

IMPLEMENTACIÓN DE UN ROBOT ESFÉRICO BASADO EN FPGA`s

BRAULIO GONZÁLEZ PALACIOS

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA
CALI
2019

IMPLEMENTACIÓN DE UN ROBOT ESFÉRICO BASADO EN FPGA`s

BRAULIO GONZÁLEZ PALACIOS

Código 9834162

Trabajo de grado

Directores:

Mario Enrique Vera Lizcano, MSc

José Tomás Buitrago Molina, MSc

UNIVERSIDAD DEL VALLE

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA

CALI

2019

DEDICATORIA

Este trabajo está dedicado a mi madre, hermanos, esposa e hijos, por haber estado en todos los momentos de esta gran meta apoyándome y dándome fuerzas para seguir adelante, a pesar de todos los obstáculos siempre creyeron en mí, se hizo todo un esfuerzo por estar ahí, perseverar hasta culminar esta etapa tan importante para mi vida, la cual me ha hecho crecer enormemente como persona, y ser cada día un mejor ser humano.

También va dedicado a todas aquellas personas que de una u otra forma están ahí alentándome cada día más a seguir adelante y cumplir a cabalidad cada una de mis metas.

AGRADECIMIENTO

Primeramente, mis agradecimientos son para Dios por ser mi guía, por darme la fuerza y persistencia para llegar a este punto.

Agradezco a la Universidad del Valle en especial a la Facultad de Ingeniería, a la Escuela de Ingeniería Eléctrica y Electrónica, por darme esta oportunidad de culminar mi carrera y así crecer cada día más.

A la parte directiva de la Escuela de Ing. Electrónica, por ser una guía y un apoyo ante cualquier dificultad durante todo este camino.

A mis directores de tesis Mario Enrique Vera Lizcano, MSc y José Tomás Buitrago Molina, MSc por derramar su confianza en mí y hacerme participe de todos sus conocimientos.

RESUMEN

El presente trabajo describe la funcionalidad de un robot que basa su movimiento en un cuerpo esférico, sus movimientos se deben a que los motores hacen parte de una estructura interna con sus respectivos actuadores en los planos X, Y y Z.

Su comando se basa en un sistema de desarrollo basado en FPGA's de Altera DEO_NANO, un *driver* para sus servomotores y mando a distancia desde un celular por medio de *Bluetooth*, se utiliza VHDL para la descripción del sistema hardware en la FPGA.

Este proyecto tiene como objeto implementar un robot en forma de esfera, electrónico, portátil, con bajo consumo de energía, y mando remoto para su operación. En el mismo se busca acoplar el sistema electrónico hardware y software, que requiere el robot esférico, para lograr el movimiento en cualquier dirección.

Para el desarrollo de este proyecto se presenta un diseño de Ingeniería que se desarrolló para llevar a cabo un prototipo de robótica móvil, el cual consta de tres partes principales como son: la interfaz de usuario, la unidad de control y los actuadores. Además, encontramos todos los componentes que implican la construcción del robot.

Se presentan cálculos realizados durante la elaboración de dicho proyecto, así como las simulaciones y sus respectivos resultados.

Para finalizar, se describen los resultados obtenidos sobre la funcionalidad del proyecto, y gracias a esto se pudo saber si es o no viable para lo que fue creado. Además, se puede decir que el proyecto realizado ha contribuido de manera muy significativa para lograr una construcción exitosa del robot esférico, esperando que sea de gran aporte en el ámbito de la robótica móvil y un preámbulo para futuros proyectos.

ABSTRACT

The present work describes the functionality of a robot that bases its movement on a spherical body, its movements are due to the fact that the motors are part of an internal structure with their respective actuators in the X, Y and Z planes.

Its command is based on a development system based on FPGA's of Altera DE0_NANO, a driver for its servomotors and remote control from a cell phone via Bluetooth, VHDL is used for the description of the hardware system in the FPGA.

This project aims to implement a robot in the form of a sphere, electronic, portable, with low power consumption, and remote control for its operation. It seeks to couple the electronic hardware and software system, which requires the spherical robot, to achieve movement in any direction.

For the development of this project an engineering design is presented that was developed to carry out a prototype of mobile robotics, which consists of three main parts such as: the user interface, the control unit and the actuators. In addition, we find all the components that involve the construction of the robot.

Calculations made during the preparation of said project are presented, as well as the simulations and their respective results.

Finally, the results obtained on the functionality of the project are described, and thanks to this it was possible to know whether or not it is viable for what was created. In addition, it can be said that the project has contributed in a very significant way to achieve a successful construction of the spherical robot, hoping it will be of great contribution in the field of mobile robotics and a preamble for future projects.

TABLA DE CONTENIDO

Pág.

CAPÍTULO 1. ASPECTOS GENERALES.....	1
1.1. INTRODUCCIÓN.....	1
1.2. ANTECEDENTES.....	1
1.3. DEFINICIÓN DEL PROBLEMA	3
1.4. JUSTIFICACIÓN	3
1.5. OBJETIVOS.....	3
1.5.1. OBJETIVO GENERAL:.....	3
1.5.2. OBJETIVOS ESPECÍFICOS:.....	3
1.6. MARCO REFERENCIAL	4
1.6.1. GENERALIDADES	4
1.6.1.1. ROBOTS ESFERAS	4
1.6.1.2. MODULACIÓN POR ANCHO DE PULSO (PWM)	5
1.6.1.3. ACTUADORES	5
1.6.1.4. MANDO A DISTANCIA.	5
CAPÍTULO 2. COMPONENTES.....	6
2.1. DESARROLLO DEL SISTEMA.....	6
Figura. 1. Diagrama de bloques del proyecto.	6
2.2. SISTEMA PARA EL MANEJO SERVOMOTOR	7
Figura. 2. Tarjeta de desarrollo para el control de servomotores.	7
Figura. 3. Conexión driver, servomotor y tarjeta DE0_NANO.	8
2.3. SERVOMOTOR.....	8
Figura. 4. Dimensiones servomotor MG996R.....	9
Tabla Nº 1. Configuración del conector.	9
2.4. TARJETA DE DESARROLLO DE0-NANO	9
Figura. 5. Vista superior, tarjeta de desarrollo DE0_NANO.....	10
Figura. 6. Vista inferior, tarjeta de desarrollo DE0_NANO.	10
2.5. ADXL345 ACELERÓMETRO	10
2.5.1. CÓMO FUNCIONA EL ACELERÓMETRO.....	11

Figura. 7. Respuesta de salida en función de la orientación y gravedad.....	11
2.6. CYCLONE IV	11
2.7. LENGUAJE VHDL.....	12
2.8. LA ESTRUCTURA INTERNA DE LA ESFERA Y SU FUNCIONAMIENTO	12
Figura. 8. Diseño estructura interna robot.	12
2.8.1. LA ESFERA.....	13
Figura. 9. Fotografía original domos esféricos.....	13
2.9. MÓDULO BLUETOOTH HC-05	13
Figura. 10. Modulo Bluetooth HC-05.	14
2.10. INTERFAZ DE USUARIO.....	14
Figura. 11. Diseño APP interfaz de usuario.	14
Figura. 12a. Diagrama de bloques APP interfaz de usuario.....	15
Figura. 12b. Diagrama de bloques APP interfaz de usuario.....	15
Tabla Nº 2. Parámetros App.	16
2.11. ALIMENTACIÓN	16
Figura. 13. Batería de 3.7V.....	16
2.12. FRECUENCIA DE OPERACIÓN SERVOMOTORES.....	17
Figura. 14. RTL Módulo divisor de frecuencia.....	17
Figura. 15. Módulo divisor de frecuencia.	18
2.13. PWM	19
2.14. MÓDULO UART_RS232	21
Figura. 18. Salida serial del módulo Bluetooth.	21
Figura. 19. Distribución de pines puerto GPIO-1 DE0_NANO.....	22
Figura. 20. Signal tap módulo UART entrada serial, salida paralela.	22
Figura. 21. RTL módulo UART.	23
Figura. 22. Foto real sin enviar datos.....	24
2.15. DISEÑO ESTRUCTURAL PARA EL MANEJO DE LOS DATOS DEL ACELERÓMETRO	24
Figura. 24. RTL diseño estructural.....	25
Figura. 25. RTL diseño estructural manejo de datos del acelerómetro.....	25
Figura. 27. RTL módulo comparador.....	28

Figura. 28. Signal tap diseño estructural del manejo de datos del acelerómetro.....	29
2.16. MÓDULO COMPARADOR FINAL.....	29
Figura. 29. RTL comparador final.	29
Figura. 30. RTL diseño estructural.....	31
CAPÍTULO 3. RESULTADOS OBTENIDOS.....	32
Figura. 31. Diseño estructural sistema completo.	32
Figura. 32. Signal Tap del sistema completo.....	34
Figura. 33. Cambio en engranaje servomotor Mg996r.....	34
Figura. 34. Diseño estructura interna de hardware.....	35
Figura. 35. Diseño hardware.	35
Figura. 36. Diseño hardware finalizado.	36
Tabla Nº 3. Resultados obtenidos.....	36
CAPÍTULO 4. CONCLUSIONES.....	37
BIBLIOGRAFÍA.....	38

CAPÍTULO 1. ASPECTOS GENERALES

1.1. INTRODUCCIÓN

La robótica comprende el diseño, fabricación y aplicaciones de los robots. Combina varias disciplinas, como son: la mecánica, la electrónica, la informática, la inteligencia artificial y la ingeniería de control.

Como es de conocimiento general, la ciencia y la tecnología crecen de manera vertiginosa y por ende, los robots se encuentran por todas partes, por ejemplo, máquinas dispensadoras, líneas de producción, fuerzas armadas, entre otras.

Se propone diseñar e implementar un robot en forma de esfera con tracción omnidireccional, tamaño reducido y de bajo consumo de energía, con capacidad de recibir señales de forma inalámbrica para su operación.

En este proyecto se caracterizará el movimiento con sus respectivos actuadores y se construirá una plataforma de robot esférica que utilice dichas determinaciones para realizar el movimiento.

El sistema tendrá un diámetro aproximado de 30cm, cuyo movimiento será realizado por los actuadores, los cuales estarán ubicados en un sistema de ejes cartesianos, donde cada motor/masa corresponde respectivamente a los ejes coordenados X, Y y Z.

1.2. ANTECEDENTES

Desde que el checo **Karel Capek** introdujo el término robot» en su drama R.U.R, en el año de 1920, ya han pasado muchos años. En dicho drama se crea una serie de hombres artificiales llamados robots, con una gran similitud con los humanos, creados principalmente para reemplazarlos en el trabajo [\[1\]](#).

La robótica puede decirse que tiene orígenes ancestrales, antiguamente conocidos como autómatas, que son los precursores de los robots actuales. Hay que anotar que la idea era reproducir la naturaleza y estos autómatas tenían forma humanoide o de animales.

Actualmente, la robótica es una ciencia que tiene gran incidencia en diferentes ingenierías, como mecánica, electrónica, informática e inteligencia artificial. Estos

campos del conocimiento y otros son la base para diseñar, construir e implementar robots.

Los robots se comportan como entidades o sistemas electromecánicos que normalmente son controlados por un software de programación o sistemas eléctricos, los cuales determinan sus movimientos para resolver una tarea específica.

La robótica cada día cobra más importancia para la vida cotidiana y esto se debe gracias a que los robots han salido de las grandes industrias, generándose aplicaciones para la medicina, el hogar, entre otras áreas que lo requieran. En el campo médico, por ejemplo se reemplazan extremidades para mejorar la calidad de vida de muchas personas, como en las fuerzas militares en las que algunos soldados las pierden, cuando están en combate.

El proyecto se basa en un robot esférico, con muchas aplicaciones, en áreas como la agricultura, fuerzas militares. Algunos de los robots móviles con estructura esférica son:

- **Rosphere**, es un robot pensado para moverse por zonas complicadas, fue desarrollado en la Universidad Politécnica de Madrid [\[2\]](#).
- **Guardbot** es un robot todoterreno, capaz de desplazarse por tierra y agua [\[3\]](#).
- **BB-8** es un androide astro mecánico que aparece en la película Star Wars. El cuerpo de este se mueve de forma independiente a la cabeza [\[4\]](#).
- **MorpHex** es un robot similar a un cangrejo, el cual se puede transformar en una esfera y viceversa. [\[5\]](#)
- La nasa se encuentra trabajando en un tipo de robot esférico espacial, que tenga la capacidad de flotar y servir como asistentes de los astronautas [\[6\]](#).
- **El Ballbot**. Aunque este no es un robot esfera, se equilibra dinámicamente sobre una sola esfera de metal cubierta de uretano [\[7\]](#).

1.3. DEFINICIÓN DEL PROBLEMA

Existe una problemática real en los sistemas robóticos actuales, basan su principio de funcionamiento en ruedas o patas, los cuales limitan su operación en diferentes áreas, además de movimientos no omnidireccionales y con propensión a volcamientos.

Debido a esto se pretende solucionar este problema, implementando un robot esfera, el cual no tenga la necesidad de utilizar ruedas internas para su operación, además su electrónica estaría basada en sistemas de desarrollo de altera, FPGA y Servo Kit Terasic.

1.4. JUSTIFICACIÓN

Muchas de las tecnologías que se han implementado en el ámbito de la robótica han sido diseñadas con la idea de brindar una solución a diferentes situaciones o necesidades.

