configuración se puede únicamente especificar el valor de inicio de dos registros, estos se inicializan con 0, ya que son también modificables por software.

El módulo que únicamente genera señales PWM es provisto por Altera, pero este fue modificado para que además maneje la señal de entrada STDBY del controlador de Pololu, esto se detalla más adelante.

El sistema usa dos módulos de este tipo, uno para cada controlador Pololu.

- **Encoder**

Este módulo está encargado de detectar y procesar las señales recibidas desde el codificador rotatorio del motor. Lleva el conteo

con signo del número de pasos que rota el eje del motor, el signo indica el sentido de giro, de esta manera se puede saber la posición angular en la que se encuentra el eje.

El módulo permite además encerar el conteo cuando el periférico maestro lo requiera. Este módulo fue escrito en VHDL junto con la interfaz para ser usado como módulo esclavo.

En la ventana de configuración no se necesita hacer ninguna configuración previa para usarlo.

La computadora usa dos módulos de este tipo, uno para cada codificador rotatorio.

COMPONENTES HDL

Una de las principales ventajas trabajar con un procesador sobre FPGA es que se puede añadir cualquier periférico deseado al sistema. Altera tiene disponible varios módulos listos para ser usados. Pero además se pude crear un módulo propio en un lenguaje de descripción de hardware y agregarlo al sistema. Para esto lo primero es describir el componente en un lenguaje HDL, lo más recomendable es que antes de agregar el componente a NOS II, este sea probado con una simulación y de ser posible sobre la FPGA, para que si existen problemas se pueda localizar el fallo.

Luego de tener el diseño del componente en HDL, se debe añadir una interfaz para comunicarse con el procesador, esta interfaz consiste en un archivo con las señales de entrada y salida. Estas tienen su nombre y función específica las cuales están documentados por Altera [14].

Una vez que se tiene todos los archivos descritos en HDL, se puede crear el componente en Qsys. Los pasos detallados que se debe seguir para la creación de un módulo en Qsys están también documentados por altera [23].

Módulo Controlador de Codificadores Rotatorios

Existen varias formas de llevar el conteo de un codificador rotatorio de dos canales usando un procesador, pero por lo general un procesador ejecuta una sola tarea a la vez, por lo que si se necesita que el procesador realice algún trabajo adicional a llevar el conteo del codificador se debe recurrir al uso de interrupciones.

Esto es posible realizarlo en software pero se agregaría un trabajo adicional al procesador quitando tiempo para la ejecución de otras tareas y se añade una complejidad innecesaria al programa, ya que se crear un componente de hardware que se dedique a llevar el conteo del codificador rotatorio.

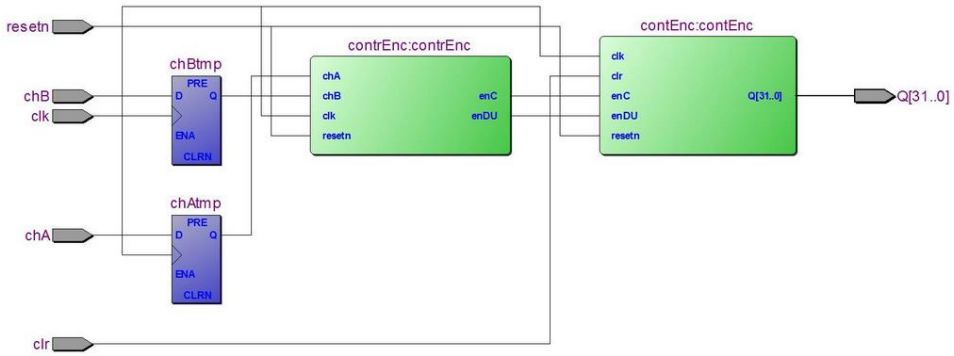


Figura 3.33 Diagrama de Bloques del módulo Encoder.

En la Figura 3.33 se presenta la partición funcional del módulo encargado del conteo. El módulo tiene 5 señales de entrada cada una de 1 bit y una señal de salida de 32 bits:

Resetn: Señal de reinicio asincrónica.

Clk: Señal de reloj.

chA: Señal de entrada corresponde a uno de los canales del codificador rotatorio.

chB: Señal de entrada corresponde a uno de los canales del codificador rotatorio.

clr: Señal de entrada, sirve para borrar el registro que lleva el conteo del codificador rotatorio.

Q: Señal de salida de 32 bits, esta señal indica el número de pasos que ha avanzado el codificador rotatorio, un bit es el bit de signo, pues el conteo puede ser negativo, dependiendo del sentido de giro.

El módulo posee cuatro bloques principales, de estos los llamados ChAtmp y ChBtmp corresponden a dos “flip flops” tipo D, estos se encargan de sincronizar las señales de los canales de los codificadores rotatorios, sin estos bloques el módulo no funciona correctamente.

De los otros dos módulos restantes, ContEnc corresponde a un contador, este aumenta o disminuye según el sentido de giro. El contador maneja las siguientes señales:

Resetn: Señal de reinicio asincrónica.

Clk: Señal de reloj.

enDU: Señal de entrada de 1 bit, si está en 1 el contador disminuye, si es 0 aumenta su valor.

enC: Señal de entrada de 1 bit, sincrónica. Cuando está en 1 el contador aumenta o disminuye según el valor de enDU, si está en cero mantiene el valor anterior.

clr: Señal de entrada de 1 bit, ya fue descrita anteriormente.

Q: Señal de salida de 32 bits, ya fue descrita anteriormente.

El bloque ContrEnc es el controlador del módulo. Todas las señales que maneja son de un bit, las entradas son: clk, resetn, chA y chB. Las salidas son: enC y enUD, todas estas señales ya fueron descritas en los bloques anteriores.

La Anexo A se muestra en detalle el funcionamiento del controlador, en resumen lo que hace el controlador es analizar en conjunto la señales de los canales A y B si los valores que toman avanzan de izquierda a derecha en la secuencia 01-11-01-00 el contador aumenta uno cada vez que se avanza y si la secuencia avanza de derecha a izquierda el contador disminuye, en cualquier otro caso el contador se mantiene en su valor.

El funcionamiento del módulo fue probado en una simulación esto se muestra en el Anexo B. Se observa como el contador avanza hasta llegar a 11, luego se intercambia los valores de las señales de los canales A y B y este empieza a disminuir hasta llegar hasta cero.

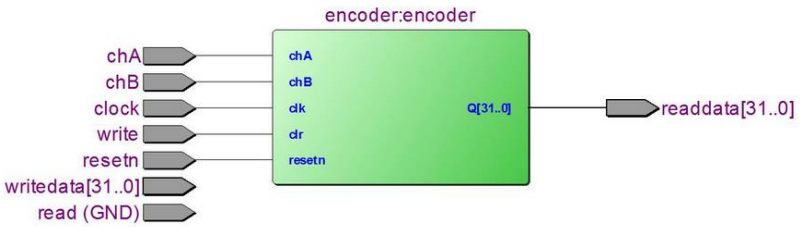


Figura 3.34 Bloque e Interfaz del módulo Encoder controller.

El módulo está agrupado en un solo componente en el cual se han añadido las señales externas, que vienen del procesador o son externas también al sistema, esto se observa en la Figura 3.34

Las señales de entrada son:

clock: señal de 1 bit, es la señal de reloj de todo el sistema.

resetn: señal de 1 bit, es la señal de reinicio de todo el sistema.

chA: señal de 1 bit, señal externa que viene de un canal del codificador rotatorio.

chB: señal de 1 bit, señal externa que viene de un canal del codificador rotatorio.

write: señal de 1 bit, esta señal viene del procesador, cuando el procesador escribe un valor en el periférico esta señal se hace uno durante un ciclo de reloj, en otro caso es cero.

Writedata: señal de 32 bit, esta señal viene del procesador es el valor que escribe el procesador en el periférico, esta señal no es usada.

Read: señal de 1 bit, esta señal viene del procesador, cuando el procesador lee un valor del periférico esta señal se hace uno durante un ciclo de reloj, en otro caso es cero, esta señal no es usada.

La única señal de salida es:

Readdata: señal de 32 bit, esta señal es el valor que escribe el periférico al procesador, se envía cuando el procesador lo solicita, esta es la señal en la que se envía el valor del conteo.

Módulo generador de la señal PWM y Controlador del Modo de Espera.

Es común que se use módulos en hardware para generar señales PWM, la mayoría de microcontroladores vienen ya con sus propios módulos de hardware dedicados a esto.

Altera proporciona los archivos de hardware y software para añadir el módulo generador de señales PWM al proyecto en Qsys, estos

son provistos en un ejemplo de creación de componentes para SOPC Builder[3] [24].

Por otro lado, en la sección 3.1.2 se trató sobre los controladores electrónicos para los motores, se comentó que la hoja de datos del fabricante recomienda, para evitar que penetre corriente desde los motores al integrado, pasar por el modo Standby durante un tiempo.

Realizar esto en software no sería posible, pues el controlador de los motores pasa al modo Short Brake cuando la señal PWM es 0V, pero esta señal es generada en hardware, el procesador no tiene información sobre los valores que toma esta señal.

Entonces lo que se hizo fue modificar el módulo PWM de Altera para que genere una señal adicional encargada de llevar el controlador de los motores al modo Standby.

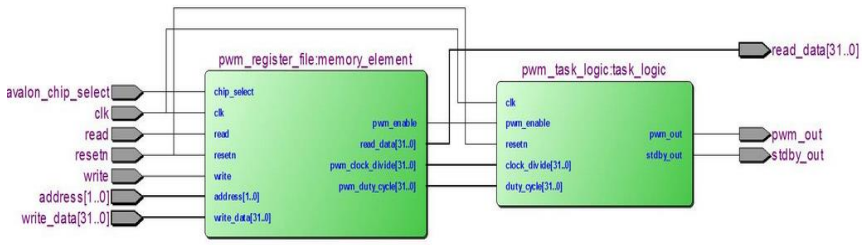


Figura 3.35 Diagrama de bloques del módulo PWM.

[3] Software predecesor de Qsys

El componente generador de PWM de Altera está descrito en hardware usando el lenguaje de descripción de hardware Verilog, son tres los archivos que describen el componente.

En la Figura 3.35 se muestra el diagrama de bloques del módulo PWM ya con la señal standby_out agregada.

El bloque Task_logic que corresponde al archivo Pwm_ task_logic es el que está encargado propiamente de generar la señal PWM, es este el bloque que fue modificado. El bloque Memory_element correspondiente al archivo Pwm_register_file contiene la lógica para leer y escribir los registros del módulo PWM. El archivo Pwm_avalon_interface corresponde al módulo agrupado con la interfaz de comunicación.

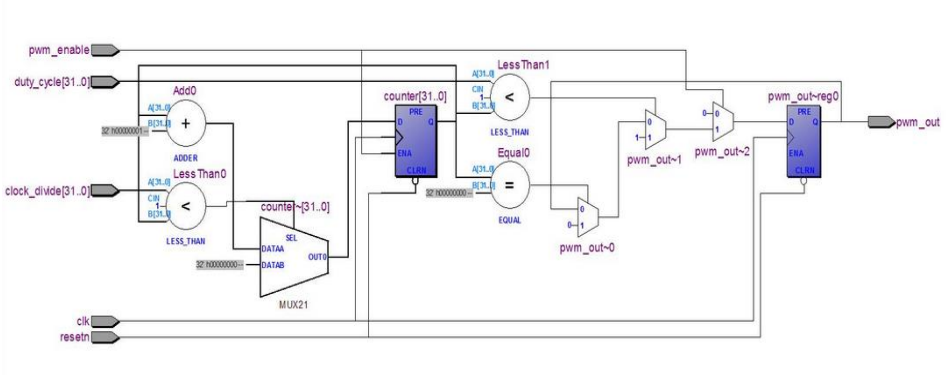


Figura 3.36 Partición Funcional Bloque Task_logic original.

La idea básica del código que proporciona Altera es, primero llevar un conteo de los ciclos de reloj que transcurren. La salida Pwm_out inicialmente toma el valor cero, luego cuando el contador de ciclos de reloj supera el valor de la señal Duty_cycle, cambia el estado a un 1 lógico, finalmente cuando el contador llega al valor Clock_divide, todo el proceso comienza de nuevo y empieza un nuevo período de Pwm_out.

En el Anexo C se muestra el bloque Task_logic luego de modificarlo para controlar la señal de modo de espera. El valor de Standby_out es cero al inicio de un período de la señal Pwm_out, luego cuando el contador de ciclos de reloj llega a un valor “t1”, Pwm_out se hace alto hasta que el contador esté a “t2” ciclos de un cambio de nivel de la señal Pwm_out, en ese momento vuelve a ser cero. En cualquier otro caso la señal Standby_out es un 1 lógico.

En el Anexo B se muestra la simulación del módulo PWM con la modificación hecha, para la simulación se usó un valor “t1” de 1 y un valor de “t2” igual a 3, se puede observar, poco después de los 2.4 us de simulación, que cuando la señal Pwm_out pasa de alto a bajo la señal Standby_out está en cero durante un ciclo de reloj, luego toma el valor de 1 hasta que está a 3 ciclos de reloj de que

la señal pwm_out sea 1, ahí vuelve a hacerse cero hasta un nuevo cambio de nivel de la señal pwm_out.

En la implementación se usa un valor t1 igual a 6 ciclos de reloj y un valor t2 igual a 24 ciclos de reloj, esto es porque se trabaja con un reloj de 100Mhz lo que implica que t1 es 60 ns y t2 es 240 ns estos valores cumplen los requerimientos que pide el fabricante, como se indicó en la sección 3.1.2.

3.2.2 Controlador de velocidad

Para definir el movimiento del robot se necesita especificar su velocidad angular y la velocidad lineal del centro de masa.

De la Ecuación 3. **4** y la Ecuación 3. **5** tiene que:

$$\omega = \frac{v_r - v_l}{l}$$

$$v_x = \sqrt{(\frac{v_r + v_l}{2})^2 + (\frac{v_r - v_l}{l} x)^2}$$

Entonces, basta con controlar la velocidad lineal en el punto de contacto de cada rueda, v_r y v_l , para controlar la velocidad angular ω y lineal v_x del robot.

Por esta razón, lo que primero se hizo fue diseñar un controlador de velocidad para cada motor. Estos controladores son a su vez

manejados por un controlador externo que modifica su referencia según la velocidad angular y lineal que el robot deba tener.

Para identificar la función de transferencia del motor se almacenó los datos en la memoria SDRAM de la DE0-Nano, luego se transmitieron inalámbricamente con el módulo Bluetooth para ser analizados con MATLAB.

La función de transferencia a identificar tiene como señal de entrada un valor de voltaje y la salida es la velocidad angular del eje, para realizar la identificación se utilizó una señal de voltaje de prueba de tipo tren de pulsos con un ciclo de trabajo de 50% y toma los valores 6V y -6V. Es importante que el valor del período de la señal de prueba garantice que el sistema alcance su punto de estabilización antes de que el siguiente pulso se aplique [25]. Esto se muestra en la Figura 3.37.La señal y1 representa la velocidad angular medida por los codificadores, u1 es la señal de entrada de voltaje, el intervalo de color rojo es el seleccionado para la identificación. Hay que notar que la señal u1 no esta medida en voltios, sino que los valores que toma corresponden al ciclo de trabajo de la señal PWM, 5000 es el máximo valor del ciclo, que corresponde a una señal de 6V.

Para la identificación del sistema se usó la herramienta Ident de MATLAB. En general, Ident necesita que se especifique las

señales de entrada y salida, el intervalo de muestreo, seleccionar el rango en el cual se quiere hacer la identificación, y especificar el orden del sistema. En este caso el sistema a identificar es de segundo orden, el modelo matemático preciso del sistema se muestra en la sección 3.1.1.

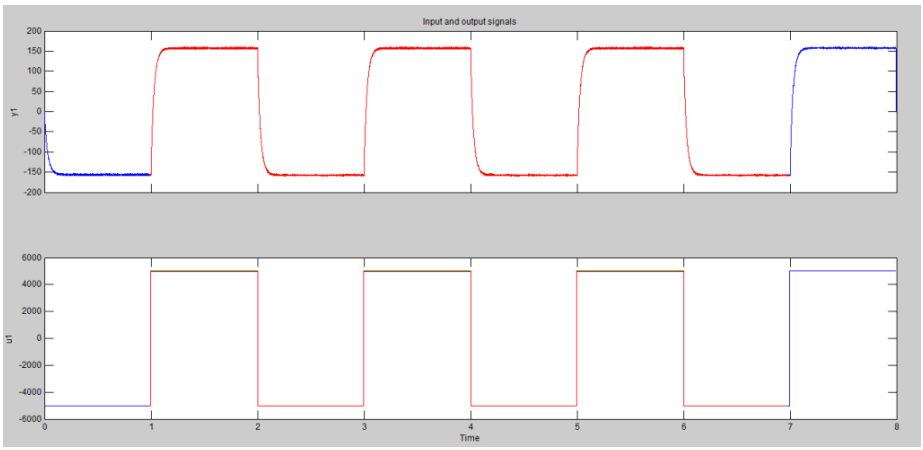


Figura 3.37 Velocidad angular y voltaje medidos para la identificación de la función de transferencia del motor.

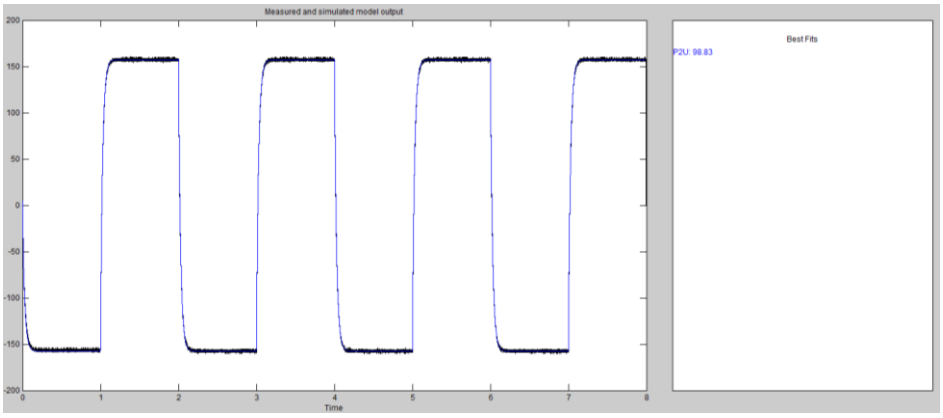


Figura 3.38 Respuesta medida y simulada para la función de transferencia del motor.

En la Figura 3.38 se muestra con azul la señal graficada a partir del modelo identificado y en negro la gráfica real. A la derecha de la gráfica se muestra el porcentaje en el que los datos medidos encajan en el modelo propuesto, un mayor número indica un mejor modelo, a esto se le llama FIT. En este modelo se obtuvo un FIT de 98.83%.

La función obtenida es de la siguiente forma:

$$MD(s) = \frac{K_p}{1 + 2\zeta\tau s + (\zeta s)^2}$$

Donde:

$K_p = 0.031492$
$\zeta = 0.002174$
$\tau = 6.2611$

El objetivo de identificar la función de transferencia es diseñar un controlador que luego será implementado en C en la FPGA, para esto se necesita trabajar en tiempo discreto, en este caso no es útil describir el sistema ni en tiempo continuo ni usando la transformada de Laplace.

Para transformar el sistema a tiempo discreto se utiliza la función c2d() de Matlab la cual devuelve un sistema discreto a partir de la función de transferencia en tiempo continuo y del tiempo de muestreo, el tiempo usado fue de 0.5 ms, sobre la elección de este tiempo se explica en la sección 3.2.5.

Un sistema discreto es representado en espacio de estados de la siguiente forma:

Ecuación 3. 19

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$

x_{k+1} es el vector de estados.
y_k es el vector de salida.
u_k es el vector de entradas .
A es la matriz de estados.
B es la matriz de entrada.
C es la matriz de salida.
D es la matriz de transmisión directa.

