Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería Electromecánica

‘‘DISEÑO Y CONSTRUCCIÓN DE UN MICRO ROBOT PARA RESOLUCIÓN DE LABERINTOS.”

Trabajo de Titulación, previo a la obtención del título de Ingeniero Electromecánico

AUTOR:

Noe Ronaldo Puglla Morocho

DIRECTOR:

Ing. Jefferson Fernando Camacho Muñoz, Mg. Sc.

Loja-Ecuador

2022

Certificación

Loja, 29 de agosto del 2022

Ing. Jefferson Fernando Camacho Muñoz, Mg. Sc.

DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICO:

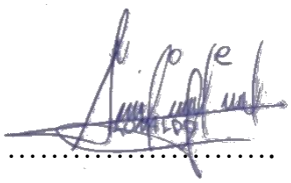
Que he revisado y orientado todo el proceso de elaboración del Trabajo de Titulación denominado: **"DISEÑO Y CONSTRUCCIÓN DE UN MICRO ROBOT PARA RESOLUCIÓN DE LABERINTOS",** previo a la obtención del título de **Ingeniero en Electromecánica,** de la autoría del estudiante **Noe Ronaldo Puglla Morocho**, con **cédula de identidad** Nro.**1150339248,** una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Ing. Jefferson Fernando Camacho Muñoz, Mg. Sc.

DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Noe Ronaldo Puglla Morocho**, declaro ser autor del presente Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Titulación, en el Repositorio Digital Institucional- Biblioteca Virtual.



Firma: ……

Cédula de identidad: 1150339248

Fecha: 27 de octubre de 2022.

Correo electrónico: noe.puglla@unl.edu.ec

Celular:0983224471

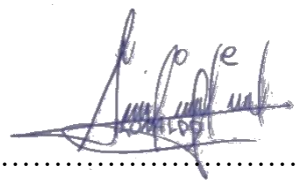
Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Titulación.

Yo, **Noe Ronaldo Puglla Morocho**, declaro ser autor del Trabajo de Titulación denominado: "**DISEÑO Y CONSTRUCCIÓN DE UN MICRO ROBOT PARA RESOLUCIÓN DE LABERINTOS",** como requisito para optar por el título de **Ingeniero electromecánico,** autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los veintiséis días del mes de octubre del dos mil veintidós.



Firma: …….

Autor: Noe Ronaldo Puglla Morocho

Cédula: 1150339248

Dirección: Selva Alegre - Saraguro

Correo electrónico: noe.puglla@unl.edu.ec

Celular: 0983224471

DATOS COMPLEMENTARIOS:

Director del Trabajo de Titulación: Ing. Jefferson Fernando Camacho Muñoz, Mg. Sc.

Dedicatoria

Quiero dedicar este proyecto de tesis de corazón a Dios, que ha sido mi guía y fortaleza durante este proceso educativo y permitirme haber llegado a este momento tan importante y especial de mi formación profesional; además de mantenerme con salud, fuerza y paz para llevar a cabo mis metas y objetivos a futuro.

A mis padres Luz Piedad y Cosme Damián, por ser ese pilar incondicional para la construcción de mi vida profesional, por siempre darme su fuerza y apoyo incondicional quienes, con su amor y cariño, me han permitido cumplir un sueño más, por ello quero expresa que ¡ustedes son mi más preciado tesoro!

A mis hermanos, Maricela, Tatiana, Karina, Henry, Talía y Sindy, por ser esa fuente de motivación durante toda mi vida, que de una u otra manera me acompañan en todos mis sueños y metas que he conseguido hasta la fecha.

Noe Ronaldo Puglla

Agradecimiento

De manera especial y exclusiva quiero agradecer a mi tutor, el Ing. Jefferson Fernando Camacho Muñoz, Mg. Sc., director del presente Trabajo de Titulación por haberme ayudado y guiado en cada paso, en cada dificultad que se ha presentado a lo largo de este trabajo investigativo y práctico, por su paciencia, buena voluntad y dedicación en la ejecución de este trabajo, demostrando sus vastos conocimientos que lo destacan como gran docente y excelente persona.

De igual manera quiero expresar mi gratitud a todas las autoridades, personal administrativo y cuerpo docente de la carrera de Ingeniería Electromecánica, Facultad de la Energía, Las Industrias y los Recursos Naturales no Renovables de la Universidad Nacional de Loja por impartirme una educación de calidad y brindarme sus conocimientos a lo largo de mi estancia dentro de este prestigioso centro de educación superior, que permitieron el desarrollo de mis estudios, hoy plasmados en el cumplimiento de esta meta.

Por último, agradezco a mis padres que merecen un reconocimiento especial por todo el apoyo moral e incondicional brindado, gracias por darme la libertad de desenvolverme como ser humano, orgullosamente en estas líneas expreso mi gratitud a mi mayor inspiración, mi madre, por hacer posible la ejecución de este proyecto mediante su apoyo económico brindado

Noe Ronaldo Puglla

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de Anexos:

1. Título

‘‘DISEÑO Y CONSTRUCCIÓN DE UN MICRO ROBOT PARA RESOLUCIÓN DE LABERINTOS.”

2. Resumen

El presente proyecto de investigación refiere al diseño y construcción de un prototipo de micro robot para la solución de laberintos, capaz de navegar y atravesar laberintos desconocidos utilizando tres sensores de distancia.

Por tal motivo, se procederá a realizar una revisión de los algoritmos de solución de laberintos comúnmente utilizados en la robótica móvil, con el fin de analizar y seleccionar cuales son los mejores en la implementación del presente proyecto, para ello se utilizaron diversos softwares con el objetivo de ejecutar una simulación de los diferentes algoritmos que se emplean para solucionar laberintos como es el caso del software Karel y Tinkercad, los algoritmos que se simularon son el algoritmo de la mano derecha, algoritmo de treumax, algoritmo aleatorio, algoritmo de la mano izquierda.

El diseño del robot fue inspirado para que tenga las dimensiones más pequeñas posibles aprovechando todos los espacios que se va a utilizar en el robot, para ello se realizará el diseño de chasis y ensamblaje de todos los componentes a través del software CAD SolidWorks, la programación de dicho robot será a través del IDE del software de arduino, además se realizará una selección de los componentes que son necesarios para el funcionamiento del robot.

El diseño y construcción del laberinto está enfocado con el propósito de que se pueda modificar sus rutas o caminos de tal manera que este laberinto sea utilizado para realizar diferentes pruebas al robot en la toma de decisiones. Finalmente se procede a validar la construcción y el diseño del micro robot sometiéndolo a pruebas de toma de decisiones dentro del laberinto construido.

Palabras claves: Robótica móvil, robot laberinto, diseño, microcontroladores, autónomo, algoritmos, Arduino, SolidWorks.

2.1. Abstract

This research project refers to the design and construction of a micro robot for maze solving prototype, capable of navigating and traversing unknown mazes using three distance sensors.

For this reason, a review of the solving labyrinths algorithms commonly used in mobile robotics will be carried out, in order to analyze and select which are the best in the implementation in our robot, for this, various software's were used in order to run a simulation of different algorithms used to solve labyrinths as is the case with Karel and Tinkercad software, the algorithms that are simulated are the right-hand algorithm, treumax algorithm, random algorithm, left hand algorithm.

The robot design was inspired to have the smallest possible dimensions taking advantage of all the spaces that are going to be used in the robot, for this the design of chassis and assembly of all components through SolidWorks CAD software, the programming of said robot will be through the IDE of the Arduino software, in addition it will be carried out a selection of the components that are necessary for the robot operation.

The design and construction of the labyrinth is focused with the purpose that its routes or paths can be modified in this way that this labyrinth serves us to carry out different tests to the robot in decision making.

Finally, the construction and design of the micro robot subjecting it to decision-making tests within the constructed labyrinth.

Key words: Mobile robotics, labyrinth robot, design, microcontrollers, autonomous, algorithms, Arduino, SolidWorks.

3. Introducción

Actualmente, en múltiples áreas como la robótica o la inteligencia artificial los avances científicos y tecnológicos son cada vez más apreciables e impresionantes. La automatización, implementación y fabricación de robots móviles son actividades que han crecido de forma significativa, muchos de ellos son usados hoy en día para diversas tareas, utilizando un suficiente nivel de inteligencia que les permita desenvolverse autónomamente en terrenos desconocidos y no programables.

La robótica móvil se aprovecha para dar solución a problemas de navegación sobre entornos peligros o de inaccesibilidad al ser humano, los robots destinados a la navegación están adquiriendo una gran demanda en el área comercial, con múltiples aplicaciones en distintas áreas de trabajo, pero en la robótica móvil existe una categoría denominada "Robot Resuelve Laberintos", es una de las más desafiantes e interesantes, debido a que se necesita crear un prototipo capaz de resolver un laberinto en el menor tiempo posible.

Para que un robot móvil autómata sea capaz de resolver un laberinto desconocido, se emplea uno de los conceptos utilizados en la robótica como es la toma de decisiones, que consiste en la acción que ejecutará un robot cuando éste se encuentre en un punto de decisión, esta decisión dependerá del algoritmo que se esté utilizando.

Para el desarrollo de este proyecto se inicia con una revisión bibliográfica en la cual se investiga sobre temas como la robótica móvil, sensores de distancia, hardware y software de arduino, redes neuronales, métodos para solución de laberintos, entre otros.

En la segunda parte se describe los algoritmos para generar laberintos como son el de aldous-broder y backtrack, igualmente los algoritmos para solucionar laberintos usando un robot móvil como son el de la mano derecha, dijkstra, optimización.

En la tercera parte se describe los materiales que se pueden implementar, la parte estructural de dicho robot, el cual servirá de bastidor para concentrar todos los componentes donde se buscará optimizar lo máximo posible las medidas y contornos del robot. Igualmente se describe los métodos utilizados en el que se analizarán, detallarán y justificarán todos los pasos seguidos hasta la obtención del modelo final del robot, así como las diferentes posibilidades ensayadas, tanto en la elaboración del código, como en la construcción física del robot y del laberinto.

En la cuarta parte se muestran los resultados, discusión, conclusiones, recomendaciones que permiten asimilar y evaluar el desarrollo del proyecto de acuerdo a los objetivos planteados.

Finalmente, se encuentra también la sección de anexos que contiene el conjunto de planos del diseño de la estructura del prototipo, así como información importante sobre los materiales utilizados para el desarrollo del mismo.

-Objetivo General

Diseñar y construir un micro robot con capacidad de aprendizaje para navegar y atravesar un laberinto desconocido.

-Objetivos Específicos

- Implementar un modelo matemático para la resolución de los laberintos.
- Validar el algoritmo utilizado por el micro robot.
- Diseñar y Construir un entorno de pruebas (Laberinto Reconfigurable).
- Analizar los costos y resultados finales obtenidos.

4. Marco Teórico

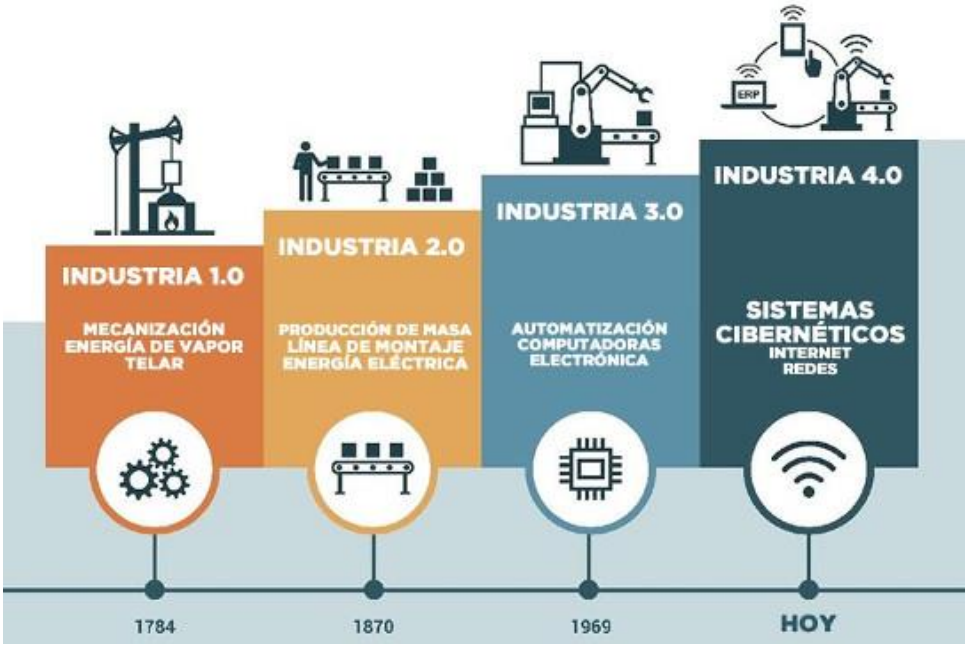
4.1. Industria 4.0

La industria 4.0 también conocida como la cuarta revolución industrial, ver figura 1, combina técnicas avanzadas de producción y operación con tecnologías inteligentes como la inteligencia artificial, la nanotecnología, la robótica, el internet de las cosas (IOT), entre otros (Val Román, 2016).

Durante la primera revolución industrial, se mecanizaron los procesos de producción. La segunda transición o evolución trajo consigo la producción en serie, que facilita la fabricación de productos para el gran consumo. La tercera transición trajo el despliegue de la electrónica y la informática en los procesos industriales permitió automatizar las líneas de producción y la cuarta revolución industrial, utiliza tecnologías anteriormente descritas para combinarlas y dar paso al internet de las cosas (IOT) (Berger, 2019).

