INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Trabajo terminal

“Robot para sistema distribuido multiagente”

TT2-2024-B173

Presenta

Bahena Brito Israel Benjamín

Directores

Dr. Gelacio Castillo Cabrera

Dr. Genaro Juárez Martínez

INSTITUTO POLITÉCNICO NACIONAL

ESCOM

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

TT2 No. 2024-B173

Documento técnico

Robot para sistema distribuido multiagente

Presenta

Bahena Brito Israel Benjamín

Directores

Dr. Gelacio Castillo Cabrera

Dr. Genaro Juárez Martínez

Resumen

Este proyecto contempla el diseño e implementación del principio de operación de un sistema multiagente basado en robots que interactúan. El sistema está definido por el conjunto de robots y las reglas de operación. Las reglas de operación consisten en: ejecución de subrutinas de movimiento, identificación de obstáculos fijos, comunicación con otros robots por medio del protocolo Wifi ESP NOW y conjunto de instrucciones de la arquitectura en el FPGA. El sistema será probado en un escenario controlado por obstáculos. El principio de operación será probado por dos agentes, considerando que este mismo principio de operación se puede escalar a un número mayor de unidades.

Palabras clave: sistema distribuido multi agente, robótica móvil, sistemas autónomos.

CARTA RESPONSIVA

Que otorga visto bueno y avala la conclusión de documentación del Trabajo Terminal bajo los lineamientos establecidos por la Comisión Académica de Trabajos Terminales (CATT)

CDMX, a 15 de Diciembre de 2024

M. EN C. ANDRÉS ORTIGOZA CAMPOS
PRESIDENTE DE LA COMISIÓN ACADÉMICA DE TRABAJOS TERMINALES
P R E S E N T E

EN ATENCIÓN A:
M. EN A. N. MARÍA MAGDALENA SALDÍVAR ALMOREJO
SECRETARIA EJECUTIVA

Por medio de la presente, se informa que el Trabajo Terminal Núm. 2024-B173

Que lleva por Título: Robot para sistema distribuido multiagente

Fue concluido satisfactoriamente por:

Bahena Brito Israel Benjamín

Se avala que la documentación entregada mediante discos en formato DVD fue **revisada de manera precisa y exhaustiva** con el propósito de asegurar que los avances desarrollados bajo la supervisión de quien o quienes suscriben, hayan cumplido con lo planteado en el protocolo original, así como en lo establecido por el Documento Rector de Operación y Evaluación para los Trabajos Terminales de la ESCOM.

ATENTAMENTE
"LA TÉCNICA AL SERVICIO DE LA PATRIA"

Gelacio Castillo Cabrera
(Nombre completo y Firma)
Director/a

Ignacio Juárez Martínez
(Nombre completo y Firma)
Director/a

ADVERTENCIA

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y, por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan."

La aplicación no convenida exime a la escuela de su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52000.

CARTA DE AUTORIZACIÓN DE DIFUSIÓN EN EL REPOSITORIO INSTITUCIONAL DE TESIS

I. IDENTIFICACIÓN DEL AUTOR E INVESTIGACIÓN

Nombre del autor (es) *Apellidos, Nombre(s)*	Bahena Brito Israel Benjamín
Correo electrónico	ibahenab1300@alumno.ipn.mx

Unidad Académica
Escuela Superior de Cómputo
Nivel Educativo
Programa Académico
Ingeniería en sistemas Computacionales
Título de la investigación
Robot para sistema distribuido multiagente

Modalidad de la investigación	
Tesis	(☐)
Seminario de titulación *Tesina o trabajo final del seminario cursado*	(☐)
Memoria de experiencia profesional *Informe escrito de las actividades profesionales realizadas*	(☐)
Proyecto de investigación *Informe técnico de la propuesta o desarrollo de un nuevo material, equipo, prototipo, proceso o sistema*	(☑)
Curricular *Informe de cursos o actividades correspondientes a esta modalidad de titulación*	(☐)
Práctica profesional *Informe escrito avalado por la empresa u organismo donde se desarrolló la práctica*	(☐)
Programa especial *Informe de asignaturas tecnológicas propias de la rama cursada previamente*	(☐)
Otro *Resumen o versión alterna autorizada por el autor*	(☐)

II. ESPECIFICACIONES DEL DOCUMENTO (FORMATO DIGITAL)

Formato del archivo "versión pública de la investigación"	PDF (☑)
Número de páginas *Incluyendo anexos. Excluyendo portada.*	_____ páginas
Formato del archivo "documentos de soporte"	PDF (☑)
Número de páginas *Incluye: autorización de difusión y formatos institucionales*	_____ páginas

CARTA DE AUTORIZACIÓN DE DIFUSIÓN EN EL REPOSITORIO INSTITUCIONAL DE TESIS

III. AUTORIZACIÓN DE DIFUSIÓN DE LA INVESTIGACIÓN

El (los) que suscribe(n) Bahena Brito Israel Benjamín, con fundamento en los artículos 21 y 27 de la Ley Federal del Derecho de Autor; así como lo dispuesto en los artículos 5, 7 fracción IV, 9 y 11, fracción XXI de la Ley General en materia de Humanidades, Ciencias, Tecnologías e Innovación, como titular(es) de los derechos moral y patrimonial de la investigación titulada: Bubat para sistema distribuido multiagente, consiento (consentimos) al Instituto Politécnico Nacional (IPN), a través de la ESCOM, así como a la Dirección de Bibliotecas y Publicaciones (DBP), para que publiquen por tiempo indefinido de forma gratuita y no exclusiva, en el Repositorio Institucional de Tesis, los contenidos de mi investigación, autorizándolo a:

A. Almacenar la obra en el citado repositorio para efectos de seguridad y preservación;
B. Publicar la obra con fines académicos y de investigación;
C. Transformar o convertir la obra a distintos formatos o soportes electrónicos, para su accesibilidad, preservación y seguridad.

Al tenor de lo previamente expuesto, manifiesto que la obra es original y no infringe los derechos de la propiedad intelectual de otras personas o entidades, por lo que excluyo al Instituto Politécnico Nacional de todo tipo de responsabilidad, ya sea civil, administrativa o penal en virtud de su contenido.

La obra se pondrá a disposición en el Repositorio Institucional de Tesis (https://tesis.ipn.mx), respetando los derechos de autor considerando como requisito la licencia de uso Creative Commons BY-NC-ND 4.0 ⓒⓘⓢⓞ.

El autor entiende que, de no autorizar la publicación de su tesis, ésta será catalogada de manera interna en la biblioteca de la ESCOM, únicamente a nivel referencial para efectos de control. El autor podrá solicitar por escrito el retiro de su obra del Repositorio Institucional de Tesis, ante su Unidad Académica si lo considera conveniente; por su parte, el Instituto Politécnico Nacional podrá en cualquier momento retirar la publicación si existieran reclamaciones de terceros que afirmen ser titulares.

Así mismo, manifiesto bajo protesta de decir verdad que he leído y conozco el Aviso de Privacidad disponible en la página electrónica de la Dirección de Bibliotecas y Publicaciones, https://www.ipn.mx/bibliotecas-publicaciones/

Ciudad de México, a 17 de Diciembre de 2024

Bahena Brito Israel Benjamín
Nombre(s) completo y firma del autor(es)

Contenido

Tabla de figuras

Capítulo 1 Presentación del proyecto

1.1 Introducción

Los sistemas robóticos tienen una creciente importancia actual tanto en el sector académico como en el económico, siendo objeto de investigación y desarrollo. son más empleados en las industrias de logística o trasporte, realizando tareas repetitivas, de alto riesgo o de relativa simplicidad. sus ciclos de operación y recarga de energía permiten impulsar una producción más rápida y una solución automatizada a tareas repetitivas. Se hace un análisis de la presencia actual en el mercado de dispositivos de este tipo en empresas de talla mundial como Amazon y Walmart, y de su estudio en instituciones académicas como Stanford.

En este trabajo se presenta un sistema conformado por dos vehículos robóticos que se desplazan controlando su posicionamiento de manera automática dentro de un espacio controlado, identifican las dimensiones de este espacio, sus bordes rectos, las esquinas y obstáculos fijos, replicando capacidades individuales de coordinación requeridas en dinámicas de sistemas multiagente. Esto facilita el desarrollo de algoritmos escalables para aplicaciones en logística, transporte o exploración autónoma. Está pensado como una sucesión de diseño y construcción de módulos en Hardware reconfigurable y no reconfigurable que al ser integrados se construye un prototipo de este vehículo robótico que pueda ser replicado. Se hace uso de un dispositivo FPGA para el control central, y se hace uso de un microcontrolador ESP-32 como interfaz para la comunicación inalámbrica. Los módulos son construidos, probados e integrados siguiendo una metodología de componentes.

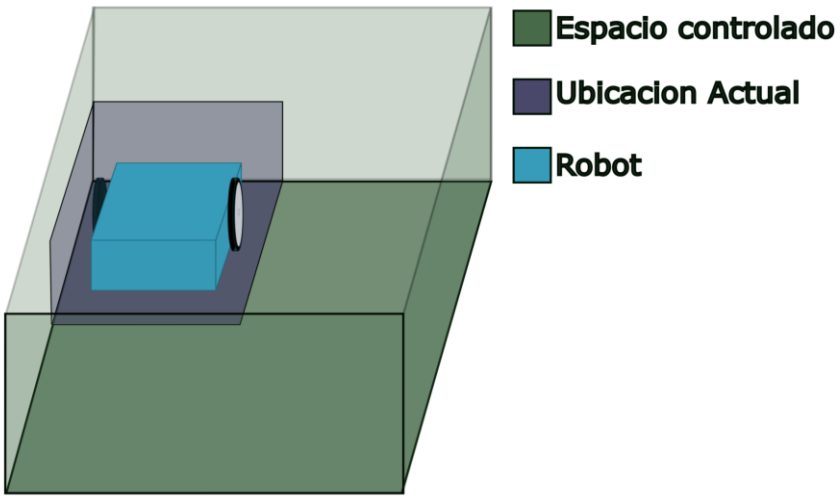


Ilustración 1 Planteamiento de Espacio Controlado

El robot puede identificar otros robots iguales a él en el medio controlado, no hay otras entidades en este espacio controlado, puede navegar a través de este sin chocar con otros robots, objetos fijos, o los límites del medio controlado.

Pueden delimitarse regiones cuadradas dentro del medio controlado, las llamaremos posiciones y serán etiquetadas con nombre, tipo de área ya sea de origen o destino, y coordenadas de su centro, todas las regiones tienen el mismo tamaño de 20cm x 20 cm.

Las tareas consisten en ocupar una posición, seguir una trayectoria o seguir a otro robot. Están definidas con un id, y contienen una posición que es el centro de un área o una posición del robot a seguir.

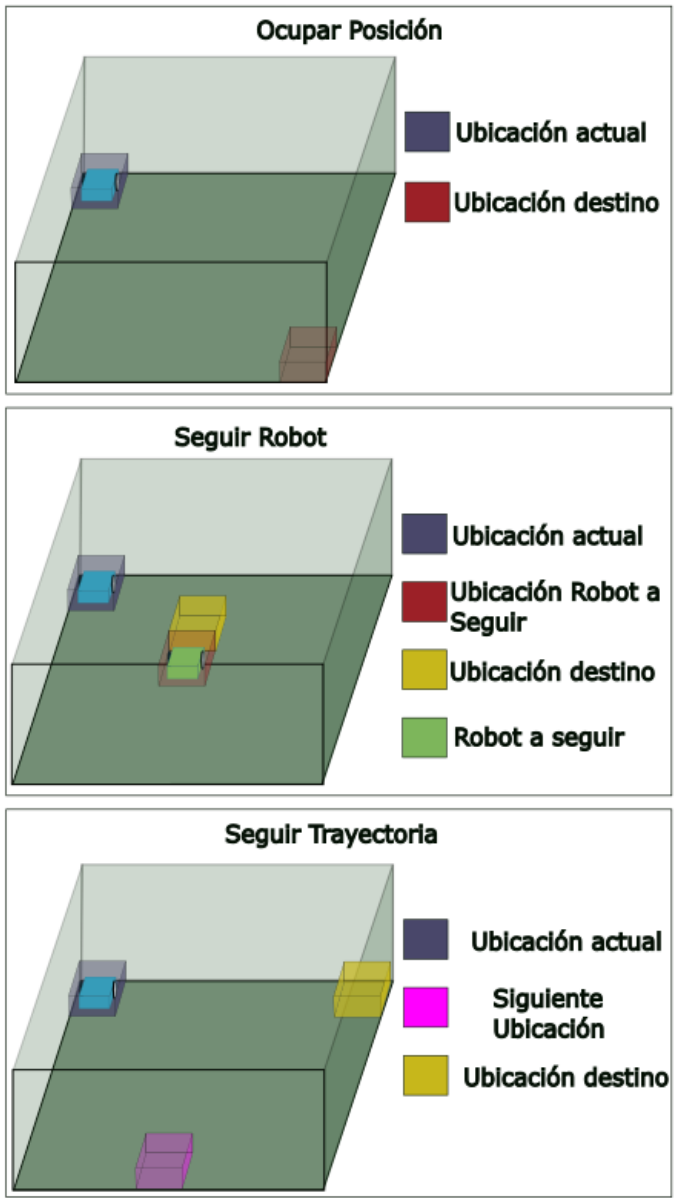


Ilustración 2 Lista de Tareas

Los robots pueden comunicarse entre ellos a través de mensajes que reciben y emiten de manera inalámbrica a través del protocolo ESP NOW. Cada mensaje está etiquetado con el id del robot que lo emite y un número de tarea, contiene un id de robot objetivo que puede estar o ser de objetivo para todos los robots que estén presentes, haciendo que solo sean procesados por las entidades para las que están destinados.

Al identificar un obstáculo fijo en el medio que se encuentra en un área específica el robot será capaz de evitar colisionar con él y seguir hacia la posición o área destino, exceptuando que se encuentre en dicha posición.

El robot puede ejecutar una lista de estas tareas.

1.2 Sistema Multiagente

Un sistema multiagente se define como una red de dispositivos que pueden resolver tareas particulares y especificadas, interactúan entre sí para resolver tareas de mayor complejidad cuya solución supera las capacidades individuales o el conocimiento de cada dispositivo. Estos dispositivos son llamados en el estado del arte, solucionadores de problemas o agentes, son autónomos y pueden ser homogéneos o heterogéneos, esto último significa que pueden estar especializados en solucionar tareas similares o totalmente diferentes. Otras características de los sistemas multiagente son las siguientes: cada agente posee información incompleta acerca de las capacidades de otros agentes, no hay un sistema global de control, los datos son descentralizados y los sistemas de cómputo son asíncronos, lo cual significa que, si bien son interdependientes, también son autónomos y resuelven sus tareas de forma inteligente [1]. Los sistemas multiagente tienen aplicaciones en distintos campos de la tecnología, como ejemplos se pueden citar redes de equipos de cómputo distribuidas, cajeros automáticos, sistemas de robots, entre muchos otros.

1.3 Sistemas Autónomos

Es un sistema que cambia su comportamiento en respuesta a eventos no anticipados, es capaz de operar en un ambiente desconocido y con circunstancias desconocidas sin intervención humana, más aún, puede tomar decisiones e implementar tareas, en el suelo, en el agua, en el aire. Durante su operación es suministrado de energía por una fuente interna [2]

1.3.1 Sistemas autónomos móviles

Consisten en una parte mecánica propiciando la movilidad con hardware y software, necesario para alcanzar el nivel deseado de autonomía [2].

1.4 Justificación

La implementación de un sistema multiagente, basado en robots, como la que se presenta en este proyecto está pensada o proyectada para ser utilizada como entrenamiento en instituciones académicas, ya que son de interés en aplicaciones industriales. Esto último es la razón principal de proponer un sistema multiagente como proyecto de entrenamiento.

1.5 Objetivos

1.5.1 Objetivo principal

Desarrollar un robot con el principio de operación multiagente que le permita ejecutar rutinas autónomas y además interactuar con otros robots, construidos de la misma forma, recibiendo y

emitiendo comandos u órdenes simples para ejecutar acciones especificadas en la construcción del prototipo.

1.5.2 Objetivos específicos

1. Diseño e implementación de la arquitectura sobre un dispositivo de lógica reconfigurable FPGA MACHX02-7000ZE. En la arquitectura se incluyen el set de instrucciones para ejecutar subrutinas y controlar movimiento.
2. Configuración de sensores de control: de objetos, distancia, codificadores.
3. Configuración del módulo de comunicación por Wifi.

1.6 Tecnología de procesadores dedicados

Los procesadores de uso general contienen un set de instrucciones amplio, enfocado a la implementación de códigos en software de alto nivel para enfrentar tareas de todo tipo, cuyas necesidades varían de manera inestimable. Por otro lado, conforme se hace más específico el marco de uso de un procesador, se le denominará dedicado, y la cantidad de instrucciones que contendrá se puede reducir únicamente al conjunto de las requeridas en la aplicación especificada, tendiendo a descartar lógica que no es útil y que por lo tanto potencialmente podría ser un gasto innecesario de recursos computacionales.

Tener una unidad de procesamiento dedicada al control de un sistema robótico especificado, que a nivel de hardware ofrece una interfaz basada en este reducido conjunto de instrucciones, ofrece una ventaja para aquel que posteriormente busque implementar un algoritmo en software haciendo uso de llamadas a instrucciones de control para sus sistemas motrices o de sensores, sin tener que involucrarse en el flujo de, de las señales de control de cada periférico.

1.6.1 Sistemas en hardware reconfigurables

En la actualidad existen compañías que fabrican procesadores de propósito general como los que se encuentran en los equipos personales. Se fabrican también procesadores avanzados para usos específicos como centros de datos, procesadores gráficos, enrutadores de red, entre otros. También se encuentran procesadores de propósito específico para dispositivos inteligentes en fábricas y en vehículos automotores, aunque son menos avanzados siguen siendo procesadores con un conjunto de instrucciones. Actualmente las compañías líderes más conocidas son Intel, AMD, ARM, Apple,

y Lattice Semiconductor. Hoy en día las compañías como Intel o AMD están desarrollando procesadores sobre dispositivos reconfigurables (FPGA). Esto se debe a la flexibilidad que presentan los dispositivos FPGA para realizar pruebas avanzadas de procesadores. Lattice Semiconductor por su parte se encuentra desarrollando dispositivos FPGA con tecnologías avanzadas, pero con menor capacidad. Esto quiere decir que son fabricados con transistores de longitudes de canal del orden de nanómetros (40nm) que les permiten integrar dispositivos de alta densidad y chips de tamaños muy pequeños, por ejemplo, el dispositivo ICE405KUP con dimensiones de 2.11mm x 2.54mm con encapsulado WLCSP de 30 pines de puertos de entrada salida (I/O) [3]. Esta tecnología resulta adecuada para desarrollar aplicaciones en robótica miniaturizada a un costo de chip de aproximadamente 10 dólares [4].

Capítulo 2 Estado del arte

Naturalmente la primera capacidad que un robot puede ofrecer para automatizar una tarea dentro de un espacio físico es la de trasladarse con el fin de mover una carga de un sitio a otro. Por esto la revolución de los robots autónomos móviles compuestos por múltiples agentes encuentra en los almacenes de los sectores industriales un nicho que ha servido para establecer una pauta sobre la eficiencia con la que se pueden gestionar las mercancías y materiales de manera automática y cimentar la presencia de estos sistemas en los procesos industriales de producción en la era de la información. A continuación, se revisarán algunos ejemplos de robots que se encuentran en la industria y en la academia.

2.1 MONA

Mona es un robot modular de código abierto, cuenta con un microcontrolador AVR de 8 bits (Atmega-328) que se programa a través de la plataforma de Arduino, dos micromotores DC con caja de reducción por engranes con una ratio de 280:1, ruedas de 28mm que le permiten moverse a 10cm/s. El principal sistema de sensores consta con 5 sensores de proximidad por infrarrojo de corto rango, cada uno localizado a una distancia angular de 35 grados [5].

Ilustración 3 Múltiples Robots MONA [5].

2.2 MiR1350 Mobile Industrial Robots

El robot MiR1350 es autónomo y colaborativo, maniobra de manera segura con tecnología de escaneo láser, proporcionando un campo de visión de 360 grados. Tiene dimensiones de 135 cm x 92 cm x 32 cm de altura, un peso de 231 kg y una capacidad de carga de 1350 kg y una velocidad máxima de 1.2 m/s, se pueden adaptar varias aplicaciones para las que se le monta encima un módulo respectivo [6].

Ilustración 4 Robot MiR1350 [6].

2.3 MiR1200 Pallet Jack

El Robot MIR1200 Pallet Jack es una unidad móvil autónoma que se asemeja a los montacargas comunes a diferencia de que no tiene un espacio para un conductor y funciona de manera completamente eléctrica, su objetivo es el de entregar de palés de hasta 1200 kg a 1,5 m/s. Consta con un sistema de percepción basado en inteligencia artificial lo que usa para navegar con seguridad alrededor de personas y otros obstáculos, constituyendo una alternativa segura a las carretillas elevadoras y montacargas tradicionales. Tiene como dimensiones 80 cm de largo, 58cm de ancho y 30 cm de alto, con un peso total de 97kg [7].

Ilustración 5 Robot MiR1200 Pallet Jack [7].