Este tipo de robot esférico, es una alternativa a los movimientos tradicionales que utilizan ruedas, orugas o patas. Se los resultados de este proyecto son satisfactorios, el disponer de estos, permitirá avanzar en proyectos futuros, como sistemas colaborativos para realizar el transporte de grandes pesos, que logren operar dentro de agua, tuberías o sistemas donde utilizar ruedas o patas no es una opción.

1.5. OBJETIVOS

1.5.1.OBJETIVO GENERAL:

Implementar un robot en forma de esfera, electrónico, portátil, con bajo consumo de energía, y mando remoto para su operación.

1.5.2.OBJETIVOS ESPECÍFICOS:

- Comparar los sistemas mecánicos estructurales usables en un robot esférico, a fin de seleccionar uno para su implementación.
- Acoplar el sistema electrónico hardware y software, que requiere el robot esférico, para lograr el movimiento en cualquier dirección, portable y con bajo consumo de potencia.

- Realizar la validación del funcionamiento del robot esférico y obtención de especificaciones.

1.6. MARCO REFERENCIAL

1.6.1. GENERALIDADES

Todo robot está compuesto por una estructura mecánica, y un sistema de accionamiento, que se relacionan entre sí, para producir un movimiento gracias a una unidad de mando. Esta unidad de mando puede ser implementada con dispositivos electrónicos

1.6.1.1. ROBOTS ESFERAS

Muchos robots con forma exterior esférica basan su principio de funcionamiento interno en ruedas tradicionales, las cuales, al interactuar internamente con la esfera, hacen que el sólido se mueva. Con el fin de eliminar el desplazamiento tradicional de ruedas, en este proyecto se utilizarán servomotores como actuadores para lograr el movimiento.

La electrónica para la actuación del movimiento estará implementada sobre el sistema “Servo Motor Kit Terasic”, el cual recibirá los comandos de actuación desde algoritmos implementados en hardware en una FPGA del sistema DE0-nano, estos algoritmos serán escritos en lenguaje de programación VHDL.

Actualmente algunos robots esfera son comercializados como juguetes y/o como elementos para aprendizaje. El ejemplo más representativo es Sphero, el cual basa su movimiento en ruedas.

Un robot esfera se compone de dos partes principales, un cuerpo exterior esférico de excelente robustez, en su interior un mecanismo encargado de inducir movimiento. Una de las más importantes características de este tipo de robots, es

la alta maniobrabilidad, con una alta capacidad de recuperación a colisiones, movimientos omnidireccionales.

Pueden comportarse como un sistema holonómico, esto quiere decir que poseen movimientos en todos sus grados de libertad controlables, dependiendo de sus consideraciones de diseño.

1.6.1.2. MODULACIÓN POR ANCHO DE PULSO (PWM)

El PWM de una señal o fuente de alimentación consiste en la modulación de su ciclo de trabajo, ya sea para transmitir información a través de un canal de comunicaciones o de controlar la cantidad de energía enviada a una carga.

Es una forma muy eficaz de establecer cantidades intermedias de potencia eléctrica entre la máxima potencia y apagado. Una de las aplicaciones típicas del PWM es el control de velocidad en motores eléctricos.

1.6.1.3. ACTUADORES

Un actuador es cualquier elemento que tiene la capacidad de convertir un tipo de energía en otra, estos dispositivos son encargados de convertir movimiento rotativo en movimiento lineal, serán los encargados de realizar los desplazamientos sobre la esfera. Para este proyecto se utilizarán servomotores con su respectivo driver, que se detallarán más adelante.

1.6.1.4. MANDO A DISTANCIA.

Se trata de un dispositivo electrónico, usado para establecer comunicación entre una máquina de manera remota, emisor y receptor, existen diferentes maneras para establecer esta comunicación, módulos *RF*, *Bluetooth*, mando cableado.

CAPÍTULO 2. COMPONENTES

2.1. DESARROLLO DEL SISTEMA

El gráfico de la figura 1, representa los módulos que componen la estructura del proyecto robot esfera. Un dispositivo móvil con tecnología Android para la cual se crea una APP, que es la encargada de enviar la información del usuario hacia el módulo *Bluetooth*, este es el encargado de recibir dicha información y enviarla al módulo PWM, este a su vez la interpreta y la envía a los actuadores. Por otro lado, se encuentra la parte sensorial que se compone del módulo acelerómetro y un comparador, los cuales en conjunto envían órdenes también hacia el módulo PWM, para obtener un correcto funcionamiento de los actuadores.

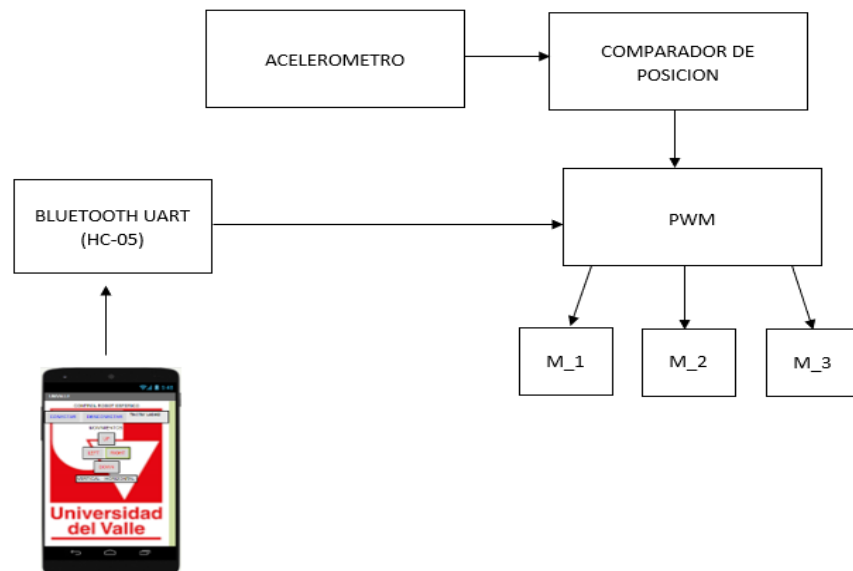


Figura. 1. Diagrama de bloques del proyecto.

Se presenta el diseño de ingeniería que se desarrolló para llevar a cabo un prototipo de robótica móvil. Este proyecto se divide en tres partes principales que son:

- Interfaz de usuario
- Unidad de control y
- Actuadores

La interfaz de usuario es una aplicación instalada en un teléfono celular con tecnología Android, cuya conexión se hace mediante un dispositivo *Bluetooth*.

La unidad de control está compuesta por una tarjeta de desarrollo de altera (DE0-nano), la cual se encarga de los procesos y flujo de información.

El sistema de actuadores, está conformado por un servomotor kit de Altera (SMK) que se compone de un circuito de control y sus servomotores, los cuales están ligados a un péndulo.

2.2. SISTEMA PARA EL MANEJO SERVOMOTOR

El sistema de desarrollo para el caso es la tarjeta DE0-nano de Altera mediante su puerto GPIO por el cual se proporciona el PWM requerido para los servomotores.

Con este driver se pueden controlar 24 servos, en este proyecto se utilizaron tres.

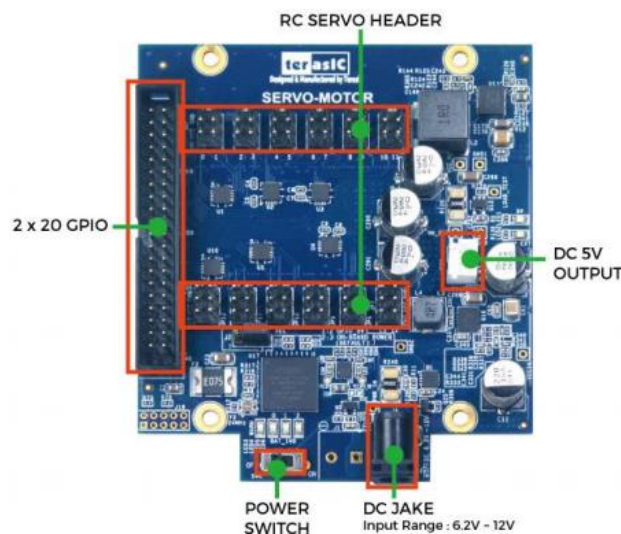


Figura. 2. Tarjeta de desarrollo para el control de servomotores.

La figura 3 muestra el modo de conexión entre el servomotor y la tarjeta DE0_NANO.

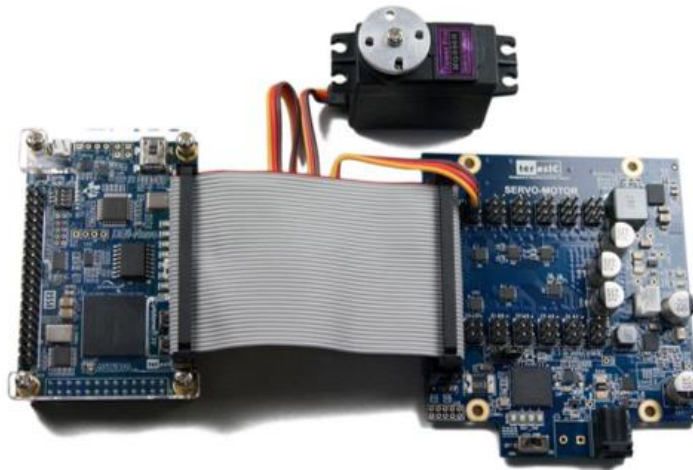


Figura. 3. Conexión driver, servomotor y tarjeta DE0_NANO.

2.3. SERVOMOTOR

El MG996R es un servomotor con engranajes metálicos de gran calidad, pensado para proyectos de robótica y mecatrónica. Su tensión de funcionamiento varía entre 4,8V y 7,2V, siendo el valor más adecuado de funcionamiento de 6V., estos voltajes de operación son regulados directamente por el driver.

Este servomotor necesita un periodo de 20ns, para su operación, posee un ángulo de rotación de 180 grados, el cual se modificó internamente para tener una operación de 360 grados

A pesar de que las velocidades de giro son pequeñas frente a motores de corriente continua, su precisión los hace ideales para muchas aplicaciones mecánicas. En la figura 4 se puede observar este servomotor, con sus respectivas dimensiones.



Figura. 4. Dimensiones servomotor MG996R

En la Tabla 1 se muestra una descripción del conector del servomotor, la función que cumple cada uno de sus pines.

Tabla N° 1. Configuración del conector.

N° de cable	Color del cable	Descripción
1	Marrón	Cable de tierra conectado a la tierra del sistema
2	Rojo	Alimenta el motor normalmente se usa + 5V
3	Naranja	La señal PWM se transmite a través de este cable para conducir el motor

2.4. TARJETA DE DESARROLLO DE0-NANO

Es una tarjeta de desarrollo FPGA, en donde se programa el código realizado, en este proyecto se utilizó el lenguaje de programación basado en VHDL, un acelerómetro que mide la acción de la gravedad en las direcciones x,y,z. y diferentes puertos de entrada y salida, entre otros dispositivos que se pueden apreciar en las figuras (5 y 6).

La herramienta de desarrollo utilizada para la realización del código y configuración de la tarjeta de desarrollo DE0_NANO fue el *Quartus Prime 18.1*

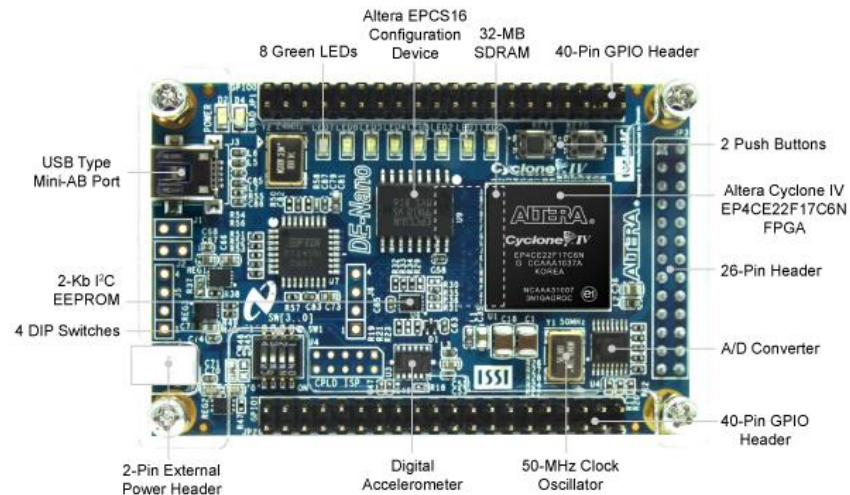


Figura. 5. Vista superior, tarjeta de desarrollo DE0_NANO.

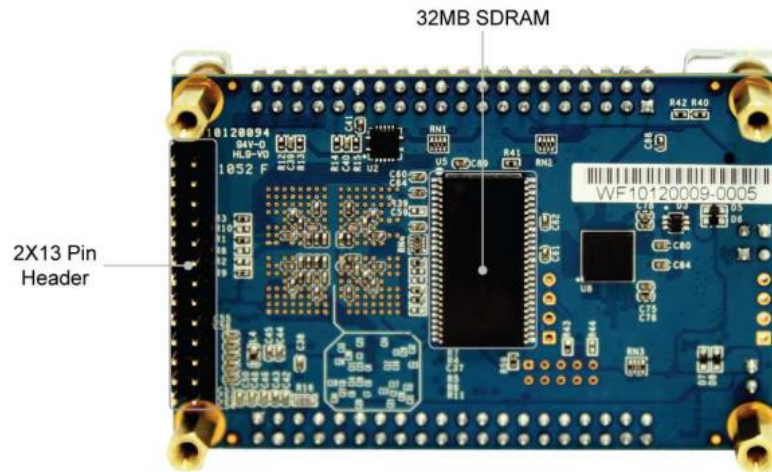


Figura. 6. Vista inferior, tarjeta de desarrollo DE0_NANO.

