FACULTAD DE INGENIERIA Y CIENCIAS APLICADAS

PROTOTIPO PARA ESCANEO DE ZONAS IMPLEMENTADO A UN ROBOT AUTONOMO MOVIL PARA LIMPIEZA DE INTERIORES

AUTOR

Francisco Adrian Guasumba Tupiza

AÑO

2019

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

PROTOTIPO PARA ESCANEO DE ZONAS IMPLEMENTADO A UN ROBOT AUTONOMO MOVIL PARA LIMPIEZA DE INTERIORES

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información

Profesor Guía

Mg. Héctor Fernando Chinchero Villacís

Autor

Francisco Adrian Guasumba Tupiza

Año

2019

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Prototipo para escaneo de zonas implementado a un robot autónomo móvil para limpieza de interiores, a través de reuniones periódicas con el estudiante Francisco Adrian Guasumba Tupiza, en el semestre 201910, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Héctor Fernando Chinchero Villacís

Magister en Domótica

C.I. 1715451330

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Prototipo para escaneo de zonas implementado a un robot autónomo móvil para limpieza de interiores, del estudiante Francisco Adrian Guasumba Tupiza, en el semestre 201910, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Luis Santiago Criollo Caizaguano

Magister en Redes de Comunicaciones

C.I. 1717112955

DECLARACIÓN DE AUTORIA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”

_____

Francisco Adrian Guasumba Tupiza

CI.1753059441

AGRADECIMIENTOS

Quiero agradecer toda mi familia que siempre estuvo apoyándome en cada decisión dándome ánimos en los momentos más difíciles.

A mis amigos más allegados que siempre me han dado su apoyo dentro y fuera de mi vida universitaria.

Y a mis profesores que me han educado a lo largo de estos años dándome el conocimiento y la guía necesaria para poder culminar con éxito este trabajo.

Les agradezco, y hago presente mi gran afecto hacia ustedes mi inmensa familia.

DEDICATORIA

Quiero dedicar este trabajo a todas las personas que me brindaron su apoyo a lo largo de los años, a mi familia, profesores amigos cercanos y lejanos que fueron construyendo el pilar de mi fortaleza en el transcurso de mi carrera, a todos aquellos que no se apartaron de mi en los momentos más difíciles y me apoyaron para que continúe con más esfuerzo y dedicación.

RESUMEN

En el presente proyecto se desarrolla la implementación de un algoritmo para escaneo de zonas en un prototipo de robot autónomo de limpieza, para lo cual es necesario realizar una investigación previa y análisis de las numerosas tecnologías que dan solución a aplicaciones robóticas. Se toma en cuenta las ventajas y desventajas que dichas tecnologías poseen con la finalidad de obtener la mejor solución posible para el desarrollo de algoritmo para evasión de obstáculos que será incorporado al prototipo de robot de limpieza.

Para la creación del algoritmo fue necesario la recolección de información del entorno en el que el prototipo trabaja, para, acorde a sus requerimientos será implementada la mejor solución.

En cuanto el prototipo de limpieza simula el movimiento de un robot autónomo móvil que limpia constantemente la zona de trabajo que se ha establecido, se realizan pruebas de funcionamiento para la demostración más concreta del algoritmo que permite que trabaje con eficiencia.

Al final se analizan los resultados para obtener la información de los datos obtenidos durante la realización de la implementación y la prueba del algoritmo en el prototipo.

ABSTRACT

In the present project the implementation is developed of an algorithm for scanning of zones in a prototype of autonomous cleaning robot, for which it is necessary to carry out a previous investigation and analysis of the numerous technologies that give solution to robotic applications.

We consider the advantages and disadvantages that we provide to achieve the best possible solution for the development of algorithm for the elimination of obstacles that was incorporated into the cleaning robot prototype.

For the creation of the algorithm it was necessary to collect information about the environment in which the prototype works, according to its requirements the best solution will be implemented.

As to the cleaning prototype simulates the movement of a mobile autonomous robot that constantly cleans the working area that has been established, performance tests are performed for the most concrete demonstration of the algorithm that allows it to work efficiently

At the end, the results are analyzed to obtain the information of the data obtained during the realization of the implementation and the test of the algorithm in the prototype.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. CAPITULO I. INTRODUCCION

En la actualidad es mucho más común encontrarnos dentro del hogar con dispositivos electrónicos capaces de suplantar tareas que normalmente las personas realizamos.

En este contexto, aparece la palabra domótica, referida a la ciencia y a los elementos desarrollados por ella que proporcionan algún nivel de automatización o automatismo dentro de la casa; pudiendo ser desde un simple temporizador para encender y apagar una luz o aparato a una hora determinada, hasta los más complejos sistemas capaces de interactuar con cualquier elemento eléctrico de la casa, en los ámbitos de seguridad, comunicación, control de dispositivos y limpieza automatizada (Tejedor, 2003).

Ya desde los años de 1950 también se empieza a conocer el concepto de robot especializado en una sola tarea partiendo de ideas básicas donde gracias a la ciencia ficción, se tenía la idea de familias que eran asistidas por sus robots domésticos cuando estaban en su casa.

En la actualidad el desarrollo que se ha ido dando por parte de la tecnología ha ido contribuyendo para que los robots domésticos vayan cumpliendo su labor más eficientemente, procesar mejor la cantidad de datos y realizar tareas más precisas gracias a su incorporación de sensores cada vez más modernos.

El desarrollo de distintos modelos de robots domésticos comenzó a crecer llegando al mercado a fines del siglo pasado. En 2002, varios prototipos de limpieza básicos empiezan a lanzarse, varias empresas han incursionado en la construcción de robots asistentes es por lo cual existen muchos modelos de robots domésticos como: El *Roomba* de *iRobot*(Robot aspiradora que tiene múltiples versiones desde 2002), *Scooba* de *iRobot*(Robot fregador de piso), *Lawnbott*(Robot cortador de césped.), *Droplet* (Robot regador de jardín), *Spotmini* de Boston Dynamics(Robot cuadrúpedo altamente estable que puede recoger y trasladar cosas) entre otros (Tedeschi, 2014).

En la actualidad es posible acceder a estas tecnologías con mayor facilidad, una de las prácticas donde se puede aplicar estos conocimientos es en robots de servicio para limpieza del hogar e interiores, aplicando lenguajes de programación como C++ y hardware de distintos fabricantes como Arduino para construir prototipos de dispositivos que puedan registrar y controlar datos del mundo real para una aplicación real (Arduino, 2018).

1.1 Alcance

El presente trabajo de titulación está enfocado a la implementación de un prototipo de un robot autónomo móvil de limpieza evasor de obstáculos, el algoritmo a usar se basa mediante la toma de datos a partir del principio de la física de medir la distancia gracias al tiempo y la velocidad que se tarda entre el envío y la recepción de un pulso sonoro al chocar con un objeto y gracias al conjunto de estos datos determinar el área del borde de un objeto para que mediante el método de restricción de distancias el prototipo evite colisionar, el prototipo será diseñado para interiores dentro de zonas con suelos planos, esto quiere decir que no contempla que este dispositivo pueda bajar o subir obstáculos como gradas o muebles.

Este prototipo móvil contará con un algoritmo de escaneo de una zona por medio de sensores de proximidad (sensor ultrasónico HC-SR04) que, a través de un servomotor (*Futaba*-S3003 de 3 Kg de torque), nos permita escanear la zona alrededor de 15 a 165 grados frente al prototipo y recopilar la distancia de cada grado para así evitar colisionar con objetos que se encuentren cerca de un rango determinado de distancia.

En segunda instancia se implementará un sistema limpieza para suelos donde a través de una bomba de agua formada por un motor de bajo voltaje (funcionando a 7.4 V) y un servomotor (*Robotek* MS-311 de 3.3Kg de torque) para el mecanismo de secado, humedecerá la zona y la secará posteriormente.

El sistema de movimiento del prototipo de robot autónomo se encontrará conformado por 4 motores de bajo voltaje alimentados de una fuente independiente (tipo lipo de 2 celdas o batería externa de 7.4 V).

Para analizar los resultados y visualizar la información que recibe el prototipo se implementará la plataforma Processing 3.14 mediante un gráfico en tiempo real, la cual nos permitirá visualizar los contornos de los objetos detectados por el escaneo del sensor de distancia de 15 a 165 grados

1.2 Justificación

El presente proyecto busca reemplazar tecnologías usadas comúnmente en la detección de objetos como sistemas de láser o sensores fijos y contacto físico, a sistemas de detección de obstáculos de tipo escaneo mediante el uso de sensores ultrasónicos, el cual es más eficiente y necesita de menos componentes.

Este proyecto ayuda a disminuir el trabajo realizado por una persona, dando la facultad de disminuir el tiempo que una persona emplea en tareas simples como trapear un suelo, siendo usado más específicamente como *gadget* de hogar o interiores, gracias a este prototipo una persona ahorrará tiempo y se dedicará a tareas más importantes.

Se busca mediante el análisis de los resultados, la precisión y el tiempo que tarda el prototipo en el procesamiento de datos para el reconocimiento de obstáculos y evasión, para así determinar la factibilidad de incorporar un sistema de evasión de obstáculos de tipo escaneo en lugar de los clásicos métodos de detección de objetos mediante contacto físico o sensores fijos.

1.3 Objetivos

1.3.1 General

Implementar a un robot autónomo móvil un sistema de evasión de obstáculos de tipo escaneo mediante sensores ultrasónicos que permitan al prototipo

evitar colisionar con objetos mientras realiza la función de limpieza de entornos domésticos de suelo plano.

1.3.2 Específicos

- Evaluar algoritmos de escaneo y visualización mediante las plataformas Processing 3.4 y Arduino 1.8 para un sistema de evasión de obstáculos.
- Implementar un sistema evasor de obstáculos en un prototipo de robot autónomo móvil para limpieza de interiores.
- Realizar un análisis de los resultados obtenidos del comportamiento del modelo de algoritmo de escaneo de evasión de obstáculos.

2. CAPITULO II. MARCO TEORICO

2.1 Introducción a los gadgets de limpieza y evaluación de mecanismos para evasión de obstáculos.

En tiempos actuales los recursos para crear objetos tecnológicos son cada vez más accesibles e intuitivos, es así como la población puede crear objetos tecnológicos que satisfagan las necesidades que se requieren (Bravo Sánchez & Forero Guzmán, 2012).

La amplia gama de accesorios y dispositivos tecnológicos permiten crear un sin límite de aparatos con distintas funcionalidades. (Baturone, 2005)

Actualmente se busca crear dispositivos capaces de moverse independientemente en un espacio físico determinado, simulando la cotidianeidad de un ser humano, se busca lograr en los dispositivos tecnológicos un movimiento natural para que sea cada vez más fácil interactuar con las personas. (Pérez V. D., 2015)

Dependiendo de la funcionalidad del dispositivo se pueden implementar distintos mecanismos de reconocimiento de objetos para evitar colisionar en el entorno trabajado.

Los *gadgets* de limpieza han sido creados a partir de la accesibilidad de tecnología robótica, para ayudar en el hogar, usando sensores junto a distintos mecanismos de movimiento para robots móviles y algoritmos de evasión logran cumplir una tarea específica dentro del hogar (Maestre, 2015).

2.1.1 Limpieza en robots:

La función de limpieza en los robots debe trabajar en conjunto con su movimiento, en el mercado tenemos varios tipos de limpieza, que va desde aspirado, barrido hasta humedecer y secar dependiendo del tipo de suelo. (Barquin, 2007)

Scooba 650 de *iRobot* tiene una dimensión de 34 cm x 34 cm x 9,2 cm con 36 cm de diámetro con un peso de 3.6 kg con una autonomía de 2 horas y su sistema de limpieza que consiste en las siguientes fases:

- Leve aspiración y mojado previo: *Scooba* trabaja con un voltaje alimentación 120-220 voltios y una potencia de 33 vatios con el cual prepara el suelo al aspirar ligeramente el polvo y la suciedad, al tiempo que humedece el suelo con una fina capa de agua o solución de limpieza.
- Fregado y leve aspiración: Un cepillo de fregado gira a más de 600 rpm para eliminar manchas.
- Cepillado de Acabado: Termina con un ciclo final de limpieza con un cepillado incorporado que tiene un ángulo de 27 grados para eliminar residuos. (iRobot Corporation, 2014)

El sistema de limpieza de *Braava Jet* tiene unas dimensiones de 17 cm x 17.8 cm x 8.4 cm un peso de 1.2 kg y usa un voltaje de alimentación de 120 - 220 voltios para la recarga con una autonomía de 2 horas . (IRobot Braava, 2013)

Para su correcto funcionamiento se coloca a 30 cm de las paredes y limpia de 20-25 m^2 donde existe un espacio del 25 % ocupado por objetos (figura.1) (Roomba, 2018).

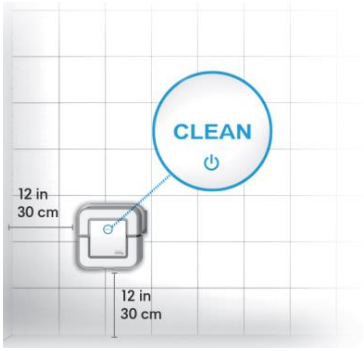


Figura1. Posicionamiento *Braava Jet*.

Tomado de (iRobot Corporation, 2017).

Para la limpieza usa un modo de fregado que consiste en humedecer la zona, donde ocupa un tanque de 0.15 lt de capacidad con una potencia de consumo de 7.1 - 33 vatios durante el limpiado y secado mientras avanza en línea recta, como se muestra en la (figura.2) (iRobot Corporation, 2017).

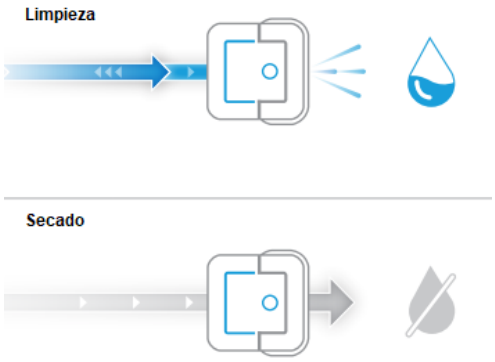


Figura2. Modo de Limpieza *Braava Jet*

Tomado de (iRobot Corporation, 2017).

2.1.2 Estructura de soporte

Los chasis son fabricados de distintos materiales dependiendo la necesidad en cuanto a resistencia y peso.

Existen en el mercado chasis prefabricados para robots móviles de ruedas, robots humanoides y drones de materiales tipo plástico impreso en impresoras 3D, fibra de carbono, fibra de aluminio, tubos PVC, placas de fibras de madera y acrílico (Butiax Robotica Educativa, 2016), como se muestra en la (figura.3).

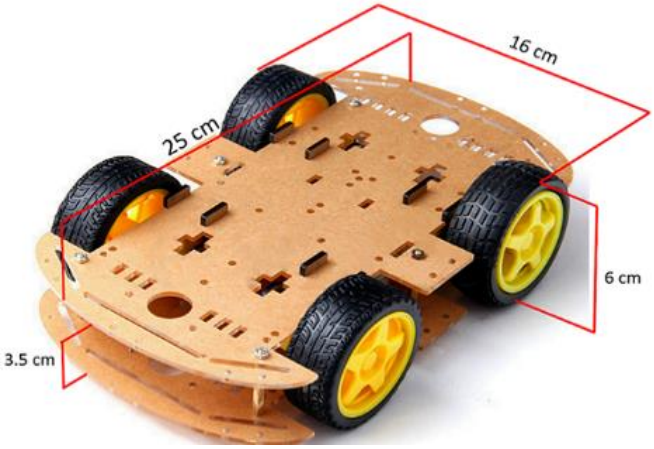


Figura3. Chasis prefabricado de Acrílico

Tomado de (spanish.alibaba, 2018).

Descripción: Chasis de acrílico prefabricado de dimensiones: 25 cm x 16 cm x 6cm

El acrílico es uno de los materiales plasticos mas usados para proyectos de electonica por sus distintas propiedades con respecto a otros elementos, y hay que tomar en cuenta sus carácterísticas dependiendo del proyecto a realizar.

Las distintas propiedades que posee el acrílico se detallan en la (tabla.1).

Tabla.1

Propiedades Mecánicas del Acrílico

Propiedad	**Medida**
Absorción de agua (24h)	0.03%
Tensión	92 mpa
Coeficiente de ruptura al estirar	760 kg / cm2
Coeficiente de ruptura al doblar	1050 kg /cm2
Coeficiente de elasticidad	28000-32000 kg /cm2
Temperatura límite de operación	80 grados Celsius
Temperatura de termoformado	140-180 grados Celsius
Dureza Barcol	50
Dureza Rockwell	M 90-100

2.2 Evasión de obstáculos en la robótica

Desde que se crearon robots móviles, siempre hubo la necesidad no solamente de moverlos de un punto a otro, sino también de realizar un movimiento inteligente que les permita interactuar con el espacio físico donde trabajan (Patricia Quintero Alvarez, 2016).

Por tal motivo se han creado distintos métodos de anticolisión y planeación de rutas, los cuales serán presentados a continuación.

2.2.1 Modelos de planeación de rutas

Estas técnicas establecen rutas que se definen como trayectorias para que el robot móvil evite obstáculos de su entorno mientras se encuentre en movimiento.

2.2.1.1 Grafo de visibilidad

Este método se aplica a espacios de trabajo bidimensionales, y con regiones poligonales, podemos denotar los puntos Qstart y Qgoal los cuales son unidos por medio de trayectorias entre vértices creados al azar.

Los polígonos creados resaltados de fondo negro representan los obstáculos a través de la trayectoria.

La ruta que une la posición inicial Qstart con respecto a la posición final Qgoal se calcula con el algoritmo de Dijkstra.

Este algoritmo determina cual determina la ruta más corta desde un nodo inicial hasta otro dentro del grafo pasando por V1,V2 y V3 que forman un camino A el cual se considera el camino óptimo de Qstart a Qgoal (figura.4).

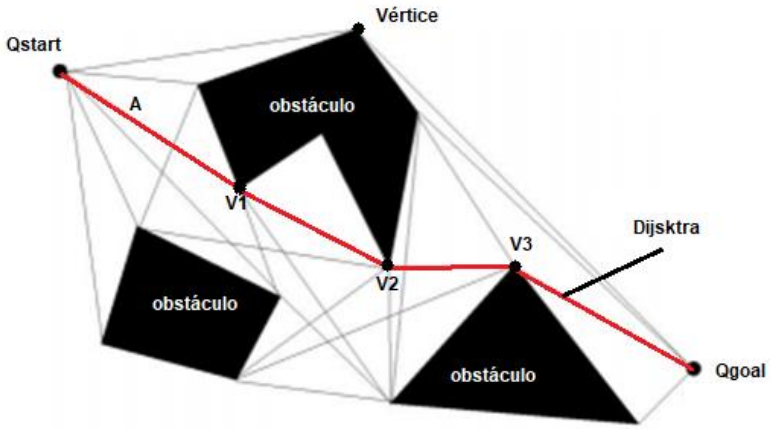


Figura4. Grafo de visibilidad

Tomado de (Daniel Estiven Álvarez Gaviria, 2012)

2.2.1.2 Diagramas de Voronoi

El diagrama de Voronoi es un método que consiste en obtener un conjunto de curvas unidimensionales que maximizan el espacio libre, este diagrama está compuesto de por lo menos dos regiones, por lo tanto, está formado de arcos los cuales pueden ser de dos tipos (Pérez D. R., 2004) :

- El arco recto: que es el conjunto de configuraciones posibles más próximas al par (lado, lado) o (vértice, vértice).
- El arco parabólico: que es el conjunto de configuraciones posibles más próximas al mismo (lado, vértice).

Entonces podemos calcular el diagrama de Voronoi suponiendo que todos los puntos con valor 0 pertenecen al espacio libre y aquellos con valor 1 pertenecen a obstáculos.

Aun así, para simplificar cálculos solamente son considerados como obstáculos los puntos correspondientes a los bordes (B) de las regiones que representen los obstáculos (Marras, 2015).

En la (figura.5), se observa el diagrama de Voronoi aplicado a un conjunto de objetos geométricos representados con líneas de color negro (B), en azul, el

espacio libre, y en rojo (R) el camino donde intersecan, formada de arcos rectos y parabólicos que muestran el camino posible para evitar colisión.

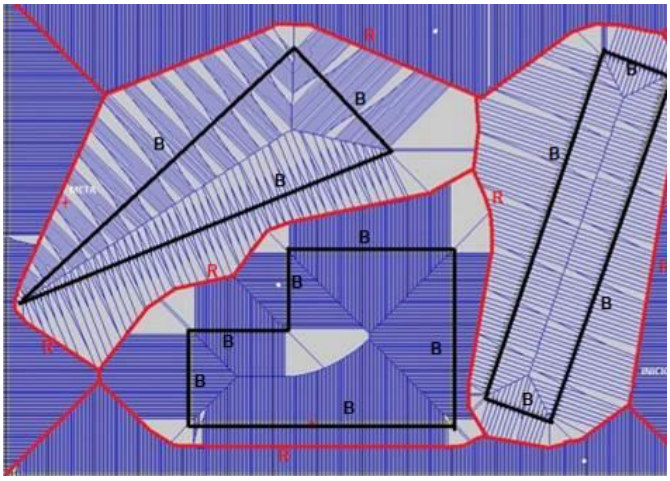


Figura5. Diagrama de Voronoi Aplicado

Tomado de (Marras, 2015).

2.2.1.3 Modelado espacio libre y restricción de distancias

Este método consta de planificar el movimiento a través de CRG (cilindros rectilíneos generalizados), el objetivo de usar CRG es hacer reconstruir un camino donde se pueda mover un objeto evitando obstáculos en el camino (Aracely Yandún, 2012).

Los CRG se da a partir de aristas de distintos obstáculos que se presentan en un área, se cumplen ciertas condiciones para que las aristas de un objeto formen un cilindro generalizado (Torres, Navegacion y Planificacion de Rutas, 2018):

- La arista ***1ai*** está contenida en una recta que divide al plano en dos regiones.
- La arista ***2aj*** debe yacer por completo en la región opuesta en la que se encuentra situada ***b1***. Este criterio es simétrico.
- El producto escalar de los vectores normales con dirección hacia el exterior del obstáculo que contiene cada arista debe resultar negativo.

Sabiendo esto para construir el CRG se debe calcular la bisectriz del ángulo α formado por el corte de las aristas ***1aj*** y ***2aj*** y construir un eje, se forman rectas paralelas al eje con inicio en los vértices de las aristas ***1aj*** y ***2aj.***

Repitiendo el proceso se construye una red de CRG en torno al robot que modela el espacio libre, el dispositivo navegará por el eje del ángulo α entre el obstáculo ***b1*** y ***b2***, así mismo podrá calcular el eje entre ***b1*** y ***b4***, ***b2*** y ***b3***.

El paso de un CRG a otro podrá ser admitido siempre que los ejes se intercepten y la intersección del rango de orientaciones admisibles en el punto de corte de ambos ejes no sea nulo (figura.6) (webpersonal, 2018).

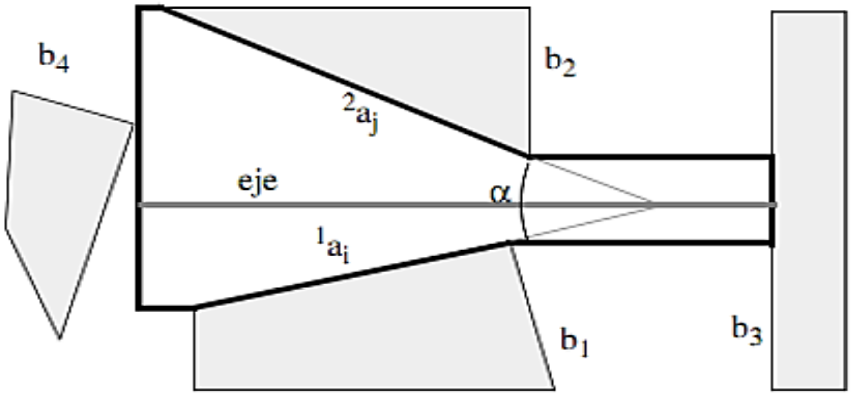


Figura6. Modelado de espacio libre

Tomado de (Torres, Navegacion y Planificacion de Rutas, 2018).

2.2.1.4 Descomposición en celdas

Este método consiste en dividir el entorno en un conjunto de celdas en el espacio libre cuyos puntos se encontrarán en los puntos medios de la sucesión de celdas que se encuentran descompuestas en trapezoides (Miralles, 2016).

La descomposición de trapezoides se da a partir de líneas verticales paralelas (L1,L2,L3…Ln) que cruzan por los vértices (v1,v2,v3…vn) de un objeto (O) que corresponderá a un obstáculo dentro del plano (figura.7) (Carlos Andrés Vélez Carvajal, 2012).

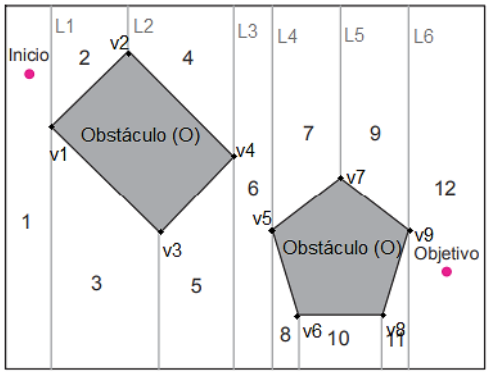


Figura7. Descomposición en Trapezoides

Modificado de (Carlos Andrés Vélez Carvajal, 2012)

La recta resultante pasa por el corte medio (pm1,pm2,pm3…pmn) de cada una de las líneas paralelas verticales, genera un grafo a partir de los puntos medios de cada línea (L) borde de las celdas creadas a partir de las aristas de los objetos que se encuentran en el camino (figura.8) (Roncancio, 2017).