Para encontrar las matrices A,B,C,D se utiliza la función ss() de MATLAB, el parámetro de entrada es la función de transferencia que se obtiene como salida de la función c2d(). Las matrices de estado del sistema son:

$$A = \begin{pmatrix} -0.005963 & -35.197 \\ 0.00016635 & 0.9522 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.00016635 \\ 2.259e-007 \end{pmatrix}$$

$$C = (0 \quad 6663.4)$$

$$D = 0$$

Una vez representado el sistema en tiempo discreto se puede diseñar el controlador, no existe ningún requerimiento específico para este diseño, de forma general lo que se busca es que el

sistema sea rápido, que no tenga demasiado sobre impulso, que sea lo más simple posible para facilitar la implementación, y que el controlador funcione considerando que el voltaje de entrada del motor está limitado a 6V.

Para el diseño del controlador se usó la herramienta Sisotool de MATLAB, esta herramienta permite diseñar diferentes modelos y analizar su desempeño observando tiempo de establecimiento, tiempo de subida, respuesta pico, valor de estado final. Sisotool permite ajustar de forma manual o automática los parámetros del controlador, como ubicación de ceros, polos y ganancias.

El ajuste automático de parámetros se hace basándose en requerimientos de entrada: límites superior e inferior, porcentaje y tiempo de establecimiento, tiempo y porcentaje de subida, porcentaje de “overshoot” y “undershoot” , estos últimos indican cuánto se sobrepasa de la unidad y cuánto abajo de cero llega la respuesta antes de estabilizarse, respectivamente.

Se analizó controladores proporcionales, de primer orden y de segundo orden, el mejor desempeño se obtuvo con los de segundo y primer orden, pero se escogió el de primer orden por el requerimiento de ser simple.

Específicamente el controlador seleccionado está formado por un polo y un cero, a continuación se presenta el diseño de este controlador.

La arquitectura del sistema de control seleccionada es la que viene por defecto en Sisotool:

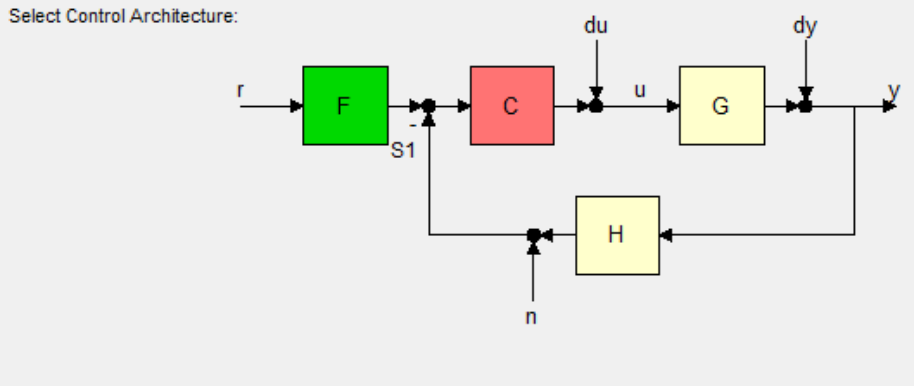


Figura 3.39 Arquitectura de Control del sistema de realimentación para el control de velocidad.

El bloque C representa el controlador que va a ser diseñado, G es la función de transferencia del sistema a controlar, en este caso sería la del motor, las funciones de transferencia F y H se fijan a un valor constante igual a la unidad.

Para la señal u de la Figura 3.39 se necesita un límite, no puede sobrepasar los 6V, ni estar por debajo de -6V, para evitar daños en los motores. Estos requerimientos se agregan en Sisotool, en la

Figura 3.40 se observa que se limita a valores entre[4] 5000 y -5000, estos números corresponden a valores entre 6V y -6V.

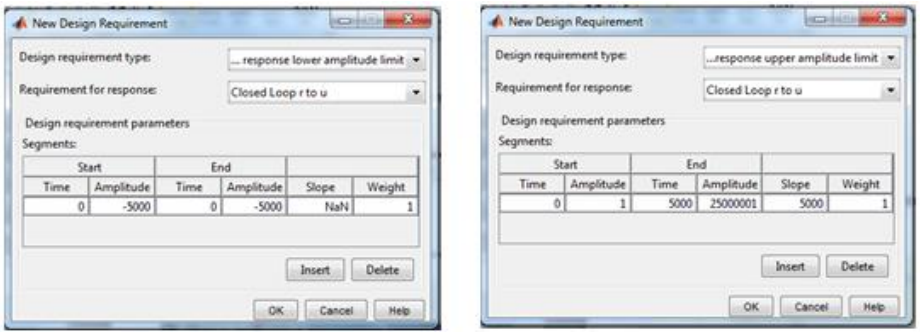


Figura 3.40 Requerimientos de diseño para la salida del controlador de los motores.

Para la señal Y de la Figura 3.39, no hay algún requerimiento específico. Lo que se hizo fue empezar por requerimientos sencillos de alcanzar y luego gradualmente se fue aumentando la complejidad de tal manera que cada vez el desempeño mejore, la mejor respuesta fue obtenida con los requerimientos que se muestran en la Figura 3.41.

[4]Estos números indican el valor en ciclos de reloj del tiempo en alto de la señal PWM generada por la FPGA, 5000 es el valor máximo.

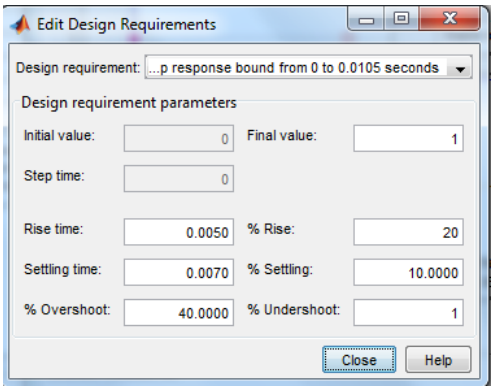


Figura 3.41 Requerimientos de diseño para la salida del sistema realimentado de los motores.

El controlador obtenido para ser implementado en C es:

$$C_M = \frac{2217 - 1167z^{-1}}{1 - 0.2545z^{-1}}$$

Por defecto los controladores se representan en Matlab usando la transformada z, pero para la implementación en C se necesita expresar los controladores en ecuaciones en diferencias, si U_M es la salida del controlador que corresponde al voltaje de entrada del motor y Err_M la señal de error de entrada, se tiene que:

$$\frac{U_M}{Err_M} = \frac{2217 - 1167z^{-1}}{1 - 0.2545z^{-1}}$$

$$U_M(1 - 0.2545z^{-1}) = Err_M\,(2217 - 1167z^{-1})$$

Pasando al dominio del tiempo:

Ecuación 3. 20

$$u_M(k) = 0.2545u_M(k-1) + 2217err_M(k) - 1167z^{-1}err_M(k-1)$$

La Ecuación 3. **20** pueden ser implementadas en C y en general en cualquier lenguaje de programación, estas ecuaciones en diferencias junto con el modelo en tiempo discreto del motor se implementan en un script de MATLAB para probar el controlador.

El script simula el sistema de realimentación a partir de su modelo en tiempo discreto y permite observar la respuesta al escalón añadiendo el hecho de que el voltaje de entrada del motor está limitado[5]. En la simulación se aplica una entrada escalón de amplitud 20, esto se muestra en la Figura 3.42.

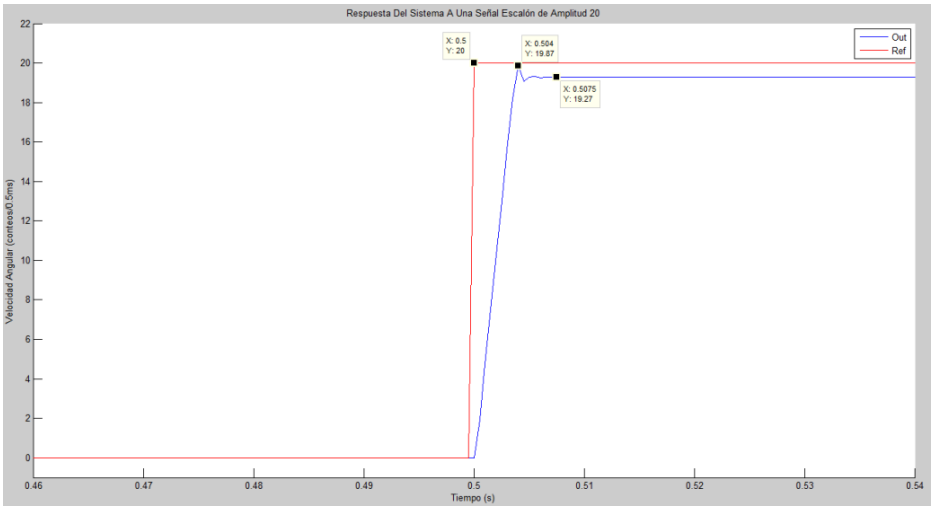


Figura 3.42 Simulación del sistema de realimentación del motor.

[5] En Sisotool al diseñar el controlador se especificó esta limitación, pero sólo corresponde a una entrada de amplitud 1.

Se observa que el tiempo de estabilización es 7.5 ms, el error estado estacionario es diferente de cero[6] (0.73) y el valor pico de la respuesta es 19.87, además el controlador funciona bien con el límite de entrada de voltaje, en la Figura 3.43 se muestra la salida del controlador.

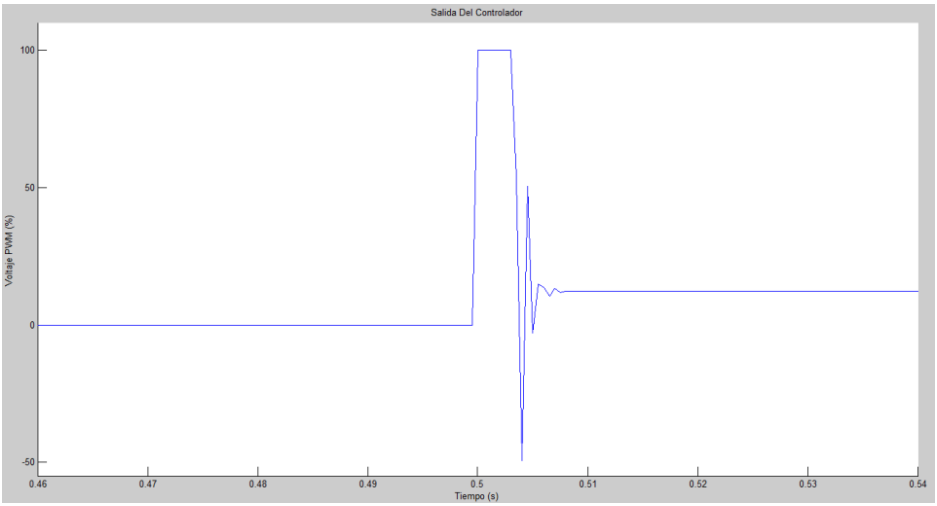


Figura 3.43 Salida del Controlador del motor.

3.2.3 Algoritmo de Filtrado

En la sección 2.5 se describió los detalles teóricos sobre el algoritmo que es utilizado en este proyecto, el Filtro de Kalman.

Las señales que pasan por el proceso de filtrado son las señales que provienen de los codificadores rotatorios, la señal que viene de los sensores infrarrojos y la del giroscopio.

[6] En la implementación se aumenta el valor de referencia para alcanzar el valor deseado, por ejemplo con una referencia de 1.73 se tendrá una salida de 1.

Lo que se hace es filtrar independientemente la señal de cada codificador rotatorio, la señal de los sensores infrarrojos y la del giroscopio se las pasa juntas por otro filtro que además fusiona ambos datos, esto se detalla más adelante. Primero se explica acerca del filtrado de las señales provenientes de los codificadores rotatorios, estas señales tienen la información de la velocidad de giro del eje del motor.

Codificadores rotatorios

Para la implementación del algoritmo de filtrado se necesita un modelo matemático expresado en matrices de estado. El modelo que se necesita debe tener como entrada la señal de voltaje que va a los motores y como salida la velocidad angular del eje del motor. Este modelo es el mismo que se usó para diseñar el controlador del motor, en la Ecuación 3. **19** se describe este modelo en tiempo discreto.

Para la implementación del filtro se necesita hacer una modificación, para esto se trabaja sobre el modelo en tiempo continuo y luego se pasa a tiempo discreto.

El modelo en tiempo continuo se lo puede escribir como:

$$\dot{x_1} = ax_1 + bx_2 + cu$$
$$\dot{x_2} = x_1$$
$$y = x_2$$

$$a = -5760$$
$$b = -2.116e5$$
$$c = 6663$$

x_2 es la velocidad angular del eje del motor
x_1 representa la aceleración angular.
y es la salida del sistema.

Lo que se hace es agregar al modelo el efecto de un torque externo, con el fin de mejorar la predicción del filtro. Este torque podría ser ocasionado por diversos factores por ejemplo la fricción o una rueda que no es perfectamente redonda causará un torque que se opone al giro.

Aplicando la segunda ley de newton para la dinámica rotacional al sistema que incluye el torque externo:

$$\sum \tau = I\alpha$$

Esta sumatoria de torques se puede separar en la sumatoria de torques del modelo que ya se tiene más el torque externo que se añade:

$$\sum \tau_{m1} + \tau_{ex} = I\alpha$$

Además:

$$\sum \tau_{m1} = I\alpha_{m1}$$

Donde α_{m1} es igual a x_1, la aceleración angular que es descrita en el modelo anterior.

$$I\alpha_{m1} + \tau_{ex} = I\alpha$$

$$\alpha = \alpha_{m1} + \tau_{ex}/I$$

Por lo tanto se tiene que la aceleración angular del nuevo modelo es simplemente la aceleración angular del modelo anterior más la cantidad τ_{ex}/I.

La nueva aceleración angular se la llama $\dot{x_3}$ y x_4 a la cantidad τ_{ex}/I, entonces el sistema de ecuaciones agregando la nueva ecuación obtenida es:

$$\dot{x_1} = ax_1 + bx_2 + cu$$

$$\dot{x_2} = x_1$$

$$\dot{x_3} = x_1 + x_4$$

Para que las variables de estado del sistema sean x_1, x_2, x_3, x_4 se necesita una ecuación adicional que describa la dinámica de x_4. Pero esta variable está relacionada con el torque externo introducido, del cual no se tiene información. Por lo que se utiliza un modelo de torque constante:

$$\dot{x_1} = ax_1 + bx_2 + cu$$

$$\dot{x_2} = x_1$$

$$\dot{x_3} = x_1 + x_4$$

$$\dot{x_4} = 0$$

A pesar de que el modelo utiliza un torque constante para hacer la predicción, esto no implica que esta variable vaya a ser constante en la implementación, esto es porque los valores que tome también dependen de lo que se lea desde los codificadores.

Entonces el nuevo modelo en tiempo continuo es:

$$\dot{x_1} = ax_1 + bx_2 + cu$$
$$\dot{x_2} = x_1$$
$$\dot{x_3} = x_1 + x_4$$
$$\dot{x_4} = 0$$
$$y = x_3$$

En matrices se tiene:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{pmatrix} = \begin{pmatrix} a & b & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} c \\ 0 \\ 0 \\ 0 \end{pmatrix} u$$

$$y = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Este es el modelo en tiempo continuo que se usa para la predicción, pero para poder implementarlo se debe pasar el modelo a tiempo discreto, esto se hace con la función c2d() de MATLAB, el tiempo de muestreo es 0.5ms.

La forma del modelo en tiempo discreto es:

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$

Las matrices obtenidas son:

$$\mathrm{A} = \begin{pmatrix} 0.051 & -34.4020 & 0 & 0 \\ 0.0002 & 0.9877 & 0 & 0 \\ 0.0002 & -0.0123 & 1 & -0.0005 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathrm{B} = \begin{pmatrix} 1.08 \\ 3.87\mathrm{e}-4 \\ 3.87e-4 \\ 0 \end{pmatrix}$$

$$\mathrm{C} = (0 \quad 0 \quad 1 \quad 0)$$

$$\mathrm{D} = 0$$

Una vez que se tiene el modelo del sistema se puede usar las ecuaciones matriciales del filtro de Kalman:

Predicción:

$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k$

$P_{k|k-1} = F_k P_{k-1|k-1} {F_k}^T + Q_k$

Actualización:

$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1}$

$S_k = H_k P_{k|k-1} {H_k}^T + R_k$

$K_k = P_{k|k-1} {H_k}^T {S_k}^{-1}$

$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$

$P_{k|k} = (I - K_k H_k) P_{k|k-1}$

Para disminuir el uso de recursos computacionales el algoritmo que se implementa en la FPGA es el filtro de Kalman de estado estable, en este algoritmo la matriz de ganancia K_k no es adaptativa, esta es reemplazada por la matriz de ganancias en estado estable, que es una matriz constante.

Lo que se hizo fue escribir un script en MATLAB donde sí se implementa el algoritmo completo con todas las ecuaciones matriciales. Para usar el script se necesita tomar datos reales de la velocidad angular medida y el voltaje de entrada del motor. Para esto se aplicó en la entrada del motor un tren de pulsos de voltaje y mientras el eje giraba se trató de detenerlo manualmente para ocasionar un torque que se oponga al movimiento, en la Figura 3.44 se muestran los datos de voltaje y velocidad angular respectivamente.

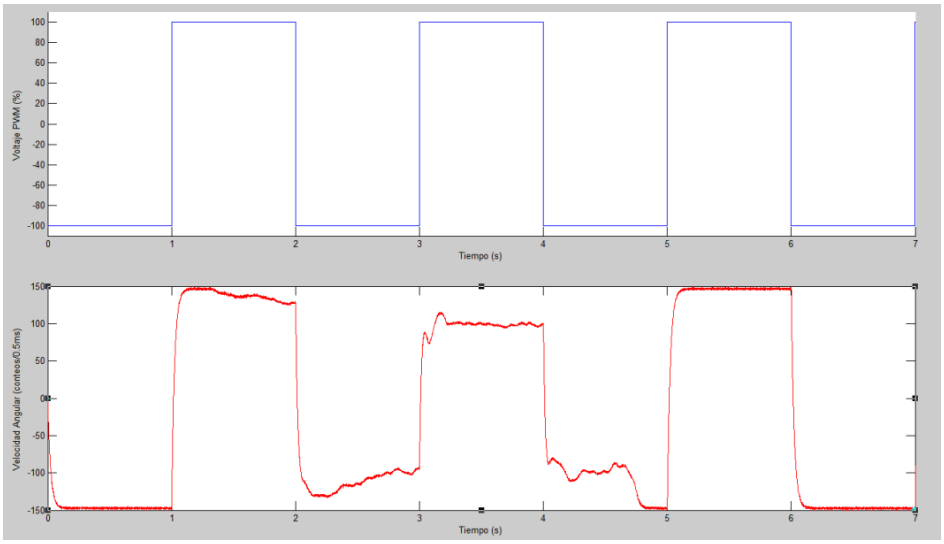


Figura 3.44 Voltaje y Velocidad Angular de entrada para la simulación del sistema de realimentación del motor.

Además de estos datos se necesita definir los valores de las matrices de covarianza Q y R, no existe un método estándar para encontrar el valor de estas matrices, estos valores fueron

encontrados manualmente hasta obtener el resultado deseado del filtrado.

En la Figura 3.45 muestra la entrada y la salida del filtro con matrices Q y R que no han sido ajustadas correctamente.

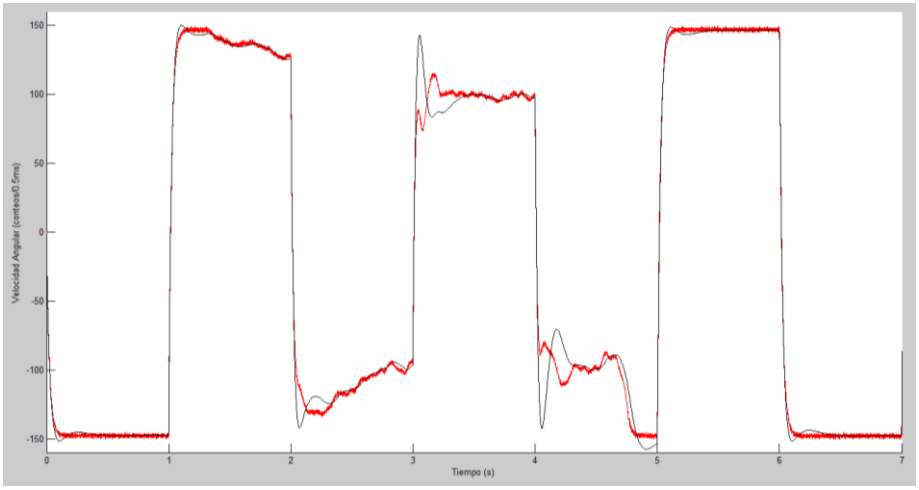


Figura 3.45 Filtro para los Encoders con Matrices Q y R no ajustadas correctamente.