2.4 Stretch Boston Dynamics

Stretch es un robot móvil autónomo con capacidades de manejo de paquetes, pallets y cajas, hace funciones de carga pesada, descarga autónoma de tráileres y contenedores en piso y viaja de un sitio a otro con movilidad libre, Su peso es de 1300kg y puede manejar una carga máxima de 23 Kg. Su base es del tamaño de una tarima con la intensión de que pueda maniobrar con facilidad. Puede haber un teleoperador desde una consola, con la que también se puede configurar, supervisar el progreso de descarga y manejar múltiples Stretch desde ella [8].

Ilustración 6 Robot Stretch [8].

2.5 OTTO 1500 RockWell Automation

OTTO 1500 es un robot móvil autónomo de la marca RockWell Automation diseñado para trasladar cargas desde un sitio a otro. Tiene una capacidad de carga de 1900kg, su velocidad máxima es de 2m/s [9].

Ilustración 7 Robot OTTO 1500 [9].

2.6 Zooids

El robot Zooid controla su movimiento de manera independiente mediante un controlador PID basado en una máquina de estados. Al establecer un objetivo final, el robot primero gira para alinearse con la dirección correcta y, una vez orientado, acelera hasta alcanzar la velocidad preferida definida por el usuario. Al lograr esta velocidad, la mantiene mediante un control PID de orientación que asegura que se mantenga dirigido hacia el objetivo final. Si se asigna un nuevo objetivo incremental, el robot continúa moviéndose a la misma velocidad, pero el control PID ajusta su orientación para dirigirse hacia el nuevo objetivo intermedio. Sus Dimensiones son de 26 mm de diámetro y 21 mm de alto, con un peso aproximado de 21g, electrodos para detección táctil capacitiva, y cuenta con Ruedas que le permiten operar de forma eficiente y adaptable dentro de sistemas multiagente [10].

Ilustración 8 Múltiples robots Zooids [10].

2.7 Amazon Proteus

Es un modelo de robot autónomo creado por la compañía Amazon Robotics, la cual es una subsidiaria de la conocida compañía Amazon. Amazon Robotics trabaja en estrecha colaboración con el proveedor de procesadores Texas Instruments [11].

Mueve carros con paquetes hacia la zona de carga de camiones, puede levantar hasta 440 kilos, con un tamaño de 80 cm de ancho por 75 cm de largo por 20 cm de alto.

Ilustración 9 Robot Proteus [11].

2.8 Walmart Alphabot

Walmart es una tienda de autoservicio que entre sus investigaciones se propone el uso de robot autónomos móviles para apoyo de gestión de inventarios en comercio electrónico y autoservicio. El proveedor líder en el desarrollo de procesadores embebidos para Walmart es la compañía Axiomtek [12]. Uno de los procesadores en desarrollo para aplicaciones de robots autónomos es ROBOX300, cuyas características principales son: Intel I5-1147GE, su diferencia principal con procesadores de propósito general es que es un procesador para navegación y control autónomo [13].

Es un sistema de automatización de almacén configurado para pedidos en línea y clientes de entrega a domicilio. Lleva a cabo el procesamiento de las ordines de manera flexible, escalable y eficiente, usando una combinación de robots móviles contenedores modulares y software de propiedad para automatizar las operaciones del almacén. El robot se traslada en tres dimensiones recogiendo y entregando objetos en estaciones de preparación de órdenes [14].

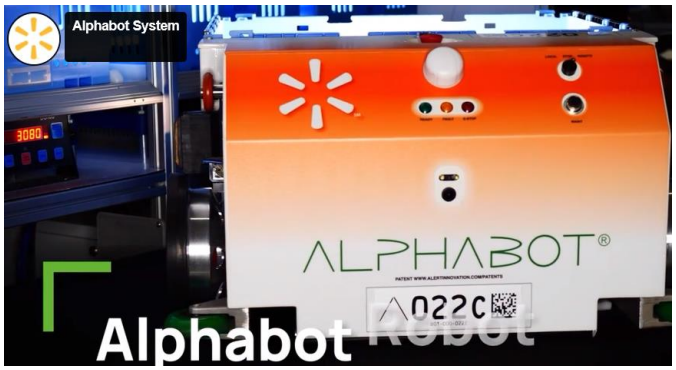


Ilustración 10 Robot Alphabot. [14].

Tabla comparativa

Vehículo Robot	Autonomía total	Autonomía parcial	Multiagente	Sensor láser
MONA	X	√	√	X
MiR1350	X	√	X	√
MiR1200	X	√	X	√
Stretch	X	√	X	√
Proteus	√	X	√	√
Alphabot	X	√	√	√
Robot Propio	X	√	√	√

Capítulo 3 Marco Conceptual

En este capítulo se describen los conceptos tecnológicos relacionados con el proyecto desarrollado y que son importantes, tanto en el planteamiento del problema como en su solución, a la vez que complementan el contexto del proyecto.

3.1 sistemas distribuidos

Un sistema Distribuido es una colección de computadoras independientes que aparecen ante los usuarios del sistema como una única computadora [15].

Los sistemas distribuidos deben su origen al concepto de sistemas de cómputo distribuido. Un sistema de cómputo distribuido consta de un conjunto de terminales conectadas en red. En general cada nodo ejecuta un sistema de procesamiento de información, la cual se coordina con un servidor central. Sin embargo, hay clasificaciones de los sistemas distribuidos las cuales son:

Clúster:

> "Conglomerado de computadoras construido mediante el uso de hardware común y que se comporta como si fuera una única computadora" [16].

Rejilla:

> "Es un paradigma del cómputo distribuido, frecuentemente usado para indicar una infraestructura de gestión de recursos distribuidos que se centra en el acceso coordinado a los recursos informáticos remotos. Estos recursos de cómputo son colectados desde múltiples localizaciones para alcanzar una meta común… Tiende a ser más heterogéneo y disperso" [16].

3.2 Wifi ESP NOW

ESP-NOW es una especie de protocolo de comunicación Wifi sin conexión definido por Espressif, los datos de aplicación son encapsulados en un marco específico de la marca, y transmitidos de un dispositivo a otro sin uso de conexión. Es ampliamente usado en controles [17].

3.3 odometría

La odometría es una técnica empleada para calcular la posición de un robot a partir del movimiento angular de las ruedas que es capturado principalmente por codificadores, dado que el error es parte de la medición, el error en la medición se acumula con el paso del tiempo. Por lo tanto, la posición debe ser actualizada recurrentemente por otros mecanismos, de otra manera el robot es incapaz de mantener una estimación de la posición que sea significativa a largo plazo

Algunas de las fuentes de error en el error por odometría son la falta de resolución en la medición, desalineación de las ruedas con sus ejes, incertidumbres en la rueda como deformidades, y variaciones en el punto de contacto de la rueda. Algunos errores son determinísticos y pueden ser eliminados calibrando el sistema, sin embargo, hay otros clasificados como errores de rango, error de giro, error de barrido, que son no determinísticos y que no pueden ser estimados y eliminados de manera lineal.

3.4 Medición de distancia con laser

Una de las maneras que existen hoy en día para medir una distancia a la es la de emplear un sensor de distancia de ondas electromagnéticas, o sonoras, ya sea contando el tiempo que toma en viajar o cuantificando la distorsión con la que regresan las ondas hacia el punto donde empieza la medición. Aplicaciones para las que deben detectarse en tiempo real la presencia de cuerpos con los que pueda haber una colisión, requieren tener una manera de registrar dichos datos con precisión y sobre todo con rapidez. Por ejemplo, aplicaciones de navegación autónoma; para identificar y evitar obstáculos en tiempo real, Topografía; para caracterizar un entorno o quizá un inmueble y modelarlo en un software que permita proyectar en tres dimensiones los datos caracterizados, en robótica; para conocer el estado presente de distancias dentro del sistema en que opera el robot, así como en el control industrial.

Con un emisor láser se emite un haz hacia adelante que idealmente choca con una superficie ubicada dentro de su rango de funcionamiento, esto regresa una porción del haz hacia un receptor de láser que se registra el regreso de la luz. Usando contadores es posible cuantificar el tiempo que transcurre entre la emisión y la posterior recepción del haz, se cuenta la cantidad de pulsos de reloj digital que se presentan entre estos dos eventos. A esta técnica se le conoce como Tiempo de Vuelo traducido al inglés como Time of Flight (ToF) y nos referiremos a ella de esta manera.

El láser tiene muchas ventajas entre las que se encuentran, la de emitir un haz estrecho que permite focalizar con precisión y la capacidad de difusión del láser, que permite hacer una medición a pesar de que la superficie con la que choca el haz no este perfectamente perpendicular a la trayectoria del haz.

3.5 Relación peso de carga potencia

Basándonos en el tamaño de un sistema de carga surge una cuestión alrededor de un factor de proporcionalidad, este factor es inversamente proporcional a la masa del sistema, y directamente proporcional a la masa de la carga que puede transportar, lo llamaremos “eficiencia de carga”. Galileo modeló una ley llamada “ley cuadrático-cúbica” Implicando que “Un pequeño incremento en el volumen lleva a un incremento grande de peso”. La masa del sistema se ve directamente ligada a su tamaño, y por tanto afecta directamente también aumentando la energía que deberá almacenar y emplear solo para moverse a sí misma conforme aumenta su masa. Disponiendo de una porción menor de energía para la carga si se conserva la misma cantidad de energía total disponible. Una tendencia se hace evidente, conforme aumenta el peso de un sistema, es más difícil aumentar su eficiencia de carga, además de tener un impacto directo en sus costos de operación,

fabricación, mantenimiento, y riesgos de operación entre muchas otras cosas. Idealmente buscamos que la eficiencia de carga sea grande.

Un maravilloso ejemplo de cómo en la naturaleza existen animales con una eficiencia de carga alta es la hormiga tejedora, que puede cargar aproximadamente 100 veces su peso, estando colgada de cabeza [18].

Capítulo 4 Análisis

En este capítulo se aborda la descripción de la solución, a través de un análisis conceptual del comportamiento global esperado del sistema. El sistema está diseñado basado en un modelo modular. Cada módulo es un componente que realiza una tarea especificada. Las tareas quedan definidas mediante el conjunto de requerimientos que a continuación se describen.

1.- El principio de funcionamiento queda implementado y es suficiente con un sistema multiagente de 2 vehículos robots.

2.- Cada robot está implementado por un vehículo capaz de desplazarse en un plano horizontal.

3.- Cada robot consta de los componentes necesarios para participar en el sistema multiagente:

a). - Motores de corriente directa con codificadores.

b). - Puente H.

c). - Módulo de comunicación Wifi implementado con ESP NOW ESP32.

d). - Sensor de distancia láser implementado con ESP32.

e). - Arquitectura de procesamiento FPGA MACHX02.

4.- La arquitectura de procesamiento referida en el punto anterior está diseñada de forma modular:

a) Módulos en Hardware (dentro de la FPGA) para el control de motores, sensores, módulo de comunicación.

b) Núcleo que consta de un set de instrucciones simples. Constan de dos campos; código de operación y campo de parámetro.

c) Conjunto de rutinas que resuelven tareas especificadas.

d) Cada rutina se compone de acciones: Avanzar, Retroceder, Girar a la derecha, Girar a la izquierda, Detectar obstáculos, Enviar mensaje, Recibir Comandos, Detenerse.

e) Una rutina no tiene que integrar todas las acciones enumeradas.

f) Las rutinas se resuelven con el set de instrucciones de la arquitectura.

4.1 Requerimientos funcionales

A continuación, se describen los requerimientos Funcionales

Tabla 1 Requerimientos Funcionales

Identificador	Nombre	Descripción
RF01	Encendido	Al presionar el botón de encendido el robot enciende y activa todos sus componentes y los configura de inicio para empezar a procesar.
RF02	Apagado	Al estar prendido el robot y presionar el botón de apagado, este se apaga.
RF03	Configuración	El robot se carga con un programa de ejecución que describe su comportamiento. Cada módulo es programable por separado.
RF04	Reinicio	Al presionar el botón de reinicio el robot reinicia todos sus componentes.
RF05	Determinación de la orientación y el desplazamiento	A partir de odometría el robot puede dar una ubicación propia relativa a un punto de interés identificado en el medio.
RF06	Determinación de la distancia frontal hacia un obstáculo.	El robot usa los datos del sensor de distancia para calcular la distancia a la que está, de el obstáculo que hay enfrente.
RF07	Desplazamiento	El robot cuenta con motores y ruedas para moverse. Puede avanzar hacia enfrente a una en línea recta automáticamente.
RF08	Determinación de la velocidad angular de las ruedas.	El robot mide la rotación de los ejes de los motores y calcula las velocidades angulares de las ruedas.
RF09	Determinación de la velocidad.	El robot usa la velocidad angular de las ruedas, la orientación y el desplazamiento del plano de referencia local para calcular la velocidad a la que avanza el robot dentro del plano global.
RF10	Regulación de la velocidad de movimiento.	El robot puede avanzar a velocidad fija regulable.

RF11	Giro	El robot puede girar sobre su propio eje mientras no se desplace, por lo que para girar primero tendrá que detenerse y posteriormente girar, antes de avanzar nuevamente.
RF12	Parada	El robot se puede detener.
RF13	Determinación de posición.	El robot determina su posición y su orientación dentro del medio controlado y la almacena.
RF14	Actualización periódica de la posición.	El robot usa la posición para determinar su nueva posición cada vez que se mueve.
RF15	Identificación de obstáculos.	El robot puede identificar que hay un obstáculo entre él y el borde del área de operación.
RF16	Identificación de áreas.	El robot asocia regiones cuadradas dentro del área de operación y almacena sus coordenadas para definir un área, de origen o de destino.
RF17	Identificación de robot.	El robot identifica la presencia de otro robot porque obtiene su ubicación a través de un mensaje.
RF18	Definición de una trayectoria.	El robot toma dos coordenadas separadas dentro del área de operación y define su trayectoria como una línea recta entre ellas. no puede haber una coordenada asociada a un obstáculo que esté entre estos dos puntos.
RF19	Conexión inalámbrica.	El robot puede establecer una comunicación inalámbrica con la función de paso de mensajes.
RF20	Gestión de tareas.	El robot recibe tareas y las guarda en una lista y las ejecuta.

Descripción de los requerimientos funcionales

1. **ID:** RF01
    - **Nombre**: Encendido
    - **Descripción**: El operador al presionar el botón de encendido el robot enciende y activa todos sus componentes y los configura de inicio para empezar a procesar.
    - **Entradas:** señal de interruptor.
    - **Procesamiento:** Al encender se alimenta todo el robot y sus circuitos.
    - **Salidas**: Alimentación 12+V o 0V.
    - **Criterios de aceptación**: El robot se mantiene encendido hasta que se presiona el botón de nuevo o se termina la batería.
    - **Actores:** Operador.
2. **ID:** RF02.

- **Nombre:** Apagado.
- **Descripción:** Al presionar el botón de apagado, este se apaga.
- **Entradas:** señal de interruptor.
- **Procesamiento:** Al presionar el botón de encendido/apagado mientras el robot se encuentra encendido, se desactivan todos los módulos y el robot se apaga.
- **Salidas:** Alimentación 12+V/0V.
- **Criterios de aceptación:** El robot se mantiene encendido o apagado hasta que se presiona el botón o se termina la batería.
- **Actores:** Operador.

3. **ID:** RF03.
   - **Nombre:** Configuración.
   - **Descripción:** El robot se carga con un programa de ejecución que describe su comportamiento. Cada módulo es programable por separado.
   - **Entradas:** un archivo con extensión. "Jed" entra por el puerto USB de la tarjeta de desarrollo donde está la FPGA.
   - **Procesamiento:** La tarjeta de desarrollo se conecta a una computadora con el software de Lattice ice cube y se descarga el archivo con extensión ". jed" en su memoria de configuración.
   - **Salidas:** Estado de programación terminada.
   - **Criterios de aceptación:** La programación fue correcta y la FPGA está cargada con el archivo de configuración ". jed" y lista para operar.
   - **Actores:** Operador.

4. **ID:** RF04.
   - **Nombre:** Reinicio.
   - **Descripción:** Al presionar el botón de reinicio el robot reinicia todos sus componentes.
   - **Entradas:** Señal de botón de reinicio.
   - **Procesamiento:** Los componentes del sistema reciben una señal de reinicio y todo vuelve al estado inicial.
   - **Salidas:** estado de reinicio listo, Robot listo para empezar a operar.
   - **Criterios de aceptación:** el robot reinicia todo el sistema.
   - **Actores:** Operador.

5. **ID:** RF05.

   Nombre: Determinación de la orientación y el desplazamiento

   - **Descripción:** El robot usa odometría para calcular la orientación y el desplazamiento y la orientación del marco de referencia local dentro del.
   - **Entradas:** Señal de inicio de lectura, datos de medición registrados por el codificador, datos de medición registrados por sensor laser.
   - **Procesamiento:** Los datos se leen desde los sensores hacia un controlador de tipo máquina de estados dentro de la FPGA.
   - **Salidas:** Orientación, y desplazamiento desde el tiempo t.

- **Criterios de aceptación:** La orientación real del robot y la medida presentan menos de 20% de error.
- **Actores:** sistema de control de movimiento, codificador, sensor laser.

6. **ID:** RF06.
   - **Nombre:** Determinación de la distancia frontal hacia un obstáculo.
   - **Descripción:** El robot usa los datos del sensor de distancia para calcular la distancia a la que está, de el obstáculo que hay enfrente.
   - **Entradas:** La señal de iniciar medición de distancia frontal, señal de medición del sensor.
   - **Procesamiento:**
     - Se genera una señal que indica el inicio de medición de distancia.
     - Se envía al controlador del sensor de medición de distancia.
     - El controlador se conecta mediante el protocolo I2C al microcontrolador dedicado de la placa del sensor.
     - Envía la configuración de lectura al microcontrolador.
     - Recopila la lectura de la distancia.
     - Se calcula la distancia y se expresa en centímetros.
   - **Salidas:** Señal de control del sensor, datos de la medición de distancia con el obstáculo de enfrente.
   - **Criterios de aceptación:** Los datos de la medición y la distancia real varían máximo con un 5% de error.
   - **Actores:** Sistema de identificación de obstáculos, Sensor de distancia.

7. **ID:** RF07.
   - **Nombre:** Desplazarse.
   - **Descripción:** El robot cuenta con actuadores y ruedas para moverse. Puede avanzar hacia enfrente a una en línea recta automáticamente.
   - **Entradas:** señal de velocidad de movimiento esperada.
   - **Procesamiento:** La señal de velocidad de movimiento se convierte en una señal de PWM y señales de control que se envían a un controlador de tipo Puente H al que están conectados los motores.
   - **Salidas:** Señal de PWM, señales de control de controlador Puente H dirección, activación de cada motor.
   - **Criterios de aceptación:** El robot avanza en línea recta.
   - **Actores:** Sistema de control de movimiento, puente H para control de motores, motores.
   - **Requerimientos de los que depende:** RF04.

8. **ID:** RF08.
   - **Nombre:** Determinación de la velocidad angular de las ruedas.
   - **Descripción:** El robot mide la rotación de los ejes de los motores y se calculan las velocidades angulares de las ruedas.

- **Entradas:** Señal de sensores de rotación de las ruedas.
- **Procesamiento:** Se calculan las velocidades angulares de las ruedas.
- **Salidas:** Velocidad angular de las ruedas.
- **Criterios de aceptación:** la velocidad medida está máximo 10% de la velocidad real.
- **Actores:** Sistema de control móvil.

9. **ID:** RF09.
    - **Nombre:** Determinación de la velocidad.
    - **Descripción:** El robot usa la velocidad angular de las ruedas, la orientación y el desplazamiento del plano de referencia local para calcular la velocidad a la que avanza el robot dentro del plano global.
    - **Entradas:** velocidad angular, orientación, desplazamiento, posición global.
    - **Procesamiento:**
    - Se conjuntan todos los valores de velocidad angular, de orientación, desplazamiento, posición global.
    - Se transforma cada uno a velocidad de movimiento.
    - Se hace una función aproximada sumando los valores anteriores.
    - Se determina la velocidad del robot dentro del plano global.
    - **Salidas:** Velocidad de movimiento, señal de velocidad lista para ser leída.
    - **Criterios de aceptación:** La velocidad está integrada así que su error debe ser de menos del 5%.
    - **Actores:** Sistema de control de movimiento, Sistema de control de velocidad.

10. **ID:** RF10.
    - **Nombre:** Regular velocidad de movimiento.
    - **Descripción:** El robot puede avanzar a velocidad fija regulable.
    - **Entradas:** Velocidad de movimiento esperada, velocidad de movimiento calculada.
    - **Procesamiento:** Con el retorno de la medición de velocidad hacia el sistema de control móvil se ajusta la velocidad real del robot para que se acerque a la velocidad esperada.
    - **Salidas:** Velocidad de movimiento deseada, ajustada con error.
    - **Criterios de aceptación:** la velocidad de movimiento tiene un máximo de 5% de error.
    - **Actores:** Sistema de control móvil, sistema de medición de velocidad.

11. **ID:** RF11.
    - **Nombre:** Girar.
    - **Descripción:** El robot puede girar sobre su propio eje mientras no se desplace, por lo que para girar primero tendrá que detenerse y posteriormente girar, antes de avanzar nuevamente.
    - **Entradas:** Señal de inicio de giro, Velocidad, posición, orientación del robot, coordenada del siguiente punto en la trayectoria,

- **Procesamiento:** El robot cuando ha recibido la señal de girar transforma su posición al girar sus ruedas en sentidos contrarios hasta estar alineado con el siguiente punto en la trayectoria.
- **Salidas:** Velocidad de movimiento deseada, ajustada con error.
- **Criterios de aceptación:** El robot se alinea con el siguiente punto con un %5 de error.
- **Actores:** Sistema de control móvil, sistema de medición de velocidad.

