ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

"SISTEMA DE UBICACIÓN ESPACIAL DE PLATAFORMA MÓVIL MEDIANTE SENSORES EN UN MAPA DETERMINADO"

Trabajo de titulación

Tipo: Dispositivo Tecnológico

Presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTOR:

BRYAN ISRAEL NÚÑEZ SÁNCHEZ

Riobamba – Ecuador

2021

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

"SISTEMA DE UBICACIÓN ESPACIAL DE PLATAFORMA MÓVIL MEDIANTE SENSORES EN UN MAPA DETERMINADO"

Trabajo de Integración Curricular

Tipo: Dispositivo Tecnológico

Presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTOR: BRYAN ISRAEL NÚÑEZ SÁNCHEZ

DIRECTOR: Ing. FAUSTO RAMIRO CABRERA AGUAYO MSc.

Riobamba - Ecuador

2021

Yo, Bryan Israel Núñez Sánchez, declaro que el presente trabajo de titulación es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este trabajo de titulación; El patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 16 de noviembre del 2021.

Bryan Israel Núñez Sánchez
180411636-4

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELCTRÓNICA

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; Tipo: Dispositivo Tecnológico, **"SISTEMA DE UBICACIÓN ESPACIAL DE PLATAFORMA MÓVIL MEDIANTE SENSORES EN UN MAPA DETERMINADO"**, realizado por el señor: **BRYAN ISRAEL NÚÑEZ SÁNCHEZ**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. Andrés Fernando Morocho Caiza Msc. **PRESIDENTE DEL TRIBUNAL**	ANDRES FERNANDO MOROCHO CAIZA Firmado digitalmente porANDRES FERNANDO MOROCHO CAIZA DN: cn=ANDRES FERNANDO MOROCHO CAIZA c=EC o=SECURITY DATA S.A. 1 ou=ENTIDAD DE CERTIFICACION DE INFORMACION Motivo:Apruebo este documento Ubicación: Fecha:2021-11-17 17:33-05:00	16 de noviembre del 2021
Ing. Fausto Ramiro Cabrera Aguayo Msc. **DIRECTOR DEL TRABAJO DE TITULACIÓN**	FAUSTO RAMIRO CABRERA AGUAYO Digitally signed by FAUSTO RAMIRO CABRERA AGUAYO Date: 2021.11.19 10:19:13 -05'00'	16 de noviembre del 2021
Ing. Edwin Vinicio Altamirano Santillán Msc. **MIEMBRO DEL TRIBUNAL**	EDWIN VINICIO ALTAMIRANO SANTILLAN Firmado digitalmente por EDWIN VINICIO ALTAMIRANO SANTILLAN Fecha: 2021.11.17 16:26:28 -05'00'	16 de noviembre del 2021

DEDICATORIA

Dedico este trabajo a Dios siendo mi compañero, a mi madre como ejemplo, a mi hermano como apoyo, a mi hija Cielo, que fue es y será el motor en mi vida y a todos aquellos que me acompañaron en este largo recorrido de vida universitaria, aportando con mi formación.

Bryan.

AGRADECIMIENTO

No encuentro las palabras para demostrar mi sentimiento de gratitud, pero menciono a mi madre Nancy Sánchez por nunca darse por vencida, a mi hermano Steven Núñez por la paciencia, a Anita Sancho por nunca soltar mi mano en tan difícil camino y a los amigos verdaderos que siempre me supieron apoyar; por entregarme el tesoro más valioso del mundo, su tiempo.

Bryan.

TABLA DE CONTENIDO

CAPÍTULO II

CAPÍTULO III

CAPÍTULO IV

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

La finalidad de este estudio fue comprobar la similitud con la que un robot y un programa de software realizan un proceso de movilidad y evasión de obstáculos en un mapa determinado, mientras se conoce su ubicación simultánea. Se construyó una plataforma móvil, para lo cual el modelo se escogió de acuerdo a las necesidades de movilidad dentro del mapa determinado, acoplándose así al algoritmo construido para el mismo. El programa de software fue diseñado de tal manera que este permita la comunicación entre la plataforma y el ordenador, demostrando así en la interfaz la movilidad simultánea. Para la correcta navegación de la plataforma dentro del mapa determinado se utilizaron sensores de distancia que permitieron conocer los obstáculos presentes en el mismo, la distancia óptima fue de 35 cm, de acuerdo a las dimensiones de la plataforma para evitar riesgos de colisión. Se utilizó un cable serial para la comunicación, el cual hace efectivo al sistema en el momento de realizar la interfaz simultánea. La plataforma llega a navegar de manera simultánea en el mapa determinado y en la interfaz de software mostrando su ubicación real. Para realizar la construcción de la plataforma, fue necesaria la búsqueda de un material óptimo para cubrir los elementos internos del mismo, se emplearon tres sensores para cubrir la parte delantera y así evitar cualquier colisión en el giro o recorrido de la misma; siendo la comunicación serial la más segura para contribuir con la ubicación. Se recomienda realizar un análisis previo de los motores para llegar a un punto ideal de velocidad de los mismos, y así se facilite el correcto reconocimiento del mapa determinado.

Palabras clave: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA> <SENSORES DE DISTANCIA> <EVASIÓN DE OBSTÁCULOS> <INTERFAZ DE SOFTWARE> <MATLAB (SOFTWARE)>.

ABSTRACT

The purpose of this study was to verify the similarity with which a robot and a software program perform a process of mobility and obstacle avoidance on a given map, while their simultaneous location is known. A mobile platform was built, for which the model was chosen according to the mobility needs within the determined map, thus coupling to the algorithm built for it. The software program was designed in such a way that it allows the communication between the platform and the computer, thus demonstrating simultaneous mobility at the interface. For the correct navigation of the platform within the determined map, distance sensors were used which allowed to know the obstacles in it, the optimal distance was 35 cm, according to the dimensions of the platform to avoid collision risks. A serial cable was used for communication, which makes the system effective at the time of performing the simultaneous interface. The platform manages to navigate simultaneously on the determined map and in the software interface showing its actual location. To carry out the construction of the platform, it was necessary to search for an optimal material to cover the internal elements in it, three sensors were used to cover the front so that it can avoid any collision in its turnor route; serial communication being the safest to contribute to the Location. It is recommended to carry out a previous analysis of the motors to reach their ideal point of speed, and thus facilitate the correct recognition of the determined map.

Keywords: <TECHNOLOGY AND ENGINEERING SCIENCES> <SENSORS OF DISTANCE> <OBSTACLE EVASION> <SOFTWARE INTERFACE> <MATLAB (SOFTWARE)>

INTRODUCCIÓN

La robótica es una rama de la tecnología que estudia la logística, el diseño y la construcción de máquinas capaces de desempeñar diversas tareas. Estas creaciones se dan, primeramente, de manera mental y luego, en forma física, controladas por un sistema computacional.

La planificación del movimiento es un área de investigación fundamental en robótica. La investigación en robótica en los últimos 20 años ha estado fuertemente dirigida hacia el desarrollo de robots autónomos y móviles, es decir de robots capaces de decidir su propio movimiento con el propósito de ejecutar una determinada tarea con objetos en un espacio de trabajo. Una tarea típica consiste por ejemplo en transportar ciertos objetos desde un lugar inicial hasta su destino final. Para la realización de la tarea el robot debe ser capaz de encontrar un camino que evite todo tipo de colisiones con los objetos del espacio de trabajo. Evidentemente, el robot debe razonar acerca de la geometría del espacio de trabajo y de los objetos del mismo.

La planificación del movimiento de robots trata del desarrollo de algoritmos geométricos eficientes que doten al robot con la capacidad de generar caminos libres de colisión. La planificación no es solo uno de los problemas fundamentales en robótica, es quizás el más estudiado. Los primeros esfuerzos para desarrollar técnicas de planificación deterministas mostraron que es un desafío computacional incluso para sistemas simples. Los métodos exactos de la hoja de ruta, como los gráficos de visibilidad, los diagramas de Voronoi, la triangulación de Delaunay intenta capturar la conectividad del espacio de trabajo del robot. Los métodos de descomposición celular, en los que el espacio de trabajo se subdivide en celdas pequeñas, se han aplicado en robótica

ANTECEDENTES

La robótica ha sido una de las disciplinas que mayor desarrollo ha experimentado en los últimos tiempos. Hoy día es frecuente encontrar robots operando en diversos ámbitos de aplicación: producción en cadena, exploración de entornos inaccesibles para el ser humano (profundidades submarinas, misiones espaciales, etc.), manipulación de materiales peligrosos (explosivos, ambientes tóxicos, etc.), trabajos de elevada precisión, etc. Sin embargo, muchos de ellos necesitan un operador humano para dirigir la tarea. El esfuerzo científico de los últimos años intenta proporcionar autonomía a los robots.

En el caso de la robótica móvil, los robots han de desplazarse en su entorno. Uno de los retos consiste en caracterizar dicho entorno a través de sensores, identificar obstáculos y zonas de paso, e incluso ubicarse con la mayor precisión posible con respecto a un sistema de referencia dado. Una vez hecho esto, el robot debe trazar una trayectoria hacia su destino, y después tratar de

seguirla lo más fielmente posible. A veces, no se dispone de ningún mapa y el robot ha de maniobrar de acuerdo a lo que detecta a través de sus sensores. Este tipo de técnicas reciben el nombre de "navegación reactiva" y una de las formas de implementarla es generando trayectorias locales destinadas a evitar los obstáculos que el robot va reconociendo. En cualquier caso, el cálculo de trayectorias en un entorno dado es un aspecto de gran relevancia en el ámbito de la robótica.

FORMULACIÓN DEL PROBLEMA

¿Es posible diseñar un sistema para ubicar una plataforma móvil en un mapa determinado?

SISTEMATIZACIÓN DEL PROBLEMA

¿Existen trabajos relacionados al tema de ubicación espacial de plataformas móviles?

¿Qué componentes de hardware y software son necesarios para la construcción de una plataforma móvil para navegación en un plano de un mapa conocido?

¿Existe un sistema que permita conocer la posición espacial de un robot móvil en un mapa conocido?

¿El sistema es capaz de navegar en un mapa conocido y el robot permite conocer su ubicación en el espacio simultáneamente?

OBJETIVOS

OBJETIVO GENERAL

-Diseñar un sistema de ubicación espacial de plataforma móvil mediante sensores en un mapa determinado.

OBJETIVOS ESPECÍFICOS

-Investigar trabajos relacionados al tema de ubicación espacial de plataformas móviles.

-Seleccionar componentes de hardware y software necesarios para la construcción de una plataforma móvil para navegación en un plano de un mapa conocido.

-Diseñar un sistema que permita conocer la posición espacial de un robot móvil en un mapa conocido simultáneamente.

-Comprobar si el robot es capaz de navegar en un mapa conocido y el sistema permite conocer su ubicación en el espacio simultáneamente.

CAPÍTULO I

1. MARCO TEÓRICO

En este apartado se detalla la información con argumentos de investigadores y obras publicadas en revistas científicas.

1.1. Plataforma

La palabra plataforma utilizada para referirse a un robot estructurado limitando costos en los procesos de diseño, así como reduciendo la complejidad del mismo para evitar fallos de tipo mecánico o de algún otro (Soriano et al. s.f.).

1.2. Robots

En una obra teatral en 1921 aparece por primera vez el termino robot haciendo referencia a uno o varios conjuntos de partes mecánicas autómatas asociadas a una idea de trabajo, tratando de imitar movimientos humanos, mediante el control automático de procesos se pretende crear sistemas en los que se minimice la intervención de agentes externos («Robótica: Manipuladores y Robots Móviles - Aníbal Ollero Baturone - Google Libros» s.f.).

Robot se define como manipulador multifuncional reprogramable; el mismo que mediante instrucciones o movimientos programados puede realizar una serie de tareas diferentes (Robótica - control de robots manipuladores - REYES, Fernando - Google Libros, s.f.).

1.2.1. Clasificación de robots

Actualmente existe gran variedad de robots para una aplicación con diferentes componentes mecánicos y estructuras geométricas, los mismos se pueden clasificar de manera general como se muestra en la Tabla 1-1.

Tabla 1-1: Clasificación de robots

CLASIFICACIÓN DE ROBOTS		
Móviles	Terrestres: Ruedas, patas	
	Submarinos, aeroespaciales	
Humanoides	Diseño complejo	
Industriales	Brazos mecánicos	Robots manipuladores

Fuente: (Baturone, s.f.).

Realizado por: Núñez, Bryan, 2020.

1.2.2. Robots Móviles

Los robots adquieren la característica de movilidad cuando cuentan con la capacidad de poder desplazarse de un lugar a otro.

Cuentan con una comunicación inalámbrica, alámbrica o actualmente con el avance de la tecnología robots capaces de decidir una trayectoria por si mismos contando con inteligencia artificial, para su desplazamiento por ruedas o extremidades; brindándoles la capacidad de realizar varias actividades. Los robots son utilizados en la industria tienen múltiples funciones os encontramos como se muestra en la Figura 1-1 (Saha, 1998).

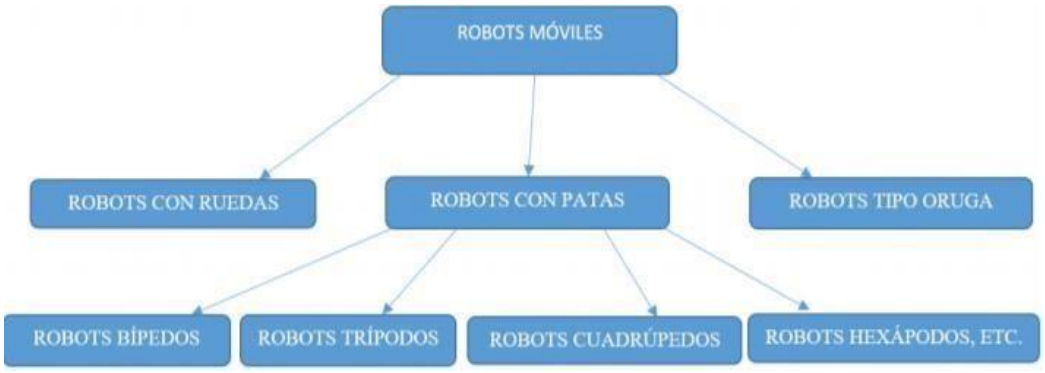


Figura 1-1: Clasificación de robots móviles

Realizado por: Núñez, Bryan, 2020.

1.3. Morfología del Robot

La morfología de un robot envuelve conceptos de las características básicas de la composición del prototipo, detalla la parte funcional, mecánica, diseño, así como el diseño de la parte lógica programable creada y ejecutada para llegar a cumplir el objetivo de darle las capacidades requeridas por el robot al cumplir un trabajo específico (Barrientos Peñin Balaguer Arcin 1996).

1.4. Robot con ruedas

Los entornos irregulares y regulares o con obstáculos surgen como un parámetro a cubrir por parte de los robots los mismos que necesitan desplazarse a través de los mismos. Los desplazamientos se ven limitados por el acceso al campo o el mismo campo donde se va a trabajar, ya que los mismos pueden tener un desplazamiento diferencial o sincronizado; esto altera el margen de estabilidad del prototipo buscando así nuevas soluciones. Se muestra en la Figura 2-1 (Ollero, 2001).

Figura 2-1: Robots con Ruedas
Fuente: (Freire et al., 2015).

1.4.1. Akerman

Robots de cuatro ruedas con diseños comunes para ser utilizados en campos abiertos, con funciones simples como el trabajo de peso, se describe de manera gráfica en la Figura 3-1 (Ollero, 2001).

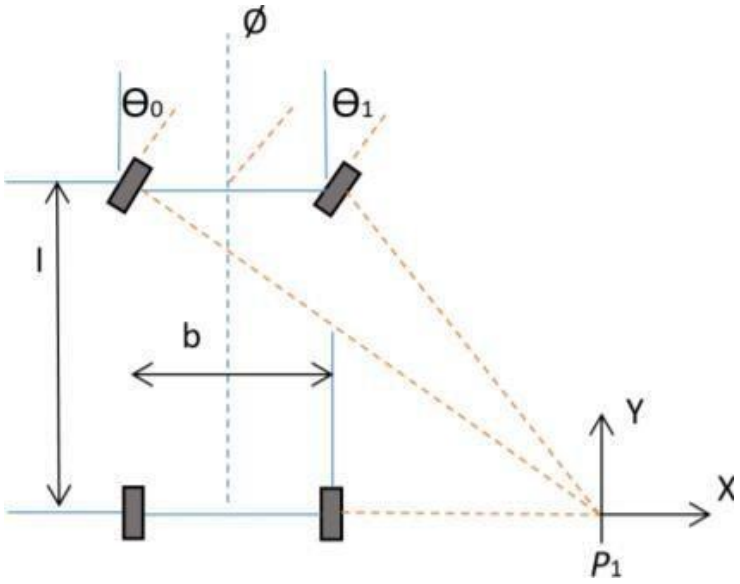


Figura 3-1: Sistema Ackerman

Fuente: (Ollero, 2001).

1.4.2. Triciclos

Prototipos construidos y diseñados en base a tres ruedas, las mismas distribuidas así: dos ruedas en la parte posterior y una rueda en la parte delantera. Estos tipos de robots presentan de alguna manera un movimiento más libre el mismo que causa conflicto dependiendo del terreno por el cual circula el mismo como podría ser un terreno irregular se como se muestra en la Figura 4-1 (Ollero, 2001).

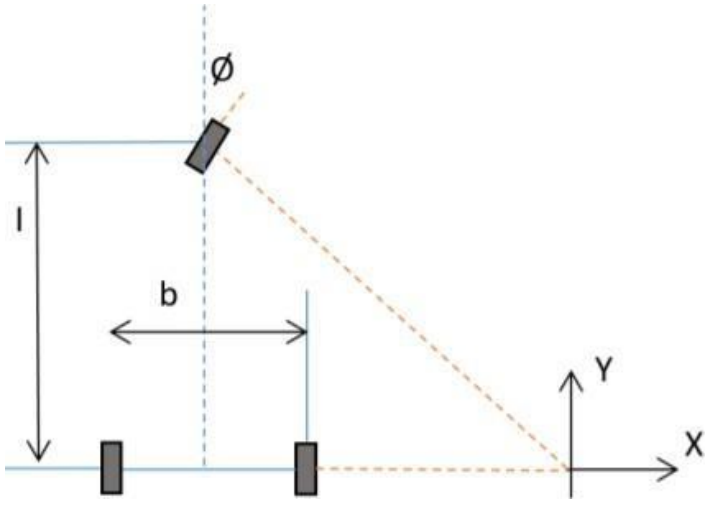


Figura 4-1: Locomoción de triciclo clásico

Fuente: (Ollero, 2001).

1.4.3. Direccionamiento diferencial

Cuentan con una diferencia de velocidad como se muestra en la Figura 5-1 y la tracción en cada una de las ruedas es su característica principal.

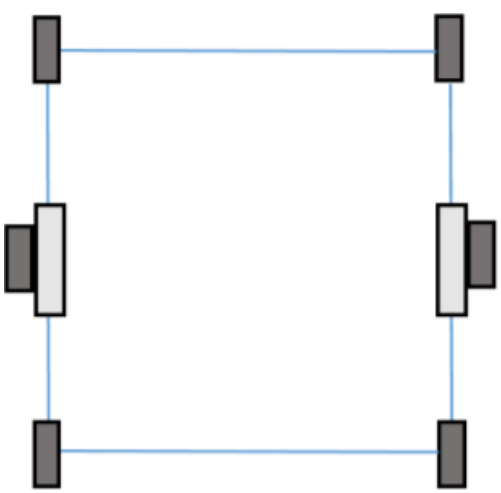


Figura 5-1: Locomoción con direccionamiento diferencial en dos ruedas laterales
Fuente: (Ollero, 2001).

1.4.4. Omnidireccional

Robots que pueden tener un desplazamiento omnidireccional el mismo que quiere decir q le permite dirigirse en cualquier dirección y sentido, en la Figura 6-1 se muestra una imagen de un robot omnidireccional.

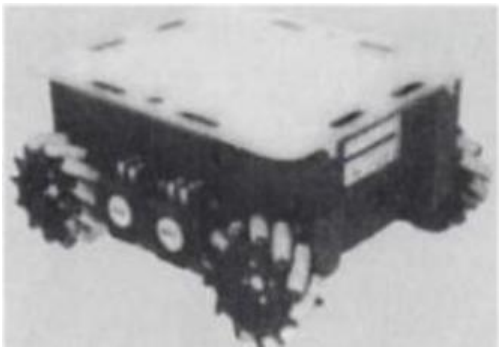


Figura 6-1: Robot Uramus con locomoción omnidireccional
Fuente: (Barrientos Peñin Balaguer Arcin, 996).

1.4.5. Tipos de articulaciones

Mediante un conjunto de secuencias a seguir las articulaciones ejecutan una orden de movimiento y posición de efector final dependiendo de la finalidad del robot. Se sigue un orden de acuerdo con el número de grados de libertad del mismo, como se muestra en la Figura 7-1.

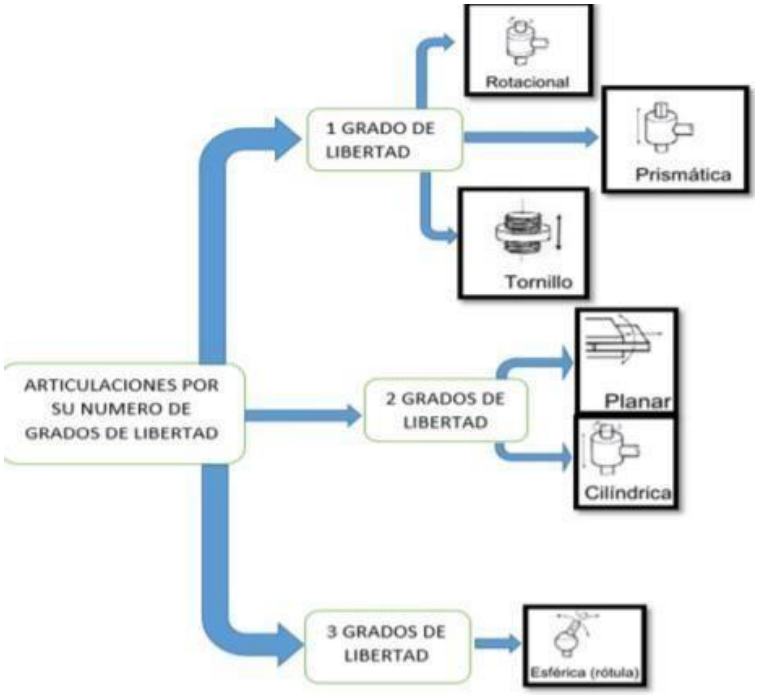


Figura 7-1: Articulaciones por su grado de libertad

Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

1.5. Cinemática

En función del tiempo la cinemática describe al robot con sus movimientos, descartando todas las fuerzas que actúan sobre los elementos del robot. Un ejemple de análisis cinemático se muestra en la Figura 8-1 de un robot articulado antropomórfico (Barrientos Peñin Balaguer Arcin 1996).