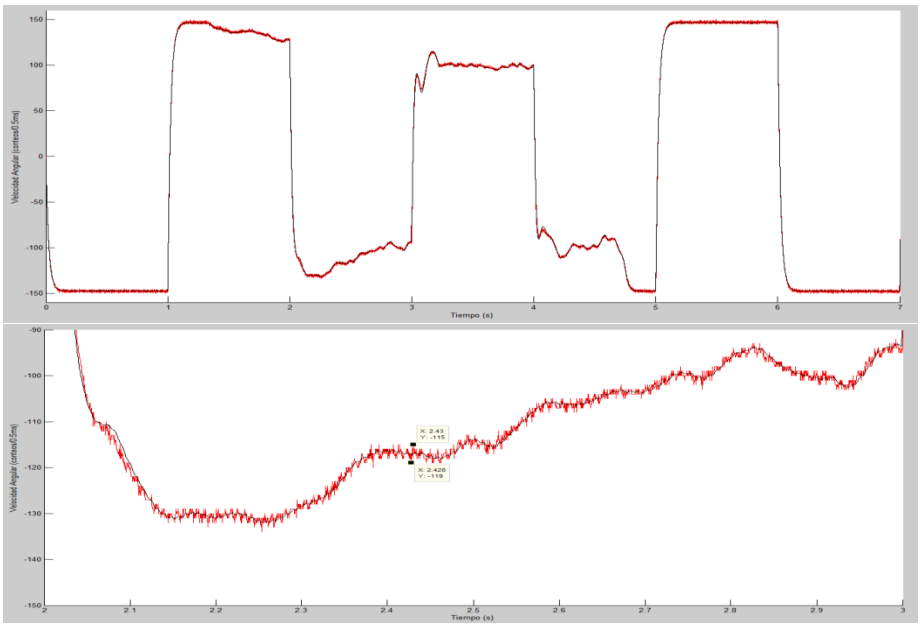


Figura 3.46 Filtro para los Encoders con Matrices Q y R correctamente ajustadas.

En la Figura 3.46 se muestra la entrada y la salida del filtro con las matrices Q y R ajustadas correctamente, estas son:

$R = 25$

$$\mathrm{Q} = \begin{pmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Además una vez que se encuentra los valores para las matrices Q y R, se puede analizar el comportamiento de los elementos de la matriz de ganancia, en la Figura 3.47 se muestran los valores que toman estos elementos en el tiempo.

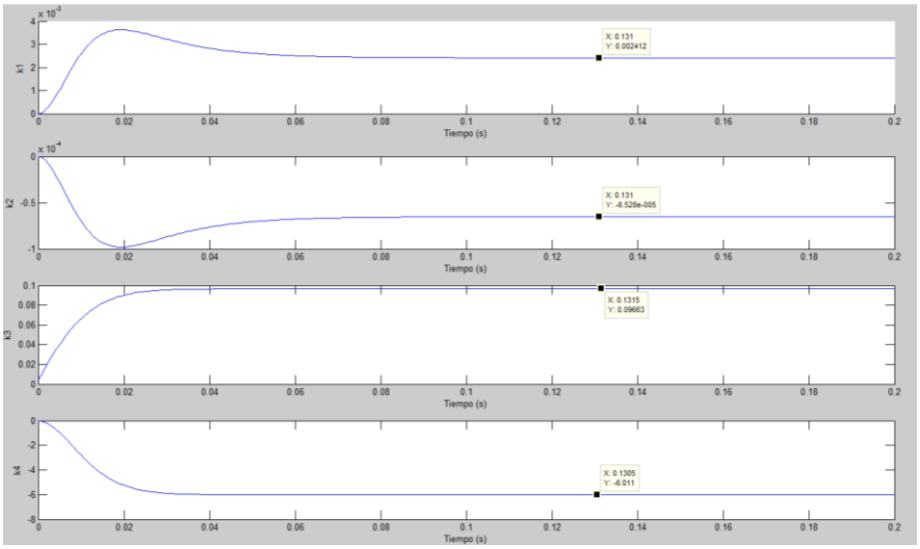


Figura 3.47 Convergencia de los elemento de la matriz de ganancia del Filtro de Kalman para el Encoder del Motor.

Se observa que los elementos de la matriz de ganancia convergen a un valor específico, este es el valor en estado estable. En la

implementación del filtro en la FPGA se usa la matriz de ganancia en estado estable, que es la matriz formada por cada uno de estos valores.

Finalmente el sistema a implementar en la FPGA es de la forma:

$$x_{k|k-1} = Ax_{k-1|k-1} + Bu_k$$
$$y_k = z_k - Cx_{k|k-1}$$
$$x_{k|k} = x_{k|k-1} + Ky_k$$

Dónde:

$$A = \begin{pmatrix} 0.051 & -34.4020 & 0 & 0 \\ 0.0002 & 0.9877 & 0 & 0 \\ 0.0002 & -0.0123 & 1 & -0.0005 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 1.08 \\ 3.87e-4 \\ 3.87e-4 \\ 0 \end{pmatrix}$$

$$C = (0 \quad 0 \quad 1 \quad 0)$$

$$K = \begin{pmatrix} 0.0024118 \\ -6.5265e-05 \\ 0.096626 \\ 6.0112 \end{pmatrix}$$

Giroscopio y Sensores Infrarrojos.

Para filtrar las señales provenientes de estos sensores se usa también un filtro de Kalman, pero es usado de forma diferente a como se implementó con los codificadores rotatorios. Esto se explica a continuación.

Los sensores infrarrojos se usan para medir la separación angular entre la línea de la pista y la parte central delantera del robot, esto se observa en la Figura 3.48 el ángulo en mención es representado con θ.

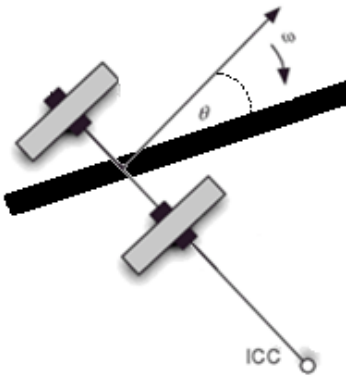


Figura 3.48 Ángulo de separación entre robot y la pista.

Lo que se hace en software es leer el valor de cada uno de los sensores infrarrojos y en base a estas lecturas una función devuelve un número que es proporcional al ángulo que se desea medir.

Los sensores infrarrojos se pueden leer de forma analógica o digital. Si se usa el valor analógico, la medición es muy sensible a cualquier cambio en la iluminación del ambiente y se debe usar el convertidor analógico digital, esto vuelve más lento el proceso de adquisición. Pero la ventaja es que la señal analógica puede ser

muestreada a cualquier rapidez y facilita el manejo por parte del controlador.

Si se usa el valor digital, la señal que se obtiene es más robusta frente al ruido ambiental, el proceso de leer los sensores se ejecuta rápidamente, pero el usar valores digitales hace que sea menos apropiada para ser manejada por el controlador[7], pues se introduce un error de cuantificación.

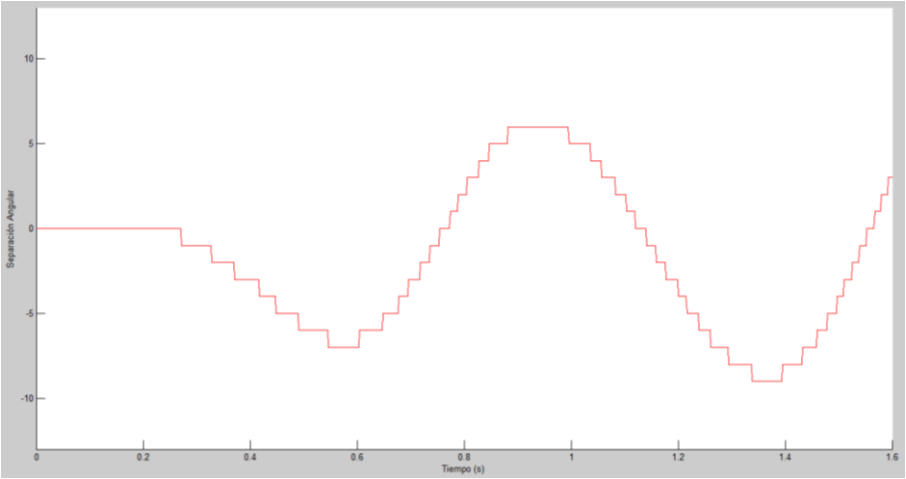


Figura 3.49 Medición de la separación angular usando los sensores infrarrojos.

En la Figura 3.49 muestra la señal proveniente de los sensores infrarrojos leídos de forma digital.

La solución que se implementó fue complementar los valores digitales de los sensores con la velocidad angular medida por el

[7] Por ejemplo, no se puede obtener una primera diferencia, pues los valores de la señal están cuantificados.

giroscopio, la unión de ambos datos da como resultado una señal analógica que es mejor manejada por el controlador.

Por ejemplo, en la situación en la que el robot gira aproximándose a un tramo recto de la pista, el giroscopio medirá la velocidad angular a la que se acerca el robot a la línea, lo cual corresponde a la derivada de la separación angular medida por los sensores infrarrojos, es decir ambos sensores miden datos que están relacionados.

Pero no siempre ocurre esto, por ejemplo, puede ocurrir que el robot se esté desplazando en línea recta y de repente la línea de la pista se curva hacia un lado. En esa situación los sensores infrarrojos miden la separación angular del robot a la línea pero el giroscopio mide la velocidad angular de giro del robot, que en ese caso es cero y no corresponde a la derivada del ángulo que miden los sensores infrarrojos.

Entonces, se tiene que, los sensores infrarrojos miden el ángulo deseado pero lo hacen con un error de cuantificación y el giroscopio mide la velocidad angular del robot, que no en todos los casos esta coincide con la derivada del ángulo deseado.

Lo que se hace es implementar un filtro de Kalman que use la información obtenida de ambos sensores y realice una estimación de la medida de este ángulo de tal manera que se obtenga un

mejor resultado que usar un único sensor para realizar la medición.

Como en el caso de los codificadores rotatorios, lo primero es establecer el modelo matemático a usar.

Para esto se debe observar el comportamiento de la rapidez de cambio del ángulo entre el robot y la línea de la pista, esta rapidez se puede dividir en:

Ecuación 3. 21

$$\omega_T = \omega_g + \omega_p$$

En la ecuación ω_T representa la velocidad a la que gira el robot respecto a la línea de la pista, ω_g es la velocidad a la que gira el robot respecto al suelo, esta es medida por el giroscopio, y ω_p es la velocidad a la que se "aleja" la pista del robot, es decir está relacionada con las curvas que tenga la pista, depende de la forma de la pista y de la velocidad lineal del robot. Por ejemplo en un tramo recto esta velocidad es cero, pero tendrá un valor diferente de cero al pasar de un tramo recto a uno curvo. Así mismo será menor si el robot avanza muy lento y será mayor si el robot avanza rápido. Se puede discutir sobre la rigurosidad de la Ecuación 3. 21, pero una ventaja del filtro de Kalman es que puede afrontar errores de modelización.

Ahora se nombra $\dot{x_1}$ a ω_T y x_2 a ω_p:

$$\dot{x_1} = x_2 + \omega_g$$

Lo que se necesita es que la salida del sistema sea x_1, esta variable es el ángulo entre la línea y el robot.

Para tomar como variables de estado a x_1 y x_2 se debe completar el sistema de ecuaciones, por lo que se necesita información sobre la dinámica de la variable x_2, esta variable representa la velocidad angular con la que se aleja la pista del robot, sobre esta variable no se tiene información ya que, como se mencionó, depende de la forma de la pista y de la velocidad del robot, así que lo que se hizo es modelizar x_2 como una constante, esto ya se hizo para el modelo matemático del filtro para los codificadores rotatorios.

Entonces el sistema de ecuaciones en tiempo continuo es:

Ecuación 3. 22

$$\dot{x_1} = x_2 + \omega_g$$

Ecuación 3. 23

$$\dot{x_2} = 0$$
$$y = x_1$$

Expresado en Matrices:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \omega_g$$
$$y = (1 \quad 0) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Se necesita que este sistema este expresado en tiempo discreto, para este sistema es simple obtener la forma en tiempo discreto para cualquier tiempo de muestreo T:

$$x_{1_{k+1}} = x_{1_k} + T(x_{2_k} + \omega_{g_k})$$
$$x_{2_{k+1}} = x_{2_k}$$

La segunda ecuación indica que la variable discreta x_2 es constante, similar a la Ecuación 3. **22** en tiempo continuo, y la primera ecuación indica que la variable discreta x_{1_k} esta definada por una razón de cambio o más específicamente una primera diferencia igual a $(x_{2_k} + \omega_{g_k})$, como en la Ecuación 3. **23** en tiempo continuo.

Entonces el sistema en tiempo discreto con un tiempo de muestreo T, expresado en matrices es:

$$\begin{pmatrix} x_{1_{k+1}} \\ x_{2_{k+1}} \end{pmatrix} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{1_k} \\ x_{2_k} \end{pmatrix} + \begin{pmatrix} T \\ 0 \end{pmatrix} \omega_{g_k}$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_{1_k} \\ x_{2_k} \end{pmatrix}$$

Para simplificar la implementación, se puede reemplazar la variable de estado x_{2_k} por una nueva variable $x_{2_k}{}' = Tx_{2_k}$y ω_g por $\omega_g{}'_k = T\omega_{g_k}$:

$$\begin{pmatrix} x_{1_{k+1}} \\ x_{2_{k+1}}{}' \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_{1_k} \\ x_{2_k}{}' \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \omega_g{}'_k$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_{1_k} \\ x_{2_k}{}' \end{pmatrix}$$

De manera similar a lo que se hizo para el filtro de los codificadores rotatorios, se implementó el algoritmo completo de filtrado de Kalman en un script de MATLAB para luego implementar el algoritmo de Kalman en estado estable en la FPGA.

Para ejecutar el script se necesita como entrada las señales medidas por el giroscopio y por los sensores infrarrojos, además se debe especificar los valores para las matrices de covarianza Q y R. Las matrices Q y R encontradas son:

$$Q = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.00005 \end{pmatrix}$$

$$R = 900$$

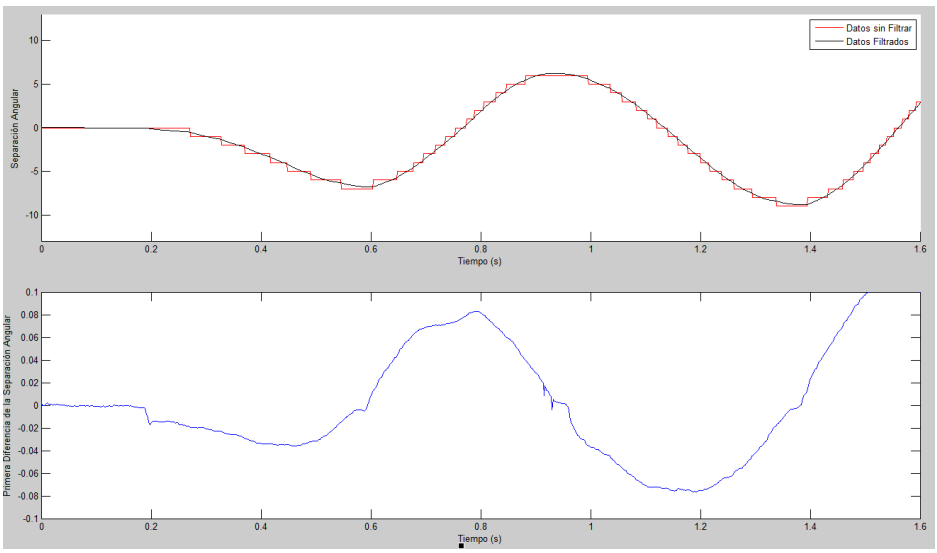


Figura 3.50 Salida del Filtro unificando datos de los sensores infrarrojos y del giroscopio.

La Figura 3.50 muestra la separación angular y su primera diferencia, es decir las señales x_{1_k} y ${x_{2_k}}' + {\omega_g}'_k$ del modelo.

La gráfica que se obtiene para los elementos de la matriz de ganancia K, se muestra en laFigura 3.51.

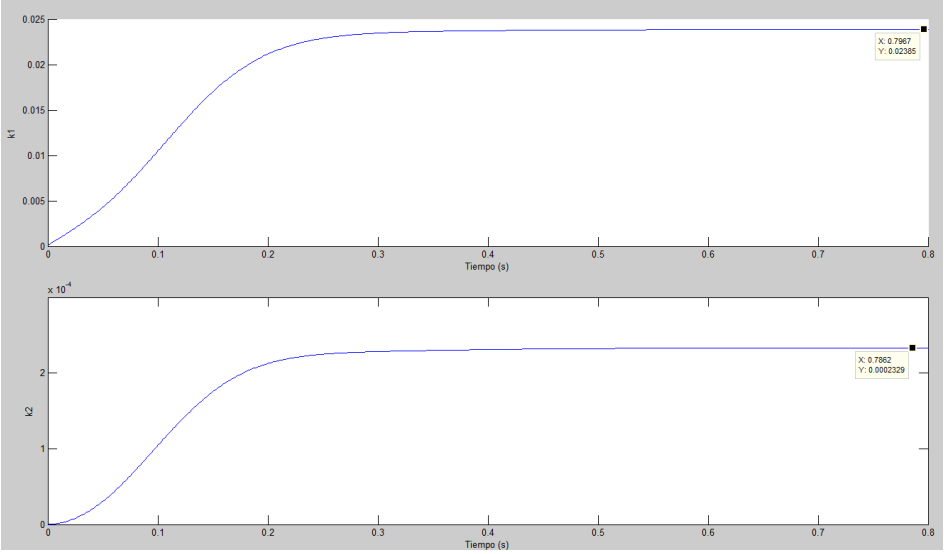


Figura 3.51 Convergencia De Los Elementos De La Matriz De Ganancia Para El Filtrado De La Señal De Los Sensores Infrarrojos.

Los valores a los que converge cada elemento forman la matriz de ganancia en estado estable, con esta matriz el sistema a implementar es:

$$x_{k|k-1} = Ax_{k-1|k-1} + Bu_k$$
$$y_k = z_k - Cx_{k|k-1}$$
$$x_{k|k} = x_{k|k-1} + Ky_k$$

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$
$$B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$C = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

$$K = \begin{pmatrix} 0.0238466 \\ 2.32875e-04 \end{pmatrix}$$

3.2.4 Controlador Principal

El controlador principal es el encargado de definir la velocidad a la que debe girar cada rueda, es decir fija la referencia del controlador de velocidad de cada motor. Para tomar esta decisión se basa en la posición angular y en la velocidad lineal que necesita tener el robot.

Se debe conocer cuál es la función de transferencia que relaciona la posición angular del robot con la velocidad de cada motor, de la Ecuación 3. **4** se tiene:

$$\omega = \frac{v_r - v_l}{l}$$

La posición angular θ es la integral de la velocidad angular, usando la transformada de Laplace se tiene que:

$$\theta = \frac{v_r - v_l}{ls}$$

Ecuación 3. 24

$$\frac{\theta}{v_r - v_l} = \frac{1/l}{s}$$

Por lo que para controlar la posición angular del robot se debe diseñar un sistema de realimentación que controle la diferencia de

velocidades de cada motor realimentándose con la posición angular medida.