2.5. ADXL345 ACELERÓMETRO

Este dispositivo viene incorporado en el sistema de desarrollo DE0-NANO. Es el encargado de dar la posición del robot en los planos x,y,z. y así se sabrá exactamente que actuador entrará en funcionamiento, después de recibir órdenes de parte del usuario.

Un acelerómetro es un dispositivo capaz de medir la aceleración de la gravedad sobre un objeto, está dada en metros por segundo cuadrado (m/s^2), se trabaja con la aceleración de la gravedad terrestre de $9,8 m/s^2$, Estos dispositivos son de gran utilidad para aplicaciones de movilidad u orientación.

2.5.1. CÓMO FUNCIONA EL ACELERÓMETRO

Estos dispositivos detectan las fuerzas de aceleración, ya sea de manera estática o dinámica. Cuando es estática influye sobre el la fuerza de la gravedad a diferencia de cuando es dinámica se pueden incluir vibraciones y movimientos.

Su respuesta depende de la gravedad en función de la posición en el plano de R3 como se logra apreciar en la figura 7, la cual ilustra sus diferentes posiciones x,y,z.

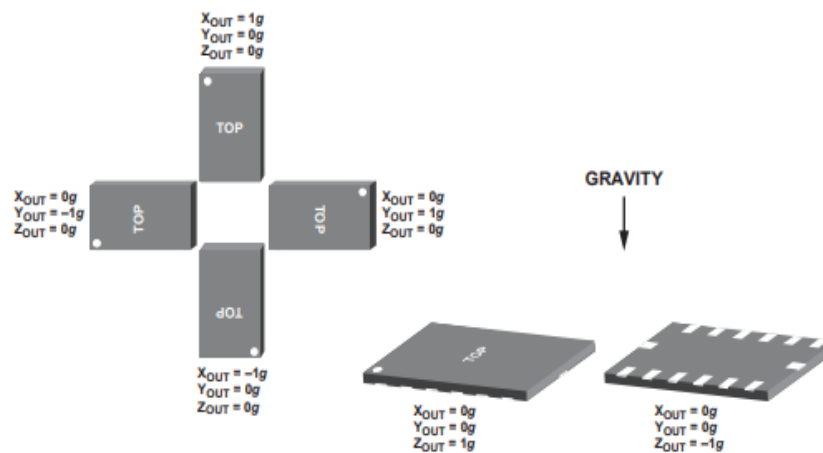


Figura. 7. Respuesta de salida en función de la orientación y gravedad.

Este dispositivo posee una resolución de hasta 16g., soporta protocolo serial SPI, una salida de 16 bits en completo A2 y soporta un rango de alimentación de 2 a 3,6v. Este tipo de dispositivo muy utilizado para aplicaciones de dispositivos móviles.

2.6. CYCLONE IV

Este tipo de producto promete ser de una gran revolución en el mercado, ya que son dispositivos de bajos costos y alta fiabilidad.

2.7. LENGUAJE VHDL

VHDL es un lenguaje de especificación definido por el IEEE (*Institute of Electrical and Electronics Engineers*) (ANSI/IEEE 1076-2008) utilizado para describir sistemas digitales y para la automatización de diseño electrónico.

Puede ser usado de forma general para describir cualquier circuito digital, se usa principalmente para configurar PLD (*Programmable Logic Device* - Dispositivo Lógico Programable), FPGA (*Field Programmable Gate Array*), A.S.I.C. y similares.

2.8. LA ESTRUCTURA INTERNA DE LA ESFERA Y SU FUNCIONAMIENTO

La estructura interna de la esfera se realizó tal cual se muestra en la figura 8. El material utilizado fue el madeflex, ubicando en la parte central las tarjetas de comando y en tres de sus extremos se ubicaron los servomotores, los cuales son los encargados de generar el movimiento mediante un péndulo ligado a su eje. En el extremo restante se ubicó una masa con un peso similar al de los servomotores para darle equilibrio al sistema.

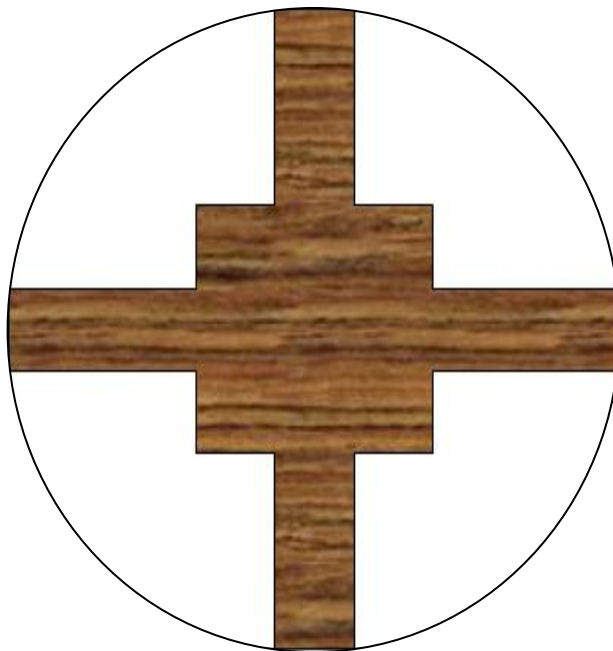


Figura. 8. Diseño estructura interna robot.

2.8.1. LA ESFERA

La carcasa de la esfera está compuesta por dos domos de 30cm de diámetro y un espesor de 3mm, estos domos no se consiguen comercialmente, por lo que tocó adquirir el modelo más parecido y hacerlo adecuar a las necesidades del proyecto, tal como se aprecia en la figura 9.



Figura. 9. Fotografía original domos esféricos.

2.9. MÓDULO BLUETOOTH HC-05

El módulo *Bluetooth* es un dispositivo, el cual se conecta a la tarjeta de desarrollo DE0_NANO. Es el encargado de recibir los datos de operación proveniente de la App creada para el robot, por parte de la interfaz de usuario, que para este caso se implementó en un dispositivo celular Android.

Este módulo *Bluetooth* maneja un protocolo UART RS232, y permite la programación de sus diferentes parámetros, tales como velocidad, paridad y nombre.

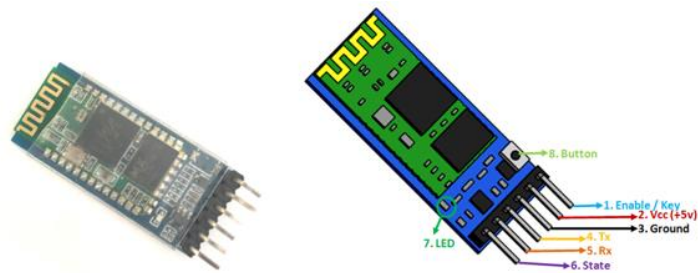


Figura. 10. Modulo Bluetooth HC-05.

La figura 10 muestra la estructura física de este dispositivo e indica sus diferentes periféricos, El tamaño del dato es de 1 byte enviado o recibido de manera serial.

2.10. INTERFAZ DE USUARIO

La interfaz de usuario permite al usuario comunicarse con el robot, esta se realiza desde un dispositivo celular Android, en este proyecto se desarrolló esta interfaz. Mostrada en la figura 11.

El objetivo de esta aplicación es lograr un perfecto control y funcionamiento del robot.

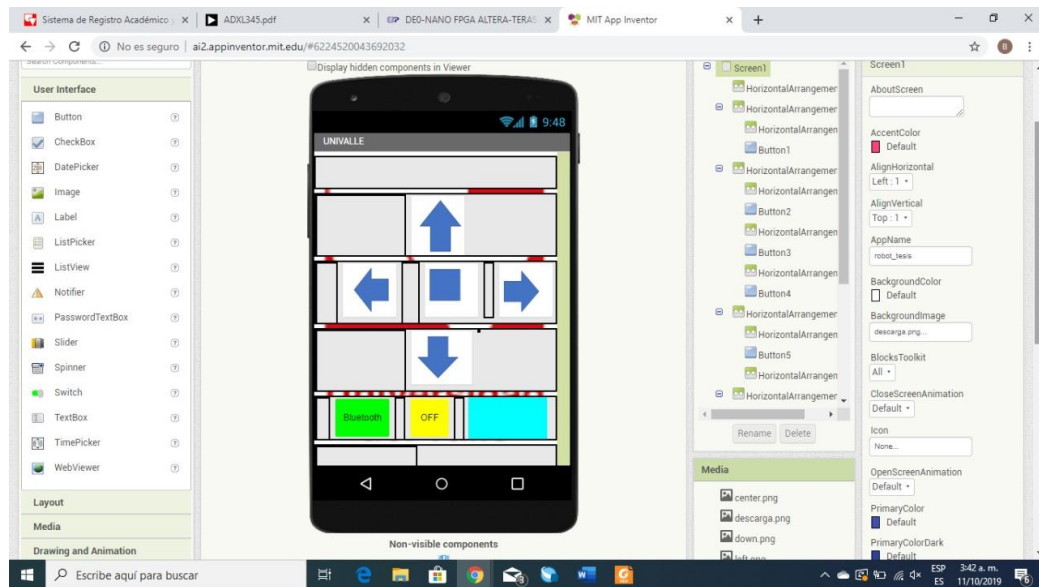


Figura. 11. Diseño APP interfaz de usuario.

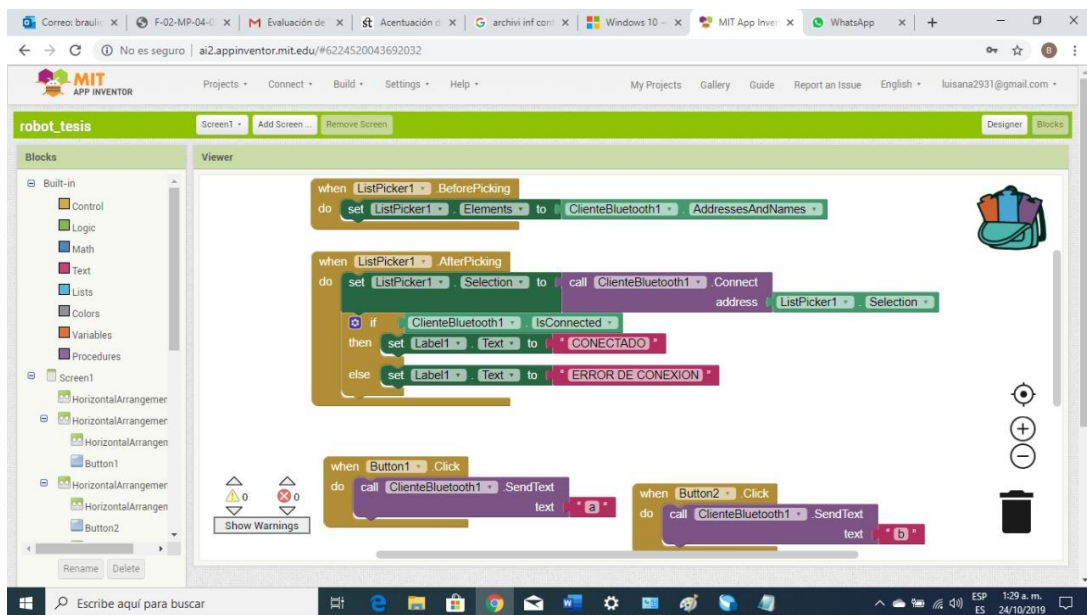


Figura. 12a. Diagrama de bloques APP interfaz de usuario.

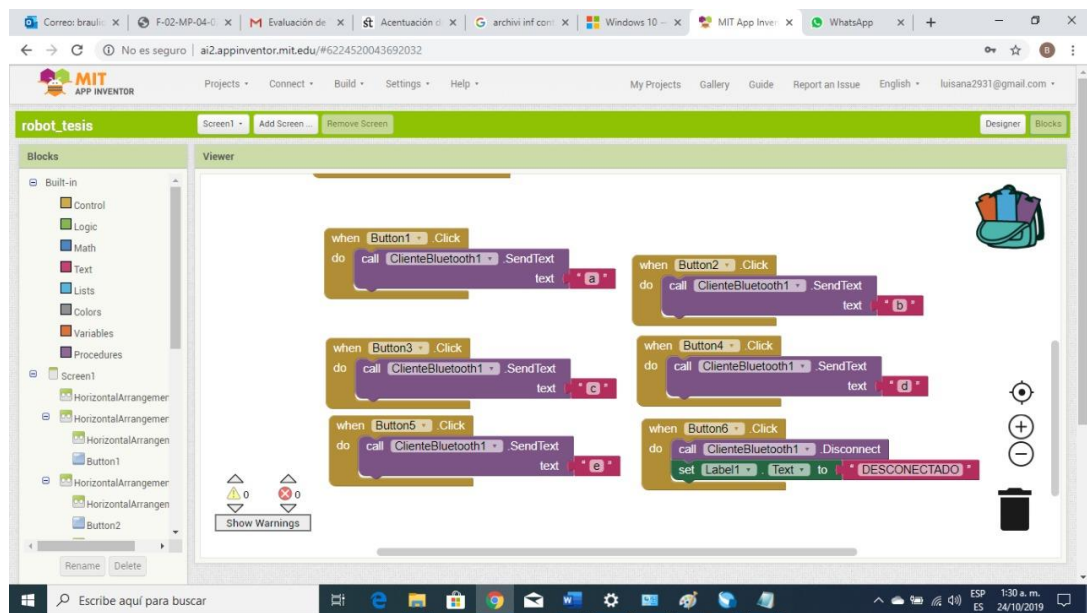


Figura. 12b. Diagrama de bloques APP interfaz de usuario.