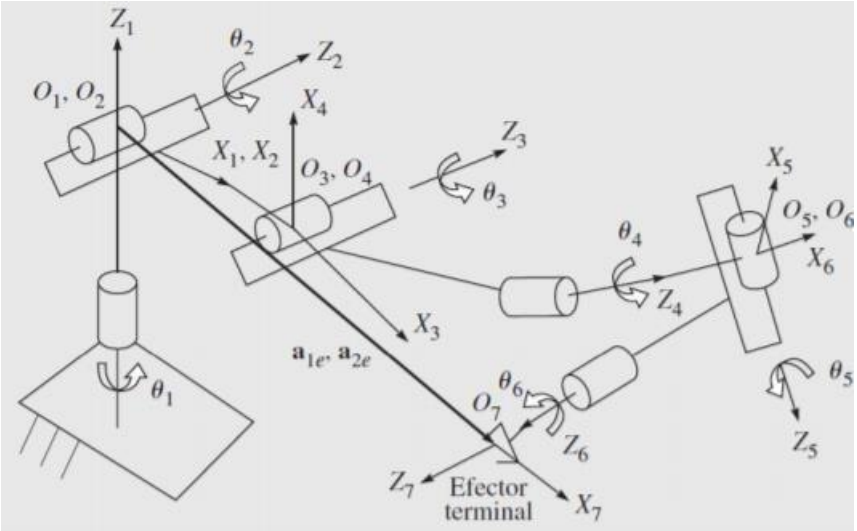


Figura 8-1: Análisis cinemático de un robot articulado antropomórfico
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

1.5.1. Cinemática Directa

Referenciando un sistema de coordenadas nos permite determinar la posición y dirección de las partes del robot (Barrientos Peñin Balaguer Arcin, 1996).

1.5.2. Cinemática inversa

Nos permite posicionar al robot correctamente al otorgar la cadena de movimiento de cada una de las articulaciones (Barrientos Peñin Balaguer Arcin, 1996).

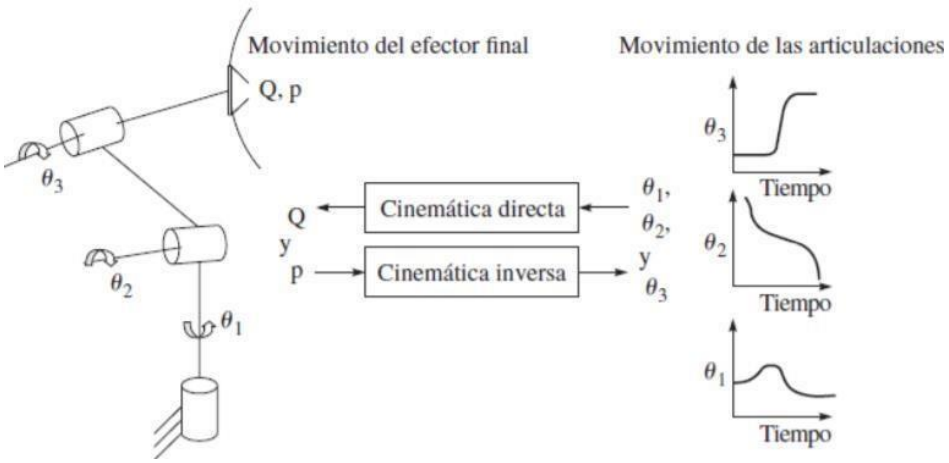


Figura 9-1: Cinemática directa e inversa
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

1.6. Dinámica

La dinámica es una ciencia que se encarga de estudiar las ecuaciones del movimiento como se muestra en la Figura 10-1, da a conocer las ecuaciones necesarias en el sistema describiendo las fuerzas utilizadas en cada articulación producida por los actuadores, demostrando una estrategia de control (Craig, 1977).

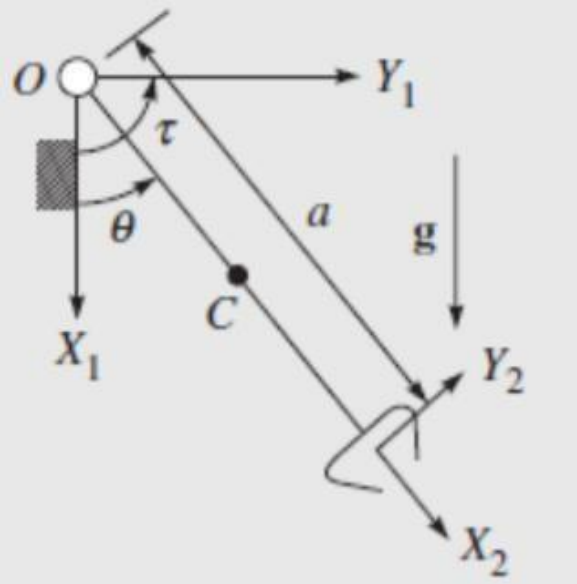


Figura 10-1: Dinámica de un brazo de un eslabón
Fuente: (Craig, 1977).

Para determinar la dinámica de un sistema a continuación se detallan los dos métodos más importantes:

-Las ecuaciones de movimiento de Lagrange, limitado a eliminar las fuerzas de restricción de las ecuaciones obteniendo un sistema más sencillo.

-Las leyes de Newton y Euler método de movimientos lineales y rotacionales necesarias para el diseño de cada eslabón de acuerdo a las fuerzas aplicadas, así como las fuerzas de restricción del sistema.

1.6.1. Dinámica inversa

La dinámica inversa determina las partes de torsión de las fuerzas de los actuadores.

1.6.2. Dinámica directa

La dinámica directa determina algunos pares de torsión o fuerzas de la articulación (Craig, 1977).

1.7. Motores

Los motores contienen dos partes que son: parte eléctrica alimentado por una fuente de corriente continua y una parte de potencia. El voltaje de salida del amplificador de potencia controla la velocidad de giro mediante un par generado por la caja de reducción que algunos motores tienen incorporado. El diagrama de bloques de un motor de corriente continua se muestra en la Figura 11-1 (Barrientos Peñin Balaguer Arcin, 1996).

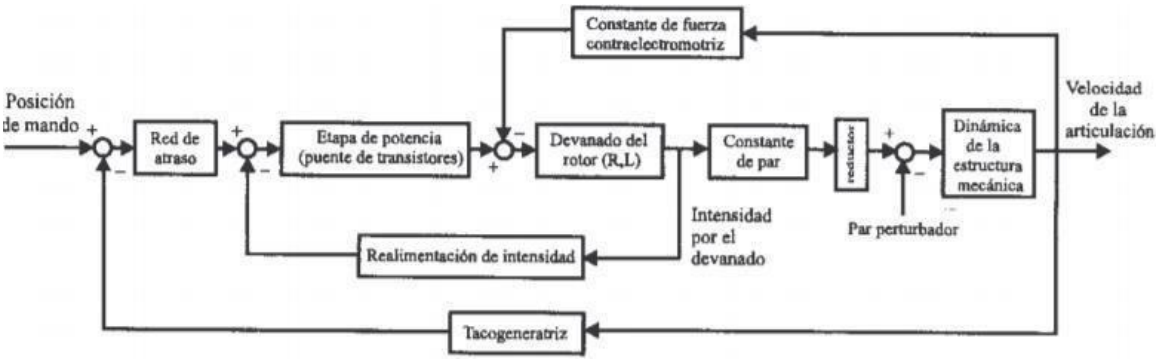


Figura 11-1: Esquema de bloques del modelo de motor Eléctrico de CC
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

A los motores se los puede clasificar de la siguiente manera:

- Motores universales.
- Motores de corriente continua.
- Motores de corriente alterna.

1.7.1. Motores universales

Los motores universales generalmente alimentados por corriente alternan monofásica. Con una apariencia física a las máquinas de corriente continua, dispone de un colector de delgas y el inducido está en el rotor como se muestra en la Figura 12-1. Estos motores son empleados en la mayoría de electrodomésticos en su versión de pequeña potencia y en su versión de potencia más alta son utilizados en tracción eléctrica. Los motores universales pueden acoplarse a un funcionamiento tanto con corriente continua, así como con corriente alterna de donde se otorga su nombre de universales (Mora, s.f.).

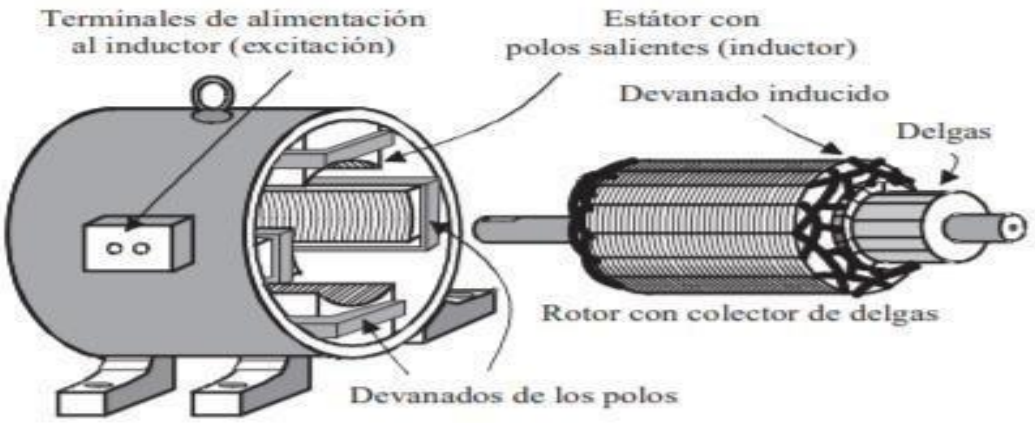


Figura 12-1: Motor Universal
Fuente: (Mora, s.f.).

1.7.2. Motores de Corriente Continua

Un motor de corriente continua nos entrega energía mecánica en su salida a base de energía eléctrica, según Faraday y Lenz en el principio de reciprocidad electromagnética consiste en un dínamo que trabaja en un régimen inverso.

Estos motores de corriente continua se pueden clasificar según su tipo de excitación y son: motores con excitación independiente, derivación, serie y compuesta (Mora, s.f.).

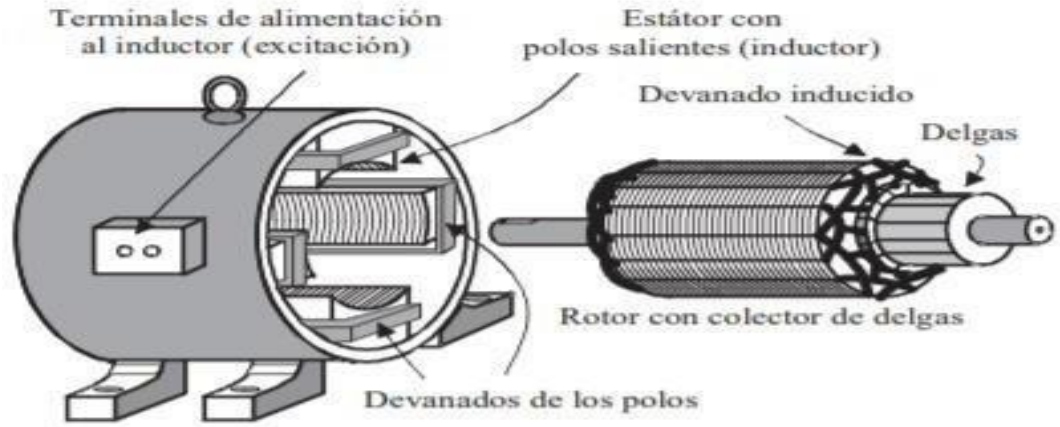


Figura 13-1: Motor de corriente continua
Fuente: (Mora, s.f.)

Motores de corriente alterna

Los motores de corriente alternan donde el campo giratorio del estator induce fuerzas electromotrices en el devanado del rotor, al estar en corto circuito a través de un reóstato de arranque aparecen corrientes que reaccionan con el campo giratorio del estator mueven la maquina a una velocidad muy cercana a la del sincronismo (Mora, s.f.).

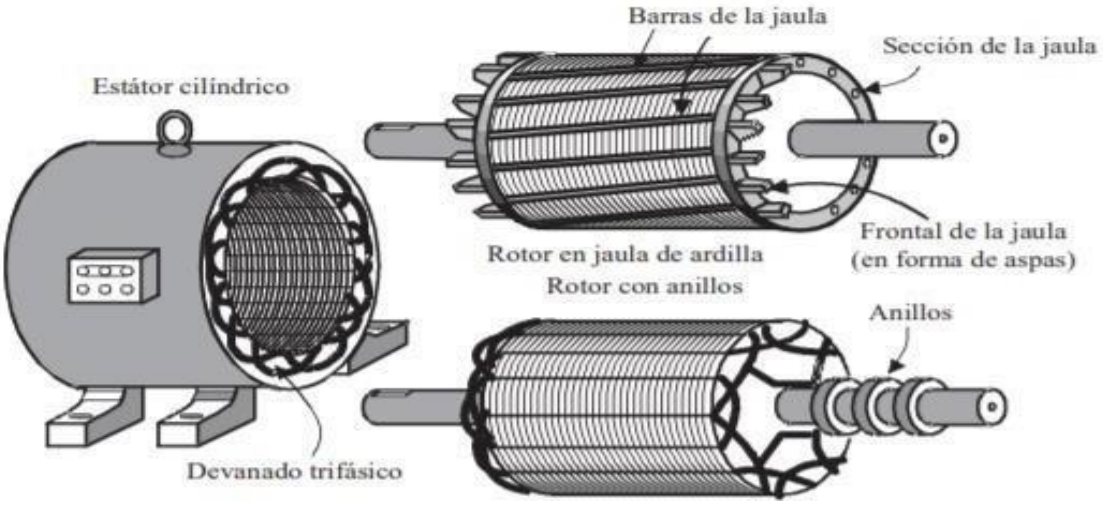


Figura 14-1: Motor de corriente alterna
Fuente: (Mora, s.f.).

1.8. Controladores

Son plataformas de desarrollo basadas en lenguajes de programación para poder ejecutar una serie de comandos o líneas programación con la finalidad de realizar una tarea específica. Estas tarjetas pueden ser de código abierto ayudando a la programación electrónica capaz de enviar y recibir información hacia la mayoría de los dispositivos conectados en la misma, incluso valerse de Internet para controlar un dispositivo electrónico específico. Las tarjetas de desarrollo tienen la capacidad de leer información de diferentes tipos de dispositivos de entradas como, por ejemplo: sensores, antenas, etc. Así como también dispositivos de salida como son leds, parlantes, motores entre otros (Mora, s.f.).

1.9. Sensores

Los sensores están relacionados directamente con la definición de transductor ya que este es el que nos permite transformar o convertir una energía de entrada en una especificada salida. La diferencia presentada entre transductor y sensor se basa en que el sensor cambia el dominio de la variable física medida además la salida será un dato útil para el sistema de medida.

Las principales clasificaciones de los sensores son: principio de transducción utilizado y por la variable a medir (Corona, Abarca y Mares, 2014).

1.9.1. Clasificación de los sensores por el principio de transducción

El tipo de transductor que utilice es una forma de clasificar a los sensores, pero se puede decir que no es clara ya que no ofrece una visión clara del tipo de variable física a medir (Corona, Abarca y Mares, 2014).

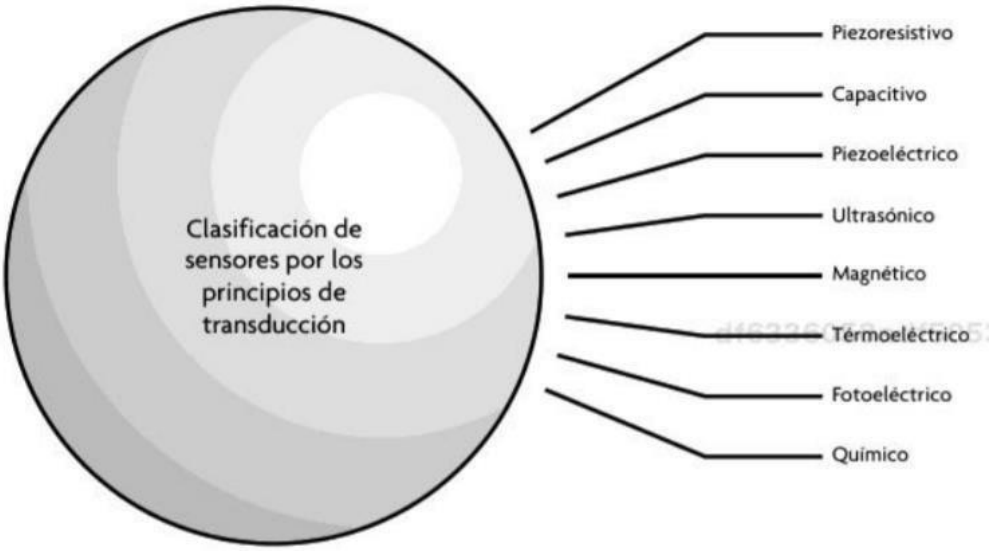


Figura 15-1: Clasificación principio de transducción
Fuente: (Corona, Abarca y Mares, 2014).

1.9.2. Clasificación de los sensores por el tipo de variable medida

La clasificación más utilizada para sensores es según la variable a determinar o calcular. Este tipo de clasificación causa una cierta incertidumbre ya que un mismo sensor puede medir una o varias variables a la vez (Corona, Abarca y Mares, 2014).

13dd00258e6f84c	De posición, velocidad y aceleración
	De nivel y proximidad
	De humedad y temperatura
	De fuerza y deformación
Clasificación de los sensores según la variable física a medir	De flujo y presión
	De color, luz y visión
	De gas y pH
	Biométricos
	De corriente

Figura 16-1: Clasificación Según la variable física
Fuente: (Corona, Abarca y Mares, 2014).

1.10. Actuadores

A partir de una transformación de energía es un dispositivo que genera una fuerza que ejerce un cierto cambio en la posición, velocidad o estado requeridos en algún proceso o sistema eléctrico y electrónico. Se clasifican en dos grupos:
Por el tipo de energía utilizada: actuador hidráulico, eléctrico y neumático.
Por el tipo de movimiento que generan: rotatorio y lineal (Corona, Abarca y Mares 2014).

1.11. Ubicación espacial

El movimiento espacial del extremo del robot es producido por la manipulación de piezas. Para realizar un trabajo o acción determinada es necesario saber la posición y orientación de la base del robot. (Barrientos Peñin Balaguer Arcin, 1996)

1.11.1. Representación de la posición

En un plano de posicionamiento depende de los grados de libertad para definir el número de componentes que se pueden emplear (Barrientos Peñin Balaguer Arcin, 1996).

Coordenadas Cartesianas

Trabaja en un plano, expresado por las coordenadas (x,y) en un eje de coordenadas OXY. Un punto a tiene un vector asociado **p**(x,y) el mismo que va desde el origen 0 hasta el punto a. demostrado en la Figura 16-1 y en el caso de trabajar en tres dimensiones el vector p está determinado por las componentes cartesianas (x,y,z) (Barrientos Peñin Balaguer Arcin, 1996).

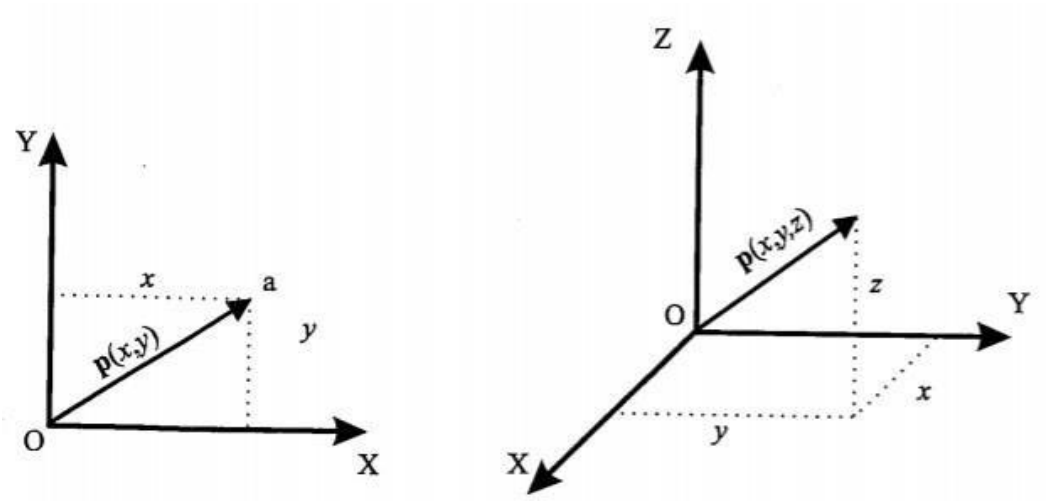


Figura 17-1: Representación de un vector en coordenadas cartesianas de 2 y 3 dimensiones
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

Coordenadas cilíndricas y polares

En un plano, se puede definir la localización de un vector **p** en un sistema de ejes cartesianos OXY, utilizando coordenadas polares **p(r, Θ)** como se muestran en la Figura 17-1. Representando como r la distancia del origen 0 del sistema hasta el extremo del vector **p** y el ángulo **Θ** que forma el vector con el eje OX. Y al trabajar en tres dimensiones OXYZ o coordenadas cilíndricas p(r, **Θ,** z) (Barrientos Peñin Balaguer Arcin, 1996).

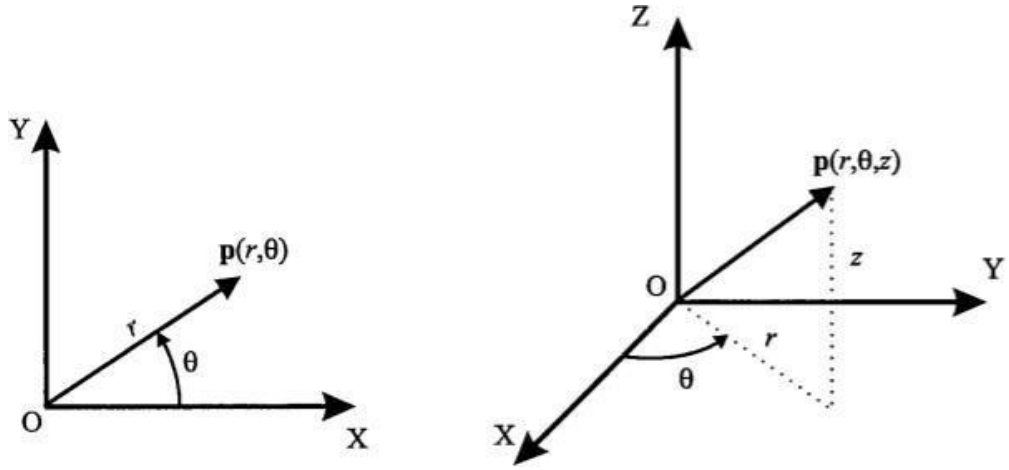


Figura 18-1: Representación de un vector en coordenadas polares y cilíndricas
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

Coordenadas esféricas

En un sistema de referencia OXYZ el vector **p** con coordenadas esféricas p(r, **Θ, φ**), r es la distancia del vector hasta el origen, **Θ** es el ángulo formado por el eje ox y el ángulo **φ** es el formado por el eje OZ, como se muestra en la Figura 17-1 (Barrientos Peñin Balaguer Arcin, 1996).