Para el caso de la velocidad lineal, de la Ecuación 3. **5** , se tiene que la velocidad del centro de masa localizado en una posición x, como se muestra en la Figura 3.3 es:

$$v_x = \sqrt{(\frac{v_r + v_l}{2})^2 + (\frac{v_r - v_l}{l} x)^2}$$

Se puede ver que la velocidad del centro de masa depende de la diferencia de velocidad de las ruedas y de la suma de velocidades. Pero la diferencia también controla la posición angular, por lo que esta cantidad no puede ser modificada.

Entonces para cambiar la velocidad del centro de masa únicamente se puede variar la cantidad $\frac{v_r+v_l}{2}$ y con esto ya la velocidad del centro de masa se define, esta cantidad equivale a la velocidad lineal cuando el centro de masa está a una distancia cero:

$$v_0 = \frac{v_r + v_l}{2}$$

Debido a que no se necesita que la velocidad del centro de masa tome un valor específico sino que simplemente se necesita que esta sea lo mayor posible, lo que se hace es controlar v_0 para que

esto implique controlar v_x, sin importar cuál sea el valor que tome v_x.

Tomando en cuenta esto, la función de transferencia que relaciona la velocidad lineal con las velocidades en el punto de contacto de cada rueda es simplemente:

$$\frac{v_0}{v_r + v_l} = \frac{1}{2}$$

Es decir que para controlar la velocidad lineal del robot se necesita controlar la suma de las velocidades de cada rueda en el punto de contacto. Para esto basta con fijar la velocidad de referencia para el controlador de cada motor.

Hay que notar que no se conoce la función de transferencia que relaciona las velocidades de referencia de los controladores de los motores con la posición angular, la Ecuación 3. **24** relaciona la velocidad real o medida por los codificadores rotatorios con la posición angular, no la velocidad de referencia.

Esto hace que sea complejo diseñar un controlador usando MATLAB, pues MATLAB solo puede identificar sistemas de primer y segundo orden. Entonces para la posición angular se diseñó un controlador proporcional derivativo, este tipo de controladores

pueden ser ajustados manualmente sin necesidad de conocer la función de transferencia de la planta.

La forma del controlador es:

$$C = \frac{(K_p + K_d)z - K_d}{z}$$

Para implementar en la FPGA el controlador se lo debe llevar al domino del tiempo, U es la salida del controlador que corresponde a la diferencia de velocidades de los motores y Err_D es la señal de error de entrada:

$$C = \frac{U_D}{Err_D} = (K_p + K_d) - K_d z^{-1}$$

Pasando al dominio del tiempo:

$$u_D(k) = K_p err_D(k) + K_d(err_D(k) - err_D(k-1))$$

Las constantes K_py K_d fueron halladas de forma manual. Para esto se fija la velocidad lineal del robot en cero y se aplica un impulso, esto significa colocar el robot a una cierta separación angular y luego activar el controlador, el robot girará hasta apuntar en dirección a la pista. La Figura 3.52 muestra este procedimiento.

Para facilitar el ajuste de las constantes del controlador se escribió una rutina en C en la FPGA, esta permite recibir los valores de las

constantes por el puerto serial a través del módulo Bluetooth y ajustarlos de forma inalámbrica.

Figura 3.52 Ajuste de las constantes del Controlador de posición.

Las constantes encontradas fueron:

$$K_p = 15$$

$$K_d = 130$$

3.2.5 Rapidez de Procesamiento

La velocidad máxima esperada para el robot es de al menos 2 m/s, con el robot moviéndose a esa velocidad el procesador debe ser lo suficientemente rápido para detectar un mínimo cambio en la lectura de los sensores infrarrojos.

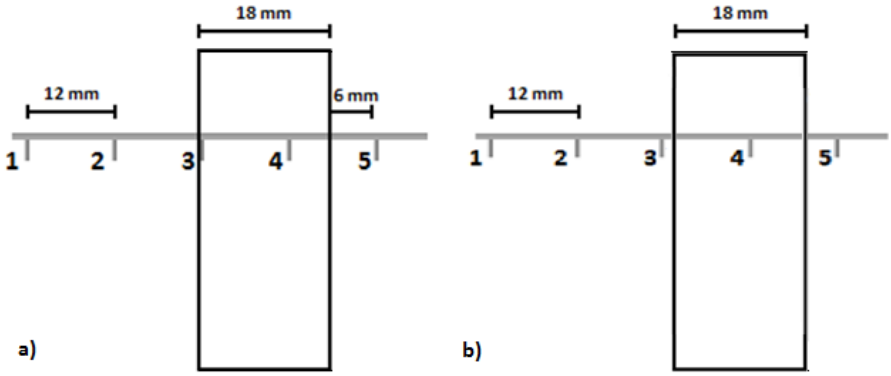


Figura 3.53 Posición del Arreglo de Sensores Infrarrojos respecto a la pista.

El arreglo de sensores detecta la línea de la pista cuando esta se encuentra bajo los sensores, la separación entre dos sensores contiguos es de 12 mm y el ancho de la línea que forma la pista es 18 mm. Si el arreglo está perpendicular a la marca de la pista entonces la pista puede ser detectada por un sólo sensor o por dos sensores a la vez. Por ejemplo, en la Figura 3.53 a) el arreglo de sensores está en una primera posición en la que se detecta la pista con dos sensores a la vez representados con los números 3 y 4, pero si el arreglo de sensores se mueve hacia la izquierda sólo el sensor 4 detectará la pista, como en la Figura 3.53 b), y si el movimiento continúa entonces los sensores 4 y 5 detectarán la pista. Continuando con este ejemplo, se tiene entonces que el arreglo puede estar en tres posiciones diferentes. El procesador debe ser capaz de detectar estas tres posiciones, aunque el robot se mueva a su máxima velocidad. Si el robot está en la primera

posición, lo más cerca que está el robot de llegar a la tercera posición es 6mm, como se muestra en la Figura 3.53 a).

Entonces para que el procesador no detecte la segunda posición lo peor que puede suceder es que la lectura del sensor ocurra justo antes de entrar a la segunda posición, por lo que el robot debe recorrer sólo 6 mm. Si el robot se mueve a 2m/s, entonces al robot le tomaría 3ms pasar de la primera posición a la tercera posición.

Por lo que para evitar no leer la segunda posición el procesador debe realizar una nueva medición antes de que transcurran 3ms, este valor es un límite superior para el tiempo con el que se debe repetir la lectura de sensores, tomando como velocidad máxima del robot 2 m/s.

Basado en esto se necesita que la interrupción ocurra por lo menos cada 3 ms, el tiempo de interrupción para la ejecución del controlador principal que se uso es 1.5 ms. Este tiempo es suficiente para leer los sensores, filtrar los datos, ejecutar el algoritmo de control y fijar el ciclo de trabajo de la señal PWM que se envía a los motores.

Adicional a esto se tiene que para la respuesta del controlador de velocidad de los motores sea rápida se debe tener un tiempo de

muestreo lo suficientemente pequeño, en Matlab se diseñó controladores con tiempo de muestreo de 1.5 ms, pero tuvieron un peor desempeño que los de tiempo de 0.5 ms. En la Figura 3.54 se muestra la respuesta del sistema a un escalón de amplitud 30 con el mejor controlador de 1.5ms encontrado.

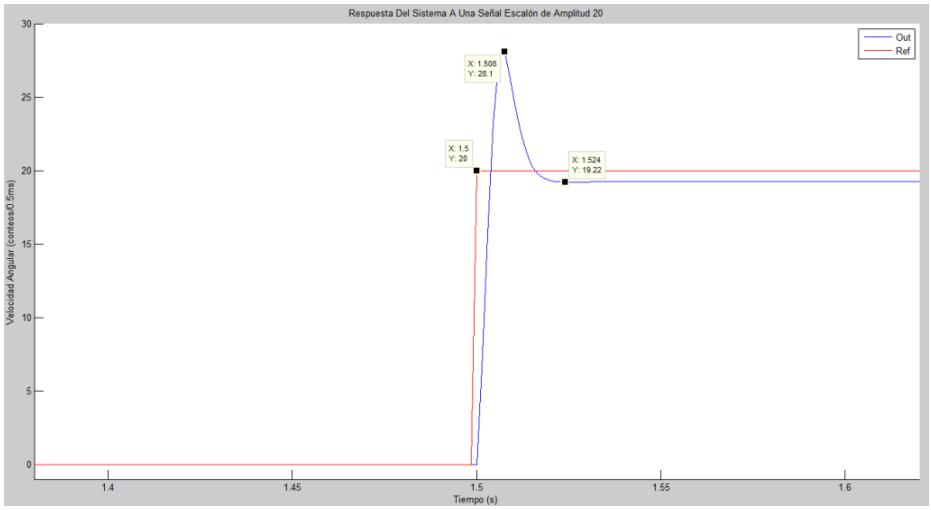


Figura 3.54 Simulación del sistema realimentado del motor con un tiempo de muestreo de 1.5ms.

Se observa que el tiempo de estabilización es 24 ms, el error estado estacionario es 0.78 y el valor pico de la respuesta es 28.1, es claro que la respuesta para el controlador de 0.5ms de la Figura 3.42 es mejor.

Para lograr que se ejecuten todas las tareas dentro del tiempo deseado y no se afecte el desempeño del controlador de velocidad, se implementó dos interrupciones una a 1.5 ms que

actualiza los datos del sistema de control principal y otra interrupción a 0.5 ms que actualiza el controlador de velocidad.

Es importante controlar el tiempo que le toma al procesador ejecutar las tareas entre interrupciones para asegurarse que no se sobrepase el tiempo establecido, para esto se usó el módulo de hardware llamado Performance Counter.

En la medición se obtuvo un tiempo mínimo de 0.829 ms y máximo de 1.14 ms, este tiempo incluye las tres rutinas de interrupción de 0.5 ms que ocurren dentro de una de 1.5 ms. En la sección de resultados se muestra el tiempo de adquisición de datos, algoritmo de control y filtrado.

3.3 Diseño Final del Robot

En las secciones anteriores se explicó sobre el sistema de control y el algoritmo de filtrado de forma independiente, en esta sección se muestra cómo se unificó estos bloques para formar el sistema completo de filtrado y control. Y en la parte final se muestra el diseño físico del robot con el circuito impreso y las partes mecánicas.

En la Figura 3.55 muestra el sistema realimentado para cada motor. El bloque M(z) representa la función de transferencia del motor, esta

función tiene como entrada el voltaje de alimentación del motor y como salida la velocidad angular del motor.

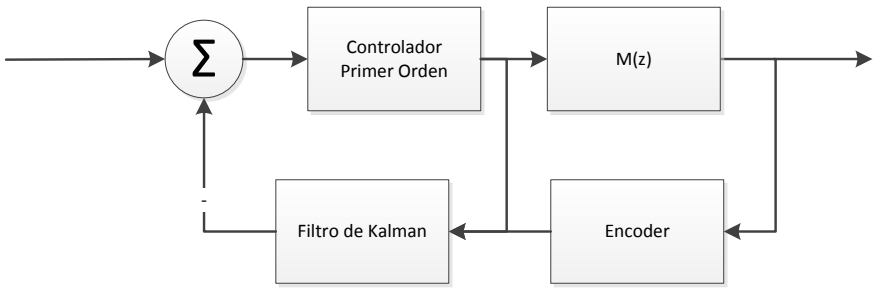


Figura 3.55 Sistema de Realimentación para Control de Velocidad del Motor.

En el código del programa la interrupción ocurre cada 0.5ms, entre cada interrupción se ejecuta el algoritmo de filtrado con parámetro de entrada la velocidad angular medida por los codificadores y el último valor de PWM enviado al motor. El algoritmo de Kalman con estos parámetros realiza una estimación de la velocidad angular del motor, este valor es restado del valor de referencia para obtener la señal de error y con esta señal como parámetro de entrada se ejecuta el controlador. El controlador en base al error medido decide el nuevo valor de PWM y el lazo continúa cuando ocurre la siguiente interrupción. Este sistema es implementado para cada motor.

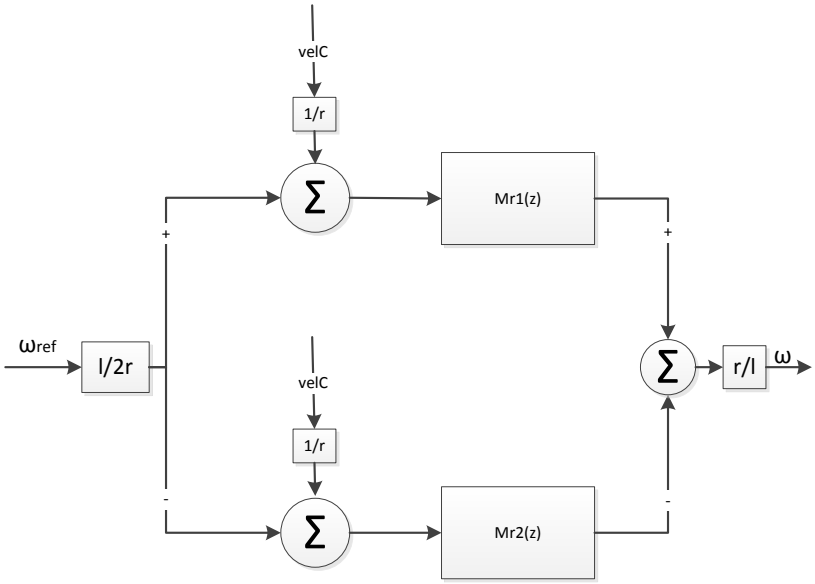


Figura 3.56 Función de transferencia entre la Velocidad Angular de Referencia y Real del Robot.

En la Figura 3.56 se muestra la función de transferencia que tiene como entrada el valor de referencia de la velocidad angular del robot y como salida la velocidad angular del robot. EL bloque Mr1 representa el sistema de realimentación mostrado en la Figura 3.55 para un motor y el bloque Mr2 representa el sistema para el otro motor. A la salida de los bloques Mr1 y Mr2 la velocidad angular será similar al valor de entrada, ya que estos son sistemas realimentados y fueron diseñados para esto.

Todo el bloque es simplemente la representación gráfica de la Ecuación 3. **4** y la Ecuación 3. **6**. Usando la velocidad angular de cada motor en lugar de la lineal se tiene:

Ecuación 3. 25

$$\omega = (\omega_r - \omega_l)\frac{r}{l}$$

Ecuación 3. 26

$$v_c = (\omega_r + \omega_l)\frac{r}{2}$$

Donde r representa el radio de la rueda.

De estas ecuaciones se tiene que para que la velocidad lineal del robot sea v_c y la velocidad angular del robot sea ω , los valores de las velocidades angulares de cada motor, ω_r y ω_l son:

$$\omega_r = \frac{v_c}{r} + \frac{\omega l}{2r}$$
$$\omega_l = \frac{v_c}{r} - \frac{\omega l}{2r}$$

Estas últimas ecuaciones son las que se representa en la Figura 3.56, el valor v_c corresponde a velC y ω corresponde a ω_{ref}. El valor velC es un valor que permanece constante y corresponde a la referencia para la velocidad lineal que tendrá el robot. En la salida del sistema, para obtener la velocidad angular del robot estos valores se restan y se multiplican por $\frac{r}{l}$, según la Ecuación 3. **25**.

Todo el sistema descrito es resumido en un bloque llamado H y es representado junto al sistema principal de realimentación en la Figura 3.57.

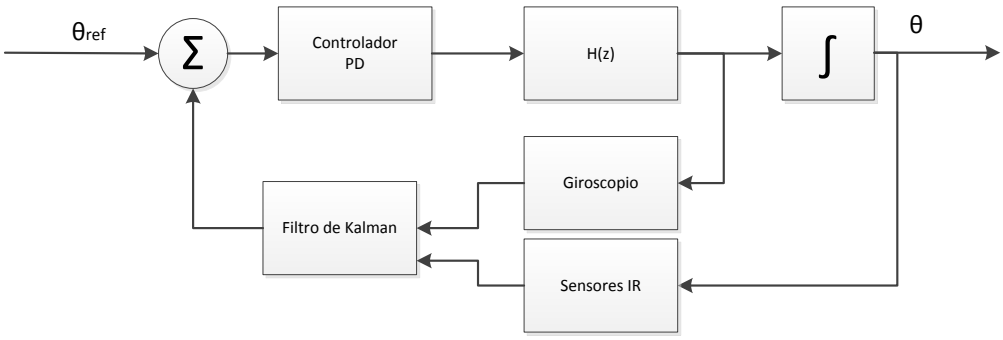


Figura 3.57 Sistema de Realimentación para el control de la Posición angular.

El sistema de la Figura 3.57 sirve para controlar la posición angular del robot. En software lo que se hace es, en cada interrupción de 1.5ms, ejecutar las ecuaciones del algoritmo de filtrado con parámetro de entrada la lectura de la velocidad angular del giroscopio y la lectura de la posición angular de los sensores infrarrojos. El algoritmo fusiona ambos datos y hace una estimación de la posición angular, este valor es restado de la referencia para obtener la señal de error que entra al controlador.

El controlador en base al error medido decide el nuevo valor de referencia para el bloque H. A la salida de este bloque se tienen la velocidad angular del robot y luego su integral que es la posición angular, estos valores son medidos nuevamente en la siguiente interrupción y el lazo se repite.

En el sistema completo se deben establecer dos niveles de referencia, los valores θ_{ref} y velC, el primero tiene el valor de cero, esto es porque este ángulo indica la separación angular del robot con la línea de la

pista, lo que se desea es que el robot no se separe de la pista. El segundo valor puede tomar cualquier valor constante dentro de los límites de velocidad del robot, este valor define la velocidad del centro de masa.

Diseño físico.

La realización del proyecto implicó la construcción de dos modelos, en la Figura 3.58 se muestra el primer prototipo construido.

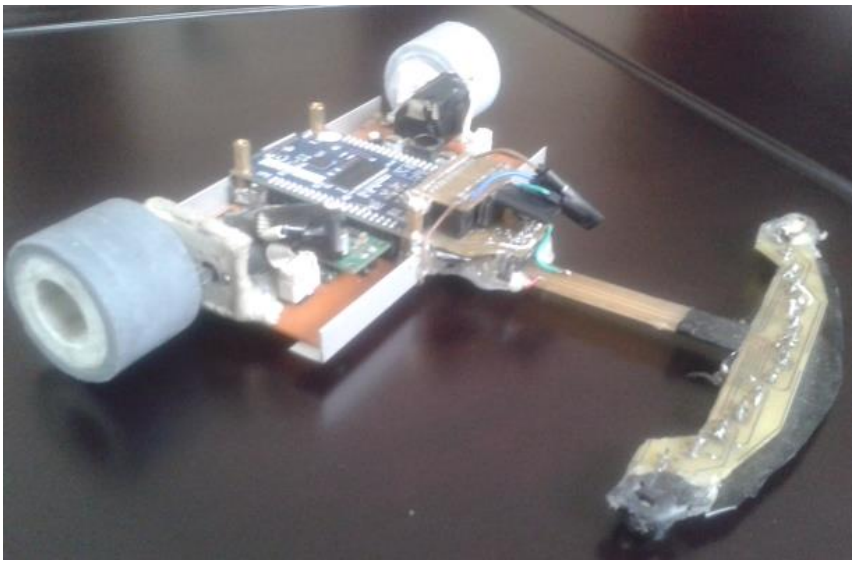


Figura 3.58 Primer prototipo Construido

En este diseño el PCB fue hecho de baquelita, este material es poco resistente por lo que se tuvo que agregar partes de aluminio para sostener el chasis, además los motores se colocaron a cada lado del robot, esto ocasionaba un incremento del momento de inercia. Para alcanzar la velocidad de 2 m/s se tuvo que usar ruedas de 2.1 cm de

radio ya que los engranes que se disponían daban una relación de 8:1 y no permitían una mayor velocidad con un radio menor.

Para el diseño final el circuito fue impreso en FR-4 a doble lado, su tamaño se redujo respecto al primer diseño. Son dos las placas que forman el robot, estas se muestran en la Figura 3.59.