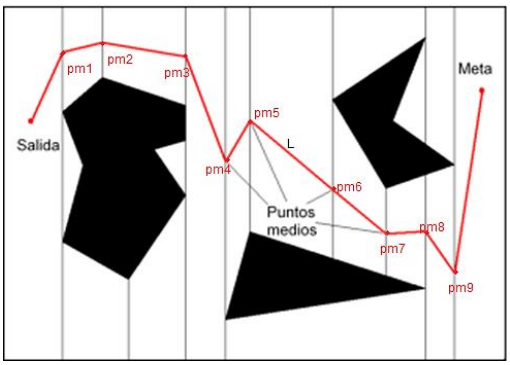


Figura8. Descomposición en Celdas

Modificado de (Roncancio, 2017) .

2.2.2 Herramientas de navegación autónoma

La navegación autónoma se refiere a la percepción del entorno a través de los distintos tipos de sensores (laser, estereovisión, tiempo de vuelo), se puede modelar un entorno, y usar técnicas de fusión de datos de sensores de navegación y control del movimiento por métodos de rutas de trayectoria (Blanco Abia, 2014).

2.2.2.1 Bumpers

Los *bumpers* conmutadores de dos posiciones donde actúa un 0 o 1 lógico, estos dispositivos son incorporados a una barra límite que permite el cambiar de estado al elemento (figura.9) (López A. M., 2015).

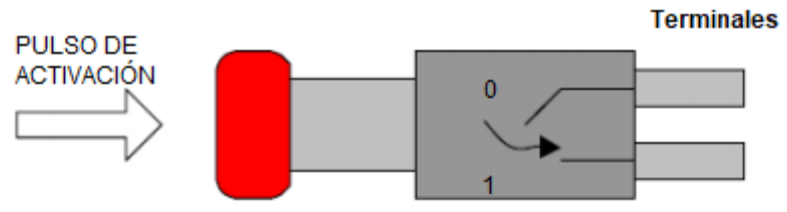


Figura9. Switch de activación

Modificado de (Oter, 2018)

El estado 0 el cual permanecerá activo mientras la barra límite no sea activada (figura 10), cuando la barra límite sea activada por medio de contacto por lo que el estado del *Bumper* cambiara a 1 lógico con un objeto, los conmutadores se activarán enviando una orden al dispositivo móvil haciendo que cumpla una determinada función (Guillermo Barbadillo Villanueva, 2012).

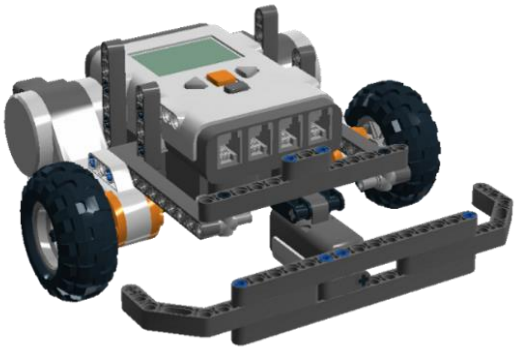


Figura110. Robot autónomo móvil con *Bumper*

Tomado de (Parker, 2018).

2.2.2.2 Detectores ultrasónicos.

Los detectores ultrasónicos son sensores que miden la distancia en función de ondas ultrasónicas desde 2 cm a 400 cm (Microkits, 2018).

El sensor HCSR-4 cuenta con 4 pines de conexión: Alimentación (+*Vcc*), *Trigger* (*Trig*) entrada (*input*) del sensor (TTL), *Echo* (*echo*) salida (*output*) del sensor (*TTL*) y *GND* (figura.11).

Figura11. Sensor ultrasónico

Tomado de (tresdprinttech, 2018)

El del sensor emite ondas que son reflejadas desde el objeto con el cual chocan, estos sensores miden la distancia en función al tiempo entre la emisión y recepción de la onda ultrasónica (figura.12) (Elecfreaks, 2018).

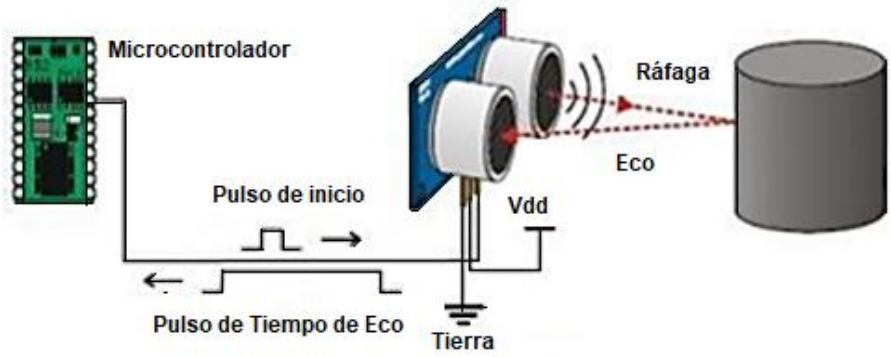


Figura12. Funcionamiento básico *HC-SR04*.

Tomado de (Flores, 2018).

Se suministra un pulso de un tiempo de 10us para iniciar el disparador, posteriormente inicia una ráfaga de ultrasonido de 8 ciclos a 40 kHz donde se produce un eco al encontrar un objeto y el sonido de rebote es detectado por el

receptor del sensor donde a continuación el pin echo del sensor cambia a estado alto (figura.13) (Fuentes, 2014).

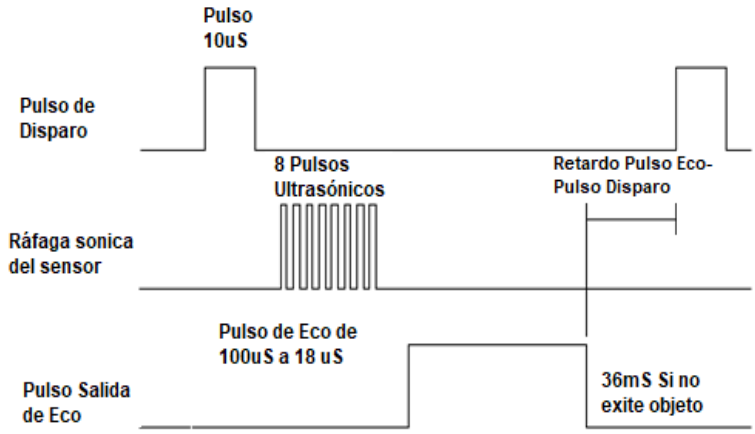


Figura13. Tiempos de funcionamiento *HC-Sr04.*

Tomado de (Cetronic, 2018)

Para poder calcular el rango o distancia a través del tiempo entre el pulso de la señal de disparo y la señal de eco recibida en cm se usa la ecuación:

$$Distancia\,(cm) = uS\ /58 \qquad \text{(Ecuación 1)}$$

Para el cálculo en pulgadas se utiliza la ecuación:

$$Distancia\,(cm) = uS\ /58 \qquad \text{(Ecuación 2)}$$

Expresando el rango en función del tiempo alto de la señal y la velocidad del sonido(340m/s) se usa la ecuación:

$$Rango(m) = Tiempo\ nivel\ alto\ (s) * \frac{340m}{s}/2 \qquad \text{(Ecuación 3)}$$

Las características que ofrece el dispositivo sensor son:

- Las ondas ultrasónicas pueden reflejarse en objetos transparentes como vidrio plástico o líquido.

- Son resistentes a la niebla o agentes externos como la suciedad, estos agendes no afectan de manera considerable al desempeño del sensor
- Estos tienen detección de presencia estable, puede obtener una medición de objetos de forma compleja tales como resortes o mallas.
- El ángulo de medición es de hasta 15 grados.
- Posee una zona ciega cuando las mediciones son menores 1cm y mayores 400 cm.

2.2.2.3 Sensores Infrarrojos

Los sensores infrarrojos son sensores de medición de distancia los cuales se basan en un sistema de emisión y recepción en el rango de espectro de frecuencia de infrarrojos (SparkFun, 2018).

Su funcionamiento esta dado por medio de la triangulación del haz de luz colimada, donde se puede estimar la distancia de un objeto físico a través de la cantidad de luz recibida que rebota en un objeto (Córdoba, 2016).

Consta de un led infrarrojo emite un haz de luz pulsada con una longitud de onda de 850 nm +/-70 nm, y un detector de posición que recibe la luz reflejada del objeto dentro del rango de distancia del sensor (figura.14) (Sharp Corporation, 2011).

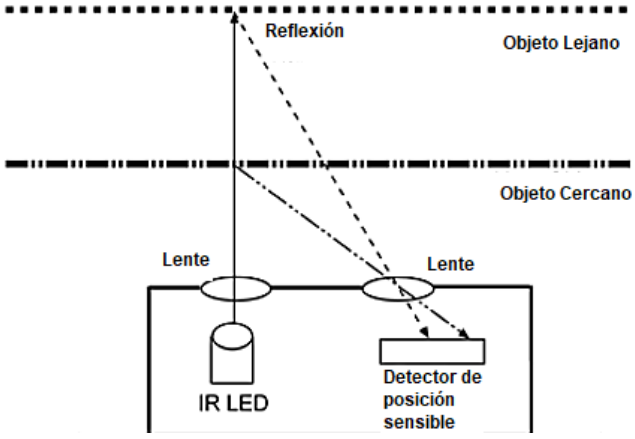


Figura14. Triangulación de luz.

Tomado de (Sharp Corporation, 2011).

La tensión de salida varía dependiendo del tipo de sensor infrarrojo, un sensor infrarrojo *Sharp 2y0A21* tiene una alimentación de 5 voltios con una tensión de salida entre los 0,3 y 3,1 voltios, la lectura es no lineal y produce voltajes de respuesta erróneos si la distancia medida es menor a 5 cm (figura.15) (Álvarez, 2016).

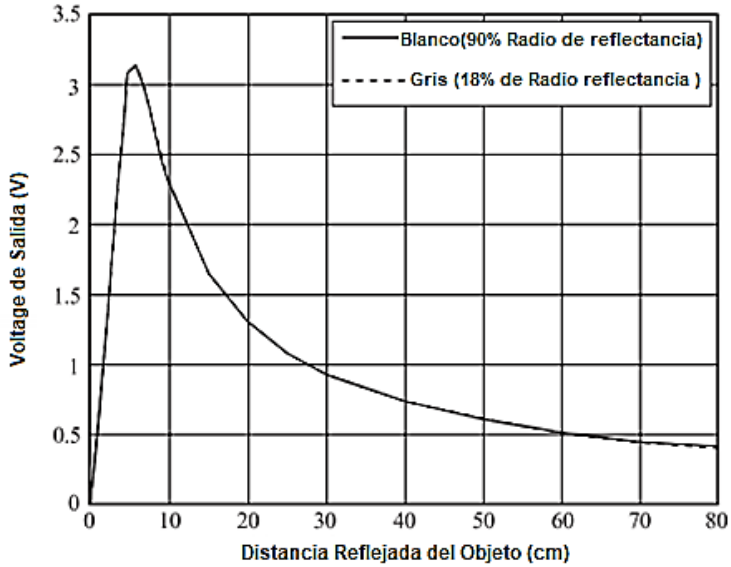


Figura15. voltaje en función de la distancia *Sharp 2y0A21*.

Tomado de (Sharp Corporation, 2011).

Las características físicas que los sensores infrarrojos poseen son (Mateu, 2014):

- El tipo de detección que usan es bidireccional lo cual quiere decir que tienen la capacidad de detectar objetos únicamente que se encuentran frente al sensor.
- Son más sensibles a la luz solar por la cual se pueden obtener datos erróneos de medición dependiendo del ambiente en donde se trabaje.
- Existen distintas configuraciones que permiten distintas funcionalidades:
    - Sensores reflexivos: Esta configuración permite medir la radiación procedente del reflejo de la luz emitida por el led emisor.
    - Sensor de modulado: Usa el principio de reflexión para la captación de datos usando una emisión de señal modulada.

- Sensor de ranura: Usa el principio de reflexión con varios elementos que comparten información en tiempo real
- Sensor de barrido: El sensor funciona realizando un barrido horizontal de la superficie que refleja usando señales moduladas.
- Configuración Óptica: Se basa en un sensor enfrentado a un cristal que genera una imagen de una sección en específica a medir.
- Configuración en array de sensores: Es un array de sensores que no usan el sistema de cristal de imagen.

Estos dispositivos son usados en el campo de movimiento para dispositivos autónomos como dispositivos de detección de objetos acoplados con distintos circuitos transformando su señal en digital se puede obtener la lectura de un objeto a cierta distancia cuando la medición supera el umbral límite definido.

Figura16. Sensor infrarrojo

Tomado de (naylampmechatronics, 2018)

2.2.2.4 Sensores de visión

Los sensores de visión nos ofrecen gran información del entorno donde se realizará el trabajo, los sensores de visión usan imágenes capturadas por una cámara para que los datos sean procesados y nos den como resultados datos de orientación y precisión de imágenes (IFM, 2018).

Para su funcionamiento existen dos modelos con los cuales el sensor capta imágenes:

- Modelo monocromático: La imagen captada por la cámara pasa a través del lente y se transforma en una señal eléctrica dada por el receptor de luz , el brillo y la forma del objeto están determinados por la intensidad de luz que se encuentra en cada píxel del receptor de luz.
- Modelo a color: El receptor es de color, para poder obtener los datos de la imagen la luz se separa en tres colores (RGB) con esta información la intensidad de cada color permite distinguir objetos (Keyenge, 2018)

Las características de los sensores de visión son :

- Los sensores de visión proporcionan millones de datos por lo cual es necesario dispositivos de procesamiento robustos.
- Los sensores de visión permiten emular la vista humana por lo que se puede recopilar gran cantidad de información del entorno en el que se está trabajando.
- Los sensores son sensibles a la luz por lo cual la extracción de datos de un entorno a otro varia, incluso en el mismo ambiente los datos recibidos variaran si existe un cambio de luz por lo que se necesitan métodos de extracción de datos robustos.

Estos sensores se usan principalmente para detección y evaluación de objetos por lo cual son usados en dispositivos móviles que requieran extraer grandes cantidades de información del entorno para ser evaluados dependiendo de la funcionalidad que se requiera (COGNEX, 2018).

Figura17. Sensor de visión

Tomado de (somosindustria, 2018)

Estos sensores de visión operan con diferentes plataformas y librerías como *OpenCV* que es un software de librerías de visión artificial y aprendizaje para

maquinas, es multiplataforma, proporcionando librerías para versiones *GNU/Linux MacOs* y *Windows* (Gobierno de España, 2018).

Las librerías que incorpora permiten identificar objetos, realizar un seguimiento de movimientos de objetos, para acciones como seguir el movimiento de ojos reconocer escenarios, *OpenCV* tiene una estructura modular (Bazaga, 2015):

Los grupos de librerías que incorpora *OpenCV* son:

1. *CXCore*: donde se encuentran las estructuras y algoritmos básicos que usan las demás funciones.
2. *CV*: donde están implementadas las funciones principales de procesamiento de imágenes
3. *HighGUI*: todo lo relacionado a la interfaz gráfica de *OpenCV* y las funciones que permiten importar imágenes y video.
4. *ML(Machine Learning)*: que cuenta con algoritmos de aprendizaje como *SVM* y *Adaboost*.
5. *CvAux*: con funciones experimentales como *BG/FG*, estéreo.

Estas librerías usan *CXCore* como base (figura.18), proporciona funciones básicas para la implementación de los módulos para finalmente ser compilados y ejecutados en una aplicación (López C. C., 2018).

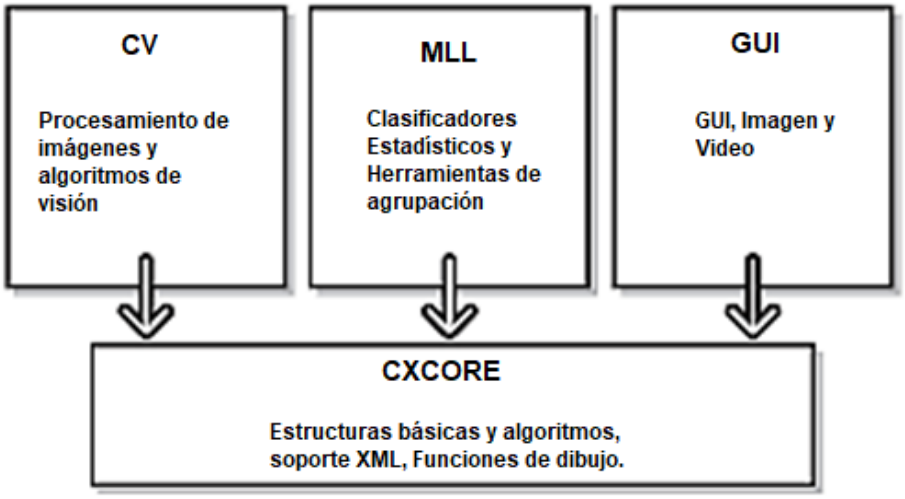


Figura18. Estructura modular *OpenCV*

Tomado de (López C. C., 2018)

2.2.2.5 Control de movimiento a cuatro ruedas

El análisis cinemático de un robot móvil se da en referencia a un punto P que se encuentra en el centro del prototipo, donde se coloca un sistema de referencia (xm− ym).

De esta manera se puede conocer en todo momento la posición y velocidad del robot móvil, al asociar, a su vez, la posición del punto P a un sistema de referencia global (xb − yb).

La posición del robot puede ser descrita en términos de las coordenadas; x e y que representan la posición del punto P respecto al plano global (x, y).

El punto P se ubica en el origen del marco de referencia (xm − ym), y a su vez tanto el punto P como el origen de dicho sistema de referencia se colocan en el centro de la plataforma.

La orientación de este punto está descrita por ϑ. Se utiliza el siguiente vector para describir la posición del punto P (figura.19) (J. A. Vázquez, 2007):

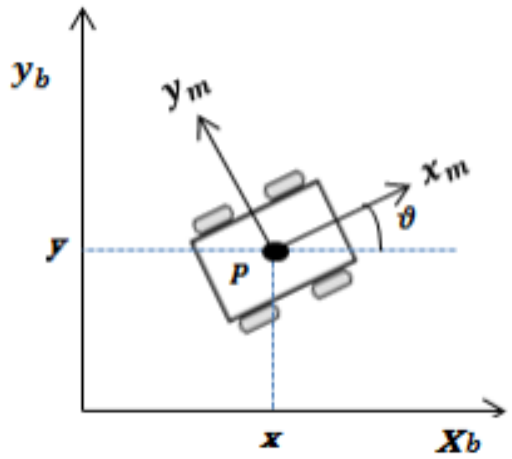


Figura19. Posicionamiento de un prototipo movil.

Tomado de (Caladin, 2008)

La matriz de rotación que permite conocer la orientación del sistema de referencia móvil respecto al sistema de referencia fijo. (Adán Esteban Suárez Arriaga, 2015)

$$R(\theta) = \begin{pmatrix} Cos\theta & Sin\theta & 0 \\ -Sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{(Ecuación 4)}$$

Para que el prototipo gire alrededor de su centro de masa, la velocidad de cada rueda de este debe tiene la misma velocidad angular.

La velocidad tangencial de cualquier punto es proporcional a su distancia del eje de rotación. Las unidades de la velocidad angular están dadas por radianes/segundo. (Caladin, 2008)

$$v = wr \ \text{ó} \ w = \frac{v}{r} \quad \text{(Ecuación 5)}$$

- v= velocidad
- w=velocidad angular
- r= radio giro

La velocidad angular es la tasa de variación del desplazamiento angular y puede ser descrito por la relación:

$$Wmedia = \frac{\Delta\theta}{\Delta t} \quad \text{(Ecuación 6)}$$

- Wmedia: desplazamiento angular medio.
- $\Delta\theta$ variación ángulo
- Δt variación del tiempo

Cuando la velocidad es constante se puede calcular el ángulo de rotación con la fórmula:

$$\theta = \theta_0 + wt \quad \text{(Ecuación 7)}$$

- θ = ángulo
- θ_0= ángulo inicial
- w=velocidad angular
- t= tiempo

2.2.3 Interfaz de Visualización de objetos

2.2.3.1 Processing

Processing es una herramienta del Grupo de Estética y Computación del *Media Lab* del *MIT*, usa lenguaje de código abierto basado en *Java 1.4.2*, el cual se traduce en puro *Java* antes de ser compilado, y permite la comunicación en tiempo real mediante una interfaz serial a 300-115200 baudios siendo 9600 baudios la transmisión por defecto (Ignacio Buioli, 2018).

Processing permite controlar una placa mediante un firmware estándar llamado *Firmata*, este contiene librerías que permite la conexión simplificada a la placa con protocolos de comunicación *UDP*.

Está orientado para crear imágenes, animaciones e interacciones, para hacer la adquisición de datos trabajando en conjunto con *OpenGL*, *X Windows, Microsoft Windows, Mac OS, VRML, X3D, POV-Ray, Renderman , Java2D, Java3D , DirectX (Direct-3D).*

Processing empieza por definir un gráfico a partir de pixeles de coordenadas (x,y) el origen se sitúa en la parte superior izquierda del plano (0,0) donde por medio de suma crea puntos dentro del plano (5,5). (figura.20) (Pellicer, 2018)

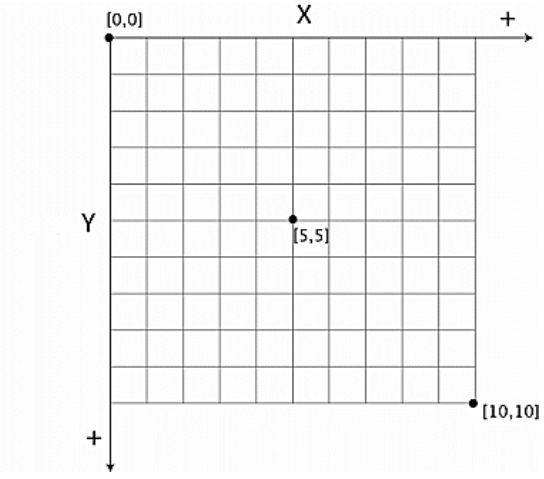


Figura20. Plano de Iniciación *Processing*.

Tomado de (Pellicer, 2018)

Permite usar *Processing Writing* que es una sub-extensión de *Processing* que usa el *IDE* de *Processing* con una versión simplificada del lenguaje *C++* para programar microcontroladores (Rodríguez, 2015).

El código pasa por un preprocesador para transformarlo a código *java* para después realizar la compilación hacia *bytecode,* lenguaje de máquina y ejecutarlo mediante la máquina virtual de java (figura.21) (Pellicer, 2018).

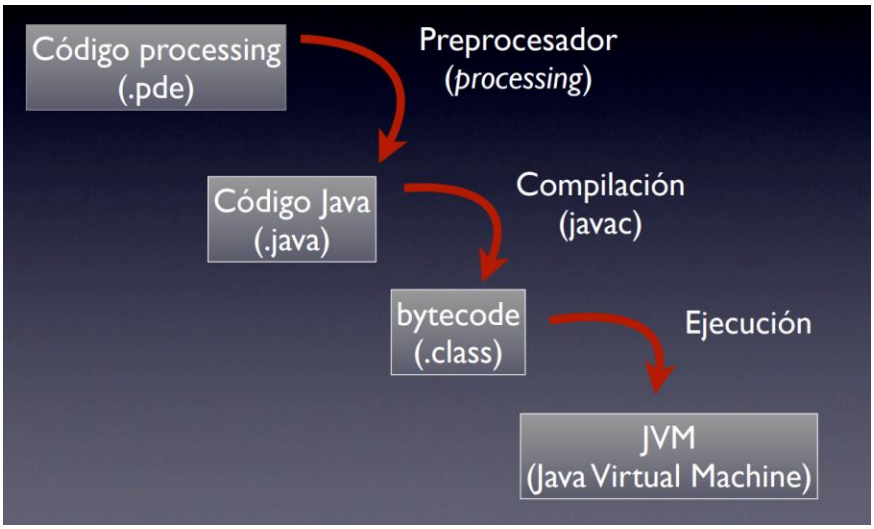


Figura21. Ejecución de codigo Processing

Tomado de (Processing, 2018)

2.2.3.2 Plataforma Arduino

La plataforma Arduino es una multiplataforma de código abierto basada en software y hardware , implementa lenguaje de programación Arduino que incorpora un editor de código, un compilador, un depurador y una interfaz grafico *GUI* basado en librerías *C++* (Herrador, 2009).

El hardware Arduino se basa en una placa con un microcontrolador incorporado, el cual el programable y es capaz de guardar sentencias programadas en el software Arduino para que funcione con sus principales unidades (Nicolas GOILAV, 2016):

- CPU: unidad central de procesamiento
- Memoria: *Flash, SRAM, EEPROM, ROM*

- Periféricos de entrada y salida: entradas analógicas, salidas analógicas *PWM, DAC, ADC , I2C, SPI* en la placa y ethernet, wifi.

Arduino da soporte a las placas Arduino *AVR*, también permite instalar soporte para los Arduino con *MCU ARM* de 32 bits como el *Arduino MKR1000* o las *Intel* como el Arduino 101 (Grupo Halley, 2018).

Las placas son programadas a partir *sketchs* o proyectos con extensión *.ino* por medio de su lenguaje *C++* proveniente de la librería *avr-lib* que es una librería de *C* para usar con *GCC* (*GNU colection compiler*) que es un compilador de C y *C++*.

A través del puerto de comunicación serial, se compila para transformar el código a binario y pasarlo a la memoria del microcontrolador (figura.22) (Johnny Novillo-Vicuña, 2018).