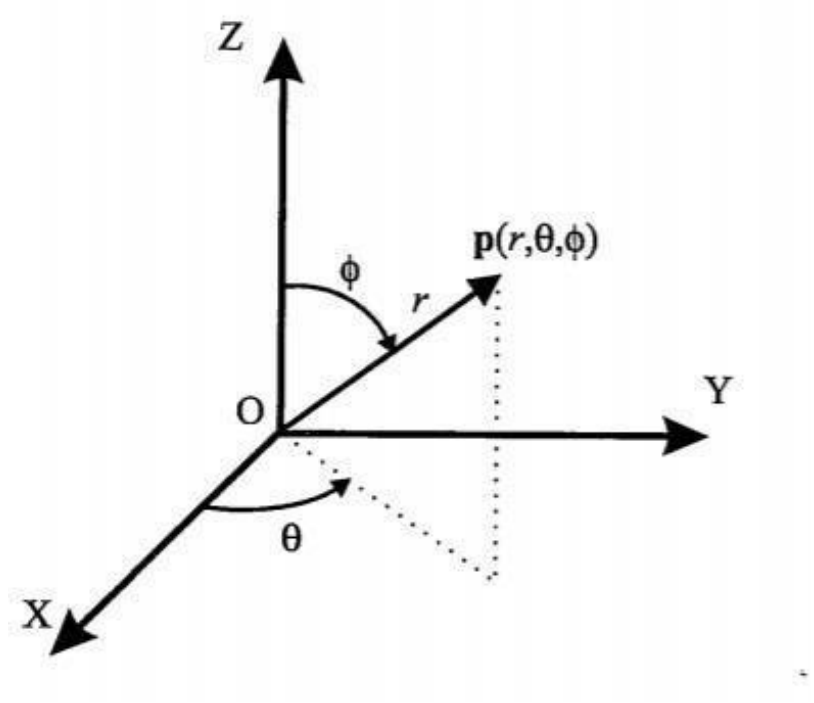


Figura 19-1: Representación de un vector en coordenadas esféricas
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

Representación de la orientación

Se describe la orientación de un objeto respecto a un sistema de referencia, se le asigna un nuevo sistema de referencia al objeto para finalizar estudiando la relación existe entre los dos sistemas presentados; suponiendo que los dos sistemas tienen un punto de origen común (Barrientos Peñin Balaguer Arcin, 1996).

Ángulos de Euler

Definidos por tres ángulos ψ, ϕ, θ girando en el sistema OXYZ, Existen 24 posibilidades siendo las tres posibilidades más optimas las que se muestran a continuación.

Ángulos de Euler ZXZ

Donde más común se realizan los giros. Se inicia en el sistema OXYZ y OUVW como se muestra en la figura 18-1, primero se de girar OUVW un ángulo de ϕ respecto al eje OZ transformándose en OU'V'W', después girar este sistema un ángulo θ con el eje OU' convirtiéndose así en OU''V''W'' y este último sistema al girarlo un ángulo ψ finalmente se obtiene el sistema OU'''V'''W''' (Barrientos Peñin Balaguer Arcin, 1996).

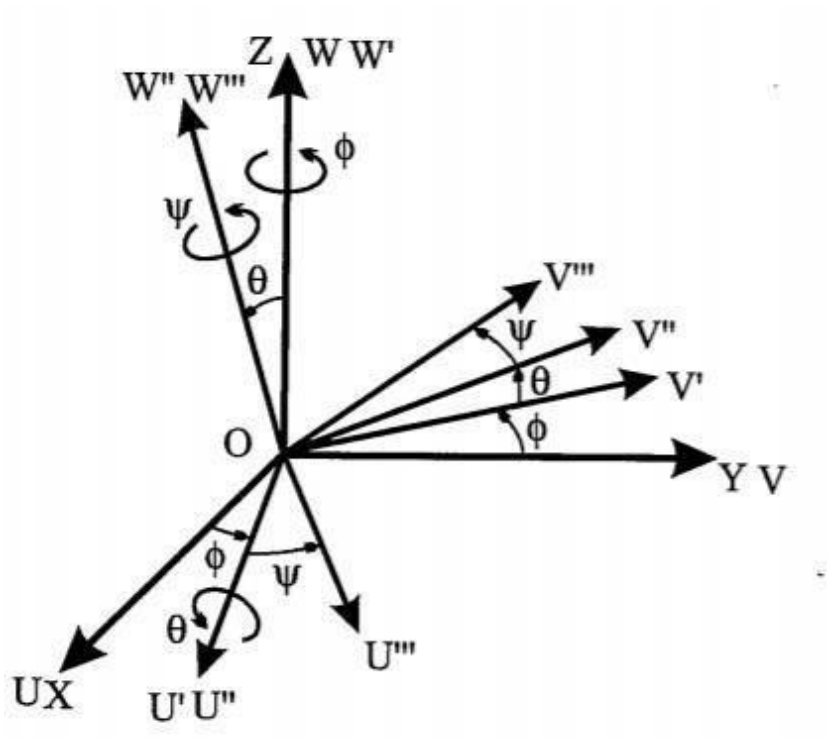


Figura 20-1: Ángulos de Euler ZXZ
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

Ángulos de Euler ZYZ

Se inicia en el sistema OXYZ y OUVW como se muestra en la figura 19-1, se siguen los siguientes pasos: primero se de girar OUVW un ángulo de ϕ respecto al eje OZ transformándose en OU'V'W', después girar este sistema un ángulo θ con el eje OV' convirtiéndose así en OU''V''W'' y este último sistema al girarlo un ángulo ψ con el eje OW'' finalmente se obtiene el sistema OU'''V'''W''' (Barrientos Peñin Balaguer Arcin ,1996).

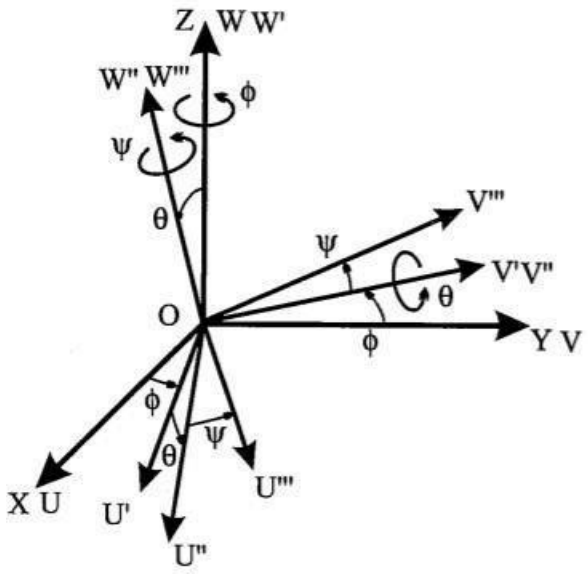


Figura 21-1: Ángulos de euler ZYZ
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

Roll, pitch and yaw

Se inicia en el sistema OXYZ y OUVW se puede realizar de igual manera al sistema en cualquier orientación siguiendo estos pasos: girar el sistema OUVW un ángulo ψ respecto de OX nombrado *Yaw*, girar OUVW un ángulo θ respecto de OY nombrado *Pitch* y girar OUVW un ángulo ϕ respecto de OZ nombrado Roll, como se muestra en la Figura 19-1 (Barrientos Peñin Balaguer Arcin, 1996)

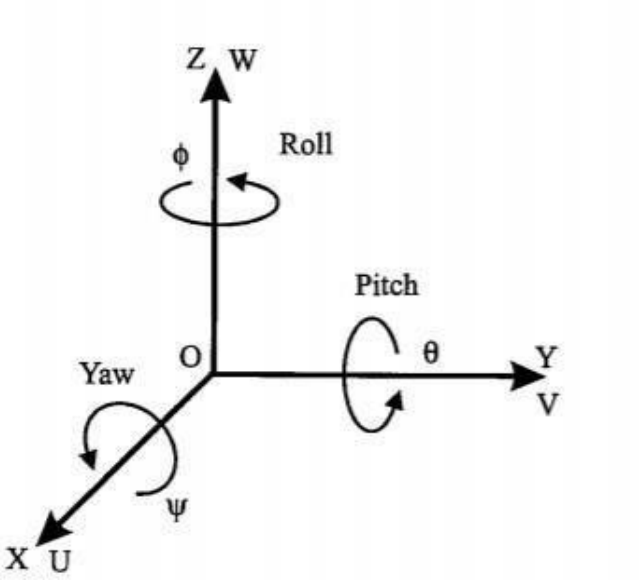


Figura 22-1: Roll, pitch and yaw
Fuente: (Barrientos Peñin Balaguer Arcin, 1996).

Cuaternios

Un cuaternio Q formado por 4 componentes (*q*0, *q1, q2, q3*) de coordenadas en una base (e, i, j, k) Es frecuente definir parte escalar del cuaternio a la componente en e: *q*0, y parte vectorial al resto de los componentes. Para la utilización de los cuaternios como metodología de representación de orientaciones se asocia el giro de un ángulo θ sobre el vector **definido** por:

$$Q = R_{ot}\,(k, \theta) = (\cos\frac{\theta}{2}, k\,\sin\frac{\theta}{2})$$

1.12. Matlab

Es un software interactivo de análisis, teniendo una evolución constante en cada una de las versiones mediante sus actualizaciones. Este programa cuenta con varios paquetes de herramientas como calculo matricial, modelado, cálculo matemático, simulación, análisis y procesamiento de varios datos. Este software es utilizado para la realización de varios proyectos de investigación y desarrollo de prototipos, compuesto por herramientas, funciones y un conjunto de librerías como se muestra en la Figura 21-1(MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink, s.f.).

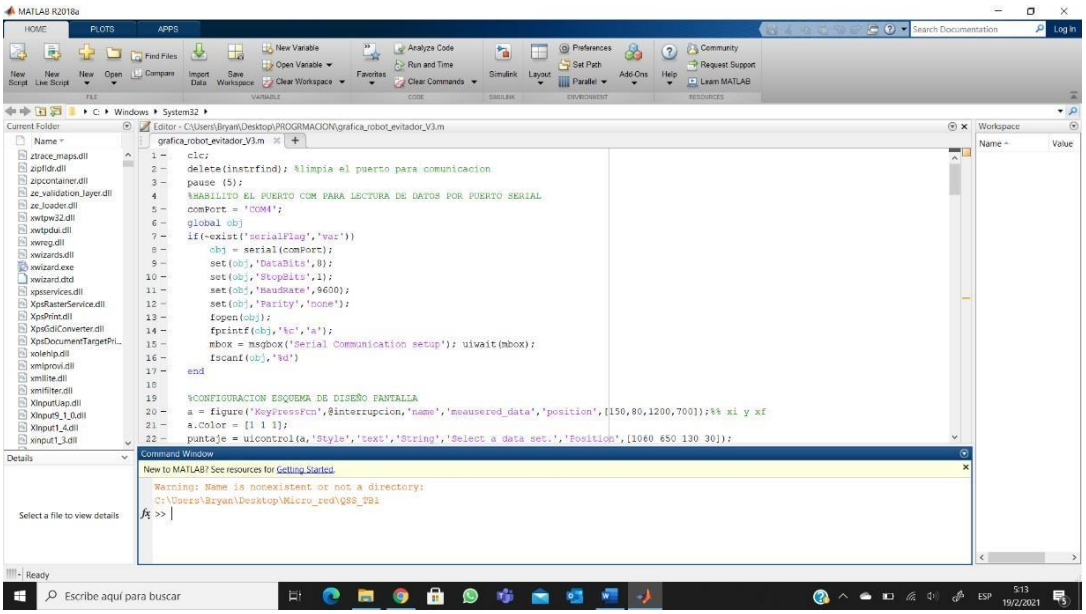


Figura 23-1: Interfaz de Matlab
Fuente: (MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink, sf).

1.13. Tipos de comunicación

Proceso por el cual se permite transmitir y recibir información.

1.13.1. Comunicación alámbrica

Generalmente utilizado por medio de cables o fibra óptica, conectados a cada uno de los dispositivos integrados a una red o sistema con un fin común. Necesita un soporte técnico de manera presencial por todos los materiales y herramientas que lo componen (Felipe et al., 2008).

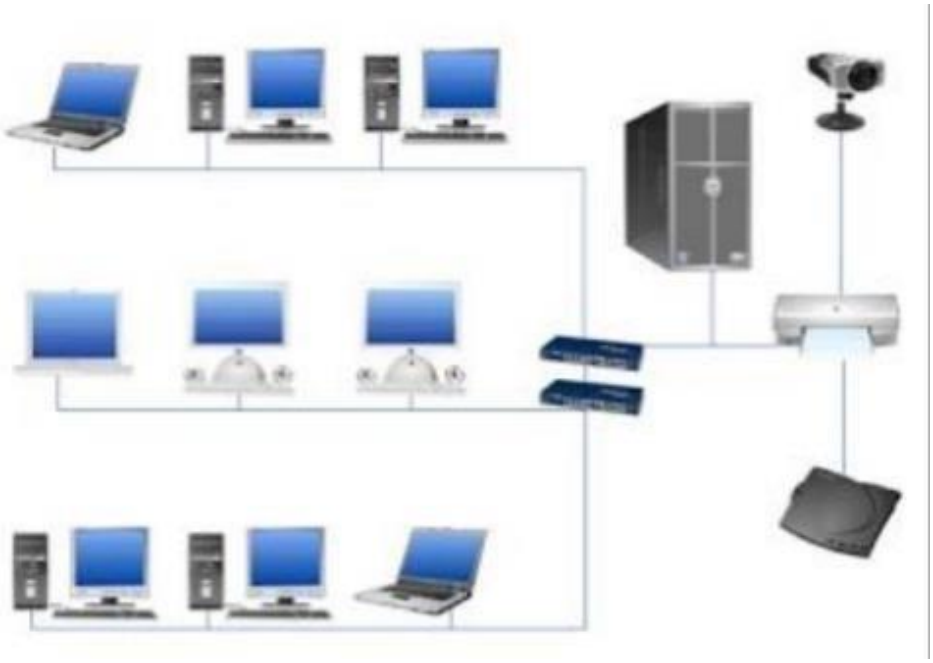


Figura 24-1: Comunicación alámbrica
Fuente: (Felipe et al., 2008).

1.13.2. Comunicación inalámbrica

El sistema de comunicación inalámbrica no necesita un soporte técnico de manera presencial. La información o datos transmitidos por medio de ondas para ampliar este tipo de señales es necesario ubicar repetidores en lugares específicos permitiendo así abarcar cada uno de los elementos o componentes de un sistema como se presenta en la Figura 23-1 (Felipe et al., 2008).

Figura 25-1: Comunicación alámbrica
Fuente: (Felipe et al., 2008).

CAPÍTULO II

2. MARCO METODOLÓGICO

Este capítulo especifica el diseño de *hardware* y *software* del sistema, definiendo cada etapa en su construcción; así como los componentes con sus características técnicas y el esquema electrónico implementado en el sistema de ubicación espacial de plataforma móvil mediante sensores en un mapa determinado.

2.1. Requerimientos para el diseño del prototipo de sistema de ubicación espacial

La información obtenida en el capítulo anterior es necesaria para plantear los requerimientos de diseño para realizar el prototipo de sistema de ubicación espacial de plataforma móvil mediante sensores en un mapa determinado, estos son:

- Evitar retardos proporcionando al sistema la capacidad de procesamiento adecuada.
- Determinar la ubicación espacial de la plataforma.
- Limitar las colisiones de la plataforma con el mapa determinado.
- Establecer la comunicación entre el ordenador y la plataforma.

2.2. Concepción del diseño del sistema de ubicación espacial

La Figura 1-2, demuestra el funcionamiento, así como la estructura de la plataforma con su conexión. Se aprecia la plataforma la cual se trasladará por una ruta dentro del mapa determinado evitando las colisiones y mediante un cable conectado al ordenador se dibuja su ruta simultáneamente. La plataforma está integrada por un sistema electrónico basado en dos motores, un micro controlador que almacena y ejecuta el programa, activando o desactivando los motores de acuerdo con la ruta, los sensores cuya función es determinar la presencia de obstáculos, evitándolos.

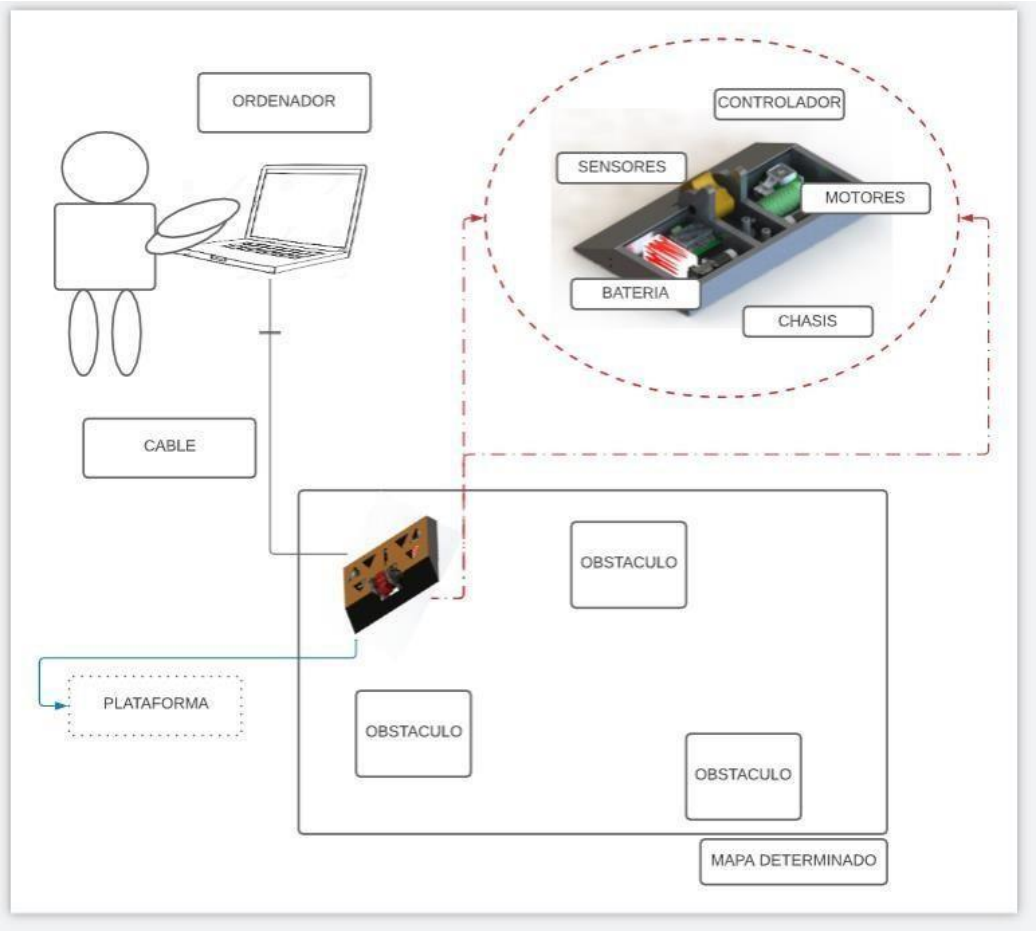


Figura 1-2: Concepción General del sistema de ubicación espacial
Realizado por: Núñez, Bryan, 2021.

2.3. Diseño de la estructura para la plataforma del sistema de ubicación espacial

Para el diseño de la estructura de la plataforma se utilizó un bloque de aluminio 7075 el mismo que es un metal óptimo para realizar el mecanizado, así como para soportar las colisiones evitando así daños en la parte interna de la plataforma donde se encuentran ubicados rodos los dispositivos electrónicos. En la Figura 2-2 se muestra el bloque de aluminio con dimensiones 13x49x28.5 cm.

Figura 2-2: Bloque de aluminio 7075
Realizado por: Núñez, Bryan, 2021.

2.3.1. Modelado de la estructura interna de la plataforma

Mediante el Software *SolidWorks* se procedió a modelar cada una de las piezas a utilizar en la parte electrónica, la misma es interna en el robot como se muestra en la Figura 3-2, permitiéndonos ubicar los materiales de manera adecuada a las dimensiones registradas por el bloque de aluminio.

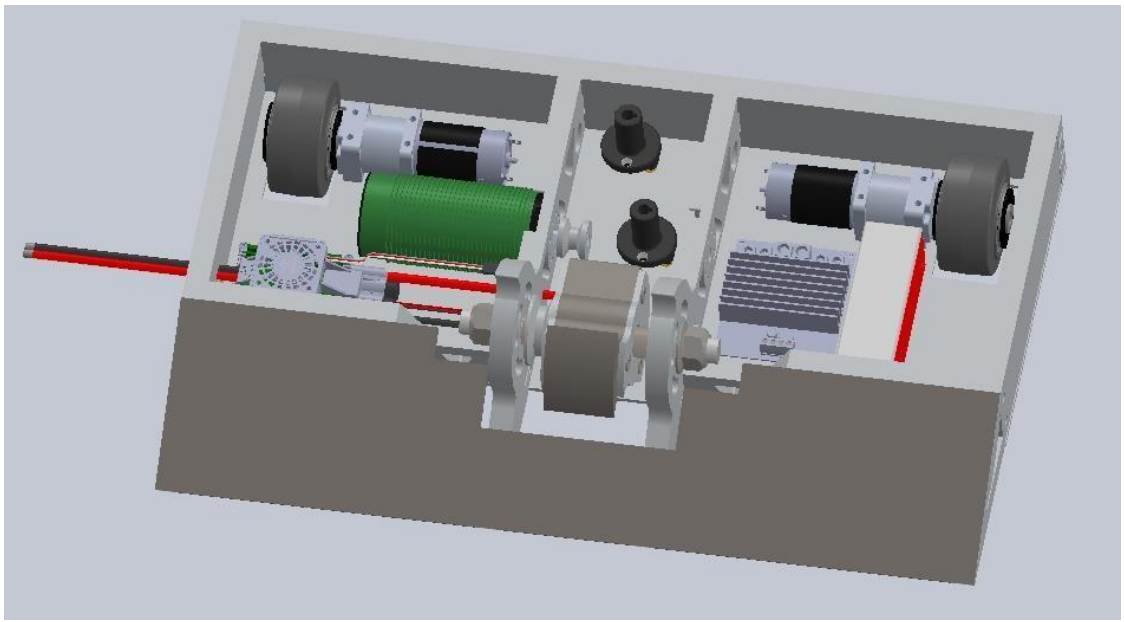


Figura 3-2: Modelo interno plataforma
Realizado por: Núñez, Bryan, 2021.