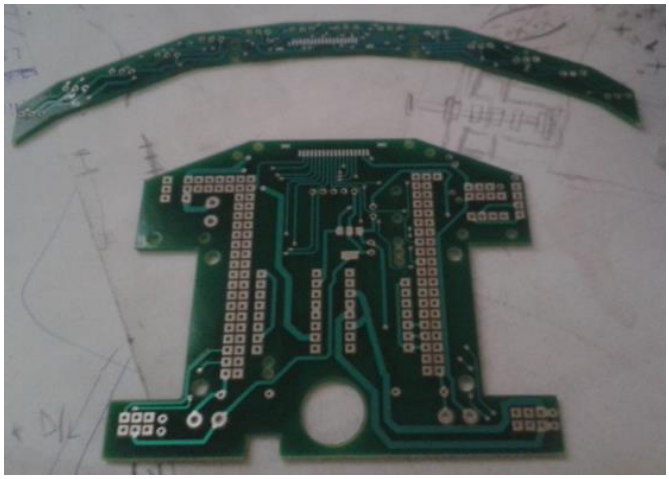


Figura 3.59 Circuito Impreso del Diseño Final

Además se utilizó el software SolidWorks para diseñar los soportes que sostienen los motores y los aros donde van las llantas del robot, los diseños se muestran en la Figura 3.60, Figura 3.61 y Figura 3.62.

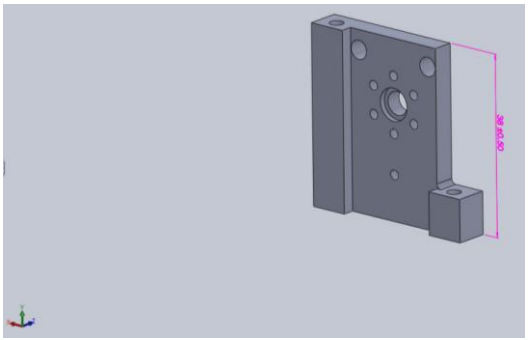


Figura 3.60 Soporte Izquierdo

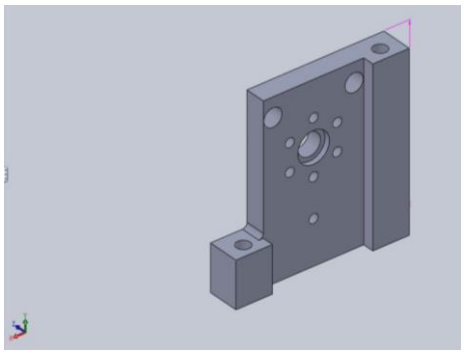


Figura 3.61 Soporte Derecho

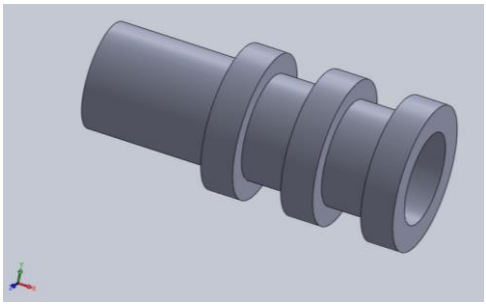


Figura 3.62 Aro

Estos diseños fueron impresos utilizando una impresora 3D, las impresiones se muestran en la Figura 3.63 y Figura 3.64.

Figura 3.63 Aro Impreso en 3D

Figura 3.64 Soporte Impreso en 3D

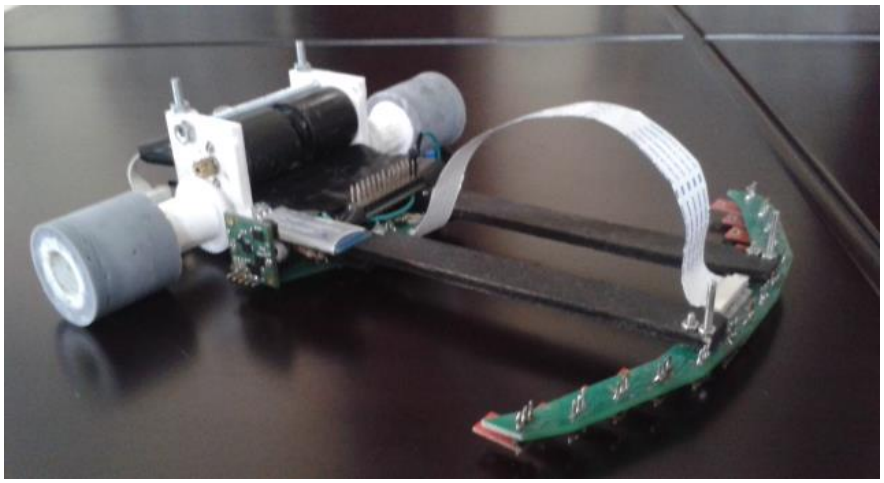


Figura 3.65 Diseño Final del Robot

En la Figura 3.65 se muestra el diseño final del robot. Las placas se unen con dos partes de madera, los circuitos se conectan usando cable plano flexible. Las ruedas tienen un radio de 1.5 cm, esta reducción del radio respecto al primer diseño fue posible cambiando la relación del sistemas de engranajes a 48:10.

CAPÍTULO 4

4 PRUEBAS Y RESULTADOS EXPERIMENTALES

En este capítulo se muestra los resultados obtenidos al realizar pruebas para analizar el desempeño del robot. En la primera sección se muestra las mediciones del tiempo que le toma al sistema en ejecutar cada una de las tareas, estas fueron divididas en adquisición, filtrado y control.

En la segunda sección se muestra el resultado de la ejecución de los algoritmos de control y filtrado ya implementados sobre la FPGA.

Finalmente en la tercera sección se muestran mediciones hechas con el robot funcionando en un circuito de competición y la medición del ángulo de inclinación.

4.1 EVALUACIÓN DE LA RAPIDEZ DE PROCESAMIENTO

En la sección 3.2.5 se explicó acerca del tiempo que debe transcurrir entre cada interrupción del controlador principal, este se fijó en 1.5ms. Dentro de este tiempo el sistema debe ejecutar tareas de adquisición, filtrado y control.

Es importante revisar que el tiempo de ejecución de estas tareas no sobrepasen el tiempo de interrupción, si lo hacen el algoritmo no se ejecutará correctamente y el robot tendrá un comportamiento errático.

A continuación se analiza el tiempo de ejecución que le toma al sistema realizar cada una de las tareas.

Tabla III Tiempos de Procesamiento

		Tiempo mínimo (uS)	Tiempo Máximo (uS)
Adquisición	Infrarrojos	54.01	97.30
	IMU	536.37	556.96
	Codificadores	0.36	1.76
	ADC	25.71	29.69
Filtrado	Motores	12.38	38.18

	Giroscopio e Infrarrojos	3.41	5.51
Control	Velocidad	49.72	100.1
	Posición	22.56	32.06

4.2 RENDIMIENTO DE ALGORITMOS

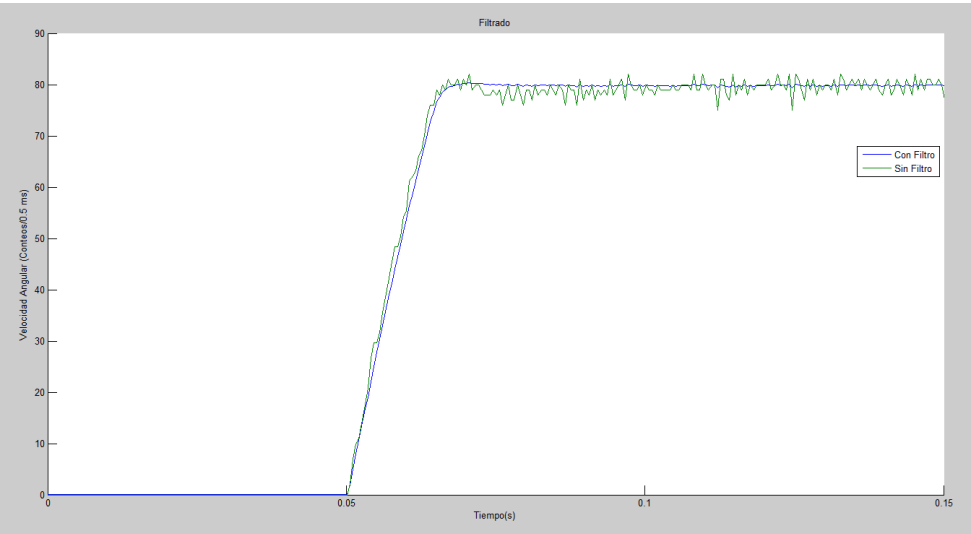


Figura 4.1 Velocidad angular del motor con y sin filtro de Kalman.

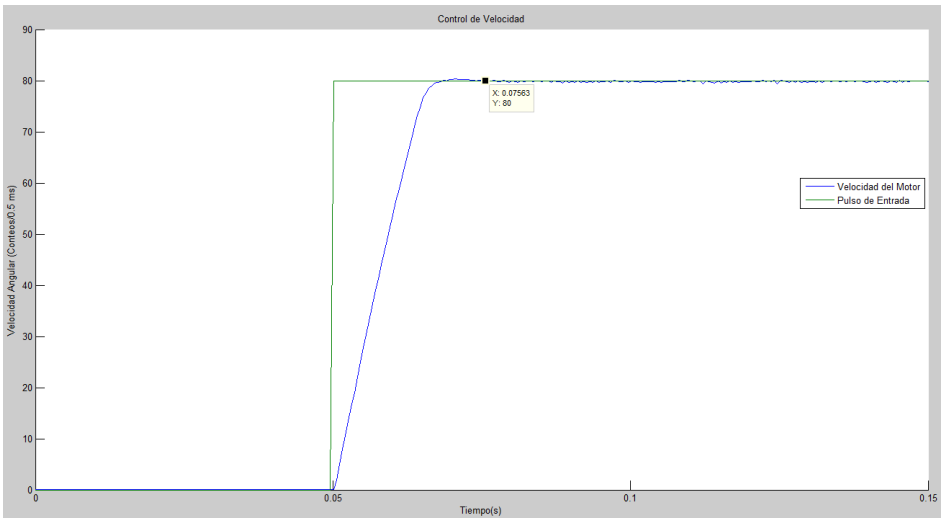


Figura 4.2 Control de velocidad angular del motor ante un pulso de entrada.

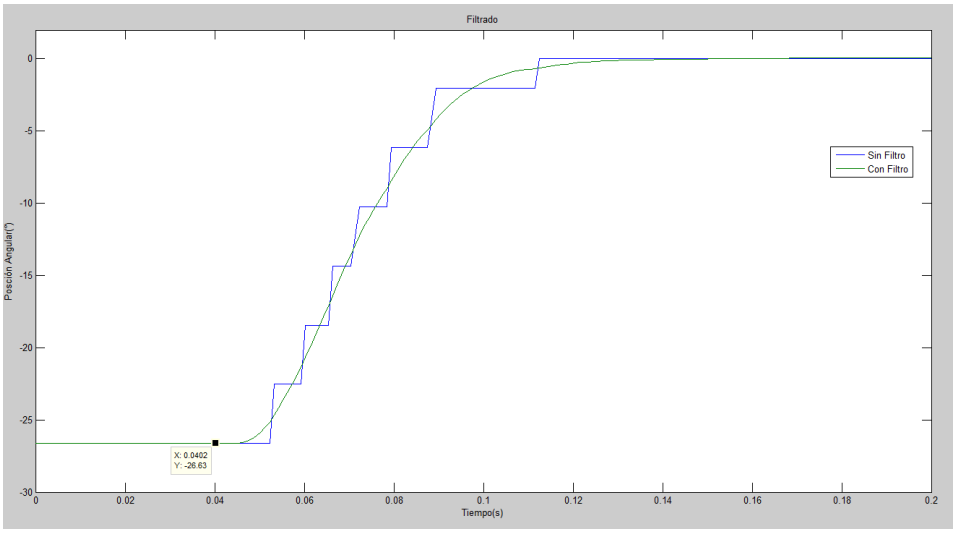


Figura 4.3 Posición angular del robot con y sin Filtro de Kalman.

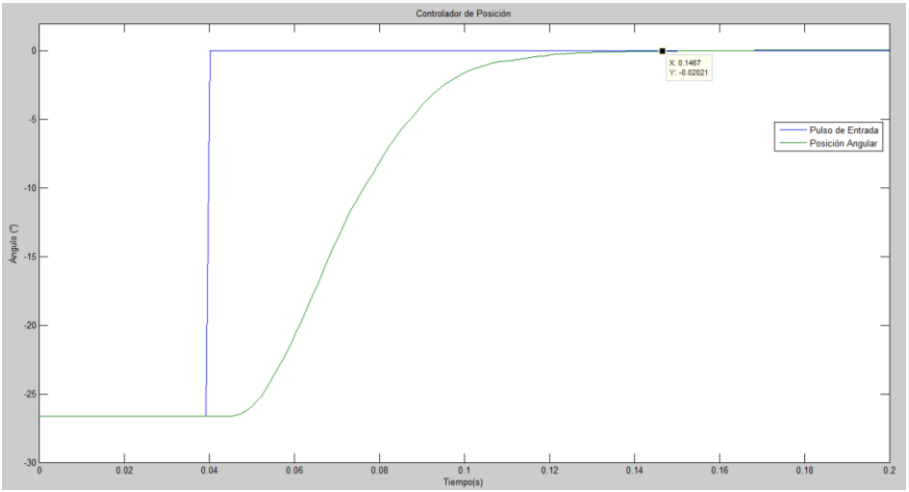


Figura 4.4 Control de la posición angular del robot.

4.3 PRUEBAS EN PISTA DEL DISEÑO FINAL

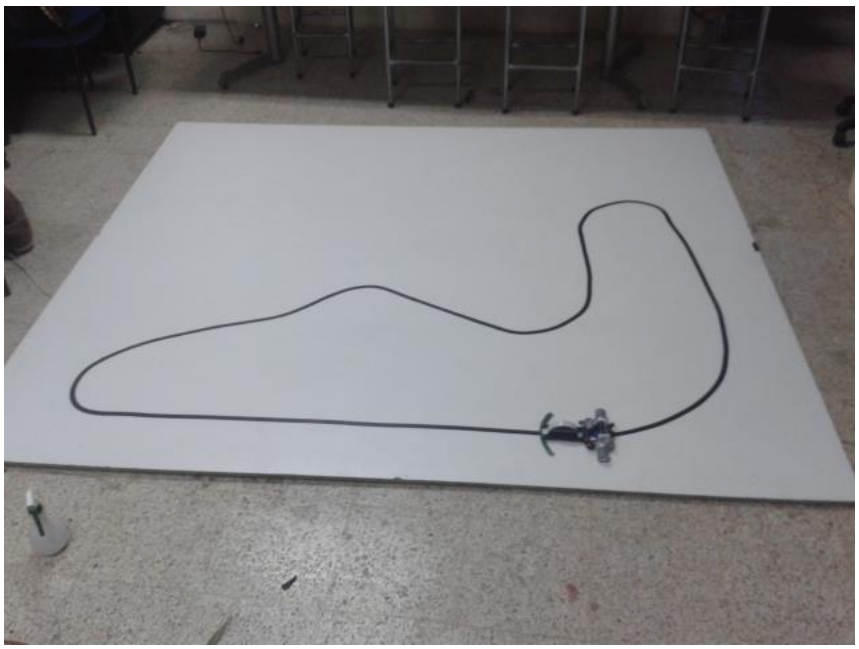


Figura 4.5 Pruebas del robot velocistas en pista.

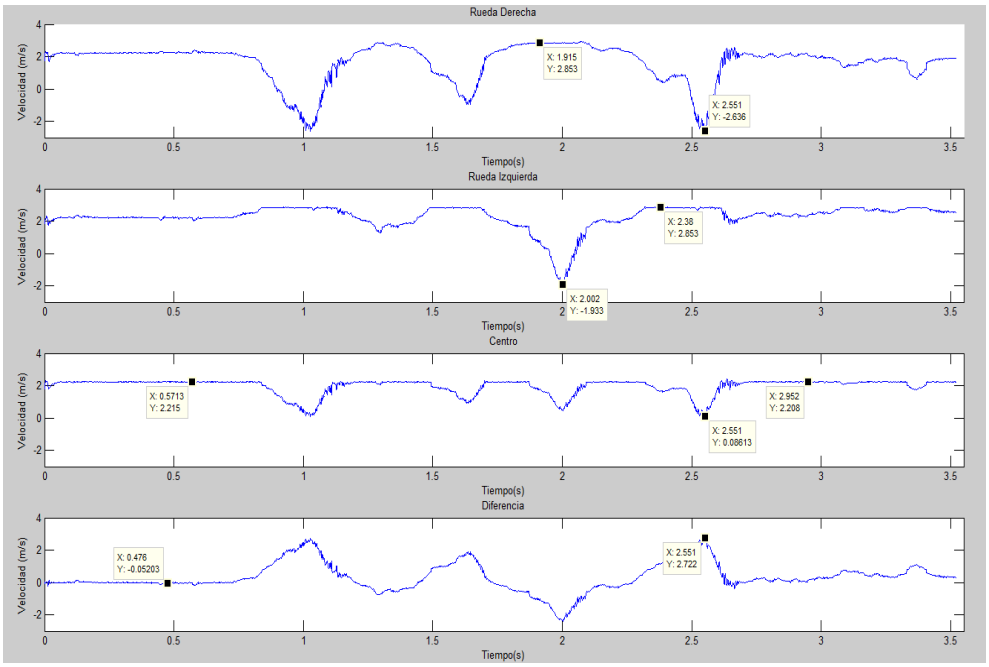


Figura 4.6 Diversas velocidades del robot velocista durante su recorrido en la pista.

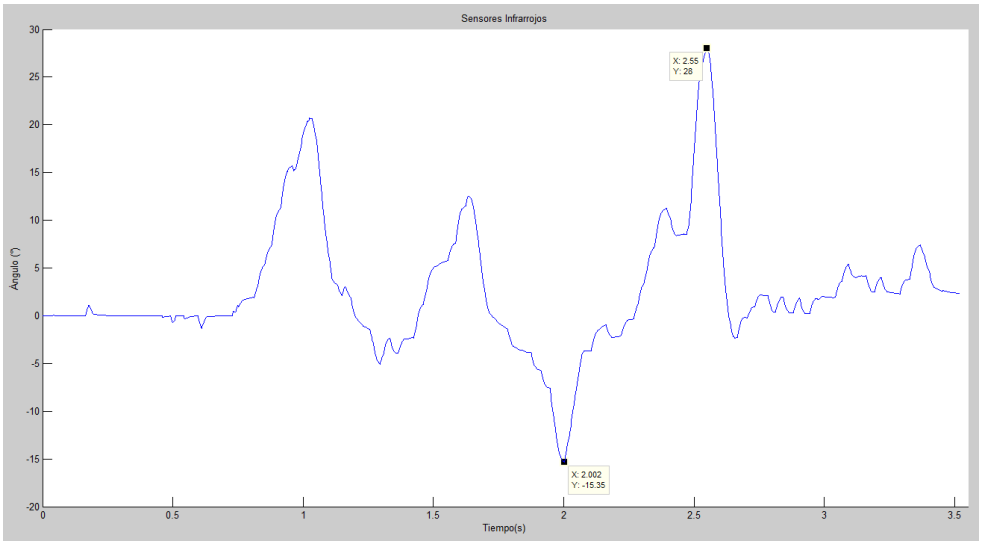


Figura 4.7 Separación angular del velocista con respecto a la pista durante su recorrido.

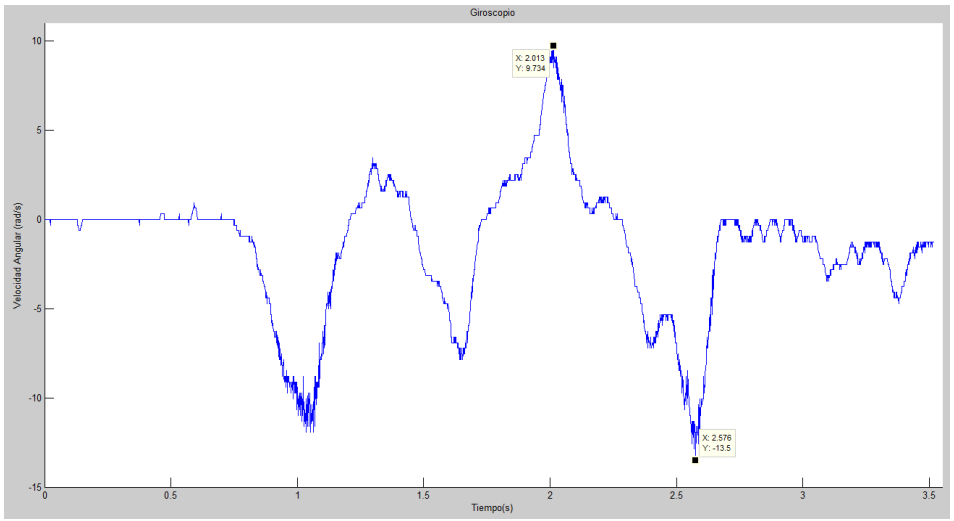


Figura 4.8 Velocidad angular de la posición del robot durante su recorrido en la pista.