Figura 1

La Cuarta Revolución Industrial



Nota. Evolución a partir de la industria 1.0 hasta la industria 4.0. Tomada de (Berger, 2019)

4.1.1. Tecnologías en que se Sustenta la Industria 4.0

De acuerdo a (Val Romàn, 2016) existen muchas tecnologías que sustentan la industria 4.0 pero las principales son las siguientes:

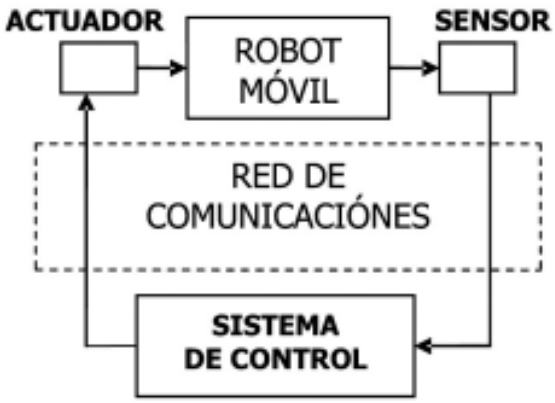
- Big data and analytics: consiste en el análisis de conjuntos de datos que, por su volumen, y velocidad a la que tienen que ser procesadas, ayudan a tomar decisiones en tiempo real.
- Robots autónomos: se están volviendo cada vez más flexibles y cooperativos, de forma que podrán interactuar entre ellos y trabajar de forma segura junto a los humanos.
- Simulación: actualmente las simulaciones en 3D están extendidos en la fase de ingeniería, se utilizarán también en algunas operaciones en las plantas de producción.
- Internet de las cosas (Internet of things, IoT): cada vez más dispositivos estarán enriquecidos con informática incrustada y conectados por medio de tecnologías estándar.
- La nube: comprende aplicaciones e infraestructuras ofrecidas como servicio por redes públicas o privadas y conseguirán tiempo de reacción de apenas algunos milisegundos.
- Realidad aumentada: un operario con gafas de realidad aumentada puede, por ejemplo, recibir instrucciones de reparación de una máquina en el propio puesto de trabajo.

4.2. Robótica Móvil

El objetivo imprescindible de los robots móviles es el desplazamiento en un ambiente conocido o desconocido. Es por ello que de manera precisa o relativa es necesario conocer la posición del robot en su universo. El desarrollo de los robots móviles se trata de incrementar la autonomía limitando todo lo posible la intervención humana. El diseño y construcción de un robot móvil implica conocimientos en mecánica, electrónica, programación, control de sistemas, entre otras, en la figura 2 se muestra el esquema general de un robot móvil y el proceso que conlleva su creación (Aguilar Castillo & Enriquez Astudillo , 2008).

Figura 2

Esquema básico de un robot móvil



Nota. Obtenido de (Suarez et. al, 2006)

Los robots móviles tienen como precedentes los dispositivos electromecánicos tales como los denominados “micro-mouse” para desarrollar funciones inteligentes como descubrir caminos en laberintos, logrando de esta manera la navegación autónoma de forma eficiente. La autonomía de un robot móvil se basa en el sistema de navegación automática que incluyen tareas de planificación, percepción y control (Baturone, 2001).

4.2.1. Funciones Principales de un Robot Móvil

- La locomoción: es la acción y efecto que tiene el robot de moverse de un punto a otro, se toma en consideración los motores y sus mecanismos (Rus, 2019).
- La percepción: mediante el uso de sensores, en este punto el robot puede distinguir el entorno en que se está desplazando. De esta manera, el robot construye un modelo de su entorno a medida que avanza en su exploración (Rus, 2019).
- La decisión: los datos que provienen de los diferentes sensores, deben ser interpretados para la toma de decisiones sobre la acción que hay que llevar a cabo, siendo el objetivo dar las órdenes correctas a los actuadores (Rus, 2019).

4.3. Inteligencia Artificial

De acuerdo a (Ocampo & Catarina, 2018) “la inteligencia artificial es una rama de las ciencias computacionales que se encarga del diseño y también de la construcción de múltiples sistemas idóneos de realizar tareas asociadas con la inteligencia humana”. Un ordenador es una mente. Pero con frecuencia los circuitos son distintos a los del cerebro, produciendo resultados semejantes a la conducta humana, al momento que estos circuitos y programas se ejecutan, hacen que la máquina piense igual que la mente cuando procesa la información.

4.3.1. Clasificación de la Inteligencia Artificial

De acuerdo a (Ponce et. al, 2014), la inteligencia artificial se las puede clasificar en enfoque simbólico y enfoque sub-simbólico.

4.3.1.1. Enfoque Sub-simbólico. Este enfoque se caracteriza principalmente por crear sistemas con capacidad de aprendizaje, se puede obtener a nivel de individuo imitando el cerebro humano (redes neuronales).

4.3.1.2. Enfoque Simbólico. En este enfoque se utiliza representaciones simbólicas basadas en un número finito de primitivas y de reglas para la manipulación de símbolos como redes semánticas, lógica de predicados, etc**.**

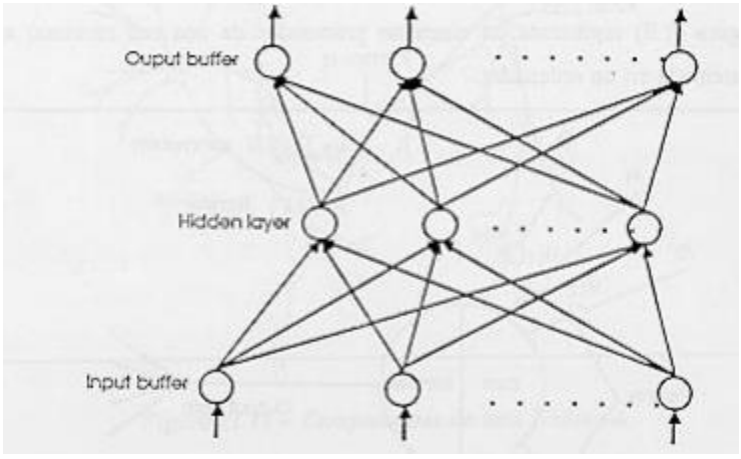
4.3.2. Redes Neuronales

Las redes neuronales aspiran a tener la capacidad del cerebro humano de pensar, recordar y resolver problemas, esto ha inspirado a muchos científicos intentar o procurar modelar en el ordenador el funcionamiento del cerebro humano, inculcando de esta manera la creación de las redes neuronales artificiales, ANN (Artificial Neural Networks). Están formadas por elementos que se comportan de forma similar a la neurona biológica, estos elementos están organizados de una manera parecida a la que presenta el cerebro humano (Atria Innovation, 2019).

En la siguiente figura, se muestra un esquema general de una red neuronal, en el ejemplo cada circulo representa una experiencia que ha tenido el sistema en el momento de su aprendizaje, dicha experiencia queda registrada y cada vez que vuelva a ocurrir un acontecimiento idéntico o muy similar a lo ocurrido, el sistema sabrá cómo reaccionar gracias a la práctica previamente obtenida (Basogain, 2008).

Figura 3

Arquitectura de una red neuronal



Nota. Obtenido de (Basogain, 2008)

Las redes neuronales artificiales al margen de "parecerse" al cerebro humano presentan una serie de particularidades propias del cerebro tales como:

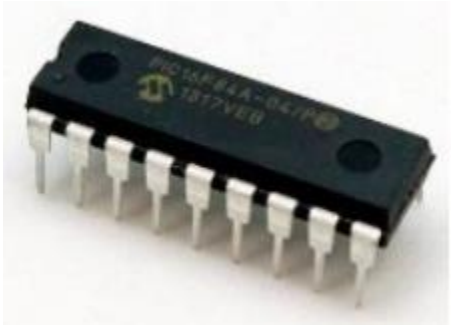
- Aprender: En este punto se adquiere el conocimiento de una cosa por medio del estudio, ejercicio o experiencia. Las ANN cambian su actuación en función del entorno. Se les muestra un conjunto de entradas y ellas se ajustan para dar origen a salidas consistentes (Basogain, 2008).
- Generalizar: extender o ampliar una cosa, estas redes pueden ofrecer dentro de un margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos de ruido o distorsión (Basogain, 2008).
- Abstraer: aislar mentalmente o pensar por separado las cualidades de un objeto. Algunas ANN tienen la capacidad de abstraer la esencia de un conjunto de entradas que supuestamente no presentan aspectos comunes o relativos (Basogain, 2008).

4.4. Sistemas Microcontroladores

Hoy en día las necesidades del ser humano en su búsqueda por hacer la vida más sencilla, dio lugar a una mejora de estos sistemas digitales. Es aquí donde nace el microcontrolador, que es un conjunto de elementos dentro de un circuito integrado, como se presenta en la figura 4 (Apaza Condori, 2017).

Figura 4

Microcontrolador de mucha aplicación en la robótica



Nota. Obtenido de (Apaza Condori, 2017)

La función principal de los microcontroladores es administrar sus labores. Igualmente, sus memorias están internamente dentro de un mismo circuito. Económicamente hablando los microcontroladores son mucho más accesibles que un microprocesador, presenta bajos niveles de interferencia debido a su arquitectura (Apaza Condori, 2017).

En un sistema microcontrolador es importante hablar sobre la memoria que se la puede dividir en dos grupos: la externa y la interna. En la memoria interna se almacenan los datos que no se pierden al desconectar la fuente; en la memoria externa pasa lo contrario, pues se almacenan los datos que si pueden ser borrados al conectarse a la fuente (Apaza Condori, 2017).

4.4.1. Arduino

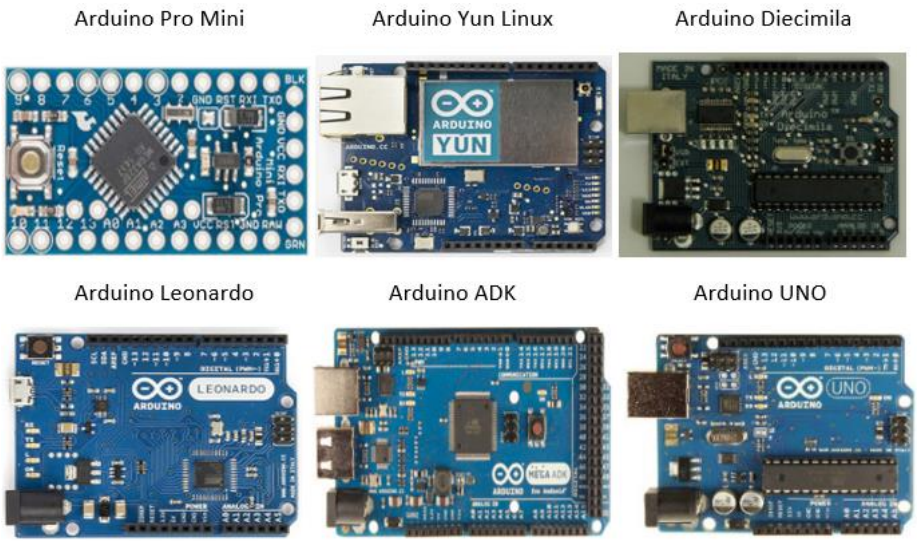
Arduino es una empresa de desarrollo de hardware, que asume como objetivo primordial el diseño y producción de circuitos electrónicos en circuitos impresos que reúnen un microcontrolador y el entorno de desarrollo para ejecutar la programación de cada placa de forma sencilla (Vicuña Novillo et. al, 2018).

4.4.1.1. Hardware de Arduino. Este hardware además de contar con un microcontrolador, cuenta con puertos de comunicación, puertos de entrada-salida digitales y analógicos, un pin de alimentación, pines de tierra, un regulador de voltaje, un puerto jack de alimentación, un puerto USB, además de un botón de reset, entre otros.

Arduino cuenta con un catálogo muy diverso de placas con microprocesadores y shields que facilitan la conexión de los circuitos necesarios para el desarrollo de los proyectos que el usuario requiera crear, en la figura 5 se muestra algunas placas de arduino disponibles que se las puede utilizar.

Figura 5

Diferentes modelos de placas arduino



Nota. Elaboración propia

4.4.1.2. Software de Arduino. Para iniciar con la programación de la placa arduino es necesario descargar un IDE (Integrated Development Environment). El IDE es un grupo de herramientas de software que facilitan a los programadores desarrollar y grabar todo el código, preciso para hacer que nuestro arduino funcione como queramos. Además, el IDE de arduino nos permite escribir, depurar, editar y grabar nuestro programa llamados "sketches" de una manera sencilla **(**Arduino**,** 2020**)**.

Las herramientas que más se utiliza cuando se elabora un programa en el IDE de arduino se describen a continuación:

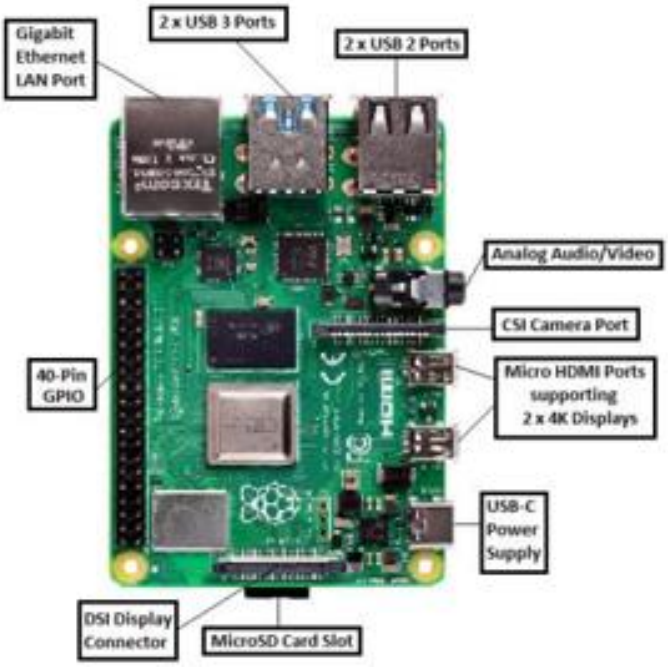
- **Verificar:** Este botón ejecuta dos funciones: prueba que no haya errores en nuestro código, y si no hay inconvenientes, lo compila.
- **Subir:** Este botón se lo utiliza después de "Verificar". Su función es cargar en la memoria del microcontrolador el programa que hemos desarrollado.
- **Guardar:** Esta opción guarda el código de nuestro sketch en un fichero, el cual tendrá la extensión ". ino".
- **Monitor Serial:** Esto nos permite ver valores o información transmitida y enviada desde nuestra placa Arduino por el puerto de comunicación serial.