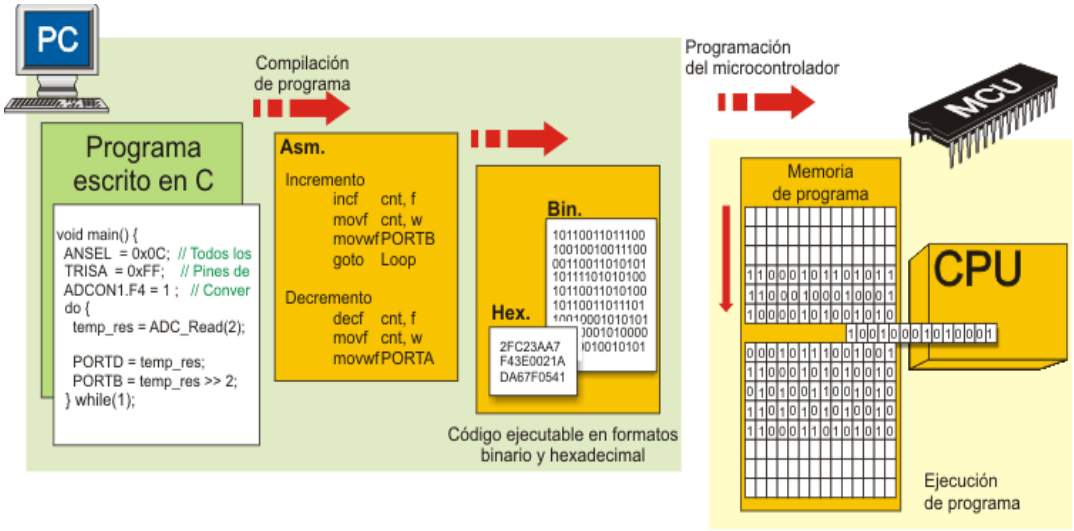


Figura22. Compilación de programa en Arduino Software

Tomado de (Arduino, 2018).

3. CAPITULO III. DISEÑO FUNCIONAL Y ESTRUCTURAL DEL PROTOTIPO

3.1 Requerimientos del sistema

- El prototipo implementa un algoritmo de escaneo de obstáculos mediante un sensor ultrasónico *HCSR-04* y un servomotor (*Futaba-*

S3003) frontal por lo que el prototipo escaneara su zona frontal para evitar obstáculos conforme avance.

- El prototipo antes de avanzar escanea la zona y verifica dentro de un rango 1 - 20 cm en una zona de 30 – 150 grados objetos que se encuentren dentro.
- Al identificar un objeto que se encuentra dentro de la zona de escaneo determina mediante el algoritmo de modelado un camino libre para poder avanzar, o tomar una decisión de cambio de posición sobre su eje para buscar un espacio disponible.
- Al avanzar accionara la función de limpieza que trabajara con un servomotor acoplado a un paño de limpieza y una bomba de agua que trabaja conjuntamente mientras avanza el prototipo.
- La autonomía de funcionamiento del prototipo deberá ser mayor a 1 hora.
- Velocidad de movimiento controlada mediante motores de reducción.

Los requerimientos técnicos se ven reflejados en la (Tabla.2).

Tabla2.
Requerimientos del Prototipo

Tipo requerimiento	**Componente**	**Descripción**
Control:	Microprocesador Microcontrolador	Gestión del prototipo, disponibilidad de pines analógicos, digitales, PWM, e interrupciones para sensores mediante placa Arduino mega2560 con conexión serial a 9600 baudios para transmisión de datos en tiempo real a la plataforma Processing.
Movimiento	Algoritmo de evasión	Algoritmo que permita buscar espacios libres en el entorno.

		Algoritmo que calcule la bisectriz del Ángulo formado entre objetos o espacio de escaneo que comprende de entre 30 a 150 grados
	Motores	Motores para el movimiento, motores con reducción 1:48 para el control de la velocidad de movimiento motores de entrega mínima a 100 mA
	Controladores motores	Controlador para motores corriente mínima de entrega de 400 mA
Detección	Sensor ultrasónico	Sensores detección y escaneo de objetos mediante sensor HC-SR04
	Servomotor	Servo para movimiento de escaneo 30-150 grados
Limpieza	Servomotor	Servo para acción de limpieza del suelo, 3kg de torque
Alimentación	Batería	Batería recargable de medio nivel de descarga externa.

3.2 Diagrama de interacción con el entorno

El proceso de operación del prototipo consiste en la limpieza que es activada por parte del motor bomba de agua durante 500 ms y el servomotor posterior que activa el movimiento del paño limpiador que funciona de 30 a 150 grados mientras el prototipo se mueve en línea recta dentro del entorno de trabajo.

Cuando existen objetos dentro de la zona de escaneo frontal del prototipo, se activa el algoritmo evasor donde se escanea objetos dentro de 20 cm de distancia para la toma de decisiones (giro a n grados sobre su eje) por medio del sensor ultrasónico *HC-SR04*.

Cuando no existen objetos en el espacio frontal después del escaneo del sensor, el prototipo se mueve frontalmente activando el sistema de limpieza nuevamente (figura.23).

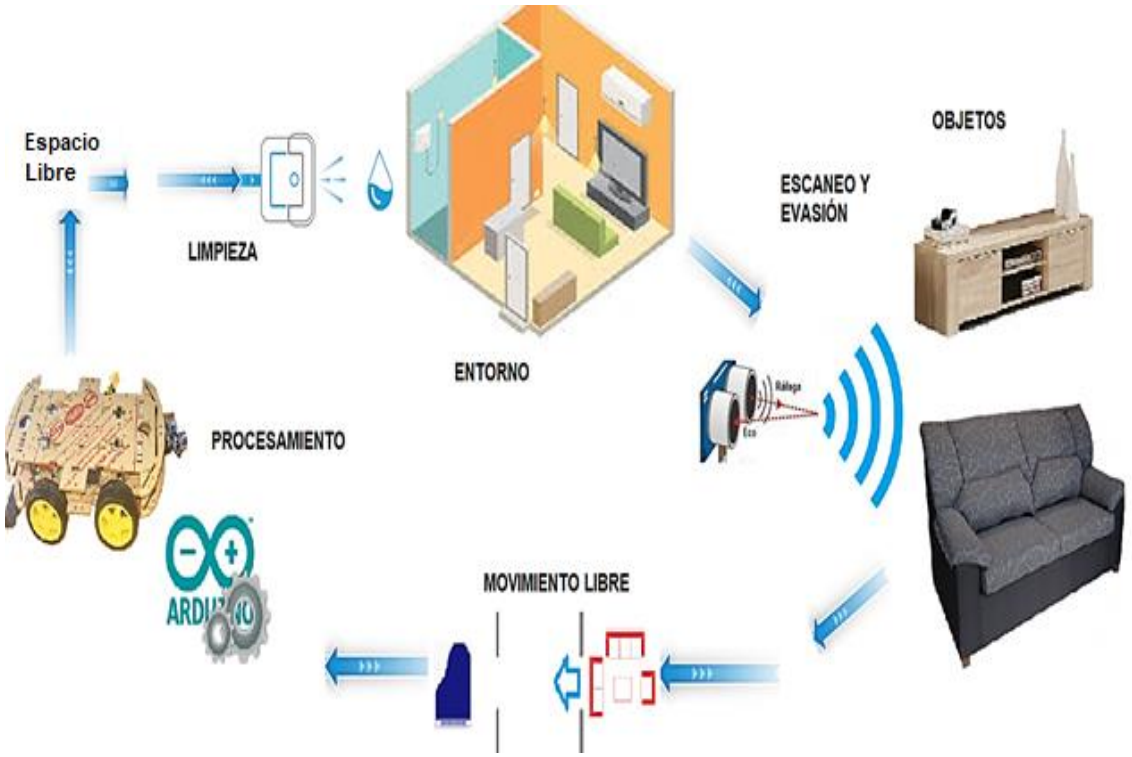


Figura23. Diagrama de interacción con el entorno.

Descripción: en la figura actual se muestra la interacción que existen entre los elementos, limpieza del prototipo en un entorno doméstico el cual evita obstáculos mediante el escaneo del sensor *HC-SR04*.

3.3 Análisis de algoritmos

De los diferentes modelos de planeación de rutas podemos determinar en la (Tabla.3) donde se compara las diferentes ventajas y desventajas de cada modelo.

Tabla3.

Comparación de modelos para evasión de obstáculos.

	Espacio libre y restricción de distancias	**Descomposición en Celdas**	**Grafos de visibilidad**
Principio	Planificación del movimiento a través de CRG(cilindros) en espacio libre	Conjunto de celdas en el Espacio	Uso de Polígonos para detección de objetos
Ventajas	Simple implementación del algoritmo a través del cálculo de la bisectriz de los vértices de objetos dentro del área de escaneo. Este modelo permite tener un constante límite para evitar la colisión de objetos. Modificado no requiere de las dimensiones del entorno en que se trabajara puede acoplarse a un movimiento autónomo. Procesamiento medio al analizar bordes de	Define una ruta a partir de celdas que se encuentren vacías a partir de una forma geométrica La cantidad de celdas dadas es menor en comparación a la cantidad de los grafos de visibilidad.	Permite definir varios puntos de un objeto mediante la generación de vértices entre sus polígonos. Calcula el camino más optimo entre nodos para llegar a un destino con mayor velocidad.

	objetos y calcular algoritmo para evasión buscando la bisectriz del área escaneada.		
Desventajas	El grado de precisión puede variar dependiendo de la precisión del sensor usado. El tipo de escaneo en espacio libre debe realizarse cada cierta trama de distancia para evitar errores al localizar y calcular objetos El paso de un CRG a otro se produce siempre y cuando sus ejes intercepten y la intersección del rango de orientaciones admisibles en el punto de corte de ambos ejes no sea nulo.	Busca únicamente puntos medios entre objetos así se tenga una distancia mayor a la aceptable para avanzar Requiere de las dimensiones del entorno La descomposición en celdas involucra grafos de adyacencia lo cual vuelven más complejo al algoritmo Las celdas expuestas pueden solaparse dando paso a errores funcionales. El grado de precisión puede variar dependiendo de la precisión del sensor	Los grafos trazados requieren de un almacenamiento entre matrices. Solo después de examinar la matriz es posible calcular el recorrido posible Alto nivel de procesamiento al procesar n vértices de nodos extraídos. Requiere de las dimensiones del entorno previamente ya que necesita un punto de

	usado.	inicio y fin .
	La conexión entre dos celdas existe si y solo si son adyacentes entre ellas.	
	El tipo de lectura requiere de sensores más potentes para cubrir área en donde trabaja.	

El algoritmo de espacio libre que usa el prototipo es una modificación al tradicional y se basa en calcular la bisectriz del ángulo formado entre dos objetos dentro de la zona de escaneo y permitir mediante un giro sobre el eje de prototipo determinar un camino libre de obstáculos.

El mejor algoritmo para el modelo de prototipo es el de espacio libre y restricción de movimiento, dado que nuestro prototipo se mueve autónomamente no necesita de una planeación previa de ruta ya que los movimientos se dan conforme se escanee la ruta.

Si bien es cierto la precisión para evitar colisiones puede variar, se puede establecer un límite que nos permita tener un margen de error en cuanto a la distancia máxima de detección de obstáculos para evitar colisiones.

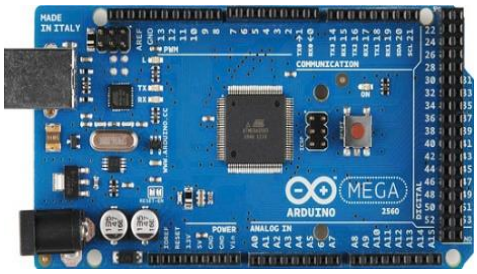
3.4 Análisis elementos

Los componentes para el armado del prototipo se detallan en la tabla.4, los materiales empleados cuentan con distintas características técnicas las cuales

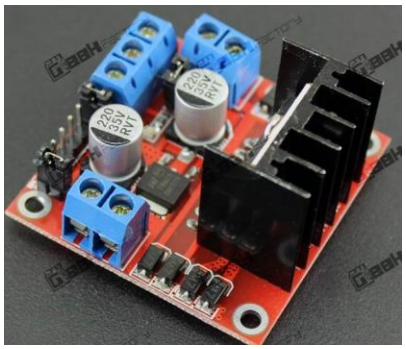
serán detalladas y analizadas para comprobar su factibilidad en la implementación del prototipo (Tabla.4).

Tabla4.

Características de Elementos físicos.

Elemento	Características	Grafico:
Placa Arduino Mega2560	• Microcontrolador AVR Atmega2560 8 bits • 16 MHz frecuencia • 8 Kib memoria RAM • Memoria flash: 128-256Kib • Memoria EEPROM: 4Kib • Pines digitales: 54 • Pines analógicos: 16 • Corriente por pin: 5 - 40mA • Tensión pin: 5v • Resolución pines analógicos: 1024 valores • Pines de interrupción: 6 • Pines PWD: 15	
Motor reductor:	• Rango voltaje de operación: 3V-6V • Rpm: 125 • Reducción: 1:48 • Corriente: 80 mA • Dimensiones: 7 cm x3.7 cm x2.2 cm • Peso c/u: 70g	

Motor Bomba de agua	• Rango voltaje de operación:3-12V • Rpm:125 • Reducción: 1:48 • Corriente: 300mA. • Dimensiones: 8 cm x2.7 cm x3.5 cm • Peso: 70 g	
Batería Lipo	• Capacidad: 2200 mA • Configuración: 2S1P/7.4v/ 2Celdas • Reducción: 1:48 • Capacidad de descarga: 65-130 • Dimensiones: 105 x33 x17mm • Peso: 134g	
Sensor de ultrasonidos HC-SR04	• Rango voltaje de operación: 5v • Rango medida: 2cm - 400cm • Señal disparo: 10us • Apertura pulso: 15 grados • Corriente: 15 mA • Dimensiones: 4.5cm x2 cm x1.5 cm.	

Driver L298N	• Rango voltaje de operación: 5-46v • Potencia: 25Watts • Corriente máxima 36mA - 2000 mA • Dimensiones: 4.3 cm x4.3 cm x2.7 cm • Peso: 70 g	
Servomotor	• Rango voltaje de operación: 4.8-6v • torque: 3,2Kg/cm • Grados libertad: 180 grados • Corriente operación: 8mA. • Dimensiones:40.4 x 19.8 x 36 mm • Peso: 37.2 g	

La distribución de pines usados en la placa *Arduino Mega2560* se ve dado por la (Tabla.5):

Tabla5.

Distribución de pines de elementos.

Material	**Pines Arduino**	**PinesVCC**	**Pines Gnd**	**TOTAL**
Driver Ln298n	6	1	1	8
Servo1	1	1	1	3
Servo2	1	1	1	3
Sensor ultrasónico	2	1	1	4

Motor bomba agua		1	1	2
Relay 5v	-	3	1	4
Transistor 2n3904	-	1	1	2
Resistencia 1k	1	-	-	
rectificador 1N4007	-	1	1	2
Batería Lipo	1	-	1	2
Switch encendido	1	1	1	3
Total	13	11	10	34

Con las características de los elementos descritos en las anteriores tablas, podemos calcular la autonomía ideal del prototipo expuesta en la (Tabla.6) :

Tabla6.

Consumo de corrientes de elementos.

Elemento	Cantidad	Voltaje	Corriente c/u	Corriente total	Potencia
Driver Ln298n	1	5v	36mA	36mA	0.18 w
Servomotor	2	5v	8mA	16mA	0.04 w
Sensor ultrasonico	1	5v	15mA	15mA	0.075 w
Motor bomba de agua	1	5v	600mA	600mA	3 w
Motores ruedas	4	5v	100mA	400mA	2 w
Placa Arduino Mega 2560	1	5v	90 mA	90mA	0.45 w
Total				857mA	5.295 w
Batería lipo(carga)	1	7v	2200mA	2200mA	16.28 w
Autonomía ideal = (Total potencia consumida)/ (Carga batería)					3.07 Horas

La batería externa lipo de 2200 mA la cual nos provee de suficiente corriente para alimentar arduino y al controlador *L298n* el cual es el que consume mas corriente de alimentacion con sus 4 motores reductores.

La corriente máxima que el dispositivo *L298n* soporta es 2000 mA en cada salida con una tensión de alimentación de 3 – 35 voltios es factible realizar el paralelo de 4 motores obteniendo una corriente de consumo para cada motor del componente de 100 mA y un voltaje de 5 voltios configurable en el driver.

3.5 Sistemas:

El esquema mostrado en la (figura.24) detalla las conexiones realizadas.

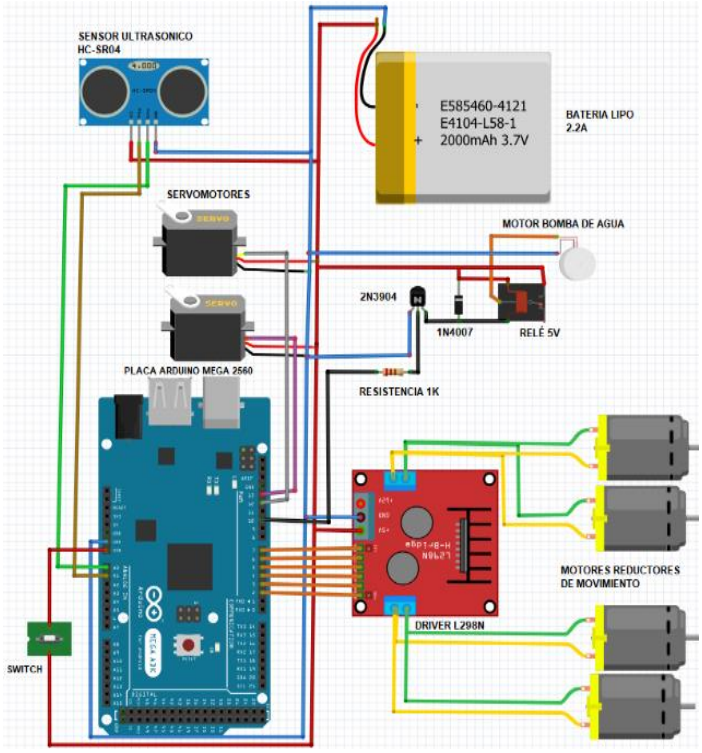


Figura24. Esquema general del prototipo.

Descripción: Esquema y conexión de pines para cada elemento (servomotores, sensor *HC-SC04*, Motores reductores, controlador *L289n*, bomba de agua, relé, transistor, batería, rectificador, resistencia) en la placa *Arduino Mega 2560.*

Siguiendo el siguiente diagrama de flujo podemos determinar los métodos y funciones necesarios para la programación detallado en la (figura.25) :

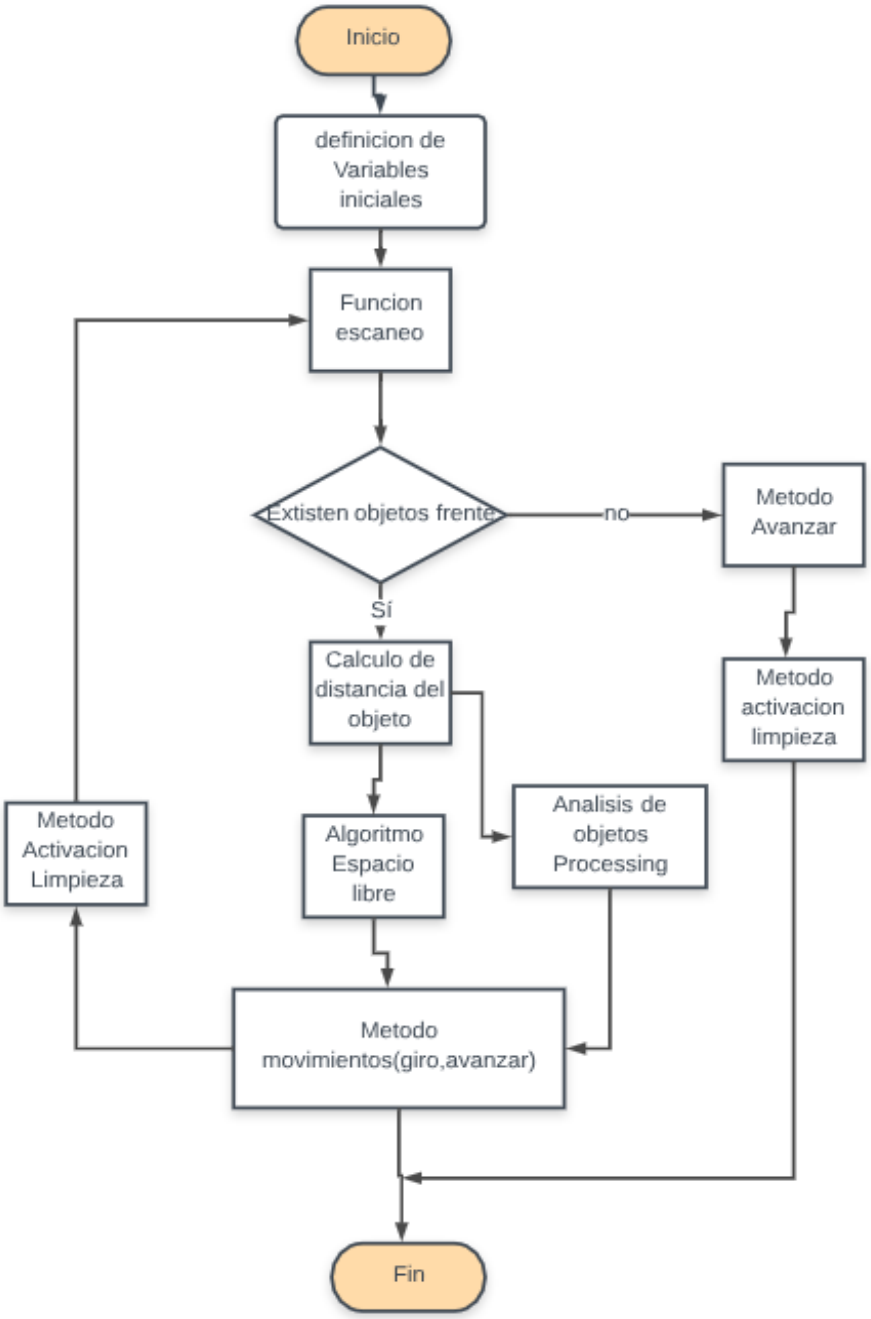


Figura25. Diagrama de flujo del prototipo.

Descripción: Diagrama de flujo donde se indica el procedimiento de funcionamiento del prototipo

El diagrama de flujo del procedimiento de reconocimiento de objetos se ve dado por la secuencia de funciones (figura.26).

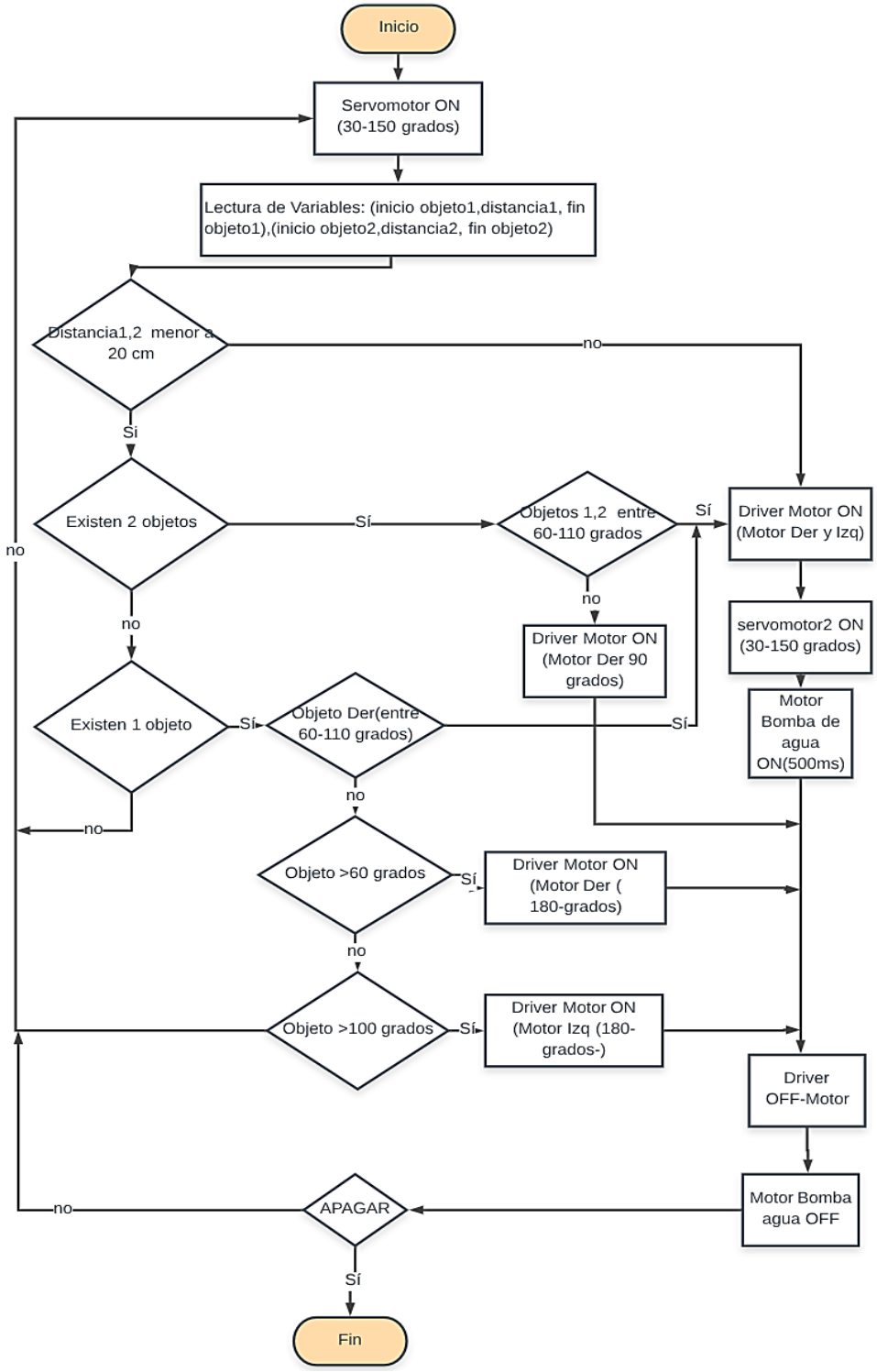


Figura26. Diagrama de flujo Reconocimiento Objetos

Descripción: Diagrama de flujo donde se indica el procedimiento de reconocimiento de objetos.