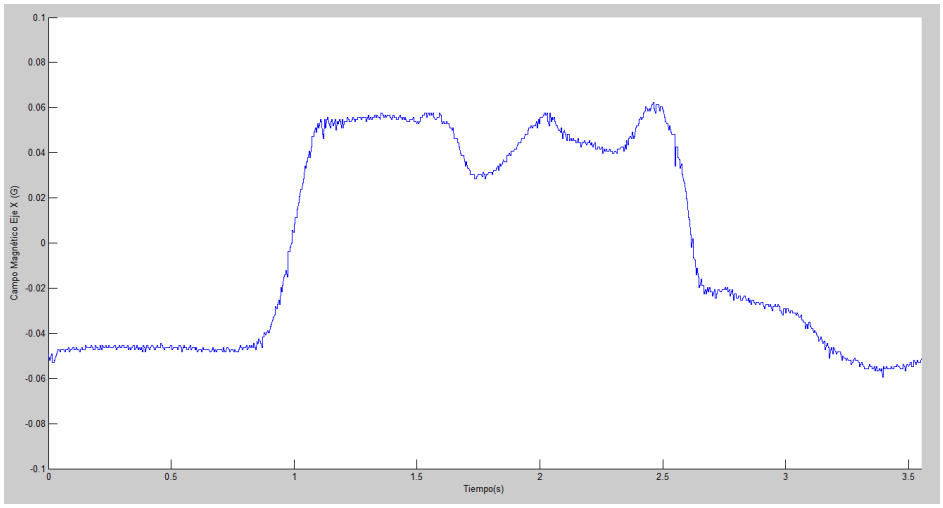


Figura 4.9 Campo magnético medido por el magnetómetro en el eje X.

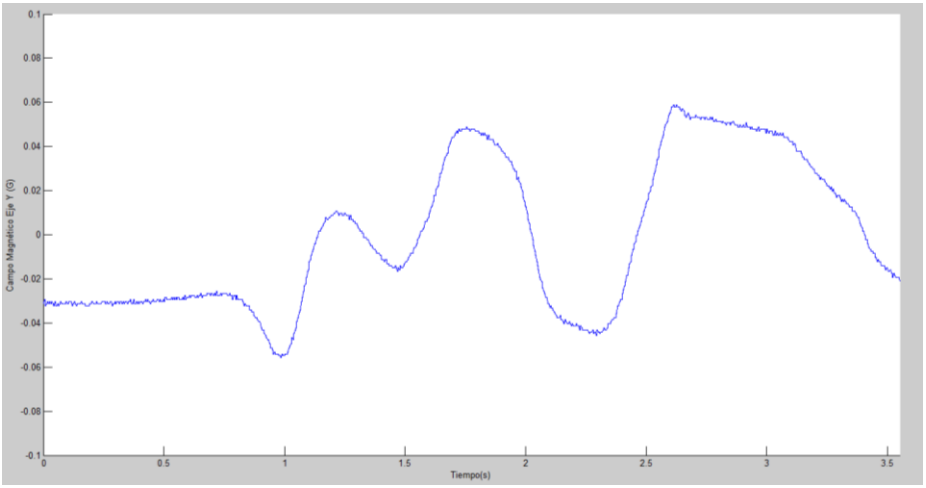


Figura 4.10 Campo magnético medido por el magnetómetro en el eje Y.

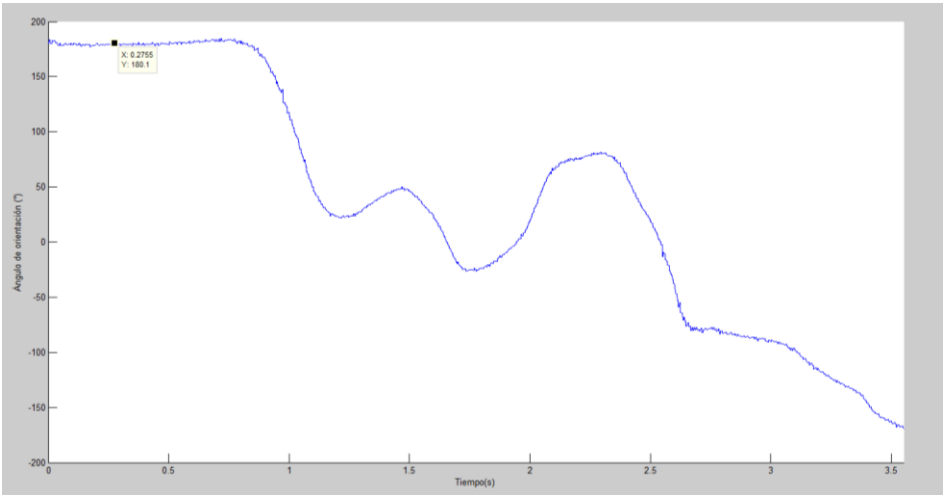


Figura 4.11 Ángulo de orientación del magnetómetro del robot con respecto al campo magnético de la tierra.

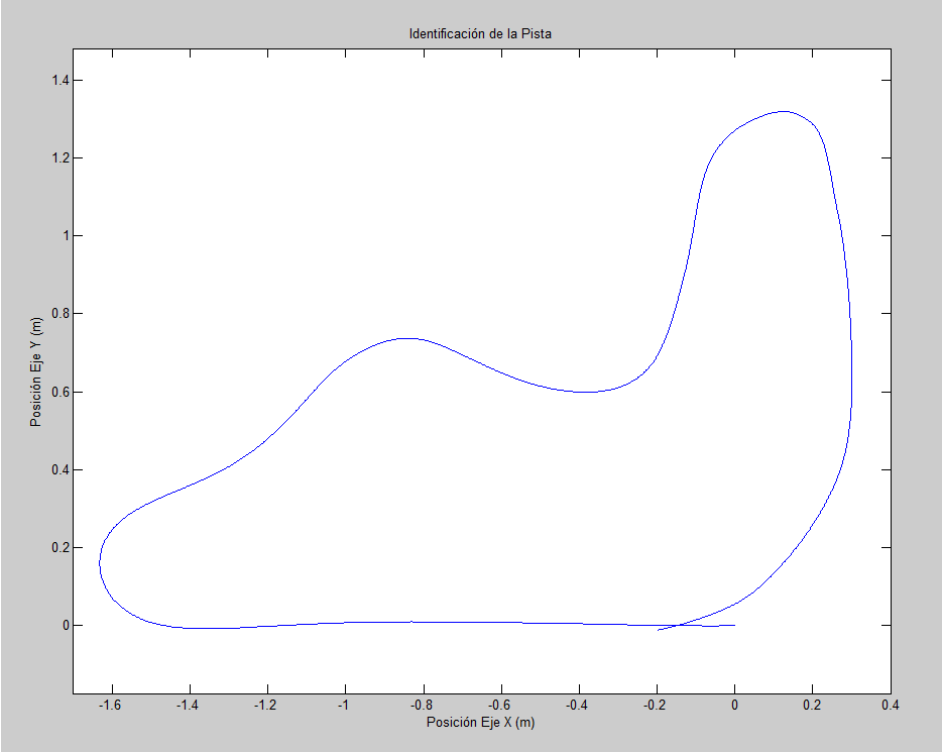


Figura 4.12 Pista recreada con diferentes datos obtenidos por sensores.

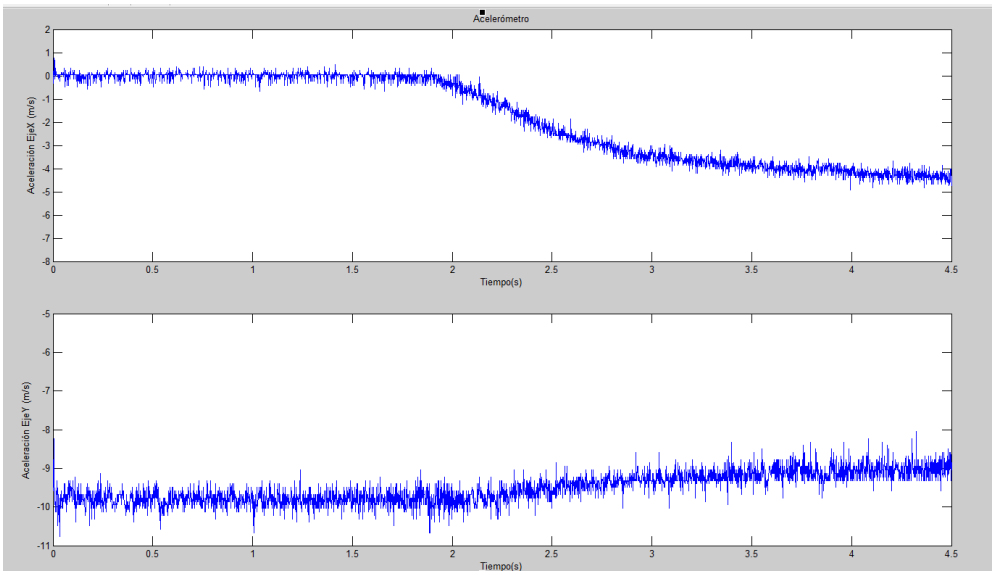


Figura 4.13 Aceleración del robot en los ejes X y Y.

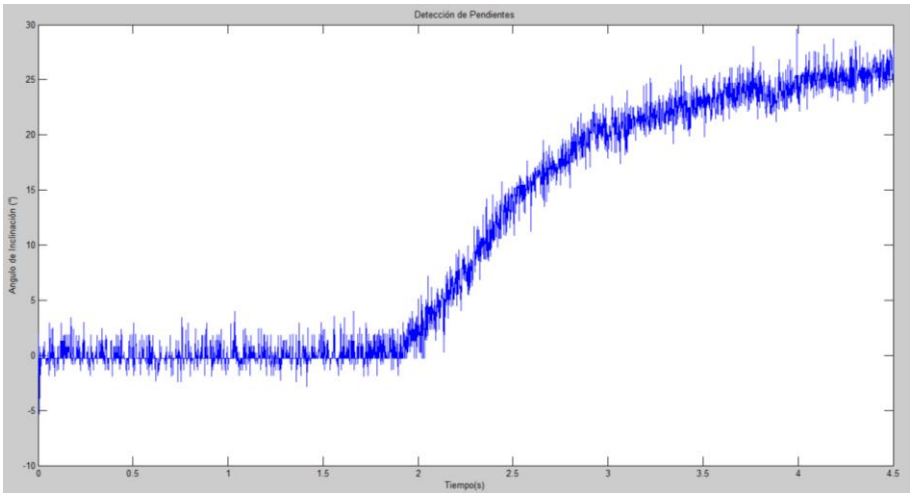


Figura 4.14 Ángulo de inclinación de la pendiente.

4.4 ANÁLISIS DE RESULTADOS

A través de los resultados obtenidos en las secciones 4.1, 4.2 y 4.3, se puede hacer el respectivo análisis del funcionamiento del diseño final del velocista.

Análisis de la rapidez de procesamiento

Con los datos de la Tabla III se puede calcular el tiempo de procesamiento de todas las tareas, entre cada interrupción de 1.5 ms se ejecutan 3 rutinas de interrupción de 0.5 ms, las tareas de lectura de codificadores filtrado de motores y control de velocidad ocurren en esta última. Entonces en el peor de los casos al sistema le toma: 1.14 ms ejecutar las tareas antes de que ocurra la siguiente interrupción de 1.5

ms, y en el mejor de los casos 0.829 ms. Por lo que se puede evidenciar que las tareas se ejecutan sin sobrepasar el tiempo de interrupción.

Además se puede observar que la lectura de la IMU es lo que más tiempo toma, esto es debido a que la lectura es serial usando el protocolo I2C.

La tarea que más rápido se ejecuta es la lectura de los codificadores rotatorios, esto es debido a que el conteo es realizado en hardware con el core implementado.

Análisis de resultados de los algoritmos aplicados

En las Figuras 4.1 y 4.2 se muestra la velocidad angular del motor medida en conteos del encoder cada 500 uS. Esta señal se obtiene al ejecutar el algoritmo de control de velocidad del motor.

En la figura 4.1 se puede observar la señal obtenida directamente desde el encoder y la señal una vez aplicado el filtrado. Se nota claramente que la señal filtrada, de color azul, sigue una tendencia suave, en comparación de la señal sin filtrar, esto se debe a que los datos provenientes del encoder son discretos por lo que tiene influencia el tiempo de muestreo, además que puede ser afectada por un torque externo, con el Filtro de Kalman lo que se logra es tener una estimación

analógica más cercana de la velocidad angular del motor, más allá de los datos obtenidos por los codificadores rotacionales, haciendo uso de estos.

En la figura 4.2 se muestra el pulso de entrada del sistema de control, que es de 80 conteos / 0.5 ms y la señal que representa la velocidad angular del motor una vez filtrada, se puede observar que el tiempo en el cual el sistema se estabiliza es de aproximadamente 25.6 ms.

En las figuras 4.3 y 4.4 se muestra cómo actúa el sistema para controlar la posición angular del velocista con respecto a la línea de la pista. En cuyas gráficas la velocidad del centro de masa del robot es cero, y lo que se requiere es que se alinee con la pista, ya que inicialmente su posición angular no es cero. Dichos datos los obtenemos tanto con el arreglo de sensores infrarrojos, como con el giroscopio.

En la figura 4.3 se observa la señal de la posición angular antes y después de aplicar el filtrado. La señal previa al filtrado corresponde a los datos obtenidos por el arreglo de sensores, cuyos datos son netamente discretos, con una resolución limitada por el número de sensores infrarrojos que posee el arreglo. Al aplicar filtro de Kalman, lo que se logra es que fusionar la señal que proviene del arreglo de sensores infrarrojos, con la señal proveniente del giroscopio,

obteniendo una mejor estimación analógica de dicha señal, ante las limitaciones causadas por los sensores.

En Figura 4.4 se muestra el control de la posición angular del robot con respecto a la pista, ante una señal de entrada, en las condiciones anteriormente mencionadas, usando la señal filtrada en el sistema de control, se obtiene una posición angular que se logra controlar de manera eficaz, llegando un estado estable en aproximadamente 0.1065 segundos.

Resultados de las pruebas del robot en la pista

Para las respectivas pruebas del diseño final del robot velocista sobre una pista, se procedió a elaborar una con cinta aislante negra sobre fondo blanco, la cual se muestra en la Figura 4.5, la que bien podría corresponder a una pista de competencia para esta clase de robots, en esta pista se distinguen cinco curvas que cambian de manera considerable el ángulo de su trayectoria, dicha pista cuenta con una longitud total de 5.84 metros. Para los resultados finales sobres esta pista, se tomó datos de los diversos sensores del robot, mientras este seguía la trayectoria, donde el inicio y fin está determinado por la posición en la que se encuentra el robot en la Figura 4.5, respetando el sentido mostrado.

Este proyecto busca el diseño de un robot velocista que sea capaz de mejorar la robótica de competencia de nuestro país en esta categoría, y sea un digno representante en competencias de carácter internacional. Por lo que los resultados vinculados a las velocidades alcanzadas son los más importantes a considerar. En la Figura 4.6, se puede observar las velocidades alcanzadas por el robot durante su recorrido en la pista de prueba. Se puede apreciar cinco gráficas las cuales están ordenadas de la siguiente manera: En la primera gráfica se muestra la velocidad de la rueda derecha durante el recorrido, en la segunda se puede observar la velocidad durante el recorrido de la rueda izquierda, ambos datos se obtienen de los codificadores rotacionales del motor que corresponde al lado respectivo, como tercera gráfica tenemos la velocidad con la que se desplaza el centro de masa del robot a lo largo de la trayectoria lo cual es el promedio de las dos mediciones anteriores, por último se muestra la diferencia de velocidades que existe entre las velocidades a ambos lados que, que es en lo que se basa la locomoción de este tipo de robot con configuración diferencial y es la medida de la velocidad angular del robot multiplicada por una constante de proporcionalidad, Ecuación 3. 4.

Debido a que el recorrido de la pista comienza en un tramo recto, se puede observar que hasta aproximadamente 0.8 segundos de haber comenzado el recorrido, las velocidades se mantienen valores

uniformes, sobrepasando los 2 m/s ambas ruedas, por ende el centro de masa se desplaza a la misma velocidad, y la diferencia entre las velocidades a los costados es prácticamente nula. A partir de los 0.8 segundos se aprecia un cambio en ambas velocidades, ya que se detecta la primera curva, ubicada en la esquina inferior izquierda de la Figura 4.5. Esta curva se encuentra hacia la derecha del recorrida, y es una de las más cerradas de este, una vez detectada esta curva a través de los sensores, debido al respectivo algoritmo de control el robot responde con una disminución considerable de la velocidad en la rueda derecha, por lo que ocurre un giro hacia ese lado, lo que también provoca un ligero incremento en la velocidad del lado izquierdo, lo que provoca que la velocidad del centro de masa disminuya, y la diferencia de velocidades aumente, dicha diferencia alcanza su máximo valor poco después de que transcurra 1 segundo, en este punto el desplazamiento del centro de masa del velocista llega a valores cercanos a cero, por lo cual se prioriza el giro ante el desplazamiento, esto ocurre a medida que el radio de curvatura de una parte de la trayectoria sea lo menor posible. Una vez que el robot gira y termina la curva, se puede apreciar como este vuelve a un comportamiento uniforme durante el siguiente tramo recto. Se puede apreciar un comportamiento similar durante las tres siguientes curvas que ejercen un cambio notable en la trayectoria y se pueden observar los puntos en

los que la diferencias de velocidades llegan a sus valores más altos en los segundos 1.7, 2 y 2.6 aproximadamente, con la diferencia de que la curva detectada a los 2 segundos cambia la trayectoria de la pista hacia la izquierda de la misma. La última de las curvas se aprecia en segundo 3.4 aproximadamente, a diferencia los cambios de las velocidades en las curvas anteriores, en esa se nota menos variación debido a que el radio de curvatura es mucho mayor al de las anteriores.

Entre los principales resultados obtenidos sobre esta pista tenemos que el tiempo total que se tomó el robot en recorrerla es de aproximadamente 3.5 segundos, por lo que la velocidad media del robot a lo largo de la trayectoria es de 1.67 m/s aproximadamente. Además los valores máximos y mínimos alcanzados por las ruedas para esta trayectoria ocurren en la tercera y cuarta curva notable del recorrido. La tercera de aquellas curvas corresponde a la única curva notable hacia el lado izquierdo, en la cual se alcanzan los valores límites aproximadamente a los 2 segundos del recorrido. Se aprecia que la rueda derecha alcanza su máximo valor y este es de aproximadamente 2.853 m/s y la rueda izquierda su valor mínimo que es de -1.933 m/s aproximadamente. En la cuarta curva notable del recorrido se obtuvieron los valores límites en una curva que gira hacia la derecha, aquí la velocidad de la rueda izquierda llegó hasta los 2.853 m/s, mientras que la rueda derecha llego a su mínimo valor en -2.638 m/s.

Finalmente considerando la velocidad en el centro de masa del robot velocista, tenemos que este alcanza su máxima velocidad en los tramos rectos, donde su valor es de 2.23 m/s aproximadamente y esta velocidad disminuye su valor en las curvas, y fue en la cuarta curva notable, ubicada en la esquina superior derecha de la Figura 2.5 que alcanza su mínimo valor que es de 0.088 m/s aproximadamente.