4.4.2. Raspberry Pi

Estas placas microcontroladoras fueron desarrolladas por la empresa Raspberry Pi Foundation en el año 2012, la raspberry pi es un mini ordenador del tamaño de una tarjeta de crédito, que contiene un procesador de 4 núcleos, conexiones RAM, USB, ethernet y HDMI, también está equipada con un módulo Wifi y bluetooth, en la figura 6 se muestran las principales características de esta placa microcontroladora (Solectro, 2020).

Figura 6

Características de una placa Raspberry Pi



Nota. Obtenido de (Solectro, 2020)

La Rasberry Pi tiene incalculables usos y es sencillo de instalar en la mayoría del software Linux, por consiguiente, permite la codificación en varios lenguajes como Python y sobre todo C++. Un raspberry pi consigue hacer cualquier cosa que realice un ordenador más grande y de mayor consumo, no obstante, puede que no lo haga tan rápido. Sin embargo, todos los modelos de raspberry pi tienen una cosa en común: son compatibles, lo que implica que el software escrito para un modelo funcionará o ejecutará en cualquier otro modelo (Solectro, 2020).

4.4.3. Placa ESP 32

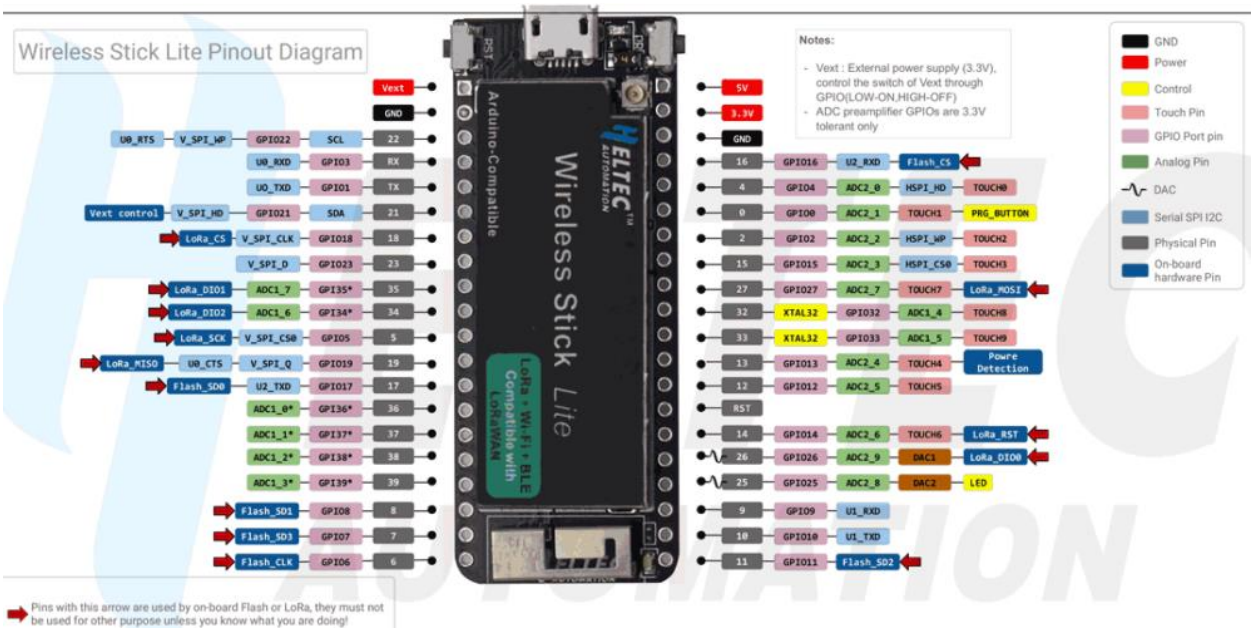
Las placas ESP32 son una familia de microcontroladores de la empresa Espressif Systems, el ESP32 en el mercado ecuatoriano son un poco costosas, esto no es de extrañar teniendo en cuenta las ventajas que conserva sobre otros microcontroladores. En las especificaciones técnicas mostradas en la figura 7 encontramos, que tiene un procesador de 32 bits, tiene una velocidad máxima de 260 MHz, tiene 36 pines GPIO; de los cuales 16 se pueden manejar como salidas PWM, 18 pines pueden portarse como entradas analógicas. Para manejar los puertos analógicos utiliza 2 ADC de entrada múltiple. Sus convertidores son de 12 bits, característica que le concede una mayor resolución para leer las señales analógicas (Guerra Carmenate, 2021).

Sus aplicaciones representativas se hallan en el llamado ‘Internet of Things’, IoT, en resumen, la supervisión y el control remoto de dispositivos ya sea por medio de una red local, LAN

o remotamente operando como servidores de internet (web server), para actividades de robótica y domótica. Los lenguajes de programación más usados para la placa ESP32, son micro Python y especialmente el IDE de arduino, ambos son de código abierto y cuentan con herramientas de desarrollo gratuitas.

Figura 7

Distribución de pines de la placa ESP32



Nota. Obtenido de (Guerra Carmenate, 2021)

Como se puede apreciar en la figura 7, la placa cuenta con tres pines destinados a la alimentación: vext, 5V y 3.3V. Igualmente, posee dos pines de tierra (GND) y un pin de reinicio (RST). Esta placa microcontroladora deja un total de 34 pines digitales que se pueden utilizar en los proyectos que se desee trabajar.

De todos los 34 pines digitales 18 pueden ser manipulados como entradas analógicas, por consiguiente, es dable agregar un buen número de sensores analógicos a los posibles proyectos. Todos los pines analógicos están marcados de color verde en la imagen anterior (Guerra Carmenate, 2021).

4.4.4. Teensy

Teensy figura 8, es una placa microcontroladora muy versátil, compatible con arduino, y de un tamaño pequeño para poderla emplear en proyectos en los que el tamaño importa. Teensy dispone de distintos modelos o versiones, en las que varían algunas prestaciones y sobre todo su

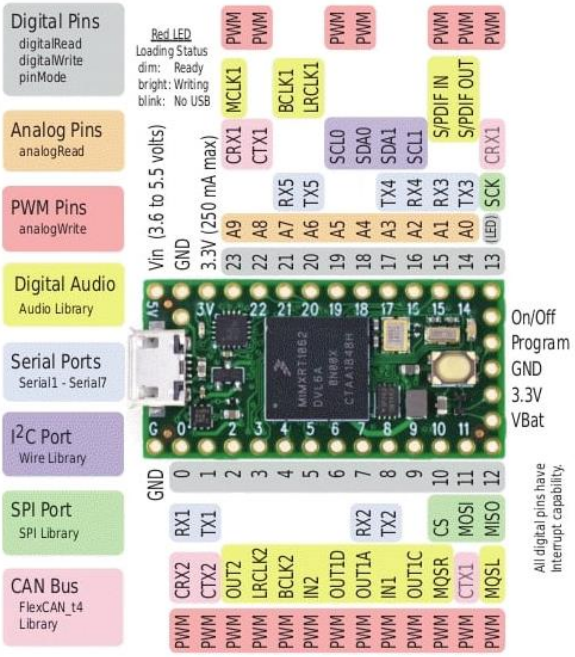
tamaño, al mismo tiempo de tener el respaldo de gran cantidad de bibliotecas de software, lista para ejecutarse con arduino IDE (Electronics Projects, 2018).

Con el fin de tener una visión algo más genérica de Teensy a continuación están algunas de sus características técnicas:

- Compatibilidad con arduino y librerías.
- Puerto USB cualquier tipo de dispositivo.
- Software de desarrollo libre.
- Soporte multiplataforma, funciona para sistemas operativos Linux, MacOS y Windows.
- Tamaño pequeño, apto para muchos proyectos.
- Disponible con pines soldados para placa de pruebas o sin ellos.
- Programación de pulsador único.
- Depuración USB.

Figura 8

Distribución de pines de una placa Teensy



Nota. Obtenido de (Electronics Projects, 2018)

4.5. Actuadores

Los actuadores son elementos o mecanismos que maniobra el robot para reaccionar a los estímulos que generan los sensores, en otras palabras, son los elementos finales de control, en

robótica entre los actuadores más importantes se tienen los siguientes: motores, electro válvulas, pinzas, brazos, etc. (Vildósola, 2008).

4.5.1. Motores

Es el medio simple y básico de movimiento que se utiliza en la electrónica, hay diversos tipos de motores que se verán a continuación, como son: los servomotores, los motores paso a paso y los motores de corriente continua (DC).

4.5.1.1. Motores de Corriente Continua (DC). Estos motores DC son los que más se utilizan actualmente, sobre todo gracias a su gran versatilidad y la facilidad de control que ofrecen como se observa en la figura 9. En el motor se aglomera un sensor destinado para la posición que se encarga de ejecutar y hacer el control sobre la acción del actuador eléctrico (Dièquez, 2020).

Figura 9

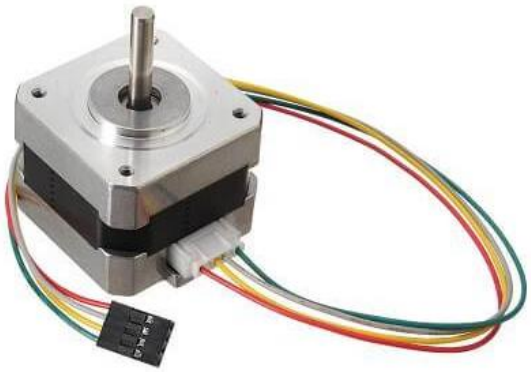
Motor de Corriente Continua

Nota. Obtenido de (Dièquez, 2020)

4.5.1.2. Motores Paso a Paso. Son otro tipo de motor muy común en el área de la robótica como se muestra en la figura 10, los motores paso a paso son un tipo de motor de CC sin escobillas, pueden girar y pararse con una precisión y exactitud del orden de centésimas de milímetro. Por esta alta exactitud y su fiabilidad los destinan para ser utilizados en una gran cantidad de aparatos electrónicos como, por ejemplo, discos, impresoras, fotocopiadoras, y robots, por último, estos motores son más complicados de manejar que los servos, existen dos tipos principales de motores paso a paso: unipolar y bipolar (Dièquez, 2020).

Figura 10

Motor Paso a Paso

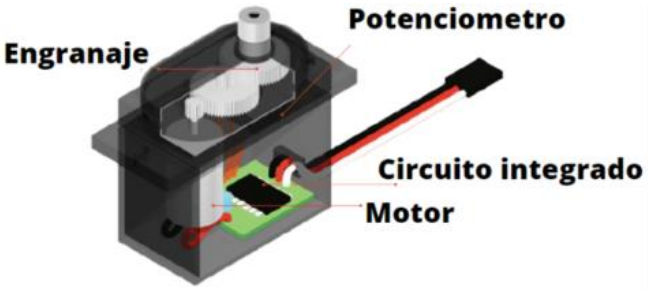


Nota. Obtenido de (Dièquez, 2020)

4.5.1.3. Servomotor. Un servomotor tiene tres cables como se puede observar en la figura 11. Por lo cual un cable está destinado para la alimentación de voltaje, otro cable para tierra y el tercero es un cable de control. Dado que un servo es un sistema de retroalimentación cerrado el cable de control es necesario para detectar la posición del eje del servo y ajustarlo si es necesario. Los servos sólo giran en un ángulo delimitado de grados que se le indique por lo que no giran continuamente (Dièquez, 2020).

Figura 11

Componentes de un Servomotor



Nota. Obtenido de (Dièquez, 2020)

4.6. Modulación por Ancho de Pulso (PWM)

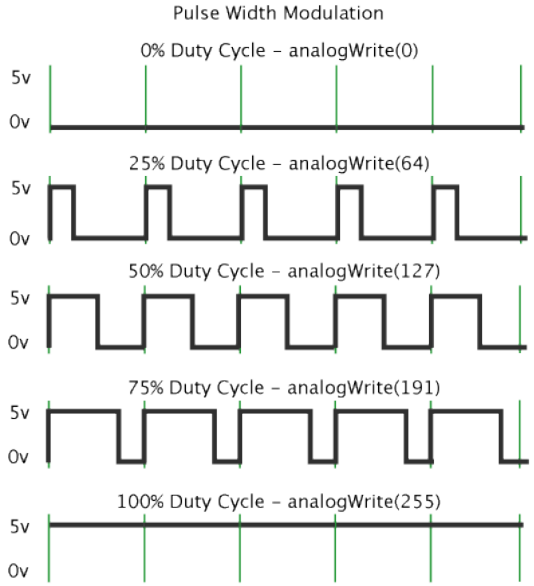
La modulación por ancho de pulso, o simplemente PWM (pulse width modulation) sirve para reducir la velocidad del motor en unas condiciones de carga determinadas, se consigue reduciendo la tensión de alimentación del motor, de manera que para controlar la velocidad del robot móvil habrá que controlar el valor de la tensión de los motores (Dìaz, Garcìa, & Rìos, 2009).

La frecuencia se gobierna o controla con pulsos positivos en el transcurso de medio período y luego con pulsos negativos durante el siguiente medio período. En dicho método el motor se alimenta con una tensión que, en lugar de ser continua todo el tiempo, conmuta entre dos estados

(encendido y apagado) de manera que si las transiciones se realizan lo suficientemente el motor no tiene tiempo de reducir su velocidad a cero. En la práctica la tensión de alimentación está constituida por un tren de pulsos cuadrados como se puede apreciar en la figura 12, en el que siempre se mantiene el valor máximo (amplitud de pulso) pero se controla el ancho de los pulsos (Abellàn, 2016).