12. **ID:** RF12.
    - **Nombre:** Detenerse.
    - **Descripción:** El robot se puede detener.
    - **Entradas:** Señal de detenerse.
    - **Procesamiento:** El robot para en una coordenada específica a la que ya ha llegado.
    - **Salidas:** Velocidad de movimiento cero.
    - **Criterios de aceptación:** la posición y la orientación dejan de variar.
    - **Actores:** Sistema de control móvil.
    - **Requerimientos de los que depende:** RF11.

13. **ID:** RF13.
    - **Nombre:** Determinación de posición y orientación.
    - **Descripción:** El robot determina su posición y su orientación dentro del medio controlado y la almacena.
    - **Entradas:** orientación, posición inicial, perímetro del área de operación, desplazamiento, medición de distancia hacia obstáculo.
    - **Procesamiento:**
        - Toma posición inicial determinada al medir el perímetro por primera vez.
        - Agrega la velocidad, el desplazamiento, la orientación
        - Ajusta su posición y orientación absoluta en el marco de referencia global.
    - **Salidas:** Posición y orientación.
    - **Criterios de aceptación:** La posición y la orientación tiene un error máximo de 5%.
    - **Actores:** Sistema de posicionamiento y orientación, sistema de medición de obstáculos.

14. **ID:** RF14.
    - **Nombre:** Actualización periódica de la posición.
    - **Descripción:** El robot usa la posición para determinar su nueva posición cada vez que se mueve.
    - **Entradas:** Velocidad angular de las ruedas, datos de los sensores de distancia, posición y orientación.
    - **Procesamiento:** Periódicamente se corrige la posición absoluta usando el sistema de medición de obstáculo para ajustarla.
    - **Salidas:** Posición y orientación absolutas ajustadas.

- **Criterios de aceptación:** La posición y la orientación tiene un error máximo de 5%.
- **Actores:** Sistema de posicionamiento y orientación, sistema de medición de obstáculos.

15. **ID:** RF15.
    - **Nombre:** Identificación de obstáculos.
    - **Descripción:** El robot puede identificar que hay un obstáculo entre él y el borde del área de operación.
    - **Entradas:** distancia con el obstáculo enfrente.
    - **Procesamiento:** si se detecta una distancia menor a la calculada entre el robot y el perímetro se mapean los puntos del obstáculo para buscar sus bordes.
    - **Salidas:** señal de evento de obstáculo encontrado.
    - **Criterios de aceptación:** el robot logra encontrar e identificar el obstáculo con tiempo para reaccionar y no colisionar.
    - **Actores:** Sistema de medición de presencia de obstáculo.

16. **ID:** RF016.
    - **Nombre:** Identificación de áreas.
    - **Descripción:** El robot asocia regiones cuadradas dentro del área de operación y almacena sus coordenadas para definir un área, de origen o de destino.
    - **Entradas:** cadena codificada de un mensaje que contiene la definición de un área.
    - **Procesamiento:** Se recibe una cadena que contiene los datos del área, se interpreta y se convierte en un conjunto de coordenadas que se almacenan en memoria.
    - **Salidas:** Coordenadas y tipo del área.
    - **Criterios de aceptación:** Ubica con un error de 10% el área dentro del área de operación.
    - **Actores:** Administrador de tareas, administrador de mensajes.

17. **ID:** RF017.
    - **Nombre:** Identificación de robot.
    - **Descripción:** El robot identifica la presencia de otro robot por el ancho del obstáculo o porque detecta que se mueve.
    - **Entradas:** distancia con el obstáculo de enfrente, posición y orientación.
    - **Procesamiento:** Se determina el ancho del obstáculo y si es del ancho de un robot se asigna como robot.
    - **Salidas:** coordenadas y velocidad de robot detectado.
    - **Criterios de aceptación:** identificar robots correctamente en todas las pruebas.
    - **Actores:** Módulo de distancia con obstáculo, Administrador de memoria de perímetro del área de operación.
    - **Requerimientos de los que depende:** RF17.

18. **ID:** RF18.
    - **Nombre:** Definir una trayectoria.
    - **Descripción:** El robot toma dos coordenadas separadas dentro del área de operación y define su trayectoria como una línea recta entre ellos. no puede haber una coordenada asociada a un obstáculo que esté entre estos dos puntos. La trayectoria puede estar compuesta por más puntos intermedios para evadir obstáculos.
    - **Entradas:** distancia con el obstáculo de enfrente, posición y orientación.
    - **Procesamiento:** El robot propone una trayectoria de dos puntos con una línea recta siempre que no haya un obstáculo en esa línea, si hay un obstáculo que se mueve propone una coordenada en medio que está desplazada hacia el lado contrario al que avanza el obstáculo, si el obstáculo no se mueve propone evadir por un lado poniendo esa nueva coordenada fuera del área del obstáculo.
    - **Salidas:** conjunto de coordenadas (trayectoria).
    - **Criterios de aceptación:** la trayectoria no cruza sobre un obstáculo.
    - **Actores:** Sistema de detección de obstáculos, Sistema de gestión de memoria del área de operación.

19. **ID:** RF19.
    - **Nombre:** Conexión inalámbrica.
    - **Descripción:** El robot puede establecer una comunicación inalámbrica con la función de paso de mensajes.
    - **Entradas:** Señal del administrador de eventos para generar un evento de envío, señal del administrador de mensajes para enviar un mensaje.
    - **Procesamiento:** Se conecta a través de un protocolo inalámbrico a una red broadcast.
    - **Salidas:** Datos de la conexión.
    - **Criterios de aceptación:** La conexión es estable.
    - **Actores:** Módulo de conexión inalámbrica.

20. **ID:** RF20.
    - **Nombre:** Gestión de tareas.
    - **Descripción:** El robot recibe tareas y las guarda en una lista y las ejecuta
    - **Entradas:** información de tareas extraída de mensajes.
    - **Procesamiento:** cuando recibe una tarea, la da de alta y la almacena.
        - Genera actualizaciones de tarea.
        - Recibe actualizaciones de tarea.
        - Ejecuta cada tarea.
        - No interrumpe una tarea que ya empezó.
    - **Salidas:** Evento de actualización de tarea.
    - **Criterios de aceptación:** Almacena y ejecuta todas las tareas
    - **Actores:** Administrador de eventos.

4.2 Requerimientos No Funcionales

1. **Rendimiento**: El robot procesa datos y toma decisiones en tiempo real.
2. **Escalabilidad**: El sistema debe poder integrar múltiples robots.

3. **Compatibilidad**: El robot debe ser compatible con diferentes sensores y módulos de función.
4. **Eficiencia energética**: El robot debe gestionar eficientemente su consumo de energía para maximizar la duración de la batería.

5. **Uso de Batería recargable**

4.3 Metodología

Se usará una aproximación de comportamientos para realización de tareas como una manera de enfrentar el problema. Donde se implementa cada capa explícitamente y luego se consolida su funcionamiento conjunto para formar un sistema robótico en Capas y niveles donde cada capa tiene una tarea diferente. Cabe la comparación del enfoque de las capas y niveles con la de la composición de un sistema distribuido en sí mismo puesto que "un sistema complejo puede ser dividido en un número de capas, donde las capas superiores hacen uso de los servicios ofrecidos por las capas inferiores… Sin que las capas deban estar al tanto de los detalles de implementación de las otras capas" [19].

La idea central es crear una capa de control correspondiente a cada nivel de competencia e ir agregando más. Por lo que se va perfilando o quizá forzando la organización de los componentes en una estructura jerarquizada donde los datos fluyen hacia arriba cuando se recopilan del entorno, y fluyen hacia abajo cuando se ejecuta una subrutina. Esto es propicio para emplear una arquitectura en capas, donde cada capa esta encima de otra y esto representa un aumento en la abstracción de la actividad de cada capa.

4.4 Arquitectura en capas

De acuerdo con la metodología se dividen las funcionalidades en bloques, los cuales dependen unos de algunos otros, mientras que existen otros que solo tienen relación con uno o unos pocos de ellos, la funcionalidad que implementa un bloque es utilizado por otro superior y así subsecuentemente, incrementando ascendentemente el nivel de abstracción en que se encuentra cada bloque, conformando así una arquitectura de cuatro niveles ascendentes de abstracción.

Planeación

Las subrutinas que componen a la realización de la tarea se enuncian como, girar en algún sentido, avanzar, retroceder, hacer una medición, hacer múltiples mediciones de las dimensiones de un entorno, para formar las rutinas se encadenan subrutinas de manera secuencial formadas ene una memoria de ejecución.

Ejecución

Las subrutinas ejecutan actividades que implican usar los actuadores, los sensores y la comunicación, algunas interactúan con la memoria que inyectan instrucciones en la memoria durante su tiempo de ejecución y encienden o apagan banderas que indican al resto de módulos en la arquitectura el estado general del robot.

Control motriz y Mapeo de datos de sensores

Las subrutinas tienen un impacto directo en el estado de los actuadores, y toman control de ciertos bloques dedicados a controlar los actuadores, estos bloques codifican las operaciones que indican las instrucciones en señales que permiten llevar a cabo el control de los motores y los sensores y consecuentemente controlan la movilidad del robot.

Los datos por sí mismos son solo un conjunto de información que necesita pasar por un análisis que permita interpretarlos como un movimiento o una distancia, particularmente se obtienen tiempos entre mediciones, y se mapean a distancia y velocidad.

Sensores y actuadores

Están enfocados a recopilar datos útiles del entorno, con el objetivo de poder actuar en este medio partiendo de la interpretación de estos datos, necesitaremos una medición de distancia, tanto de espacio entre el sensor y un obstáculo presente, así como de la posición de las llantas que se interpreta como una medición del movimiento del robot sobre el plano que se encuentra.

Capítulo 5 Diseño

Estas etapas nos dan una pista de las partes que incluimos en nuestro diseño y se empieza por la recepción y envío de mensajes, cada mensaje tiene una etiqueta de instrucción y un parámetro, los mensajes hacen que la memoria se cargue de instrucciones, las instrucciones permiten que los bloques de instrucción se activen y hagan actuar y sentir, los datos que esto genera modifican la memoria. Cada instrucción produce diferentes comportamientos y cuando estas han sido ejecutadas se genera un mensaje de que la tarea está terminada.

5.1 Diseño de la Arquitectura

La arquitectura de capas ascendentes de abstracción permite integrar múltiples funcionalidades separando en su nivel de abstracción a unos módulos de otros, diciendo que los elementos de una misma capa comparten información entre ellos puesto que los datos tienen las misma naturaleza e involucra solamente capacidades análogas entre funcionalidades al mismo nivel de operación, entre una capa y otra existe también la comunicación pero la dirección de los datos es de un único sentido ascendente o descendente entre dos capas actuando como una caja negra para la capa contigua , simplificando la construcción de las partes del sistema al modularizar y atomizar la parte de la cual se encarga cada módulo.

5.1.1 Capas

La arquitectura en capas está conformada de la siguiente manera:

primera capa de hardware se refiere a los módulos físicos de hardware, actuadores, sensores, y estructurales, digamos son todos los elementos tangibles que construyen físicamente al robot refiriéndonos a Motores DC, moto-reducciones directas, codificadores, tarjeta de FPGA, chasis, ruedas, batería, fuente reguladora de alimentación, cables, conectores, sensores de distancia, módulo Wifi.

En la segunda capa se tienen los controladores de los actuadores y los sensores. Se encarga de comunicar los datos resultantes de la capa superior con los elementos físicos que componen a la capa inferior para poder ejercer un control físico del robot, así mismo se encarga de pasar hacia capas superiores los datos de los sensores encargándose de la caracterización de datos obtenidos del ambiente, con tal de ponerlos interpretados como parámetros del medio real en tipos de datos manejables.

La tercera capa se encarga de emitir instrucciones para la conducción únicamente dando datos que puede interpretar la capa adyacente y que no implican más información de cálculo puesto que las capas inferiores ya cuentan con el manejo de esa información para convertir el comportamiento que se les indica en las acciones físicas consecuentes que provocan dicha conducción, automatiza el control de la velocidad de los motores, caracteriza los mensajes recibidos y extrae su información para poder manejar los mensajes como eventos, parametriza los eventos registrados por los sensores habiendo obtenido los datos a su vez parametrizados y convertirlos en una

interpretación de la presencia o ausencia de obstáculos, distancia a la que se encuentra del perímetro exterior en el espacio controlado , y también recopila la información necesaria para construir mensajes a partir de las condiciones en las que se encuentra el sistema y dando lugar a que se produzcan eventos puntuales dentro del medio, de esta información puede llevar a cabo el reconocimiento de objetos, y determinar su propia localización así como la de los demás objetos . Controla la demanda de acciones para controlar su posicionamiento y orientación, así como los eventos de puntos dentro de una trayectoria que ya han sido alcanzados.

La cuarta capa control central que tiene la tarea de almacenar algoritmos de comportamiento, en forma de una lista de subrutinas, permite el seguimiento de la conducción sobre una trayectoria, registra los datos que está obteniendo del entorno útiles para la navegación en el espacio físico.

Se muestra un diagrama de la arquitectura en capas de abstracción ascendente:

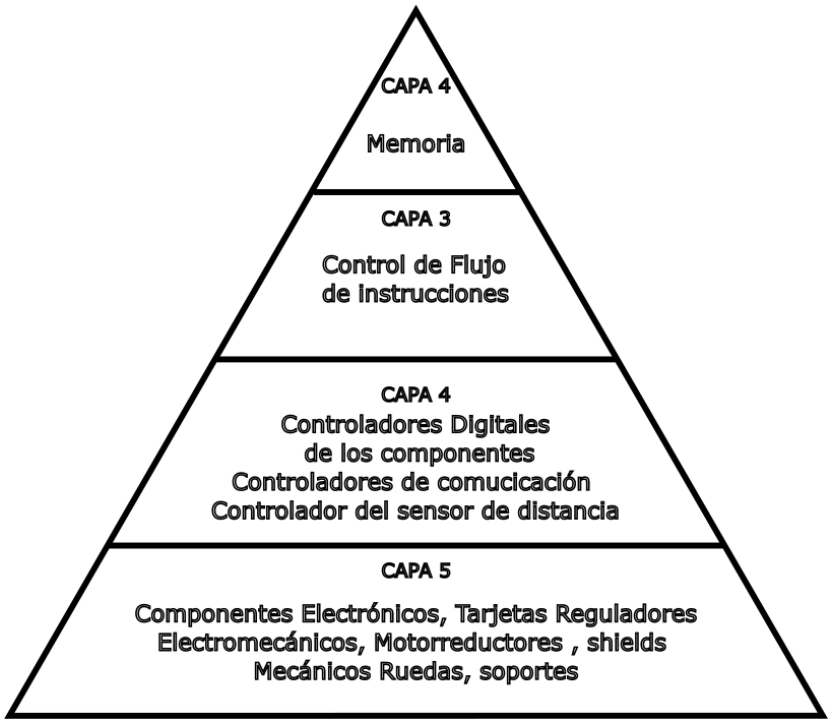


Ilustración 11 Diagrama de Capas ascendentes de la Arquitectura

El siguiente diagrama de componentes ilustra la arquitectura en capas del sistema, mostrando la interacción jerárquica entre sus diferentes niveles. Desde la base de hardware hasta el control central, cada capa desempeña funciones específicas que permiten la integración y operación del robot. Este esquema refleja la relación funcional entre sensores, actuadores, controladores y algoritmos de alto nivel, destacando cómo se gestionan las tareas y el flujo de información en el sistema.

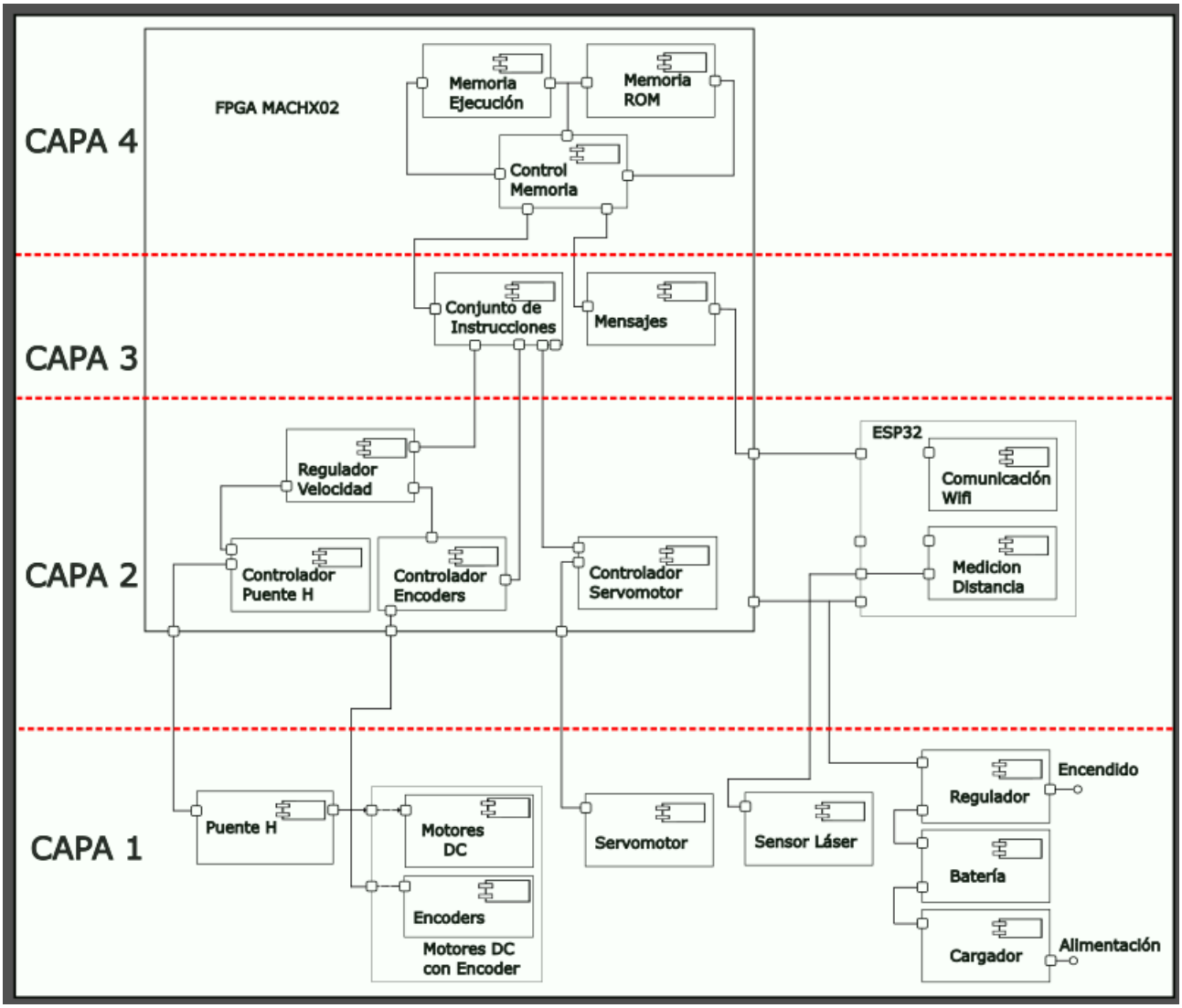


Ilustración 12 Diagrama de Componentes de la Arquitectura

5.1.2 Diagrama Eléctrico

Para cada componente se requiere una alimentación de voltaje estable, a continuación, se presenta un diagrama de bloques de la arquitectura desde la perspectiva de las conexiones entre los componentes y su fuente de alimentación, considerando que los motores demandan mucha corriente y generalmente inducen ruido electromagnético, se ha separado en dos etapas la de Control y la de Actuadores.