Hay que tomar en consideración que estos valores son exclusivos de la trayectoria que sigue esta pista y que pueden variar dependiendo del recorrido de la misma, y los límites se establecerán de acuerdo a la curva con el menor radio de curvatura posible, así como del ángulo del arco de circunferencia que esta tenga.

Bajo estos datos se aprecia que el robot velocista alcanza velocidades muy buenas para este tipo de competencias lo cual consigue que de manera implícita un aumento en el nivel de las competencias de robótica nacionales.

En la Figura 4.7 se puede observar la separación angular del robot con respecto a la pista, estos datos son obtenidos por el arreglo de sensores infrarrojos, durante la trayectoria en la pista de la Figura 4.5.

Debido a la cantidad de sensores utilizados y a la ponderación respectiva de cada sensor, los valores obtenidos se encuentran entre -28 y +28. Con respecto a las velocidades, dada que estas responden a

los cambios en la posición del robot con respecto a la pista, el comportamiento en las variaciones de estas lecturas debe estar ligado al comportamiento en las velocidades de los motores, ya que de eso se trata el control. Es así como se observa que los picos que se encuentran en zonas de gran variación coinciden en el tiempo en donde la velocidad del centro de masa disminuye, también es notable que después de cada curva estos valores no se mantienen lo más estables posibles en cero, esto depende a la respuesta del sistema, ya que necesita un tiempo para que ocurra esto, el cual se logra solo en tramos rectos lo suficientemente largos, y antes de que esto ocurra, en la pista de prueba ya se encuentra un nuevo tramo curvo, esto no significa problema alguno, ya que si se logra un posicionamiento lo suficientemente cerca.

Se alcanzan valores en las curvas con menor radio de curvatura de 28 y -15.35.

Además de los valores obtenidos por los sensores infrarrojos, en la Figura 4.8 se muestra la velocidad angular del centro de masa en el plano horizontal durante el recorrido por la pista, dado en rad/s.

Teniendo un comportamiento en las variaciones muy similar al que nos proporcionan los sensores infrarrojos, considerando el signo de este, los valores límites para la velocidad angular se presentan en las

mismas curvas, y esto será la tendencia para todas las gráficas, cuyo valores llegan a 9.734 rad/s y -13.5 rad/s.

En las Figuras 4.9, 4.10 y 4.11 se tienen gráficas obtenidas a través del magnetómetro, el cual es un elemento muy útil en el tema de la orientación, en las dos primera se obtienen los valores en Gauss del flujo magnético sobre el magnetómetro a lo largo de la trayectoria en los ejes X y Y respectivamente. Una vez obtenido estos datos se pueden usar para saber el ángulo de orientación con respecto al campo magnético, esto se logra aplicando la función arcotangente, como se explica en la sección 2.6.2, este ángulo se puede observar en la Figura 4.11, comenzando y terminando el recorrido en la misma orientación. Existen competencias de persecución en la cual se ubican dos velocistas en lugares opuestos de una pista simétrica, ganando el que logre alcanzar al otro, o el que se mantenga más tiempo en pista, para ello es útil identificar el recorrido de la pista, y aplicar estrategias una vez logrado esto puede marcar la diferencia, gracias al uso del magnetómetro en conjunto con las velocidades del robot durante su trayectoria es posible identificar la forma y dimensiones de una pista, lo cual se puede apreciar en la Figura 4.12, podemos comparar dicha gráfica con aquella de la Figura 4.5 y darnos cuenta la similitud y precisión con lo que se logra esto.

Finalmente la IMU provee valores de la aceleración en tres dimensiones. Las gráficas de aceleración en una pista plana nos proporciona información poco útil, debido a que ya se provee con otros tipos de sensores que nos ofrecen mejores datos sobre el posicionamiento general. Pero es de mucha utilidad cuando la pista presenta pendientes, dado que las aceleraciones en los ejes X y Y velocidades variaran de manera significativa, siendo posible encontrar las pendientes con las que se encuentra el robot. La Figura 4.13 muestra la medida de la aceleración del robot en los ejes X y Y cuando el robot se inclina en una pendiente que se alcanza a los 2 segundos aproximadamente. El acelerómetro del eje y mide un valor cercano a -10 m/s^2 que corresponde a la gravedad y en el eje Y el valor es 0. Cuando el robot enfrenta la pendiente la medición del eje Y aumenta y la del eje X disminuye.

Con estos resultados se puede obtener una gráfica estimada del ángulo de inclinación de la pendiente, como se muestra en la figura 4.14.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. El uso de la FPGA permite diseñar un microcontrolador a la medida, integrar en un solo chip los módulos necesarios para el control del robot sin realizar alguna modificación física. Entre las tareas que se realizó gracias al uso de la FPGA, destacan el aumentar la frecuencia de trabajo del sistema añadiendo un módulo PLL digital, leer las señales provenientes de lo codificadores rotatorios sin usar interrupciones y se pudo proteger la FPGA de corrientes entrantes al llevar los integrados puente H al modo de espera. Esto último no se hubiese podido hacer por software en un microcontrolador común, en ese caso se debería

buscar otras soluciones como agregar opto-acopladores o utilizar módulos puente H que incluyan una protección en Hardware.

2. Usar las herramientas NIOS II y Qsys de Quartus II permite realizar un co-diseño hardware-software, estar en una línea intermedia entre un diseño en lenguaje HDL y uno en lenguaje secuencial en un procesador, de tal manera que se puede aprovechar el paralelismo del primero y la facilidad de implementación del segundo.
3. El uso del Filtro de Kalman en las señales leídas desde los codificadores rotatorios permitió mejorar el control de velocidad de los motores, ya que sin el filtrado, la señal que iría al controlador sería digital y con ruido debido a que los motores y codificadores usados son reutilizados y no estaban en un estado óptimo.
4. El uso del Filtro de Kalman permitió combinar la información obtenida del arreglo de sensores infrarrojos y del giroscopio para realizar una estimación de la separación angular. La señal obtenida es robusta respecto al ruido ambiental y al no ser digital puede ser muestreada a cualquier rapidez. El resultado obtenido es mejor que usar un único sensor para realizar la medición.
5. El controlador diseñado tiene una buena respuesta en pista, un avance respecto a otros diseños más comunes es que la realimentación en cascada permite estabilizar el robot usando datos de dos mediciones, de la velocidad de los motores y de la posición angular del robot.

6. La adquisición de datos permite analizar en detalle el comportamiento del robot, ya sea respecto a los algoritmos de control, filtrado o estrategias implementadas para la competencia. Estos detalles no se pueden notar si se trata de analizar a simple vista la conducta del robot.
7. Matlab permite trabajar en mejoras del código haciendo simulaciones con los datos adquiridos. Es decir se puede implementar el algoritmo en un script de Matlab y usar como entrada los datos tomados, luego observar el resultado en gráficas de salida y una vez que se tiene el desempeño deseado se implementa el diseño final en el procesador del robot. En el caso del algoritmo de control se necesita la respuesta real de los motores por lo que no se puede trabajar sólo con simulaciones, pero para el filtrado las simulaciones simplificaron mucho el trabajo de diseño.
8. La elección de las ruedas es un punto importante en la construcción física del robot, se debe escoger las ruedas del material que proporcione mayor tracción, además el radio y el ancho de la rueda influyen también en el desempeño del robot. Con respecto al radio se tiene que un mayor radio aumenta la velocidad en estado estable del robot pero también aumenta la carga equivalente sobre los motores, por lo que la aceleración del robot disminuye. El efecto contrario se logra disminuyendo el radio. Respecto al ancho de la rueda se puede mencionar que existe un valor óptimo. Un análisis sobre esto último

está más allá de lo que este proyecto abarca pero se puede experimentar con el tipo de rueda que brinde el mejor desempeño.

9. El uso de un dispositivo de transmisión inalámbrica, permite enviar y recibir información mientras el robot está funcionando, esto no se puede hacer si se transmite usando un cable. Hay varias opciones de dispositivos inalámbricos, pero la ventaja que brindan los dispositivos Bluetooth es que permiten además de la conexión al PC conectarse fácilmente a otros sistemas como smartphones o cualquier dispositivo inteligente. Por ejemplo, se puede conectar un teléfono al robot y utilizarlo como pantalla para visualizar si todos los sensores están trabajando o como una interfaz para enviar información de configuración al robot para hacer pruebas previo a las competencias.
10. La tarjeta electrónica y en general el diseño de hardware del robot se debe hacer enfocándose en disminuir la masa del robot, mantener el centro de gravedad bajo y disminuir el momento de inercia alrededor del centro de curvatura. En este aspecto usar la tarjeta DE0-nano fue una desventaja ya que ocupa demasiado espacio, el desempeño del robot puede mejorar si se diseña la tarjeta electrónica con el chip de la FPGA sin hacer uso de la DE-0 Nano.
11. El uso sensores adicionales a los infrarrojos permitió complementar la información proporcionada por estos. Los sistemas de control y filtrado usan tres tipos de sensores: infrarrojos, codificadores rotatorios y

giroscopio. El acelerómetro y el magnetómetro sirven principalmente para hacer análisis del desempeño del robot, observar la aceleración del centro de masa y la trayectoria que describe el robot. Además son útiles en competencias en la que el robot tenga que superar pruebas como subir o bajar pendientes o esquivar obstáculos.

RECOMENDACIONES

1. Es un error común en el diseño de robots velocistas elegir motores que tengan una velocidad final muy elevada, pero no se considera el valor del torque máximo y de la aceleración, estas cantidades son importantes para el arranque, para los giros y, en ciertas competencias, para el frenado del robot. Lo aconsejable es fijar una velocidad máxima realista, un valor entre 1.5 m/s y 2 m/s es un muy buen comienzo. Una vez fijada la velocidad máxima se debe elegir la caja de reducción y el radio de la rueda que genere esa velocidad máxima. Se debe tener en cuenta que una rueda de radio pequeño mantiene bajo el centro de masa y disminuye la carga equivalente del robot, aunque también una rueda de radio pequeño se empolvará más rápido y también el efecto de resistencia a la rodadura es mayor que en una de más radio [26].

2. Previo a un concurso es común realizar varias pruebas y mediciones para preparar el robot para competir, en esta situación se necesita una batería que no se descargue tan rápido sino que permita realizar las pruebas con comodidad. Pero una carrera, según el tipo de competencia, puede dura entre unos 10s y 5min, no es necesario tener una batería con demasiada capacidad, ya que estas son grandes, ocupan espacio y aumentan la masa del robot. Se recomienda tener baterías para la competencia y para hacer pruebas, en este robot se usó de 1000mAh y constante de descarga entre 20-30C para las pruebas y de 300mAh y constante de descarga 35-70C para la competencia.
3. La tarjeta DE0-Nano provee varios pines con un voltaje de 3.3 V, estos pines están conectados a un regulador de voltaje que además alimenta varios componentes de la tarjeta, para pequeñas aplicaciones se puede usar estos pines de alimentación, pero en proyectos más complejos es preferible tener un regulador dedicado a la alimentación de los elementos adicionales y dejar el de la tarjeta sólo para sus componentes, ya que no se tienen información de cuánta corriente consume la tarjeta y más los elementos adicionales del circuito se puede exigir demasiado al regulador y ocasionar daños.
4. Altera provee una aplicación que es muy útil para trabajar con la DE0-Nano, esta se llama DE0-Nano Control Panel, con este programa se

puede probar el funcionamiento de algunos elementos de la tarjeta , leds, botones, interruptores, convertidor analógico digital, acelerómetro y las memorias que posee la tarjeta: SDRAM, EEPROM y EPCS. Además si se desea borrar las memorias no volátiles con esta aplicación se lo puede hacer fácilmente.

5. Es común en la robótica de competencia usar controladores que se ajustan de forma manual, en este proyecto el controlador de posición angular es un controlador tipo PD que se ajustó manualmente. Para este tipo de controladores es de gran utilidad escribir una rutina en el programa que permita ajustar constantes de forma inalámbrica, en este proyecto las constantes proporcional y derivativa del controlador y la velocidad del robot se pueden ajustar usando un dispositivo con comunicación Bluetooth como un smartphone y enviar los valores de forma serial.
6. El magnetómetro es sensible a cualquier campo magnético no sólo el terrestre, por lo que, si va a ser usado en una carrera, para evitar fallos es preferible re-calibrarlo en el lugar de la competencia.

BIBLIOGRAFÍA

[1] Dieck Assad Graciano, *Instrumentación, acondicionamiento electrónico y adquisición de datos,* Trillas, 2000.

[2] Hamacher Carl, *Organización de computadores (quinta edición)*, Mc Graw Hill, 2003.

[3] Terasic Technologies Inc, *DE0-Nano User Manual,* http://www.altera.com/literature/ug/DE0_Nano_User_Manual_v1.9.pdf USA, 2012.

[4] Rodríguez Araujo Jorge, *Estudio del microprocesador Nios II*, https://es.scribd.com/doc/28358833/Estudio-del-microprocesador-Nios-II 2010.

[5] ALTERA, *Nios II Processor Reference Handbook*, http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf, 2014.

[6] Universidad de Málaga, *Introducción a los sistemas de control discretos*, http://www.isa.uma.es/C14/Presentaciones%20de%20Clase%20(ppt)/Document%20Library/SISTEMAS%20DE%20CONTROL%20DISCRETOS.pdf, fecha de consulta enero 2015.

[7] Mongi A. Abidi, Rafael C. González, *Data fusión in robotics and machine intelligence*, Academic Press, 1992.

[8] Schmid Christian, Ruhr Universitat Bochum, *Introduction into System Control*, http://www.atp.ruhr-uni-bochum.de/rt1/syscontrol/node1.html, 2005.

[9] Smith Steven W, *The Scientist and Engineer´s Guide to Digital Signal Processing*, California Technical Publisher, 1997.

[10] STMicroelectronics, *L3GD20H: MEMS motion sensor: three-axis digital output gyroscope*, https://www.pololu.com/file/0J731/L3GD20H.pdf, 2013.

[11] STMicroelectronics, LSM303D: *Ultra compact high performance e-Compass 3D accelerometer and 3D magnetometer module*, https://www.pololu.com/file/0J703/LSM303D.pdf, 2012.

[12] Guangzhou HC Information Technology, *HC-06 datasheet*, http://silabs.org.ua/bc4/hc06.pdf, 2011.

[13] Dudek Gregory, Jenkin , *Computational Principles of Mobile Robotics*. Cambridge University Press, 2010.

[14] Davies Matthew, Schmitz Tony L, *System Dynamics for* Mechanical Engineers, Springer, 2014.

[15] Fischer Michael, *Altera DE0-Nano*, http://www.emb4fun.de/fpga/nutos1/, 2012.

[16] Altera, *RS232 UART for Altera DE-Series Boards,* ftp://ftp.altera.com/up/pub/Altera_Material/11.1/University_Program_IP_Cores/Communications/RS232.pdf, 2011.

[17] Altera, *NIOS II Hardware Development Tutorial*, http://www.altera.com/literature/tt/tt_nios2_hardware_tutorial.pdf, 2011.

[18] Altera, *Basic Computer System for the Altera De0-Nano*, ftp://ftp.altera.com/up/pub/Altera_Material/12.0/Computer_Systems/DE0-Nano/DE0-Nano_Basic_Computer.pdf, 2010.

[19] Altera, *Embedded Peripheral IP User Guide,* http://www.altera.com/literature/ug/ug_embedded_ip.pdf, 2014

[20] Altera, *Parallel Port for Altera DE-Series Boards,* ftp://ftp.altera.com/up/pub/Altera_Material/9.1/University_Program_IP_Cores/Input_Output/Parallel_Port.pdf*,* 2010.

[21] Altera, *DE0-Nano ADC Controller,* ftp://ftp.altera.com/up/pub/Altera_Material/12.1/University_Program_IP_Cores/Input_Output/DE0-Nano_ADC_Controller.pdf*,* 2012.

[22] OpenCores*. I2C controller core: Overview*, http://opencores.org/project,i2c, 2014.

[23] Altera, Making Qsys Components, ftp://ftp.altera.com/up/pub/Altera_Material/12.0/Tutorials/making_Qsys_components.pdf, 2012

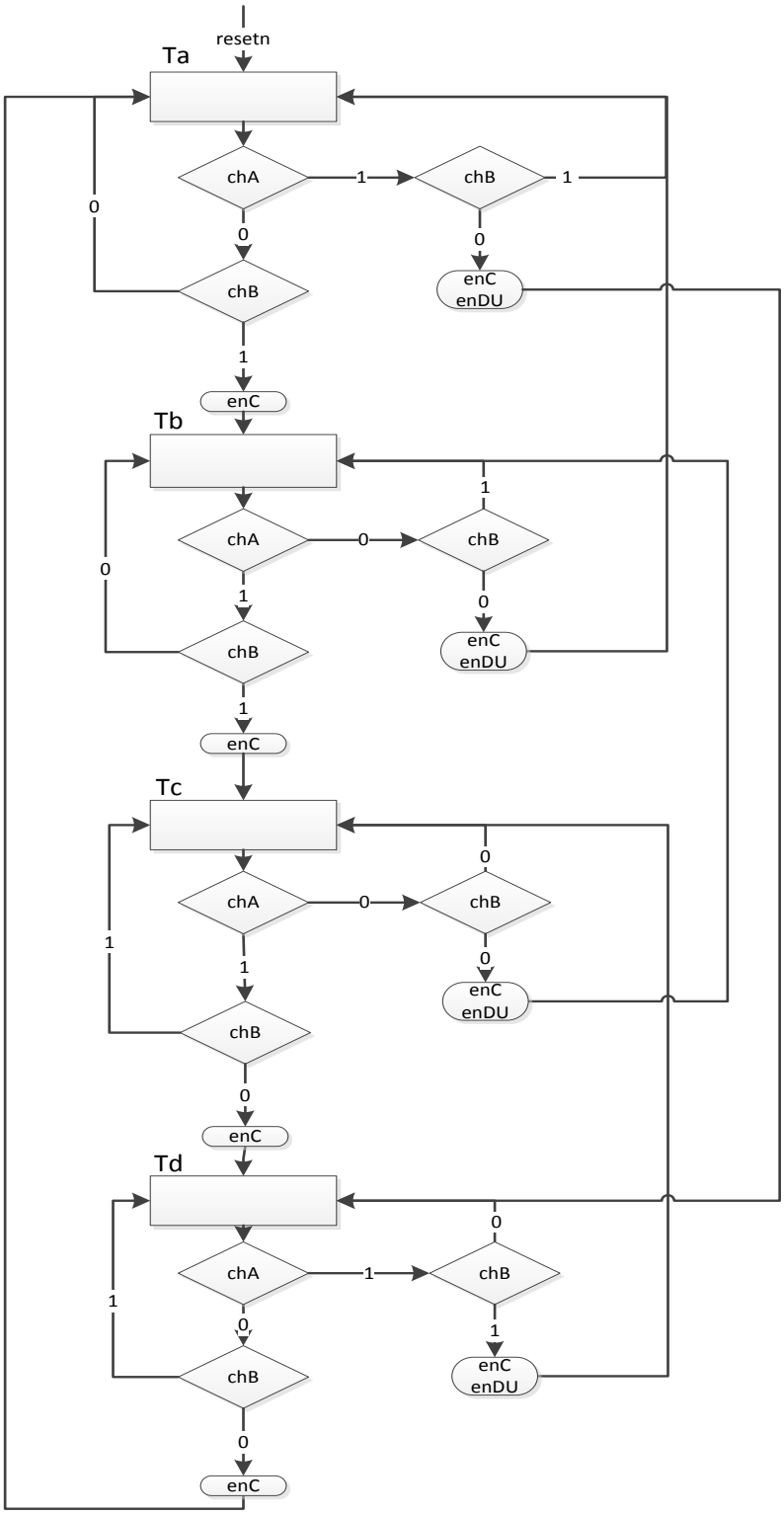
[24] Laboratorio de Control Automático ESPOL, *Identificación de sistemas usando MATLAB a través del cfp-2100*, Ecuador, 2013.

[25] FAULHABER, *DC-Micromotors Precious Metal Commutation Series 2224…SR,https://fmcc.faulhaber.com/resources/img/EN_2224_SR_DFF.PDF*, 2015.