Para la implementación de esta App se utilizó una aplicación online llamada *App Inventor*, a manera de descripción las flechas indican la dirección hacia donde se desea que vaya el robot, el recuadro en el centro de las flechas es el *stop*, en la parte inferior se logran visualizar tres recuadros, el primero es para gestionar la

conexión con el *Bluetooth*, con el segundo botón se realiza la desconexión y el último indica tres estados que son: conectado, desconectado y error de conexión.

Para la implementación en diagrama de bloques de la APP, tal como se ilustra en las figuras 12a y 12b, lo que se pretende es que los botones de las flechas envíen datos hacia el robot, para este caso el dato a enviar es indiferente, la idea es saber interpretar esos datos enviados.

Tabla N° 2. Parámetros App.

Button	Carácter	Código ASCII	Binario
"UP"	a	97	01100001
"LEFT"	b	98	01100010
"STOP"	c	99	01100011
"RIGHT"	d	100	01100100
"DOWN"	e	101	01100101

Al ser enviados estos datos a la unidad de control, interpreto el dato binario de tal forma que me realice los desplazamientos solicitados por el usuario.

2.11. ALIMENTACIÓN

La alimentación se realizó con dos baterías cada una de 3,7v 3000mA, tal como la indicada en la figura 13, las cuales se encargarán de alimentar la tarjeta de desarrollo (DE0_nano) y al driver; los actuadores se alimentan a través del driver.



Figura. 13. Batería de 3.7V.

2.12. FRECUENCIA DE OPERACIÓN SERVOMOTORES

La mayoría de los sistemas digitales vienen equipados con un reloj, a no ser que sean de tipo asíncrono. La tarjeta de desarrollo DE0-nano viene con un reloj de 50Mhz (20ps), ya que los servomotores trabajan a 50hz (20ms).

Se escribió un código para dividir la frecuencia de la tarjeta de desarrollo y así obtener un reloj de 50hz a la salida.

$$\frac{50 \times 10^6 \text{ Hz}}{50 \text{ Hz}} = 1.000.000 \text{ ciclos de reloj}$$

Este incorpora un contador que es el encargado de llevar la frecuencia de 50Mhz a 50hz cada 10^6 ciclos.

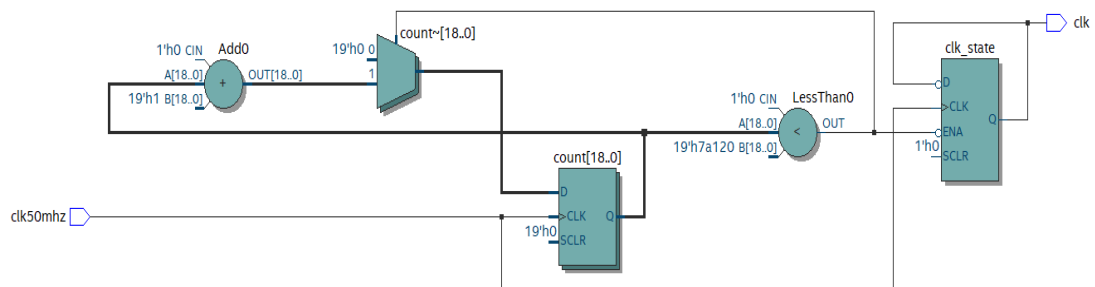


Figura. 14. RTL Módulo divisor de frecuencia.

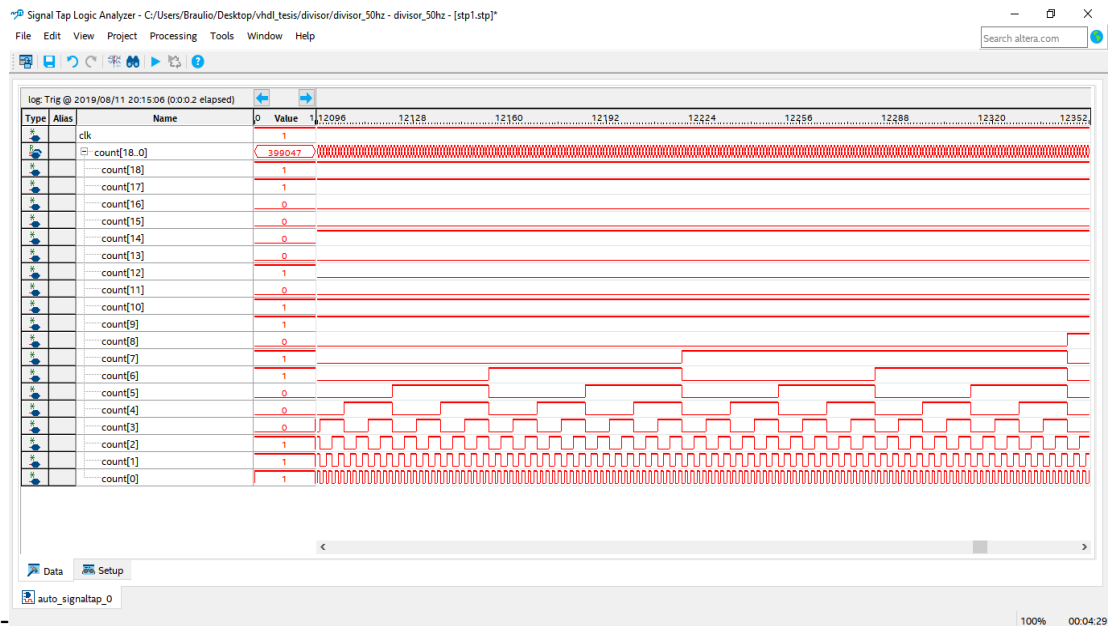


Figura. 15. Módulo divisor de frecuencia.

Las figuras 14 y 15 muestran el RTL y la señal de salida *Signal Tap* resultantes una vez creado y compilado el código.

Código divisor de frecuencia

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
entity divisor_50hz is
port (
    clk50mhz:    in STD_LOGIC;
    clk:         out STD_LOGIC
);
end divisor_50hz;

architecture behavior of divisor_50hz is
    constant max_count: INTEGER := 1000000;
    signal count: INTEGER range 0 to max_count;
    signal clk_state: STD_LOGIC := '0';

begin
    process(clk50mhz, clk_state, count)
    begin
        if clk50mhz'event and clk50mhz='1' then
            if count < max_count then
                count <= count+1;
            else
                clk_state <= not clk_state;
                count <= 0;
            end if;
        end if;
    end process;

    process (clk_state)
    begin
        clk <= clk_state;
    end process;

end behavior;
```

2.13. PWM

Ya con la frecuencia de 50hz, el siguiente paso es obtener la señal PWM que va al driver (SMK), esta señal será la encargada de controlar los servomotores.

De acuerdo con las características del servomotor Mg996r

- Frecuencia de actualización 20ms
- Ancho de pulso 600us y 2400us

$$rango = t_{max} - t_{min}$$

$$rango = 2400us - 600us = 1800us$$

PWM. Para el diseño del PWM se debe crear un contador con dos señales de entrada, las cuales se utilizan como el control de los servomotores avance y retroceso.

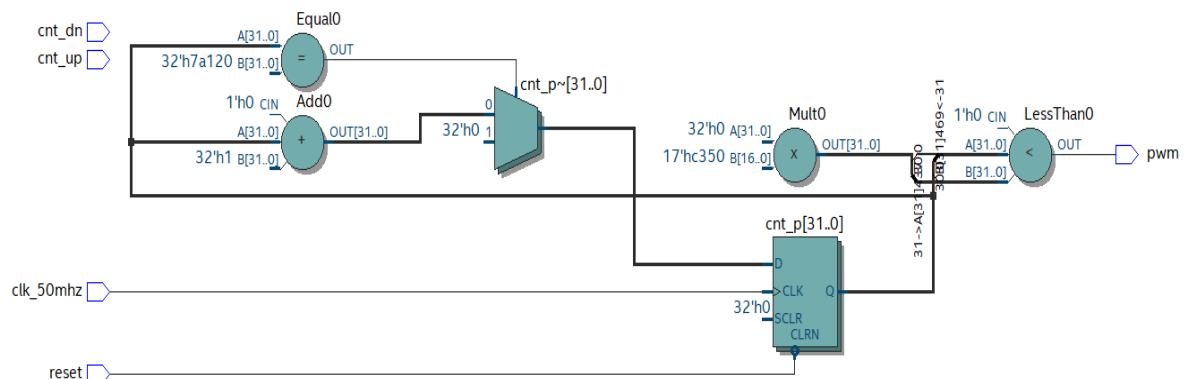


Figura. 16. RTL módulo de PWM.

Código PWM

```

library IEEE;
use IEEE.std_logic_1164.all;

entity pwm_motor is
generic(periodo: integer:=1000000);
port(clk_50mhz: in std_logic;
reset: in std_logic;
--inc_dec: in std_logic;
rotar: in std_logic_vector(1 downto 0);
pwm: out std_logic);
end entity pwm_motor;
architecture comportamental of pwm_motor is
signal cnt_p: integer;
signal angulo_ciclos: integer;
begin
process(clk_50mhz, angulo_ciclos)
begin
if reset = '0' then
cnt_p <= 0;
elsif clk_50mhz' event and clk_50mhz = '1' then
if cnt_p = periodo then
cnt_p <= 0;
else
cnt_p <= cnt_p + 1;
end if;
if cnt_p = angulo_ciclos then
pwm <= '1';
else
pwm <= '0';
end if;
end if;
end process;