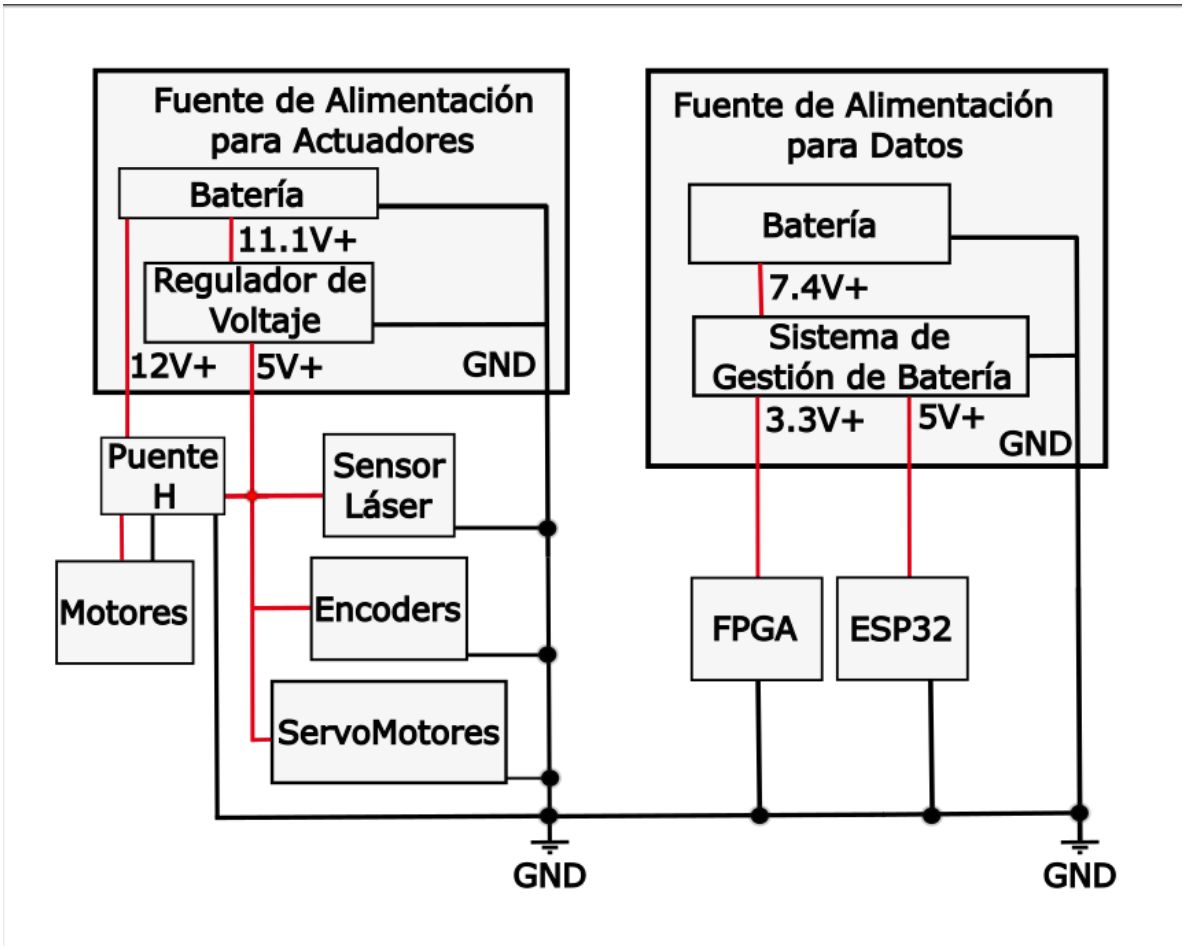
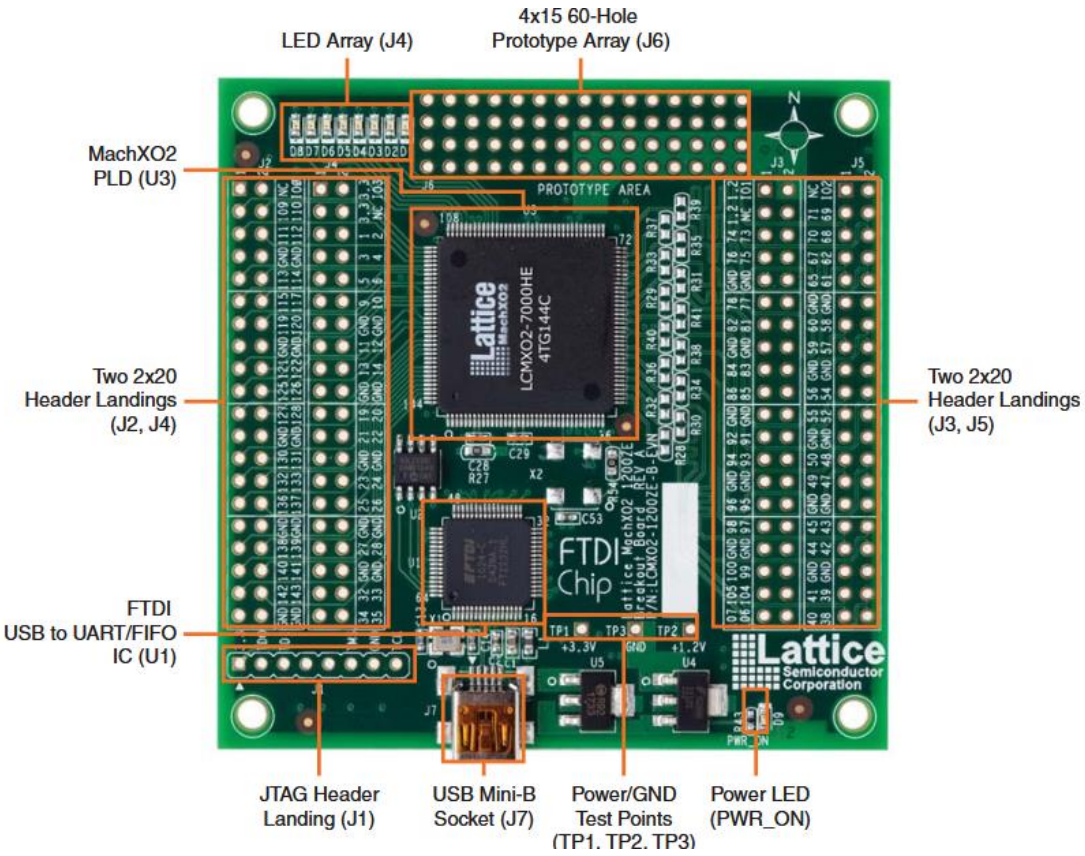


Ilustración 13 Diagrama de bloques del diseño eléctrico

5.3 Componentes físicos

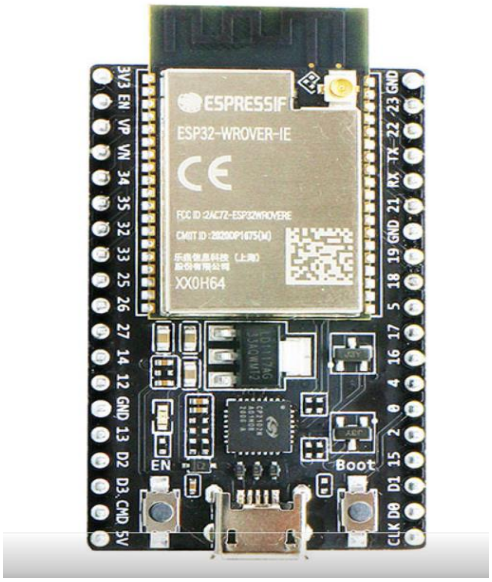
5.3.1 FPGA

Por las ventajas mencionadas antes y por un enfoque de buscar diseñar una arquitectura de un microcontrolador específico para un robot autónomo móvil que tenga funciones en modo multiagente. Se emplea la FPGA MACHX02 de Lattice en la MachX02 breakout development board que tiene 6864 LUTS y 112 GPIO disponibles y una frecuencia de mínimo 2.08Mhz.



5.3.2 ESP32

Es un microcontrolador de la marca esspressif que tiene 32 bits y que puede ser programado a través del ide de Arduino en lenguaje C se conecta a la FPGA a través de 8bits en paralelo y al sensor de distancia TOF por medio de las señales SDA y SCL. Además, se emplea para la transmisión de datos entre tarjetas de dos robots por medio del protocolo ESP NOW.



5.3.3 Sensor de distancia

El sensor láser que se utiliza es un sensor del tipo Time of Flight TOF. Modelo VL53L0X que se comunica mediante el protocolo I2c a un microcontrolador en este caso la esp32 por medio de los puertos SDA SCL.

5.3.4 Baterías

En la etapa de actuadores se utiliza una fuente de alimentación que empieza con el almacenamiento, una batería litio-ion de 3 celdas cilíndricas modelo 18650 B10 que se colocan en una porta pilas de 3 unidades conformando una batería de 11.1 V y 2200mAh.

La fuente de alimentación de la etapa de controladores empieza por la fuente de voltaje misma, una batería de tipo litio-ion cilíndricas modelo 18650 B10 de 3.7 volts y 2200 mAh formando una batería de 7.4V, cuya capacidad de corriente en miliamperes por hora es de 2200 mAh.

5.3.5 Motorreductores con codificadores

Para los motores se emplea un modelo de motores de corriente directa con codificadores y motorreducción de engranes planetarios, por la facilidad que supone controlar un motor de este tipo y la ventaja que ofrece en torque a pesar de que su tamaño es pequeño y adecuado. Integrada con los motores tiene una relación de 1:26 por lo que da una salida de revoluciones de 330 RPM a 12 V sin carga 1.5 A en parada

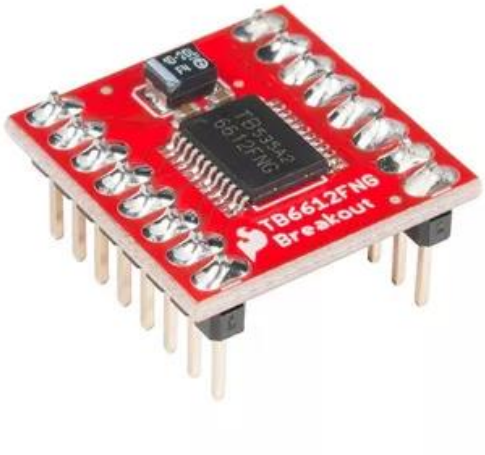
Ofrecen otra ventaja que es el de tener integrado un arreglo de codificadores con imán y sensores hall que permite contar 11 regiones del imán por cada vuelta que da el eje del motor.

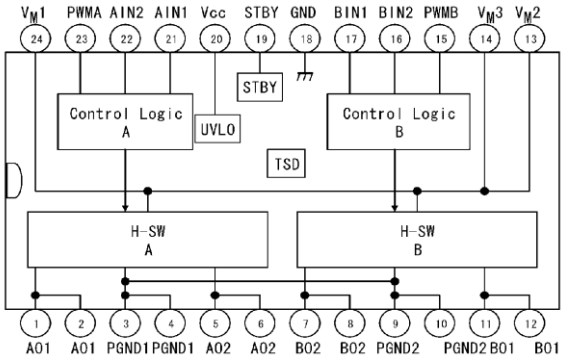
Los codificadores se conectan a través de una señal y un negativo a la FPGA mediante resistencias de pull down con dos señales a y b para cada sensor.

Puente H

el puente h tiene entradas para dos motores de máximo 3 A en parada, una alimentación de 3.3v a 5v para el control y un voltaje de motores de máximo 12 volts, se controla con una señal PWM para cada motor y tiene la capacidad de girar ambos motores regulando su sentido y velocidad de giro de manera independiente. Se utiliza un controlador dual de motores de corriente continua

TB6612FNG. A continuación, se presenta un diagrama de los puertos de este controlador, así como sus modos de operación.





Ain1 y Ain2 forman el un arreglo de 2 bits indicando al puente H la dirección de giro de la Rueda A. Por su parte Bin1 y Bin2 forman el un arreglo de 2 bits indicando al puente H la dirección de giro de la Rueda B. PWMA y PWMB son señales de ancho de pulso modulado que regulan la velocidad de giro de la rueda A y B respectivamente.

Servomotor

El servomotor es requerido para girar 180 grados el sensor de distancia y así ofrecer un campo de medición más amplio que no requiere que el robot modifique su posición o su orientación. Se escogió este modelo por su tamaño y facilidad de uso. EL servomotor es controlado con una señal digital de PWM con una fase de trabajo de entre 1 y 2 ms para posicionarse entre 0 y 180 grados respectivamente.

5.4 Componentes en software

5.4.1 Bloque de comunicación

El bloque de comunicación está dedicado a enviar 8 bits entre dos tarjetas ESP32 que se conectan a través del protocolo ESP NOW, hay una parte del código que envía y una parte que recibe haciendo las dos partes una sola que permite la conectividad bidireccional hacia las tarjetas conectadas en la red. A continuación, se muestran diagramas de flujo de ambas partes.

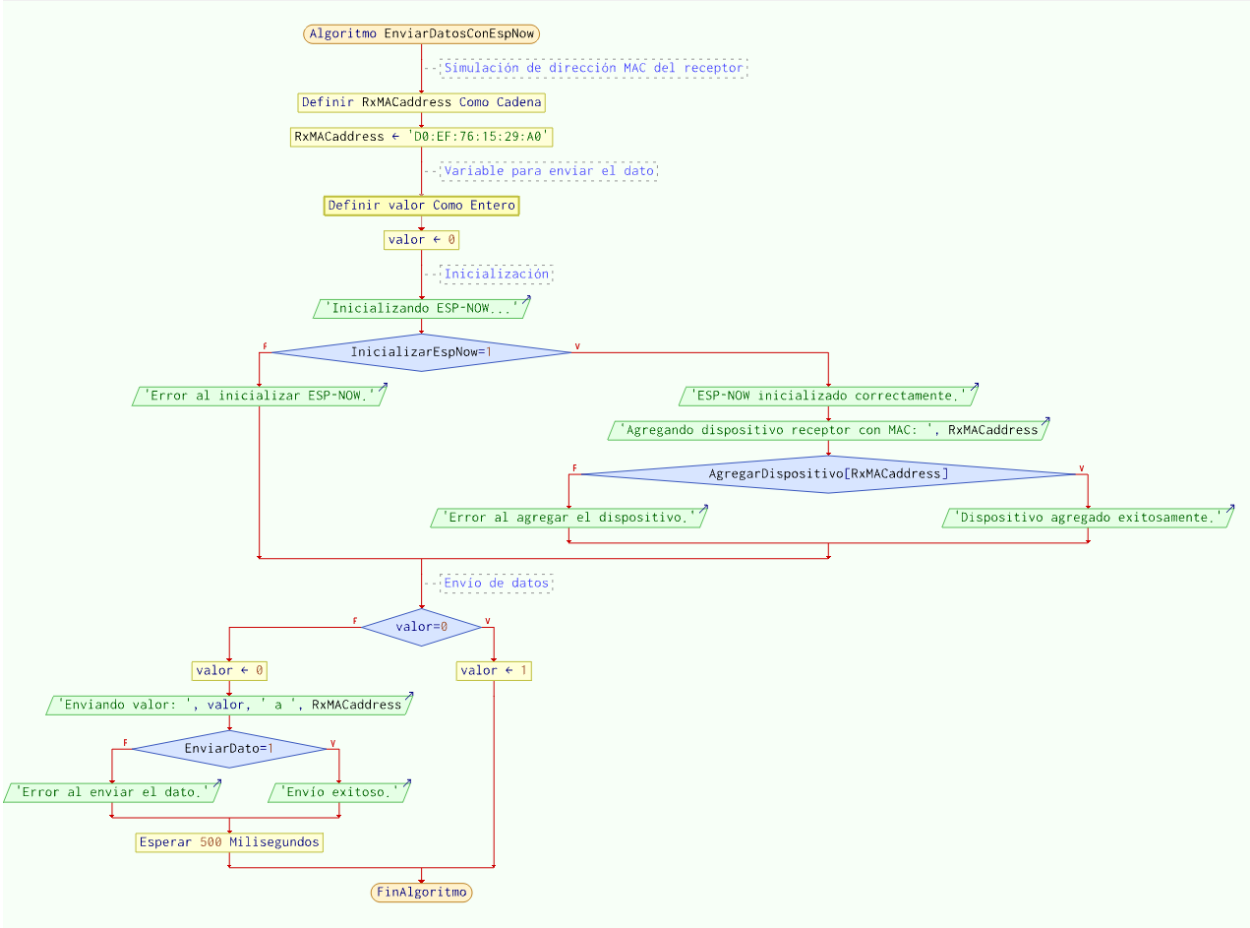


Figure 1 Diagrama de flujo de Envío ESPNOW

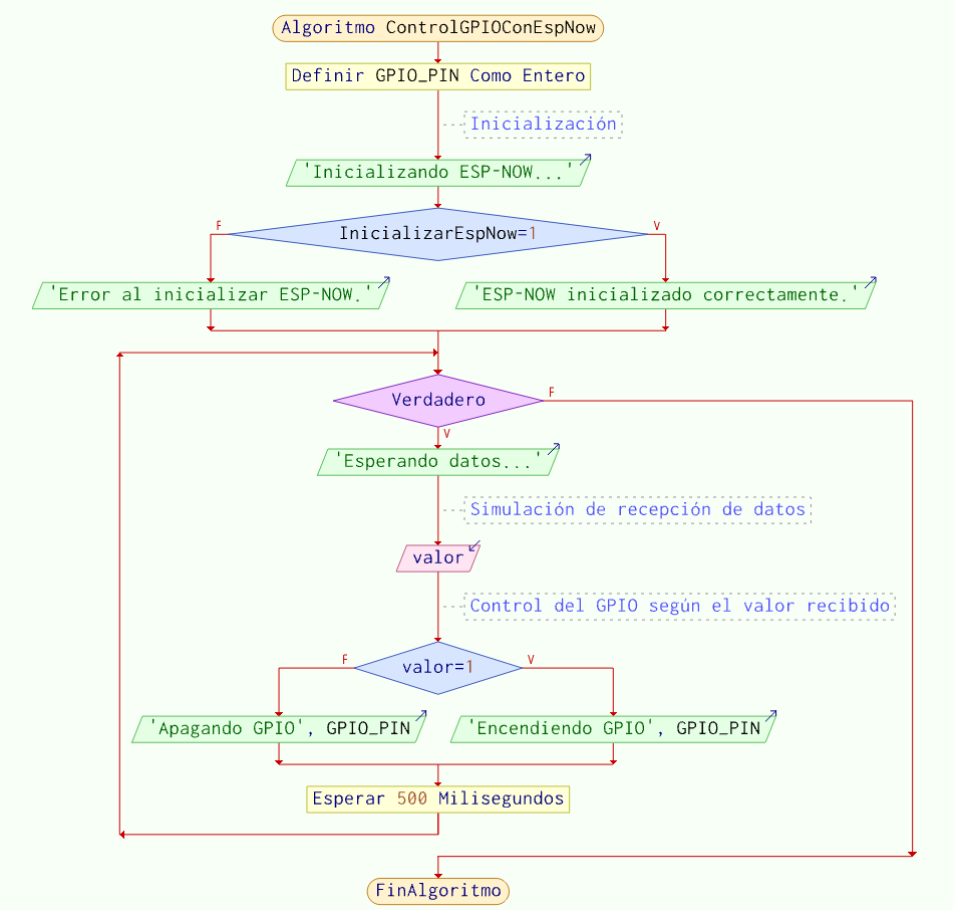


Figure 2 Diagrama de flujo Recibo ESPNOW

5.4.2 bloque controlador del sensor de distancia con láser

El uso de la ESP32 está enfocado al empleo de la biblioteca Adafruit_VL53L0X diseñada para el sensor ToF, por la facilidad que tiene la tarjeta de emplear el protocolo I2C con el cual se establece la comunicación y se hacen peticiones de mediciones al sensor, pasando posteriormente estos datos en un bus de 8 bits dirigido hacia la FPGA, específicamente hacia los bloques de detección de obstáculos. el sensor láser envía distancia en 8 bits al módulo detector de obstáculo, codifica los datos recibidos del sensor por medio del protocolo i2c a un bus paralelo que va hacia la FPGA en 8 bits con la distancia en centímetros.
Z

5.5 Componentes en Hardware de lógica reconfigurable

5.5.1Controlador del Puente H
En la FPGA se implementa el bloque que controla al puente H. Tiene como entradas las velocidades de control de los motores en PWM, las direcciones individuales del sentido de giro , de salida tiene dos terminales para cada motor y Entonces el controlador para esta unidad deberá tener de salidas Ain1 y Ain2 y PWMA, Bin1 y Bin2 y PWMB donde la señal PWM regula la

velocidad de giro de las ruedas y las señales Ain y Bin regulan el comportamiento de cada motor haciendo que la rueda que se detenga o que gire en sentido horario o en sentido antihorario como se muestra en la siguiente imagen.

Visto desde arriba, la configuración de las ruedas diferenciales se aprecia de la siguiente manera.

A partir de las entradas de las señales Ain 1 y 2 así como de Bin1 y 2 se producen una serie de movimientos en la posición y orientación en el plano local del robot dado el sentido de giro de las ruedas. Las combinaciones de la señal de entrada y su respectivo efecto en los movimientos del robot se muestran en la siguiente tabla.

Tabla 2 Movimientos posibles del robot

Movimiento	Vector "Ain1 Ain2 Bin1 Bin2"	Giro de las ruedas
Avanzar	**0101**	
Retroceder	**1010**	
Girar a la Derecha	**1001**	
Girar a la Izquierda	**0110**	
Detenerse	**0000**	

A continuación, se muestra un diagrama de bloque que representa los componentes internos del Controlador del Puente H, así como de las conexiones entre ellos y sus puertos de entrada y salida.