Figura 12

Características de una señal PWM



Nota. Obtenido de (Abellàn, 2016)

4.7. Sistema Sensorial del Robot

Para el presente proyecto de tesis los sensores son una parte primordial, debido a que estos permiten relacionarse con el medio y despachar información al cerebro del robot para que consiga realizar acciones, en este caso se utilizarán para censar las paredes del laberinto tanto frontales como laterales sea izquierda o derecha y así evadirlas o tomar vías alternativas. a continuación, se puntualizan algunos tipos de sensores.

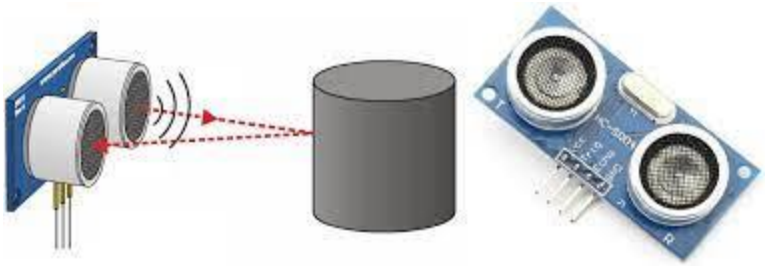
4.7.1. Sensores Ultrasónicos

Los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas es decir son vibraciones del aire de la misma naturaleza que el sonido audible, pero a una frecuencia más elevada que parte de 20 Khz. hasta $5x10^8$ Hz no audibles estos por el oído humano, El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto como se muestra

en la figura 13. Los sensores ultrasónicos calculan la distancia al objeto contando el tiempo entre la emisión y la recepción de la onda ultrasónica (Keyence, 2019).

Figura 13

Sensor ultrasónico



Nota. Tomado de (Keyence, 2019)

En la estructura de un sensor ultrasónico característico el elemento primordial es un transductor electroacústico, frecuentemente del tipo cerámico piezoeléctrico. Sus aplicaciones son muy diversas entre ellas tenemos la modelación del medio ambiente, detección de movimiento, medición de distancias, detección de obstáculos.

4.7.2. Sensores Infrarrojos

En la figura 14 se muestran este tipo de sensores que están diseñados fundamentalmente para la detección, clasificación y posicionado de objetos; la localización de formas, colores y contrastes de superficies, incluso bajo condiciones ambientales extremas.

Este tipo de sensores, se encuentra basado principalmente en la emisión de luz, y en la localización de esta emisión realizada por los foto-detectores. Los infrarrojos tienen restricciones ya que pueden recoger muchas interferencias tanto de luz ambiente sobre todo si son de tipo neón, como por fuentes de calor. Según la forma en que se origine esta emisión y detección de luz, se puede clasificar este tipo de captadores en: captadores por barrera y captadores por reflexión (Mardones & Gonzàlez, 2019).

Figura 14

Sensor Infrarrojo

Nota. Obtenido de (Mardones & Gonzàlez, 2019)

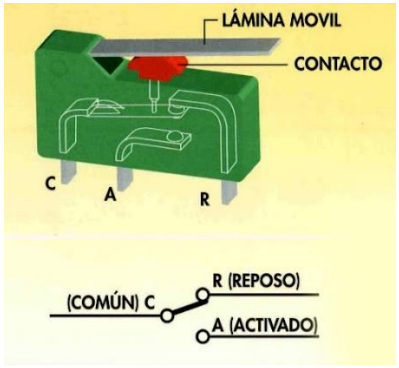
4.7.3. Sensores de Contacto

Estos sensores son los dispositivos más simples y fáciles de usar de todos los sensores que podemos encontrar, debido a que son interruptores que se activan o desactivan si se encuentran en contacto con un objeto, por lo que de este modo se reconoce la presencia de un objeto en un determinado lugar.

Son dispositivos que muestran variación en una de sus características físicas, únicamente cuando se origina un contacto de algún objeto sólido con ellos mismos. Suelen ser empleados en los extremos de los brazos de robot (pinzas) para controlar la manipulación de objetos, asimismo hay que decir que este tipo de sensores se usan primordialmente para impedir daños en el robot ante cualquier posible choque. A continuación, en la figura 15 se muestra un esquema básico de cómo es el sensor de contacto (Olmeda, 2016).

Figura 15

Sensor de Contacto



Nota. Obtenido de (Olmeda, 2016)

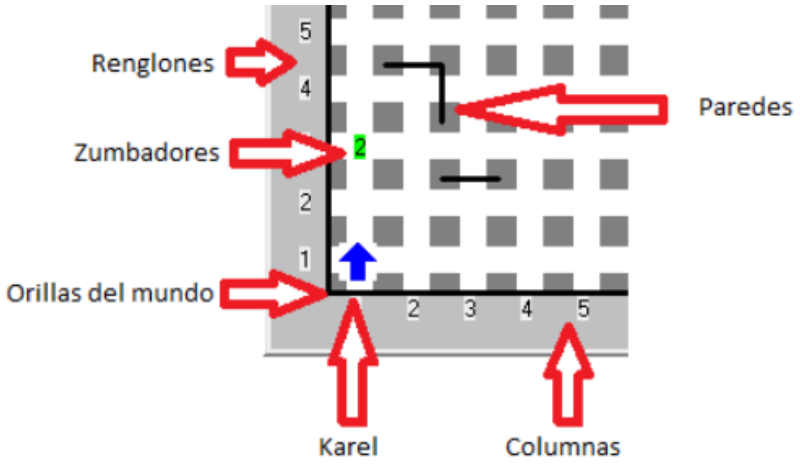
4.8. Software Karel

Karel es un programa que simula un mundo virtual de un robot en forma de flecha como se aprecia en la figura 16, donde el objetivo es la solución de inconvenientes de carácter logístico. El

robot se llama Karel y lo conseguimos controlar por medio de un algoritmo que inicialmente se diseña y después se captura como instrucciones reconocidas por el programa para llegar a la solución del problema en cuestión. Asimismo, karel es distribuido como software educativo de código abierto dirigido a principiantes en el estudio de lenguajes de programación, el lenguaje de programación que utiliza karel es una versión especial de java y pascal (Cepeda , 2015).

Figura 16

Ejemplo del mundo de Karel



Nota. Obtenido de (Cepeda , 2015)

4.8.1. Instrucciones de Karel

Son 6 las instrucciones que karel entiende y sabe ejecutar, parecen pocas, pero la cantidad de situaciones que karel puede resolver con esto son numerosas, entre ellas se encuentran la opción de poder resolver hasta laberintos (Cepeda , 2015).

- **Avanza:** Al ejecutar esta instrucción el robot karel avanza un cuadro en la orientación hacia la que está mirando.
- **Gira-izquierda:** Al ejecutar esta instrucción el robot girará un cuarto de vuelta o 90 grados hacia la izquierda.
- **Gira-derecha:** Tenemos que ejecutar 3 veces la instrucción gira-izquierda y karel dará un giro hacia la derecha.
- **Apágate:** Esta instrucción le indica a karel que debe apagarse y terminar la ejecución del programa.
- **Coge-zumbador**: Karel almacenará un zumbador del suelo y lo guardará en su mochila.

- **Deja-zumbador:** El robot abandonará uno de los zumbadores que posee guardados en su mochila en el suelo.

4.9. Laberintos

4.9.1. Origen del Laberinto

Procede del latín “labyrinthus”, y este a su vez del griego “λαβύρινθος labýrinzos”, como definición tienen que es un espacio o lugar formado por calles, carriles, intersecciones y encrucijadas, exclusivamente complejo para desorientar a quien se introduzca en él, es decir, un laberinto es un perímetro aislado del exterior diseñado por multitud de caminos que se entrelazan y cruzan entre sí, muchos de ellos no tienen salida. El origen del laberinto es completamente mitológico y desde la antigüedad muchos se han maravillado por estas formas, por lo que han intentado resolver y solucionar sus diferentes salidas que tiene un laberinto. El primer laberinto que se conoce a lo largo de la historia es el que se encuentra en Creta.

A lo largo de la humanidad, comenzando en la prehistoria hasta nuestros días, los laberintos han experimentado diferenciaciones de forma y de tamaño, pero han sostenido su esencia originaria. Circular por laberintos puede ser una experiencia corporal, visual, y táctil, pero además puede suponer un viaje mental y ficticio por el mundo de la magia, la ciencia, la religión, la moral, el símbolo y la estética (Compromiso Social Bancaja, 2011).

4.9.2. Tipos de laberinto

Los laberintos se clasifican fundamentalmente en dos grandes grupos, según la relación que existe con el centro y la salida del mismo.

4.9.2.1. Laberinto Unicursal. Pese a su aparente complejidad como se puede observar en la figura 17, tiene un único camino, sin ninguna encrucijada ni intersecciones, y mucho menos calles sin salidas o atajos. La persona o robot que lo recorra este laberinto no tendrá posibilidad de elección, ni de error, desde el inicio el camino conducirá directamente al centro, a su corazón, y una vez allí deberá retornar por el mismo lugar que entramos. Este tipo de laberinto se origina en la antigüedad (Compromiso Social Bancaja, 2011).

Figura 17

Laberinto Unicursal

Nota. Obtenido de (Compromiso Social Bancaja, 2011)

Por otro lado, este grupo de laberintos unicursales se divide a su vez en subcategorías, atendiendo a la forma en que fue construido el laberinto como, por ejemplo:

- Laberinto clásico o cretense: Es un laberinto unicursal de forma ovoidal y de diseño muy sencillo.
- Laberinto romano: En un principio era de forma cuadrada, dividido en cuatro cuadrantes alrededor del centro; más tarde, estaba formado de círculos concéntricos, con la misma subdivisión de cuadrantes o zonas enmarcando el centro del laberinto.

4.9.2.2. Laberinto Multicursal. Aparece en el renacimiento, su entramado es complejo como se observa en la figura 18, pues quien lo recorre se verá obligado a decidir por donde debe de continuar su camino. Cada vez que llegue a las numerosas encrucijadas, tendrá el peligro de perder el rumbo que le conducirá al centro o de no encontrar la salida. Para salir de este laberinto es indispensable poseer buena memoria (Compromiso Social Bancaja, 2011).

Figura 18

Laberinto Multicursal

Nota. Obtenido de (Compromiso Social Bancaja, 2011).

Para este tipo de laberintos multicursales se puede citar las siguientes subdivisiones:

- Laberinto barroco: Es un tipo de laberinto que se caracteriza por tener diversas vías muertas o caminos sin salida, además de solo tener una vía o camino correcto para salir de él.
- Laberinto rizoma: Es un tipo de laberinto que se caracteriza principalmente por tener varias ramificaciones infinitas.
- Laberinto manierista: Laberinto con una estructura arbórea, es decir, al final de un corredor o carril localizaremos una bifurcación en Y.
- Laberintos modernos: Son aquellos donde todos los corredores o caminos que lo conforman se interconectan entre sí y no tiene caminos o senderos de circuito cerrado, es decir, aquel corredor que consigue de nuevo el mismo punto de partida.

4.10. Métodos para generar laberintos

4.10.1. Algoritmos Aldous-Broder y Backtrack

En esta sección se abordan dos formas para generar laberintos, los algoritmos aldous-broder y backtrack, que son algoritmos bien conocidos para generar laberintos difíciles de resolver. Los dos métodos asumen el mismo principio, donde se hace y realiza una semejanza en la que se tiene una construcción que se encuentra formada por una matriz de cuartos, que al comienzo ninguno de ellos posee comunicación por ello el algoritmo se encarga de ir atravesando y rompiendo "paredes" y de acuerdo a sus normas, reglas particulares formará el laberinto. Se utilizan los números 1 para las columnas de los cuartos, 2 para las paredes y 3 para indicar el espacio central que se crea al tener las cuatro paredes.

Esta información se guarda en una matriz cuadrada de tamaño $(2n + 1) \times (2n + 1)$, donde n es el número de cuartos por fila o por columna. Se puede ver en la figura 19, la generación de una matriz de tamaño 7×7 en la que la primera línea está hecha por columnas "1" y paredes "2", la segunda línea está hecha por paredes "2", espacios "3" y así sucesivamente (Cruz Ruiz et. al, 2019).

Figura 19

Matriz para la generación de laberintos.

1	2	1	2	1	2	1
2	3	2	3	2	3	2
1	2	1	2	1	2	1
2	3	2	3	2	3	2
1	2	1	2	1	2	1
2	3	2	3	2	3	2
1	2	1	2	1	2	1

Nota. Obtenido de (Cruz Ruiz et. al, 2019)

4.11. Métodos para la resolución de laberintos

4.11.1. Algoritmo de Trèumax

El escritor Charles Pierre Trémaux ideó su propio sistema para salir de cualquier laberinto en cuatro pasos, este método solo funciona si se memoriza el camino o, si no es posible, si se marca la ruta que se está realizando.

Paso 1: nunca pasar por el mismo sitio dos veces.

Paso 2: si se llega a una encrucijada en la que no se ha estado previamente, selecciona al azar un camino a seguir.

Paso 3: si el camino escogido te lleva otro cruce donde un camino es nuevo, pero otro no, escoge la ruta que no se ha explorado.

Paso 4: Por último, cuando un camino te traslada a una encrucijada donde ya has pasado, sigue el primer camino utilizado para llegar allí (Dìaz E. , 2020).

4.11.2. Algoritmo de Tarry y Edouard

Este algoritmo sugiere una manera de salir de cualquier laberinto la cual consiste en que mientras se va recorriendo por los diferentes caminos hay que ir señalando la pared derecha del laberinto. Si llega a una intersección o encrucijada se puede dirigir por cualquier camino y si retorna a una unión, intersección donde ya ha estado, o llega a un camino sin salida tiene que regresar por donde vino (Martín del Rey, 2018).

Por otro lado, si se está dirigiendo por un camino antes señalado o marcado en la parte izquierda y llega a un cruce, intersección ya cursado o visitado, tiene que optar por un camino nuevo, en caso de que no encuentre ningún camino por descubrir, tiene que pasar otra vez por el camino en el que solo se haya dirigido en una dirección. Lo más trascendental es que en ningún

tiempo por ningún motivo se debe dirigir a un camino o carril que esté marcado por ambos lados del pasillo (Martín del Rey, 2018).