process(rotar)
begin
case rotar is
when "00" =>
angulo_ciclos <= 25000; --(duty cycle 0,5ms) 0 grados
when "01" =>
angulo_ciclos <= 75000; --(duty cycle 1,5ms) 90 grados
when "10" =>
angulo_ciclos <= 125000; --(duty cycle 2,5ms) 180 grados
when others =>
angulo_ciclos <= 125000; --(duty cycle 2,5ms) 180 grados
end case;
end process;
end architecture comportamental;

```

El módulo del acelerómetro se encarga de entregar el valor de las coordenadas x,y,z dependiendo de la posición en la cual se encuentre ubicado el robot.

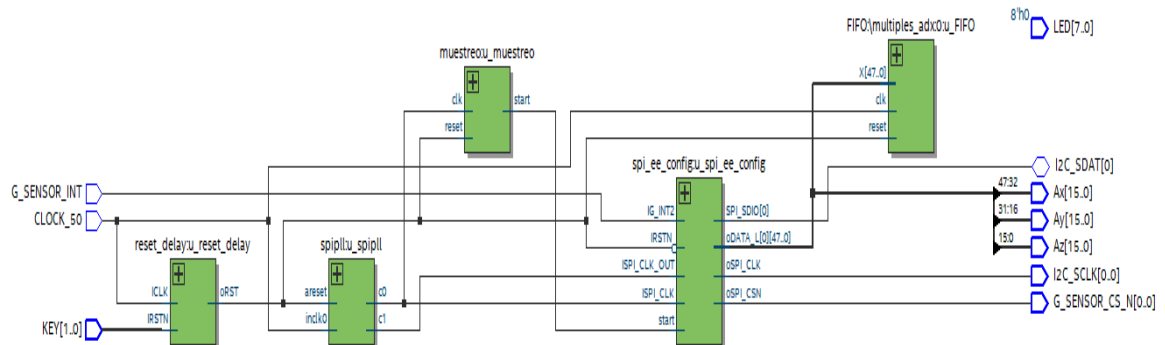


Figura. 17. RTL módulo del acelerómetro.

Ya que los motores están ubicados sobre el semiplano positivo de cada eje, los datos de salida de este módulo se comparan para saber cuál motor entra en funcionamiento cuando según sea la orden desde la interfaz de usuario.

2.14. MÓDULO UART_RS232

Este módulo es el encargado de recibir los datos provenientes del módulo *Bluetooth* HC-05 de forma serial, tal cual como se muestra en la figura 18. Los interpreta y entrega una respuesta de un byte de forma paralela. Este se implementó con un sistema de comunicación asíncrona UART (*Universal Asynchronous Receiver/Transmitter*)

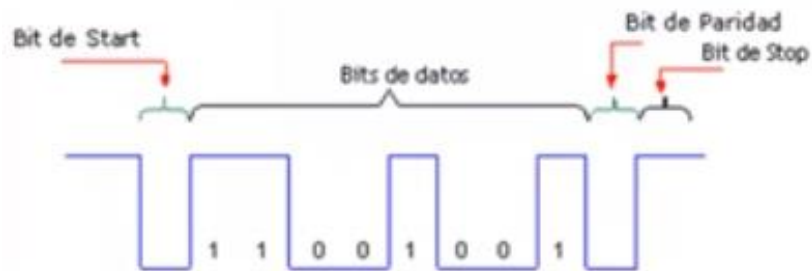


Figura. 18. Salida serial del módulo Bluetooth.

La implementación de este módulo en la FPGA presentó muchos problemas de inestabilidad, debido a que inicialmente el módulo *Bluetooth* estaba alimentado independiente, además, la frecuencia de operación no fue la adecuada. Después de muchos intentos, se cambió la velocidad de transmisión del *Bluetooth* a 38400 bps y se utilizó el puerto GPIO-1 de la DE0-nano, de la que se obtuvo la alimentación para dicho módulo.

Tal como se aprecia en la figura 19, los pines 11 y 12 son vcc y gnd respectivamente.

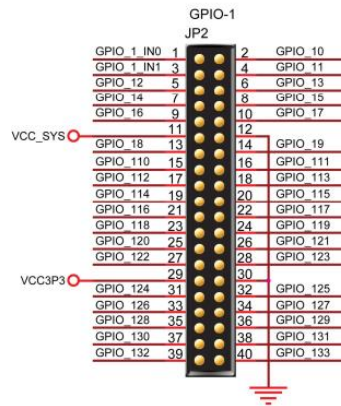


Figura. 19. Distribución de pines puerto GPIO-1 DE0_NANO.

Las figuras 20 y 21 ilustran el resultado *Signal tap* y *RTL de Quartus 18.1*. Una vez asignados los pines de entrada y realizando la conexión del módulo *Bluetooth* a la tarjeta de desarrollo y enganchando el teléfono celular Android con una APP creada para tal fin.

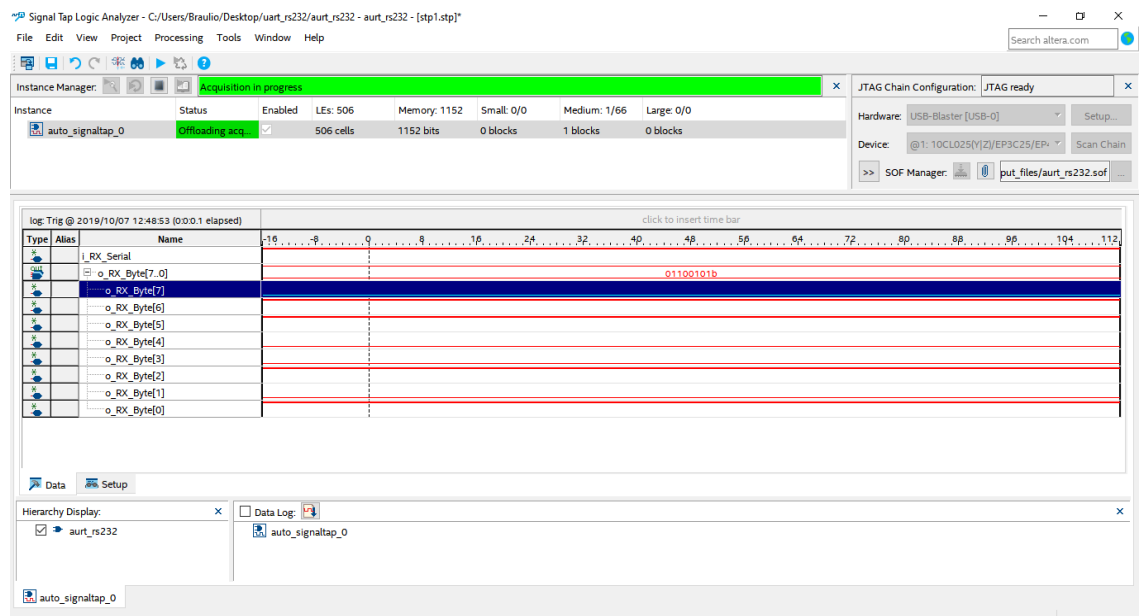


Figura. 20. Signal tap módulo UART entrada serial, salida paralela.

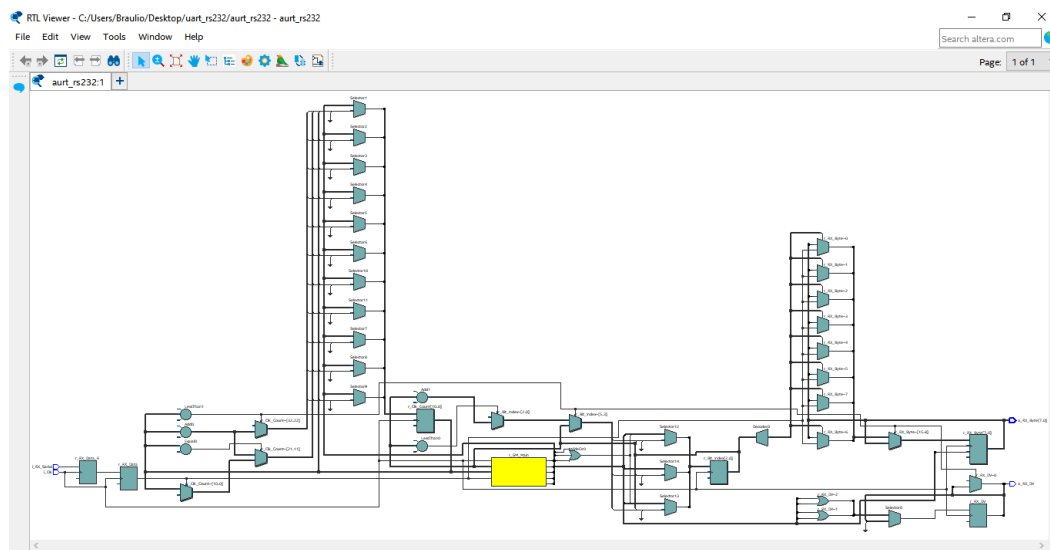


Figura. 21. RTL módulo UART.

Código Bluetooth

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity aurt_ns232 is
  generic (
    g_CLKS_PER_BIT : integer := 1302 -- FRECUENCIA DEL BLUETOOTH 36400 BPS (50MHz/36400 = 1302)
  );
  port (
    i_Clk : in std_logic; -- RELOJ DEL SISTEMA 50MHz
    i_RX_Serial : in std_logic; -- Entrada serial proveniente del modulo Bluetooth HC-05
    o_RX_DV : out std_logic;
    o_RX_Byte : out std_logic_vector(7 downto 0) -- Salida paralela de 1 byte
  );
end aurt_ns232;

architecture rtl of aurt_ns232 is

  type t_SM_Main is (s_idle, s_RX_Start_Bit, s_RX_Data_Bits, s_RX_Stop_Bit, s_Cleanup);
  signal r_SM_Main : t_SM_Main := s_idle;
  signal r_RX_Data_R : std_logic := '0';
  signal r_RX_Data : std_logic := '0';
  signal r_Clk_Count : integer range 0 to g_CLKS_PER_BIT-1 := 0;
  signal r_Bit_Index : integer range 0 to 7 := 0; -- 8 Bits Total
  signal r_RX_Byte : std_logic_vector(7 downto 0) := (others => '0');
  signal r_RX_DV : std_logic := '0';

begin

  p_SAMPLE : process (i_Clk)
  begin
    if rising_edge(i_Clk) then
      r_RX_Data_R <= i_RX_Serial;
      r_RX_Data <= r_RX_Data_R;
    end if;
  end process p_SAMPLE;

  p_UART_RX : process (i_Clk)
  begin
    if rising_edge(i_Clk) then
      case r_SM_Main is
        when s_idle =>
          r_RX_DV <= '0';
          r_Clk_Count <= 0;
          r_Bit_Index <= 0;
          if r_RX_Data = '0' then -- Start bit detected
            r_SM_Main <= s_RX_Start_Bit;
          else
            r_SM_Main <= s_idle;
          end if;
        when s_RX_Start_Bit =>
          if r_Clk_Count = g_CLKS_PER_BIT-1/2 then
            if r_RX_Data = '0' then
              r_Clk_Count <= 0; -- reset counter since we found the middle
              r_SM_Main <= s_RX_Data_Bits;
            else
              r_SM_Main <= s_idle;
            end if;
          else
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main <= s_RX_Start_Bit;
          end if;
        when s_RX_Data_Bits =>
          if r_Clk_Count < g_CLKS_PER_BIT-1 then
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main <= s_RX_Data_Bits;
          else
            r_Clk_Count <= 0;
            r_RX_Byte(r_Bit_Index) <= r_RX_Data;
            if r_Bit_Index < 7 then
              r_Bit_Index <= r_Bit_Index + 1;
              r_SM_Main <= s_RX_Data_Bits;
            else
              r_Bit_Index <= 0;
              r_SM_Main <= s_RX_Stop_Bit;
            end if;
          end if;
          --verifica el bit 1 de stop para finalizar el dato
        when s_RX_Stop_Bit =>
          if r_Clk_Count < g_CLKS_PER_BIT-1 then
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main <= s_RX_Stop_Bit;
          else
            r_RX_DV <= '1';
            r_Clk_Count <= 0;
            r_SM_Main <= s_Cleanup;
          end if;
        when s_Cleanup =>
          r_SM_Main <= s_idle;
          r_RX_DV <= '0';
          when others =>
            r_SM_Main <= s_idle;
          end case;
      end case;
    end process p_UART_RX;

    o_RX_DV <= r_RX_DV;
    o_RX_Byte <= r_RX_Byte;

  end rtl;

```

Este módulo se compone de dos entradas y dos salidas, entradas señal de reloj de 50 Mhz, esta señal internamente es dividida 38400 bps, la cual es la velocidad de transmisión del módulo *Bluetooth* HC-05, que es el que controla la otra entrada de este módulo, las salidas son de 1 byte correspondiente al dato recibido.

Las figuras 22 y 23 son fotografías de la operación de este módulo ya programado el código en la tarjeta de desarrollo DE0_NANO.



Figura. 22. Foto real sin enviar datos.



Figura. 23. Foto real recibiendo datos.

2.15. DISEÑO ESTRUCTURAL PARA EL MANEJO DE LOS DATOS DEL ACELERÓMETRO

Para este módulo se emplearon tres sub-módulos como se puede apreciar en la figura 24.

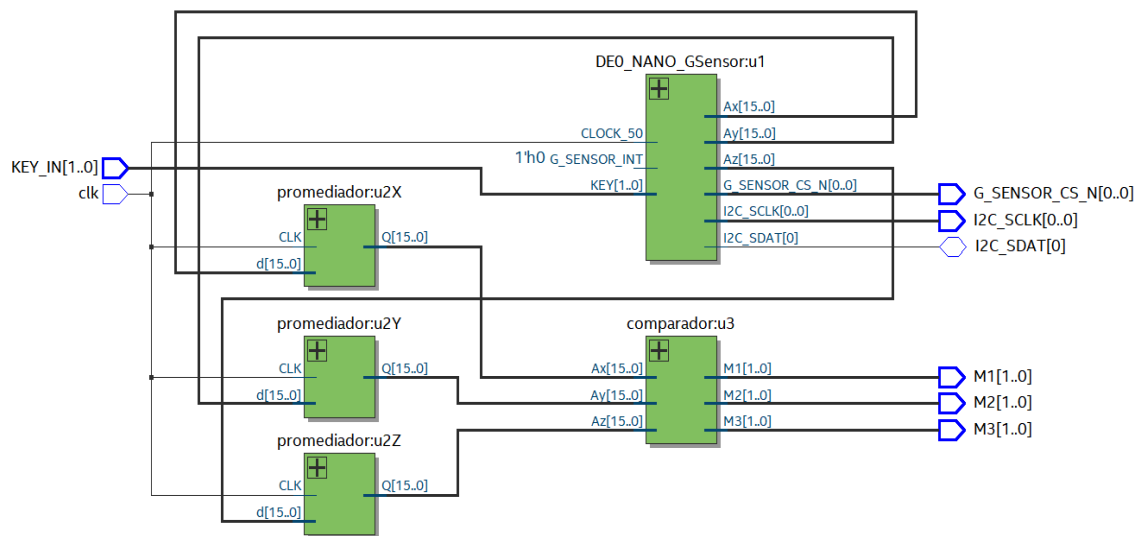


Figura. 24. RTL diseño estructural.

DEO_NANO_GSensor, es el modulo encargado de capturar la lectura de datos dependiendo cada uno de los ejes ordenados x,y,z, estos datos se entregan en un vector de 16 bits en complemento A2, ver figura 25.

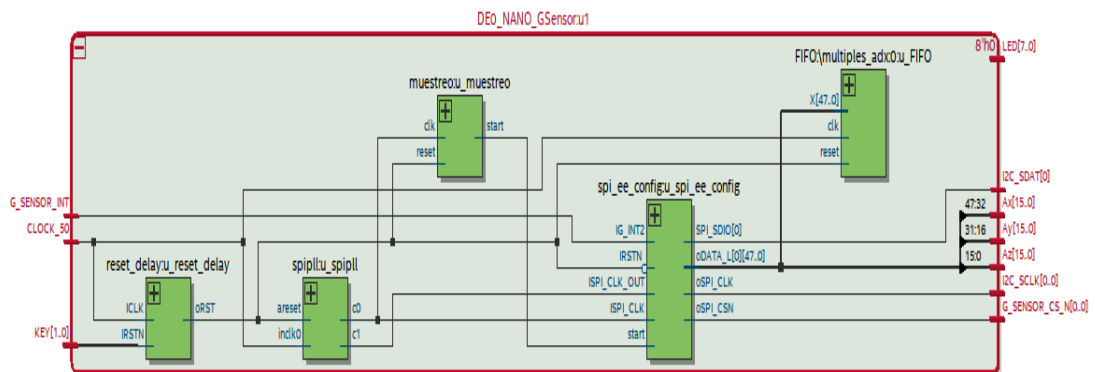


Figura. 25. RTL diseño estructural manejo de datos del acelerómetro.