El código desarrollado se observa detallando en el (Anexo.1).

3.5.1 Módulo de navegación:

El algoritmo usado para la navegación del prototipo es una modificación del modelado en espacio libre y restricción de distancias permitiendo desplazarse de manera autónoma conforme vaya analizando el entorno en el que se encuentra gracias al sensor ultrasónico *HC-SR04*,el cual esta alimentado directamente de la placa *Arduino Mega2560.*

El movimiento es controlado por un Servomotor de 3.7 voltios que es controlado desde la placa *Arduino Mega2560* para realizar un movimiento de 120 grados para que, el sensor acoplado sobre este servomotor obtenga los datos en estos 120 grados de libertad (figura.27).

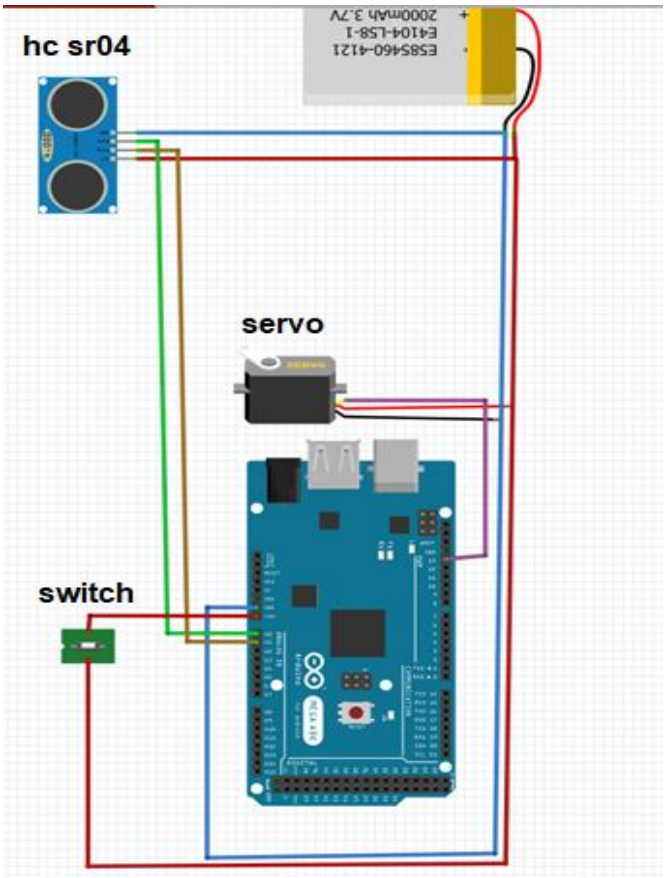


Figura27. Esquema del Módulo de Navegación

Descripción: Esquema donde se muestran los componentes necesarios para la construcción del módulo de navegación del prototipo implementando un servomotor de 3.7 voltios y un sensor ultrasónico *HC-SR04*

Para la construcción del módulo en el prototipo se incorporan los elementos físicos (figura.28).

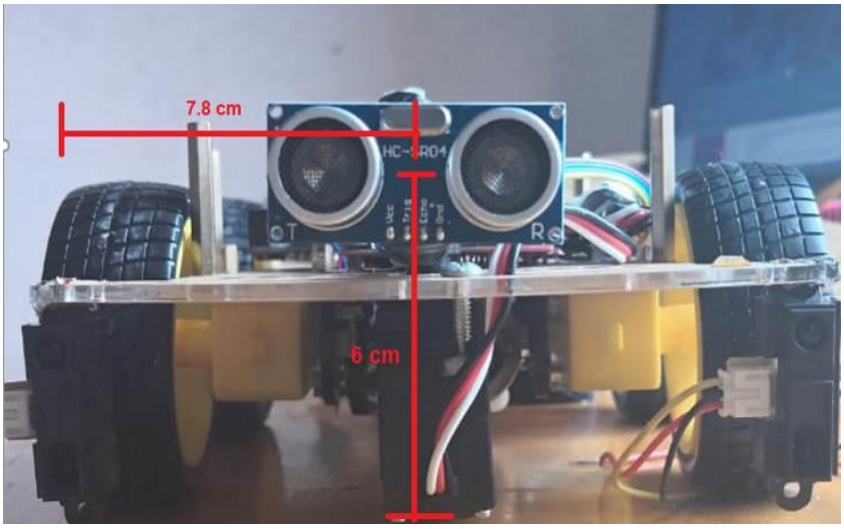


Figura28. Sensor *HC-SR04* prototipo

Descripción: Sensor *HC-SR04* a una distancia de 6cm desde el suelo y a 7.8 cm desde el borde del prototipo

Desde una vista superior se puede observar las medidas que pertenecen a la colocación del sensor en el prototipo (figura.29).

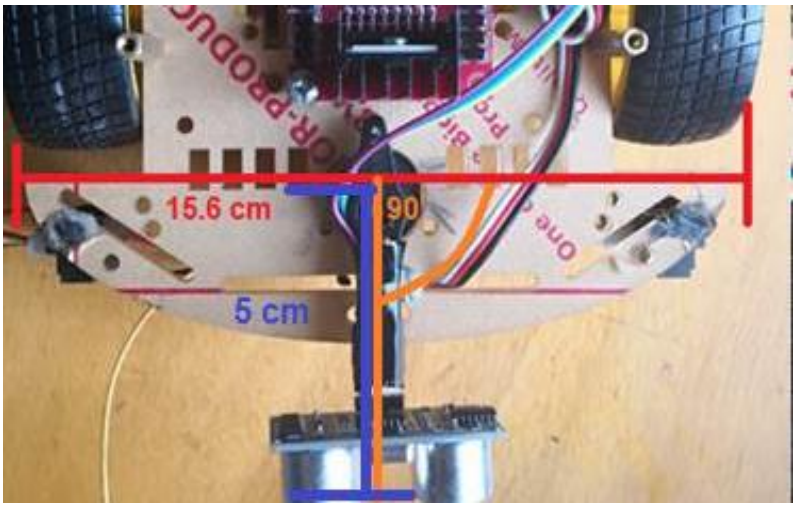


Figura29. Medidas superiores sensor *HC-SR04*

Descripción: Sensor *HC-SR04* a una distancia de 5cm y a 90 grados desde el eje del servomotor.

3.5.2 Módulo de movimiento:

El prototipo es un robot de tipo diferencial, este tipo de robots poseen tres grados de libertad se mueven en cualquier posición y orientación en el plano (X, Y, θ).

Estos utilizan una configuración de cuatro ruedas (figura.30), y la configuración utiliza un motor por cada rueda para obtener movimientos separados y girar en diversas direcciones o movimientos unidos para moverse en un mismo sentido como un solo sistema.

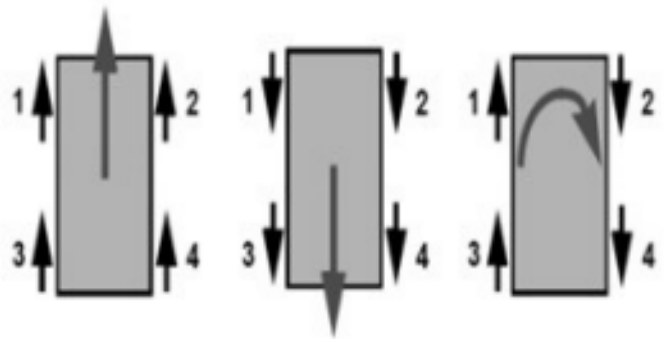


Figura30. Configuracion de movimiento de motores.

Tomado de (Caladin, 2008)

El sistema de locomoción para el vehículo expuesto consta de las siguientes características:

- El prototipo y todas sus partes forman un objeto rígido.
- Las 4 ruedas del prototipo no cuentan con un eje para rotación independiente.
- Las 4 ruedas se mantienen paralelas entre sí.
- El desplazamiento angular se ve dado por la sincronización de sus 4 ruedas funcionando coordinadamente.
- Los ejes de la dirección son perpendiculares al suelo.

El control de los cuatro motores se ve dado por el *controlador l298n*, este módulo posee dos puentes H que permiten controlar 2 motores *DC* y con una

conexión paralela entre 2 pares de motores se obtendrá un control de los 4 motores (figura.31).

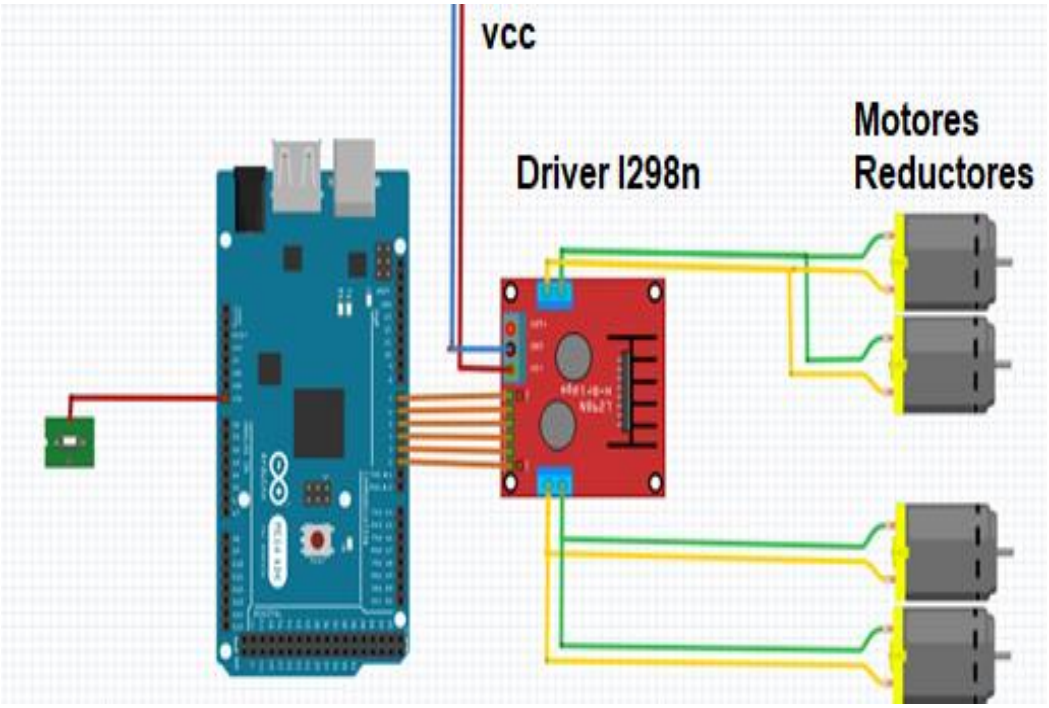


Figura31. Esquema de Control de Movimiento.

Descripción: Esquema donde se muestran los componentes necesarios para la construcción del módulo de Movimiento del prototipo implementando un controlador *l298n* y motores reductores para el movimiento.

El esquema se encuentra configurado con el jumper de selección de 5 voltios activo, así el módulo permite una alimentación de entre 6 – 12 voltios dado que nuestra fuente es una batería tipo lipo de 7.4 voltios.

El regulador se encuentra activo, el pin marcado como 5 voltios tendrá un voltaje de 5 voltios, el voltaje se puede usar para alimentar la parte de control del módulo.

Las dimensiones laterales del prototipo implementado están dadas por la (figura.32) y (figura.33) que muestran la distancia de construcción entre las ruedas en una vista lateral e inferior del prototipo.

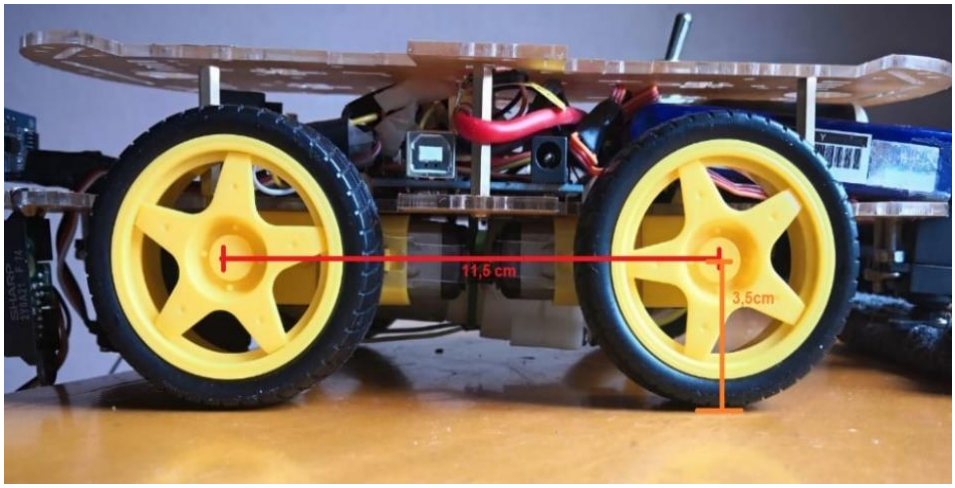


Figura32. Vista lateral Prototipo.

Descripción: se observa el alto del radio de las ruedas con respecto al suelo de 3,6cm y la distancia entre ejes centrales de 11,5 cm .

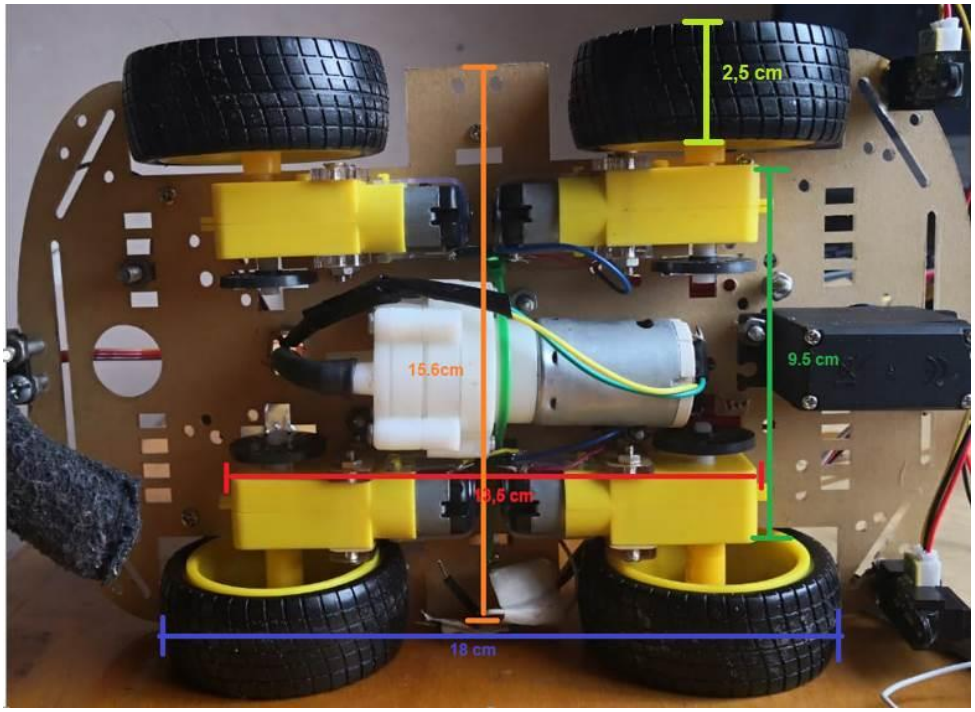


Figura33. Vista inferior prototipo

Descripción: Se muestra el área que ocupan los motores de reduccion de movimiento de 9,5 cm de ancho y 13,5 cm de largo , asimismo como el largo de las ruedas que ocupan 18 cm entre las ruedas delanteras y traseras que tienen una distancia de 2,5 cm para cada rueda.

3.5.3 Módulo de limpieza

El módulo para limpieza consta de una bomba de agua funcionando a 5 voltios directamente de la batería lipo de 2200 mA, la cual se activa al recibir la señal del pin digital de la placa de 5 voltios da un pulso de corriente al transistor *2n3094*, permitiendo el paso de 5 voltios de la batería lipo.

La utilización del transistor se realiza para reducir la corriente usada por el *Arduino Mega2560* ya que la bomba de agua ocupa 300 mA, lo que es demasiada corriente para un pin Arduino que puede ocupar alrededor de 5 - 40 mA.

Un servomotor acoplado a un paño realiza la función de limpieza, controlado por la placa Arduino la cual le da movimientos de 30 a 150 grados continuos mientras esté en funcionamiento (figura.34).

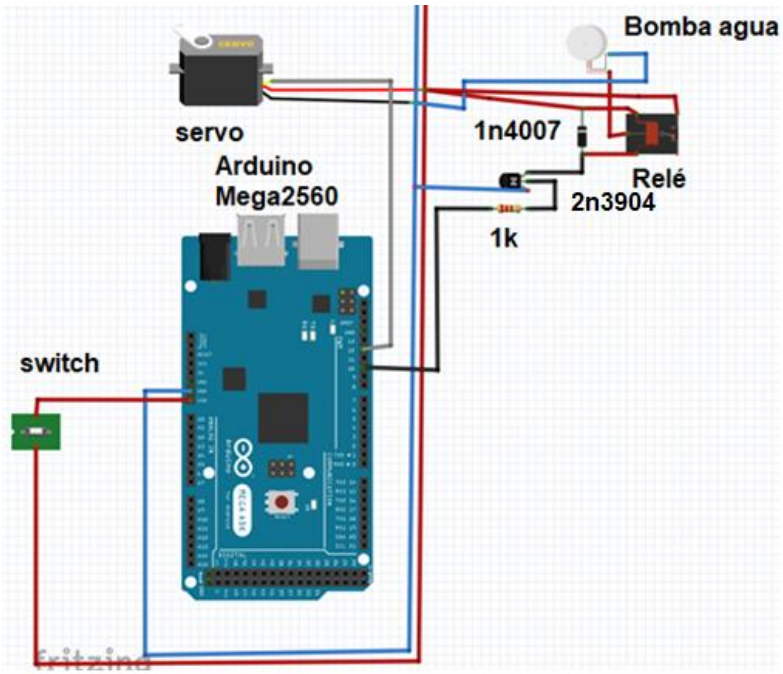


Figura34. Esquema de Modulo Limpieza.

Descripción: Esquema donde se muestran los componentes necesarios para la construcción del módulo de limpieza del prototipo implementando un transistor 2n3904 conectado a relé de 5voltios para la apertura de la bomba de agua y un servomotor para el movimiento de limpieza.

La construcción del prototipo se ve dada por las medidas reflejadas en la (figura.35) y (figura.36).

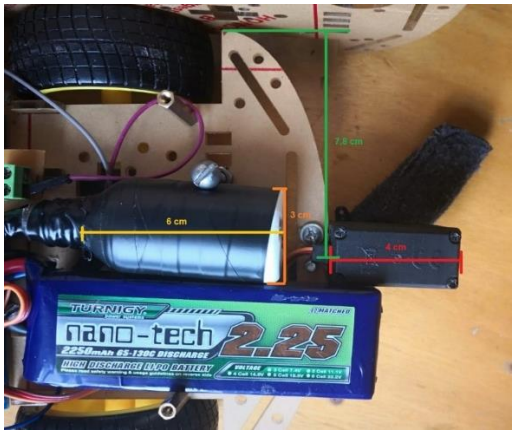


Figura35. Vista superior limpieza de prototipo

Descripción: Se muestra la adecuación de un tanque de depósito de líquido de 6cm x 3cm total de 18 ml cuadrados y un servomotor que sobresale 4 cm por la parte trasera que se encuentra a 7,8 cm desde el borde del prototipo.

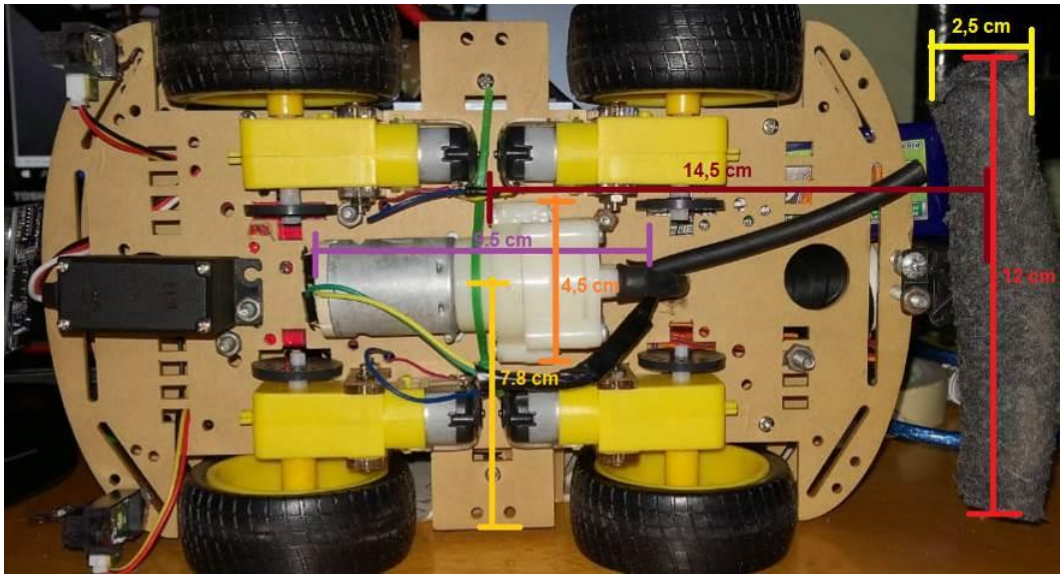


Figura36. vista inferior limpieza prototipo

Descripción: Se observa que el motor bomba de agua se encuentra a 7,8 cm desde el borde del prototipo y cuenta con 9,5 cm de largo y 4,5 cm de ancho, el paño limpiador se encuentra a 14,5 cm desde el centro del prototipo y tiene un ancho de 12 cm y 2,5 cm de largo.

4. CAPITULO IV. ANALISIS DE RESUTADOS

4.1 Detección de objetos en Processing:

Se determina mediante la combinación del sensor *HCSR-04* y *Processing* gráficamente la detección de objetos dentro de 150 grados de reconocimiento.

En color rojo para objetos sólidos que tienen más de 6 cm de altura y de color verde para los objetos que se encuentran fuera del área de reconocimiento el código detallado para captura de datos se ve reflejado en el (Anexo.2).

Para la primera prueba se reconoce un objeto que se encuentra a 10 cm con un espesor de 6,5 cm y altura de 6 cm (figura.37).

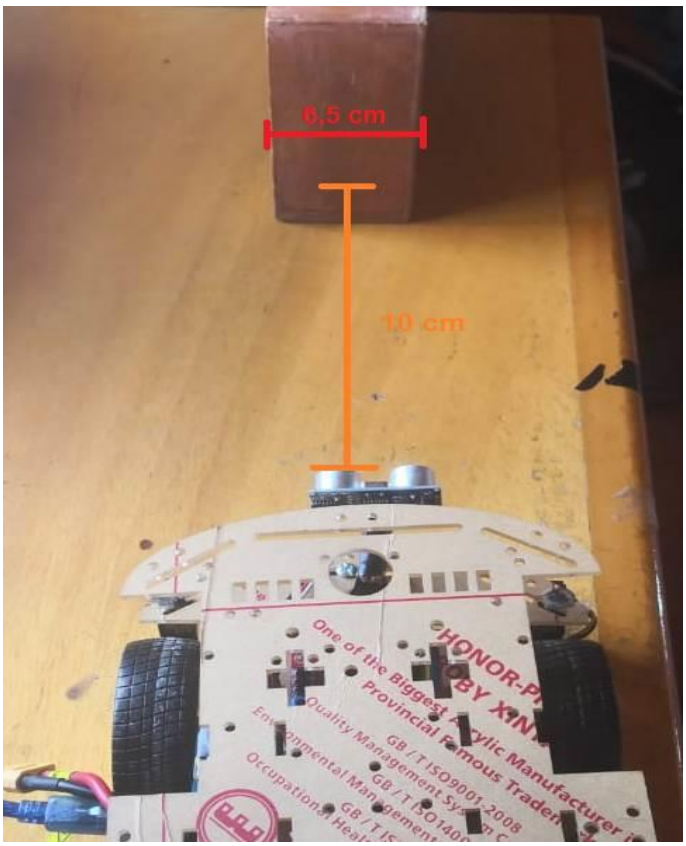


Figura37. Prueba uno para Processing.

Descripción: Se observa el objeto a 10 cm de distancia con 6,5 cm de ancho.

La primera prueba arroja una gráfica en la cual el objeto que se encuentra dentro de la zona de escaneo queda impreso de color rojo (figura.38).

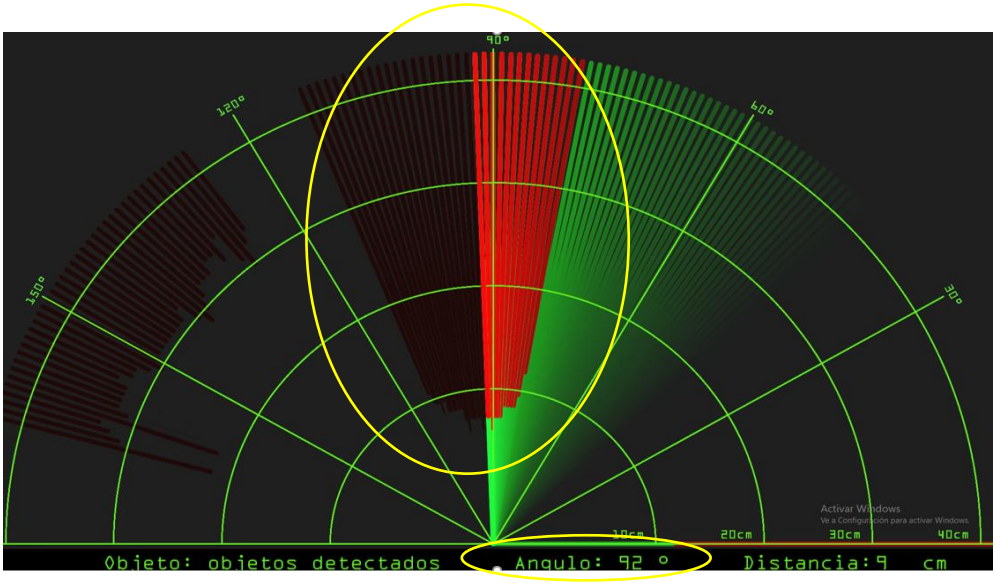


Figura38. Reconocimiento objeto uno

Descripción: Gráfica radar reconociendo el objeto a aproximadamente 9 cm de distancia en un angulo de 92 grados.