4.11.3. Método de la Mano Derecha

Este algoritmo es muy sencillo y únicamente requiere hacer una cosa, no separar la mano derecha de la primera pared que se encuentre. De este modo, habiendo siempre una pared a nuestra derecha se recorrerá toda su superficie y caminaremos hasta que descubramos la salida.

Con este algoritmo es seguro que se localizará y encontrará la salida, por otro lado, a pesar de que siguiendo esta ruta, es muy probable que de todos los caminos que existan, el que se obtiene con la mano derecha no será la más corta y con seguridad no funcionará de la mejor manera en los laberintos que posean su meta o salida en la parte céntrica y además de un circuito cerrado a su alrededor (Yepes, 2019).

4.11.4. Método Aleatorio

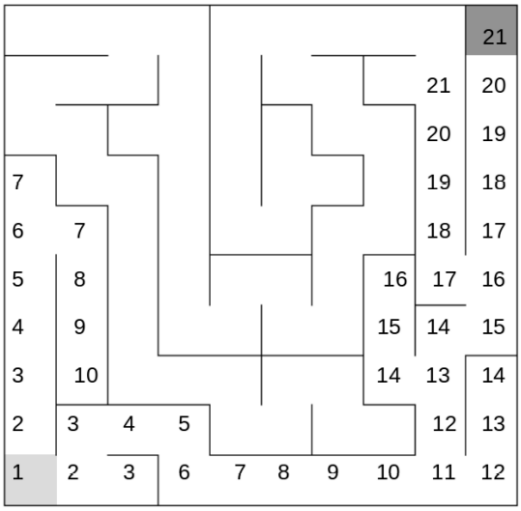
Este método o algoritmo es uno de los más sencillos que existen, pero no garantiza la resolución del laberinto a comparación con los otros métodos ya descritos, este algoritmo de resolución de laberintos se fundamenta en moverse al azar por cualquier camino, callejón, intersección, encrucijada. Esto simboliza desplazarse en una dirección y seguir el pasadizo hasta la siguiente unión o cruce de caminos, algo importante en este algoritmo es que no se debe hacer ningún giro de 180 grados hasta que no sea necesario (Dìaz E. , 2020)**.**

4.11.5. Algoritmo de Dijkstra

El algoritmo de dijkstra se realizó como punto de comparación. Como se muestra en la figura 20, al crecer el índice en las celdas visitadas se puede ver el orden en que se van reconociendo las celdas. La solución se puede reconstruir a partir del punto final es decir la salida o meta del laberinto, la ruta está dada por un único camino que llega hasta el número 1. Este algoritmo muestra un efecto de inundación, donde abarca todo el laberinto examinando todas las vías posibles hasta alcanzar la meta (Cruz Ruiz et. al, 2019).

Figura 20

Solución por el algoritmo de Dijkstra



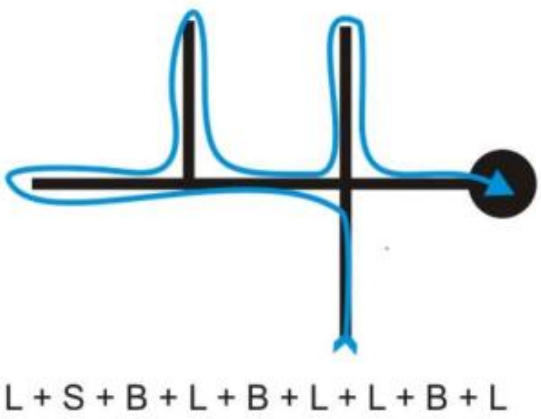
Nota. Obtenido de (Cruz Ruiz et. al, 2019).

4.11.6. Algoritmo de Optimización

Para este algoritmo se tiene que representar cada punto de decisión como un caso el cual puede ser R, L, S o B como se muestra en la figura 21. La R hace referencia a un giro de 90 grados a la derecha, la L hace referencia a un giro de 90 grados a la izquierda, la S avanza recto y la B hace referencia un camino sin salida o un giro de 180 grados.

Figura 21

Algoritmo de optimización

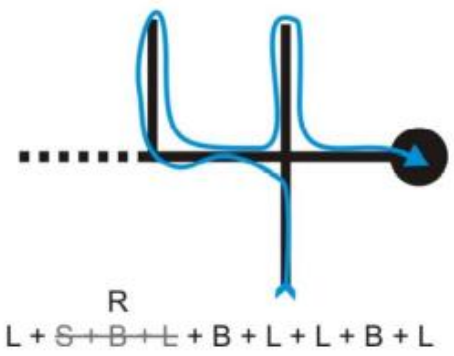


Nota. Obtenido de (Martín del Rey, 2018)

Para evitar y optimizar el camino falso SBL se puede reemplazar con un giro a la derecha R como se puede apreciar en la siguiente figura.

Figura 22

Eliminación del camino "SBL"



Nota. Obtenido de (Martín del Rey, 2018)

4.12. Estructura Mecánica y Alimentación

4.12.1. Estructura Mecánica

Los robots móviles necesitan de una estructura mecánica o de un chasís que forme la plataforma en la que se van a albergar todos los componentes principales como sensores, actuadores, controladores, baterías, llantas, cables.

4.12.2. Sistemas de Locomoción

Según (Andaluz Ortiz, 2011), todo robot móvil se clasifica o se divide según la clase de locomoción por movimiento:

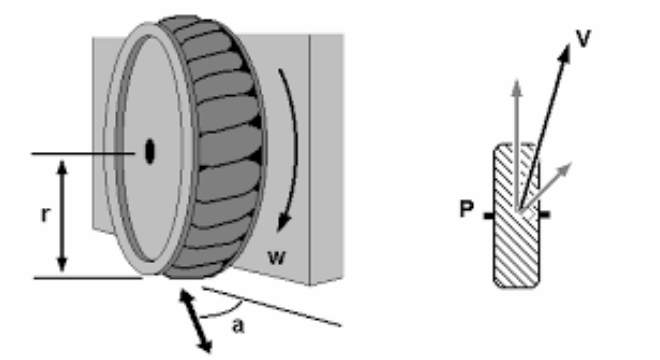
- Locomoción por ruedas.
- Locomoción por patas.
- Locomoción por orugas.

El desarrollo de los robots móviles, más se adapta a los de locomoción por ruedas que los otros, puesto que a diferencia de los demás, este se desempeña fácilmente en lugares remotos.

4.12.2.1. Rueda Fija. El eje de la rueda está sujeto al esqueleto de robot; está directamente relacionada con la tracción del mismo. En la siguiente figura se muestra una representación de lo mencionado anteriormente.

Figura 23

Esquema de Rueda fija

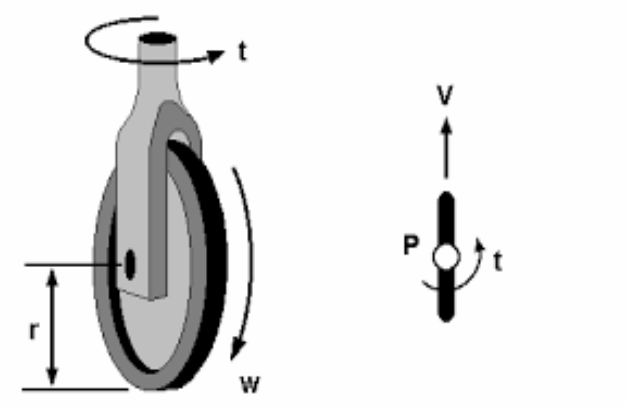


Nota. Obtenido de (Andaluz Ortiz, 2011)

4.12.2.2. Rueda Orientada Centrada. Este tipo de rueda desempeña funciones muy importantes en el robot móvil como por ejemplo la de tracción y dirección, esto se puede observar en la siguiente imagen.

Figura 24

Rueda Orientada Centrada

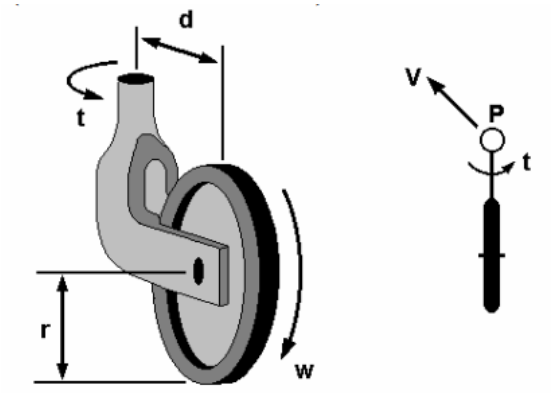


Nota. Obtenido de (Andaluz Ortiz, 2011)

4.12.2.3. Rueda Orientada Descentrada. La funcionalidad primordial de este tipo de rueda es la de proporcionar firmeza a la estructura mecánica del robot. A continuación, se muestra un ejemplo.

Figura 25

Rueda Orientada descentrada



Nota. Obtenido de (Andaluz Ortiz, 2011)

4.12.3. Sistema de alimentación

4.12.3.1. Batería. En la figura 26 se muestra un tipo de batería que por lo general es la alimentación de muchos robots móviles, sin ella, nuestro robot no podría navegar, atravesar y resolver el laberinto debido a que sería inviable tenerlo al robot conectado o acoplado a una fuente de alimentación externa al prototipo. A la hora del diseño hay que tener en cuenta donde ira situada la batería, debido a que es un componente relativamente pesado y ello conllevara un desplazamiento del centro de masas y por la tanto puede desequilibrar el robot (Enegon, 2020).

Figura 26

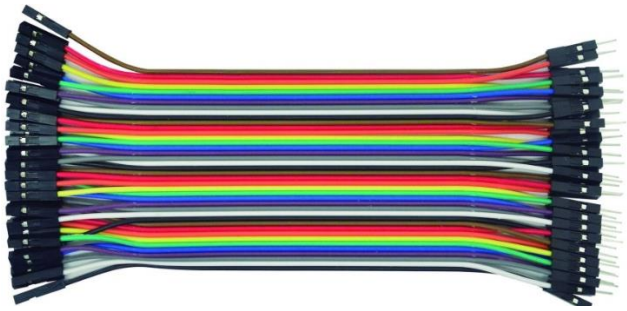
Baterías comúnmente utilizadas en la robótica móvil

Nota. Obtenido de (Enegon, 2020)

4.12.3.2. Cable Puente. Es un cable que tiene un conector en cada punta o muchas veces viene sin ellos como se aprecia en la figura 27, se usa normalmente para interconectar entre sí los componentes en una placa de pruebas por lo que a la hora de diseñar habrá que dejar espacio suficiente para poder interconectar todos los elementos.

Figura 27

Cable Puente



Nota. Obtenido de (Grupo Velasco, 2018)

5. Metodología

5.1. Materiales

Para la realización del presente proyecto de tesis se utilizaron los siguientes recursos y materiales:

Recursos humanos:

- ✓ Tutor de proyecto de tesis.
- ✓ Asesor de proyecto.
- ✓ Autor de proyecto de tesis.

Recursos bibliográficos:

- ✓ Tesis sobre micro robots móviles.
- ✓ Libros de robótica.
- ✓ Artículos de Algoritmos para resolución de laberintos.

Recursos de oficina:

- ✓ Computadora.
- ✓ Paquete de Microsoft Office 2019®.
- ✓ Software de Modelado 3D.
- ✓ Software de simulación KAREL.
- ✓ Software de Arduino para la implementación de la programación sobre el algoritmo del robot.
- ✓ Calculadora.
- ✓ Papel, lápiz y borrador.

Materiales para la construcción del prototipo:

- ✓ Motores DC.

- ✓ Pernos.
- ✓ Destornillador.
- ✓ Cables Jumper.
- ✓ Sensor Ultrasónico.
- ✓ Módulo L298N.
- ✓ Llantas de 4 cm de diámetro.
- ✓ Protoboard.
- ✓ Rueda Loca.
- ✓ Placa de Arduino Uno.
- ✓ Resistencias de 100 Ω, 220 Ω.
- ✓ Diodos LED.
- ✓ Batería de 800 mAh.
- ✓ Multímetro.

5.2. Métodos

La presente investigación está encaminada hacia el diseño y construcción de un micro robot móvil que atraviese laberintos desconocidos, para el cumplimiento de los objetivos planteados en el presente proyecto, los métodos manejados en este proyecto de investigación y construcción se describen a continuación:

5.2.1. Buscar Información

Se realizará una exploración, indagación y recolección de bibliografía a través de libros, artículos científicos, tesis, y material validado referente al tema de investigación, con el fin de acumular la mayor cantidad de información que servirá como sustento del presente proyecto.

Se ejecutó una exploración de información sobre temas como:

- Robótica.
- Cinemática y dinámica de robots.
- Algoritmos para resolver laberintos.
- Aplicaciones de los robots.
- Criterios de implementación de un micro robot laberinto.
- Análisis de materiales de construcción.