Código acelerómetro

```

library IEEE;
use IEEE.std_logic_1164.all;
use work.spi_param.all;

entity DE0_NANO_GSensor is
port
(
    -- Input ports
    CLOCK_50          : in std_logic;
    KEY               : in std_logic_vector(1 downto 0);
    G_SENSOR_INT      : in std_logic;
    ~G_SENSOR_INT     : in std_logic_vector(No_ADX-1 downto 0);

    -- Input ports
    I2C_SDA1          : inout std_logic_vector(No_ADX-1 downto 0);

    -- Output ports
    ~spiControlExt    : out std_logic_vector(No_ADX-1 downto 0);
    Ax                : out std_logic_vector(15 downto 0);
    Ay                : out std_logic_vector(15 downto 0);
    Az                : out std_logic_vector(15 downto 0);

    ~I2C_SCLK         : out std_logic;
    I2C_SCLK          : out std_logic_vector(No_ADX-1 downto 0);
    ~G_SENSOR_CS_N    : out std_logic;
    G_SENSOR_CS_N     : out std_logic_vector(No_ADX-1 downto 0);
    LED               : out std_logic_vector(7 downto 0)
);
end DE0_NANO_GSensor;

architecture rtl of DE0_NANO_GSensor is
-- DECLARATIONS
    signal sI2C_SCLK : std_logic;
    signal sG_SENSOR_CS_N : std_logic;

-- Components
    component reset_delay is
    port
    (
        -- Input ports
        CLK : in std_logic;
        RSTN : in std_logic;

        -- Output ports
        aRST : out std_logic
    );
    end component;

    component spipll is
    port
    (
        -- Input ports
        areset : in std_logic;
    );
    end component;

    component FIFO is
    port
    (
        -- Input ports
        clk : in std_logic;
        reset : in std_logic;
        X : in std_logic_vector(50_Data, downto 0);

        -- Output ports
        Z : out tipoDatosM
    );
    end component;

    signal start :
        dly_rst,
        muestrador,
        spy_clk,
        spy_clk_out : std_logic;

    signal data_x :
        soDATA_L : tipoDatos;
        soDATA_L : tipoDatosM;
        spiControl : std_logic;

begin
    u_reset_delay : reset_delay
    port map (
        -- Input
        CLK => CLOCK_50,
        RSTN => KEY(0),
        -- Output
        aRST => dly_rst,
        -- Inout
        ~formal_inout => signal
    );

    u_muestreo : muestrador
    port map (
        -- Input
        inclk => CLOCK_50,
        areset => dly_rst,
        -- Output
        c0 => spy_clk,
        c1 => spy_clk_out
    );

    u_spi_ee_config : spi_ee_config
    port map (
        -- Input
        start : in std_logic;
        ~G_SENSOR_INT : in std_logic;
        RSTN : in std_logic;
        SPI_CLK : in std_logic;
        SPI_CLK_OUT : in std_logic;
        -- Output
        spiControl : out std_logic;
        sSPI_CLK : out std_logic;
        sSPI_CS_N : out std_logic;
        sDATA_L : out std_logic;
        sDATA_H : out std_logic_vector(7 downto 0);
        -- Inout
        SPI_SDA1 : inout std_logic_vector(No_ADX-1 downto 0);
    );

    multiples_adx : for i in 0 to No_ADX-1 generate
        for j in 0 to No_ADX-1 generate
            G_SENSOR_CS_N(i) <= sG_SENSOR_CS_N;
            ~spiControlExt(i) <= spiControl;
        end generate;
    end generate;

    u_FIFO : FIFO
    port map (
        clk => CLOCK_50,
        reset => dly_rst,
        X => soDATA_L(i),
        Z => soDATA_L
    );

    and generate;
    Ax <= soDATA_L(0)(47 downto 32);
    Ay <= soDATA_L(0)(31 downto 16);
    Az <= soDATA_L(0)(15 downto 0);
end rtl;

```

Promediador. Debido a la cantidad de muestras arrojadas por el módulo DEO_NANO_GSensor, se implementó este módulo que básicamente toma una cierta cantidad de muestras en un intervalo de tiempo, las promedia y este resultado se lleva al siguiente módulo, ver figura 26.

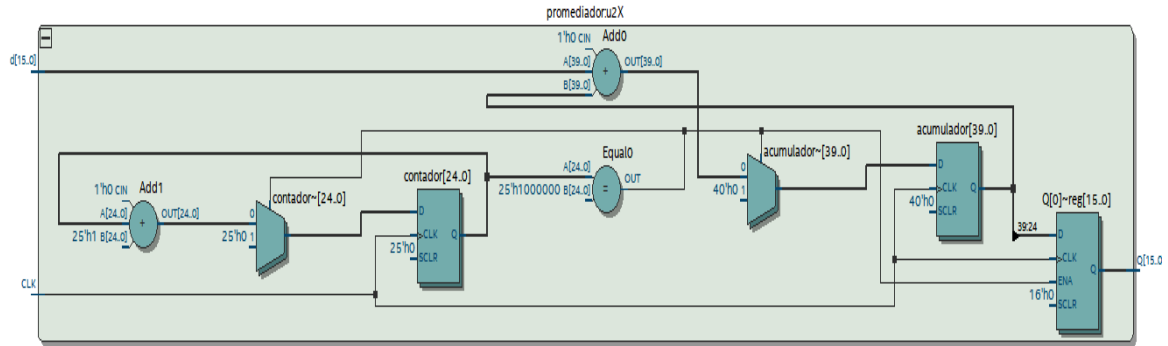


Figura. 26. RTL módulo promediador.

Código promediador

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_signed.all;

entity promediador is
generic(m: natural:= 24);
port( CLK: in std_logic; -- frecuencia de muestreo de aceleracion
      d: in std_logic_vector(15 downto 0);
      Q: out std_logic_vector(15 downto 0)
    );
end entity promediador;

architecture comportamental of promediador is
--zona de declaracion arquitectura
signal acumulador: std_logic_vector(15+m downto 0);
signal contador: std_logic_vector(m downto 0);
begin
-- zona concurrente

process(clk)
--zona de declaracion del process
begin
-- zona secuencial
if(dk="1" and clk'event) then
acumulador <= d + acumulador;
if (contador = 2**m) then --** exponente
q <= acumulador(15+m downto m); --division
contador <= (others => '0');
acumulador <= (others => '0');
else
contador <= contador + 1;
end if;
end if;

end process;

-- zona concurrente
end architecture comportamental;
```

Comparador. Como el sistema tiene tres servomotores, este módulo se encarga de activar cuál de ellos entrará en operación al recibir una señal del usuario, dependiendo de la posición en la que se encuentre en los planos x,y,z, tal cual como se observa en la figura 27, recibe las señales Ax, Ay y Az ya promediadas y como resultado genera tres señales M1, M2 y M3.

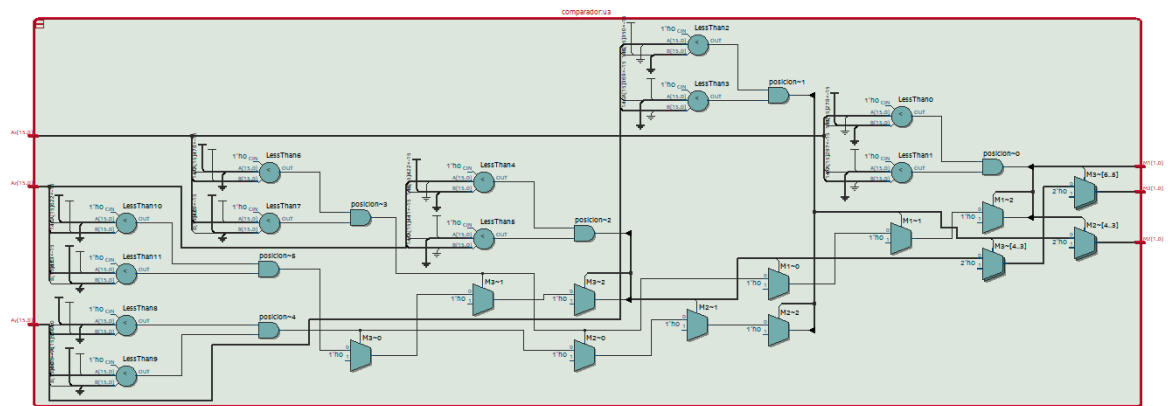


Figura. 27. RTL módulo comparador.

Código comparador

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity comparador is
    GENERIC(K:NATURAL:=255; P:NATURAL:=128);
    PORT (
        Ax :IN STD_LOGIC_VECTOR(15 downto 0);
        Ay :IN STD_LOGIC_VECTOR(15 downto 0);
        Az :IN STD_LOGIC_VECTOR(15 downto 0);
        M1 :OUT STD_LOGIC_VECTOR(1 downto 0);
        M2 :OUT STD_LOGIC_VECTOR(1 downto 0);
        M3 :OUT STD_LOGIC_VECTOR(1 downto 0)
    );
end comparador;

architecture Behavioral of comparador is
begin
    posicion:process(Ax, Ay, Az)
    begin
        if (Ax > Ay AND Ax > P) then
            M3 <= "00"; M2 <= "00"; M1 <= "01";
        elsif (Ay > Az AND Ay > P) then
            M3 <= "00"; M2 <= "01"; M1 <= "00";
        elsif (Az > Ax AND Az > P) then
            M3 <= "01"; M2 <= "00"; M1 <= "00";
        elsif (<Ax AND Ax < P) then
            M3 <= "00"; M2 <= "00"; M1 <= "10";
        elsif (<Ay AND Ay < P) then
            M3 <= "00"; M2 <= "10"; M1 <= "00";
        elsif (<Az AND Az < P) then
            M3 <= "10"; M2 <= "00"; M1 <= "00";
        else
            M3 <= "00"; M2 <= "00"; M1 <= "00"; -- este caso no esta controlado
        end if;
    end process posicion;
end Behavioral;
```

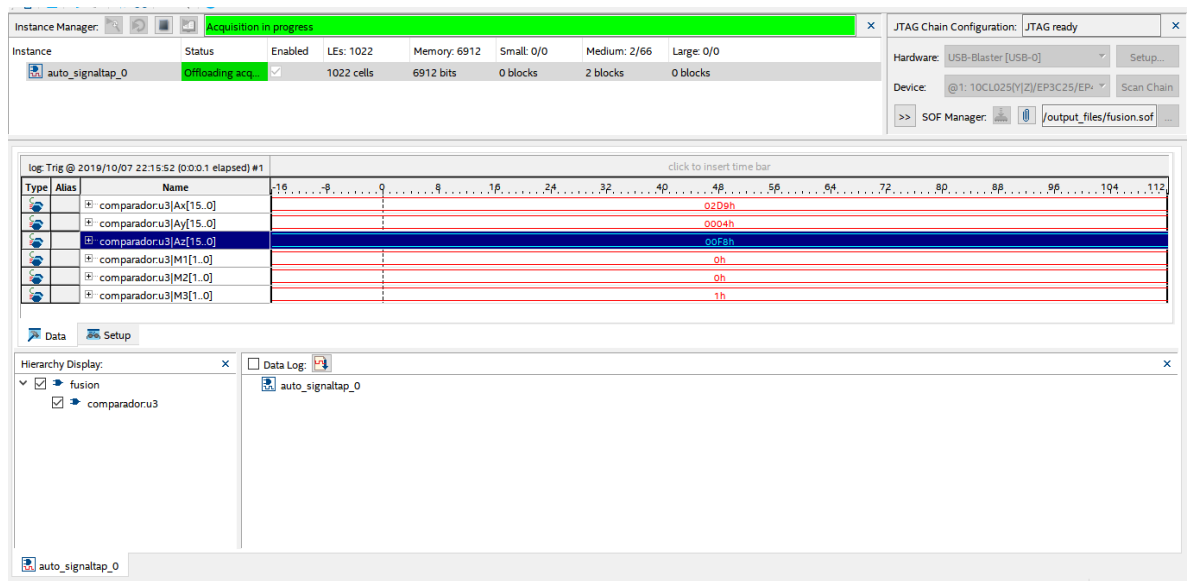


Figura. 28. Signal tap diseño estructural del manejo de datos del acelerómetro.

2.16. MÓDULO COMPARADOR FINAL

La figura 29 presenta el RTL del módulo creado para recibir tanto la salida del módulo UART como la salida del módulo del sensor acelerómetro.

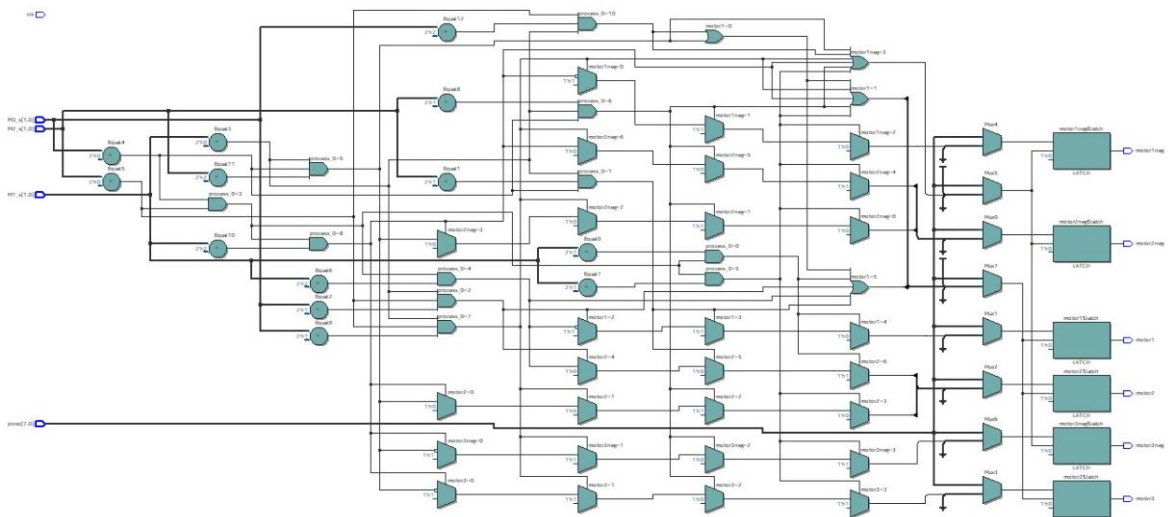


Figura. 29. RTL comparador final.