2.3.2. Modelado de la estructura externa de la plataforma

Utilizando el software *SolidWorks* se modelo una estructura externa que resguarde los materiales electrónicos en su interior convirtiéndolo en una plataforma óptima para receptar datos, limitando los daños interiores. Como se muestras en la Figura 4-2, Figura 5-2, Figura 6-2 y Figura 7-2.

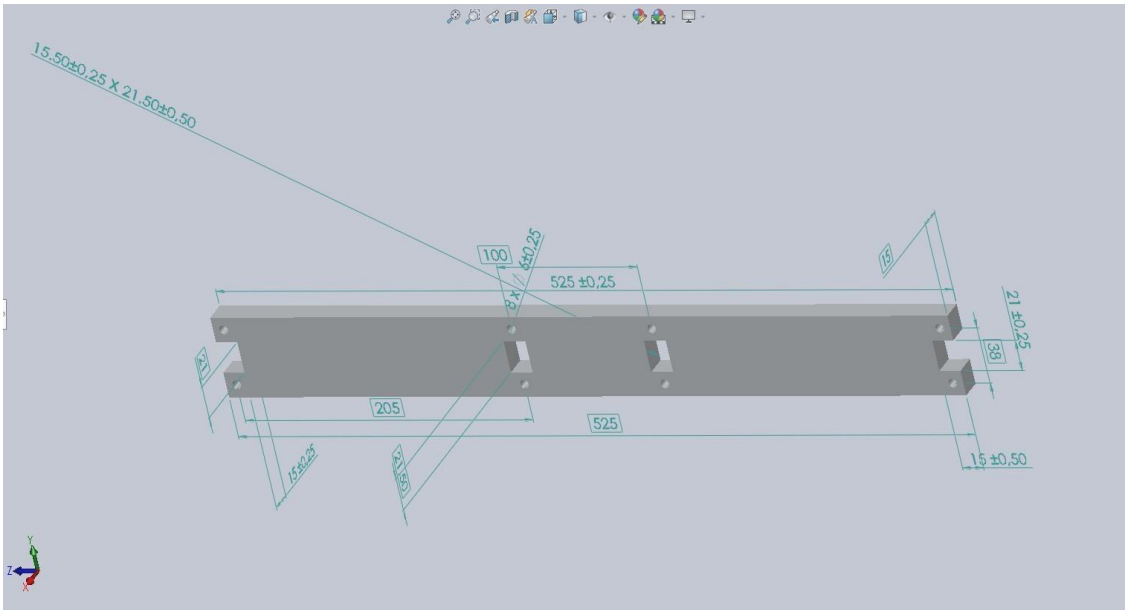


Figura 4-2: Modelo cara de atrás
Realizado por: Núñez, Bryan, 2021.

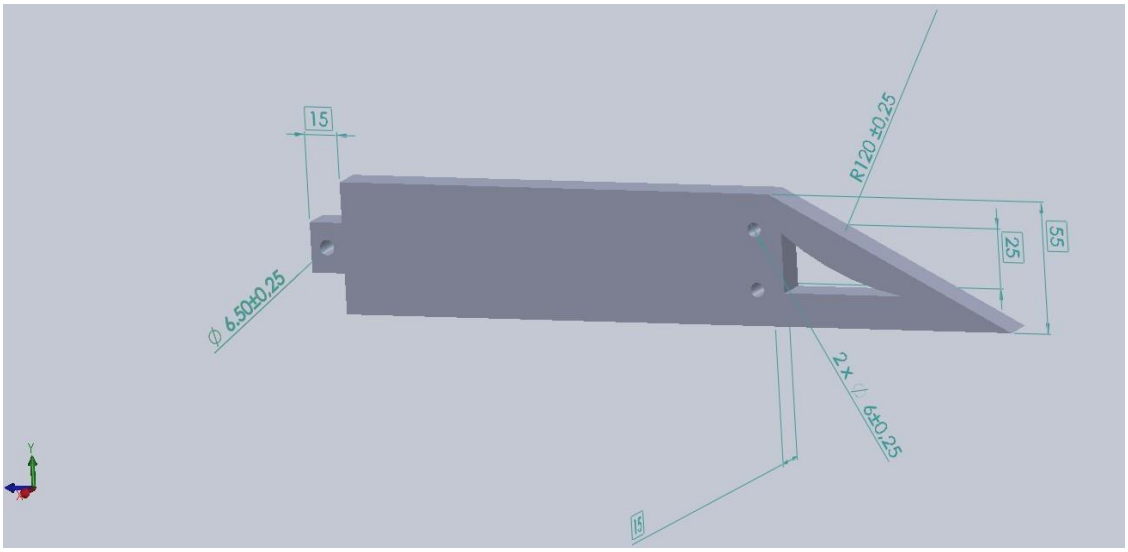


Figura 5-2: Modelo cara de lateral
Realizado por: Núñez, Bryan, 2021.

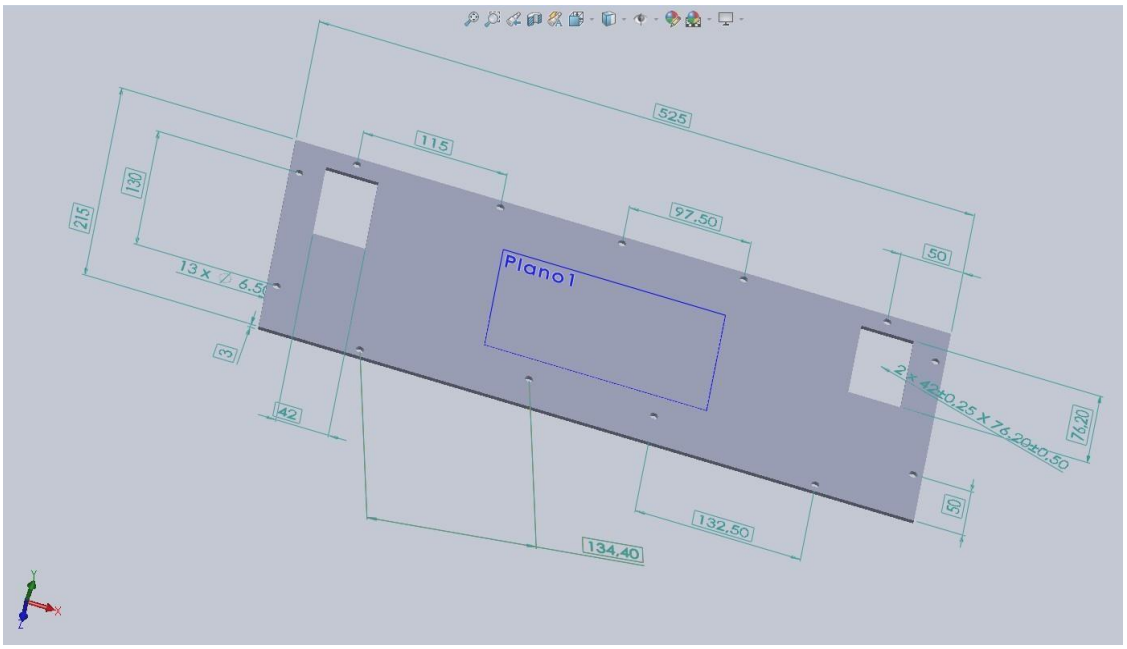


Figura 6-2: Modelo tapa inferior
Realizado por: Núñez, Bryan, 2021.

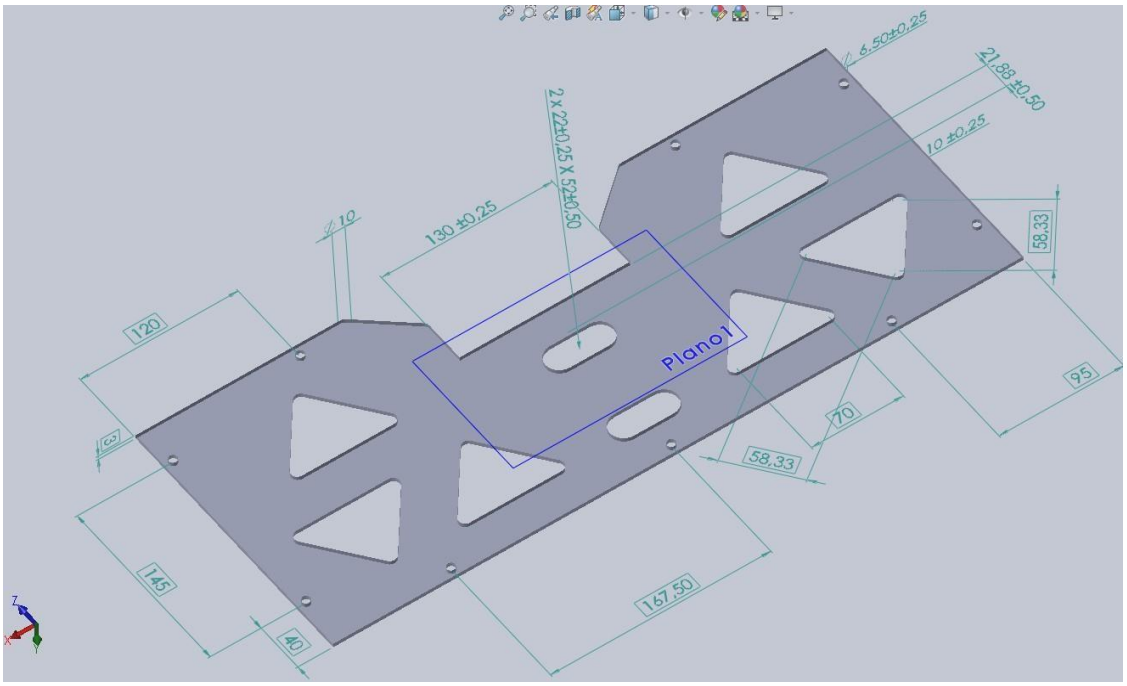


Figura 7-2: Modelo tapa superior
Realizado por: Núñez, Bryan, 2021.

2.4. Mecanizado de la estructura del robot

El mecanizado de la plataforma consta de un proceso principal que es el corte por chorro de agua para cada una de las tapas o caras exteriores como se muestra en la Figura 8-2. Un proceso en

torno en el cual se construyeron los ejes y acoples como se muestra en la Figura 9-2. En la Figura 10-2 se muestras la estructura completa de la plataforma.

Figura 8-2: Corte por chorro de agua
Realizado por: Núñez, Bryan, 2021.

Figura 9-2: Corte en torno
Realizado por: Núñez, Bryan, 2021.

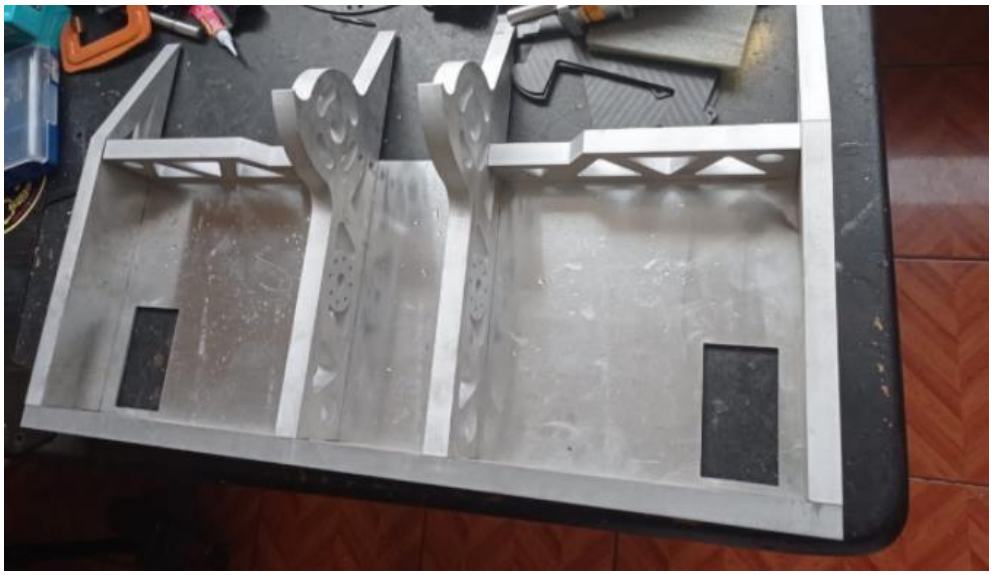


Figura 10-2: Estructura de la plataforma
Realizado por: Núñez, Bryan, 2021.

2.5. Construcción del mapa determinado

Para la elaboración del mapa determinado se tomó en cuenta las dimensiones del robot, así como las velocidades de los motores, como se muestra en la Figura 11-2.

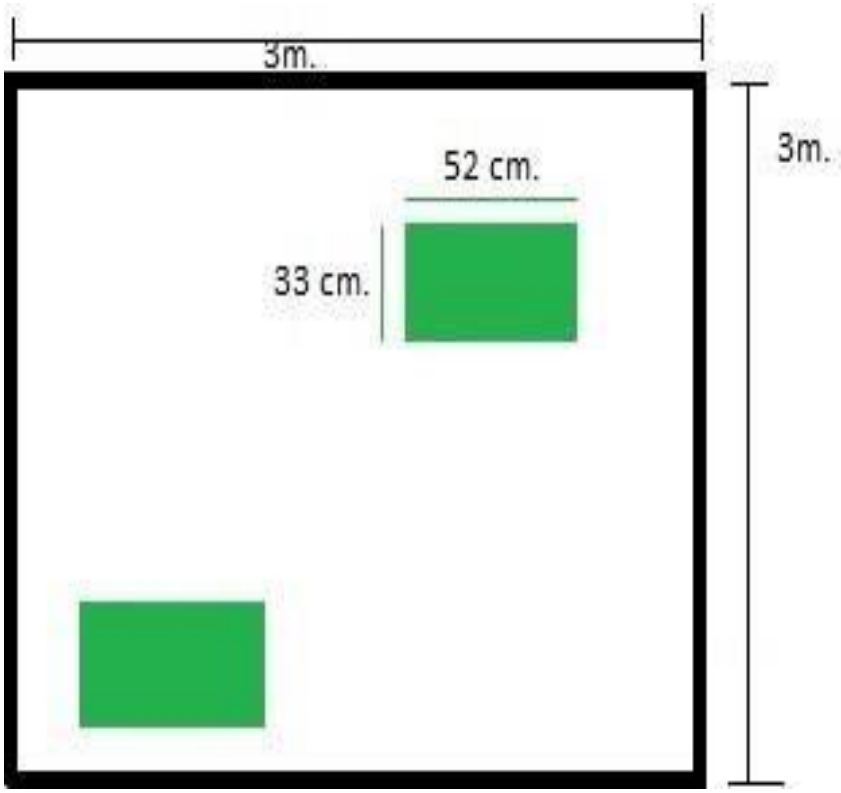


Figura 11-2: Estructura del mapa determinado
Realizado por: Núñez, Bryan, 2021.

2.6. Diseño de bloques del Sistema de ubicación espacial

Al realizar la concepción general del sistema de ubicación espacial demostrado en la Figura 1-2, se concluye que consta de 3 bloques definidos como: alimentación, bloque de control y bloque de interfaz mostrados en la Figura 11-2, describiendo cada uno en los siguientes apartados.

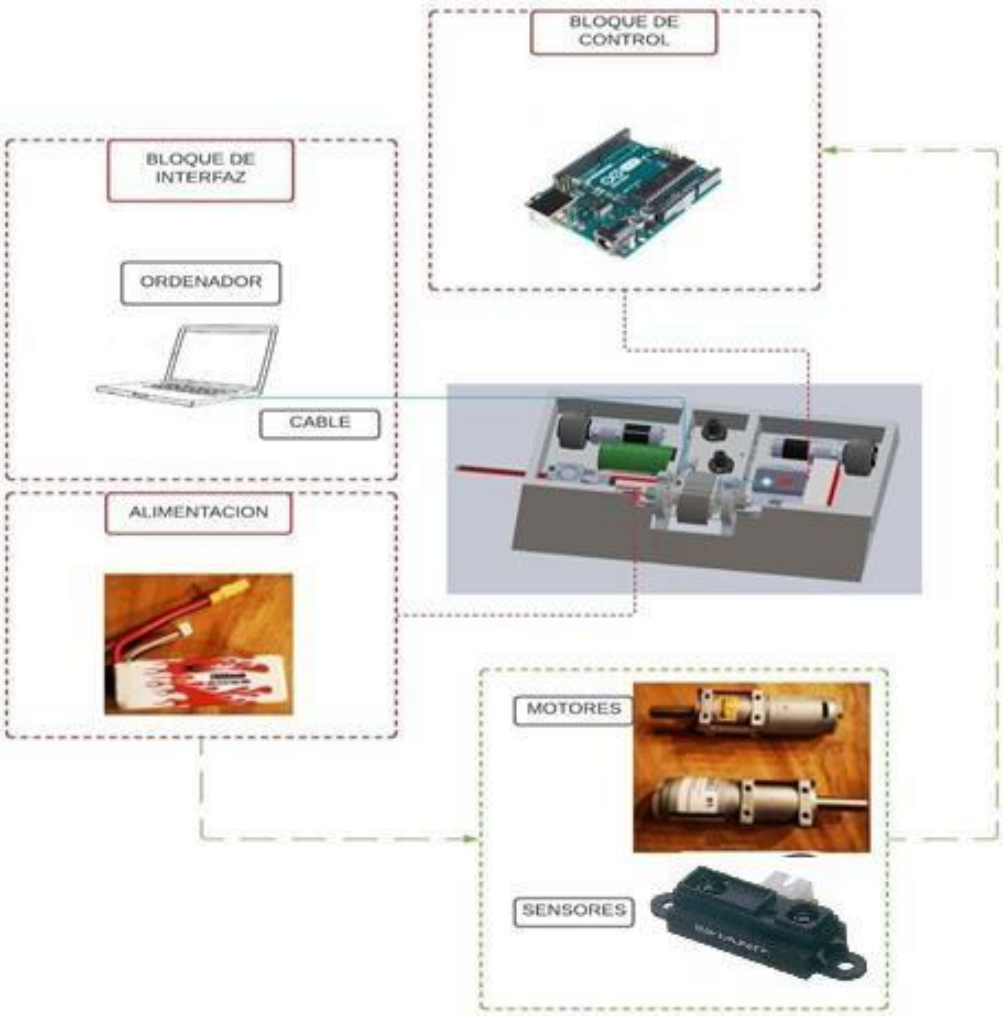


Figura 12-2: Diseño por bloques del sistema de ubicación espacial
Realizado por: Núñez, Bryan, 2021.

2.6.1. Bloque de alimentación

Partiendo del bloque de poder o energía donde se encuentra una batería de polímero de litio la misma que reparte carga eléctrica para los actuadores de la planta como los motores y sensores. Ubicado en la parte interior de la plataforma.

2.6.2. Bloque de control

Constituido por un microcontrolador reprogramable que mantiene conexión directa con el ordenador transmitiendo simultáneamente la ruta de la plataforma, recibiendo datos constantes de los sensores para evitar colisiones.

2.6.3. Bloque de interfaz

Basado en la ejecución de un programa elaborado en lenguaje *Matlab* que leerá los movimientos del robot proyectándolos simultáneamente en el ordenador, el mismo se encargará de diferenciar los obstáculos presenten evitando las colisiones.

2.7. Descripción de los elementos *hardware* del sistema de ubicación espacial

En este apartado se describen los componentes de *hardware* que forman parte del sistema con sus características más importantes. En los anexos A, B y C se adjuntan las respectivas hojas de datos.

2.7.1. Arduino uno

Es una tarjeta de desarrollo reprogramable basado en el chip ATmega328p como se muestra en la figura 12-2, utilizando el lenguaje C++, pero Arduino ofrece un *core* facilitando la programación, como como algunas otras librerías para operaciones específicas. Consta de: *USB plug*, fuente de alimentación externa, botón de *reset*, microcontrolador, pines analógicos, pines digitales, tierra digital y analógica, pines de alimentación, y un programador de circuito, este es otra fuente para cargar su programa (Badamasi, 2014). En la tabla 1-2 se muestran sus características técnicas principales.

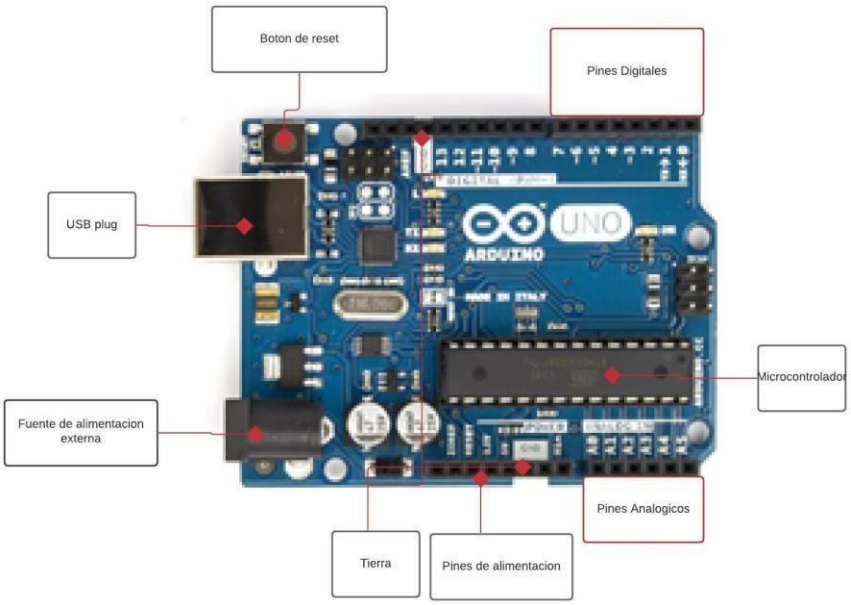


Figura 13-2: Arduino uno con sus partes
Realizado por: Núñez, Bryan, 2021.

Tabla 1-2: Características del ARDUINO UNO

DESCRIPCIÓN	CARACTERISTICA
Versión	R3
Voltaje de funcionamiento	5 V.
Corriente continua por pin	20 mA.
Corriente CC para 3.3V. por pin	50 mA.
Velocidad del reloj	16 Mhz
Precio	$ 9,50

Fuente: Tabla de datos Arduino uno
Realizado por: Núñez, Bryan, 2021.