Para el objeto escaneado sólido no se observan mayores espacios de lectura verdes, que representan espacios vacios en el objeto hasta la finalización del borde. Después que termina el escaneo se observa los datos obtenidos del área libre en color verde (figura.39).

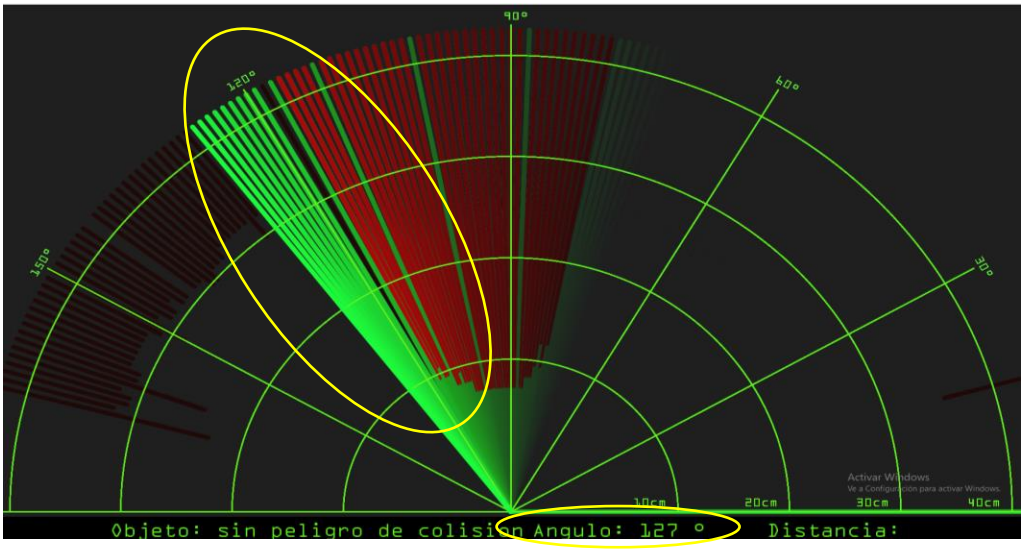


Figura39. Reconocimiento objeto uno área libre.

Descripción: Gráfica radar reconociendo el área libre que se encuentra fuera del rango donde no se aprecia la distancia en un ángulo de 127 grados .

Cuando termina el reconocimiento del objeto se puede apreciar en color verde el espacio sin objetos en el entorno y la distancia no será visible, debido a que, se encuentra fuera del rango de escaneo del sensor.

La segunda prueba se ve dada por objeto que consta de irregularidades, espacios libres y cuyo ancho es de 3 cm (figura.40).

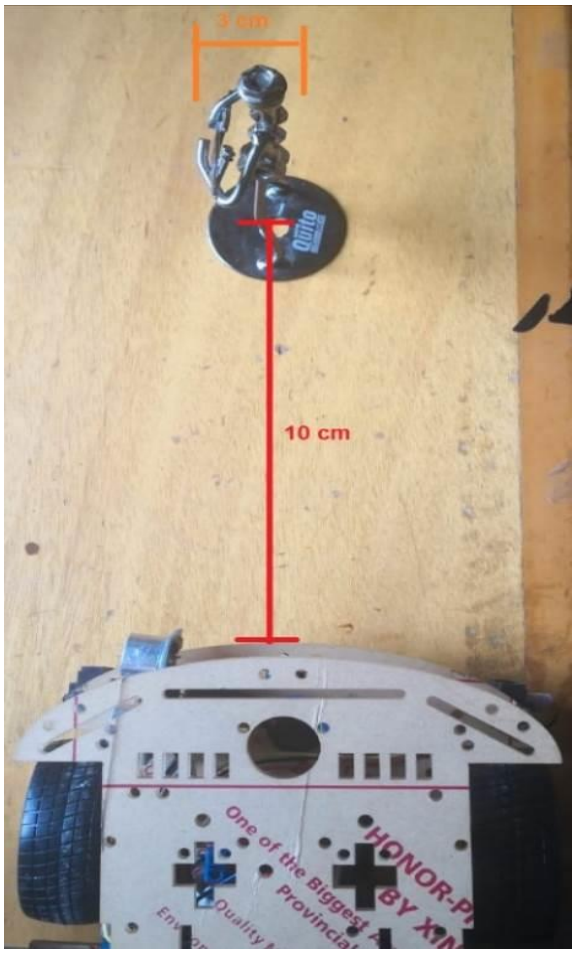


Figura40. Prueba dos Processing.

Descripción: Objeto a 10 cm del prototipo con 3 cm de ancho que consta de irregularidades.

Se obtienen los datos de la detección del objeto irregular donde se observa que el reconocimiento del objeto consta de mayor error representado por el color verde entre las líneas de color rojo que marcan el inicio y final del reconocimiento del objeto(figura.41).

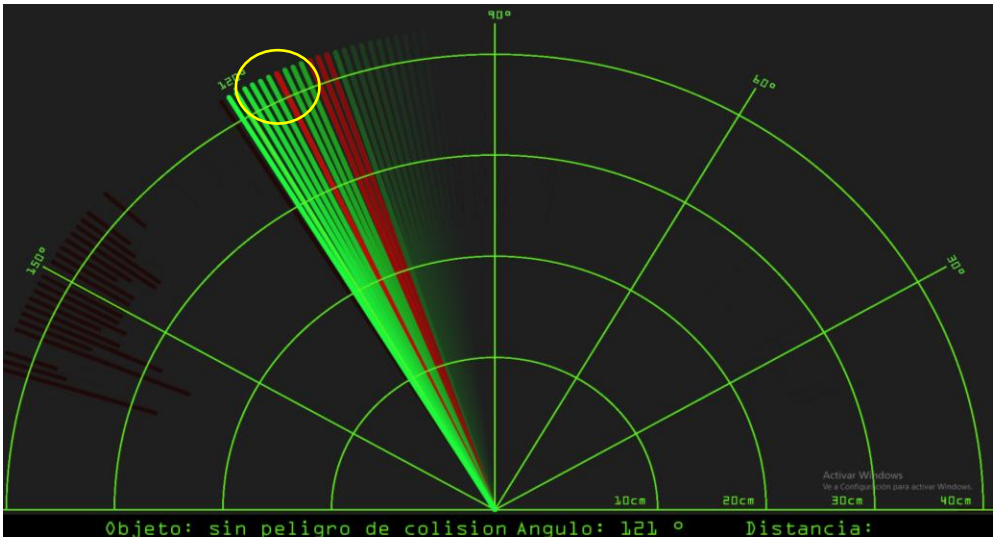


Figura41. Resultado Processing prueba dos.

Descripción: Gráfica resultante de la prueba dos dentro de 90 a 120 grados se observa una pequeña detección no clara del objeto en 3 grados de detección.

La prueba tres comprende el escaneo con objetos a 20 cm de distancia con diferente grado de espesor de 6,5 cm y 8 cm en los objetos (figura.42):

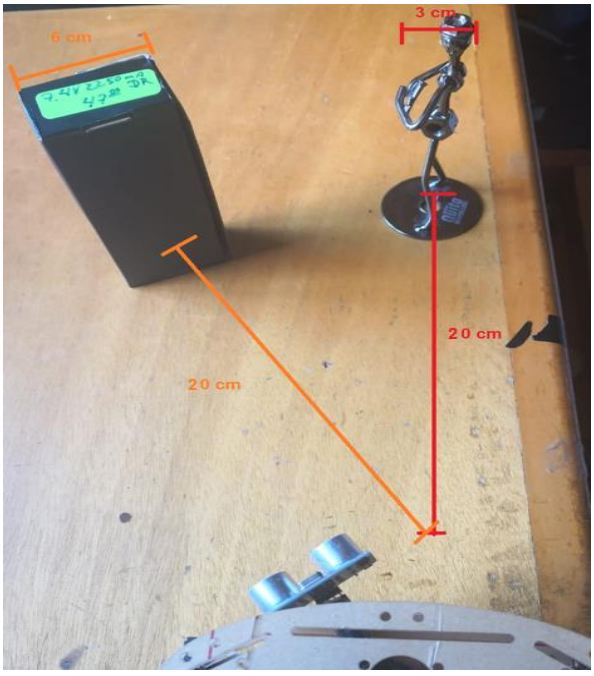


Figura42. prueba tres Processing

Descripción: Se observan dos objetos a 20 cm de distancia con 6 cm y 3 cm de ancho.

Se obtiene la gráfica de detección de los objetos donde se observa el primer y el segundo objeto resaltados de rojo (figura.43):

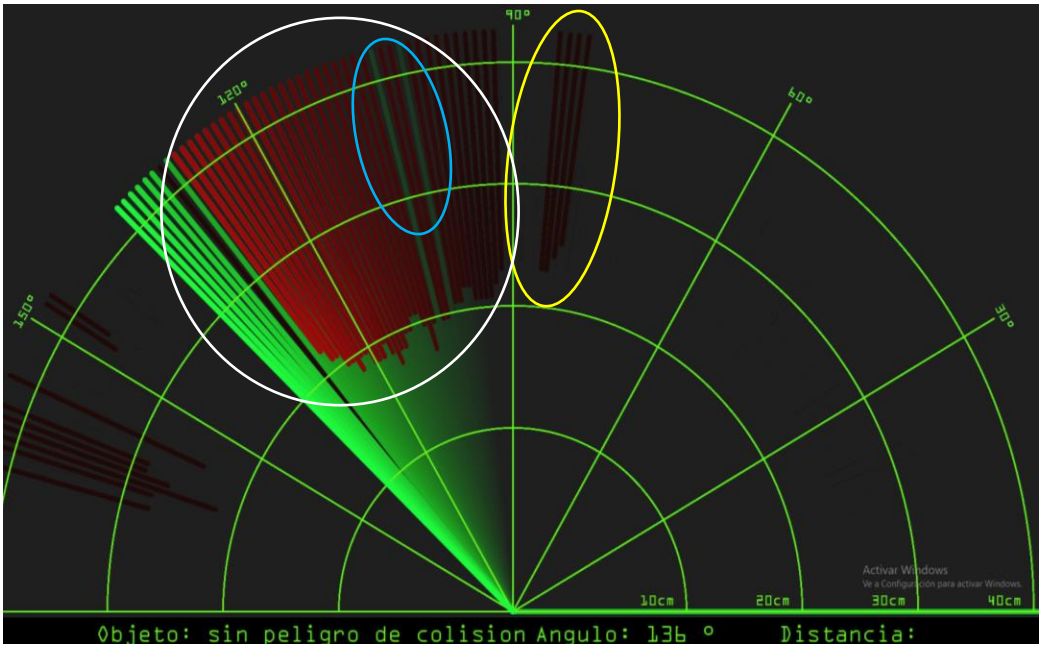


Figura43. Resultado Processing prueba tres.

Descripción: Gráfica resultante de la prueba dos dentro de 60 a 90 grados se observa una el primer objeto encerrado en un círculo amarillo y del segundo objeto encerrado en un círculo blanco de 90 a 130 grados con errores de lectura encerrados en un circulo azul.

Como resultado, podemos analizar el comportamiento del sensor con objetos que no tienen una forma definida, objetos irregulares que pueden afectar el rebote de pulso del sensor.

El primer objeto encerrado en amarillo que marca 4 grados de detección, mientras que en el segundo objeto encerrado en blanco la detección es más imprecisa observando 2 grados de error, debido a los errores de pulso del sensor.

Una detección más completa que se refleja en el área pintada en rojo que marca la detección del objeto de 92 – 130 grados.

La prueba cuatro comprende dos objetos que tienen un ancho mayor, 6 cm y 8 cm a 10 cm de distancia (figura.44):

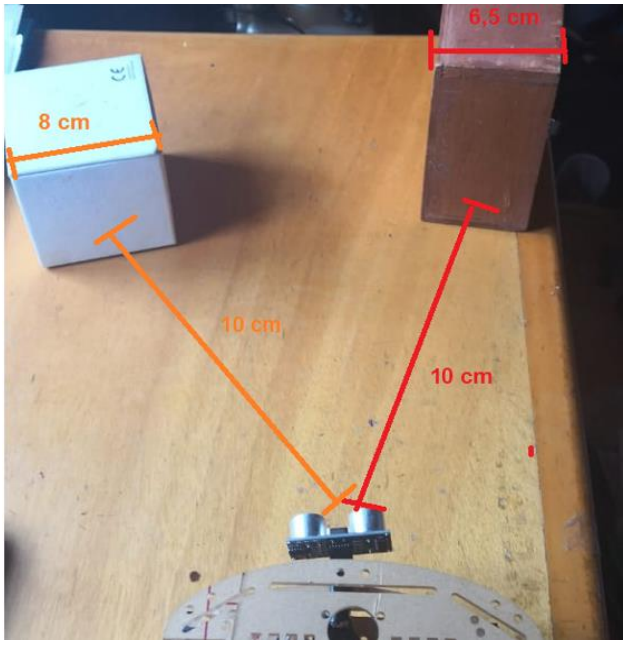


Figura44. Prueba cuatro Processing.

Descripción: Se encuentran dos objetos a 10 cm de distancia con 6,5 cm y 8 cm de ancho.

La detección del primer objeto sólido se ve dada por la (figura.45).

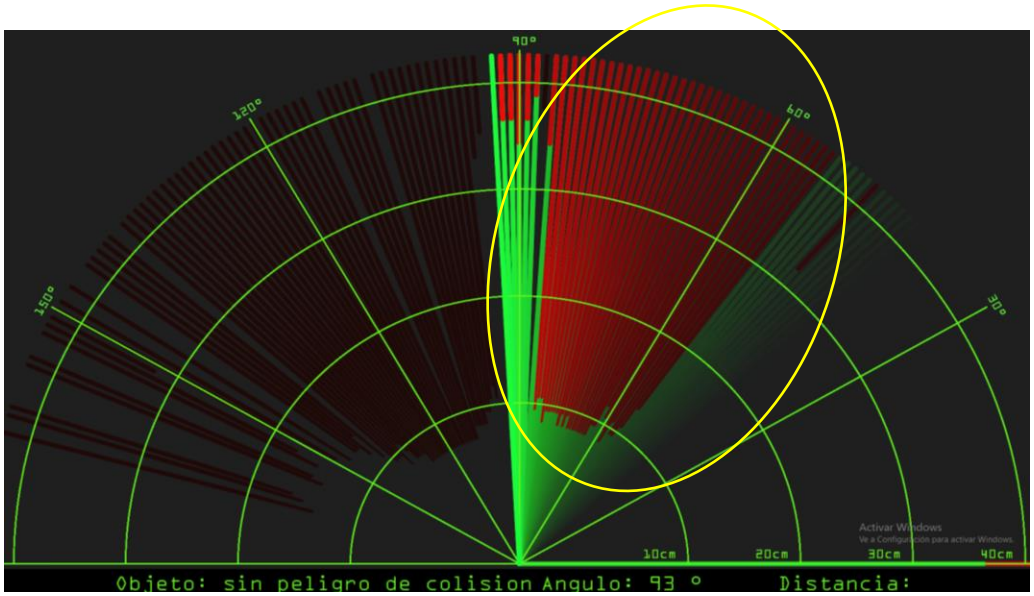


Figura45. Reconocimiento objeto uno prueba cuatro

Descripción: Gráfica resultante de la prueba cuatro dentro de 50 a 90 grados se observa detección del primer objeto de 6,5 cm de ancho encerrado en un círculo amarillo.

Se observa la detección del segundo objeto sólido (figura.46).

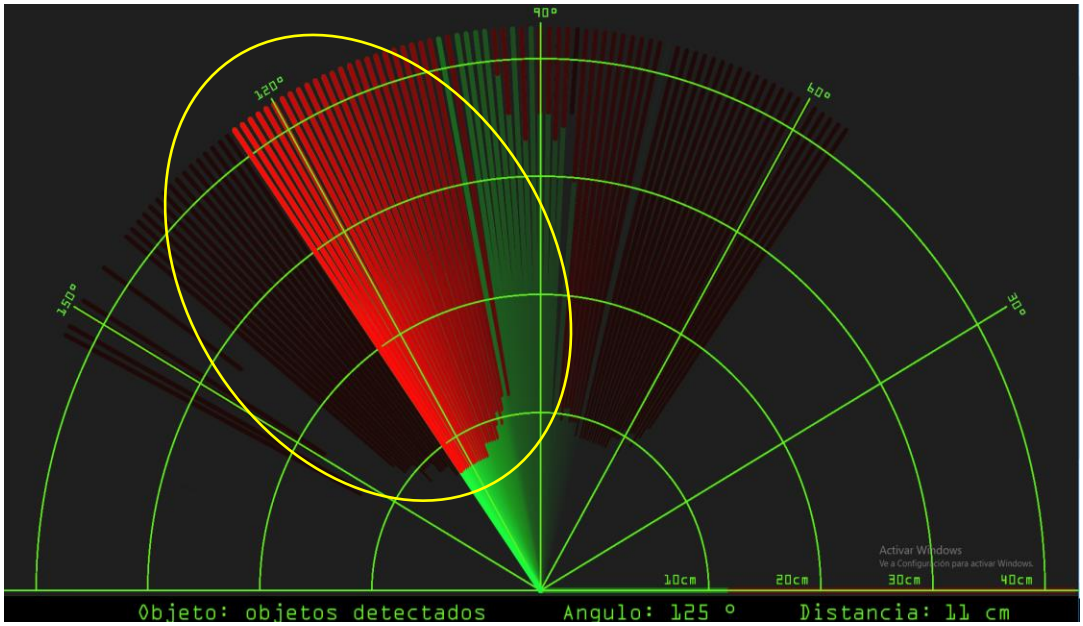


Figura46. Reconocimiento objeto dos prueba cuatro

Descripción: Gráfica resultante de la prueba cuatro donde se observa el reconocimiento del objeto de 8 cm de ancho de entre 90 a 150 grados encerrado en un circulo amarillo.

Los resultados de la deteccion (figura.46) y (figura.47) los objetos no presentan errores mayores a 1 grado en color verde entre el área roja que representa la detección del objeto

4.2.Análisis de área.

Para analizar cómo funciona el prototipo en un ambiente real es necesario definir el espacio en el cual va a trabajar, los objetos que servirán de prueba para la evasión y recrear de la mejor manera como puede ser la interacción con los mismos.

El área de trabajo donde el prototipo funciona se refleja en la (figura.47) donde consta de la estructura básica de un entorno de trabajo doméstico el cual está compuesto por una cocina, un comedor, dos dormitorios, una sala y un baño, todos son de suelo plano.

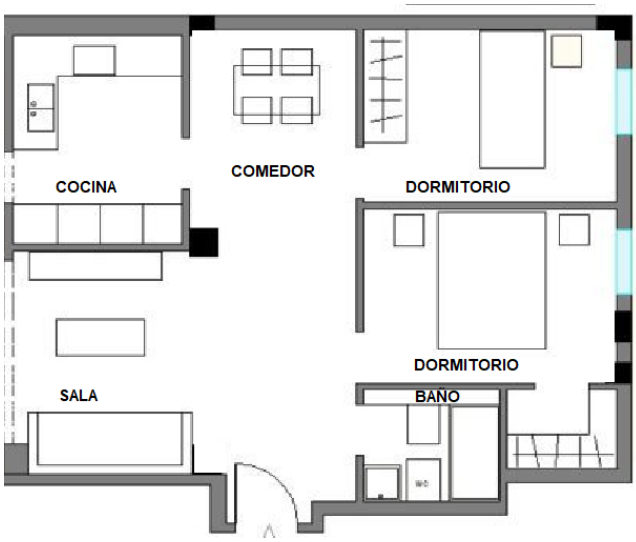


Figura47. Area de trabajo.

Adaptado de (Re-Trust, 2018). Estructura básica de una casa simple (cocina, comedor, dormitorio ,sala, baño)

Dentro de un área interior existen ciertos puntos críticos que el prototipo debe reconocer (figura.48), la descripción que corresponde a cada área(Tabla.7).

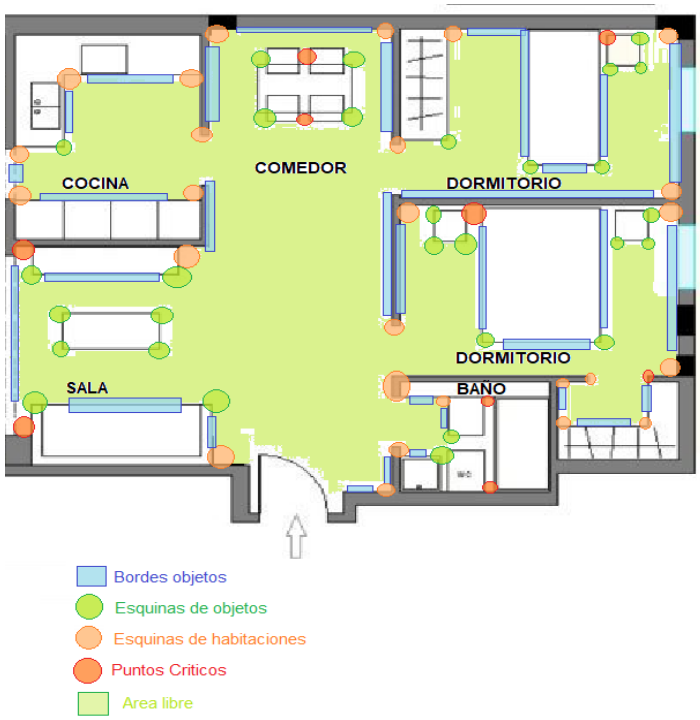


Figura48. Puntos de evasión en el Área de trabajo.

Adaptado de (Re-Trust, 2018)

Tabla7.

Descripción de Puntos de Escaneo.

Puntos de escaneo.	
Nombre	Descripción
Bordes Objetos	Referente a los objetos que no presentan mayores irregularidades en su estructura como paredes y objetos rectos.
Esquinas Objetos	Referente a los bordes que existen comúnmente en objetos domésticos, como mesas, bancos sillas , estos bordes tienen cierto grado de irregularidades.
Esquinas Habitaciones	Esquinas de las habitaciones donde el ángulo que forman entre paredes generalmente es de 90 grados.
Puntos Críticos	Todo tipo de objetos que presentan mayores irregularidades en su estructura , puntos donde no existe espacio suficiente para el movimiento del prototipo
Área Libre	Toda el área que se encuentra libre de obstáculos y que se puede trabajar sin necesidad de realizar operaciones de escaneo.

Simulamos los puntos que necesitan ser reconocidos por el prototipo para la toma de pruebas y posteriormente el mejoramiento de algoritmo.

El área comprendida para el escaneo es 150 grados(figura.49):

Figura49. Área de Escaneo prototipo.

Descripción: Área en grados de escaneo frente al prototipo (150 grados) de escaneo.

Empezamos a determinar de un total 10 intentos por cada prueba el porcentaje de fallas que tiende a presentar el prototipo.

4.3 Análisis de modelos de pruebas.

4.3.1 Modelo de pruebas 1

La prueba de bordes1 consiste en que el prototipo deba reconocer los objetos que impiden cualquier decisión de giro frente a él (figura.50) :

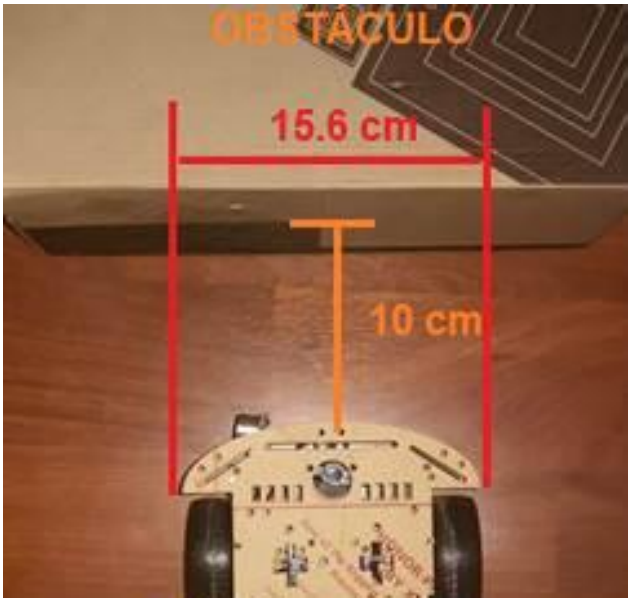


Figura50. Prueba uno prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia de un obstáculo mayor al ancho frontal del prototipo (16.5 cm).

Los datos recopilados se muestran en la (Tabla.8) con su gráfico (figura.51).

Tabla8.
Prueba uno de Evasión.