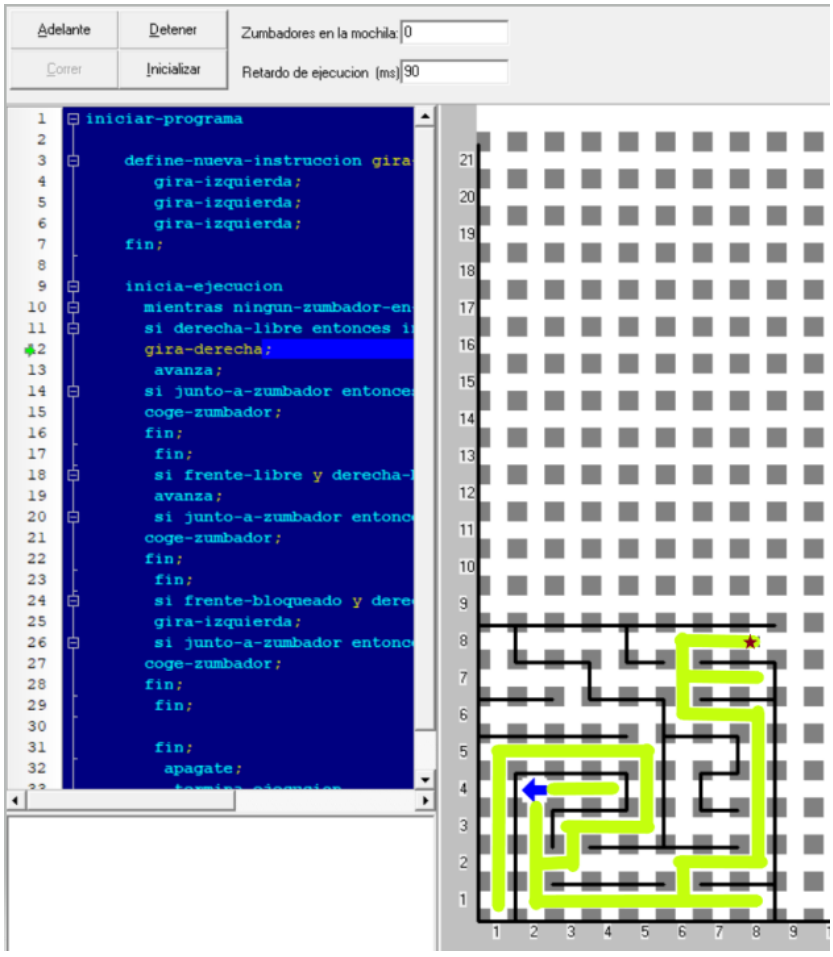
5.2.2. Método para Seleccionar el Algoritmo

Con la finalidad de elegir las mejores alternativas de algoritmos para la navegación del robot dentro del laberinto, así como la optimización de la ruta se debe tener en cuenta varios criterios claves como el diseño del laberinto, las medidas de los carriles.

Por ello se utilizó un software llamado Karel para probar y simular todos los algoritmos de navegación y exploración de laberintos sin la necesidad de construir un entorno de pruebas ni un robot. A continuación, en la figura 28 se muestra la simulación del algoritmo de la mano derecha, a este algoritmo le tomó un tiempo de 60 segundos para resolver el laberinto.

Figura 28

Simulación del algoritmo de la mano derecha en Karel



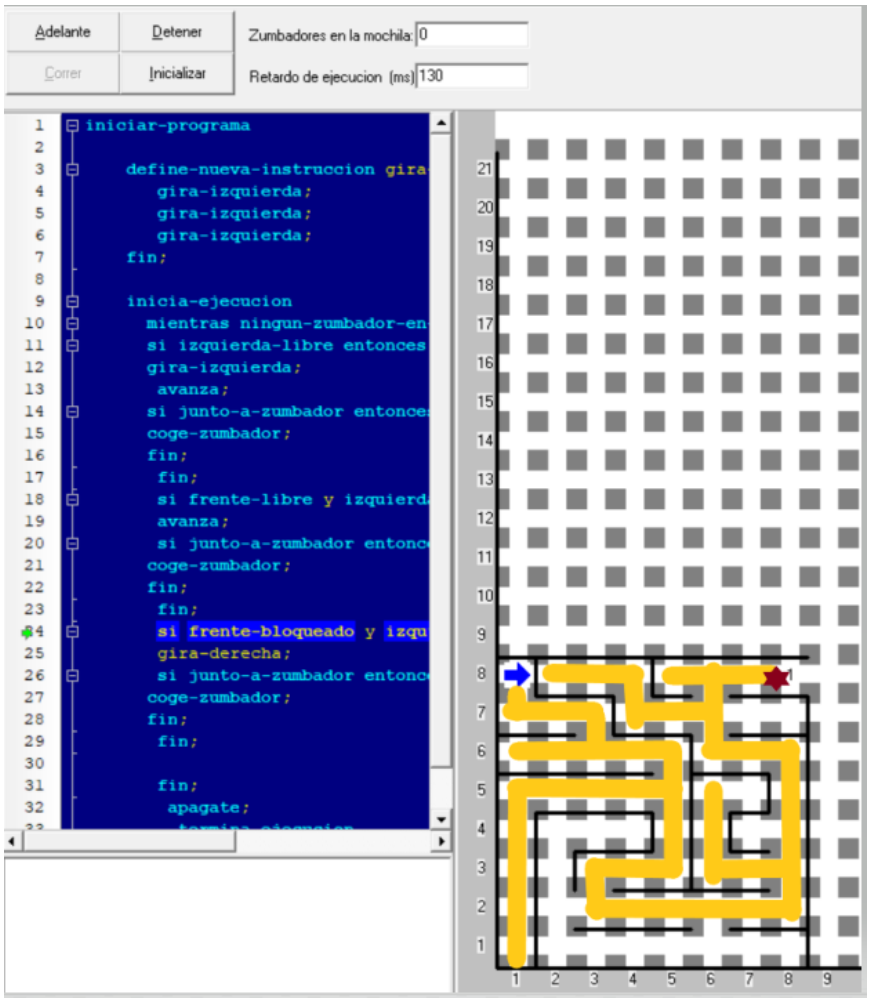
Nota. Elaboración propia

En la figura 29 se muestra la simulación del algoritmo de la mano izquierda, al cual le tomó un tiempo de 100 segundos para resolver el laberinto, mediante esta simulación podemos observar

que el algoritmo de la mano izquierda le toma 40 segundos más en resolver el laberinto en comparación que el de la mano derecha.

Figura 29

Simulación del algoritmo de la mano izquierda en Karel

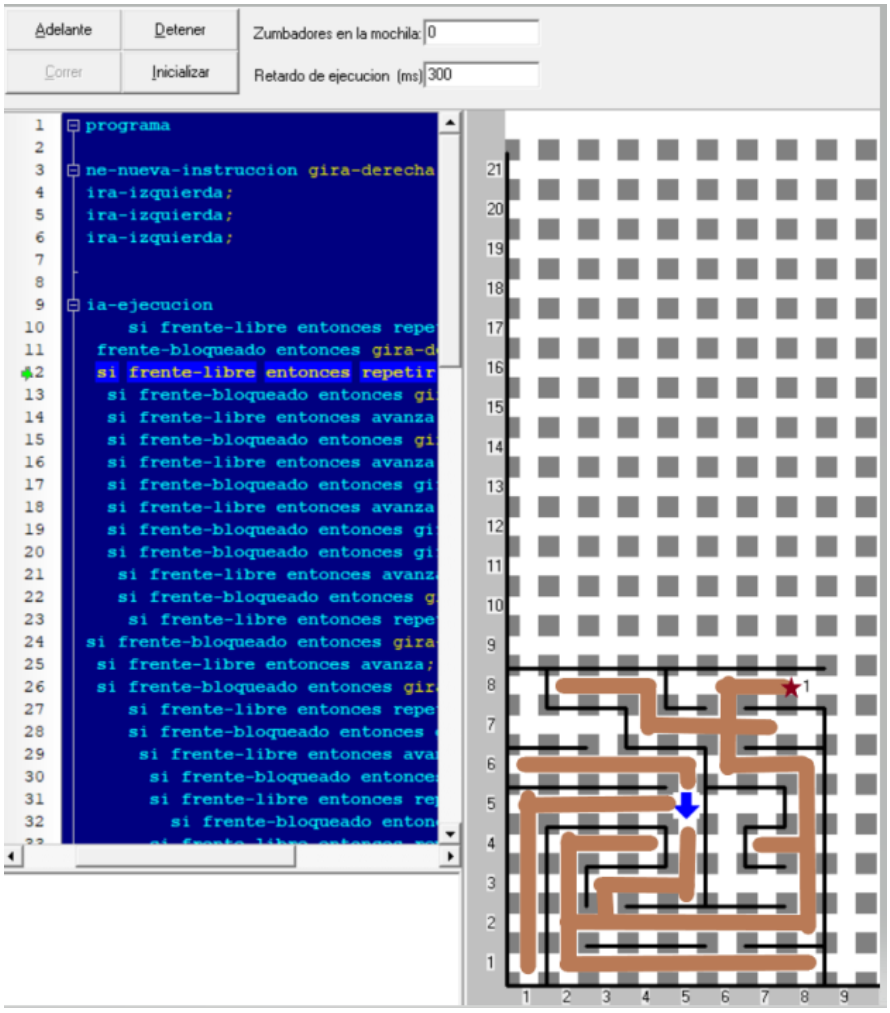


Nota. Elaboración propia

Como se puede apreciar en la figura 30 se muestra la simulación del algoritmo de modo aleatorio el cual se demoró 270 segundos en resolver el laberinto, mediante esta simulación podemos observar que este algoritmo le toma 210 segundos más que el algoritmo de la mano derecha y 170 segundos más que el algoritmo de la mano izquierda.

Figura 30

Simulación del algoritmo aleatorio en Karel



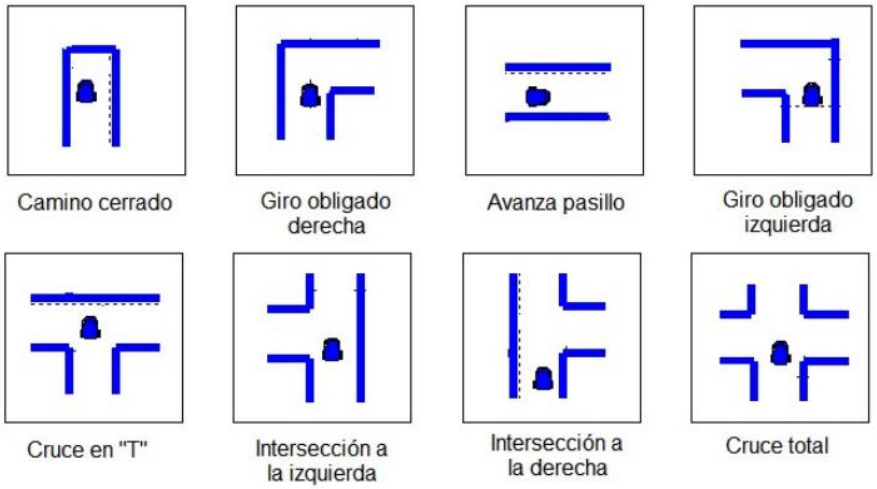
Nota. Elaboración propia

Una vez realizadas las simulaciones de los algoritmos se elige el algoritmo denominado de la mano derecha con el fin de realizar la exploración y navegación cuya tarea es detectar la pared del costado derecho del robot en cada instante de muestreo, permitiendo realizar un seguimiento continuo y efectivo de la trayectoria del laberinto, aun cuando ésta no sea conocida.

Igualmente, para efectuar la navegación del mini-robot Karel dentro del laberinto se debe tener en cuenta las peculiaridades que pueden hallarse en el mismo, como son: camino cerrado, avanza pasillo, giro obligado a la derecha, giro obligado izquierda, cruce en “T”, intersección a la derecha, intersección a la izquierda y cruce total, como se aprecia en la figura 31. Asimismo, el programa de control debe ser idóneo para examinar, reconocer y decidir la acción a tomar cuando se presenten este tipo de casos una vez que se encuentre implementado en el prototipo y avanzar de acuerdo al algoritmo de la mano derecha, es por ello que en el anexo 5 se puede apreciar un esquema y simulación realizada en Tinkercad validando el algoritmo escogido.

Figura 31

Diversos caminos que el robot debe tomar



Nota. Obtenido de (Rodriguez et. al, 2014)

Lo primero que se tiene en cuenta para aplicar el algoritmo de la mano derecha es identificar cada caso posible. para la configuración de los sensores que disponemos en este algoritmo se han encontrado 8 posibles casos, los cuales siguiendo la lógica de la mano derecha obligan al robot a girar a la derecha, girar a la izquierda, seguir recto o dar un giro de 180 grados, en el software del robot se ha configurado para que este sea capaz de solucionar cada caso al que se enfrente una vez que se halle en el laberinto, la decisión que tome el robot en cada caso se encuentra detallado en la siguiente tabla.

Tabla 1

Casos a los que se va a encontrar el robot en el laberinto

	Sensor Derecha	Sensor frontal	Sensor Izquierdo	Decisión
CASO 1 (Camino cerrado)	Pared	Pared	Pared	Girar 180 grados
CASO 2 (Giro obligado derecha)	Libre	Pared	Pared	Girar hacia la derecha
CASO 3 (Avanza pasillo)	Pared	Libre	Pared	Seguir en línea recta

CASO 4 (Giro obligado izquierda)	Pared	Pared	Libre	Girar hacia la izquierda
CASO 5 (Cruce en T)	Libre	Pared	Libre	Girar hacia la derecha
CASO 6 (Intersección a la izquierda)	Pared	Libre	Libre	Seguir en línea recta
CASO 7 (Intersección a la derecha)	Libre	Libre	Pared	Girar hacia la derecha
CASO 8 (Cruce total)	Libre	Libre	Libre	Girar hacia la derecha