2.7.2. Motor PDX26 – 26:1 GEARMOTOR

Es un motor empleado para funcionar con peso ya que es de alta potencia, este motor combina una caja robusta de cambios *banebots* 26:1 y está compuesto por un motor *duraTrax* 550. Se muestra en la Figura 14-2 las dimensiones del servomotor las mismas que van a ser útiles para la construcción de la plataforma y en la tabla 2-2 se muestran las características del servomotor. (MG995, 2018). En la figura 15-2 se muestra el motor (Sabertooth 2X60 regenerative dual motor driver, s.f.).

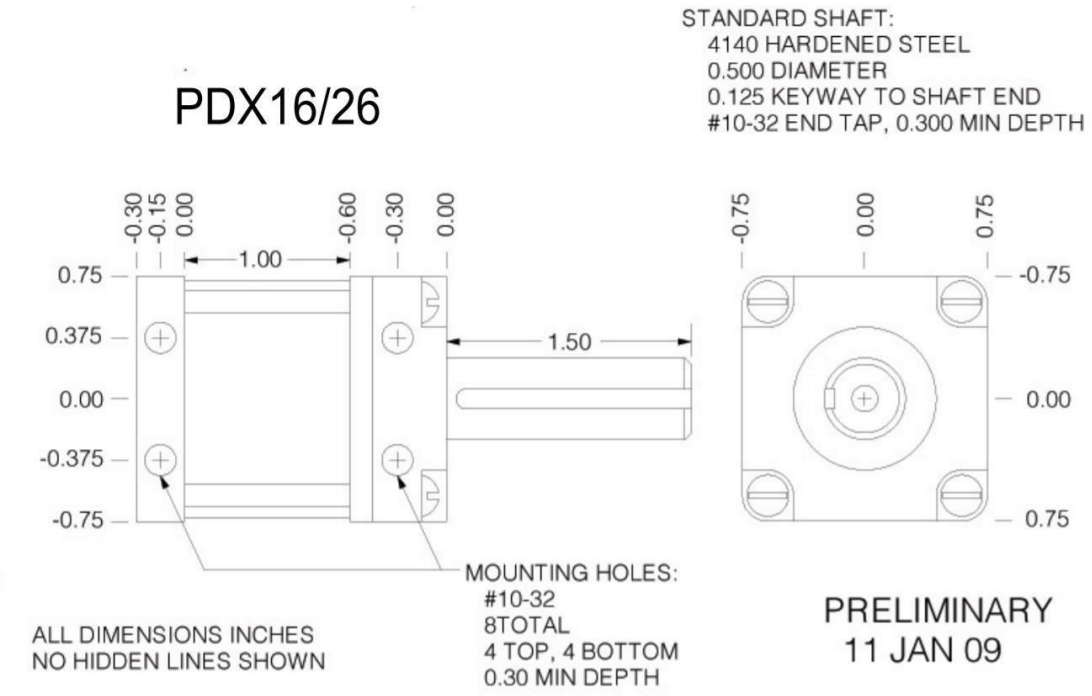


Figura 14-2: Motor PDX 26-26:1 Gearmotor dimensiones
Fuente: (Sabertooth 2X60 regenerative dual motor driver, s.f.).

Figura 15-2: Motor PDX 26-26:1 Gearmotor
Realizado por: Núñez, Bryan, 2021.

Tabla 2-2: Características del PDX26 - 26:1 Gearmotor

DESCRIPCIÓN	CARACTERÍSTICAS
Etapas	2 - 5:1, 5:1
Velocidad sin carga	900 rpm
Amps @ nominal:	1,5 Amps
Eficiencia	45,33 %
Punta de poder	0,55 hp
Corriente actual	148A
Torque actual	145 in-lb
peso	16,05 oz (455 grams)

Fuente: (Sabertooth 2X60 regenerative dual motor driver, s.f.).
Realizado por: Núñez, Bryan, 2021.

2.7.3. Controlador Sabertooth 2x60

El *Sabertooth* 2X60 caracterizado por su versatilidad, eficiencia y facilidad de usar en todo el mercado de los controladores de motores duales. Es un controlador óptimo si se quiere utilizar en competencias de robots de alta potencia de hasta 120 libras en combate y 1000 libras en uso para robótica, como se muestra en su Tabla 3-2 además de otras características. Se muestra en la Figura 16-2 (Sabertooth 2X60 regenerative dual motor driver, s.f.).

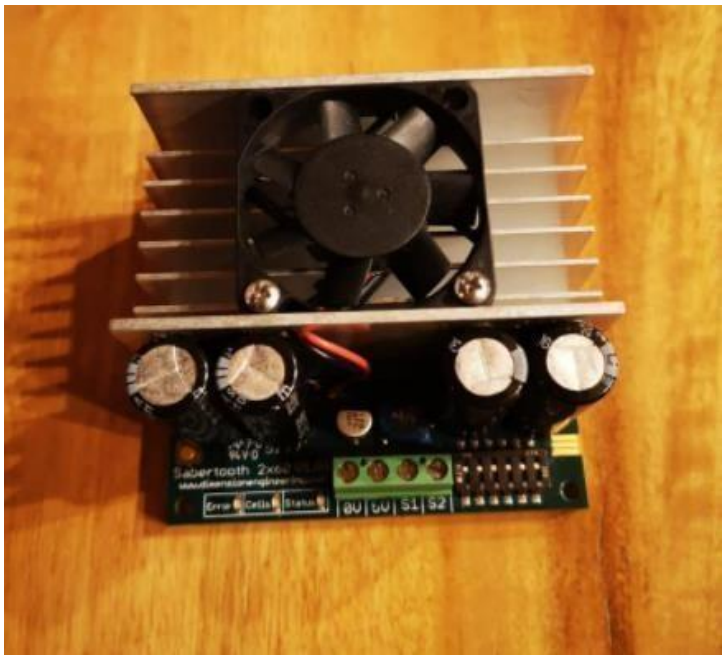


Figura 16-2: Sabertooth 2X60
Realizado por: Núñez, Bryan, 2020.

Tabla 3-2: Características del Sabertooth 2X60

DESCRIPCIÓN	CARACTERÍSTICAS
Corriente continua	60A.
Corriente pico	120 A.
Voltaje nominal	6-30 V.
Voltaje Max.	33,6 V.

Fuente: (Sabertooth 2X60 regenerative dual motor driver, s.f.).
Realizado por: Núñez, Bryan, 2021.

2.7.4. Baterías de litio

Las baterías son bancos de almacenamiento, cuenta con 6 celdas con un voltaje de 22.2V como se muestra en la Figura 16-2, un cableado capaz de cargar y descargar las celdas. El voltaje entregado es inducido directamente para la alimentación de los motores del prototipo y parte del sistema eléctrico.

Figura 17-2: Batería de polímero de litio
Realizado por: Núñez, Bryan, 2021.

Tabla 4-2: Características de la batería de polímero de litio

DESCRIPCIÓN	CARACTERÍSTICAS
Voltaje	22,2 V.
Numero de celdas	6
Capacidad	2800 mAh
Tasa de descarga	100 C
Tipo	recargable

Realizado por: Núñez, Bryan, 2021.

2.7.5. Switch

Son dispositivos electrónicos que permiten de una manera mecánica el control del encendido y apagado del sistema al que se encuentre conectado, el mismo que en un punto determinado del circuito permite o interrumpe el paso de la corriente eléctrica, como se muestra en la Figura 18-2 (Sanmartín, 2020). Se lo escogió como facilidad para adquirir la autonomía de encenderlo o apagarlo, para prevenir cualquier problema.

Figura 18-2: Switvh
Realizado por: Núñez, Bryan, 2021.

2.7.6. Sensores Sharp GP2Y0A21

Es un sensor que integra un detector de posición y un diodo emisor infrarrojo usando triangulación, este sistema permite determinar la distancia con un objeto (Nugroho, 2013). El seleccionado contiene el rango de medición de 10 cm a 80 cm. Como se muestra en la Tabla 5-2.

Figura 19-2: Sensor sharp
Fuente: (Nugroho, 2013).

Tabla 5-2: Características de la batería de polímero de litio

DESCRIPCIÓN	CARACTERÍSTICAS
Distancia de medición	10 cm. a 80 cm.
Salida Voltaje Analógico	(1V-3,3V)
Voltaje de alimentación	4,5V-5,5V DC
Consumo de corriente	30 mA.
Conector JST PH	3 pines
precio	$8,99

Fuente: (Nugroho, 2013).
Realizado por: Núñez, Bryan, 2021.

2.7.7. Procesador

Como se muestra en la Figura20-2 se utilizó una laptop HP *envy*x360 octava generación con Windows 10, la cual contribuyo con la capacidad de procesamiento requerida, así como tiempos de respuesta rápidos. En la Tabla 5-2 se presentan sus características principales.

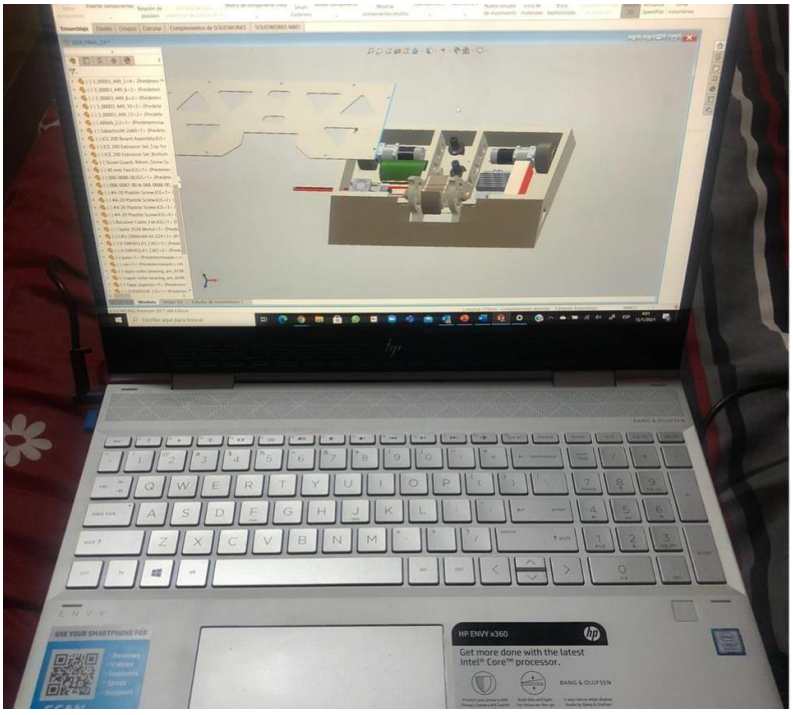


Figura 20-2: HP envy x360
Realizado por: Núñez, Bryan, 2020.

Tabla 6-2: Características de la HP *envy* x360

DESCRIPCIÓN	CARACTERÍSTICAS
Procesador	Intel Core i7
RAM	8
Tipo de sistema	Sistema operativo de 64 bits
Tarjeta grafica	NVIDIA® GEFORCE® MX150
Disco Duro	512Gb
precio	$899,99

Realizado por: Núñez, Bryan, 2021.

2.8. Esquema de conexión electrónica

La conexión electrónica de cada uno de los elementos utilizados en el sistema de ubicación espacial esta mostrada en la Figura 21-2.

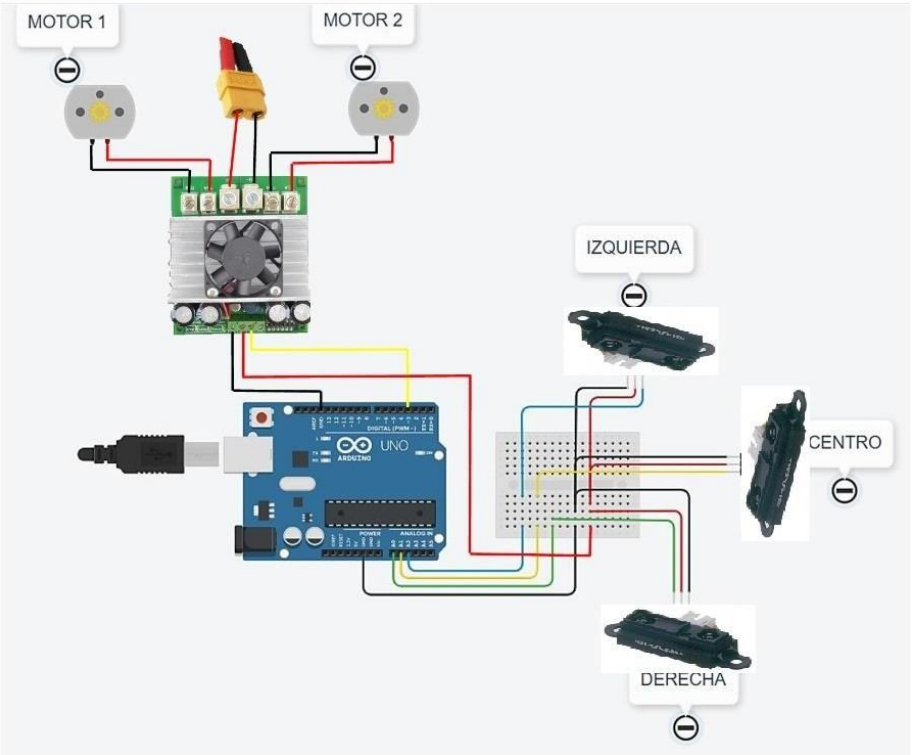


Figura 21-2: Esquema de conexión electrónica
Realizado por: Núñez, Bryan, 2020.

Tabla 7-2: Esquema de conexión

Terminales	Conexión
A0	SF1
A1	SF2
A2	SF3
D1	In1
D2	In2
RX	Recepción
TX	Transmisión
Vin	Terminal positivo batería
Gnd	Terminal negativo batería tierra

Realizado por: Núñez, Bryan, 2021.

Para la alimentación de todos los elementos que componen el sistema de ubicación espacial se utilizaron los terminales Vin y Gnd respectivamente. Para los sensores fotoeléctricos (SF1, SF2 Y SF3) se utilizaron los terminales analógicos A0, A1 Y A2. Para el controlador del motor se utilizaron los terminales D1 y D2 se la *saberthoot* que son aquellos que manejan la señal de modulación de ancho de pulso (PWM). Para la comunicación serial con el ordenador se utilizaron los puertos RX y TX.

2.9. Herramientas software de desarrollo

En la elaboración del sistema de ubicación espacial se utilizaron dos herramientas *software* para el movimiento y localización simultanea de la plataforma, las cuales son descritas a continuación.

2.9.1. IDE Arduino 1.8.13

El entorno de programación de Arduino es de código abierto basándose en un lenguaje de programación C++, es compatible con cualquier sistema operativo. Cuenta con librerías que facilitan la interacción con el hardware y el manejo de funciones para el desarrollo. Se define la librería *EEPROM.h, TimerOne.h, SoftwareSerial.h*, para obtener los valores de distancias del sensor *Sharp 2Y0A21*. Asignación de entradas analógicas A0, A1, A2, para los valores de distancia de los sensores respectivamente. Asignación de salidas digitales, como salidas los terminales D1, D2 respectivamente para los motores controlados a través de la tarjeta *saberthoot*.
Se declaran las variables para los límites y valores simplificados para cada motor:
SBT_MOTOR1_FULL_ADELANTE 58 y SBT_MOTOR1_FULL_REVERSA 70
SBT_MOTOR2_FULL_ADELANTE 186 SBT_MOTOR2_FULL_REVERSA 198.
Se escribe las variables de almacenamiento int: int promedio = 80, lejania = 150, valor_actual = 2, valor_anterior = 2, bandera1 = 0, bandera 2 =0, contador 1=0, val_imp =0.
Inicialización del puerto serial, con comunicación a 115200 baudios; inicialización de timer1 con su valor en micro segundos y activación de la interrupción del timer1 y la asocia a Isr_Timer.

Bloque repetitivo:

Lectura de sensores; leemos el promedio de la entrada analógica 0 ADC_SHARP0 = ADC0_promedio(promedio); Leemos el promedio de la entrada analógica 1 ADC_SHARP1 = ADC1_promedio(promedio); Leemos el promedio de la entrada analógica 2 ADC_SHARP2 = ADC2_promedio(promedio); Ejecutamos la función adelante con un valor actual de 2, giro a la derecha con un valor actual de 4, giro a la izquierda con un valor actual de 3 y reversa con un valor actual de 1. Mostrados en la imagen a través de la interrupción del *timer* y los valores asignados para cada movimiento se genera tres indicadores los mismo que serán utilizados en el software Matlab para la comunicación simultánea.

El código completo se muestra en el Anexo D.

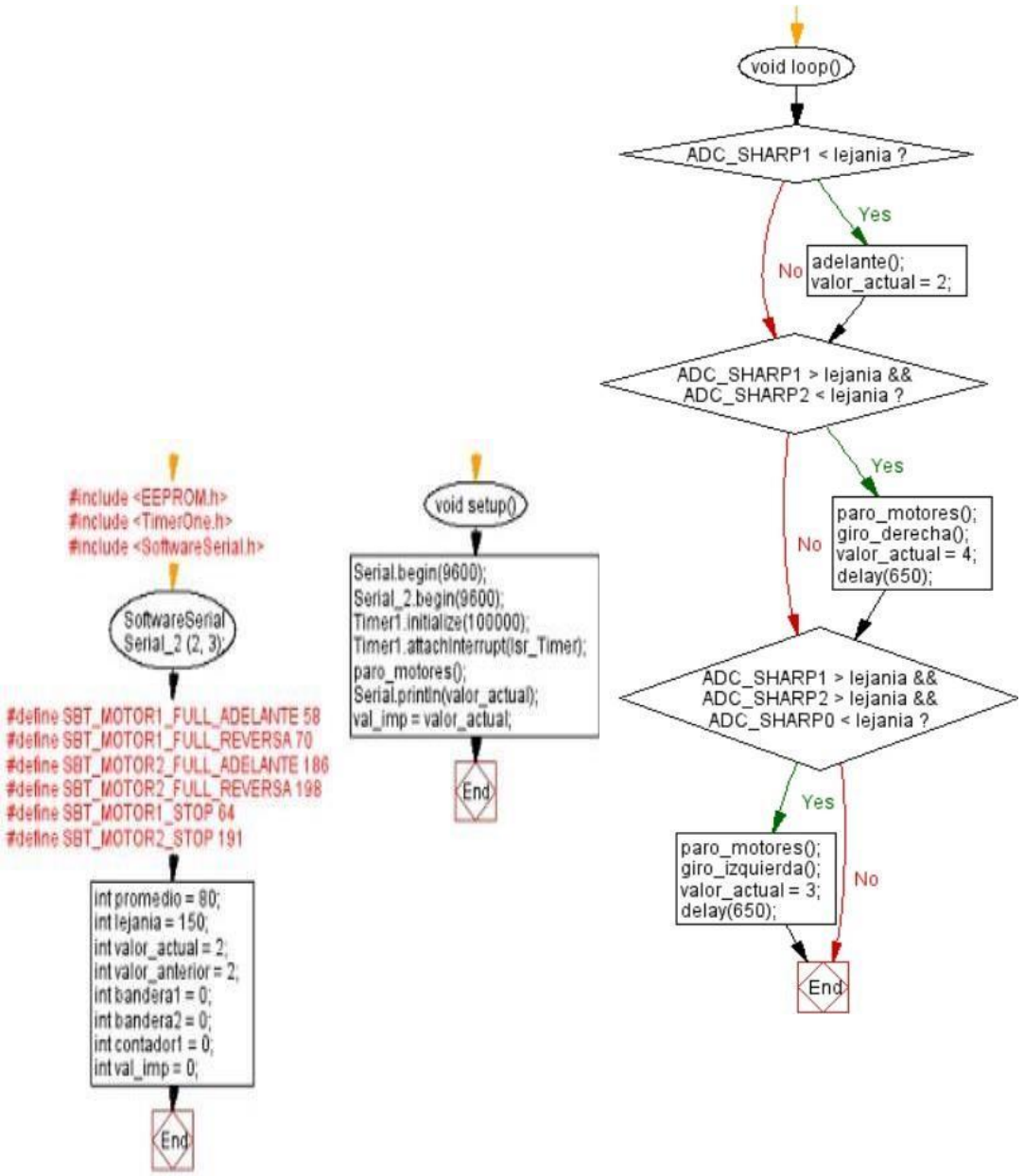


Figura 22-2: Código Arduino
Realizado por: Núñez, Bryan, 2020.

2.9.2. Matlab 2018b

Se inicializa el puerto serial en el que se encuentre ubicado del Arduino. A continuación, se configura el esquema del diseño de pantalla, así como la del mapa cartesiano en 2D. Declaración de variables globales, así como la coordina de inicio vertical y posición inicial del robot.

Se determina el tamaño del vector de la cola que significa el número de posiciones que avanza el robot mostradas en la interfaz. Para los obstáculos se crea simultáneamente un bloque de color verde. Se inicia la función de comparación de los valores obtenidos por el Arduino en el movimiento del robot para ser visualizados en la interfaz de Matlab.

Como es un mapa determinado en la interfaz se implementó una función que permite recircular

el robot al llegar a los límites planteados en la creación del mapa. Se muestra parte del código en la imagen, para más información ver Anexo E

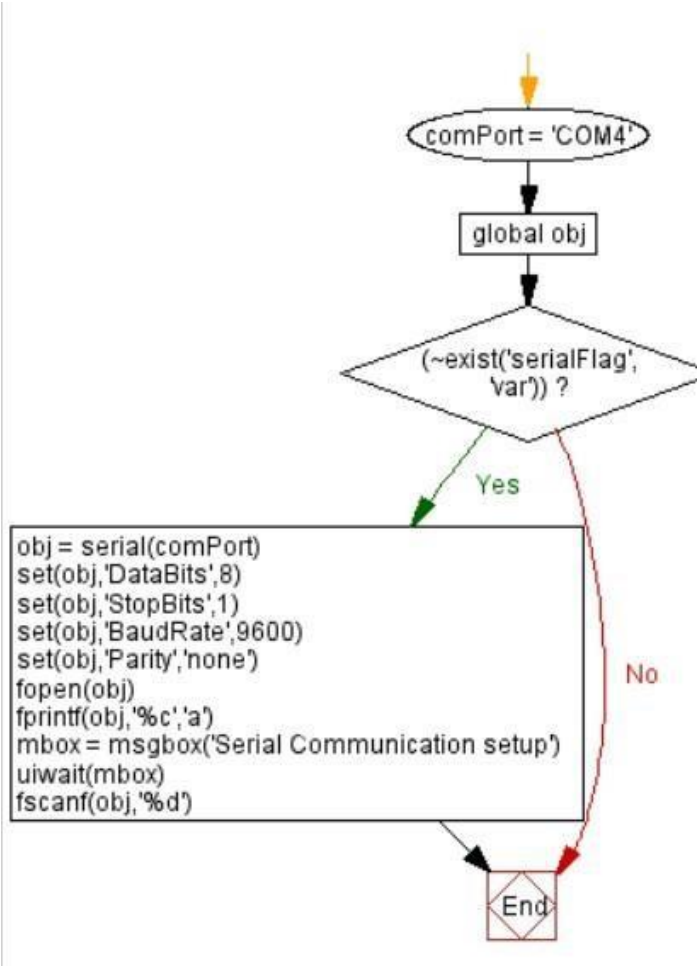


Figura 23-2: Código Matlab
Realizado por: Núñez, Bryan, 2020.