Prueba en Bordes 1				
N DE PRUEBAS	10			
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	GRADOS

1 cm	0	10	adelante	0
2 cm	6	4	derecha	87,5
5 cm	8	2	derecha	86,4
10 cm	9	1	derecha	85,5
15 cm	10	0	derecha	87,3
20 cm	10	0	derecha	92,4

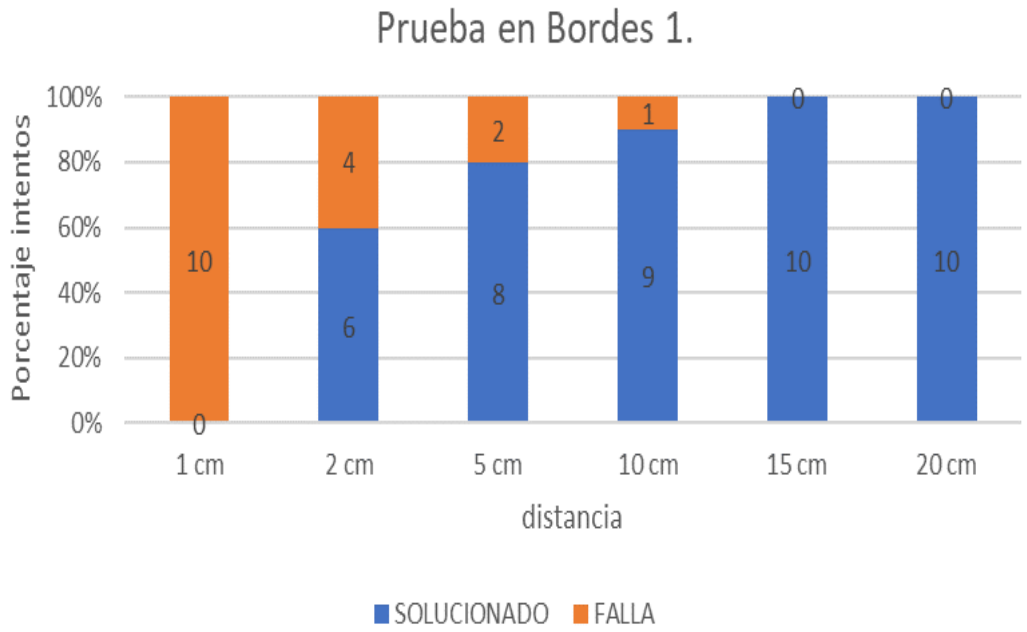


Figura51. Porcentaje de fallas prueba uno.

Descripción: Porcentaje de fallas totales analizadas de 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm para el primer modelo de Prueba (Figura.23) .

4.3.2 Modelo de pruebas 2

La prueba de bordes2 consiste en que el prototipo deba reconocer los objetos que implica una decisión de giro frente a él (figura.52):

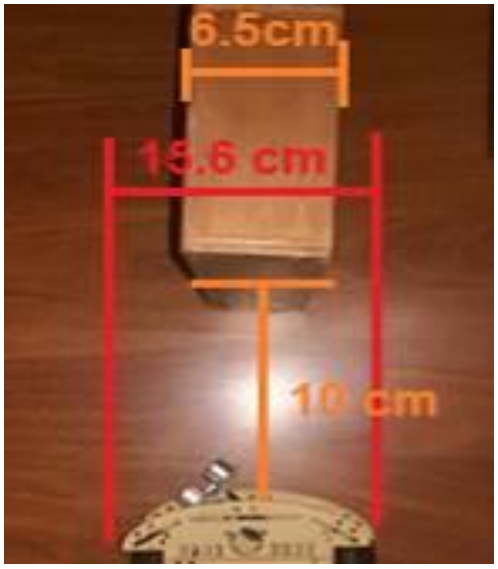


Figura52. Prueba dos prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia de un obstáculo menor al ancho frontal del prototipo (16.5 cm).

Los datos recopilados se muestran en la (Tabla.9) con su gráfico (figura.53).

Tabla9.
Prueba dos de Evasión.

Prueba en Bordes 2				
N DE PRUEBAS			10	
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	GRADOS
1 cm	1	9	derecha	89
2 cm	3	7	derecha	87
5 cm	7	3	derecha	86
10 cm	9	1	derecha	90
15 cm	10	0	derecha	91
20 cm	10	0	derecha	90

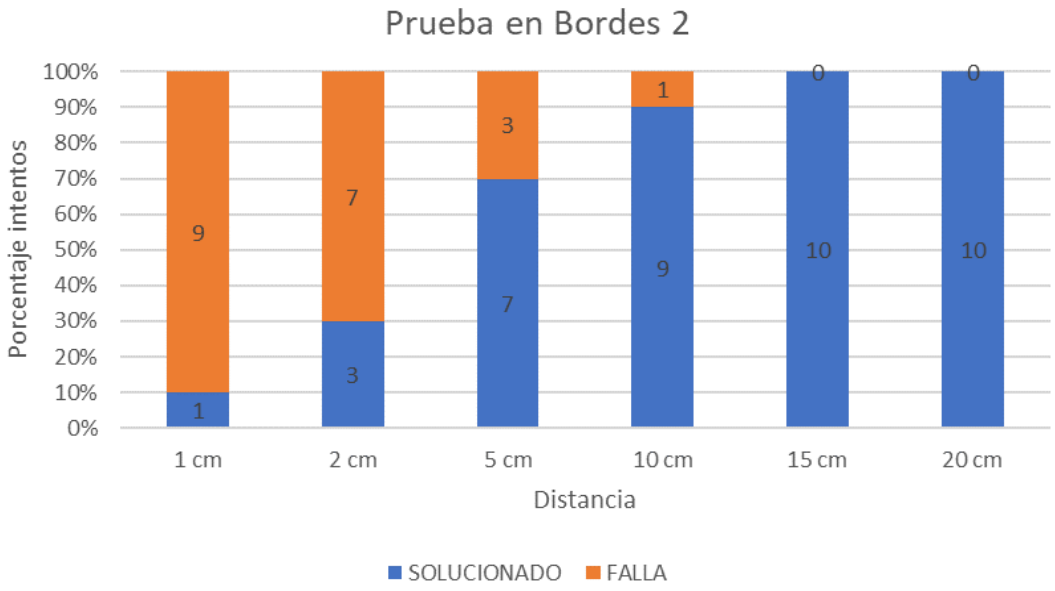


Figura53. Porcentaje de fallas prueba dos.

Descripción: Porcentaje de fallas analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm para el segundo modelo de Prueba (Figura.25) .

4.3.3 Modelo de pruebas 3

La prueba de esquinas consiste en que el prototipo deba reconocer los límites de las esquinas, tomando una decisión de giro frente a los objetos que se encuentran en la parte lateral izquierda (figura.54):

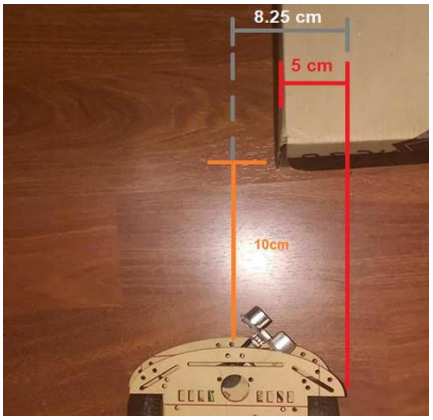


Figura54. Prueba tres prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia de un obstáculo menor a la mitad derecha del ancho prototipo (5 cm).

Los datos recopilados se muestran en la (Tabla.10) con su gráfico (figura.55).

Tabla10.

Prueba tres de Evasión.

Prueba en Esquinas 1				
N DE PRUEBAS		10		
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	GRADOS
1 cm	0	10	adelante	0
2 cm	2	8	izquierda	32,5
5 cm	7	3	izquierda	31,5
10 cm	9	1	izquierda	33,2
15 cm	9	1	izquierda	32,5
20 cm	10	0	izquierda	31,2

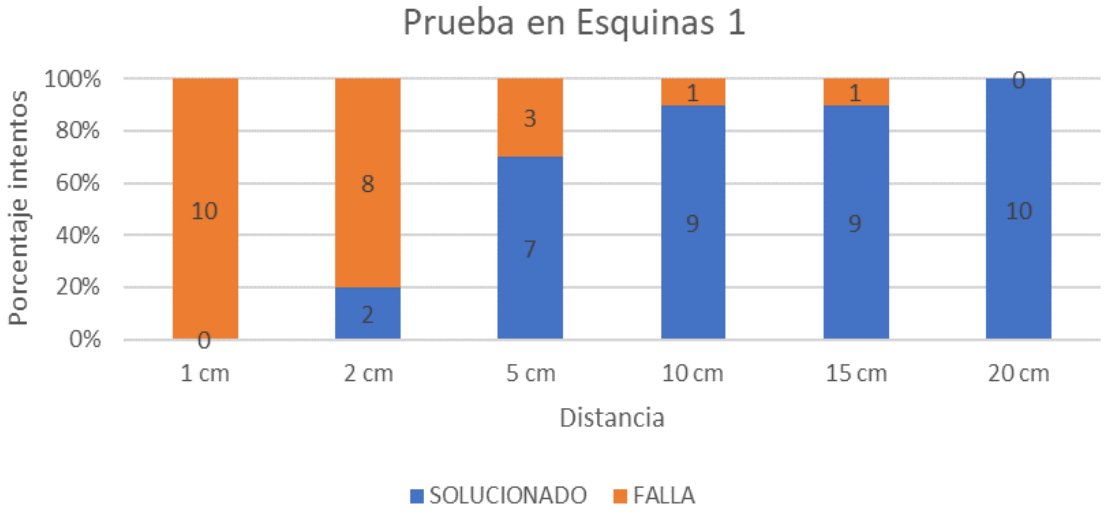


Figura55. Porcentaje de fallas prueba tres.

Descripción: Porcentaje de fallas analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm para el tercer modelo de Prueba (Figura.27) .

4.3.4 Modelo de pruebas 4

La prueba de esquinas consiste en reconocimiento de los límites de las esquinas de los objetos que se encuentran en la parte lateral derecha frente al prototipo (figura.56).

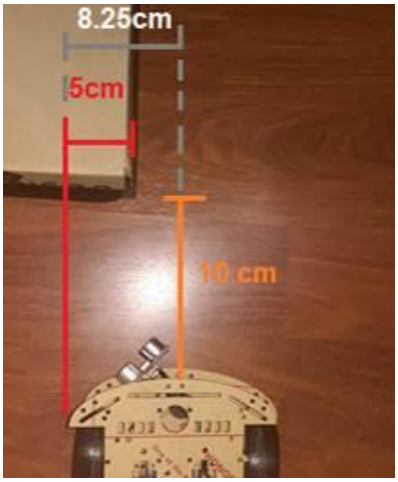


Figura56. Prueba cuatro prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia de un obstáculo menor a la mitad izquierda del ancho frontal del prototipo (5 cm).

Los datos recopilados se muestran en la (Tabla.11) con su gráfico (figura.57).

Tabla11.

Prueba cuatro de Evasión.

Prueba en Esquinas 2				
N DE PRUEBAS			10	
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	GRADOS
1 cm	1	9	derecha	32,3
2 cm	2	8	derecha	31,9
5 cm	8	2	derecha	31,5
10 cm	9	1	derecha	31
15 cm	10	0	derecha	29,6
20 cm	10	0	derecha	28,7

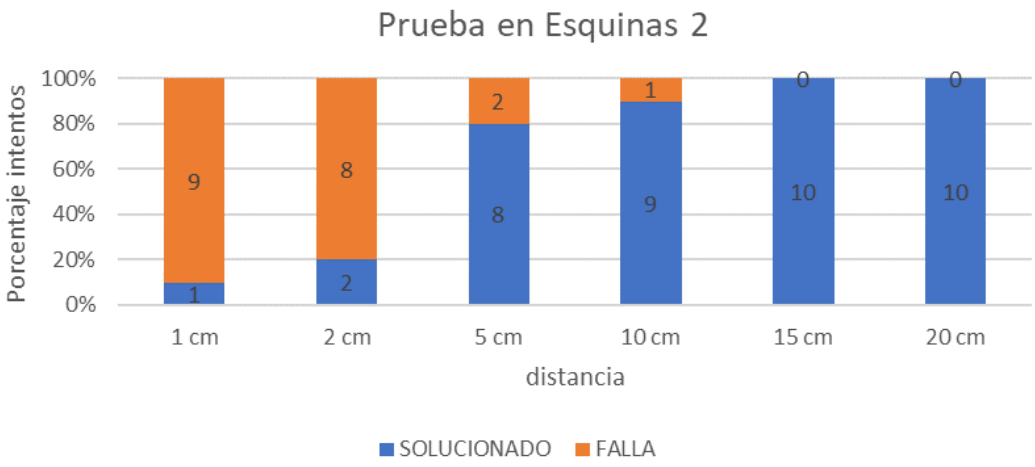


Figura57. Porcentaje de fallas prueba cuatro.

Descripción: Porcentaje de fallas totales analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm para el cuarto modelo de Prueba (Figura.29) .

4.3.5 Modelo de pruebas 5

La prueba de esquinas consiste en que el prototipo deba reconocer los límites de las esquinas de las habitaciones que se encuentran frente al prototipo y forman un ángulo de 90 grados tomando una decisión de giro (figura.58):

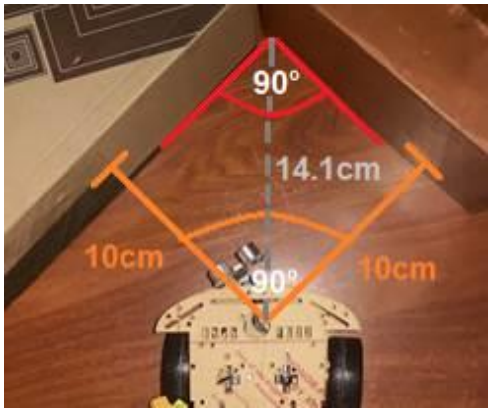


Figura58. Prueba cinco prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 14.1 cm de distancia de un obstáculo cuyo Angulo forma 90 grados frente al prototipo y a

10 cm en cada borde dentro de 90 grados de apertura de escaneo del prototipo.

Los datos recopilados se muestran en la (Tabla.12) con su gráfico (figura.59).

Tabla12.

Prueba cinco de Evasión.

Prueba en Esquinas Habitaciones				
N DE PRUEBAS		10		
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	GRADOS
1 cm	1	9	izquierda	88,3
2 cm	3	7	izquierda	88,2
5 cm	5	5	izquierda	87,3
10 cm	7	3	izquierda	90,2
15 cm	9	1	izquierda	91,3
20 cm	9	1	izquierda	90,5

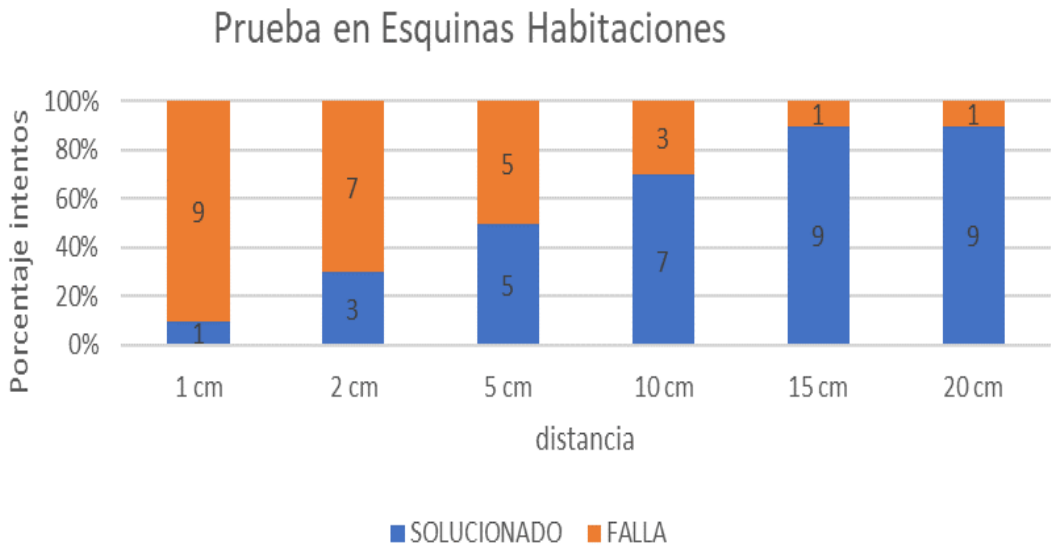


Figura59. Porcentaje de fallas prueba cinco.

Descripción: Porcentaje de fallas analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm para el quinto modelo de Prueba (Figura.48) .

4.3.6 Modelo de pruebas 6

La prueba de puntos críticos de pared derecha objeto frontal consiste en detectar el límite de los bordes de la pared derecha, con un objeto que se encuentra dentro del camino frontal del prototipo (figura.60).

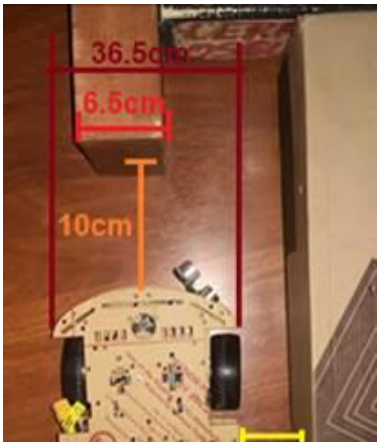


Figura60. Prueba seis prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia de un obstáculo menor a la mitad izquierda del ancho frontal del prototipo (6.5 cm) y a 5cm de separación de un objeto lateral derecho.

Los datos recopilados se muestran en la (Tabla.13) con su gráfico (figura.61).

Tabla13.
Prueba seis de Evasión.

Puntos críticos Pared derecha objeto frontal				
N DE PRUEBAS	10			
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	GRADOS
1 cm	0	10	derecha	33,4
2 cm	1	9	izquierda	89,2
5 cm	6	4	izquierda	88,3
10 cm	7	3	izquierda	87,9
15 cm	7	3	izquierda	91,5
20 cm	7	3	Izquierda	90,6

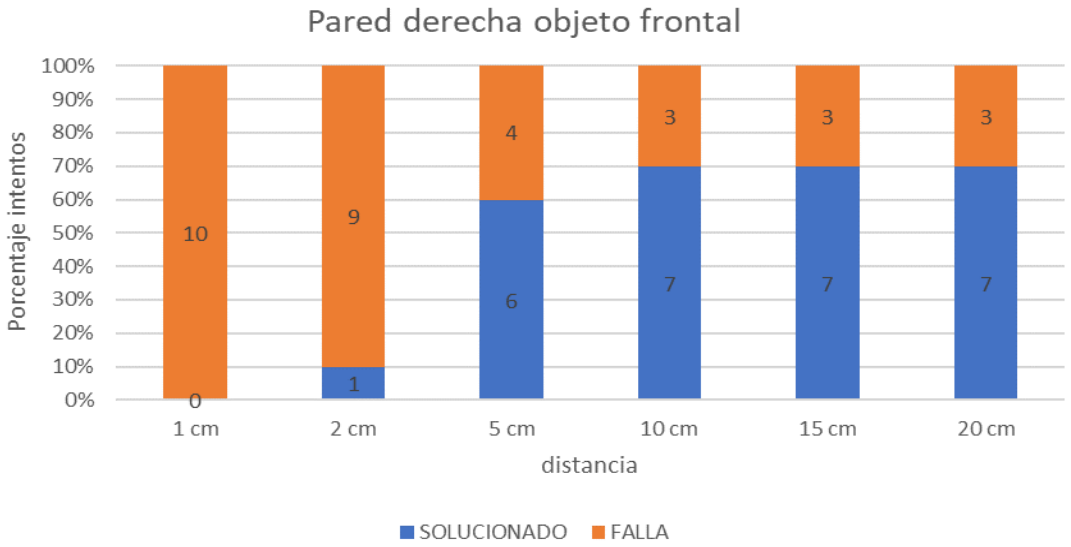


Figura61. Porcentaje de fallas prueba seis.

Descripción: Porcentaje de totales analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm para el sexto modelo de Prueba (Figura.33) .

4.3.7 Modelo de pruebas 7

La prueba de puntos críticos de pared izquierda objeto frontal consiste en detectar el límite de los bordes de la pared izquierda con un objeto que se encuentra dentro del camino frontal del prototipo (figura.62).

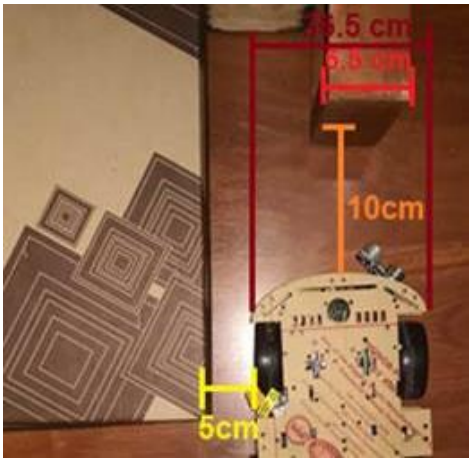


Figura62. Prueba siete prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia de un obstáculo y a 5 cm de separación de un objeto lateral derecho.

Los datos recopilados se muestran en la (Tabla.14) con su gráfico (figura.63).

Tabla14.

Prueba siete de evasión.

Puntos críticos Pared izquierda objeto frontal				
N DE PRUEBAS		10		
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	GRADOS
1 cm	0	10	izquierda	32,3
2 cm	1	9	derecha	31,9
5 cm	4	6	derecha	31,5
10 cm	6	4	derecha	31
15 cm	7	3	derecha	30,6
20 cm	8	2	derecha	30,7

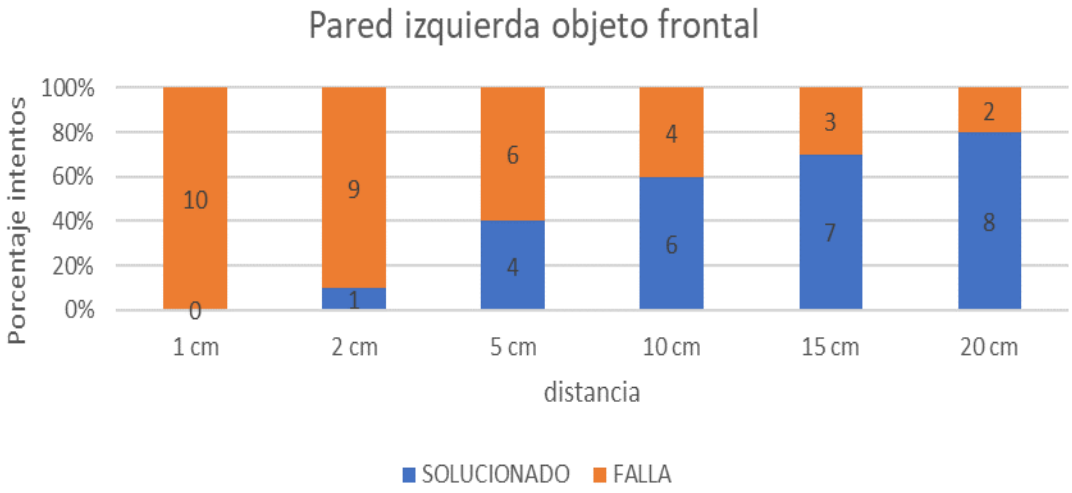


Figura63. Porcentaje de fallas prueba siete.

Descripción: Porcentaje de fallas analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm para el séptimo modelo de Prueba (Figura.35) .

4.3.8 Modelo de pruebas 8

La prueba de puntos críticos de pared izquierda pared derecha objeto frontal consiste en detectar límites de paredes en un espacio reducido donde no exista salida frontal posible (figura.64).

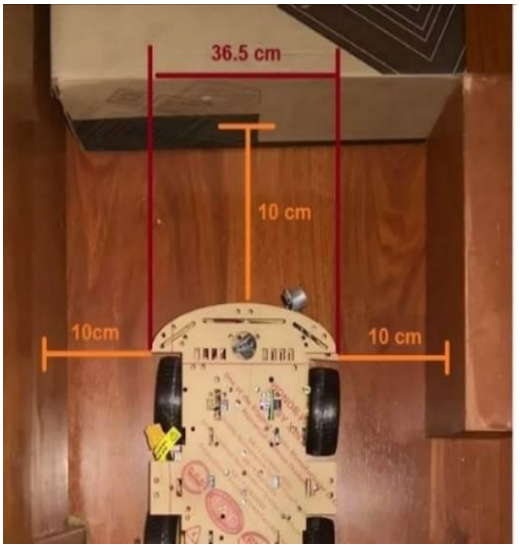


Figura64. Prueba ocho prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia de un obstáculo mayor al ancho frontal del prototipo (16.5 cm). con obstáculos lateral derecho e izquierdo a 10 cm de distancia.

Los datos recopilados se muestran en la (Tabla.15) con su gráfico (figura.65).

Tabla15.

Prueba ocho de Evasión.

Puntos críticos Pared izquierda Pared derecha objeto frontal				
N DE PRUEBAS			10	
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO C	GRADOS
1 cm	0	10	adelante	0
2 cm	1	10	derecha	89,2
5 cm	2	8	derecha	88,6
10 cm	8	2	derecha	90,3
15 cm	9	1	derecha	91
20 cm	9	1	derecha	90,2

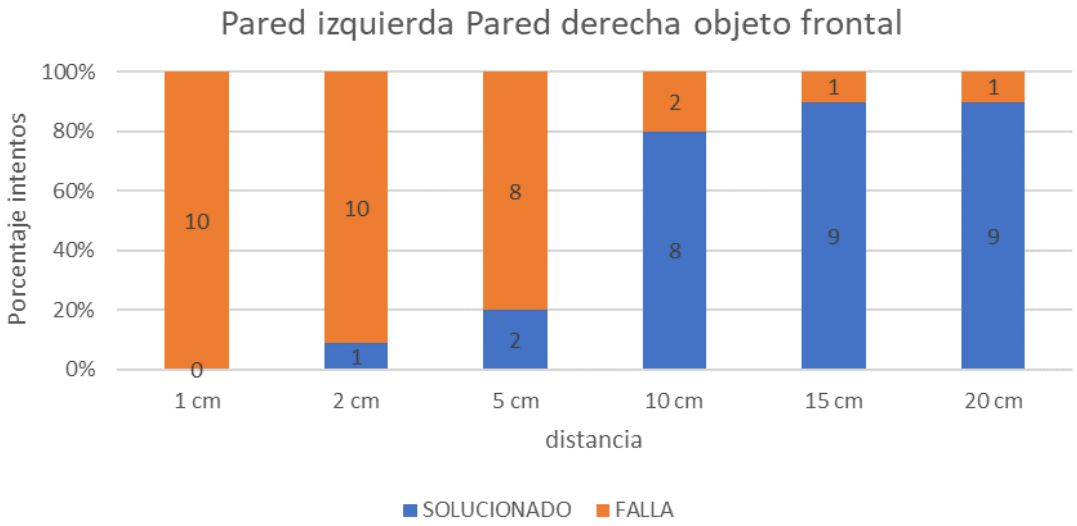


Figura65. Porcentaje de fallas prueba ocho.