Codigo comparador final

```

library IEEE;
use IEEE.STD_LOGIC_3164.ALL;
use IEEE.std_logic_arith;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity acc_uart_pwm is
port(clickin std_logic);
    posic: in std_logic_vector(7 downto 0);
    M1_s: in STD_LOGIC_VECTOR(1 downto 0);
    M2_s: in STD_LOGIC_VECTOR(1 downto 0);
    M3_s: in STD_LOGIC_VECTOR(1 downto 0);
    motor1: out std_logic;
    motor2: out std_logic;
    motor3: out std_logic;
    motor3neg: out std_logic;
    motor2neg: out std_logic;
    motor3neg: out std_logic;
end acc_uart_pwm;

architecture Behavioral of acc_uart_pwm is
begin
    process(posic,M1_s,M2_s,M3_s)
    begin
        case posic is
            -- avance up
            when "0100001" =>
                if(M1_s = "00" AND M2_s = "00" AND M3_s = "01") then
                    motor2<='1'; motor1<='0'; motor3<='0';
                    --end if;

                    elsif(M3_s = "00" and M2_s = "01" and M1_s = "00") then
                        motor1<='1'; motor2<='0'; motor3<='0';
                        --end if;

                    elsif(M3_s = "01" and M2_s = "00" and M1_s = "00") then
                        motor2<='1'; motor1<='0'; motor3<='0';
                        --end if;

                    elsif(M3_s = "00" and M2_s = "00" and M1_s = "10") then
                        motor2<='1'; motor1<='0'; motor3<='0';
                        --end if;

                    elsif(M3_s = "00" and M2_s = "10" and M1_s = "00") then
                        motor1<='1'; motor2<='0'; motor3<='0';
                        --end if;

                    elsif(M3_s = "10" and M2_s = "00" and M1_s = "00") then
                        motor3<='1'; motor1<='0'; motor2<='0';
                        --end if;

                    -- avance derecha el mismo desplazamiento de la izquierda pero en sentido contrario
                    when "0100010" =>
                        if(M3_s = "00" AND M2_s = "00" AND M1_s = "01") then
                            motor3neg<='1'; motor1neg<='0'; motor2neg<='0';
                            --end if;

                            elsif(M3_s = "00" and M2_s = "01" and M1_s = "00") then
                                motor2neg<='1'; motor3neg<='0'; motor1neg<='0';
                                --end if;

                                elsif(M3_s = "01" and M2_s = "00" and M1_s = "00") then
                                    motor1neg<='1'; motor3neg<='0'; motor2neg<='0';
                                    --end if;

                                    elsif(M3_s = "00" and M2_s = "00" and M1_s = "10") then
                                        motor3neg<='1'; motor1neg<='0'; motor2neg<='0';
                                        --end if;

                                        elsif(M3_s = "00" and M2_s = "10" and M1_s = "00") then
                                            motor1neg<='1'; motor2neg<='0'; motor3neg<='0';
                                            --end if;

                                        elsif(M3_s = "10" and M2_s = "00" and M1_s = "00") then
                                            motor2neg<='1'; motor1neg<='0'; motor3neg<='0';
                                            --end if;

                                        -- avance down es el mismo up pero en sentido contrario
                                        when "0100101"
                                        =>
                                            if(M3_s = "00" AND M2_s = "00" AND M1_s = "01") then
                                                motor2neg<='1'; motor1neg<='0'; motor3neg<='0';
                                                --end if;

                                                elsif(M3_s = "00" and M2_s = "01" and M1_s = "00") then
                                                    motor1neg<='1'; motor2neg<='0'; motor3neg<='0';
                                                    --end if;

                                                    elsif(M3_s = "01" and M2_s = "00" and M1_s = "00") then
                                                        motor3neg<='1'; motor2neg<='0'; motor1neg<='0';
                                                        --end if;

                                                        elsif(M3_s = "00" and M2_s = "00" and M1_s = "10") then
                                                            motor2neg<='1'; motor1neg<='0'; motor3neg<='0';
                                                            --end if;

                                                            elsif(M3_s = "00" and M2_s = "10" and M1_s = "00") then
                                                                motor1neg<='1'; motor2neg<='0'; motor3neg<='0';
                                                                --end if;

                                                                elsif(M3_s = "10" and M2_s = "00" and M1_s = "00") then
                                                                    motor3neg<='1'; motor2neg<='0'; motor1neg<='0';
                                                                    --end if;

                                                                    WHEN OTHERS =>
                                                                        motor1<='0';
                                                                        motor2<='0';
                                                                        motor3<='0';
                                                                        motor1neg<='0';
                                                                        motor2neg<='0';
                                                                        motor3neg<='0';

                                                                    end case;

                                                                end process;

                                                                end Behavioral;

```

La figura 30 muestra el diseño estructural completo donde se incorporan todos los módulos creados para este fin.

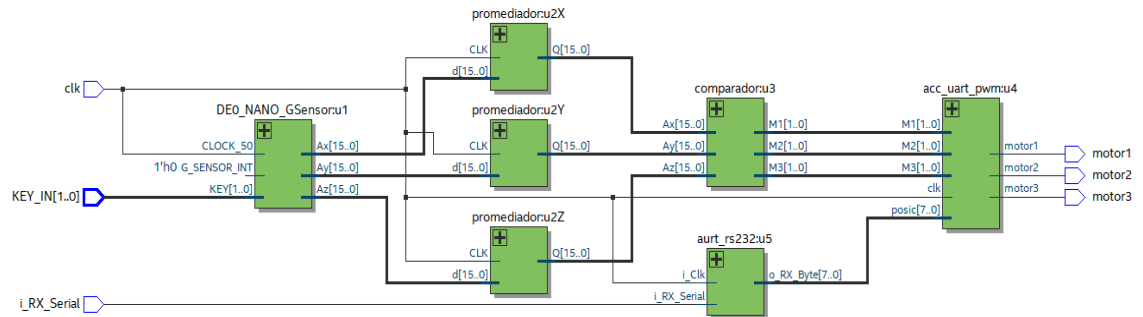


Figura. 30. RTL diseño estructural.

CAPÍTULO 3. RESULTADOS OBTENIDOS

Como resultado obtenido de este proyecto se puede resaltar:

- Un robot esférico compacto, con las dimensiones apropiadas para ser portado por cualquier persona, bajo consumo de energía, comparado con otros tipos de sistemas y mando remoto para su funcionamiento.

La figura 31 corresponde al diseño estructural final del proyecto en lo concerniente al lenguaje de descripción de hardware VHL. Se puede evidenciar las entradas y salidas del sistema, para el caso, la entrada *i_RX_Serial* corresponde a la entrada serial proveniente del módulo *Bluetooth*, la señal *clk* que corresponde al reloj incorporado cuya frecuencia es de 50 Mhz, una entrada *KEY_IN* que va a los pulsadores de la tarjeta de desarrollo y actúa como un *reset* del módulo del acelerómetro. Como salidas se tiene *m_1*, *m_2*, *m_3* que van directamente a los actuadores o servomotores. Las demás salidas corresponden al protocolo serial SPI que maneja el módulo del acelerómetro.

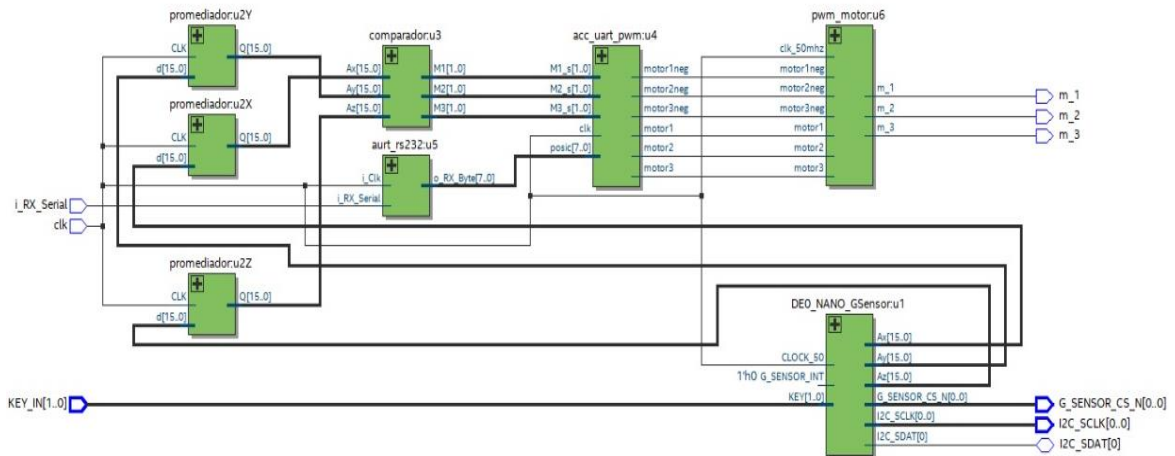


Figura. 31. Diseño estructural sistema completo.

Código de diseño estructural final

```

library IEEE;
use IEEE.std_logic_1164.all;
use work.spi_param.all;

entity fusion is
generic(m: natural:= 24; K:NATURAL:=500; P:NATURAL:=128; g_CLKS_PER_BIT : integer :=
1302; periodo: integer:=1000000); --velocidad de respuesta M= 24
port(clk: in std_logic);

KEY: IN in std_logic_vector(15 downto 0);
I2C_SDA7 : inout std_logic_vector(No_ADX-1 downto 0);
I2C_SCL6 : out std_logic_vector(No_ADX-1 downto 0);
G_SENSOR_CS_N : out std_logic_vector(No_ADX-1 downto 0);
I_RX_Serial: in std_logic;
m_1: out std_logic;
m_2: out std_logic;
m_3: out std_logic;
end fusion;

architecture structural of fusion is
component DIO_NANO_GSensor is
port
(
-- Input ports
CLOCK_50 : in std_logic;
KEY : in std_logic_vector(15 downto 0);
G_SENSOR_INT : in std_logic;
--G_SENSOR_INT : in std_logic_vector(No_ADX-1 downto 0);

-- Inout ports
I2C_SDA7 : inout std_logic_vector(No_ADX-1 downto 0);

-- Output ports
--spiControlExt : out std_logic_vector(No_ADX-1 downto 0);
Ax : out std_logic_vector(15 downto 0);
Ay : out std_logic_vector(15 downto 0);
Az : out std_logic_vector(15 downto 0);

--I2C_SCL6
I2C_SCL6 : out std_logic_vector(No_ADX-1 downto 0);
G_SENSOR_CS_N : out std_logic;
G_SENSOR_CS_N : out std_logic_vector(No_ADX-1 downto 0);
LED : out std_logic_vector(7 downto 0)
);
end component DIO_NANO_GSensor;

component promediador is
generic(m: natural:= 24);
port CLK: IN std_logic; -- frecuencia de muestreo de aceleracion
d: IN std_logic_vector(15 downto 0);
Q: out std_logic_vector(15 downto 0)
);
end component promediador;

m_3: out std_logic );
end component pwm_motor;

signal node_AX: std_logic_vector(15 downto 0);
signal node_AY: std_logic_vector(15 downto 0);
signal node_Az: std_logic_vector(15 downto 0);
signal node_AX_P: std_logic_vector(15 downto 0);
signal node_AY_P: std_logic_vector(15 downto 0);
signal node_Az_P: std_logic_vector(15 downto 0);
signal o_RX_Byte_s: std_logic_vector(7 downto 0);
signal M1_p: STD_LOGIC_VECTOR(1 downto 0);
signal M2_p: STD_LOGIC_VECTOR(1 downto 0);
signal M3_p: STD_LOGIC_VECTOR(1 downto 0);
signal motor1_p: std_logic;
signal motor2_p: std_logic;
signal motor3_p: std_logic;
signal motor1neg_p: std_logic;
signal motor2neg_p: std_logic;
signal motor3neg_p: std_logic;
begin
u1: component DIO_NANO_GSensor
port map
(
-- Input ports
CLOCK_50 => clk,
KEY => KEY_IN,
G_SENSOR_INT => '0',
--G_SENSOR_INT : in std_logic_vector(No_ADX-1 downto 0);

-- Inout ports
I2C_SDA7 => I2C_SDA7,

-- Output ports
--spiControlExt : out std_logic_vector(No_ADX-1 downto 0);
Ax => node_AX,
Ay => node_AY,
Az => node_Az,

--I2C_SCL6
I2C_SCL6 => I2C_SCL6,
G_SENSOR_CS_N : out std_logic;
G_SENSOR_CS_N => G_SENSOR_CS_N,
LED => open
);

u2X: component promediador
generic map(m => m)
port map( CLK => clk, -- frecuencia de muestreo de aceleracion
d => node_AX,
Q => node_AX_P
);

u2Y: component promediador
generic map(m => m)
port map( CLK => clk, -- frecuencia de muestreo de aceleracion
d => node_AY,
Q => node_AY_P
);

u2Z: component promediador
generic map(m => m)
port map( CLK => clk, -- frecuencia de muestreo de aceleracion
d => node_Az,
Q => node_Az_P
);

u3: component comparador
generic map (K=> K, P=>P)
PORT map ( Ax=> node_AX_P, Ay=> node_AY_P, Az=> node_Az_P, M1 => M1_p, M2 =>
M2_p, M3 => M3_p );
u4: component acc_uart_pwm
port map(clk=>clk,
posic => o_RX_Byte_s,
M1_s => M1_p,
M2_s => M2_p,
M3_s => M3_p,
motor1 => motor1_p,
motor2 => motor2_p,
motor3 => motor3_p,
motor1neg => motor1neg_p,
motor2neg => motor2neg_p,
motor3neg => motor3neg_p
);

u5: component aurt_r1232
generic map (
g_CLKS_PER_BIT => g_CLKS_PER_BIT
);
port map(
i_Clk => clk,
i_RX_Serial => i_RX_Serial,
--o_RX_DV => o_RX_DV,
o_RX_Byte => o_RX_Byte_s
);

u6: component pwm_motor
generic map(periodo => periodo)
port map(clk_50mhz=>clk,
--reset => reset,
motor1=>motor1_p,
motor2=>motor2_p,
motor3=>motor3_p,
motor1neg=>motor1neg_p,
motor2neg=>motor2neg_p,
motor3neg=>motor3neg_p,
m_1=>m_1,
m_2=>m_2,
m_3=>m_3
);
end structural;