2.10. Diseño de la interfaz del sistema

Como se presenta en la figura se realizó una interfaz en la que se muestra en el lado derecho el número de obstáculos presentes en el mapa determinado, la posición del robot y la posición del obstáculo. Como se muestra en la interfaz el movimiento del robot es simultáneo al de plataforma móvil, tomando en cuenta que los cuadros verdes son obstáculos presentes en el mapa, los cuadros rojos la ruta por donde circulo la plataforma y el cuadro azul el punto actual donde se encuentra el robot.

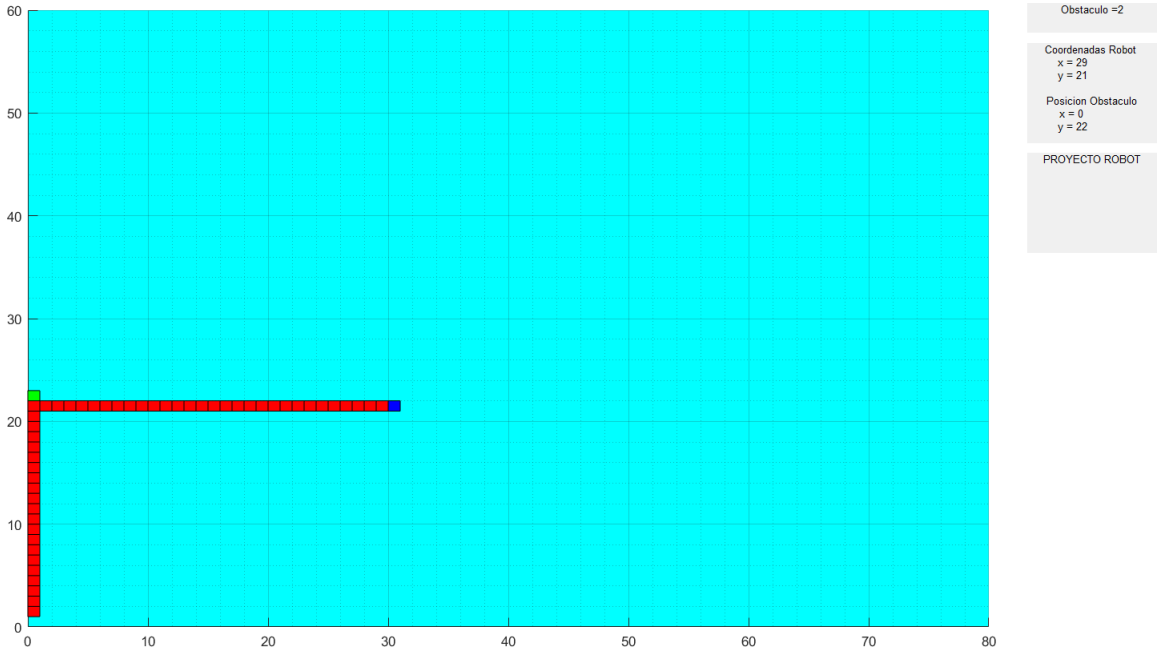


Figura 24-2: Interfaz del sistema
Realizado por: Núñez, Bryan, 2020.

CAPÍTULO III

3. ANÁLISIS DE RESULTADOS

Este capítulo describe los resultados obtenidos al finalizar el dispositivo tecnológico. Realizando las validaciones de cada uno de los bloques más importantes realizados en el proyecto. Las validaciones consisten en: medición de distancia, del movimiento de la plataforma, del algoritmo del movimiento de la plataforma y de la comunicación entre el ordenador y la plataforma móvil.

3.1. Validación de medición de distancia

Estas pruebas se realizaron para determinar el punto exacto en el que la distancia entre un obstáculo y la plataforma móvil no afecte al movimiento del mismo de acuerdo a sus dimensiones. En la Figura 1-3 se muestra el prototipo implementado.

Figura 1-3: Prototipo implementado
Realizado por: Núñez, Bryan, 2020.

En Tabla 1-3 se muestran los valores obtenidos de las mediciones en los sensores *Sharp*, concluyendo como punto óptimo para el funcionamiento y giro 155 de valor y con un promedio de 80 repeticiones como se muestra en la Figura 2-3. La curva de mediciones del sensor *Sharp* se presenta en la Figura 2-3 donde el rango está limitado por 0 V y 5V. El código para el análisis de la medición de distancias del robot se encuentra en el Anexo F.

Tabla 1-3: Valores sensor *Sharp*

Distancia (cm)	Valor mínimo	Valor máximo	Promedio
5	639	650	644,5
10	434	444	439
15	331	343	337
20	261	271	266
25	212	225	218,5
30	182	192	187
35	138	172	155
40	149	168	158,5
45	99	140	119,5
50	79	137	108
55	67	102	84,5
60	50	100	75
65	48	79	63,5
70	46	66	56
75	39	54	46,5
80	26	42	34

Realizado por: Núñez, Bryan, 2021.

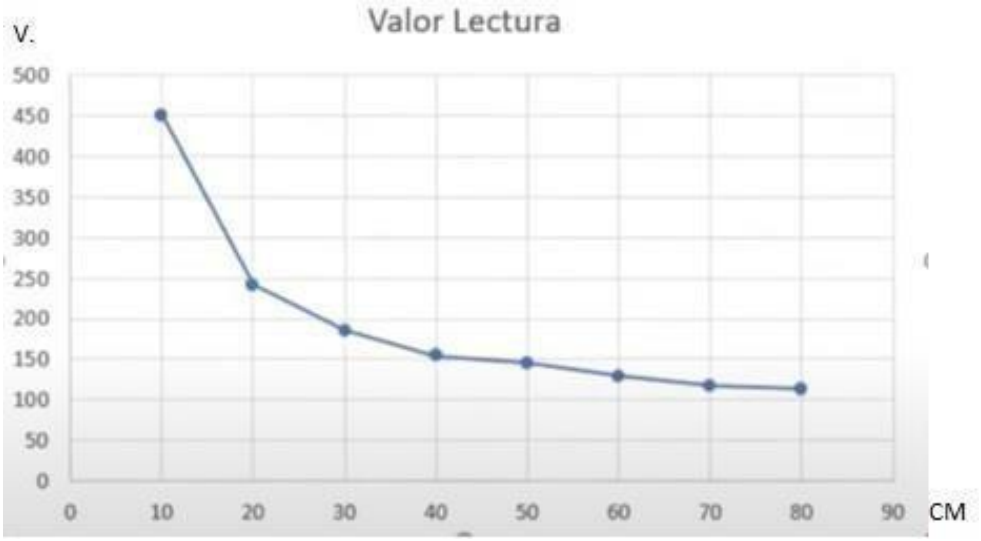


Figura 2-3: Valores de voltaje analógico entre 0 V. y 5 V
Realizado por: Núñez, Bryan, 2020.

3.2. Validación del movimiento de la plataforma

Para cada movimiento del robot se decidió poner un número fijo, el mismo que es empleado para interactuar con la interfaz de *Matlab*, resumiendo así el código y haciéndolo más simple; como se observa en la Figura 2-3. El código para el análisis del censado de direcciones del robot se encuentra en el Anexo G. Se realizó un arreglo que está definido por tres números, el primero es el actual que refleja la dirección en el que se encuentra, el segundo el numero para donde va a girar y el tercero y último el numero en la dirección que va a girar. El número 2 refleja una trayectoria delantera, el numero 3 refleja un izquierdo en dirección izquierda, el número 4 representa una dirección a la derecha y el número 1 representa un recorrido hacia atrás.

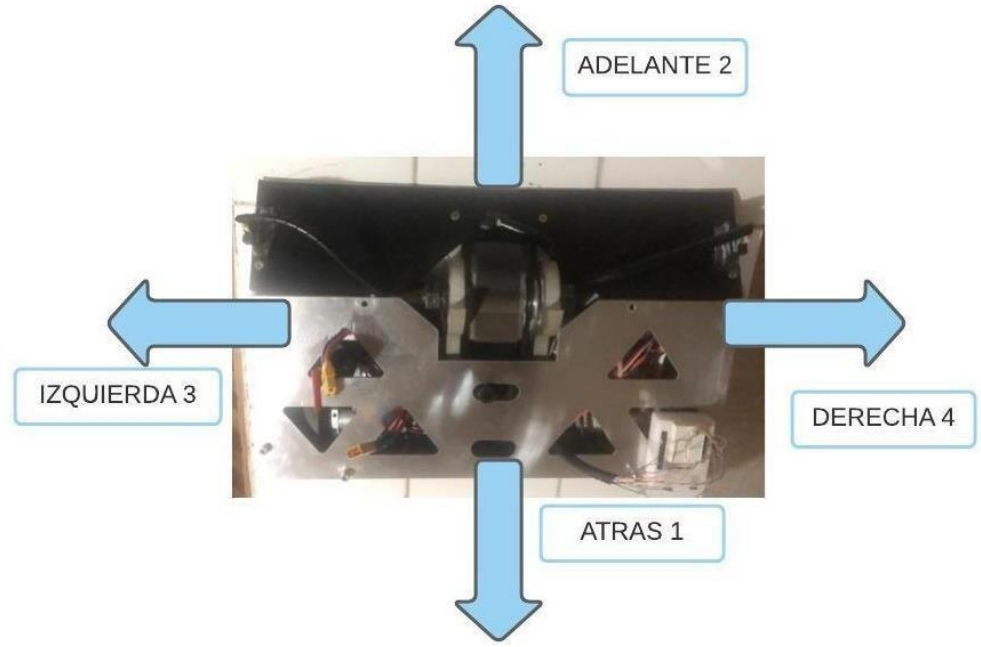


Figura 3-3: Plataforma y sus movimientos
Realizado por: Núñez, Bryan, 2020.

3.3. Validación de la comunicación entre el ordenador y la plataforma móvil

Se confirma la conexión serial a través de la interfaz de Matlab al visualizarse simultanéame con la trayectoria de la plataforma.

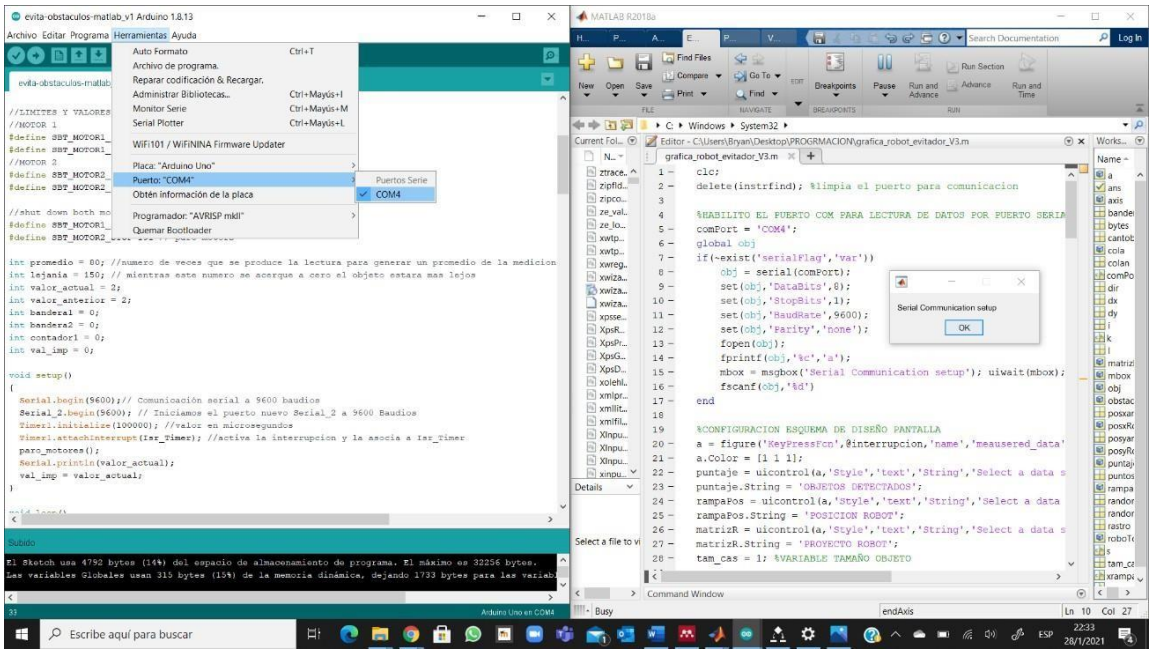


Figura 4-3: Conexión serial
Realizado por: Núñez, Bryan, 2020.

3.4. Validación entre Matlab y mapa determinado

En este apartado se realiza la evaluación entre la relación de la plataforma con *Matlab* comparándola en simultaneo con la relación de la plataforma con el mapa determinado. Para este análisis se toman los parámetros de tiempo, distancia y velocidad.

3.4.1. Distancias

Para la medición de distancia se utilizó un flexómetro para corroborar las respectivas mediciones tanto del mapa determinado como la interfaz de Matlab, como se muestra en la Figura 4-3.

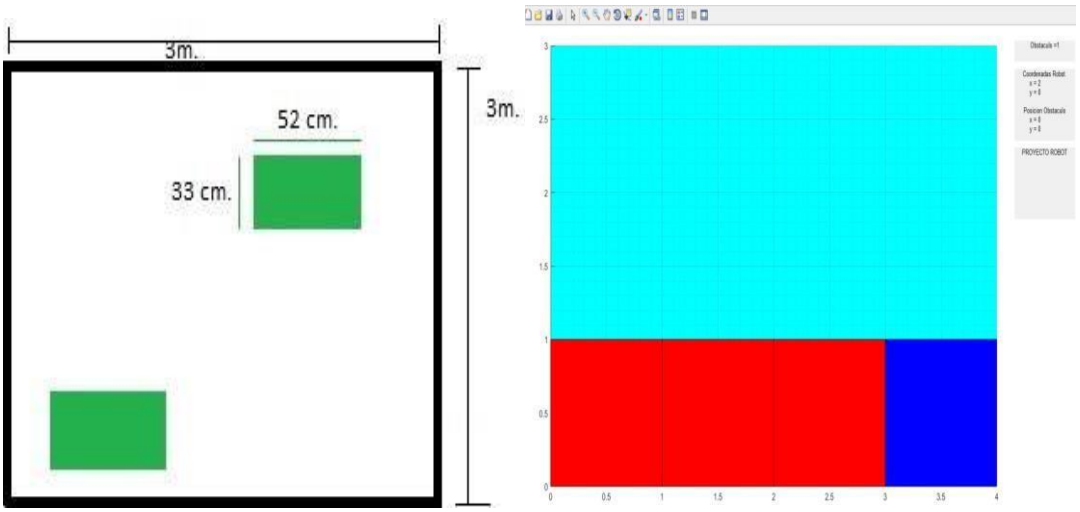


Figura 5-3: Relación entre la interfaz y matlab
Realizado por: Núñez, Bryan, 2021.

3.4.2. Velocidad

El valor de velocidad óptimo tomando en cuenta las dimensiones del robot y la distancia de medición de un obstáculo para cada motor tanto en dirección delantera como en una dirección trasera es de 21.48 rpm, los motores están trabajando a un ancho de pulso de 8 bits, cada motor ocupa la mitad de la capacidad del PWM ver Anexo H. Para escoger la velocidad óptima se realizaron diferentes pruebas cuyos valores son mostrados en la siguiente Tabla 2-3.

Tabla 2-3: Valores sensor *sharp*

	Motor 1		**Motor 2**	
Valores PWM	**Velocidad adelante (rpm)**	**Velocidad atrás (rpm)**	**Velocidad adelante (rpm)**	**Velocidad atrás(rpm)**
58	**21,42857151**	-	-	-
64	**PARO**	-	-	-
70	-	**21,42857151**	-	-
191	-	-	**PARO**	-
198	-	-	-	**21,42857151**

3.4.3. Tiempo

Para el cálculo del tiempo se utilizó un cronómetro en el que evaluamos en repetidas ocasiones el desplazamiento de la plataforma móvil desde un punto de inicio hasta un punto final. El tiempo

4,05 segundos en promedio a lo largo del eje Y como se muestra en la Tabla 3-3; mientras que en el eje X 2,55 segundos en promedio, como se muestra en la Tabla 4-3.

Tabla 3-3: Distancia vs Tiempo

#	Distancia (cm)	Tiempo (s)
1	190	3,67
2	190	3,9
3	190	4,55
4	190	4,666
5	190	4,15
6	190	3,88
7	190	3,87
8	190	3,68
9	190	3,94
10	190	4,2

Realizado por: Núñez, Bryan, 2021.

Tabla 4-3: Valores de tiempo

#	Distancia (cm)	Tiempo (s)
1	120	2,33
2	120	2,44
3	120	2,67
4	120	2,45
5	120	2,46
6	120	2,99
7	120	2,51
8	120	2,71
9	120	2,33
10	120	2,58

Realizado por: Núñez, Bryan, 2021.

3.5. Tiempo de giro de la plataforma

Para evaluar el tiempo de repuesta a un obstáculo se utilizó un cronómetro, el mismo que ayuda a calcular el tiempo de giro de la plataforma como se muestra en la Figura 4-3. Los datos recolectados se encuentran en la tabla, tomando como un valor aproximado de giro de 1,60 segundos.

Figura 6-3: Tiempo de respuesta a un giro de 90 grados
Realizado por: Núñez, Bryan, 2021.

Tabla 5-3: Valores de tiempo - Giro

	Giro	
#	**Derecha**	**Izquierda**
1	1,58	1,55
2	1,63	1,49
3	1,6	1,59
4	1,55	1,65
5	1,72	1,69
6	1,71	1,71
7	1,7	1,68
8	1,55	1,51
9	1,53	1,63
10	1,56	1,58

Realizado por: Núñez, Bryan, 2021.

3.6. Validación de la evasión de obstáculos

El algoritmo implementado en la plataforma para el mapa determinado, al detectar un obstáculo en la dirección que se encuentre lo evade; simultáneamente se produce la comunicación serial para *Matlab* haciendo que se muestre su trayectoria gráficamente.

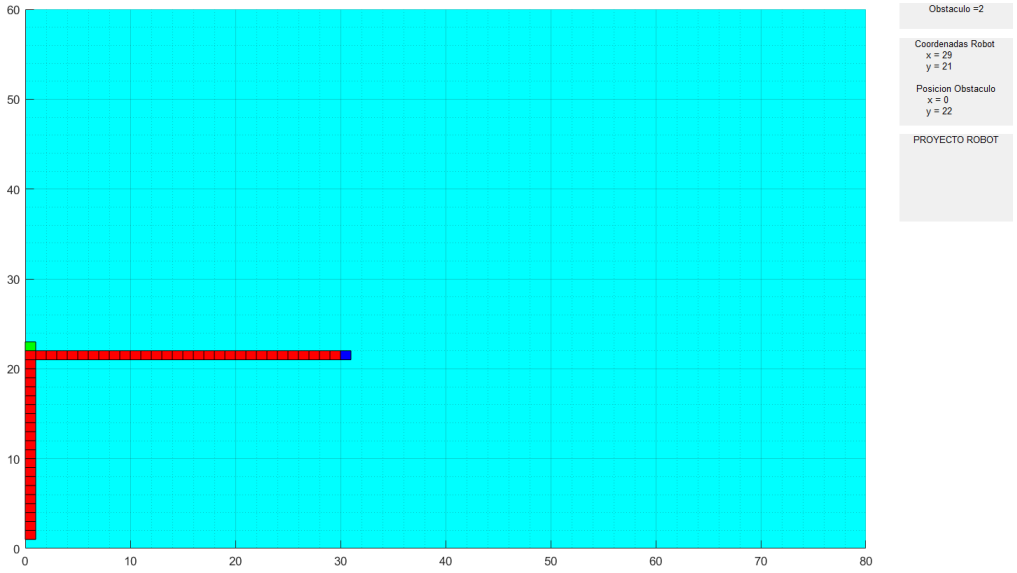


Figura 7-3: Interfaz con obstáculo
Realizado por: Núñez, Bryan, 2021.

3.7. Validación de movimiento simultaneo entre la plataforma y Matlab

Para una respuesta simultánea en tiempo real u óptimo se utilizó cable serial, ya que esta es la mejor opción para garantizar la comunicación con una velocidad de 9600 baudios, la misma utilizada en el código implementado y en la plataforma móvil, como se muestra en la Figura 6-3

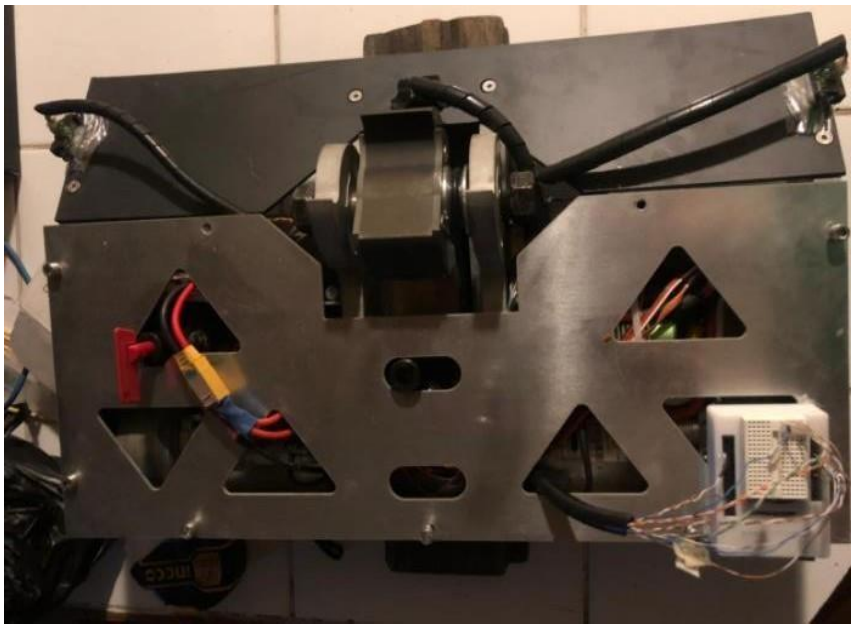


Figura 8-3: Conexión serial
Realizado por: Núñez, Bryan, 2021.

CAPÍTULO IV

4. EVALUACIÓN ECONÓMICA

En este apartado se realiza la relación costo/beneficio del prototipo diseñado en el cual se detallan el costo de los materiales de fabricación y sus características en comparación con un prototipo comercial.