Descripción: Porcentaje de fallas totales analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm delante del prototipo para el octavo modelo de Prueba (Figura.37) .

4.3.9 Modelo de pruebas 9

La prueba de área libre consiste en que el prototipo deba reconocer el área donde no exista ningún obstáculo para tomar una decisión de giro (figura.66).

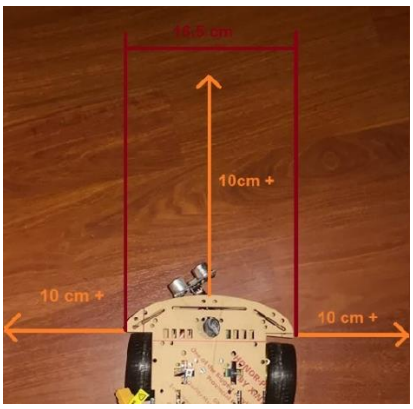


Figura66. Prueba Nueve Prototipo.

Descripción: Ejemplo de toma de datos del modelo de prueba a 10 cm de distancia en espacio libre, y sin obstáculos que tengan mayor ancho al ancho frontal del prototipo (16.5 cm).

Los datos recopilados se muestran en la (Tabla.16) con su gráfico (figura.67).

Tabla16.

Prueba nueve de Evasión.

Área Libre				
N DE PRUEBAS			10	
Distancia	SOLUCIONADO	FALLA	DECISIÓN GIRO	DISTANCIA cm
1 cm	10	0	adelante	16
2 cm	10	0	adelante	16
5 cm	10	0	adelante	16
10 cm	10	0	adelante	16
15 cm	10	0	adelante	16
20 cm	10	0	adelante	16

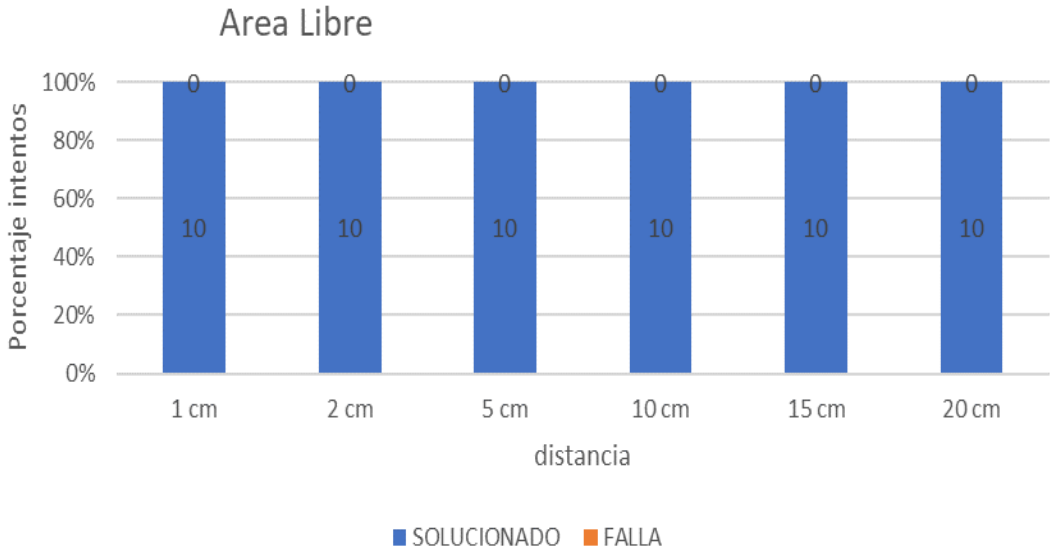


Figura67. Porcentaje de fallas prueba nueve.

Descripción: Porcentaje de fallas totales analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm de distancia para el noveno modelo de Prueba (Figura.56) .

4.3.10 Resultados de pruebas

En los resultados finales se muestra el total de fallas y soluciones realizadas por el prototipo a diferente distancia de los objetos (Tabla.17).

Tabla17.
Resultados de pruebas.

Resultado de pruebas Total			
Pruebas	10		
Distancia	SOLUCIONADO	FALLA	COMENTARIOS
1 cm	13	77	En giros las paredes chocan con el chasis del prototipo, el principal problema es la detección de obstáculos cercanos definido por el comportamiento técnico del sensor donde obtiene lecturas erróneas a partir de distancias menores a 2cm. Adicionalmente se necesita un margen de error para realizar un giro mayor al radio del tamaño del prototipo , es por eso que a 1 cm de distancia es imposible realizar un giro, es necesario retroceder para realizar una acción.
2 cm	29	62	En giros las paredes chocan con el chasis del prototipo, falla en detección de objetos cercanos, de acuerdo con el comportamiento técnico el sensor trabaja a partir de 2cm siendo este el límite ideal, sin embargo, trabajando a esta distancia existen errores de lectura .
5 cm	57	33	Errores en la toma de decisiones

			incorrecta , se debe a posibles lecturas erróneas por la superficie escaneada .
10 cm	74	16	Pocas fallas al momento de la detección, los errores más comunes se dan por el escaneo de objetos irregulares.
15 cm	81	9	Errores en la detección de objetos de forma irregular
20 cm	83	7	Errores en detección de objetos de forma irregulares

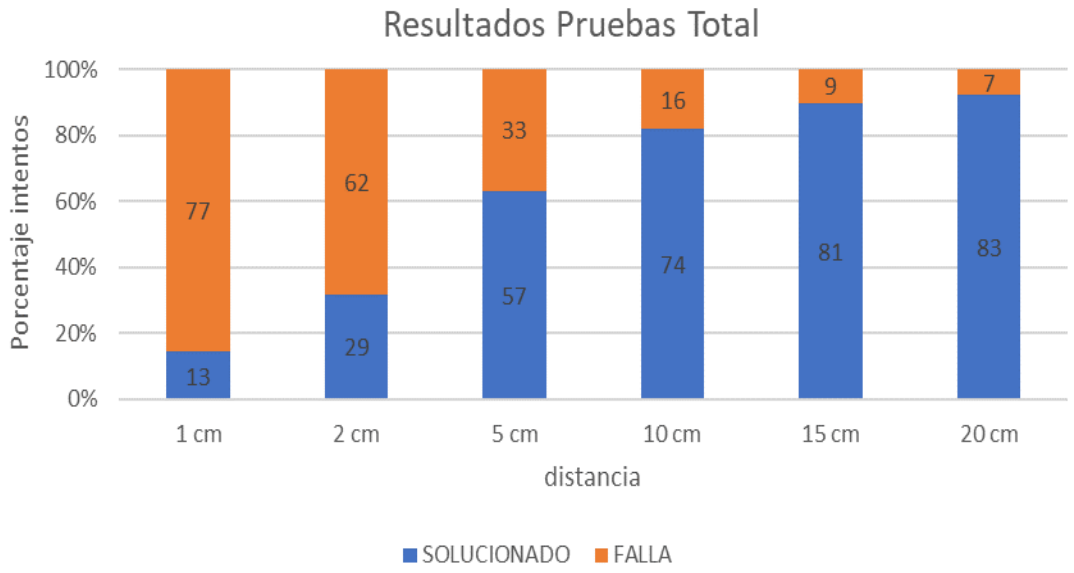


Figura68. Porcentaje de resultados de pruebas.

Descripción: Porcentaje de fallas totales analizadas de entre 10 pruebas de 1cm, 2cm, 5cm ,10cm, 15cm, 20 cm de distancia recopilados de los nueve modelos de Prueba .

4.4 Diseño de Optimización:

Los resultados de las pruebas realizadas arrojan problemas con la detección de obstáculos a distancias menores a 10 cm motivo por el cual se presentan elementos adicionales para mejorar la detección de obstáculos (figura.69).

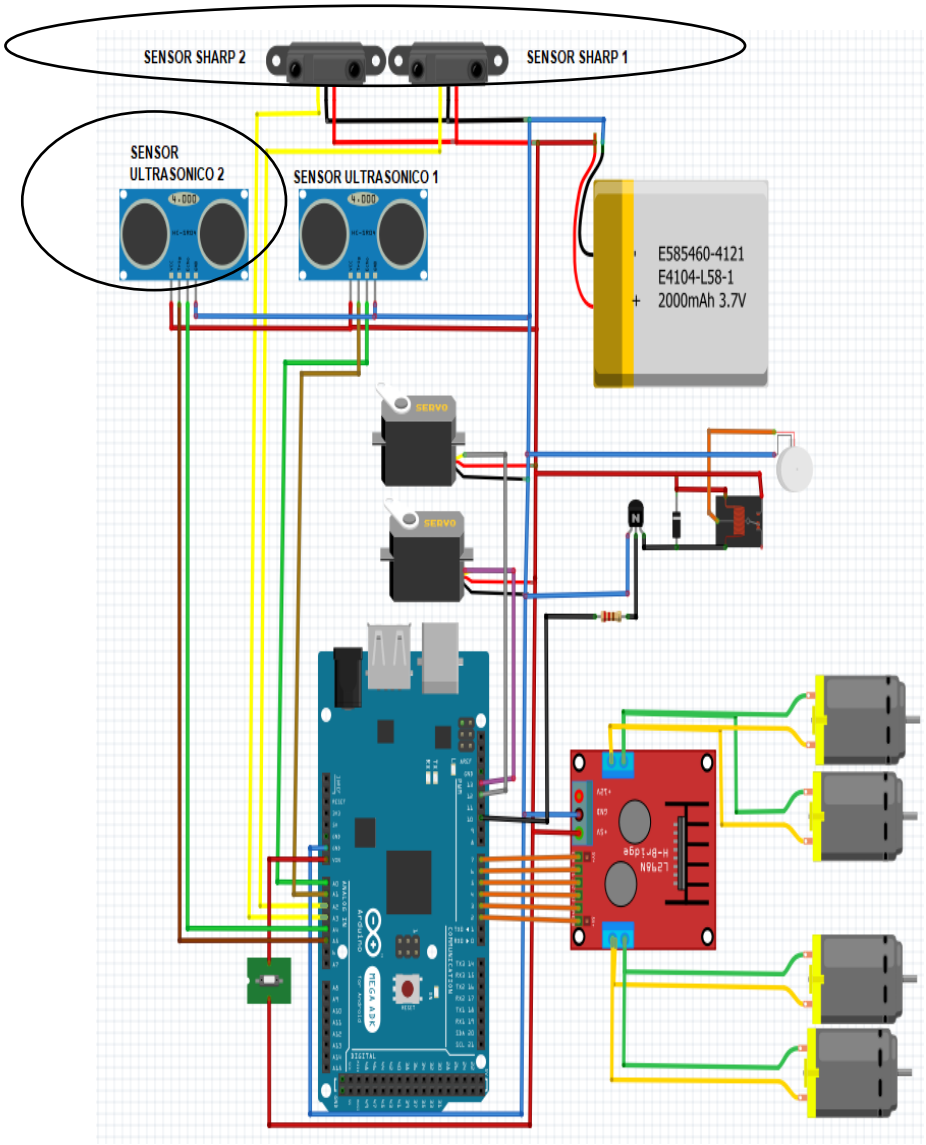


Figura69. Esquema optimización

Descripción: Se muestran elementos adicionales para mejorar la detección de obstáculos, un sensor ultrasónico *HC-SR04* y 2 sensores *Sharp GP2Y0A21*.

Para optimizar el escaneo se incorpora un sensor ultrasónico (*HC-SR04* dos) adicional en la parte posterior del prototipo para escanear objetos que se encuentran tras el prototipo en un área de 180 grados mientras retrocede.

Mientras avanza el prototipo el sensor ultrasónico (*HC-SR04* uno) escanea a 180 grados y los sensores *Sharp* 1-2 son implementados para detectar si un objeto frontal se encuentra a menos de 10 cm como medida de seguridad para evitar que avance el prototipo.

Durante el giro los sensores ultrasónicos serán ubicados a 180 grados del prototipo, si existe un objeto a menos de 20 cm durante el giro evitará la colisión , pero si no existe un objeto a menos de 20 cm podrá girar libremente.

El sensor (HC-SR04 dos) funcionara únicamente si el prototipo usa el método de reversa, que es activado si los sensores Sharp 1 o 2 detectan que un objeto se encuentra a menos de 10 cm.

Con el diseño de optimización las fallas que de 16% a 10 cm , 33% a 5 cm, 62% a 2 cm y 77% a 1 cm, deberán reducirse o en su defecto eliminarse por la incorporación de los Sensores Sharp que evitan que el prototipo este a menos de 10 cm de distancia de cualquier objeto detectado.

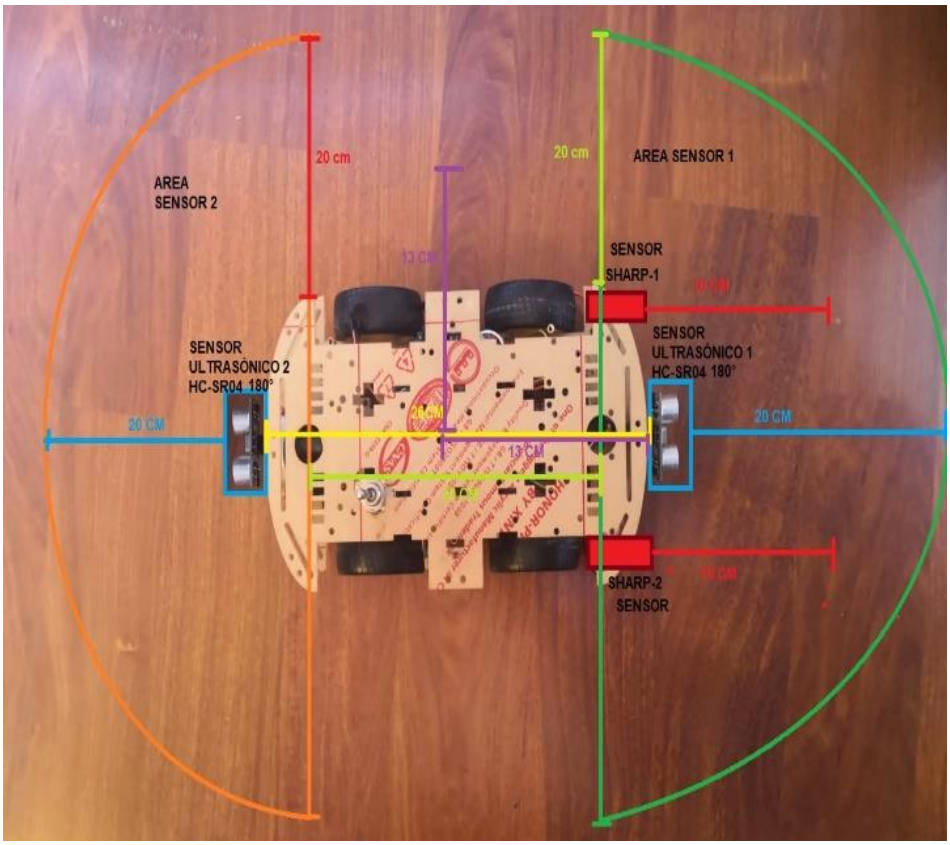


Figura70. Distribución sensores para optimización.

Descripción: Distribución de sensores Sharp 1-2 para detectar objetos a 10 cm, sensores ultrasónicos 1-2 para detectar área a 20 cm en un área 180 grados, con espacio de 20 cm laterales para detectar objetos en el giro.

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

La limpieza que realiza el prototipo se basa en el movimiento de los modelos de *iRobot* incluyendo movimiento en la limpieza, con un movimiento activo de 30 a 150 grados a la estructura del paño para mejorar la limpieza.

La bomba de agua del prototipo es el componente de mayor consumo de energía a 600 mA, no es posible conectar directamente a los pines Arduino porque esta corriente supera la corriente limite permitida para cada pin que es de 40 mA y no tendría la corriente suficiente para funcionar además de generan.

La batería Lipo del prototipo construido tiene una autonomía de 3.07 horas de trabajo, en comparación con los modelos de limpieza actuales que tienen 2 horas de autonomía esto quiere decir que el prototipo consume menos energía al realizar la misma función.

Al sensor *HC-SR04* no le afecta el índice de reflectancia de las escalas de colores para detectar objetos frente al prototipo por lo que se puede escanear todo tipo de objetos así sean transparentes.

Los objetos escaneados que presentan irregularidades y espacios en su estructura dan una lectura menos clara que objetos solidos con un ancho de 6 cm o más.

El controlador *l298n* y los motores reductores 1:48 son fundamentales para el control de la velocidad del prototipo, sin estos el movimiento serio poco fiable y aumentarían los errores en las pruebas de evasión al no tener una velocidad controlada.

El modelo de pruebas 8 nos revela que el prototipo tiene mayor dificultad a la hora de tomar una decisión, cuando la zona se encuentra a menos de 5 cm de distancia donde solo un 20% de las pruebas las realiza con éxito por ende se puede deducir que el prototipo no es óptimo para trabajar en zonas estrechas.

El escaneo mediante el uso del sensor *HC-SR04* falla más del 50% en el total de las pruebas realizadas a menos de 2cm de distancia, por consecuencia se puede deducir que se necesita de sensores de mayor precisión a distancias cortas.

En las pruebas totales se registra más de un 60% de éxito al tomar decisiones en zonas que se encuentra a más de 5cm de distancia del prototipo y llega a un máximo de 92.2% de éxito en decisiones cuando se encuentra a 20 cm de distancia del prototipo

Los objetos detectados por el prototipo deben tener una anchura mínima de 6.5 cm para ser detectados por el *HC-SR04* de tener una anchura menor se puede tener lecturas erróneas.

5.2 Recomendaciones

Cuando se incorpora el relé de 5v para la activación del motor bomba de agua o de otro elemento es necesario incorporar un diodo rectificador para evitar enclavamientos del relé.

Es recomendable no exigir la lectura de datos de medida en un tiempo menos a 20 ms para el sensor ultrasónico *HC-Sr04* ya que es el mínimo tiempo necesario para que el pulso sea disparado según datos de los fabricantes, si se usa a más de 20 ms puede dar lecturas erróneas .

Es fundamental no sobrecargar los pines Arduino exigiéndole mayor corriente de la permitida por pin 40mA ya que este puede sufrir errores de lectura y sobrecalentamiento de la placa.

Es importante tener en cuenta que el suelo es un factor que incide en el movimiento de rotación del prototipo, mientas más rugoso y menos liso es el suelo más precisión tendrá el giro y movimiento del prototipo.

Se recomienda la carga y descarga total de la batería lipo para lograr un tiempo de vida mayor, usar un balanceador de carga para su recarga y no superar la temperatura de funcionamiento de entre 20 a 60 grados.

REFERENCIAS

iRobot Corporation. (2017). *Braava Jet: guia de usuario.* Estados Unidos: iRobot Corporation. Recuperado el 20 de noviembre de 2018 de https://homesupport.irobot.com/euf/assets/images/faqs/braavajet/200/manual/es-ES.pdf

Adán Esteban Suárez Arriaga, A. M. (2015). *Plataforma movil omnidireccional de cuatro llantas Suecas (Mecanum) en configuracíon "AB".* México: Universidad Autonóma de México.

Álvarez, I. B. (2016). *Construcción y control de un robot móvil.* Sevilla: Universidad de Sevilla.

Angle, C. (2014). *Estados Unidos Patente nº WC: 4440171.*

Aracely Yandún, N. S. (2012). *Planeación y seguimiento de trayectorias para un robot móvil.* Quito: Escuela Politécnica Naciona.

Arduino. (2018). *What is Arduino?*. Recuperado el 22 de noviembre de 2018 de https://www.arduino.cc/en/Guide/Introduction

Barquin, J. M. (2007). *Aspiradora inteligente para limpieza automatica de suelos.* Madrid: Universidad Pontificia Comillas.

Baturone, A. O. (2005). *Robótica: manipuladores y robots móviles.* Barcelona: Universidad de Sevilla.

Bazaga, A. R. (2015). *OpenCV: Librería de Visión por Computador*. Recuperado el 10 noviembre del 2018 de oficina de software libre Universidad de la Laguna: https://osl.ull.es/software-libre/opencv-libreria-vision-computador/

Blanco Abia, C. (2014). *Desarrollo de un sistema de navegación para un robot móvil.* España: Universidad de Valladolid.

Bravo Sánchez, F. Á., & Forero Guzmán, A. (2012). *La robótica como un recurso para facilitar el aprendizaje y desarrollo de competencias generales.* Salamanca: Universidad de Salamanca.

Butiax Robotica Educativa. (2016). *Robot Butiá 2.0.* Recuperado el 2 de diciembre de 2018 de fing.edu.uy: https://www.fing.edu.uy/inco/proyectos/butia/files/docs_butia2/manual_de_construccion_rev1.pdf

Caladin, L. I. (2008). *Modelado Cinematico y Control de Robot Moviles con Ruedas.* Valencia: Universidad Politecnica de Valencia.

Carlos Andrés Vélez Carvajal, J. A. (2012). Techniques' evaluationfor motion planning, using exact and adaptive cell descomposition. *Revista Facultad de Ingenieri a, UPTC*, 47.

Cetronic. (2018). *Ultrasonic Ranging Module HC - SR04.* Recuperado el 04 de diciembre de 2018 de https://descargas.cetronic.es/HCSR04.pdf

COGNEX. (2018). *Introduccion a la visión artificial.* Recuperado el 11 de diciembre de 2018 de https://www.cognex.com/es-ar/library/media/files/17151.pdf

Córdoba, I. S. (2016). *MOPFLOOR ROBOT.* Terrassa: Universidad Politécnica de Cataluña.

Daniel Estiven Álvarez Gaviria, J. A. (2012). Identificación de un ambiente desconocido e implementación de un grafo de visibilidad para el cambio de configuración de un robot. *Revista virtual Universidad Catolica del Norte*, 229.

Elecfreaks. (2018). *HCSR04 Module.* Recuperado el 12 de diciembre de 2018 de https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf

Flores, M. S. (2018). *Curso básico de Arduino.* Mexico: Mecatronica Latam.

Fuentes, V. M. (2014). *Introducción a la plataforma Arduino y al Sensor ultrasónico HC-SR04.* Madrid: Universisad Carlos III de Madrid.

Gobierno de España. (2018). *Vision Artificial.* Recuperado el 13 de diciembre de 2018 de http://visionartificial.fpcat.cat/wp-content/uploads/Conocimientos.pdf

Grupo Halley. (2018). *Introducci´on a Arduino.* Santander: Universidad Industrial de Santander.

Guillermo Barbadillo Villanueva, A. P. (2012). *PROYECTO ARDUINO: SUMO ROBÓTICO.* España: Universidad pública de Navarra.

Herrador, R. E. (2009). *Guía de Usuario de Arduino.* Córdoba: Universidad de Córdoba.

IFM. (2018). *Sensores de visión para la detección y evaluación de objetos y escenas.* Recuperado el 24 de noviembre de 2018 de https://www.ifm.com/ifmweb/downcont.nsf/files/ifm-vision-sensors-industrial-imaging-ES/$file/ifm-vision-sensors-industrial-imaging-ES.pdf

Ignacio Buioli, J. P. (2018). *Processing un lenguaje al alcance de todos.* Recuperado el 15 de novimebre del 2018 de http://laurence.com.ar/artes/comun/Processing_un_lenguaje_al%20alcance_de_todos.pdf

IRobot Braava. (2013). *Manual de propietario serie 300 Robot para fregar suelo.* Estados Unidos: iRobot.

iRobot Corporation. (2014). *iRobot, Scooba.* Estados Unidos: iRobot Corporation. .

J. A. Vázquez, M. V.-V. (2007). *Seguimiento de Trayectorias de un Robot Movil Omnidireccional Basado en el Modelo Dinamico.* México: CINVESTAV-IPN.

Johnny Novillo-Vicuña, D. H. (2018). *Arduino y el internet de las cosas.* Alicante: 3ciencias.

Keyenge. (2018). *Que son los sensores de vision*. Recuperado el 05 de diciembre de 2018 de https://www.keyence.com.mx/ss/products/sensor/sensorbasics/vision/info/

López, A. M. (2015). *Control de robots autónomos mediante microcontrolador Arduino.* España: Universidad de Valladolid.

López, C. C. (2018). *OpenCv.* Recuperado el 25 de noviembre de 2018 de Universidad del Itsmo: www.unistmo.edu.mx/~jjap/Opencv.pptx

Maestre, J. M. (2015). *Dómotica para ingenieros.* España: Ediciones Paraninfo.

Marras, J. J. (2015). *Diagramas de Voronoi*. Recuperado el 29 de noviembre de 2018 de https://jjromeromarras.wordpress.com /2015/07/12/diagramas-de-voronoi/

Mateu, J. C. (2014). *Desarrollo de un pequeño robot móvil basado en microcontrolador.* Valencia: Universitat Politècnica de València.

Microkits. (2018). *Microkits Electrónica y robotica.* Recuperado el 28 de noviembre de 2018 de https:// www.microkitselectronica.com/media/attachment/file/h/c/hcsr04.pdf

Miralles, Á. S. (2016). *Modelos de entorno y planificación de trayectorias.* Madrid: Universidad Pontificia Comillas.

naylampmechatronics. (2018). *Sensor Infrarrojo CNY70*. Recuperado el 02 de diciembre de 2018 de https://naylampmechatronics .com/sensores-proximidad/15-sensor-infrarrojo-cny70.html

Nicolas GOILAV, G. L. (2016). *Arduino: Aprender a desarrollar para crear objetos inteligentes.* Barcelona: ENI.