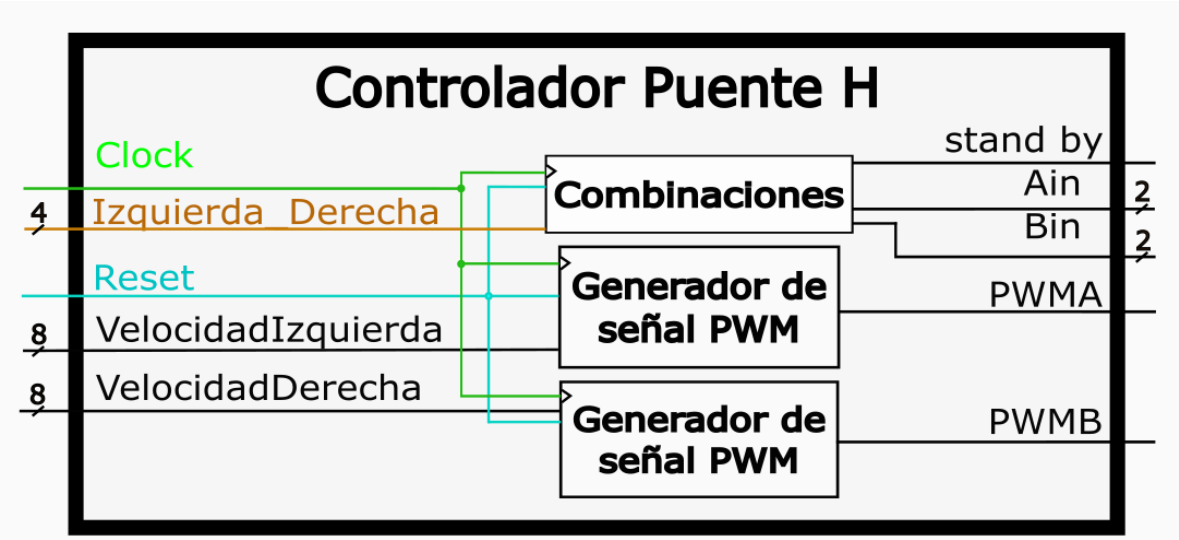


Ilustración 14 Diagrama de bloque de Controlador Puente H

5.5.2Controlador de los codificadores

el controlador del codificador está implementado como una máquina de estados basada en un registro de 3 Flip Flops t con un circuito combinacional que determina las transiciones en las entradas de los Flip Flops para conducir las transiciones de la máquina entre sus estados y determinar el sentido de giro, así como los pulsos que son detectados, de manera que contar los pulsos que llegan en un determinado tiempo sirve para calcular los grados que gira cada rueda independientemente.

La máquina de estados fue diseñada con una metodología de máquina de Mealy a partir de una tabla de estados y una tabla de Karnaugh esto es mostrado a continuación en la tabla.

Las funciones de transición que se encuentran son las que se van a configurar dentro del elemento Combinaciones que es parte del controlador, conformando la parte lógica combinacional de este componente digital, secuencial.

Tabla 3 Funciones de transición FlipFlops Detector de codificadores

E2	E1	E0	B	A	Estado siguiente	Q2	Q1	Q0	FQ2	FQ1	FQ0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	1	0	0	1
0	0	0	1	1	error	1	0	1	1	0	1
0	0	0	1	0	2	0	1	0	0	1	0
0	0	1	0	0	error	1	0	1	1	0	0
0	0	1	0	1	1	0	0	1	0	0	0
0	0	1	1	1	3	0	1	1	0	1	0
0	0	1	1	0	error	1	0	1	1	0	0
0	1	1	0	0	error	1	0	1	1	1	0
0	1	1	0	1	error	1	0	1	1	1	0
0	1	1	1	1	3	0	1	1	0	0	0
0	1	1	1	0	7	1	1	1	1	0	0
0	1	0	0	0	error	1	0	1	1	1	1
0	1	0	0	1	error	1	0	1	1	1	1
0	1	0	1	1	4	1	0	0	1	1	0
0	1	0	1	0	2	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	1	1	0
1	1	0	0	1	6	1	1	0	0	0	0
1	1	0	1	1	error	1	0	1	0	1	1
1	1	0	1	0	error	1	0	1	0	1	1
1	1	1	0	0	0	0	0	0	1	1	1
1	1	1	0	1	error	1	0	1	0	1	0
1	1	1	1	1	error	1	0	1	0	1	0
1	1	1	1	0	7	1	1	1	0	0	0
1	0	1	0	0	0	0	0	0	1	0	1
1	0	1	0	1	error	1	0	1	0	0	0
1	0	1	1	1	error	1	0	1	0	0	0
1	0	1	1	0	error	1	0	1	0	0	0
1	0	0	0	0	error	1	0	1	0	0	1
1	0	0	0	1	6	1	1	0	0	1	0
1	0	0	1	1	4	1	0	0	0	0	0
1	0	0	1	0	error	1	0	1	0	0	1

Esta tabla se utiliza para generar tres mapas de Karnaugh en los cuales se encuentran los maxitérminos que permiten encontrar las funciones lógicas de transición de cada Flipflop, esto conducirá al comportamiento deseado, que es el de detectar la llegada de un polo del imán frente al sensor Hall del codificador, así como su sentido de giro.

Tabla 4 Mapa Karnaugh Flip Flop 1

E2E1\E0B, A=0	.00	.01	11	10	E2E1\E0B, A=1	0	1	11	10
.00	0	0	1	1	0	0	1	0	0
.01	1	0	1	1	1	1	1	0	1
11	1	0	0	1	11	0	0	0	0
10	0	0	0	1	10	0	0	0	0
	E1 and not E0 and not B and not A								
	not E2 and E0 and not A								
	E0 and not B and not A								
	not E2 and E1 and not B								
	not E2 and not E0 and B and A								

Table 5 Mapa Karnaugh Flip Flop 2

E2E1\E0B, A=0	0	1	11	10	E2E1\E0B, A=1	0	1	11	10
0	0	1	0	0	0	0	0	1	0
1	1	0	0	1	1	1	1	0	1
11	1	1	0	1	11	0	1	1	1
10	0	0	0	0	10	1	0	0	0
	E1 and not B and not A								
	E1 and E0 and not B								
	E2 and E1 and not E0 and not A								
	not E2 and E1 and not E0 and A								
	E2 and E1 and B and A								
	not E2 and not E1 and E0 and B and A								
	E2 and not E1 and not E0 and not B and A								
	not E2 and not E1 and not E0 and B and not A								

Table 6 Mapa Karnaugh Flip Flop 3

E2E1\E0B, A=0	0	1	11	10	E2E1\E0B, A=1	0	1	11	10
0	0	0	0	0	0	1	1	0	0
1	1	0	0	0	1	1	0	0	0
11	0	1	0	1	11	0	1	0	0
10	1	1	0	1	10	0	0	0	0
	not E2 and E1 and not E0 and not B								
	E2 and E1 and not E0 and B								
	E2 and not E1 and not E0 and not A								
	not E2 and not E1 and not E0 and A								
	E2 and E0 and not B and not A								

5.5.3Bloque de Detección de obstáculo

El bloque detector de obstáculo recibe la medición de la distancia entre el robot y un obstáculo frente al sensor, dependiendo de la calibración (distancia que se considera suficientemente pequeña para ser un problema), este bloque genera una señal que indica si hay un obstáculo tan cerca como para que se considere que está en medio del camino del robot, por lo que este se detendrá, este bloque trabaja de manera concurrente a los demás bloques y en cualquier momento se le puede consultar la presencia o ausencia de un obstáculo considerablemente cerca. A continuación, se presenta un diagrama de flujo que representa la actividad de este componente.

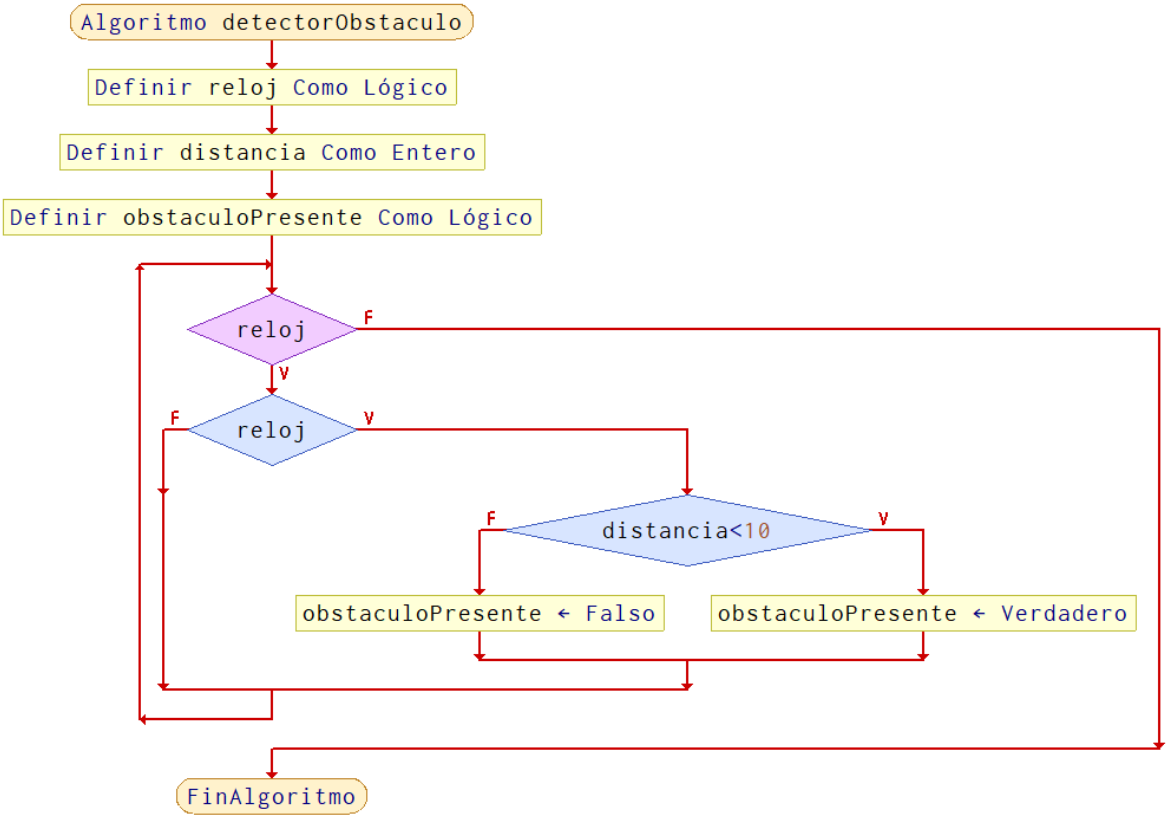


Ilustración 15 Diagrama de flujo del Detector de obstáculo

5.5.4 La unidad de control de memoria

maneja la salida y la entrada secuencial de las instrucciones desde y hacia la Memoria de ejecución, el estado de la Memoria de ejecución sirve cómo árbitro entre los bloques de las instrucciones que son componentes que despliegan un comportamiento físico de los actuadores, y la lista de instrucciones que contiene, van saliendo de la memoria de ejecución y los bloques de instrucciones que son activados de manera secuencial, de acuerdo con la instrucción que corresponda a ser procesada.

5.5.5Conjunto de instrucciones

Las instrucciones son bloques de código que se ejecuta mientras que un código de operación especificado se encuentre presente para activarlo. Para cada instrucción, el código de operación viene dentro de cada instrucción como el primer campo, con 4 bits de longitud, generando así un set de instrucciones de máximo 16 instrucciones, diferentes, las cuales solo hay 8 que serían: Avanzar, Retroceder, Girar a la derecha, Girar a la izquierda, Alinearse con pared, Alinearse con objeto, Alinearse con esquina, Detenerse. A continuación, se presentan las descripciones y los diagramas respectivos de cada instrucción.

Avanzar

La primera instrucción es la de avanzar, recibe como parámetro una cantidad de centímetros a avanzar con 8 bits se pueden indicar máximo 255cm de carrera, cuando el controlador de los codificadores ha contado los pulsos de codificadores correspondientes a la distancia solicitada en ambas ruedas al mismo tiempo, el robot se detiene y aumenta en uno el contador de memoria que le indica en que instrucción seguirá la ejecución.

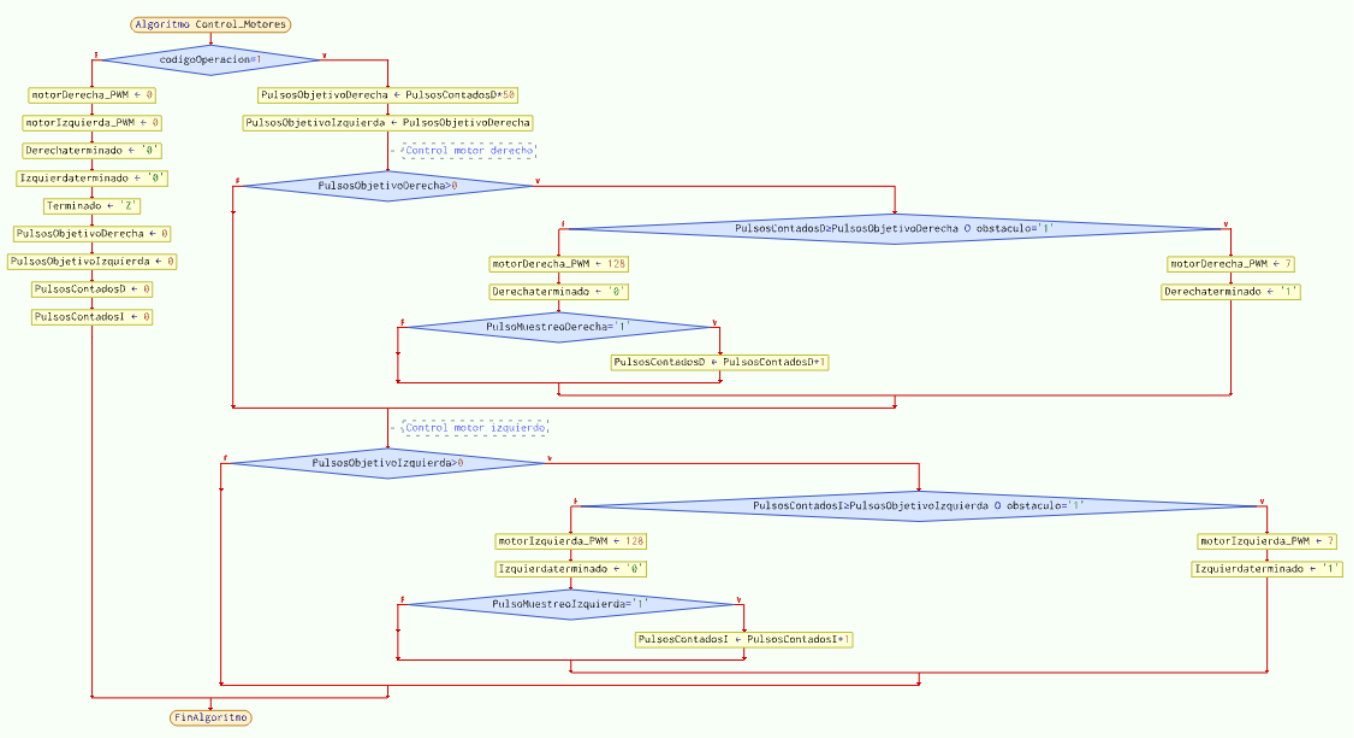


Ilustración 16 Diagrama de Flujo de Instrucción AVANZAR

Retroceder

La segunda instrucción es la de retroceder, recibe como parámetro una cantidad de centímetros a retroceder con 8 bits, se pueden indicar máximo 255cm de carrera, cuando el controlador de los codificadores ha contado los pulsos de codificador correspondientes a la distancia solicitada en

ambas ruedas al mismo tiempo, el robot se detiene y aumenta en uno el contador de memoria que le indica en que instrucción seguirá la ejecución.

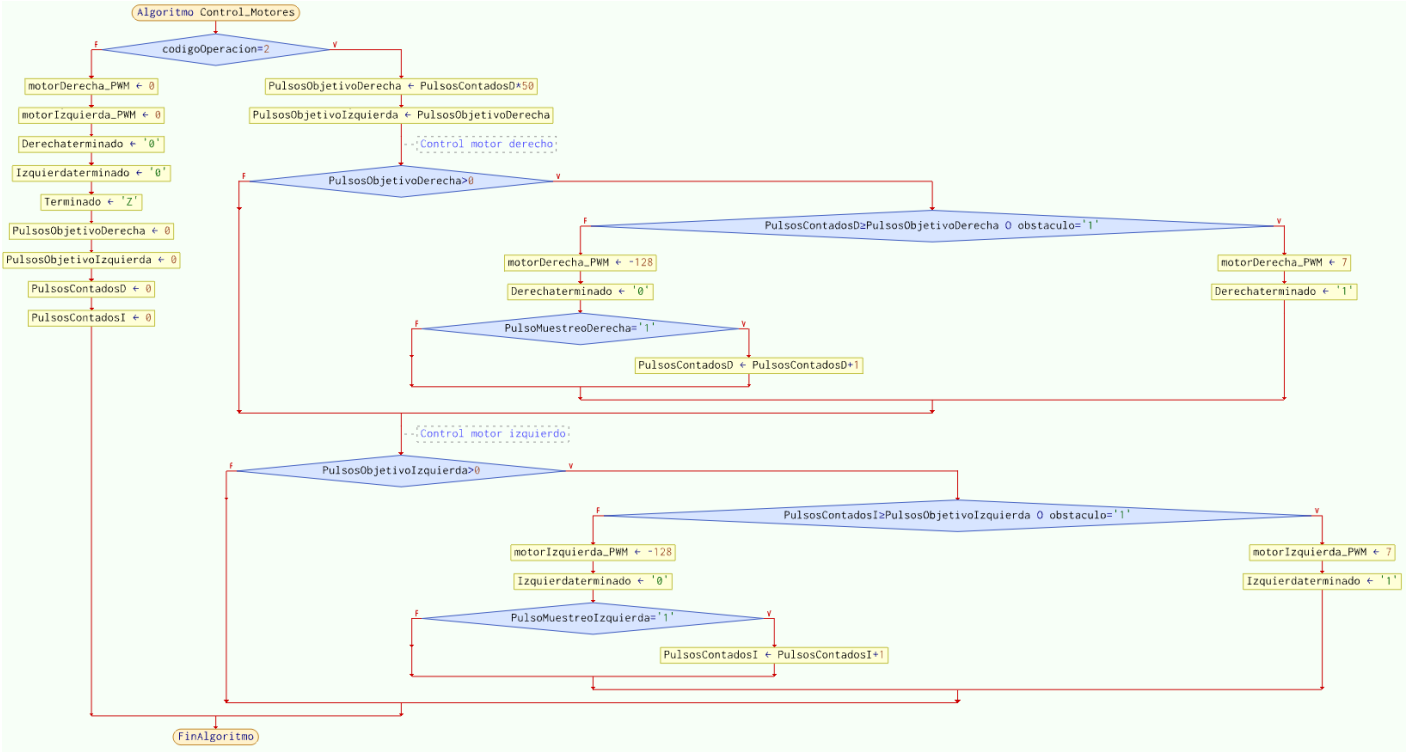


Ilustración 17 Diagrama de Flujo de Instrucción Retroceder

Girar a la Derecha

La tercera instrucción es la de Girar a la derecha, recibe un parámetro que son los grados por girar, prácticamente hace lo mismo pero la diferencia yace en que la rueda derecha gira hacia atrás y la izquierda gira hacia adelante y en este caso también, ambas giran a la misma velocidad angular, generando que el robot rote sobre su propio centro hasta que se ha detectado la cantidad de pulsos en el codificador respectiva de los grados que indica el parámetro.

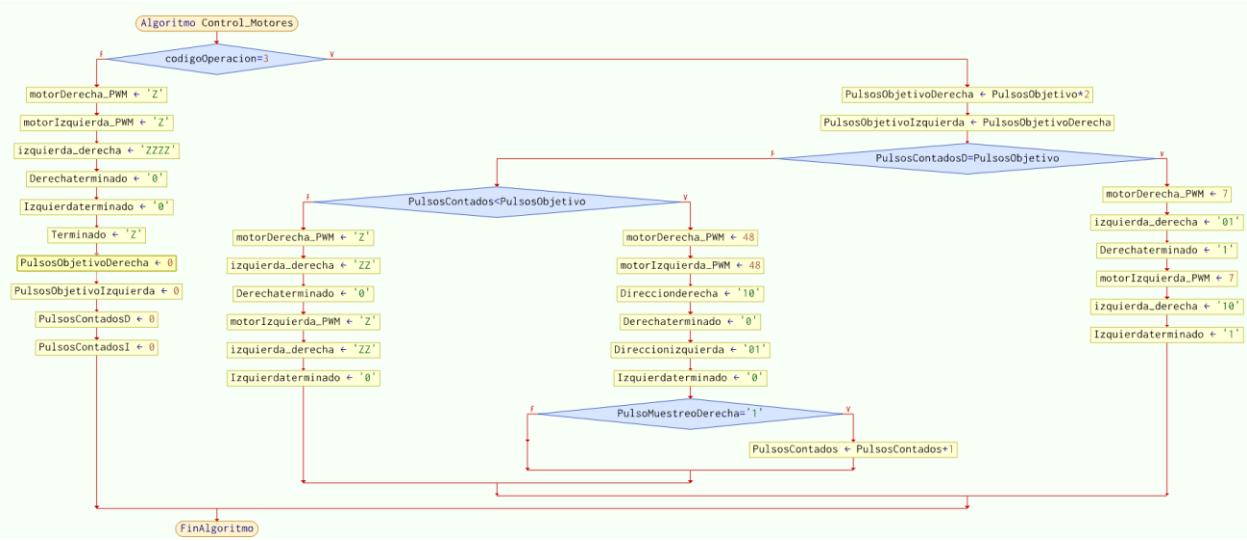


Ilustración 18 Diagrama de flujo de Girar a la Derecha

Girar a la izquierda

La cuarta instrucción es la de girar a la izquierda, recibe un parámetro que son los grados por girar, prácticamente hace lo mismo que la anterior, exceptuando que ahora la rueda derecha gira hacia adelante y la izquierda gira hacia atrás y en este caso también, ambas giran a la misma velocidad angular, generando que el robot rote sobre su propio centro hasta que se ha detectado la cantidad de pulsos en el codificador respectiva de los grados que indica el parámetro.