Tabla 1-4: Matriz de presupuesto del prototipo diseñado

Precio de los componentes				
No	**Nombre**	**Cantidad**	**Precio U. $**	**Total $**
1	Bloque de aluminio	1	130	130
2	Arduino uno	1	20	20
3	Motor PDX26 – 26:1 GEARMOTOR	2	89,99	179,98
4	Controlador Sabertooth 2x60	1	189,99	189,99
5	Baterías de litio	1	119,99	119,99
6	Switch cutoff	2	15	30
7	Sensores Sharp GP2Y0A21	3	8,99	26,97
8	Llantas	2	9	18
9	cables de arduino	1	3	3
TOTAL $				717,93

Realizado por: Núñez, Bryan, 2021.

El precio total del prototipo diseñado fue de 717,93 cada uno de los componentes fue seleccionado bajo un análisis previo para cumplir con los objetivos.

Tabla 2-4: Matriz de comparación de presupuestos

Característica/nombre	**Autor**	***Autonómous Programmable ROS SLAM Tracked Robot Package***	***Lidar slam robot car ROS-01***
Dimensión	52 cm * 33 cm	25,6 cm * 33cm	28cm * 24,2 cm
Peso	28,5 lb	8,3 lb	6,6 lb
Tarjeta de desarrollo	Arduino uno	*BeagleBone Black (Rev C)*	*Respberry Pi 3B+*
Lenguaje de programación	C++	C++	*Python*

Velocidad	900 rpm	300 rpm	360 rpm
Precio	$ 717.93	$ 4260	$1200

Realizado por: Núñez, Bryan, 2021.

En comparación con el prototipo comercial *Autonomous Programmable ROS SLAM Tracked Robot Package* el mismo que ofrece funciones similares, se diferencia en la tarjeta de desarrollo que controla todo el proceso, tiene integrado un sensor YDLIDAR G4 el mismo que permite una alta precisión al detectar obstáculos y está valorado en $4260. Otro robot comercial es el *Lidar slam robot car ROS-01* cuyo valor comercial es de $1200.

El costo del prototipo diseñado por el autor es significativamente menor para las prestaciones que ofrecen los demás teniendo características similares de durabilidad y resistencia como la adquirida por una estructura de aluminio resistente a golpes.

CONCLUSIONES

Para el diseño de un sistema de ubicación espacial de plataforma móvil mediante sensores en un mapa determinado se realizó un estudio previo del estado del arte; dentro del cual los temas relevantes para el diseño y construcción del mismo son los siguientes: programación, comunicación y modelado. Se encontraron robots móviles que utilizan varios sistemas de ubicación espacial tanto en interiores como exteriores, estos robots son utilizados ampliamente en sectores industriales, de exploración entre otros. En los que es indispensable conocer su posición y orientación en busca de cumplir con tareas específicas.

Se seleccionaron los componentes después de realizar un análisis al aluminio 7075, siendo éste fácil de mecanizar, desarrolla una plataforma estable que protege a los materiales internos con menos resistencia a la corrosión y éste al ser un material que no se puede soldar permite tener una plataforma modular. Para los componentes electrónicos se priorizó los parámetros de voltaje, consumo de corriente, potencia y en el caso de los motores, el número de revoluciones por minuto para lo cual se seleccionó los motores de engranaje que ofrece una velocidad de 900 rpm de las cuales para el giro y evasión de obstáculos se necesita 21.48 rpm por cada motor, siendo el peso total de la plataforma 28,5 lb. Se utilizaron las librerías de *matlab* para que el sistema de ubicación y evasión de obstáculos de la plataforma realice una comunicación simultánea en el proceso visualizando los objetos y la trayectoria de la plataforma. Para optimizar la detección de obstáculos se utilizó sensores *Sharp* y como punto óptimo de distancia 35cm.

Para el diseño del sistema de posición espacial se utilizó comunicación *serial* entre la plataforma móvil y el computador para el envío y recepción de datos, garantizando la visualización de los objetos y de la trayectoria simultánea a medida que esta se desplace.

Mediante el sistema de percepción implementado, el robot se desplazó en el entorno de trabajo evadiendo obstáculos y se ubicó su posición en el espacio.

RECOMENDACIONES

En la navegación se detectaron problemas cuando existen cables u objetos en el mapa, para evitar dichos inconvenientes y garantizar el desplazamiento adecuado de la plataforma se requiere una superficie uniforme.

Se recomienda usar sensores láser para evitar obstáculos pequeños de menor altura a 20 cm, sin una correcta ubicación de los sensores *sharp* no pueden ser detectados.

Se recomienda realizar un análisis de los sensores *Sharp* los mismos que identifican obstáculos a una distancia máxima de 80 cm. y la distancia óptima para la evasión de estos es de 35 cm, por la estructura y dimensión de la plataforma, permitiéndole realizar el giro adecuado.

Para trabajos futuros se podría considerar utilizar un sistema de control adecuado para optimizar los recursos de la plataforma en batallas de robots, donde los sistemas de reducción de velocidad podrían ser aplicados ganado torque, fundamental en esta aplicación.

BIBLIOGRAFÍA

BADAMASI, Y.A., The working principle of an Arduino. *Proceedings of the 11th International Conference on Electronics, Computer and Computation, ICECCO* [en línea]. S.l.: Institute of Electrical and Electronics Engineers Inc., 2014. pp. 1-4. [Consulta: 15 enero 2021]. ISBN 9781479941063. DOI 10.1109/ICECCO.2014.6997578. Disponible en: http://ieeexplore.ieee.org/document/6997578/.

BARRIENTOS PEÑIN BALAGUER ARCIN, *(Barrientos Peñin Balaguer Arcin) Fundamentos de Robotica.pdf*. 1996 *2007*. S.l.: s.n.

CORONA, L.G., ABARCA, G.S. y MARES, J., Sensores y actuadores aplicaciones con Arduino. *Publicacion En Internet* [en línea], no. October, 2014. pp. 304. ISSN 2198-6452. Disponible en: http://es.slideshare.net/kaluisitoblog/sensores-yactuadoresseat.

CRAIG, J.J., *John Craig.* 1977. S.l.: s.n. ISBN 9702607728.

FELIPE, F., MONDRAGÓN, M., SÁNCHEZ, M., DURÁN, F.F., M, N.M. y M, M.S., Redes cableadas e inalámbricas para transmisión de datos. *Científica*, vol. 12, no. 3, 2008. pp. 113-118. ISSN 1665-0654.

Anónimo., MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink. [en línea], sf. [Consulta: 17 agosto 2020]. Disponible en: https://es.mathworks.com/products/matlab.html.

MECÁNICA, F. DE, POR, P. y DIEGO CRUZ FREIRE DARWIN VINICIO CHIMBO CHIMBO, J., Escuela Superior Politécnica De Chimborazo. , MORA, F., *Máquinas eléctricas (6a. ed.)*. 2015. S.l.: s.n. ISBN 9788448161125.

NUGROHO, M.B., 済無 No Title No Title. *Journal of Chemical Information and Modeling*, vol. 53, no. 9, 2013. pp. 1689-1699. ISSN 1098-6596. DOI 10.1017/CBO9781107415324.004.

OLLERO, A.Bb., ***Ollero Baturone, Aníbal.*** *Robótica, manipuladores y robots móviles.pdf*. 2001. S.l.: s.n.

Aníbal Ollero Baturone., Robótica: Manipuladores y Robots Móviles - Google Libros. [en línea], sf. [Consulta: 17 agosto 2020]. Disponible en: https://books.google.es/books?hl=es&lr=&id=TtMfuy6FNCcC&oi=fnd&pg=PR13&dq=ro

bot+que+es&ots=32P_I2x58R&sig=udN- sDUzt9UEX_DP90uzBDZoecI#v=onepage&q=robot que es&f=false.

REYES, Fernando., Robótica - control de robots manipuladores - Google Libros. [en línea], sf. [Consulta: 17 agosto 2020]. Disponible en: https://books.google.es/books?hl=es&lr=&id=cULVDQAAQBAJ&oi=fnd&pg=PT5&dq= que+es+un+robot&ots=LQ6Ittu0c0&sig=UaahDBuUq_boAO89ulZi0ZqQLw#v=onepage&q=que es un robot&f=false.

Anónimo., Sabertooth 2X60 regenerative dual motor driver. [en línea],sf. [Consulta: 17 agosto 2020]. Disponible en: https://www.dimensionengineering.com/products/sabertooth2x60.

SAHA, S.K., *Introduccion a la Robotica*. 1998. S.l.: s.n. ISBN 9786071503138.

SANMARTÍN, F.E.L., Facultad de Ingeniería Carrera de Ingeniería de Sistemas. , SORIANO, A., MARÍN, L., JUAN, R., CAZALILLA, J., VALERA, A., VALLÉS, M. y ALBERTOS, P., PLATAFORMA ROBÓTICA DE BAJO COSTE Y RECURSOS LIMITADOS BASADA EN ARDUINO Y DISPOSITIVOS MÓVILES. 2020. S.l.:

ANEXOS

ANEXO A: HOJA DE DATOS ARDUINO WEMOS D1R1

2. Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

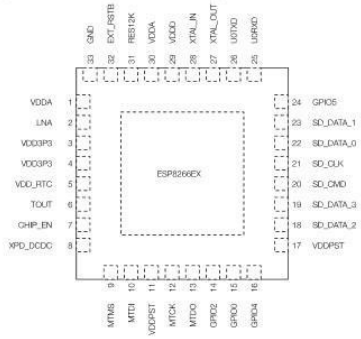


Figure 2-1. Pin Layout (Top View)

Table 2-1 lists the definitions and functions of each pin.

Table 2-1. ESP8266EX Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5 V ~ 3.6 V
2	LNA	I/O	RF antenna interface Chip output impedance = 39 + j5 Ω. It is suggested to retain the π-type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V

Pin	Name	Type	Function
4	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
5	VDD_RTC	P	NC (1.1 V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.
7	CHIP_EN	I	Chip Enable High: On, chip works properly Low: Off, small current consumed
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 20 Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200 Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200 Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200 Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200 Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200 Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART TX during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5 V ~ 3.6 V
30	VDDA	P	Analog Power 2.5 V ~ 3.6 V

Pin	Name	Type	Function
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: active)

Note:

1. *GPIO2, GPIO0, and MTDO are used to select booting mode and the SDIO mode.*
2. *U0TXD should not be pulled externally to a low logic level during the powering-up.*

3. Functional Description

The functional diagram of ESP8266EX is shown as in Figure 3-1.

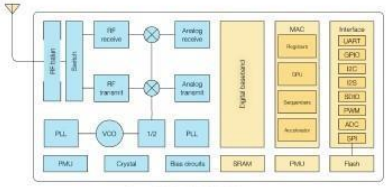


Figure 3-1. Functional Block Diagram

3.1. CPU, Memory, and Flash

3.1.1. CPU

The ESP8266EX integrates a Tensilica L106 32-bit RISC processor, which achieves extra-low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development. The CPU includes the interfaces as below:

- Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit flash.
- Data RAM interface (dBus), which can connected with memory controller.
- AHB interface which can be used to visit the register.

3.1.2. Memory

ESP8266EX Wi-Fi SoC integrates memory controller and memory units including SRAM and ROM. MCU can access the memory units through iBus, dBus, and AHB interfaces. All memory units can be accessed upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

Realizado por: Núñez, Bryan, 2021.

ANEXO B: HOJA DE DATOS SABERTHOOT

Battery Terminals B+ and B-

The battery or power supply is connected to terminals B- and B+. B- connects to the negative side of the battery (usually black.) B+ connects to the positive side of the battery (usually red or yellow.) It is usually best to connect the battery through a connector (a big one!) instead of directly to the motor driver. This makes it easy to unplug the battery for charging, and prevents plugging in the battery backwards.

Another possibility is to use a ***heavy duty*** disconnect rated at ***100A or more*** to switch power on and off to the 2x60. This will allow for easy shut down and will reduce the chances of a reverse hookup.

The battery connects to terminals B+ and B-

Warning! Be very careful to wire and plug in the battery and connector correctly. Connecting the battery backwards will destroy the Sabertooth and will void the warranty.

Motor Terminals

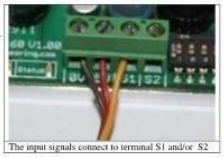
Motor 1 is connected to terminals M1A and M1B. If the motor runs in the opposite way that you want, you may reverse the motor wires to reverse rotation.

Motor 2 is connected to terminals M2A and M2B as shown to the right.

The motors connect to terminals M1A/B and M2A/B

Signal Input Terminals S1 and S2

The input signals that control the Sabertooth are connected to terminals S1 and S2. If you are running in analog mode, it is important to have both the signal wires connected before applying power to the device. Otherwise, the motors may start unexpectedly.

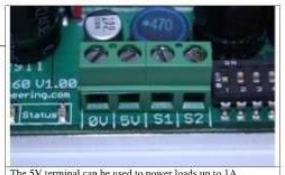


The input signals connect to terminal S1 and/or S2

Power terminals 0V and 5V

The 0V and 5V connections are used to power and interface to low-power control circuits.

The 5V connection is a 5v power **output**. The 2x60 utilizes a **1 Amp switching BEC** to power the onboard electronics as well as to provide power to your receiver and up to 4 standard analog servos. You can power anything that requires 5V straight from the Sabertooth 2x60. There is no need for an external BEC unless you need more than 1 Amp. The BEC will work at full rated output throughout the Sabertooth's operating voltage range. You can use the BEC at full capacity whether you are running 7V or 24V in.



The 5V terminal can be used to power loads up to 1A continuously and 1.5A for peaks. The 0V signal must be connected to the ground of the device generating the input signal.

The 0V connection is the signal ground for the Sabertooth. In order to receive input signals correctly, it must be connected to the ground of the device sending the signals. (Note: Internally connected to B-)

Using the 0V and 5v connections to power a radio receiver in R/C mode and potentiometer in analog mode is shown in Figures 2.1 and 2.2. If you are using multiple Sabertooths running from the same radio receiver, only one should have the 5v line connected.

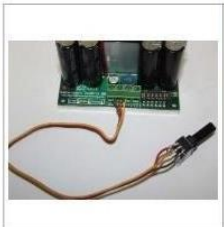


Figure 2.1: Analog input using a potentiometer powered from terminal 5V.

Figure 2.2: R/C input using a receiver powered from terminal 5V

Status and Error LEDs

Sabertooth 2x60 has three indicator LEDs.

The green LED marked Status is used to communicate various information about the current state. In most cases Status acts as a power indicator. In R/C mode, it glows dimly if there is no RC link present and brightly if there is an RC link.

The green LED marked Cells will blink the amount of cells you have attached when running in Lithium mode. Cells will also illuminate along with the Error LED if you have tripped the under-voltage alarm.

All Status LEDs on

The red Error LED illuminates if the Sabertooth has detected a problem. It will light if the driver has shut down due to a depleted battery or due to overheating, overcurrent or overvoltage. The Error LED will flash along with Cells if there is an issue with your battery. If both of those are blinking simultaneously, your battery is depleted.

Mounting your Sabertooth 2x60

The Sabertooth is supplied with four mounting holes. These can be used to attach it to your robot. The centers of the mounting holes form a 3.25" x 2.75" rectangle. The holes are .125 inches in diameter. The proper size screw is a 4-40 round head machine or wood screw. Four 5/8" long machine screws and nuts are included.

Sabertooth 2x60 has an onboard fan and heat sink, so it has slightly different mounting requirements than other Dimension Engineering motor drivers which are passively cooled. You do not need to worry about whether your mounting surface is thermally conductive or insulating - standoffs are not required from a thermal perspective. However, to ensure adequate airflow, please ensure that the top and sides of the unit are not tightly enclosed. Air is drawn in by the fan on top of the unit, blown through the large heat sink, and exhausted out the sides of the heat sink. These three sides should be no less than 3/4" from the faces of any enclosure to allow for adequate airflow.

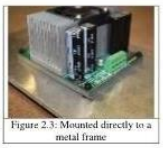


Figure 2.3: Mounted directly to a metal frame

Realizado por: Núñez, Bryan, 2021.

ANEXO C: HOJA DE DATOS SENSOR SHARP

SHARP GP2Y0A21YK0F

GP2Y0A21YK0F

Distance Measuring Sensor Unit
Measuring distance: 10 to 80 cm
Analog output type

■Description

GP2Y0A21YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector) , IRED (infrared emitting diode) and signal processing circuit.
The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.
This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

■Features

1. Distance measuring range : 10 to 80 cm
2. Analog output type
3. Package size : 29.5×13×13.5 mm
4. Consumption current : Typ. 30 mA
5. Supply voltage : 4.5 to 5.5 V

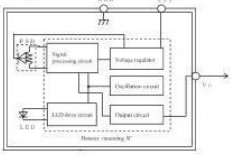
■Agency approvals/Compliance

1. Compliant with RoHS directive (2011/65/EU)

■Applications

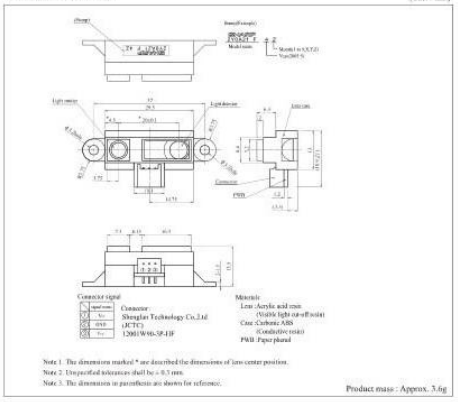
1. Touch-less switch
   (Sanitary equipment, Control of illumination, etc.)
2. Robot cleaner
3. Sensor for energy saving
   (ATM, Copier, Vending machine)
4. Amusement equipment
   (Robot, Arcade game machine)

■Block diagram



■Outline Dimensions

(Unit : mm)



Connector signal

1	Signal output	Connector:
2	V_{CC}	Shinglan Technology Co.,Ltd.
3	GND	(JCTC)
4	V_{CC}	12001W04-3P-1H

Materials
Lens :Acrylic acid resin
(Visible light cut-off type)
Case :Carbonic ABS
(Conductive resin)
PWB :Paper phenol

Note 1. The dimensions marked * are described the dimensions of lens center position.
Note 2. Unspecified tolerances shall be ± 0.3 mm.
Note 3. The dimensions in parenthesis are shown for reference.

Product mass : Approx. 3.6g

■Absolute Maximum Ratings

(T_a=25°C,V_{CC}=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	-0.3 to +7	V
Output terminal voltage	V_O	-0.3 to V_{CC}+0.3	V
Operating temperature	T_{opr}	-10 to +60	°C
Storage temperature	T_{stg}	-40 to +70	°C

■Electro-optical Characteristics

(T_a=25°C,V_{CC}=5V)

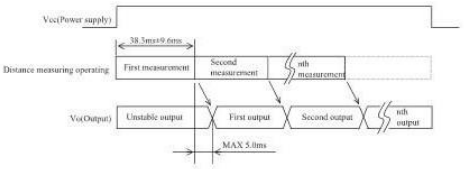
Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Average supply current	I_{CC}	L=80cm (Note 1)	—	30	40	mA
Distance measuring	ΔL	(Note 1)	10	—	80	cm
Output voltage	V_O	L=80cm (Note 1)	0.25	0.4	0.55	V
Output voltage differential	ΔV_O	Output voltage difference between L=10cm and L=80cm (Note 1)	1.65	1.9	2.15	V

* L : Distance to reflective object
Note 1 : Using reflective object : White paper (Made by Kodak Co., Ltd. gray cards R-27・white face, reflectance: 90%)

■Recommended operating conditions

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	4.5 to 5.5	V

Fig. 1 Timing chart



Realizado por: Núñez, Bryan, 2021.