[26] Altera, *Avalon Interface,* http://www.altera.com/literature/manual/mnl_avalon_spec.pdf, 2015.

ANEXOS

A. Diagrama ASM del Controlador del Módulo Encoder



B. Diagramas de Tiempo

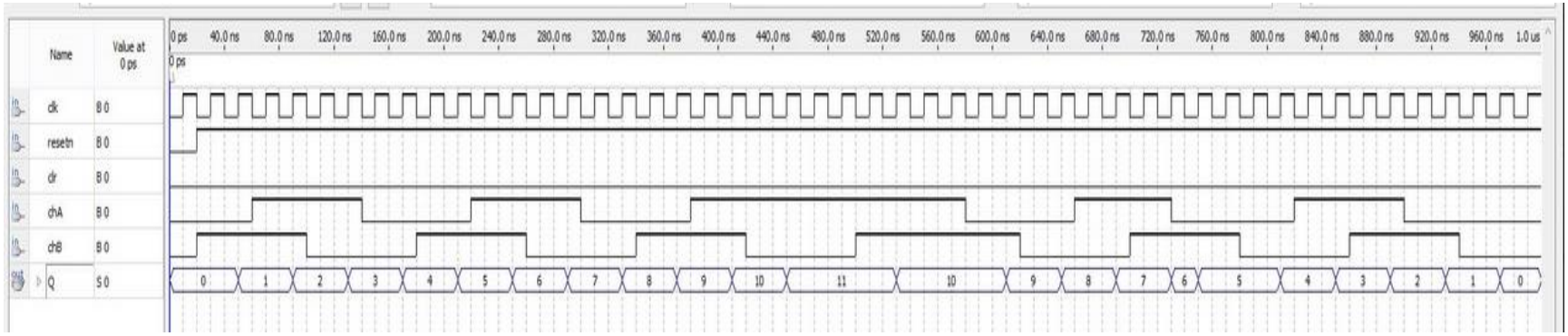


Diagrama de tiempo del funcionamiento del módulo Encoder

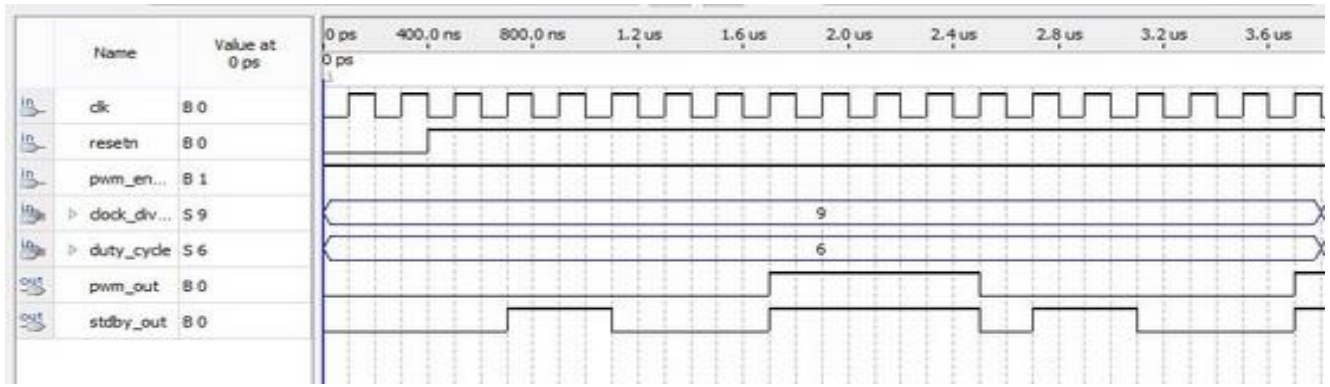
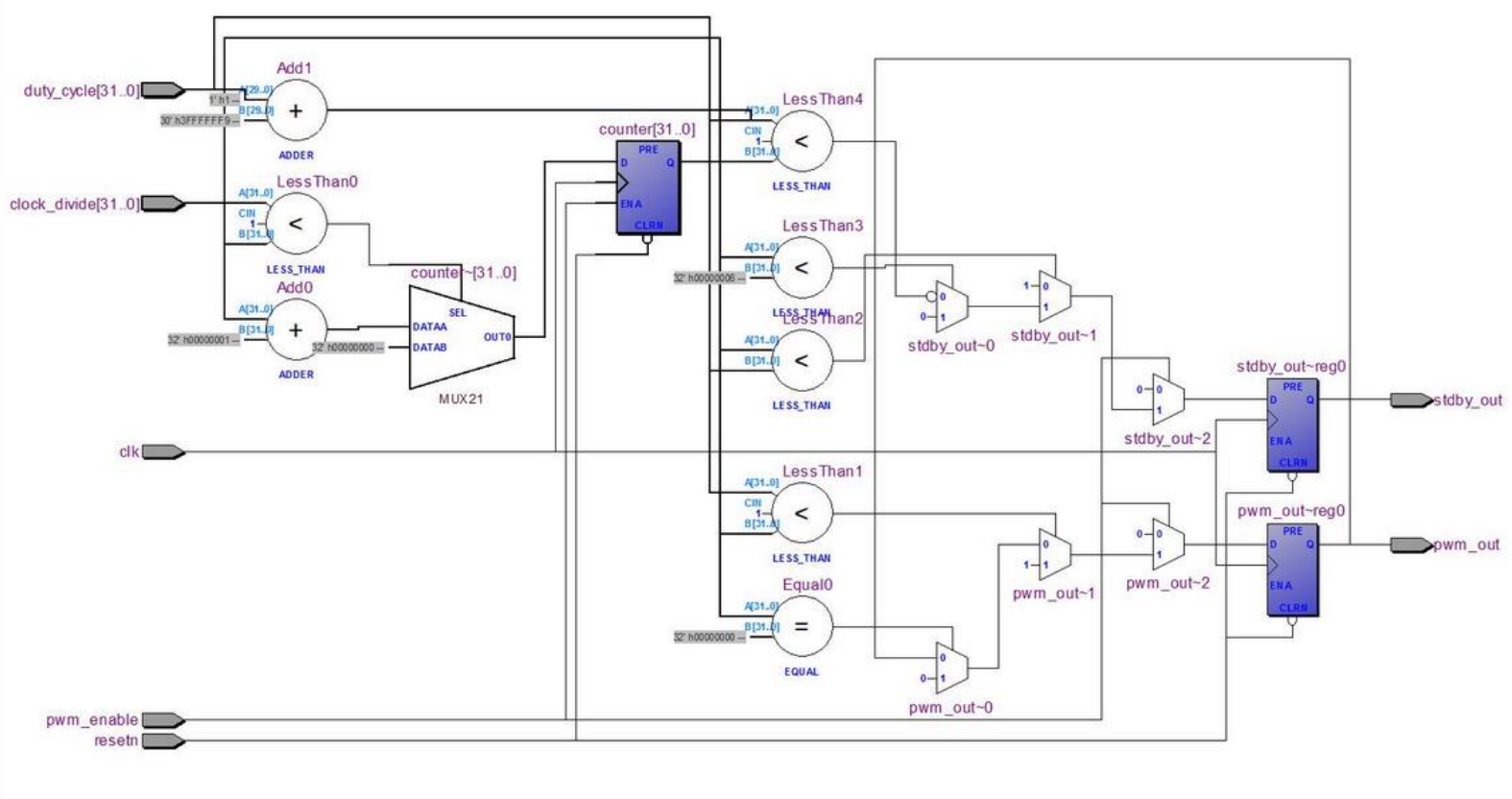


Diagrama de tiempo del módulo PWM modificado

C. Partición Funcional del Bloque Task_logic modificado



D. Calibración del Magnetómetro.

Existen diferentes modelos para realizar una calibración. Aquí se presenta un modelo sencillo, lo que se hará es modificar las mediciones multiplicándolas por un factor de escala y desplazándolas una cierta cantidad.

Dado que el campo magnético medido en el plano paralelo a la superficie debe ser constante se tiene que:

$$(kx + l)^2 + (my + n)^2 = r^2$$

x es la medición del campo magnético en el eje X.
y es la medición del campo magnético en el eje Y.
r es la magnitud constante del campo magnético total.
k es el factor de escala constante para las mediciones del eje X.
m es el factor de escala constante para las mediciones del eje Y.
l es el valor constante de desplazamiento para las mediciones del eje X.
n es el valor constante de desplazamiento para las mediciones del eje Y.

Esta expresión se puede escribir de la forma:

Ecuación D. 1

$$ax^2 + bx + cy^2 + dy + e = 0$$
$$a = k^2$$
$$b = 2kl$$
$$c = m^2$$
$$d = 2mn$$
$$e = l^2 + n^2 - r^2$$

La razón para expresar la ecuación de esta forma es que MATLAB permite encontrar los valores a,b,c,d,e que hacen que los vectores de medición se ajusten lo mejor posible a un cierto modelo. El modelo

que se usa para hacer el ajuste es el de un polinomio cuadrático de dos variables, en MATLAB se lo nombra como poly22:

$$poly22: p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 = z$$

X corresponde a la medición en el eje x, Y a la medición en el eje Y y Z es un vector de ceros, además $p_{11} = 0$. Para realizar el ajuste se utiliza la función fit() de MATLAB, la descripción de cada parámetro de entrada de la función se puede encontrar en la sección de ayuda o en la web de MATLAB.

Una vez realizado el ajuste se debe encontrar los valor k,l,m,n. Se puede igualar la Ecuación D. 1 con el polinomio cuadrático de dos variables, se tiene que:

$$k = \sqrt{p_{20}}$$
$$m = \sqrt{p_{02}}$$
$$l = \frac{p_{10}}{2p_{20}}$$
$$n = \frac{p_{01}}{2p_{02}}$$
$$r = \sqrt{l^2 + n^2 - p_{00}}$$

```
% (kx+l)^2+(my+n)^2=r^2
% ax^2+bx+cy^2+dy+e=0
% a=k^2
% b=2kl
% c=m^2
% d=2mn
% e=l^2+n^2-r^2
% poly22: p00 + p10*x + p01*y + p20*x^2 + p11*x*y + p02*y^2

% ajuste
tamM=4000;
mX=Datos(1:tamM,1);
mY=Datos(1:tamM,2);
ZerostamM=zeros(tamM,1);
sf = fit([mX(1000:tamM), mY(1000:tamM)],ZerostamM(1000:tamM),'poly22','Lower',[-Inf,-Inf,-Inf,1,0,1],'Upper',[Inf,Inf,Inf,Inf,0,Inf

k=sqrt(sf.p20);
m=sqrt(sf.p02);
l=sf.p10/(2*sf.p20);
n=sf.p01/(2*sf.p02);
r=sqrt(l^2+n^2-sf.p00);

% circulo
angtmp=0:0.01:2*pi;
xC=r*cos(angtmp);
yC=r*sin(angtmp);

% Grafico
hold on
plot(k*mX+l,m*mY+n,mX,mY,xC,yC)
axis equal
xlabel('Campo Magnético Eje X')
ylabel('Campo Magnético Eje Y')
hold off
legend( 'Datos Calibrados','Datos sin Calibrar','Circunferencia r=1410.9 C=(0,0)');
```

Figura D.1 Código de MATLAB para calibración del Magnetómetro

Las ecuaciones son implementadas en Matlab, esto se presenta en la Figura D.1

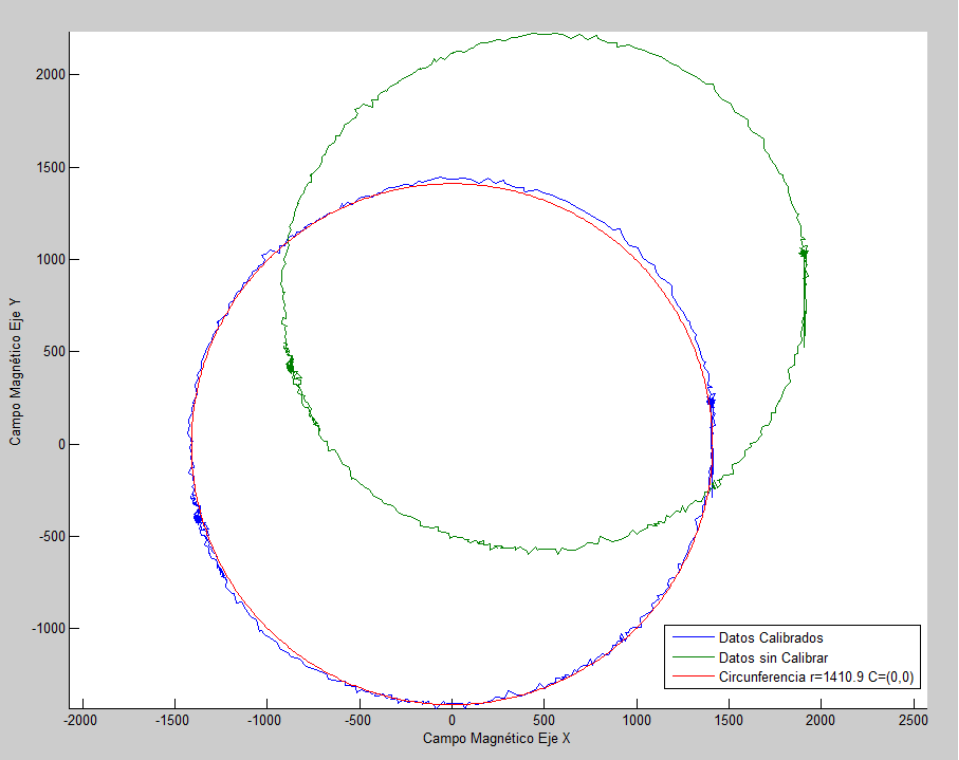


Figura D.2 Magnitud Constante del Campo magnético.

Además se grafica la magnitud del campo magnético en el eje X versus el campo en el eje Y, con los datos calibrados y no calibrados y un circulo centrado en el origen con el radio obtenido. Esto se muestra en la Figura D.2.

E. Manejo de Unidades de Medida

Se puede encontrar la relación entre las mediciones de sensores diferentes de forma teórica consultando la hoja de datos de cada sensor y multiplicar por el factor de escala necesario o en ciertos casos haciendo los cálculos considerando la geometría del robot e inclusive puede influir el tiempo de muestreo. Esta forma puede ser un poco tediosa y no es práctica.

En este proyecto lo que se hizo fue tomar datos con los sensores y vectores de mediciones, uno para cada sensor. Luego para encontrar las constantes que relacionan estos vectores se realiza una regresión lineal usando la técnica de mínimos cuadrados.

El modelo para una regresión lineal es:

$y = X\beta + \varepsilon$

Donde

- y es el vector de respuestas de tamaño nX1.
- β es el vector de coeficientes de tamaño mX1.
- X es la matriz de diseño del modelo de tamaño nxm.
- ε es el vector de errores de tamaño nx1.

La idea es encontrar el vector β que minimice el error ε. El vector que minimiza este error está dado por:

$$b = (X^T X)^{-1} X^T Y$$

Para este caso solo se necesita encontrar dos valores constantes $\beta_1 \, y \, \beta_2$ que minimicen el error entre los vectores de medición de dos diferentes sensores. Se tienen entonces que:

$$X = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}$$

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

Donde

El vector $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ contiene las “n” mediciones de un sensor.

El vector Y contiene las “n” mediciones de otro sensor.

El vector β contiene a las constantesβ_1y β_2 que minimizan el error entre ambas mediciones.

```
 3
 4 -    tamD=4000;
 5 -    Xmin2=zeros(tamD,2);
 6 -    Xmin2(:,2)=zeros(tamD,1)+1;
 7 -    Xmin2(:,1)=aGyr;
 8 -    Din=IR;
 9 -
10 -    res=(Xmin2'*Xmin2)^-1*Xmin2'*Din;
11
```

Figura E. 1 Código de Matlab para Realizar Regresión Lineal.

Estas ecuaciones se implementan de forma sencilla en Matlab, como se muestra en Figura E. 1. El script permite encontrar el factor de escala y la constante de desplazamiento que relaciona mediciones de sensores diferentes.

En el ejemplo se usa vectores con 4000 mediciones, IR es el vector que contiene datos de los sensores infrarrojos y aGyr el vector que contiene el ángulo medido usando el giroscopio.

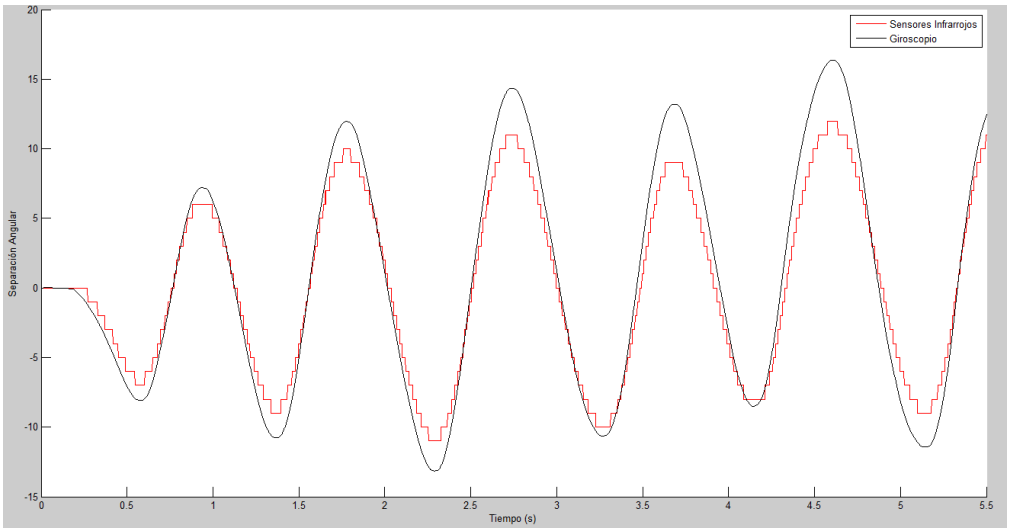


Figura E. 2 Señales Medidas por el Giroscopio y por los Sensores infrarrojos.

En la Figura E. 2 se muestra las mediciones hechas con los sensores infrarrojos y con el giroscopio, se observa que ambos miden la misma señal pero están en diferentes escalas.

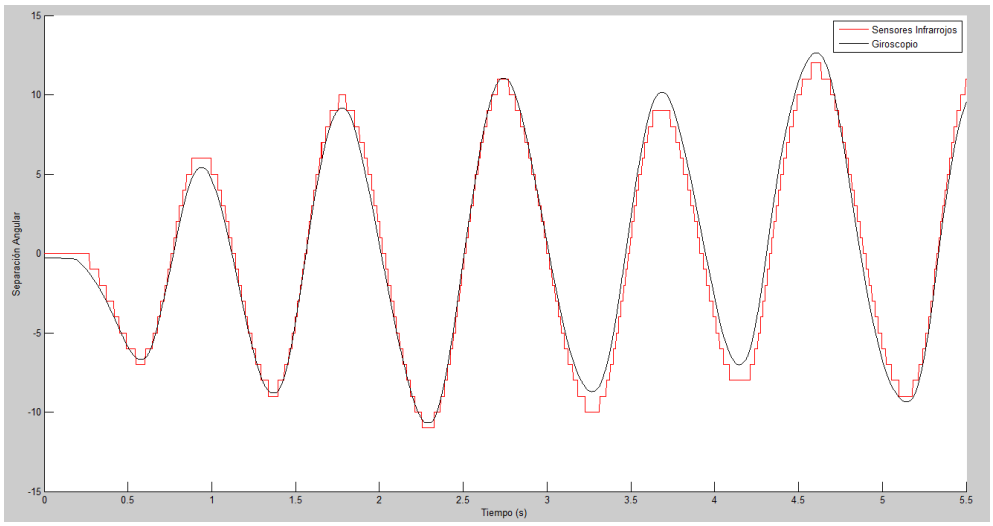


Figura E. 3 Señales medidas por el Magnetómetro y Sensores Infrarrojos con la Regresión Lineal.

En la Figura E. 3 luego de ejecutar el algoritmo de mínimos cuadrados se grafica la señal ya con las constantes añadidas y se ve que la diferencia entre ambas mediciones ahora es menor.