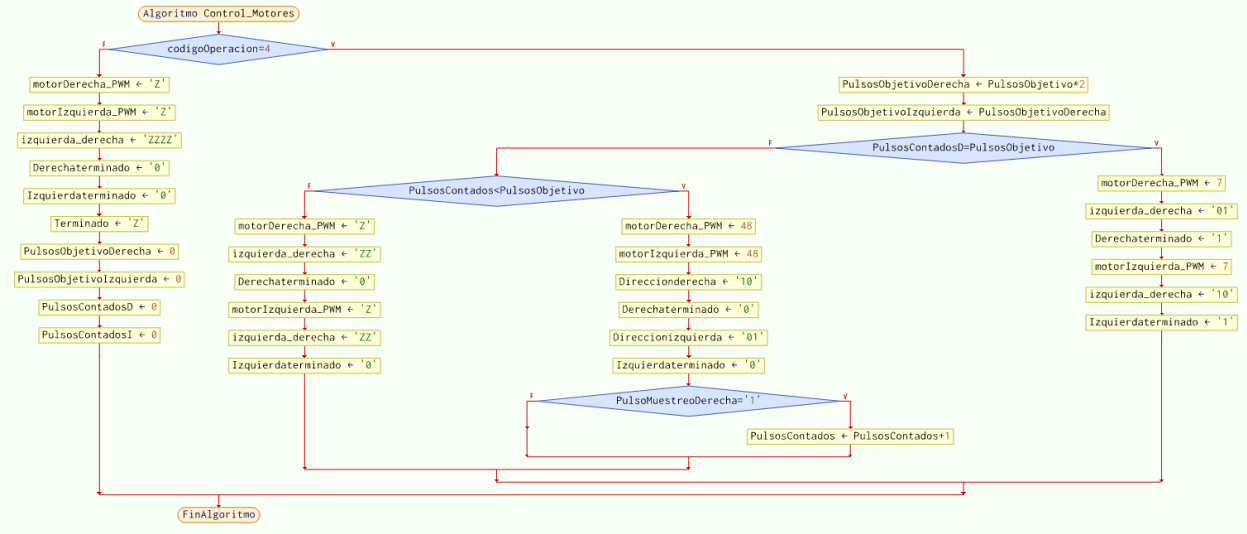


Ilustración 19 Diagrama de flujo de Girar a la Izquierda

Diseño de las rutinas

La rutina tiene una base en un conjunto de instrucciones que se encuentran en la memoria y que al ser procesadas generan un comportamiento especificado de los actuadores, que genera un cambio en el sistema y subsecuentes cambios en los datos del sistema. Cuando una rutina es terminada o interrumpida, esta puede generar una inyección de otra instrucción en la memoria o de varias, así cuando la ejecución vuelve a procesar una instrucción siguiendo con la que está a continuación, se encuentra con la que ha sido recientemente introducida. Una instrucción que es inyectable no puede inyectar otras instrucciones.

Memoria de ejecución

La memoria de ejecución es una entidad que Maneja 32 arreglos de 12 bits, cada arreglo está dispuesto para almacenar una instrucción, compuesto de su código de operación junto con su parámetro, la memoria está direccionada con un vector de 4 bits, tiene una bandera de entrada de lectura y una bandera de entrada de escritura, un bus de entrada con los 12 bits que tiene de longitud el dato que se almacena o que se lee.

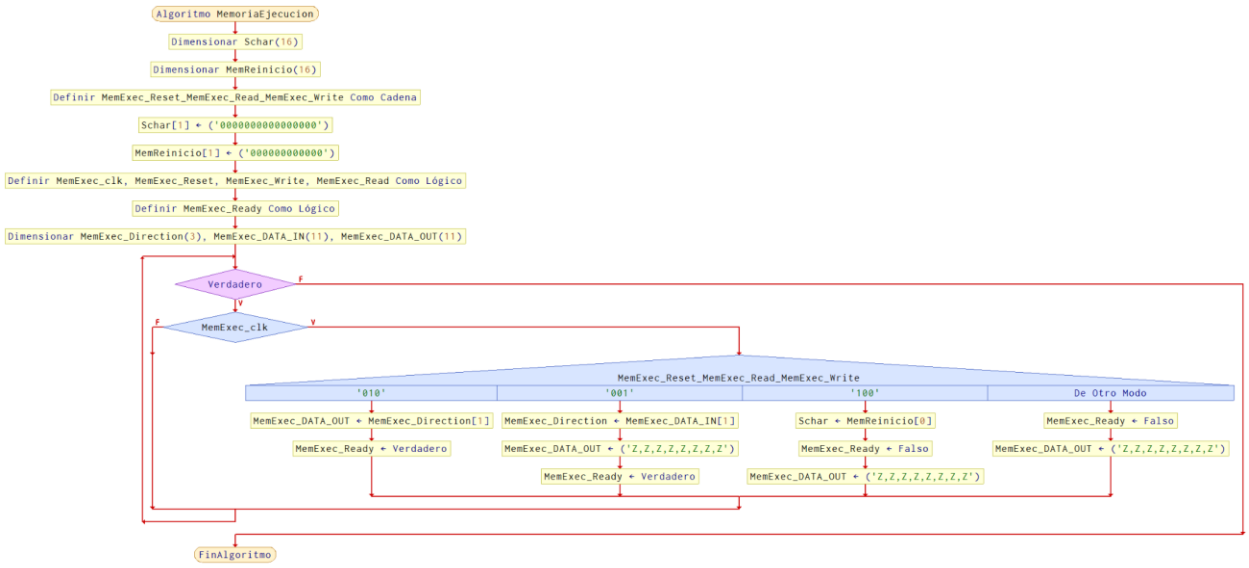


Figure 3 Diagrama de flujo de Memoria de ejecución

5.5.6 Memoria ROM

La Memoria Rom tiene

Un Respaldo de los datos que se cargan al inicio del funcionamiento, cuando se enciende el robot y se carga el primer conjunto de instrucciones, este sale de la Memoria ROM, esta memoria solo puede ser Editada en La configuración de la FPGA, dado que son Vectores que se han inicializado durante la codificación.

5.6 Diseño de la tarjeta

En el desarrollo de sistemas electrónicos complejos, la integración de múltiples tarjetas o módulos es un aspecto clave para lograr una operación eficiente y organizada. Este reporte presenta el

diseño de una tarjeta de interconexión, cuyo propósito principal es facilitar la conexión estructurada entre diversos módulos electrónicos.

Esta tarjeta no contiene componentes activos ni realiza procesamiento directo de señales; su función se centra exclusivamente en servir como un puente de comunicación, proporcionando las conexiones físicas necesarias para unir distintas tarjetas mediante cables y conectores. Este diseño responde a la necesidad de simplificar el manejo de señales, mejorar la organización del sistema y garantizar la fiabilidad de las conexiones en aplicaciones de alta complejidad.

La implementación de esta tarjeta de interconexión obedece a razones técnicas de gran relevancia, como la reducción de ruido eléctrico, la optimización del espacio dentro del sistema y la mejora en el modularidad del diseño. Este enfoque no solo facilita el mantenimiento y las actualizaciones del sistema, sino que también asegura un nivel elevado de integridad en la transmisión de señales entre las tarjetas conectadas.

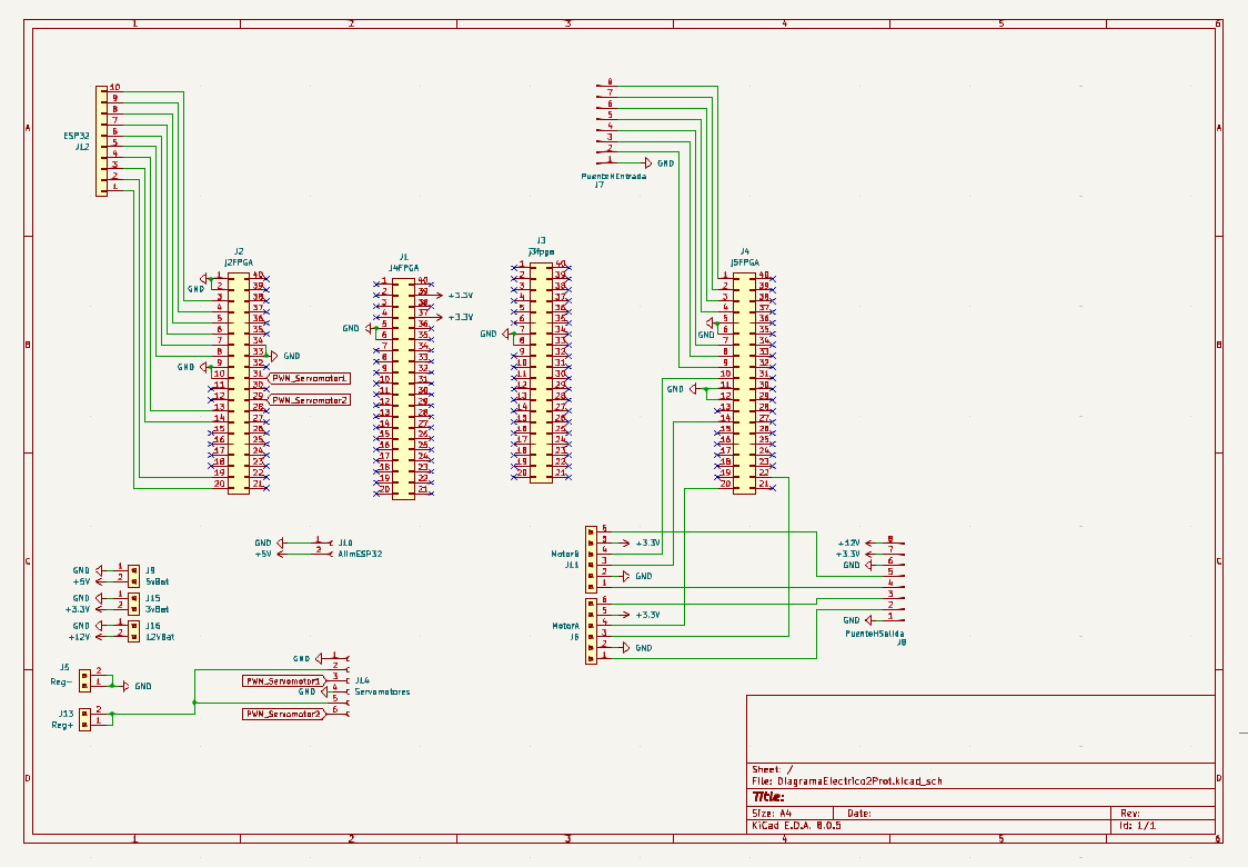


Ilustración 20 Diagrama Eléctrico

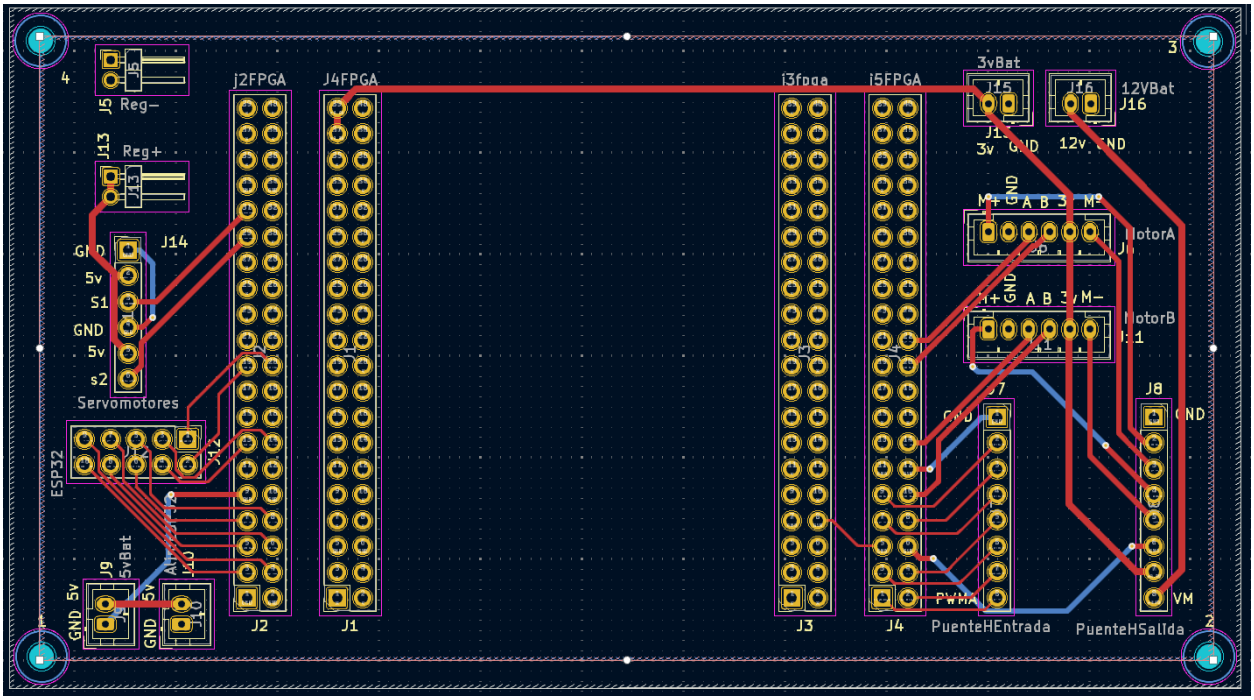


Ilustración 21 Diseño de PCB

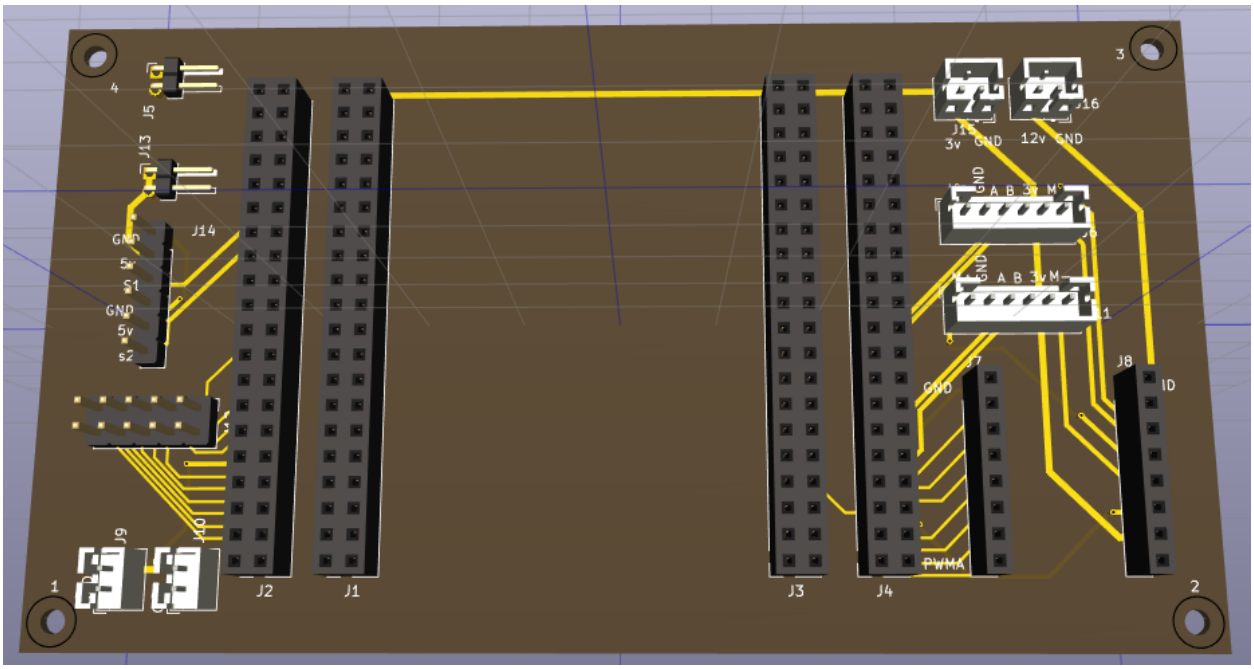


Ilustración 22 Simulación de la PCB terminada

Capítulo 6 Implementación y Resultados

Durante el siguiente capítulo se analizarán los pasos que se siguieron para la construcción de los diseños antes planteados, la corrección de sus errores y el ciclo completo de integración hasta obtener el producto del prototipo planteado en el diseño., se documentó el proceso práctico que llevó a transformar los diagramas y modelos diseñados en un sistema físico funcional. La correcta implementación de los módulos y su integración aseguraron que la arquitectura propuesta se materializara en un robot operativo, preparado para ejecutar las tareas específicas para las que fue diseñado.

6.1 Montaje mecánico del vehículo

6.1.1 chasis

El chasis es una lámina de aluminio con la forma de una T, donde se fija con tornillos la tarjeta base que integra todas las conexiones físicas de los componentes, los soportes de los motores y el servomotor. Primero hice un prototipo en papel cascarón y posteriormente cuando las dimensiones de la tarjeta y de los demás componentes incluidos en el primer prototipo estaban confirmadas, procedí a comprar y cortar con la misma forma la lámina usando una segueta delgada, y un taladro para las perforaciones donde entran los tornillos que sirven de mecanismo de fijación.

6.1.2 Conexiones en protoboard

Preliminarmente cuando los componentes estuvieron reunidos los ensamblé sobre el chasis inicial, fijé dos protoboards a este chasis e hice las conexiones entre los pines libres de la FPGA hacia el puente H pues le entrega las señales de PWM directamente, hacia la ESP32 para la comunicación y la medición de la distancia, la conexión hacia el servomotor de su señal de PWM, las señales de los codificadores directamente a los puertos de la FPGA. Para todo esto me apoyé de la flexibilidad que ofrece la protoboard al momento de hacer conexiones temporales que permiten corregir y entender los problemas que van surgiendo cuando los distintos componentes interactúan.

Algunos problemas que encontré en esta etapa fueron inestabilidades en el comportamiento debidas a los falsos contactos o incluso desconexiones que causa el repetido uso de las pistas de la protoboard, además de que la marca de la protoboard no era muy buena y en algún momento se salió la lámina que lleva dentro que produce el contacto , haciendo parecer que el puente H había dejado de servir o que la ESP32 no estaba siendo alimentada, problemas que resolví tras muchas pruebas y que finalmente desaparecieron con la implementación de la tarjeta, pues esta eliminó todos esos falsos negativos y más conflictos que tuvieron más que ver con errores de conexión y corto circuitos causados por el movimiento de los cables.

6.1.3 Placa

La placa fue diseñada en el software KICad primero se realizó el diagrama de conexiones entre los componentes, luego se asignaron las diferentes huellas a cada componente, de acuerdo con los componentes reales que pude conseguir para tener la seguridad de que podría fabricarse sin que me sorprendiera haber considerado emplear alguna pieza que no pudiera encontrar en el mercado.

El resultado de la placa impresa se muestra en la siguiente figura

6.1.4 Soportes para los motores

Los soportes de los motores están hechos en Nylamid y abrazan el motor por la parte más externa aprisionándolo con los tornillos, las perforaciones en el chasis están hechas de manera que se puede ajustar su posición y alinear bien ambos motores, esto lo hice para evitar sujetar los motores con sockets frontales dado que en mi experiencia la cuerda de las perforaciones donde entran los tornillos que sujetan la unión entre el motor y el socket son muy delicados dado que son de cuerda fina y tras algún tiempo de manipulación tienden a barrerse y dejan de cumplir su función de sostener el motor firmemente.

Los soportes se sujetan a la base del chasis mediante tornillos que atraviesan todo el soporte y son al mismo tiempo la pieza que aprisiona el motor. Son soportes especiales hechos de la siguiente manera 2 soleras de 1 pulgada de ancho, 2 de alto y 1.5 de largo, puesto que el diámetro del motor es de 1 pulgada, agregué un cuarto de pulgada de cada lado. Se emplean 4 tornillos de 1/8 de pulgada y 2.5 pulgadas de largo. Se requiere una broca de 3/16 de pulgada, una broca del diámetro del motor, en este caso contraté un servicio de torno donde realizaron dicha perforación, y también emplee un punzón para marcar los puntos donde perforaría la lámina y los soportes transversalmente para atravesar los tornillos.

Los pasos que seguí para fabricar los soportes son los siguientes.