ANEXO D: CÓDIGO DE ARDUINO

```
#include <EEPROM.h>

/*PROGRAMA PARA DESPLAZAR UN ROBOT A MANERA DE EVITADOR DE OBSTACULOS
  REALIZADO CON UN ARDUINO UNO, 3 SENSORES SHARP 2Y0A2. ELARDUINO SE ENCUENTRA CONECTADO A TRAVES DE PUERTO VIRTUAL (2,3) A UNA SABERTOOTH 2X60
LAS CONEXIONES DEL ARDUINO SON
A0 ---> SHARP IZQ
A1 ---> SHARP CENTRAL
A2 ---> SHARP DER
D3(TX) ---> S1 SABERTOOTH
GND ---> V0 SABERTOOTH

EL ORDEN DE PRIORIDAD ES EL SIGUIENTE:
EL ROBOT SE DESPLAZA HACIA AL FRENTE SI ENCUENTRA UN OBSTACULO ANALIZA SU DERECHA E IZQUIERDA, SI LA DERECHA ESTA LIBRE
GIRA A LA DERECHA Y SI PRESENTA OBSTÁCULO A LA IZQUIERDA
*/
#include <TimerOne.h>
#include <SoftwareSerial.h>
SoftwareSerial Serial_2 (2, 3); // Crea nueva conexion- Pin2(RX)y Pin3(TX)


//LIMITES Y VALORES SIMPLIFICADOS PARA CADA MOTOR
//MOTOR 1
#define SBT_MOTOR1_FULL_ADELANTE 58 //1 (rapido) --->63 (lento) #define SBT_MOTOR1_FULL_REVERSA 70 //65 (lento) --->127 (rapido)
//MOTOR 2
#define SBT_MOTOR2_FULL_ADELANTE 186 //128 (rapido) --->191 (lento)
#define SBT_MOTOR2_FULL_REVERSA 198 //193 (lento) --->255 (rapido)

//shut down both motors
#define SBT_MOTOR1_STOP 64 //paro motor1
```

```
#define SBT_MOTOR2_STOP 191 // paro motor2

int promedio = 80; //numero de veces que se produce la lectura para generar un promedio de la medicion
int lejania = 150; // mientras este numero se acerque a cero el objeto estara mas lejos
int valor_actual = 2;
int valor_anterior = 2;
int bandera1 = 0;
int bandera2 = 0;
int contador1 = 0;
int val_imp = 0;

void setup()
{
Serial.begin(9600);// Comunicación serial a 9600 baudios Serial_2.begin(9600); // Iniciamos el puerto nuevo Serial_2 a 9600 Baudios Timer1.initialize(100000); //valor en microsegundos
Timer1.attachInterrupt(Isr_Timer); //activa la interrupcion y la asocia a Isr_Timer paro_motores();
Serial.println(valor_actual);
val_imp = valor_actual;
delay(10000);
}


void loop()
{

//LECTURA SENSORES
// Leemos el promedio de la entrada analógica 0
int ADC_SHARP0 =ADC0_promedio(promedio);
// Leemos el promedio de la entrada analógica 1
int ADC_SHARP1 =ADC1_promedio(promedio);
// Leemos el promedio de la entrada analógica 2
```

```
int ADC_SHARP2 = ADC2_promedio(promedio);
//  Serial.println(ADC_SHARP0);
//  Serial.println(ADC_SHARP1);
//  Serial.println(ADC_SHARP2);
//   delay(500);



if (ADC_SHARP1 < lejania)
{
adelante();
valor_actual = 2;
//Serial.println("delante");
}
if (ADC_SHARP1 > lejania && ADC_SHARP2 < lejania)
{
paro_motores();
giro_derecha();
valor_actual = 4;
//Serial.println("derecha");
delay(650);
}
if (ADC_SHARP1 > lejania && ADC_SHARP2 > lejania && ADC_SHARP0 < lejania)
{
paro_motores();
giro_izquierda();
valor_actual = 3;
//Serial.println("izquierda");
delay(650);
}
// if (ADC_SHARP1 > lejania && ADC_SHARP2 > lejania && ADC_SHARP0 > lejania)
//  {
//          paro_motores();
//          reversa();
```

```
//          valor_actual = 1;
//          //Serial.println(1);
//          delay(1000);
//          paro_motores();
//          giro_derecha();
//          delay(650);
//          giro_derecha();
//          delay(650);
//  }

}

//FUNCION LECTURA SENSOR SHARP IZQUIERDA
int ADC0_promedio(int n)
{
long suma = 0;
for (int i = 0; i < n; i++)
{
suma = suma + analogRead(A0);
}
return (suma / n);
}
//FUNCION LECTURA SENSOR SHARP CENTRO
int ADC1_promedio(int n)
{
long suma = 0;
for (int i = 0; i < n; i++)
{
suma = suma + analogRead(A1);
}
return (suma / n);
}
```

```
 //FUNCION LECTURA SENSOR SHARP DERECHA
 int ADC2_promedio(int n)
 {
 long suma = 0;
 for (int i = 0; i < n; i++)
 {
 suma = suma + analogRead(A2);
 }
 return (suma / n);
 }

 //FUNCION CONTROL DE MOTORES
void adelante() { //motors fast forward
Serial_2.write(SBT_MOTOR1_FULL_ADELANTE);
Serial_2.write(SBT_MOTOR2_FULL_ADELANTE);
 //Serial.println("adelante");
 }


void reversa() { //motors fast reverse
Serial_2.write(SBT_MOTOR1_FULL_REVERSA);
Serial_2.write(SBT_MOTOR2_FULL_REVERSA);
 //Serial.println("reversa");
 }

void giro_izquierda() {
Serial_2.write(SBT_MOTOR1_FULL_REVERSA);
Serial_2.write(SBT_MOTOR2_FULL_ADELANTE);
 //Serial.println("motor 1 en reversa y motor 2 adelante para girar a la izquierda");
 }

void giro_derecha() {
Serial_2.write(SBT_MOTOR1_FULL_ADELANTE);
```

```
 Serial_2.write(SBT_MOTOR2_FULL_REVERSA);
 // Serial.println("motor 1 adelante y motor 2 en reversa para girar a la derecha");
 }

 void paro_motores() {
 Serial_2.write(SBT_MOTOR1_STOP);  //kill  motors  for  0.5  second
 Serial_2.write(SBT_MOTOR2_STOP); //kill motors for 0.5 second
 //Serial.println("detiene a los motores por un momento");
 delay(500);//cambiar valor de tiempo espera
 }
 void Isr_Timer()
 {
 if (valor_actual != valor_anterior) {
 //Serial.println("Nuevo");

 if (valor_actual == 4 && valor_anterior == 2) { //significa giro derecha o izquierda primera vez
 //Serial.println("DERECHA");



if (val_imp == 2 && bandera1 == 0) {
val_imp = 4;
Serial.println(val_imp);
bandera1++;
 }

if (val_imp == 4 && bandera1 == 0) {
val_imp = 1;
Serial.println(val_imp);
bandera1++;
 }

if (val_imp == 1 && bandera1 == 0) {
val_imp = 3;
```

```
  Serial.println(val_imp);
  bandera1++;
  }
if (val_imp == 3 && bandera1 == 0) {
val_imp = 2;
Serial.println(val_imp);
bandera1++;
  }
  }
  if (valor_actual == 3 && valor_anterior == 2) { //significa giro derecha o izquierda primera vez
  //Serial.println("IZQUIERDA");
if (val_imp == 2 && bandera1 == 0) {
val_imp = 3;
Serial.println(val_imp);
bandera1++;
  }
if (val_imp == 3 && bandera1 == 0) {
 val_imp = 1;
Serial.println(val_imp);
bandera1++;
  }
if (val_imp == 1 && bandera1 == 0) {
val_imp = 4;
Serial.println(val_imp);
bandera1++;
  }
if (val_imp == 4 && bandera1 == 0) {
val_imp = 2;
Serial.println(val_imp)

  bandera1++;
  }
  }
  }
  bandera1 = 0;
  valor_anterior = valor_actual;
  }
```

ANEXO E: CODIGO DE MATLAB

```
clc;
delete(instrfind); %limpia el puerto para comunicacion
%HABILITO EL PUERTO COM PARA LECTURA DE DATOS POR PUERTO SERIAL
comPort = 'COM4';
global obj
if(~exist('serialFlag','var'))
   obj = serial(comPort);
   set(obj,'DataBits',8);
   set(obj,'StopBits',1);
   set(obj,'BaudRate',9600);
   set(obj,'Parity','none');
   fopen(obj);
   fprintf(obj,'%c','a');
   mbox = msgbox('Serial Communication setup'); uiwait(mbox);
   fscanf(obj,'%d')
end

%CONFIGURACION ESQUEMA DE DISEÑO PANTALLA
a=figure('KeyPressFcn',@interrupcion,'name','meausered_data','position',[150,80,1200,700]);  %
xi y xf
a.Color = [1 1 1];
puntaje = uicontrol(a,'Style','text','String','Select a data set.','Position',[1060 650 130 30]);
puntaje.String = 'OBJETOS DETECTADOS';
rampaPos = uicontrol(a,'Style','text','String','Select a data set.','Position',[1060 540 130 100]);
rampaPos.String = 'POSICION ROBOT';
matrizR = uicontrol(a,'Style','text','String','Select a data set.','Position',[1060 430 130 100]);
matrizR.String = 'PROYECTO ROBOT';
tam_cas = 1; %VARIABLE TAMAÑO OBJETO

fwrite(obj, 10, 'uchar'); %INICIO DE ARDUINO

%MAPA CREACION
global axis;
axis = axes;
axis.GridAlpha = 0.2;
```

```
axis.Color = 'cyan';
axis.XLim = [0,50];
axis.YLim = [0,50];

set (axis,'position',[0.05 0.08 0.8 0.88]) %posicion de los ejes cartesianos
grid on
grid minor

%VARIABLES
global posxantes;
global posyantes;
global posxRobot;
global posyRobot;
global cantobstaculo;
cantobstaculo = 0;
global obstaculon;
global randomXn;
global randomYn;
global rastro;
rastro=1;
global cola;
global colan;

posxRobot =java.util.ArrayList();
posyRobot =java.util.ArrayList();
global dx ;
global dy;
%COORDENADAS DE INICIO VERTICAL (ALTERNAR PARA UN INICIO EN HORIZONTAL)
dx = [0 0 -1 1];
dy = [-1 1 0 0];

%POSICION INICIAL ROBOT
posxRobot.add(25);
posyRobot.add(0);
```

```
%POSICION CAJA
global randomX;
global randomY;
randomX = 0;
randomY = 0;

%tamaño max del seguimiento
colan = zeros(1,80); %CAMBIAR ESTE NUMERO PARA LOS CUADRADOS QUE MUESTRAN EL RASTRO
%tamaño max de los obstaculos
obstaculon = zeros(1,10); %CAMBIAR ESTE NUMERO PARA LOS CUADRADOS QUE MUESTRAN LOS OBSTACULOS
randomXn =zeros(1,10);
randomYn =zeros(1,10);
%%matriz tamaño de la cola
for i = 1:length(colan)
   colan(i) = rectangle('Position',[-10 -10 tam_cas tam_cas],'FaceColor', 'black');
end
%%matriz tamaño de los obstaculos
for p = 1:length(obstaculon)
   obstaculon(p) = rectangle('Position',[-10 -10 tam_cas tam_cas],'FaceColor', 'black');
end

global puntos;
puntos  = 0;
global dir;
dir = 2;
global roboTd;
global obstaculo;
global bandera1;
bandera1=2;

%CREO RECTANGULOS PARA ANIMACION
cola = rectangle('Position',[-10 -10 tam_cas tam_cas],'FaceColor', 'r');
obstaculo = rectangle('Position',[-10 -10 tam_cas tam_cas],'FaceColor', 'c');
roboTd = rectangle('Position',[randomX randomY tam_cas tam_cas],'FaceColor', 'r');
```

```
while true
   endAxis();
   %Elimino la posibilidad de leer en vacio al contar los bytes de la cadena antes de leer
   bytes = obj.BytesAvailable;
   if (bytes >= 1)
      dir = fscanf(obj,'%d')
   end

   posxRobot.add(0,(posxRobot.get(0)+dx(dir)));
   posyRobot.add(0,(posyRobot.get(0)+dy(dir)));

   s = num2str(cantobstaculo);

   xrampa = num2str(posxRobot.get(0));
   yrampa = num2str(posyRobot.get(0));

 k = ['Coordenadas Robot',newline,'x = ',xrampa,newline,'y = ',yrampa,newline,...
      newline,'Posicion Obstaculo',newline,'x = ',num2str(randomX),newline,'y =
',num2str(randomY),...
      newline,newline];

   puntaje.String = strcat('Obstaculo = ',s);

   if bandera1~=dir

      if posxantes~=posxRobot.get(0)
         if bandera1==1
            randomX=posxantes;
            randomY=posyantes-1;
         end
         if bandera1==2

            randomX=posxantes;
            randomY=posyantes+1;
         end

         randomXn(cantobstaculo+1)=randomX;
```

```
        randomYn(cantobstaculo+1)=randomY;
        set(obstaculo,'Position',[randomX randomY 1 1],'FaceColor', 'g');

        for m = 1:(cantobstaculo+1)
            set(obstaculon(m),'Position',[ (randomXn(m)) (randomYn(m)) 1 1],'FaceColor', 'g')
        end

        cantobstaculo =cantobstaculo+1;
        puntos = puntos +1;
    end
    if posyantes~=posyRobot.get(0)
        if bandera1==3

            randomX=posxantes-1;
            randomY=posyantes;
        end
        if bandera1==4

            randomX=posxantes+1;
            randomY=posyantes;
        end

        randomXn(cantobstaculo+1)=randomX;
        randomYn(cantobstaculo+1)=randomY;
        set(obstaculo,'Position',[randomX randomY 1 1],'FaceColor', 'g');

        for m = 1:(cantobstaculo+1)
            set(obstaculon(m),'Position',[ (randomXn(m)) (randomYn(m)) 1 1],'FaceColor', 'g')
        end

        cantobstaculo =cantobstaculo+1;
        puntos = puntos +1;
    end

end

set(roboTd,'Position',[(posxRobot.get(0)) (posyRobot.get(0)) 1 1],'FaceColor', 'b');
```

```
    set(cola,'Position',[(posxRobot.get(1)) (posyRobot.get(1)) 1 1],'FaceColor', 'r')
    for l = 2:(rastro)
        set(colan(l),'Position',[ (posxRobot.get(l)) (posyRobot.get(l)) 1 1],'FaceColor', 'r')
    end

    pause(0.5)
    rampaPos.String = strcat(k);
    bandera1=dir;
    posxantes=posxRobot.get(0);
    posyantes=posyRobot.get(0);
    rastro=rastro+1;
end

function endAxis()  %Funcion que permite "recircular" el robot dentro del plano
global posxRobot;
global posyRobot;
global dir

posxRobot.get(0);

if (posxRobot.get(0)==48 && dir == 4)
    posxRobot.add(0,0);
end
if (posxRobot.get(0)==0 && dir ==3)
    posxRobot.add(0,48);
end
if (posyRobot.get(0)== 48 && dir == 2)
    posyRobot.add(0,0);
end
if (posyRobot.get(0)==0 && dir ==1)
    posyRobot.add(0,48);
end

end
```

ANEXO F: ANÁLISIS DEL CENSADO POR PARTE DE LOS SENSORES SHARP

```
void setup() {

// // Comunicación seria a 9600 baudios
Serial.begin(9600);
}
void loop() {

// Leemos el promedio de9 la entrada analógica 0
int ADC_SHARP0 = ADC0_promedio(80);
int ADC_SHARP1 = ADC1_promedio(80);
int ADC_SHARP2=ADC2_promedio(80);
Serial.println("izquierda");
Serial.println(ADC_SHARP0);
Serial.println("delante");
Serial.println(ADC_SHARP1);
Serial.println("derecha");

Serial.println(ADC_SHARP2)
delay(1000);

}
int ADC0_promedio(int n)
{
long suma = 0;
for (int i = 0; i < n; i++)
{
suma = suma + analogRead(A0);
}
return (suma / n);
}
int ADC1_promedio(int n)
{
long suma = 0;
for (int i = 0; i < n; i++)
{
```

```
suma = suma + analogRead(A1);
}
return (suma / n);
}
int ADC2_promedio(int n)
{
long suma = 0;
for (int i = 0; i < n; i++)
{

suma = suma + analogRead(A2);
}
return (suma / n);

}
```

ANEXO G: ANÁLISIS DE LAS DIRECCIONES DEL ROBO

```
int cont = 1;

void setup() {
//inicamos el puerto serie
Serial.begin(9600);
//Serial.println("");
}

void loop() {
Serial.println("2");
delay(8000);
Serial.println("4");
delay(8000);
Serial.println("1");
delay(8000);
Serial.println("3");
delay(8000);
}
```

ANEXO H: TABLA RELACIÓN ENTRE VELOCIDAD Y PWM

	Motor 1		Motor 2	
Valores PWM	**Velocidad adelante (rpm)**	**Velocidad atrás (rpm)**	**Velocidad adelante (rpm)**	**Velocidad atrás (rpm)**
1	225	-	-	-
2	221,4285714	-	-	-
3	217,8571429	-	-	-
4	214,2857143	-	-	-
5	210,7142857	-	-	-
6	207,1428572	-	-	-
7	203,5714286	-	-	-
8	200	-	-	-
9	196,4285714	-	-	-
10	192,8571429	-	-	-
11	189,2857143	-	-	-
12	185,7142857	-	-	-
13	182,1428572	-	-	-
14	178,5714286	-	-	-
15	175	-	-	-
16	171,4285715	-	-	-
17	167,8571429	-	-	-
18	164,2857143	-	-	-
19	160,7142857	-	-	-
20	157,1428572	-	-	-
21	153,5714286	-	-	-
22	150	-	-	-
23	146,4285715	-	-	-
24	142,8571429	-	-	-
25	139,2857143	-	-	-
26	135,7142858	-	-	-
27	132,1428572	-	-	-
28	128,5714286	-	-	-
29	125	-	-	-
30	121,4285715	-	-	-
31	117,8571429	-	-	-
32	114,2857143	-	-	-
33	110,7142858	-	-	-
34	107,1428572	-	-	-
35	103,5714286	-	-	-
36	100,0000001	-	-	-
37	96,42857148	-	-	-
38	92,85714291	-	-	-
39	89,28571434	-	-	-

40	85,71428577	-	-	-
41	82,1428572	-	-	-
42	78,57142863	-	-	-

43	75,00000006	-	-	-
44	71,42857149	-	-	-
45	67,85714292	-	-	-
46	64,28571435	-	-	-
47	60,71428578	-	-	-
48	57,14285721	-	-	-
49	53,57142864	-	-	-
50	50,00000007	-	-	-
51	46,4285715	-	-	-
52	42,85714293	-	-	-
53	39,28571436	-	-	-
54	35,71428579	-	-	-
55	32,14285722	-	-	-
56	28,57142865	-	-	-
57	25,00000008	-	-	-
58	21,42857151	-	-	-
59	17,85714294	-	-	-
60	14,28571437	-	-	-
61	10,7142858	-	-	-
62	7,14285723	-	-	-
63	3,57142866	-	-	-
64	PA RO	-	-	-
65	-	3,57142866	-	-
66	-	7,14285723	-	-
67	-	10,7142858	-	-
68	-	14,28571437	-	-
69	-	17,85714294	-	-
70	-	21,42857151	-	-
71	-	25,00000008	-	-
72	-	28,57142865	-	-
73	-	32,14285722	-	-
74	-	35,71428579	-	-
75	-	39,28571436	-	-
76	-	42,85714293	-	-
77	-	46,4285715	-	-
78	-	50,00000007	-	-
79	-	53,57142864	-	-
80	-	57,14285721	-	-
81	-	60,71428578	-	-
82	-	64,28571435	-	-
83	-	67,85714292	-	-
84	-	71,42857149	-	-
85	-	75,00000006	-	-
86	-	78,57142863	-	-
87	-	82,1428572	-	-
88	-	85,71428577	-	-

89	-	89,28571434	-	-
90	-	92,85714291	-	-
91	-	96,42857148	-	-
92	-	100,0000001	-	-
93	-	103,5714286	-	-
94	-	107,1428572	-	-
95	-	110,7142858	-	-
96	-	114,2857143	-	-
97	-	117,8571429	-	-
98	-	121,4285715	-	-
99	-	125	-	-
100	-	128,5714286	-	-
101	-	132,1428572	-	-
102	-	135,7142858	-	-
103	-	139,2857143	-	-
104	-	142,8571429	-	-
105	-	146,4285715	-	-
106	-	150	-	-
107	-	153,5714286	-	-
108	-	157,1428572	-	-
109	-	160,7142857	-	-
110	-	164,2857143	-	-
111	-	167,8571429	-	-
112	-	171,4285715	-	-
113	-	175	-	-
114	-	178,5714286	-	-
115	-	182,1428572	-	-
116	-	185,7142857	-	-
117	-	189,2857143	-	-
118	-	192,8571429	-	-
119	-	196,4285714	-	-
120	-	200	-	-
121	-	203,5714286	-	-
122	-	207,1428572	-	-
123	-	210,7142857	-	-
124	-	214,2857143	-	-
125	-	217,8571429	-	-
126	-	221,4285714	-	-
127	-	225	-	-
128	-	-	225	-
129	-	-	221,4285714	-
130	-	-	217,8571429	-
131	-	-	214,2857143	-
132	-	-	210,7142857	-
133	-	-	207,1428572	-
134	-	-	203,5714286	-

135	-	-	200	-
136	-	-	196,4285714	-
137	-	-	192,8571429	-
138	-	-	189,2857143	-
139	-	-	185,7142857	-
140	-	-	182,1428572	-
141	-	-	178,5714286	-
142	-	-	175	-
143	-	-	171,4285715	-
144	-	-	167,8571429	-
145	-	-	164,2857143	-
146	-	-	160,7142857	-
147	-	-	157,1428572	-
148	-	-	153,5714286	-
149	-	-	150	-
150	-	-	146,4285715	-
151	-	-	142,8571429	-
152	-	-	139,2857143	-
153	-	-	135,7142858	-
154	-	-	132,1428572	-
155	-	-	128,5714286	-
156	-	-	125	-
157	-	-	121,4285715	-
158	-	-	117,8571429	-
159	-	-	114,2857143	-
160	-	-	110,7142858	-
161	-	-	107,1428572	-
162	-	-	103,5714286	-
163	-	-	100,0000001	-
164	-	-	96,42857148	-
165	-	-	92,85714291	-
166	-	-	89,28571434	-
167	-	-	85,71428577	-
168	-	-	82,1428572	-
169	-	-	78,57142863	-
170	-	-	75,00000006	-
171	-	-	71,42857149	-
172	-	-	67,85714292	-
173	-	-	64,28571435	-
174	-	-	60,71428578	-
175	-	-	57,14285721	-
176	-	-	53,57142864	-
177	-	-	50,00000007	-
178	-	-	46,4285715	-
179	-	-	42,85714293	-
180	-	-	39,28571436	-

181	-	-	35,71428579	-
182	-	-	32,14285722	-
183	-	-	28,57142865	-
184	-	-	25,00000008	-
185	-	-	21,42857151	-
186	-	-	17,85714294	-
187	-	-	14,28571437	-
188	-	-	10,7142858	-
189	-	-	7,14285723	-
190	-	-	3,57142866	-
191	-	-	PA RO	-
192	-	-		3,57142866
193	-	-	-	3,57142866
194	-	-	-	7,14285723
195	-	-	-	10,7142858
196	-	-	-	14,28571437
197	-	-	-	17,85714294
198	-	-	-	21,42857151
199	-	-	-	25,00000008
200	-	-	-	28,57142865
201	-	-	-	32,14285722
202	-	-	-	35,71428579
203	-	-	-	39,28571436
204	-	-	-	42,85714293
205	-	-	-	46,4285715
206	-	-	-	50,00000007
207	-	-	-	53,57142864
208	-	-	-	57,14285721
209	-	-	-	60,71428578
210	-	-	-	64,28571435
211	-	-	-	67,85714292
212	-	-	-	71,42857149
213	-	-	-	75,00000006
214	-	-	-	78,57142863
215	-	-	-	82,1428572
216	-	-	-	85,71428577
217	-	-	-	89,28571434
218	-	-	-	92,85714291
219	-	-	-	96,42857148
220	-	-	-	100,0000001
221	-	-	-	103,5714286
222	-	-	-	107,1428572
223	-	-	-	110,7142858
224	-	-	-	114,2857143
225	-	-	-	117,8571429
226	-	-	-	121,4285715

227	-	-	-	125
228	-	-	-	128,5714286
229	-	-	-	132,1428572
230	-	-	-	135,7142858
231	-	-	-	139,2857143
232	-	-	-	142,8571429
233	-	-	-	146,4285715
234	-	-	-	150
235	-	-	-	153,5714286
236	-	-	-	157,1428572
237	-	-	-	160,7142857
238	-	-	-	164,2857143
239	-	-	-	167,8571429
240	-	-	-	171,4285715
241	-	-	-	175
242	-	-	-	178,5714286
243	-	-	-	182,1428572
244	-	-	-	185,7142857
245	-	-	-	189,2857143
246	-	-	-	192,8571429
247	-	-	-	196,4285714
248	-	-	-	200
249	-	-	-	203,5714286
250	-	-	-	207,1428572
251	-	-	-	210,7142857
252	-	-	-	214,2857143
253	-	-	-	217,8571429
254	-	-	-	221,4285714
255	-	-	-	225

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 09 / 11 / 2021

INFORMACIÓN DEL AUTOR/A (S)
Nombres – Apellidos: BRYAN ISRAEL NÚÑEZ SÁNCHEZ
INFORMACIÓN INSTITUCIONAL
Facultad: INFORMÁTICA Y ELECTRÓNICA
Carrera: ELECTRÓNICA Y AUTOMATIZACIÓN
Título a optar: INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN
f. Analista de Biblioteca responsable: Lcdo. Holger Ramos, MSc.

Firmado electrónicamente por:
HOLGER GERMAN RAMOS UVIDIA

1862-DBRA-UPT-2021