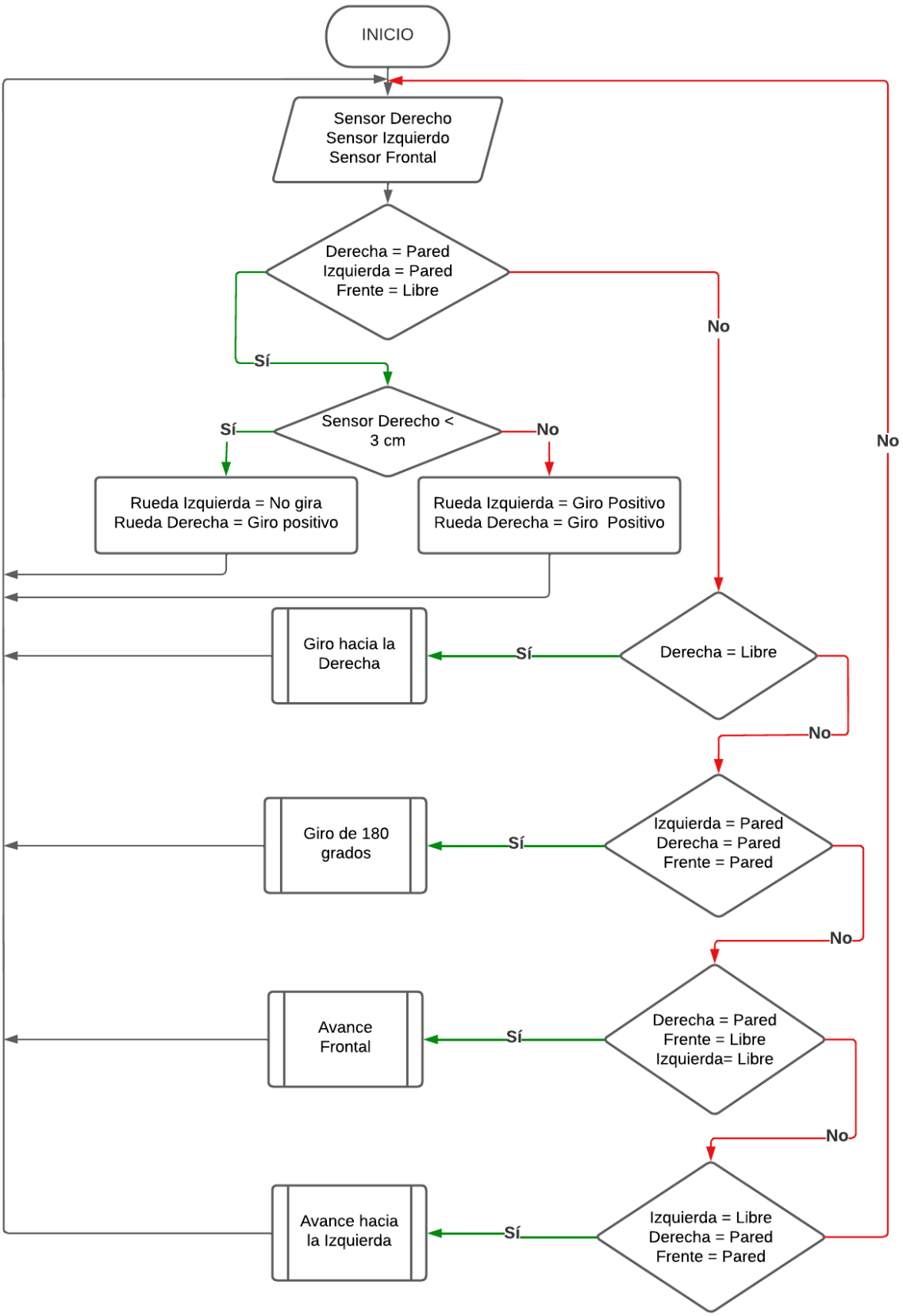
Nota. Elaboración propia

Asimismo, en la revista Nexos Científicos existe un artículo escrito por (Toca & Romo , 2019) en el cual validan el algoritmo de la mano derecha, mediante robots de competencia en la categoría seguidor de línea, solucionando un laberinto en 25 segundos mientras que otros algoritmos también ejecutados, les toma mayor tiempo en encontrar la salida del laberinto.

Por último, en la siguiente imagen, se mostrará el flujograma general del algoritmo de control para el desplazamiento y navegación del robot.

Figura 32

Diagrama de Flujo del Algoritmo para el desplazamiento y navegación en el laberinto



Nota. Elaboración propia

5.2.3. Selección de componentes

5.2.3.1. Fuente de Alimentación. Cuando en proyectos como este, donde se utilizan motores, surge la necesidad de utilizar una alimentación externa, no solo por la búsqueda de autonomía, como se recoge en los requisitos de diseño, sino que, por lo general los motores realizan un elevado consumo y con los 40 mA y 5V que ofrece arduino es insuficiente. Como bien se comentó anteriormente, existen diversas formas de alimentar circuitos electrónicos.

A continuación, se presentarán las principales opciones que se plantearon a la hora de elegir un método de alimentación en concreto:

La batería de marca múltiple power proporciona un total de 9V. Su capacidad está en niveles superiores a cuatro pilas de serie pila de 6V (600mAh), y más si hablamos de pilas alcalinas (600-700 mAh), capaces de aportar una intensidad de 1A. Aunque mejora notablemente la capacidad de la opción anterior, tampoco son recargables, por lo que no son la opción óptima. Además, 6V se hacen cortos para la alimentación de los dos motores CC con los que contará el robot, aunque con 6V funcionen perfectamente, las pilas en serie nunca aseguran un valor exacto, y con menos de 4.8V los motores trabajarían a unos rpm demasiado bajos, por no hablar de que las pilas convencionales pronto sufren caídas de tensión y ni por asomos aportan una tensión constante de 1.2V por unidad.

El modelo elegido ha sido de la marca MP (múltiple power), con unas especificaciones de capacidad de 800 mAh, con capacidad de volverse a cargar una vez que se descargue, y en cuanto a su voltaje es de 9 V.

Para seleccionar la batería es necesario hacer una estimación de la corriente mínima que va a consumir el sistema, es decir, de la carga total mediante la ecuación 1 planteada por (Costa Rodriguez, 2017).

Ecuación 1

Fórmula para estimar la corriente que consume el robot

$$CE = Ca + Cm + Cs + Ccm + Cl \quad (1)$$

Donde:

CE = Consumo Estimado (mA)

Ca = Consumo de Arduino (mA)

Cm = Consumo de motores (mA)

Cs = Consumo de sensores (mA)

Ccm = Consumo del controlador de motores (mA)

Cl = Consumo de los diodos LED (mA)

$$CE = 40\ mA + (2 * 180mA) + (3 * 25mA) + 150\ mA + (4 * 20mA)$$

$$CE = 705\ mA$$

Por lo tanto, se considera el consumo calculado anteriormente como una estimación de carga mínima que consumirá el sistema, asumiendo que la carga final consumirá más corriente. En la figura 33 se muestra la batería que cumple con las necesidades del proyecto.

Figura 33

Baterías seleccionadas para alimentar al robot

Nota. Elaboración propia

La batería finalmente seleccionada tiene una capacidad de 800mAh. Una vez ya elegida calculamos el tiempo de duración que nos puede brindar sin recargarla, usando la ecuación 2 planteada por (Costa Rodriguez, 2017), lo que el robot tendrá una autonomía de:

Ecuación 2

Fórmula para calcular el tiempo de duración de una batería

$$TD = \frac{CB}{CE} \qquad (2)$$

Donde:

TD = Tiempo de duración de la batería

CB = Capacidad de la batería (mAh)

CE = Consumo estimado (mA)

$$TD = \frac{800(mAh)}{705\ (mA)}$$

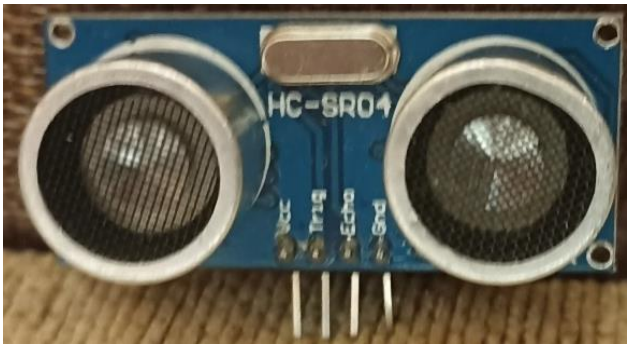
$$TD = 1{,}12h = 67\ min$$

Con estos cálculos comprobamos que el robot tendrá suficiente autonomía para atravesar y navegar por el laberinto sin necesidad de recargar la batería.

5.2.3.2. Sensores Ultrasónicos. Para este proyecto de tesis se ha preferido el sensor de ultrasonidos HC-SR04 que se puede apreciar en la figura 34. Se trata de un sensor que maneja un sistema de emisión/recepción autónoma, por lo que dispone con dos membranas o transductores que son el altavoz y el micrófono.

Figura 34

Sensores seleccionados para medir distancias



Nota. Elaboración propia

Este sensor no destaca principalmente por su precisión o exactitud, la cual es bastante baja de alrededor un centímetro de diferencia lo cual la hace imprecisa, pero si es conveniente en cambio, por su bajo coste y su minúsculo consumo, de alrededor de 15 a 25 mA, así como por su sencillez de uso. A pesar de la comentada falta de precisión, estos sensores sí son efectivos en este tipo de robots al cual queremos construir por lo que es perfecto para nuestro proyecto debido a que la precisión no es determinante, son óptimos y económicos, además en el anexo 3 se muestra un pequeño código para verificar el buen funcionamiento de estos sensores y asegurarnos que las distancias que marquen sean las correctas, todo esto antes de colocarlos en el prototipo. En la tabla 2 se presentan las principales características de este sensor HC-SR04.

Tabla 2

Características del sensor ultrasónico seleccionado

Sensor Ultrasónico HC-SR04	
Voltaje Operativo	5V
Ángulo de detección	No más de 15º
Distancia de detección	Desde los 2 cm hasta los 450 cm
Precisión	De hasta 3 mm
Consumo de corriente	De15 a 25 mA
Cables de conexión	Macho-Hembra

Nota. Elaboración propia

5.2.3.3. Placa Arduino. El hardware de Arduino UNO es la placa ideal para implementar en este proyecto debido a que cuenta con los pines necesarios y suficientes para conectar los tres sensores ultrasónicos, el driver L293D y otros componentes que forman parte del micro robot. Además, es la placa ampliamente documentada en la familia Arduino y la que lleva más tiempo en el mercado, siendo la base para el desarrollo del resto de placas.

Por otro lado, esta placa de Arduino UNO cuenta tanto con entradas como salidas digitales. Estas se hallan en un zócalo de la placa, numeradas, una detrás de la otra y perfectamente indicadas, estas permiten conectar y trabajar con sensores, actuadores que utilicen señales digitales. Aparte de las señales digitales también se pueden conectar señales analógicas como son: sensores de temperatura, sensores de distancia, entre otros. En la tabla 3 se detallan las características principales sobre la placa de Arduino UNO, por último, esta placa microcontroladora es muy fácil de encontrar en el mercado local y sobre todo es económica a comparación de otras como la Placa Raspberry Pi.

Tabla 3

Características de la placa Arduino seleccionada

Arduino UNO	
Microcontrolador:	ATmega328P
Voltaje de Operación:	5V
Pines digitales:	14
Pines PWM:	6
Pines de entradas análogas:	6

Corriente DC en cada pin I/O:	20 mA
Corriente DC en los pines de 3V y 5V:	50 mA
Memoria Flash:	32KB
Memoria SRAM:	2KB
Memoria EEPROM:	1KB
Velocidad de reloj:	16 MHz
Puertos de alimentación	Jack, USB, pin de alimentación

Nota. Elaboración propia

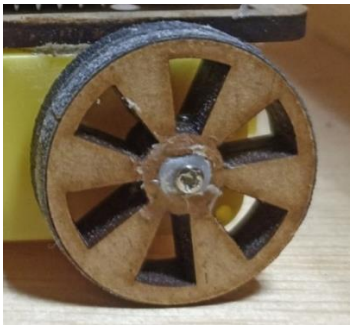
5.2.3.4. Sistema de Locomoción

- **Ruedas Laterales**

En el mercado local no se dispone de ruedas pequeñas que sean adaptables al presente proyecto, es por ello la necesidad de diseñar y construir ruedas con medidas propias que se ajusten perfectamente a los motores elegidos, las ruedas fueron diseñadas mediante el software de SolidWorks y construidas con madera MDF de 6 cm de espesor y un radio de 2 cm, utilizando una cortadora láser es posible tener esa precisión en las medidas.

Figura 35

Ruedas laterales construidas



Nota. Elaboración propia

El prototipo necesita de dos ruedas, mostradas en la figura 35, esto para estabilizar el movimiento del robot sea a la izquierda o derecha.

- **Rueda Loca**

También se la conoce como rodillo de transferencia (figura 36), se la puede conseguir en las tiendas locales de electrónica, esta rueda es de metal y ayuda al robot a darle sentido, estabilidad

y dirección. Estas ruedas, no acarrean ningún motor, giran libremente según la velocidad que tenga el robot. Además, pueden orientarse con facilidad según la dirección del movimiento.

Figura 36

Rueda loca seleccionada

Nota. Elaboración propia

Esta rueda es ideal utilizarla para este proyecto debido a que nos permite que el robot tenga la posibilidad de girar sobre sí mismo al manejar dos motores A continuación, en la tabla 4 se especifican sus principales características.

Tabla 4

Características de la rueda loca

Rueda Loca	
Material	metal
Altura	20 mm
Distancia entre agujeros	40 mm
Agujero de montaje	4 mm
Peso máximo soportable	10 kg
Giro	360 grados de libertad, giro sinfín.

Nota. Elaboración propia

5.2.3.5. Actuadores

- **Motor de Corriente Continua**

El micro robot funcionará con dos motores de corriente continua, cuya función es la de hacer desplazar al robot por el laberinto, tanto en sentido de avance, como de retroceso o giro en

ambos sentidos. Esto se consigue debido a que ambos motores incorporan una rueda que es la que facilita el desplazamiento.

Para seleccionar el motor, necesitamos saber el torque necesario que requiere el robot con el fin de que se pueda desplazar por el laberinto, para ello según (Neal, 2010), el torque se puede calcular mediante la ecuación 3, utilizando la masa total del robot, el radio de la rueda y la aceleración que se desee.

Ecuación 3:

Fórmula para calcular el torque necesario del motor.

$$T = m * (a + g * \sin(\theta)) * \mathrm{r} \quad (3)$$

Donde:

T = Torque del motor.

M = Masa total del robot.

a = Aceleración.

θ = Ángulo del plano.

g = Gravedad.

r = Radio de las ruedas.

Como datos se posee, que la masa del prototipo es 250 g, se desea manejar una aceleración de 2 m/s^2, el radio de la rueda es de 2 cm y el ángulo θ es cero debido a que la pista del laberinto es completamente plana. Usando la ecuación 3 se tiene:

$$T = 0.25\ kg * \left(2\frac{\mathrm{m}}{s^2} + 9.81\frac{\mathrm{m}}{s^2} * \sin(0)\right) * 0.02m$$

$$T = 0.01\ Nm$$

Existen varios motores que cubrirían este torque, pero en la figura 37 se muestra el motor seleccionado con un torque de 0.108Nm y velocidad de 250 RPM.