Se ponen ambas soleras juntas una contra la otra encontrando sus caras más grandes, las de largo por alto que son perpendiculares al eje del motor, una vez alineadas, se marca sobre una de ellas el centro de donde estará el centro del motor, se marcan con el punzón y martillo, se perforan con un taladro de banco empezando con una broca de 1/8 y ampliando el diámetro gradualmente con brocas de ¼, 3/8, ½ ,10/16 y por último 3/4 , el lugar donde quedará el centro del motor, se marca con un punzón y con un taladro de banco se perfora para que quede derecho se verifica con la marca del otro lado. Se abre el barreno hasta el diámetro del motor poco a poco.
Se marcan las posiciones de los 4 tornillos que sujetaran a ambas partes de cada soporte juntas después de que se corten, se perforan para los tornillos con el taladro de banco y luego se cortan a la mitad justo sobre el diámetro horizontal del y listo. Se pone el motor dentro y se sujeta con los tornillos, se hacen las perforaciones inferiores con los que se sujetará al chasis el soporte

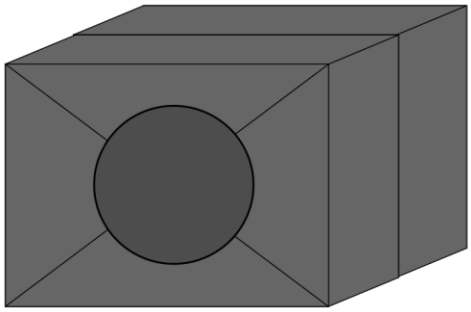


Ilustración 23 Soportes alineados

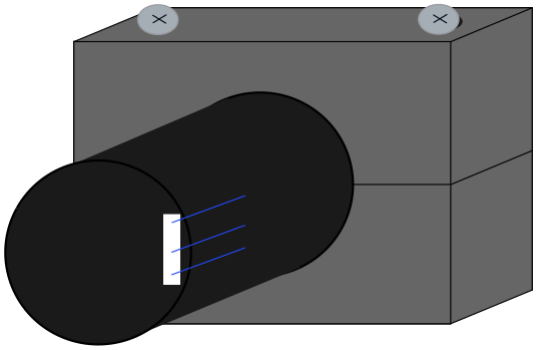


Ilustración 24 Soporte con Motor montado

6.2 Implementación de la arquitectura en la FPGA

Para la implementación del código que configura la FPGA se utilizó el lenguaje de descripción de Hardware VHDL en el entorno de desarrollo Lattice Diamond, utilizando una licencia estudiantil proporcionada por Lattice con ayuda de un breve registro en línea usando el correo institucional. Una vez ensamblados los módulos de hardware, se procedió a implementar los algoritmos definidos en la etapa de diseño. Los diagramas de la lógica de control y los flujos de datos se tradujeron en código que permitió coordinar las acciones de los diferentes subsistemas. Cada módulo fue probado individualmente, verificando que las señales enviadas y recibidas cumplían con las expectativas del diseño. Esta estructura modular propicia la integración eficiente del sistema, al eliminar falsos negativos causados por errores que yacen en otro bloque, y que por esto se complicaría hacer

La FPGA MACHX02

Se utilizó una metodología de encapsular los componentes en paquetes de manera jerárquica. El código implementado en la FPGA controló funciones críticas como la generación de señales PWM para los motores, el procesamiento de datos de los codificadores y la comunicación con la ESP32. Para garantizar la funcionalidad de cada subsistema, se realizaron pruebas individuales y luego pruebas de integración, verificando que las señales cumplieran con las expectativas del diseño.

Durante la integración, se resolvieron desafíos relacionados con la sincronización de los módulos, ajustando parámetros en el software y asegurando que las señales físicas fueran compatibles con las características del hardware. Por ejemplo, se validó que las señales de los codificadores fueran interpretadas correctamente por la FPGA, lo que permitió calcular con precisión la velocidad y posición del robot.

La estructura jerárquica de los componentes de cada entidad en la arquitectura se va poniendo dentro de un paquete de VHDL y se van usando conforme se va necesitando cada uno de ellos, muchos componentes son instanciados múltiples veces por lo que se hace uso concurrente de ellos.

6.3 Resultados

Resultados del chasis

El chasis se hizo con una lámina de aluminio que se marcó con todos sus componentes encima, se cortó, perforó, lijo, pulió y pintó. A continuación, se muestra una foto del resultado final

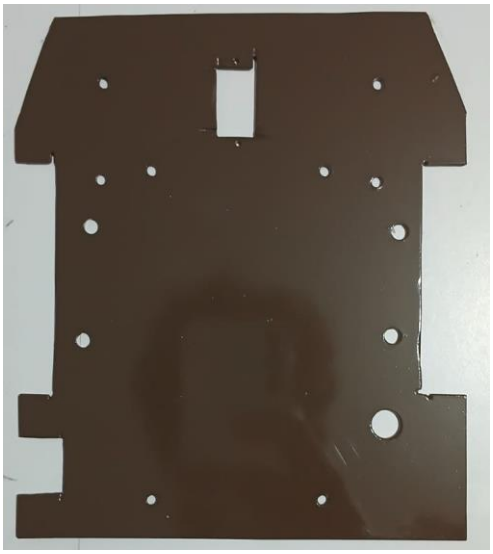


Ilustración 25 Chasis

Resultados del Armado de Soportes de motor Motores y ruedas.

Al fijar los soportes con los motores al chasis y ponerle las ruedas el resultado es el de mantener los ejes de los motores alineados gracias a las perforaciones que permiten ajustar la posición y los motores quedan bien fijos sin peligro de que ningún estrés barra o afloje ningún tornillo. Además, las ruedas están fijadas con tornillos a los ejes de las ruedas por lo que cada componente podría quitarse y cambiarse en cualquier momento en caso de descompostura. A continuación, se muestra una imagen de los motores fijados al chasis.

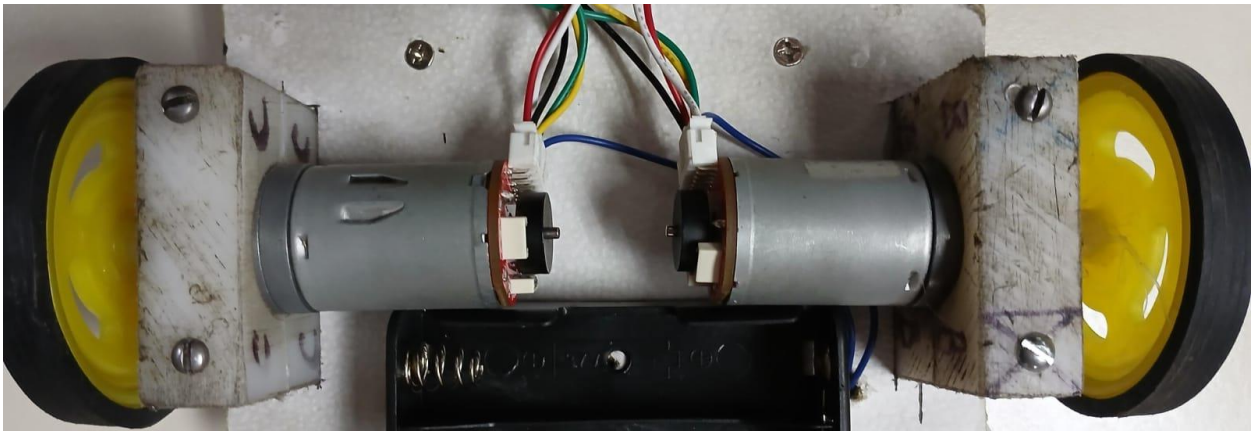


Ilustración 26 Soportes, Motores y Ruedas

Resultados del PWM

La FPGA es una solución muy eficiente para producir señales de PWM quizá está sobrada, quizá en algún punto sería bueno hacer uso de herramientas de sincronización que el proveedor definió en el producto como bloques que pueden instanciarse. El ciclo puede ajustarse para tener una frecuencia de Trabajo definida, también el ancho de pulso puede llevarse desde 0 hasta 100% en intervalos regulares producto de dividir el ciclo entre potencias de 2.

El control del servomotor con la FPGA resulta muy preciso ya que al momento de asignar una Señal de PWM bien modulada se puede controlar con precisión la ubicación del servomotor, a pesar de que el fabricante indica que la señal debe modularse entre 1ms y 2 ms para llevarlo de 0 a 180 grados, en respuesta a esto se encontró que debe ajustarse tanto el ancho mínimo como el máximo para que estos valores de la posición real sean lo más precisos posibles, con el diseño de este componente , configurado en la FPGA esto resulta sencillo y flexible.

Resultados del sensor láser

Para poder hacer uso del sensor láser hizo falta emplear un dispositivo ESP-32, para la que hay una librería que implementa los distintos estados de la maquina de estados correspondiente tanto al protocolo I2C, así como las particulares cadenas de comunicación que deben llevarse a cabo con el circuito integrado dentro del sensor para conseguir que realice muestreos de la distancia y que entregue esa información en el bus I2C para almacenarlo en una variable de programa dentro del microcontrolador. A partir del uso de esta librería se vuelve muy sencillo controlar y obtener la información requerida de distancia, lo restante es hacer llegar en 8 bits esa información hacia la FPGA de manera íntegra. El sensor queda montado en la parte superior del servomotor como se muestra en la siguiente imagen.

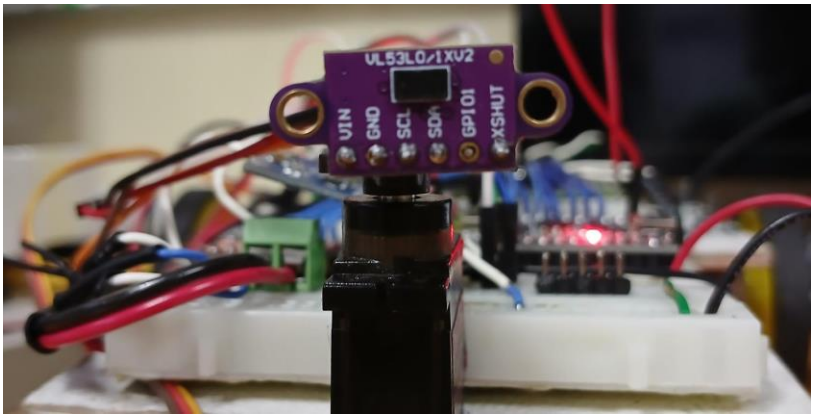


Ilustración 27 Sensor Distancia sobre Servomotor

Resultados de la Comunicación WIFI

La parte complicada de esto fue darme cuenta que la comunicación Wifi presenta fallas en la configuración e inicialización cuando la tarjeta está alimentada con menor voltaje, por alguna razón, cuando la batería llega a descargarse por debajo de cierto nivel aproximadamente cuando las baterías están entregando 3.1 volts cada una empieza a haber una falta de configuración, haría falta poder poner a disposición una señal que responda al estado de preparación del dispositivo que permita hacer uso de él cuando se sabe que está disponible puesto que se ha inicializado de manera correcta.

Resultados del uso de ESP-32

La ESP-32 fue fijada al chasis del robot y sus puertos fueron conectados de la siguiente manera, 8 bits hacia la FPGA y dos bits hacia el sensor de distancia. El resultado de esta construcción se muestra en la siguiente imagen.

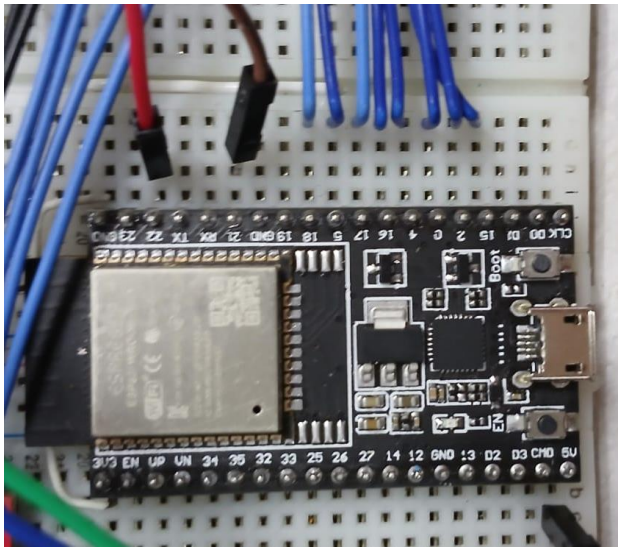


Ilustración 28 ESP-32 en protoboard

En la siguiente imagen se muestra el resultado del armado completo del primer prototipo

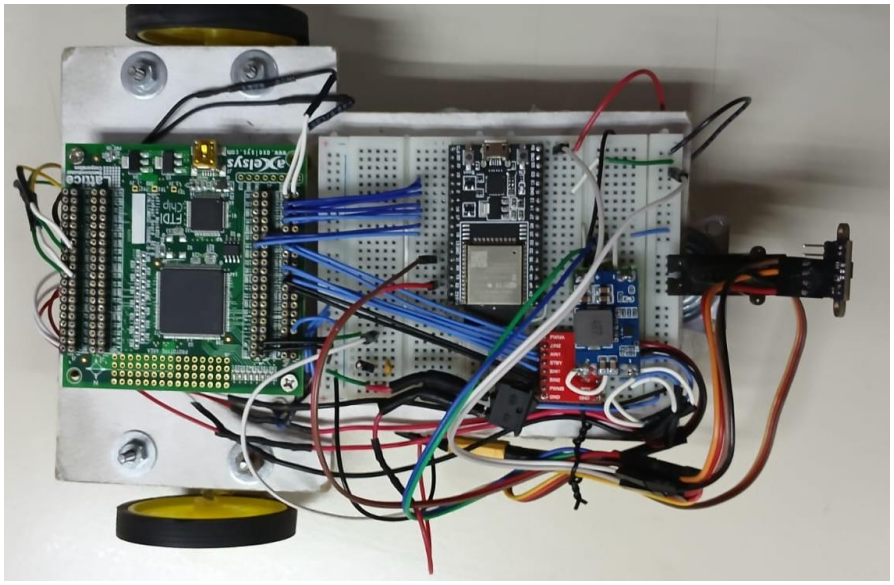


Ilustración 29 Foto del prototipo en protoboard

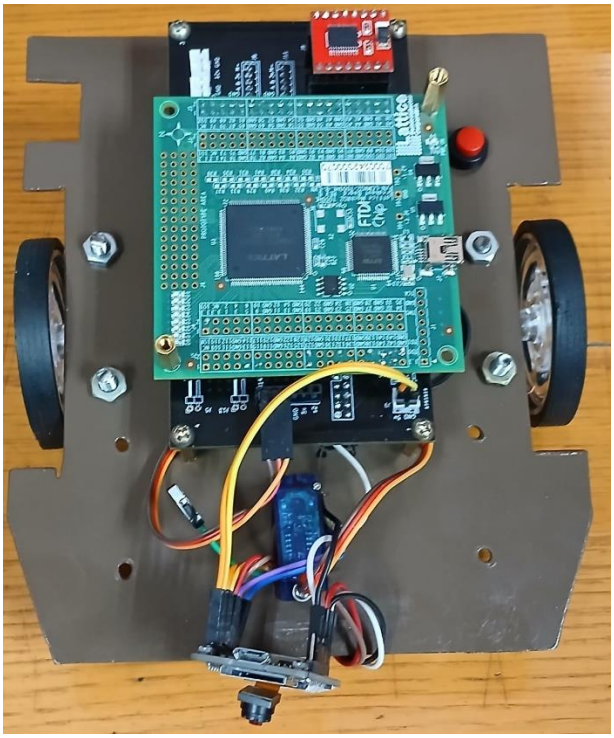


Ilustración 30 foto del prototipo en PCB

Capítulo 7 Pruebas

Cada módulo fue probado con pruebas unitarias a las que se le enfrenta aislado de otros componentes, una vez que se simula el comportamiento de la entidad, se encuentran errores en la idea principal del diseño o fallas de falta de consideración de fenómenos físicos o márgenes de tolerancia que deben considerarse en un sistema en hardware que lo último que puede hacer es tener un desempeño ideal como lo haría la composición abstracta de los algoritmos y su integración como cuando utilizamos software.

Las pruebas de los módulos configurados en la FPGA fueron realizadas en un simulador integrado dentro del paquete de Diamond y al que tenemos acceso con la misma licencia estudiantil.
Las señales dentro de cada elemento pueden ser simuladas y seguidas pulso a pulso para corroborar que el comportamiento tanto de las máquinas de estado como de los procesos y los bloques lógicos más simples es el esperado, y en caso de que no fuera así comprender lo que está pasando en el sistema y corregir esos detalles conforme van surgiendo problemas que no fueron posibles de prever durante la etapa de diseño.

7.1 Simulaciones de los componentes

Las simulaciones se realizan sobre la plataforma de ModelSim que es una herramienta a la que se tiene acceso gracias a la licencia estudiantil para Lattice Diamond ya que viene en el mismo archivo de instalación.
Durante el uso de esta herramienta se considera que todos los componentes son sintetizables, y aunque se utiliza una simulación con la frecuencia de reloj real a la que se lleva a cabo el proceso dentro de la FPGA, primero se pone el reloj a 2ps para que la simulación sea más rápida dado que solo tenemos acceso a un segundo virtual de tiempo total de simulación
En todos los casos existe una señal de restablecimiento para cada bloque, donde esta señal inicializa las variables que no están inicializadas, y establece los contadores en 0 así como el estado de las máquinas de estados al estado de reinicio, esto con el objetivo de eliminar inconsistencias en el cálculo debidas a la incertidumbre de algún dato que pudiera faltar, haciendo así fallar la simulación y las posibles consecuencias de ello.
Los bloques de las instrucciones son ligeramente distintos en este sentido dado que el código de operación que los activa, funciona como un restablecimiento en su ausencia, digamos que cuando el bloque no está seleccionado, se lleva a restablecimiento y sus salidas a alta impedancia dado que solo una instrucción a la vez puede ejecutarse puesto que es hardware lo que controlan y efectivamente es un solo comportamiento a la vez el que puede desplegar un actuador.

7.1.1Controlador Puente H

En el caso del Puente H se utiliza un generador de PWM , el generador de PWM es una entidad que a partir de un contador de pulsos de reloj que tiene una cuenta máxima igual a 2 a la n , cada pulso contado es una porción del ciclo de trabajo, primero se cuentan los pulsos respectivos a la fase activa de la señal, luego se termina de contar los ciclos restantes correspondientes a la fase inactiva, y terminados estos dos lapsos, se termina el ciclo de trabajo y se comienza de nuevo, efectivamente generando una señal cuyo ancho de pulso puede ser regulado cuando se hacen variaciones el número de pulsos que correspondan a la fase activa, número de pulsos que tendrá que contar el contador antes de hacer la transición a la fase inactiva. Se muestra una simulación del generador de PWM con un ciclo de trabajo de 2080 ciclos de reloj y con un ancho de pulso de 1/255 del ciclo de trabajo como fase activa.

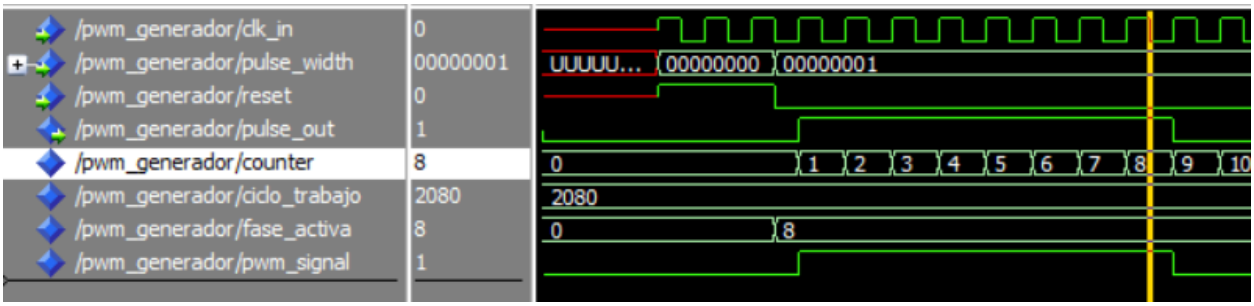


Ilustración 31Flujo de señales del Generador de PWM

Es esta primera fase, se pone un ancho de pulso de 3, dados los cálculos con un ciclo de trabajo de 2080 ciclos de reloj, 8 bits nos permiten hacer que cada nivel tenga 8 ciclos de reloj, puesto que 2080 ciclos / 255 niveles de PWM, = 8 ciclos de reloj por cada nivel. Así que la fase activa en este caso durará 3 niveles*8 ciclos = 24 ciclos, y claramente se aprecia como al llegar a los 25 ciclos, la fase activa ha terminado y la señal va a 0.

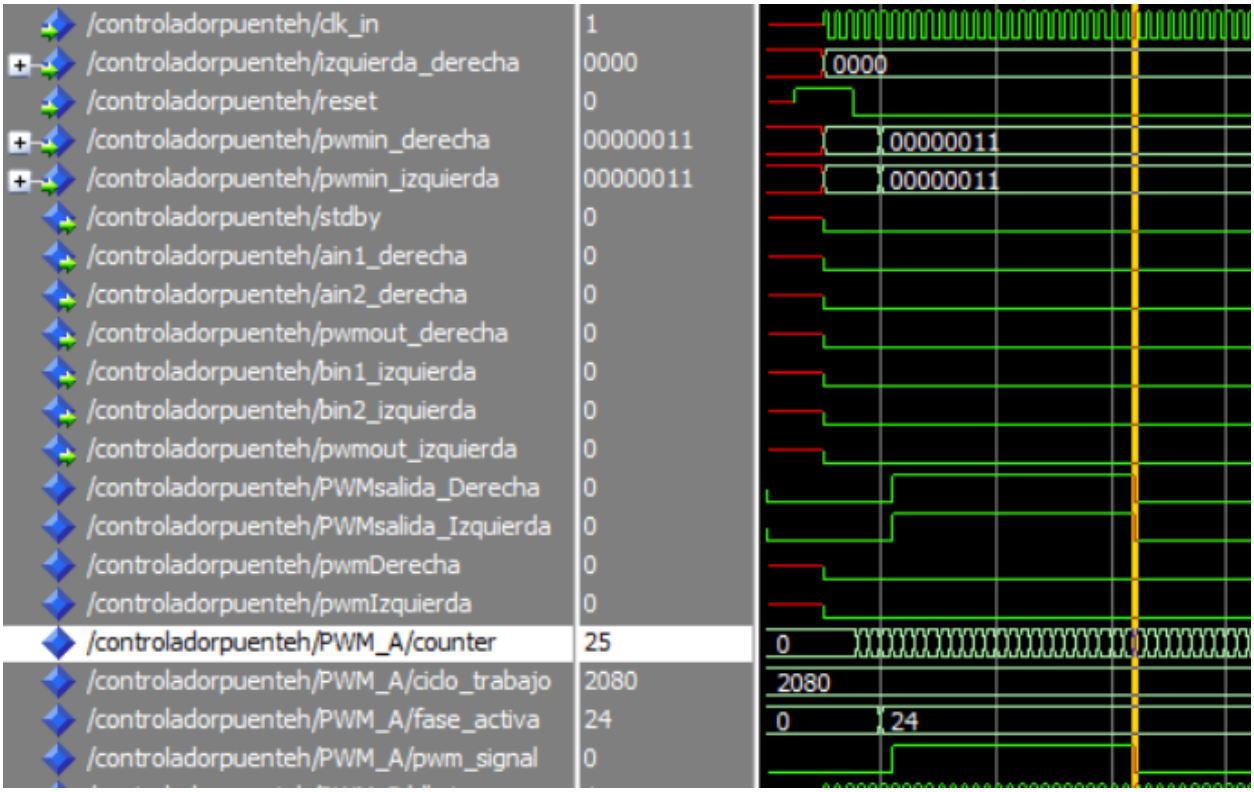


Ilustración 32 Flujo de señales del controlador para Puente H

Para confirmar ahora se simula con un ancho de pulso de 128, sabemos que 128*8 =1024 así que esta vez contará 1024 ciclos de reloj manteniendo la fase activa para después pasar a la fase no activa, nótese que el tamaño de pantalla es distinto y la señal de reloj lo revela pues parece que se hace una raya gruesa en lugar de una onda cuadrada.