```

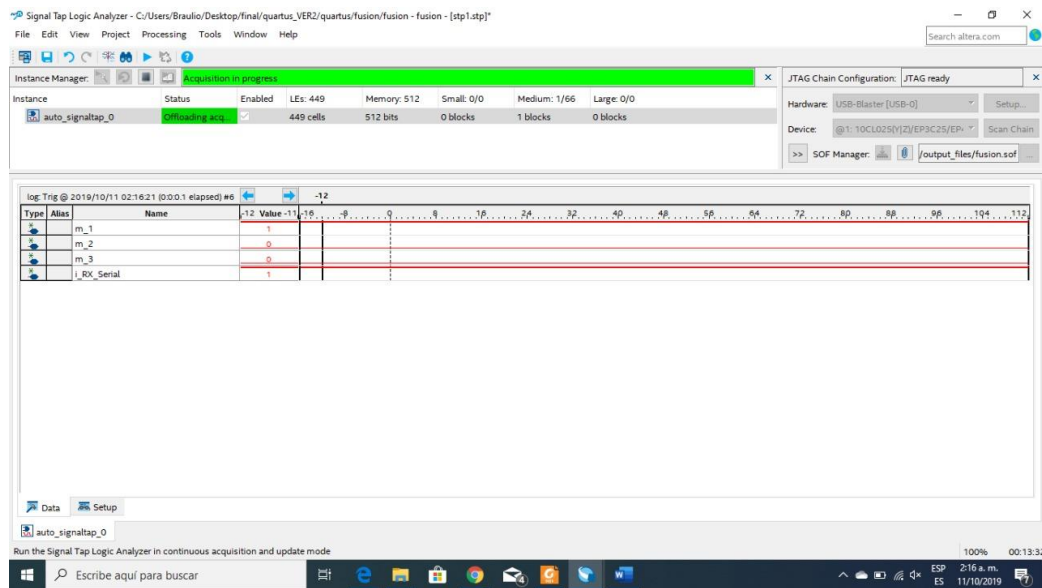


Figura. 32. Signal Tap del sistema completo.



Figura. 33. Cambio en engranaje servomotor Mg996r.

Ya que el servomotor Mg996r viene diseñado para 180° , fue necesario realizar adecuaciones para que éste operara de manera continua, para esto fue necesario abrir el servomotor como se aprecia en la figura 33, para ello se eliminó el tope que viene en el engranaje y el potenciómetro de control desanclarlo del mismo y calibrarlo manualmente, con esto se obtuvo la rotación continua y el control de avance y retroceso en el servo.

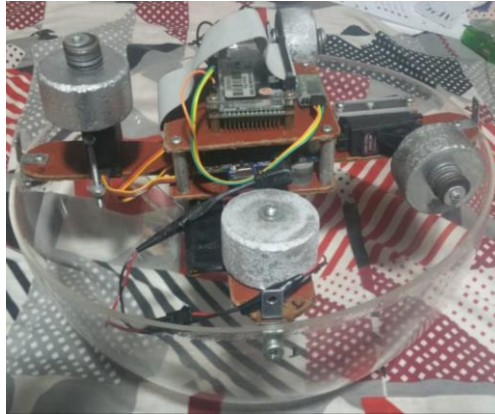
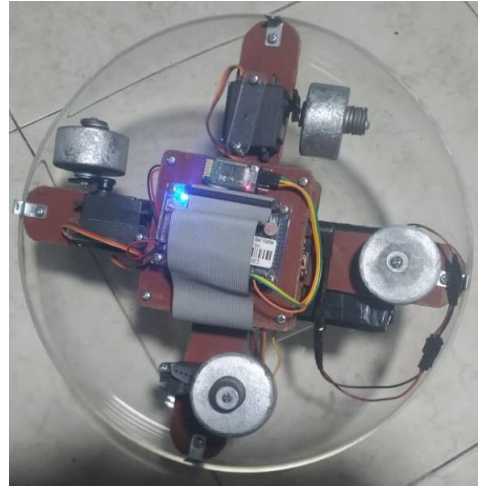
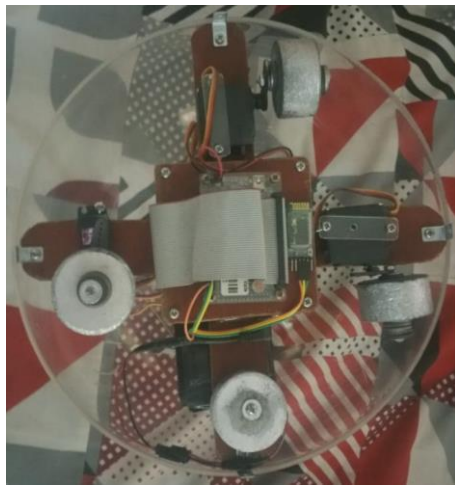


Figura. 34. Diseño estructura interna de hardware.



Figura. 35. Diseño hardware.

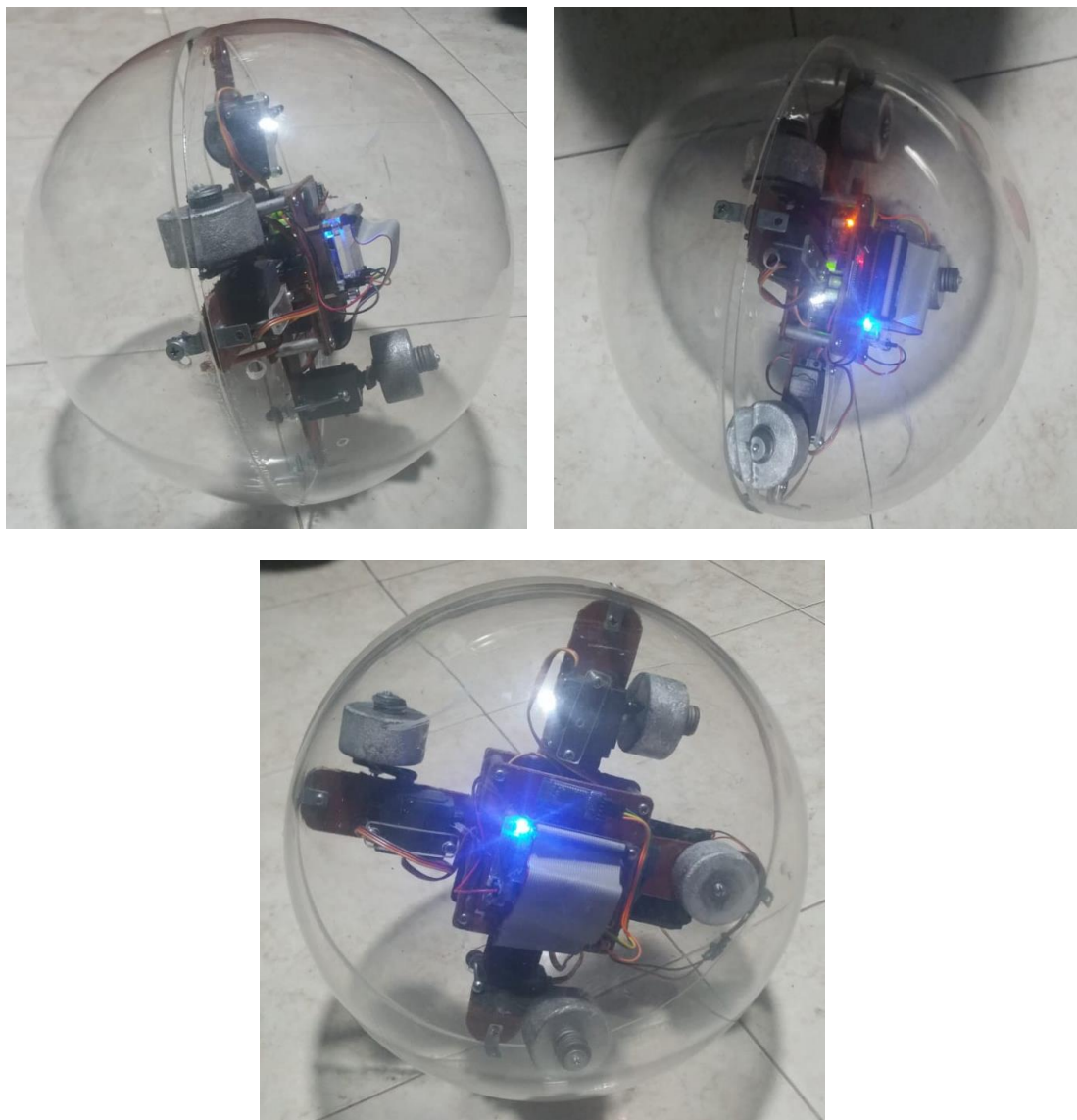


Figura. 36. Diseño hardware finalizado.

Tabla N° 3. Resultados obtenidos.

PESO MASAS SERVOS	130grs aprox
PESO TOTAL	1790grs aprox
VOLTAJE	7,4Vdc
CORRIENTE stand by	170mA
CORRIENTE en Ope.	300mA
POTENCIA	1,258VA – 2,22VA aprox
DIÁMETRO	30cm

CAPÍTULO 4. CONCLUSIONES

- Se logra diseñar un robot esférico electrónico con bajo consumo de energía comparado con otros tipos de sistemas y un mando remoto para su comunicación, a través, de un módulo *Bluetooth*
- Se diseñó y se implementó un robot esférico, se tomaron en cuenta varios diseños estructurales y se optó por el que menor peso le introdujera al sistema, su movimiento se basó en masas inerciales ancladas a cada uno de sus servomotores.
- Se logró acoplar el sistema de tal manera que cada uno de sus dispositivos y actuadores realicen su respectiva función, para una correcta operación del robot
- No se alcanza un funcionamiento adecuado del sistema debido a que requiere de un modelamiento en cuanto a las masas inerciales y el balanceo de las mismas.

BIBLIOGRAFÍA

- [1] <https://www.fabulantes.com/2013/02/rur-karel-capek/>
- [2] [\[2\] https://www.xataka.com/robotica-e-ia/rosphere-robot-espanol-de-pura-cepa-agricultor-con-forma-de-esfera.](https://www.xataka.com/robotica-e-ia/rosphere-robot-espanol-de-pura-cepa-agricultor-con-forma-de-esfera)
- [3] [\[3\] https://www.xataka.com/robotica-e-ia/guardbot-es-un-robot-rodante-que-se-atreve-con-tierra-y-mar](https://www.xataka.com/robotica-e-ia/guardbot-es-un-robot-rodante-que-se-atreve-con-tierra-y-mar)
- [4] <https://howbb8works.com/es/>
- [5] <https://laughingsquid.com/morphex-mkii-a-six-legged-robot-that-transforms-into-a-ball/>
- [6] <https://www.eltiempo.com/archivo/documento/MAM-656439>
- [7] <http://axxon.com.ar/not/166/c-1660126.htm>
- <https://psicologiaymente.net/miscelanea/tipos-de-investigacion>
- Hernández, R., Fernández, C., y Baptista, M.P. (2010) Metodología de la Investigación (5ª Ed.). México: McGraw Hill Educación.
- <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=67&No=1028&PartNo=1>
- <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=593>
- <http://blog.electricbricks.com/2010/03/pwm-control-de-velocidad-en-motores-electricos/>
- <http://blog.electricbricks.com/2010/07/sistemas-holonomicos/>
- http://www.academia.edu/25165599/Design_and_analysis_of_a_spherical_mobile_robot

- https://www.researchgate.net/publication/3632854_Motion_control_of_a_spherical_mobile_robot
- <http://www.aie.cl/files/file/comites/ca/abc/actuadores.pdf>