Oter, O. O. (2018). *Trabajo de Microrrobots: Sensores de medida por Contacto.* Recuperado el 05 de diciembre de 2018 de http://www.alcabot.com/alcabot/seminario2006/Trabajos/OlgaOlme daOter.pdf

Parker, D. (2018). *Express bot con sensor de tacto*. Recuperado el 28 de noviembre de 2018 de http://intelirobot.com.mx/blog/lego-mindstorms/express-bot-con-sensor-de-tacto/

Patricia Quintero Alvarez, ,. J. (2016). *Técnicas para evasión de obstáculos en Robótica Móvil.* Mexico: Instituto Tecnológico de Nuevo León.

Pellicer, J. L. (2018). *Universidad Politécnica de Valencia.* Recuperado el 18 de noviembre de 2018 de http://users.dsic.upv.es/~jlinares/grafics/processing_spa_1.pdf

Pérez, D. R. (2004). *Ruteo en Grafos Geométricos.* Madrid: Universidad politécnica de Madrid.

Pérez, V. D. (2015). *Implementacion de algoritmos de determinacion de rutas para el robotino de festo.* Quito: Escuela Politecnica Nacional.

Processing. (2018). *Processing*. Recuperado el 23 de noviembre de 2018 de Processing.org: https://processing.org/download/

Re-Trust. (2018). *Planos*. Recuperado el 15 de noviembre de 2018 de http://re-trust.org/2018/07/21/planos-de-casas-una-planta/planos-de-casas-120-m2-una-planta-dise-o-simple-planos-de-casas-una-planta/

Rodríguez, C. (2015). *Arduino + Processing: Parte I.* Recuperado el 20 de octubre de 2018 de http://panamahitek.com/arduino-processing-parte-i-comandos-basicos/

Roncancio, B. R. (2017). *Generacion de trayectos en un entorno dinamico usando una plataforma robotica diferencial.* Bogota: Fundacion Universitaria Los Libertadores.

Roomba. (2018).*Braava jet*. Recuperado el 28 de octubre de 2018 de https://www.roomba.it/braava-jet.html

Sharp Corporation. (2011). *GP2Y0A21YK0F.* Estados Unidos: Sharp.co.

somosindustria. (2018). *Lanza Keyence nuevo sensor de visión simple*. Recuperado el 02 de noviembre de 2018 de https://www.somosindustria.com/articulo/lanza-keyence-nuevo-sensor-de-vision-simple/

spanish.alibaba. (2018). *Chasis acrilico*. Recuperado el 15 de noviembre de 2018 de https://spanish.alibaba.com/product-detail/Acrylic-Laser-cut-Robot-Pieces-Perspex- 60793293466.html?spm=a2700.galleryofferlist.normalList.352.775d63d3u1Kx7w

SparkFun. (2018). *GP2Y0A21YK.* Recuperado el 20 de octubre de 2018 de https://www.sparkfun.com/datasheets/Components/GP2Y0A21YK.pdf

Tedeschi, B. (2014). *Home Robots*. Recuperado el 25 de octubre de 2018 de https://www.nytimes.com/2014/12/25/garden/10-home-robots-to-lighten-your-domestic-chores.html

Tejedor, R. J. (2003). *Vivienda Domótica* Recuperado el 14 de octubre de 2018 de https://www.ramonmillan.com/tutoriales/domotica.php

Torres, T. D. (2018). *Navegacion y Planificacion de Rutas*. Recuperado el 16 de noviembre de 2018 de porprofesionalmic.files.wordpress.com: https://porprofesionalmic.files.wordpress.com/2015/09/investigacion-documental-navegacion-planificacion-rutas.pdf

Torres, T. D. (2018). *Navegacion y Planificacion de Rutas*. Recuperado el 26 de noviembre de porprofesionalmic.files.wordpress.com: https://porprofesionalmic.files.wordpress.com/2015/09/investigacion-documental-navegacion-planificacion-rutas.pdf

tresdprinttech. (2018). *ModuloHC-06*. Recuperado el 14 de octubre de 2018 de https://tresdprinttech.com/sonido/150-sensor-ultrasonico-hc-sr04.html

webpersonal. (2018). *ModeloCRG.* Recuperado el 22 de octubre de 2018 de http://webpersonal.uma.es

ANEXOS

Anexo 1. Código Arduino Para detección de objetos.

```
//Librerias
#include <math.h>
#include <Servo.h>.

//CONTROL MOTORES DERECHO
int enA = 7;  // control velocidad por ancho de pulso
int driver_motor_IN1 = 6;
int driver_motor_IN2 = 5;
//CONTROL MOTORES IZQUIERDO
int enB = 2;  // control velocidad ancho de pulso
int driver_motor_IN3 = 4;
int driver_motor_IN4 = 3;
const int trigPin = A1;
const int echoPin = A0;
//VARIABLES
long duracion;
int distancia;

//Variables de deteccion de puntos de inicio y fin de objetos
int inicio;
int dist1_1;
int dist1_2;
int fin;

int inicio2;
int dist2_1;
int dist2_2;
int fin2;

int inicio3;
int dist3_1;
int dist3_2;

int fin3;
int inicio4;
int fin4;
//variables de ventana de error para reconocimiento
//de los objetos
int Verror;
int verdad;

int obj1;
int obj2;

int resultado;
int resultado2;
int modelo;
```

```
//Variables para calcular angulo y giro de ruedas

int angulo;
int angulorueda;
double angulogiro;

Servo Servomotor1;
Servo Servomotor2;

int SIR1 = A2;
int SIR2 = A3;
//variables para lectura del sensor en distancia
int lectura,cm;
int s1;
int s2;
int s3;

void setup() {
  //control radar
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(10, OUTPUT);
  Serial.begin(9600);
  Servomotor1.attach(12);
  Servomotor2.attach(13);

//control motores declaracion SALIDAS
//  pinMode(enA, OUTPUT);
  pinMode(driver_motor_IN1, OUTPUT);
  pinMode(driver_motor_IN2, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(driver_motor_IN3, OUTPUT);
  pinMode(driver_motor_IN4, OUTPUT);
}


 void loop() {//FUNCION PRINCIPAL

 //Inicio variables

  resultado=0;
  verdad=0;
  s1=40;
  s2=40;


 s3=sensor3();// Metodo de escaneo y limpieza

 //Envio datos Serial
 //Serial.print("resultado FINAL");
 //Serial.print("seguir ");
 //delay(100);

 //Condicion de distancia**
 if (s3<=25){
   Parar();//Metodo Parar
   escaneo1();//Metodo de escaneo
   }

  s1=0;
  s2=0;
  s3=0;

 }
```

```
//FUNCION LIMPIEZA Y ESCANEO
 int sensor3(){
//inicio de variables
int calcular=0;
int agua=0;
int j=3;
//INICIO de servomotor
  for (int i = 60; i <= 130; i++) {
    Servomotor1.write(i);
    delay(10);



//Contador servomotor Limpiesa grados
    if(j>=120){
      j=30;
      }
    //Condicion de calculo de distancia
    if(calcularDistancia2()<=20||s1<=25||s2<=25){
    calcular = calcularDistancia2();
    //Calculo de distancia de sensor
    if(calcular<=25){
       return calcular;
       }if(s1<=25){
           return s1;
                    }
        if(s2<=25){
           return s2;
           }else{
            j=j+6;
   Servomotor2.write(j);//movimiento sevo limpiesa
   // condicion activacion bomba agua
    if(agua>=40){
       Parar();
       digitalWrite(10, HIGH);//activacion bomba
       delay(400);
       digitalWrite(10, LOW);//desactivacion bomba
       delay(400);
       agua=0;

                   }
     adelante();//metodo para avansar
    agua=agua+1;
```

```
void escaneol(){
      Parar();//Metodo parar
        //CONTROL DE ESCANEO
        //Control servomotor grados de escaneo
        //condicion For para movimiento de servo
       Servomotor1.write(15);
       Servomotor2.write(30);
  for (int i = 15; i <= 165; i++) {

  distancia = calcularDistancia();//Calcular distancia
     //Pruebas serial
     //Serial.print(i);
     // Serial.print(",");
     // Serial.print(distancia);
     //  Serial.print(".");
     // delay(5);
  Servomotor1.write(i);
  delay(35);
    // Serial.print("Sensor:" );
    //Serial.print(cm);
    //calcular distancia de sensor



      calcular=40;
      }
     }else{
        j=j+6;// contador servomotor limpieza
        Servomotor2.write(j);//movimiento servo limpieza
        if(agua>=40){
           Parar();
           digitalWrite(10, HIGH);//activacion bomba
           delay(400);
           digitalWrite(10, LOW);//desactivacion bomba
           delay(400);
           agua=0;
            }
         adelante();
         agua=agua+1;
         calcular=40;
        }
      Servomotor1.write(60);//inicio servomotor sensor
       }
       return calcular;//
      }
```

```
//RECONOCIMIENTO DE OBJETO
if(resultado2==0){ //variable para deteccion de objetos previos
  if(inicio2<=0){  //valor inicial de objetol encontrado angulo
      if( distancia>=1&&distancia<=25){
         inicio2 = i ;
        inicio=i;
       dist2_1=distancia;
    //Serial.print("inicio ");
    //Serial.print(inicio2);
    //Serial.print(",");
    //Serial.print(distancia);
    //Serial.print(".");
    //delay(2000);
            }
           }
  if(fin2<=0){ // valor final de objeto encontrado angulo
      if( distancia<=25){
          dist2_2=distancia; //toma de dato distancia
                        }
        if(inicio2>=1){
        if(i>=165){
         fin2=i;
         fin=i;
                   }  }
       if (inicio2>=1&&distancia>=25) {
         fin2 = i;
         fin=i;
    //Prueba serial
    //Serial.print("final ");
    //Serial.print(fin2);
    //Serial.print(",");
    //Serial.print(dist2_2);
    //Serial.print(".");
    //delay(2000);
                              }
            }
```

```
//Condicion deteccion mas de un objeto
 if(inicio2>=1&&fin2>=1){

    resultado= obstaculo(inicio2,fin2);

    //Prueba serial
    /*Serial.print(" OBST1= ");
    Serial.print(resultado);
    Serial.print(" I1= ");
    Serial.print(inicio2);
    Serial.print(" DF1= ");
    Serial.print(dist2_1);
    Serial.print(" F1= ");
    Serial.print(fin2);
    Serial.print(" DF11= ");
    Serial.print(dist2_2);
    delay(2000); */

//Condicion si el reconocimiento resulto exitoso

      if (resultado==1){
         resultado2=resultado;
         obj1=1;
         //Reinicio de Variables
          inicio=0;
          fin=0;
          inicio=inicio2;
          fin=fin2;
          inicio4=inicio;
          fin4=fin;

      //angulorueda = calculolobjeto(inicio2,fin2);
      //Serial.print("ANGULO = ");
      //  Serial.print(angulorueda);
                        }

      inicio2=0;
      fin2=0;
                    }
                        }
```

```
 ////////////////////////////////
////RECONOCIMIENTO DE OBJETO2////
////////////////////////////////

if(resultado2==1){
 if(inicio3<=0){//valor inicial de objetol encontrado angulo
 if( distancia>=1&&distancia<=25){
    inicio3 = i ;
    dist3_1=distancia;

    //Prueba serial
    /*Serial.print(" inicio  2 ");
    Serial.print(inicio3);
    Serial.print(",");
    Serial.print(dist3_1);
    Serial.print(".");
    delay(2000);*/
  }
  }
   if(fin3<=0){ // valor final de objeto 2 encontrado angulo
 if( distancia<=25){
  dist3_2=distancia;
 }
   if (inicio3>=1&&distancia>=30 ) {
      fin3 = i;
    //Prueba serial
    /*Serial.print("final 2 ");
    Serial.print(fin3);
    Serial.print(",");
    Serial.print(dist3_2);
    Serial.print(".");
    delay(2000);*/
    }
      }
```

```
if(inicio3>=1&&fin3>=1){
     resultado= obstaculo( inicio3,fin3);

   //Prueba serial
   /*Serial.print(" OBSTACULO 2 ES = ");
   Serial.print(resultado);
   Serial.print(" INICIO 2 ES = ");
   Serial.print(inicio3);
   Serial.print(" DINICIO 2 ES = ");
   Serial.print(dist3_1);
   Serial.print(" FIN 2 ES = ");
   Serial.print(fin3);
   Serial.print(" DFIN 2 ES = ");
   Serial.print(dist3_2);
   delay(2000);*/

     if(resultado==1){
       resultado2=3;
       obj2=1;
       }
   inicio3=0;
   fin3=0;
        }
         }
           }
resultado2=0;


//////////////////////
//Condicion sin objetos
   if(obj1==0&&obj2==0){
  //Prueba serial
  //Serial.print("sin objetos de frente ");
  //delay(2000);
  //adelantelimpia();
  //adelante();
  //Parar();
  }
//Condicion si se encontro objeto delante
 if(obj1==1&&obj2==0){

    //Prueba serial
    //Serial.print("slo 1 obj  detectado ");
    //Serial.print("ANGULO");
    //Serial.print(angulorueda);
    //delay(1000);
    /*Serial.print(" INICIO4 ");
     Serial.print(inicio4);
     delay(1000);
     Serial.print(" FIN4 ");
     Serial.print(fin4);
     delay(1000);*/
```

```
 if(fin4<=100&&inicio4>=75){
  //Prueba Serial
  /*  Serial.print("GIRO DERECHA COMPLETO");
     delay(1000);*/
 int a=0;
 int b=0;
 //Calculo de distancias criticas
 Servomotor1.write(150);
 a=calcularDistancia();
 delay(100);
 Servomotor1.write(30);
 b=calcularDistancia();
 delay(100);
 Servomotor1.write(90);
 delay(10);
   if(a>=b){
        Parar();
        giroIzquierda();
        inicio4=0;
        fin4=0;
    }else{
        Parar();
        giroDerecha();
        inicio4=0;
        fin4=0;}
   }else{
        Parar();
        calculolobjeto2(inicio4,fin4);
        inicio4=0;
        fin4=0;
          }
   Parar();//Metodo Parar
          }
 if(obj1==1&&obj2==1){

//Prueba serial
/*Serial.print(objetos detectados ");
delay(2000);*/
Parar();
giroDerecha();

}
Parar();

obj1=0;
obj2=0;
```

```
  //Control servomotor grados escaneo regreso
  for (int i = 165;i>= 60; i--) {
    Servomotor1.write(i);
    delay(10);
    //Prueba serial
    //calcular distancias sensor
    //distancia = calcularDistancia();
    //pruebas de serial
    //Serial.print(i);
    //Serial.print(",");
    //Serial.print(distancia);
    //Serial.print(".");
    //delay(5);
  }
  //Reinicio de Variables
  fin=0;
  inicio=0;
  inicio2=0;
  fin2=0;
  inicio3=0;
  fin3=0;

  }

//Metodo para ventana de error de escaneo de objetos
int obstaculo(int a,int b){
verdad= 0;
Verror = 8;
inicio = a ;
fin = b;
  if(fin-inicio>= Verror){
  //Prueba serial
  //Serial.print("OBJETO ENCONTRADO");
  //delay(20);
  verdad=1;
  }
    return verdad;
    }
```

```
///Metodo para calculo de Bisectriz espacio libre

//Parametros de inicio y fin de objeto
void calculolobjeto2(int inicio, int fin){

  int ini=inicio;
  int fi=fin;
  double x=0;
  double y=0;
  int agiro;
  double variable;

    //Pruebas Serial
    /*Serial.print(" INICIO ");
     Serial.print(ini);
     delay(1000);
     Serial.print(" FIN ");
     Serial.print(fi);
     delay(1000);*/
//Condiciones para objetos sin peligro choque

  if (ini>=135&&fi<=180 || fi<=45&&inicio >=0){
    adelantelimpia();

    //Prueba Serial
    /*Serial.print("ADELANTE CALCULO ");
    delay(1000);*/
    }

  //Calculo de  Bisectriz bordes de objetos y angulo libre
  if(ini>=15&&fi<=165){
    x=(165-fi)/2;
    y=(ini-15)/2;
        //Prueba serial
        /*Serial.print(" CALCULO X Y : X ");
        Serial.print(x);
        Serial.print(" Y ");
        Serial.print(y);
        delay(2000);*/
  }

  if(x>=y){
        //Prueba Serial
        //Serial.print(" GIRO IZQUIERDA ");
        //delay(2000);
        giroIzquierdal(x);//Giro con angulo calculado

    }else{
        //Prueba Serial
        //Serial.print(" DERECHA GIRO ");
        //delay(2000);
        giroDerechal(y);}//Giro con angulo Calculado
        Parar();
  }
```

```
////////
//Metodo para limpieza y movimiento frontal
void adelantelimpia() {

   //Activacion servomotores y ruedas
   Servomotor2.write(30);
   digitalWrite(10, HIGH);
   delay(200);
   digitalWrite(10, LOW);
   delay(200);
   digitalWrite(driver_motor_IN1, LOW);
  digitalWrite(driver_motor_IN2, HIGH);
  analogWrite(enA, 90);
  digitalWrite(driver_motor_IN3, HIGH);
  digitalWrite(driver_motor_IN4, LOW);
  analogWrite(enB, 90);
  //Movimiento servomotor limpieza
  for (int i = 30; i <= 150; i++) {
   Servomotor2.write(i);
  delayMicroseconds(1145);
    }
  for (int i = 150; i > 30; i--) {
  Servomotor2.write(i);
  delayMicroseconds(1145);
  }

}

//Metodo avanzar
void adelante() {
   //Activacion ruedas
  digitalWrite(driver_motor_IN1, LOW);
  digitalWrite(driver_motor_IN2, HIGH);
  analogWrite(enA, 80);
  digitalWrite(driver_motor_IN3, HIGH);
  digitalWrite(driver_motor_IN4, LOW);
  analogWrite(enB, 80);
}
```

```
//Metodo Retroceder
void atras() {
  //Activacion de ruedas
  digitalWrite(driver_motor_IN1,HIGH );
  digitalWrite(driver_motor_IN2,LOW);
  analogWrite(enA, 128);
  digitalWrite(driver_motor_IN3, LOW);
  digitalWrite(driver_motor_IN4, HIGH);
  analogWrite(enB, 128);
  delay(1900);
}
//Metodo Giro Derecha sin angulo
void giroDerecha()
{

  digitalWrite(driver_motor_IN1, LOW);
  digitalWrite(driver_motor_IN2, HIGH);
  analogWrite(enA, 95);
  digitalWrite(driver_motor_IN3, LOW);
  digitalWrite(driver_motor_IN4, HIGH);
  analogWrite(enB, 95);
  delay(950);
}
 //Metodo Giro derecha con angulo
void giroDerechal(int a)
{
  //Calculo de tiempo activacion
  angulogiro= a*12.2;
  //Activacion ruedas
  digitalWrite(driver_motor_IN1, LOW);
  digitalWrite(driver_motor_IN2, HIGH);
  analogWrite(enA,95);
  digitalWrite(driver_motor_IN3, LOW);
  digitalWrite(driver_motor_IN4, HIGH);
  analogWrite(enB, 95);
  delay(angulogiro);
}
```

```
//Metodo Giro Izquierda sin angulo
void giroIzquierda()
{
   //Activacion ruedas
  digitalWrite(driver_motor_IN1, HIGH);
  digitalWrite(driver_motor_IN2, LOW);
  analogWrite(enA, 95);
  digitalWrite(driver_motor_IN3, HIGH);
  digitalWrite(driver_motor_IN4, LOW);
  analogWrite(enB, 95);
  delay(950) ;
}
void giroIzquierdal(int a)
{
  //Calculo de tiempo activacion
  angulogiro= (a*12.2)+100 ;
  //Activacion ruedas
  digitalWrite(driver_motor_IN1, HIGH);
  digitalWrite(driver_motor_IN2, LOW);
  analogWrite(enA, 128);
  digitalWrite(driver_motor_IN3, HIGH);
  digitalWrite(driver_motor_IN4, LOW);
  analogWrite(enB, 128);
  delay(angulogiro) ;
}

//Metodo para detener ruedas
void Parar()
{
  //Desactivacion ruedas
  digitalWrite(driver_motor_IN1, LOW);
  digitalWrite(driver_motor_IN2, LOW);

  digitalWrite(driver_motor_IN3, LOW);
  digitalWrite(driver_motor_IN4, LOW);
  delay(500);
}
```

```
//Metodo 1 para calcular distancia HCSR-04
int calcularDistancia() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(150);
  digitalWrite(trigPin, HIGH);      //Envio de pulso
  delayMicroseconds(150);
  digitalWrite(trigPin, LOW);
  duracion = pulseIn(echoPin, HIGH);//obtencion ancho de pulso
  distancia=duracion*0.034/2;
  return distancia;
}
//Metodo 2 para calcular distancia HCSR-04
int calcularDistancia2(){
  long t; //timepo que demora en llegar el eco
  long d; //distancia en centimetros

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);      //Enviamos un pulso de 10us
  digitalWrite(trigPin, LOW);

  t = pulseIn(echoPin, HIGH); //obtenemos el ancho del pulso
  d = t/59;                   //escalamos el tiempo a una distancia en cm
 // Prueba Serial
 // Serial.print("Distancia: ");
 // Serial.print(d);
 // Serial.print("cm");
 // Serial.println();
 // delay(10);
  return d;
  }
```

Anexo 2. Código Processing Para detección de objetos.

radar

```
import processing.serial.*; //lbrerias
import java.awt.event.KeyEvent;
import java.io.IOException;
Serial myPort;           //variable serial
                        //variables
String angulo="";
String distancia="";
String datos="";
String objeto;
float distanciaPixeles;
int iAngulo, iDistancia;
int indice1=0;
int indice2=0;
PFont Fuente;
void setup() {

 size (1920, 1080);                          // generar area de trabajo
 smooth();                                  //generar geometria con bordes lisos
 myPort = new Serial(this,"COM3",9600);   // abrir interfaz serial
 myPort.bufferUntil('.');                  // leer valor mientras exista un punto
 Fuente = loadFont("OCRAExtended-30.vlw"); // estilo de letras
}
void draw() {                                 //funcion de dibujo principal area de trabajo

  fill(98,245,31);                           //cargar el color de letras RGB
  textFont(Fuente);                         //fuente
  noStroke();
  fill(0,4);                                 //fondo color opacidad, valor entre negro y gris
  rect(0, 0, width, height-height*0.065);    //simulacion del valor de movimiento recta
  fill(98,245,31);                          //color verde recta

  drawRadar();                               //dibujar radar
  drawLine();                                //dibujar lineas
  drawObject();                             //dibujo de obstaculo
  drawText();                               //texto del radar
}
```

```
void serialEvent (Serial myPort) {   //interrupcion dentro del programa

  datos = myPort.readStringUntil('.');   //leer los datos en data
  datos = datos.substring(0,datos.length()-1); //guardar  subcadena inicializada en indice 0, eliminar ultimo caracter del .

  indice1 = datos.indexOf(",");                          // separador de variables
  angulo= datos.substring(0, indice1);                    // angulo tomado de la subcadena del indice 0 separado por la coma
  distancia= datos.substring(indice1+1, datos.length());    //subcadena de distancia tomada indice siguiente a la coma
  iAngulo = int(angulo);                                 //transoformar angulo en entero
  iDistancia = int(distancia);                            //transformar la distancia en entero
}
void drawRadar() {
  pushMatrix(); //guardar la matriz de coordenadas del grafico
  translate(width/2,height-height*0.074);
  noFill(); //sin relleno
  strokeWeight(2); //ancho de los bordes
  stroke(98,245,31); //bordes de lineas
// dibujar lineas del arco (x,y(centro), ancho, alto, inicio ,fin )
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI); //funcion arco (tamaño arco inicio y fin)
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);

  line(-width/2,0,width/2,0);                                    //funcion linea , coordenadas x1,y1,x1,y2)(usamos grados para ubicar)
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);

  popMatrix(); //reestablece las coordenadas del grafico despues de las transformaciones

}
```

```
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074);
  strokeWeight(9);
  stroke(255,10,10); // red color
  distanciaPixeles = iDistancia*((height-height*0.1666)*0.025); // distancia del sensor de cm a pixeles.

  if(iDistancia<40){

  line(distanciaPixeles*cos(radians(iAngulo)),-distanciaPixeles*sin(radians(iAngulo)),
  (width-width*0.505)*cos(radians(iAngulo)),-(width-width*0.505)*sin(radians(iAngulo)));
  }
  popMatrix();
}
void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // traslada el objeto linea en coordenadas x y y
  //se dibujan las lineas acorde la variable angulo que retorna por el puerto serial
  line(0,0,(height-height*0.12)*cos(radians(iAngulo)),-(height-height*0.12)*sin(radians(iAngulo)));
  popMatrix();
}
```

```
void drawText() {

  pushMatrix();
  if(iDistancia>40) { // condicion que indique que se encuentra dentro o fuera del rango
  objeto = "sin peligro de colision"; // fuera de rango si es verdadero
  }
  else {
  objeto = "objetos detectados";  //dentro del rango si es falso
  }
  fill(0,0,0); //relleno
  noStroke();
  rect(0, height-height*0.0648, width, height); //dibujo de la interfaz rectangulo
  fill(98,245,31);
  textSize(25);

  // texto dibujado en la pantalla para la lectura
  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
  textSize(40);
  text("Objeto: " + objeto, width-width*0.875, height-height*0.0277); // nos buestra la visualizacion si el objeto se encuentra en rango
  text("Angulo: " + iAngulo +" °", width-width*0.48, height-height*0.0277); // nos muestra la visualizacion del angulo
  text("Distancia: ", width-width*0.26, height-height*0.0277); //texto distancia

  if(iDistancia<40) {
  text("        " + iDistancia +" cm", width-width*0.225, height-height*0.0277); //nos muestra la distancia del objeto en tiempo real
  }


  // texto de los angulos. (with/2 mitad de pantalla  heigh-heig valor desde el tope de pantalla)
  textSize(25);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();
  translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-width/2*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();
  translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-width/2*sin(radians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();
  translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
```