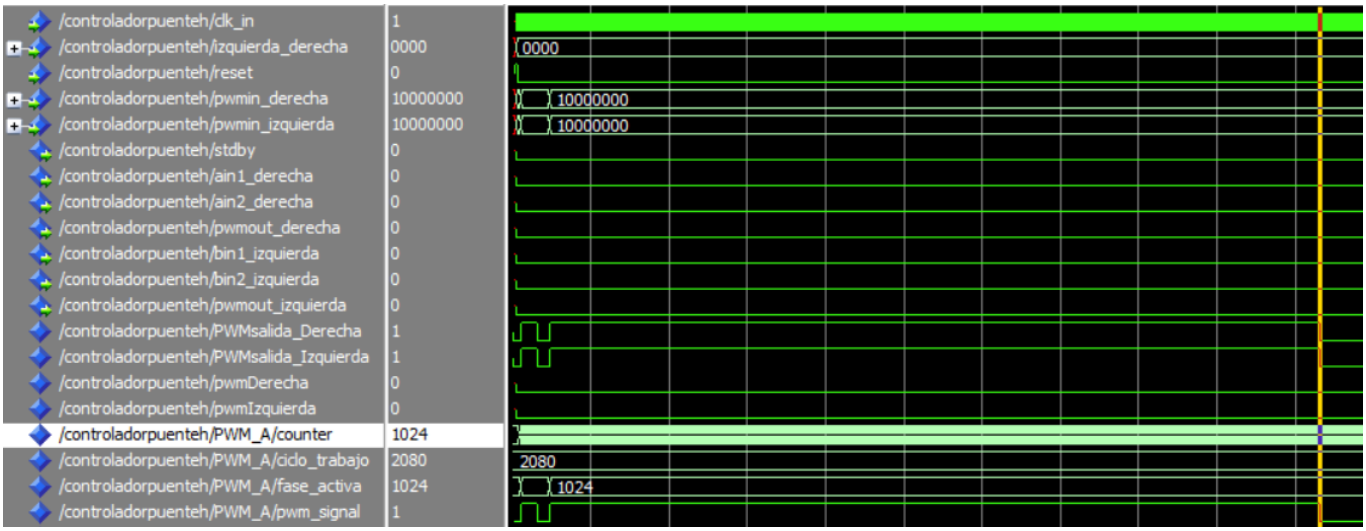


Ilustración 33 Flujo de señales 2 de controlador para Puente H

La última prueba es activar la dirección de los motores, pues, aunque hay PWM no hay un sentido de giro así que, no es suficiente, pongamos ambos motores en el mismo sentido y veamos como la salida que controla la dirección de los motores se activa por primera vez.

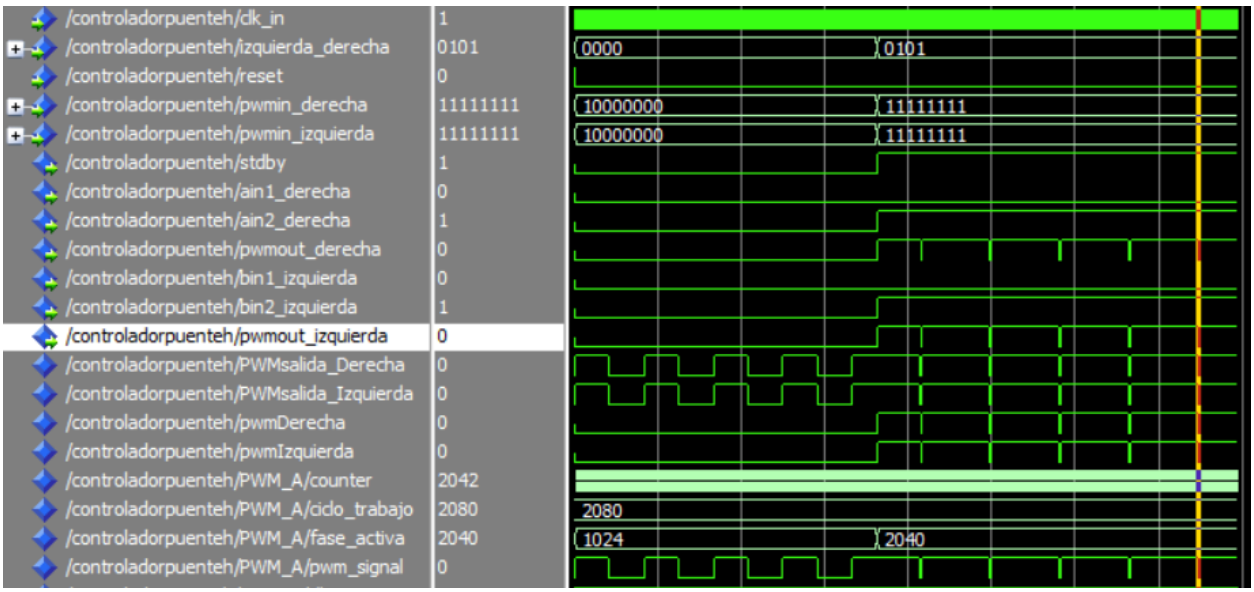


Ilustración 34 Simulación de controlador de Puente H

Ahora es un poco más evidente que la fase activa está correspondiendo a los valores que ingresamos puesto que pusimos el doble y gráficamente se aprecia el doble de tiempo en alto, igualmente al llegar a 255*8 =2040 pulsos, termina la fase activa y bueno hay una ligera discordancia puesto que el ciclo de trabajo completo es de 2080 ciclos, hay 40 ciclos de reloj que se están perdiendo debido a que la división de 2080 entre 8 no es entera. Más importante aún es notar que, aunque los generadores de PWM están operando, es hasta que se pone una dirección de giro que sale la señal de PWM, así como la de activación hacia el puente h, antes de eso no hay salida de PWM en los puertos PWMDerecha y PWMIzquierda.

Estas pruebas confirman que, en las condiciones mencionadas, a reserva de que la velocidad de reloj es distinta.

Controlador de codificadores

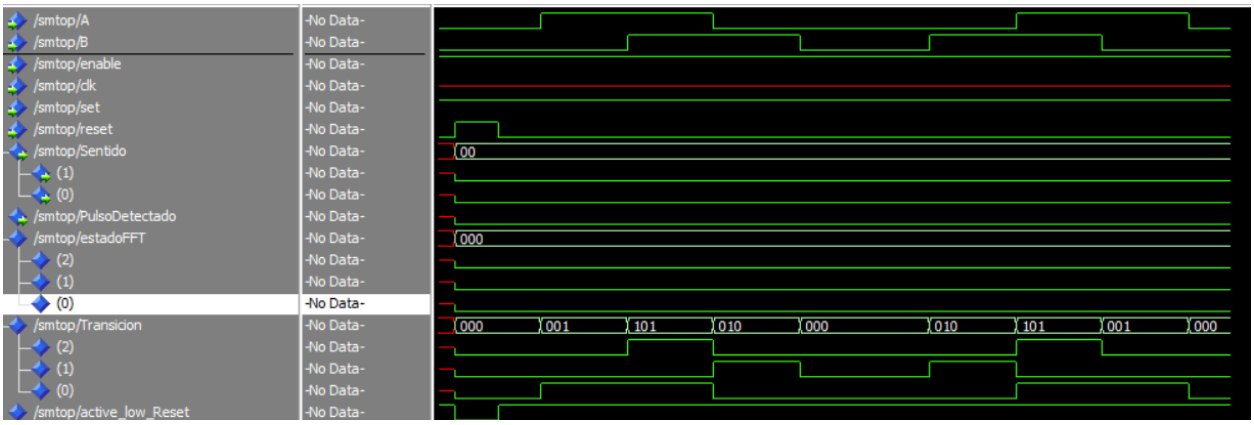


Ilustración 35 Simulación de Controlador de codificadores

La máquina de estados
La máquina tiene 7 estados así que los vamos a simular todos

Gestor de flujo de instrucciones
Cuando la memoria se encuentra cargada y lista para ser leída y el contador de memoria se ha establecido exitosamente en la dirección de la primera instrucción, se va tomando desde la más alta hasta la más baja, de una en una , cada instrucción es separada en Código de operación que tiene 4 bits y Parámetro con 8 bits, en el bus de Código de operación se pone el código de la instrucción actual y en el bus de parámetro se pone el parámetro leído en la instrucción actual, este bus se conecta con una entrada a cada bloque de instrucción, activando únicamente una instrucción a la vez, con su código respectivo. Una vez que el bloque ha sido activado, se ejecuta y lleva a cabo su finalización de manera independiente de los demás bloques, independiente de la unidad que gestiona las instrucciones e independiente de la memoria y al terminar pone un 1 en la señal de terminado que funciona como una bandera que permite al gestor de instrucciones saber cuando es el momento de finalización de la ejecución de esa instrucción y cuándo es momento de pasar a ejecutar la siguiente. Para esto la unidad gestora incrementa la dirección del contador de memoria y vuelve a hacer una lectura, repitiendo el mismo proceso que termina cuando recibe nuevamente la señal de terminación de una instrucción, emitida por dicha instrucción. A continuación, se muestra una simulación de la actividad del control de instrucciones.

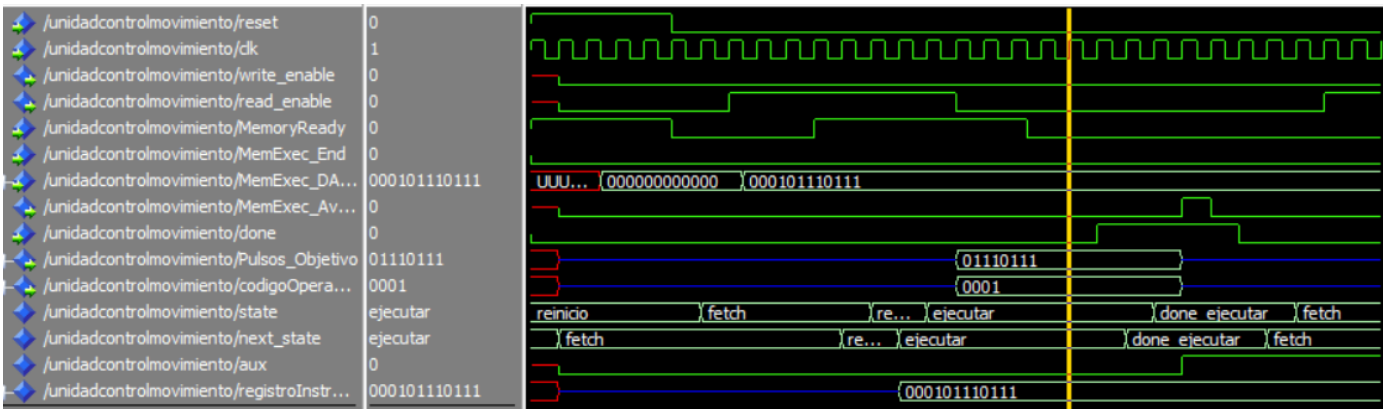


Ilustración 36 Simulación de la unidad de flujo de instrucciones

7.3 Pruebas en Prototipo

Se realizó la carga de un código que incluye todas las instrucciones diseñadas para el sistema, demostrando que cada una es ejecutable y funcional dentro del entorno controlado. Las pruebas confirmaron que el hardware responde correctamente a los comandos, permitiendo su reutilización en distintas secuencias y ciclos de operación. Aunque el sistema tiene limitaciones propias del hardware utilizado, se comprobó que las capacidades del controlador son claras y suficientes para garantizar un control estable y confiable sobre los movimientos de los robots.

Primero se pone el robot en el espacio controlado y se inicia con la instrucción de localizar y orientarse con una esquina, como se puede notar el sensor de distancia esta rotado hacia la derecha en 0 grados, dado que buscará la esquina como el punto más lejano detectado frente a él, girando el servomotor hasta 180 grados y muestreando el espacio cada 10 milisegundos. Como se muestra en la siguiente imagen.

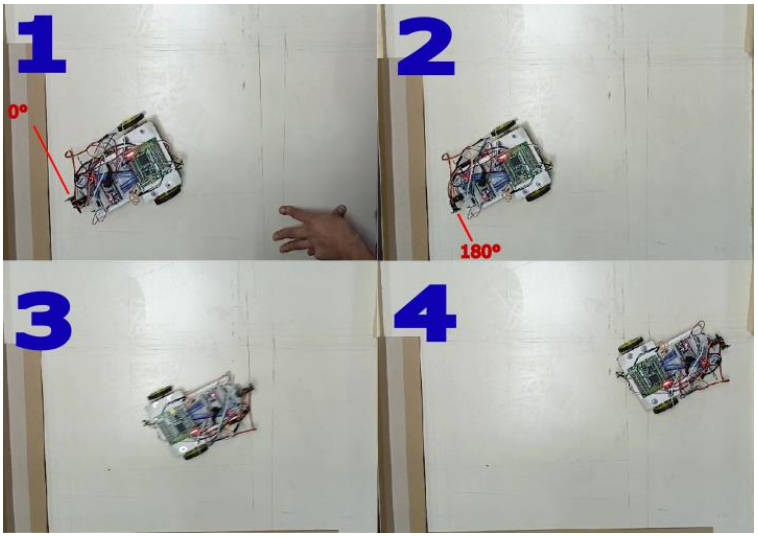


Ilustración 37 Prueba en prototipo inicio

7.4 Limitaciones

A partir de que los componentes están implementados como un diseño en hardware a partir del lenguaje VHDL surge un primer conflicto que es la ausencia de bibliotecas que contengan instrucciones como las que tendría un procesador, aunque existen y se utilizan bibliotecas de métodos aritméticos de osciladores proporcionados por el fabricante y por el estándar de la IEEE, lo que no tenemos son bibliotecas que permitan el control de ningún protocolo, memoria, divisor de reloj, registro ni prácticamente ningún código que pueda ser usado tal como es.

En esta parte de la implementación a partir del lenguaje VHDL encontré muchas complicaciones en la construcción de una máquina de estados que pudiera sincronizarse por medio del protocolo I2C a otro dispositivo, por lo que tuve que sortear esa limitación usando un microcontrolador

externo que cuenta con una biblioteca para manejar ese protocolo, y sumando a eso, el uso de la biblioteca para el manejo del sensor de distancia con láser.
Otra limitación que surgió a partir de que la complejidad de implementar el protocolo es la de proveer a la arquitectura del bloque de comunicación wifi, que, si bien puede ser implementada con módulos muy pequeños e independientes, esto supondría que debe haber una implementación del protocolo, así como del conjunto de señales como instrucciones que recibe el módulo y usa como datos en la configuración de la comunicación y durante la transmisión.

Durante la implementación del control de velocidad, fue muy complicado implementar un control diferencial integral que usara lógica de punto flotante, por lo que se pasa a unidades en notación científica y se evita el uso del punto flotante de manera directa, por ejemplo, en lugar de usar segundos, se usan milésimas de segundos para simular la aritmética con punto flotante, arriesgando así la precisión de los cálculos del control de velocidad.

Una limitación importante al momento de tener un dispositivo configurable , en lugar de un procesador con instrucciones definidas, es que justamente deben implementarse los mecanismos necesarios basándonos en señales y componentes, y no en llamadas a instrucciones, por supuesto que esta limitación es a su vez una libertad de implementar instrucciones que quizá puedan ser puntuales y sencillas, o difíciles y rebuscadas, así se pueden generar instrucciones nuevas apegadas a las necesidades de la tarea o acción específica.

En el armado del Hardware encontré una limitación bastante considerable que es el precio de desarrollar un prototipo que en mayor medida de que se pueda simular y estimar el comportamiento individual de cada componente, puede ahorrarse dinero al escoger componentes que encajen, que coincidan en sus parámetros por ejemplo de consumo y entrega de carga eléctrica.

Conclusiones

Este proyecto ha sido una experiencia intensa y enriquecedora. Me permitió adentrarme de lleno en el diseño, control y programación de un sistema robótico, enfrentándome a desafíos técnicos que pusieron a prueba mis conocimientos y mi capacidad para resolver problemas. Desde la implementación de los componentes digitales hasta el manejo de componentes electromecánicos, cada etapa del proceso requirió análisis detallado, y creatividad.

Me llevó a comprender que no solo se trata de mecánica o programación, sino de integrar múltiples disciplinas para lograr un sistema funcional. Hacer que puedan comunicarse que me acercó más a la interacción entre hardware y software. Este camino me dejó claro que la ingeniería no solo se trata de resolver problemas técnicos, sino de buscar soluciones que sean aplicables, funcionales y sostenibles en el mundo real. Más allá del trabajo en sí, me llevo el aprendizaje de que los desafíos son una oportunidad para innovar, y que cada avance cuenta.

Referencias

[1] P. Argoneto, G. Perrone, P. Renna, G. Lo Nigro, M. Bruccoleri y S. Noto La Diega, roduction Planning in Production networks, Models for Medium and Short-term Planning, Londres: Springer, 2008.

[2] J. K. T. B. M. Z. K. K. Rihard Karba, Terminological Dictionary, Denver: Springer, 2023.

[3] Lattice Semiconductor, «ICE40 UltraPlus Family Data sheet,» Lattice Semiconductor, Hillsboro, Oregon, United States, 2023.

[4] Mouser Electronics, «Mouser Electronics,» Mouser Electronics, 2024. [En línea]. Available: https://www.mouser.mx. [Último acceso: 2024].

[5] F. E. M. J. L. B. B. W. A. W. S. &. L. B. Arvin, «Mona: an affordable Mobile Robot for Swarm Robotics,» de *UK-RAS Conference on 'Robotics and Autonomous Systems*, Manchester, 2017.

[6] MIR, «mobile-industrial-robots,» Mobile-Industrial-Robots, 2024. [En línea]. Available: https://mobile-industrial-robots.com/es/productos/robots/mir1350. [Último acceso: 2024].

[7] MIR, «mobile-industrial-robots.com,» Mobile-industrial-robots, 2024. [En línea]. Available: https://mobile-industrial-robots.com/es/productos/robots/mir1200-pallet-jack. [Último acceso: 2024].

[8] BostonDinamics, «bostondynamics.com,» BostonDinamics, 2024. [En línea]. Available: https://bostondynamics.com/wp-content/uploads/2024/01/Stretch-Brochure_2024.pdf. [Último acceso: 2024].

[9] OTTO Motors, «ottomotors.com,» OTTO Motors, 2024. [En línea]. Available: https://ottomotors.com/1500/. [Último acceso: 2024].

[10] Stanford, «github.com/,» [En línea]. Available: https://github.com/ShapeLab/SwarmUI.

[11] Amazon, «aboutamazon.com,» AmazonRobotics, 2023. [En línea]. Available: https://www.aboutamazon.com/news/operations/amazon-robotics-autonomous-robot-proteus-warehouse-packages.

[12] Axiomtek, «axiomtek.com,» Axiomtek, 2024. [En línea]. Available: https://www.axiomtek.com/. [Último acceso: 2024].

[13] Axiomtek, «Axiomtek.com,» Axiomtek, 2024. [En línea]. Available: https://www.axiomtek.com/Download/Spec/robox300.pdf. [Último acceso: 2024].

[14] Walmart, «www.walmartasr.com,» Walmart, 2024. [En línea]. Available: https://www.walmartasr.com/technologies/alphabot-asrs-system/. [Último acceso: 2024].

[15] A. S. Tanenbaum, Sistemas operativos distribuidos, Prentice Hall, 1996.

[16] F. d. A. L. Fuentes, Sistemas distribuidos, Cuajimalpa: UAM, 2015.

[17] Espressif Systems, «espressif.com,» 2024. [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-reference/network/esp_now.html. [Último acceso: 2024].

[18] B. H. a. E. O. Wilson, The Superorganism: The Beauty, Elegance, and Strangeness of Insect Societies., New York: Norton & company, 2009.

[19] M. M. P. P. C. P. U. Chavan, «A Review on sftware Architecture Styles with layered Robotic Software Architecture,» de *International Conference on Computing Communication Control and Automation*, Pune India, 2015.