A pesar de que el robot no necesita una velocidad de desplazamiento excesiva, se puede calcular la velocidad máxima a la que podrá desplazarse el prototipo con la ecuación 4 que propone (Neal, 2010) .

Ecuación 4

Velocidad máxima del robot

$$V = \frac{V\ RPM}{60} * 2 * \pi * r \quad (4)$$

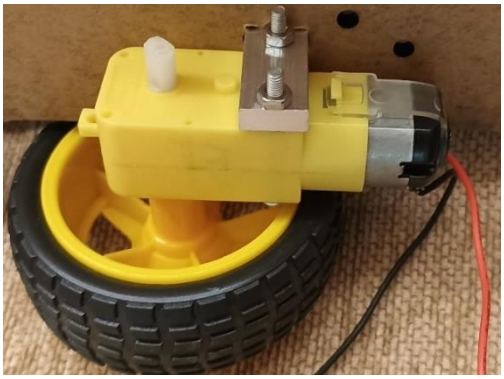
$$V = \frac{250}{60} * 2 * \pi * 0.02m$$

$$V = 0{,}524\ m/s$$

Entonces la velocidad máxima que se podría alcanzar con el robot es de 0.524m/s

Figura 37

Motores reductores seleccionados



Nota. Elaboración propia

El modelo de motores elegido ha sido seleccionado especialmente por su reducido precio y su sencillez, que sin lugar a dudas es el modelo más frecuente en proyectos como este, es por ello que es importante probar y verificar su buen funcionamiento usando el anexo 4 antes de implementarlos en el prototipo, el tipo de motores seleccionado es uno de 5V genérico. Este motor presenta unas características muy ajustadas, pero totalmente válidas para el objetivo del robot. El modelo designado además cuenta con una caja reductora 1:48. La caja reductora es un mecanismo que forma un conjunto de engranajes, con el que se logra mantener la velocidad de salida. A continuación, en la tabla 5 se detallan sus principales características.

Tabla 5

Principales características del motor seleccionado

Motor CC	
Voltaje de alimentación:	Entre 3V y 9V
Corriente de trabajo:	150 mA
Torque máximo:	0.108 N*m
Velocidad máxima:	250 rpm
Peso:	16,5 g
Caja reductora:	1:48

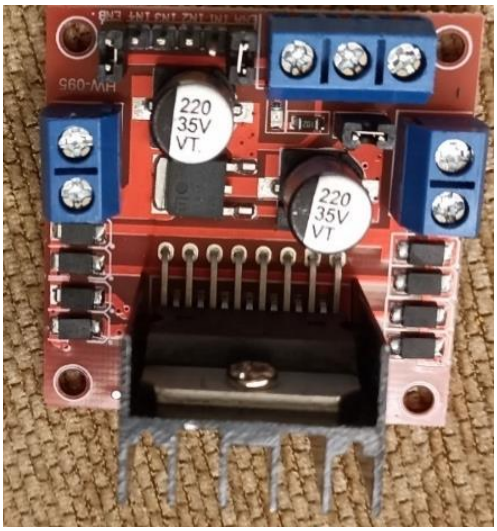
Nota. Elaboración propia

5.2.3.6. Controlador de Motores (Driver L98N). El controlador seleccionado ha sido el L298N que se puede observar en la figura 38, un módulo integrado que permitirá controlar dos motores de corriente continua. Este controlador se basa en el concepto de puente en H. El puente en H es un circuito electrónico cuyo propósito es la de convertidor en puente continua-alterna.

El trabajo de este tipo de circuitos es sencillo, a partir de una entrada de corriente continua se conseguirá una salida de corriente alterna cerrando y abriendo interruptores en una estipulada secuencia. El controlador LN298N dispone de 2 puentes en H, permitiendo controlar 2 motores CC (un puente para cada motor) por intermedio de 4 salidas digitales de arduino (2 para cada motor). Para hacer girar cada motor en un determinado sentido, se ha de disponer una patilla a HIGH y la otra a LOW.

Figura 38.

Controlador L98N seleccionado



Nota. Elaboración propia

Este módulo cuenta con diversos elementos adicionales necesarios para trabajar correctamente. Además de un integrado L298N, encargado de ejercer el control, tiene un regulador lineal LM7805, similar al que tiene arduino, que permitirá alimentar a la parte lógica con 5V indistintamente de la tensión con la que alimentemos los 2 motores por la patilla de 12V y con diodos de protección, que salvaguardarán el sistema de inversiones de polaridad o sobreintensidades.

5.2.3.7. Diodos LED. Se van a utilizar 3 diodos led, que nos indicarán cuando el robot gire a la derecha, gire a la izquierda y cuando el robot encuentre la salida del laberinto, como los pines de salida de la placa arduino entregan 5 voltios es necesario utilizar resistencias para proteger a los diodos LED que consumen menos voltaje, para ello utilizamos la ecuación 5 propuesta por (Irwin, 2004).

Ecuación 5

Fórmula para calcular resistencias

$$R = \frac{Vcc - Vd}{Id} \tag{5}$$

Dónde:

Vcc = Voltaje de salida del Pin de Arduino

Vd = Voltaje soportado por el diodo

Id = Corriente a través del diodo LED

Cuando el robot gire a la derecha se encenderá un LED color azul que tiene los siguientes datos:

- Voltaje de salida del pin de Arduino = 5 V.
- Voltaje del diodo = 3.4 V.
- Corriente a través del diodo = 20 mA.

Utilizando la ecuación 5 se calcula la resistencia para el diodo LED azul.

$$R = \frac{5V - 3.4V}{20mA}$$

$$R = 80\ \Omega$$

Como no se encuentran resistencias con valores de $80\ \Omega$ se procede a seleccionar el inmediato superior que es de 100 Ω.

Ahora cuando el robot gire a la izquierda se encenderá un LED color verde que tiene los siguientes datos:

- Voltaje de salida del pin de Arduino = 5 V.
- Voltaje del diodo = 2.4 V.
- Corriente a través del diodo = 15 mA.

Utilizando la ecuación 5 se calcula la resistencia para el diodo LED verde.

$$R = \frac{5V - 2.4V}{15mA}$$

$$R = 173.3\ \Omega$$

Como no se encuentran resistencias con valores de 173.3 Ω se procede a seleccionar el inmediato superior que es de 180 Ω.

Por último, cuando el robot encuentre la salida del laberinto se encenderá un LED color rojo que tiene los siguientes datos:

- Voltaje de salida del pin de Arduino = 5 V.
- Voltaje del diodo = 1.9 V.
- Corriente a través del diodo = 15 mA.

Utilizando la ecuación 5 se calcula la resistencia para el diodo LED rojo.

$$R = \frac{5V - 1.9V}{15mA}$$

$$R = 206.7\ \Omega$$

Como no se encuentran resistencias con valores de 206.7 Ω se procede a seleccionar el inmediato superior que es de 220 Ω.

5.2.3.8. Interruptor ON-OFF. Para encender y apagar el circuito eléctrico se utilizará un interruptor tipo rocker (figura 39), muy utilizado en proyectos de electrónica, es necesario utilizarlo en el presente proyecto para interrumpir la corriente de alimentación, que pase desde la batería hacia el circuito cuando sea necesario, más específicamente al inicio cuando se ponga en funcionamiento el robot, se presionará este dispositivo y el prototipo ejecutará sus funciones para resolver el laberinto y una vez encontrado la salida se procederá a desactivar o apagar el robot mediante el interruptor con la función on/off.

Figura 39

Interruptor tipo rocker on/off

Nota. Elaboración propia

5.2.3.9. Chasis del Robot. El chasis del kit smart car para arduino, fue la primera estructura utilizada en el robot como se puede observar en la figura 40. En su momento se tomó esta decisión para tener los primeros contactos con un robot de esta cualidad. Este prototipo, contaba exclusivamente con los elementos de dicho kit, sin ninguna alteración.

A la vez que se iban efectuando las primeras pruebas, salió la necesidad de ir incorporando nuevos componentes. Por ello, y por ciertos problemas que surgían, se apostó por la construcción de nuevos diseños más adecuados a las necesidades existentes en su momento.

Después de muchas pruebas se apostó por instalar las ruedas en la parte posterior del chasis y la rueda loca en la parte trasera del mismo. De esta manera se consiguió un funcionamiento favorable en las rectas. En la parte delantera de la estructura, el sensor ultrasónico controlaba la distancia lateral con la pared, y de acuerdo con esta distancia, los motores regulaban su velocidad para girar en un sentido u otro. La regulación era adecuada, algo que con otros diseños, no se conseguía. Esto se debía a la colocación opuesta longitudinal, de ambas ruedas y del sensor, de esta manera el eje de giro se encontraba en la parte posterior de la estructura, por lo que se regulaba excelente la trayectoria.

Figura 40

Chasis del primer prototipo con los motores en la parte trasera

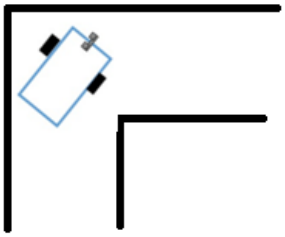
Nota. Elaboración propia

El inconveniente del diseño anterior aparecía a la hora de ejecutar los giros. Al situarse en la parte posterior las ruedas, el radio de giro abarcaba toda la longitud del robot, por lo que los giros debían de ser muy finos, un gran problema porque de eso se trata el proyecto. Por ello, el diseño de este prototipo fue rechazado.

Posteriormente, tras realizar cuantiosas pruebas, se decidió por volver a utilizar el primer chasis y realizar sobre el las modificaciones necesarias, esto se puede apreciar en la figura 41. Aunque para este montaje la regulación del avance por el pasillo no parecía tan buena, como solución general es mejor, ya que los giros los ejecuta sin ningún problema.

Figura 41

Giro del primer prototipo hacia la derecha

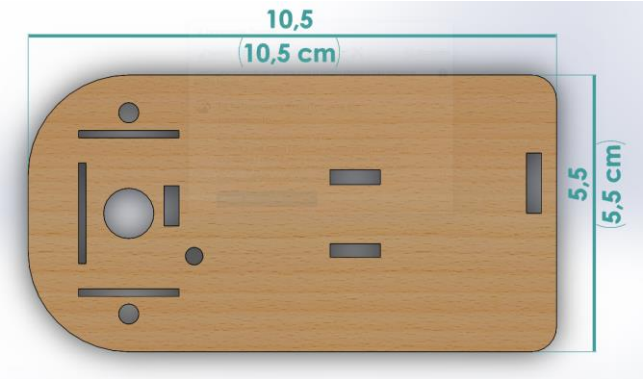


Nota. Elaboración propia

Finalmente, como la idea de este proyecto es elaborar un mico robot se apostó a que este tenga una medida máxima de 11 cm de largo y 6 cm de ancho, para lograr este objetivo se diseñó en SolidWorks todas las piezas del chasis que conforman el robot a continuación en la figura 42 se muestra la parte que va a sujetar los motores y las ruedas tanto las ruedas laterales como la rueda loca.

Figura 42

Pieza del chasis que sujeta a los motores y ruedas.



Nota. Elaboración Propia

La pieza que va a sostener al protoboard se muestra en la figura 43 la misma que tiene una abertura para poder pasar los cables fácilmente, esta pieza va a estar en la parte superior del chasis.

Figura 43

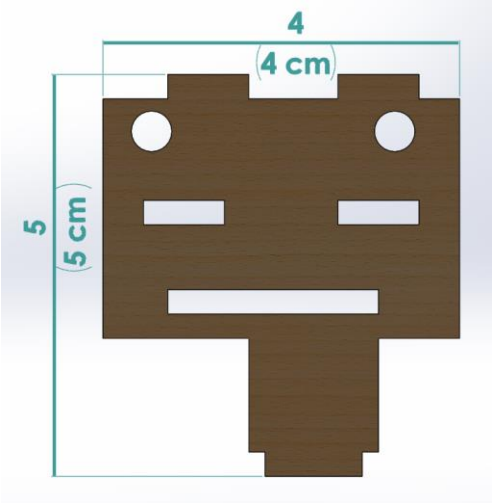
Pieza que sostiene el protoboard

Nota. Elaboración Propia

En la siguiente figura se muestra le diseño de la pieza que va a sujetar la fuente de alimentación esta pieza va a ir insertada entre las piezas que sujetan los motores como la pieza que sujeta el protoboard.

Figura 44

Pieza sujetadora de la fuente de alimentación

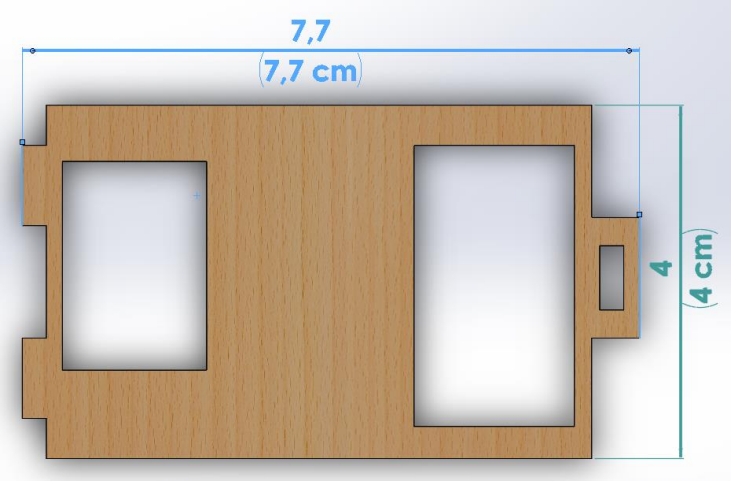


Nota. Elaboración Propia

La siguiente pieza mostrada en la figura 45 cumple con la función de dar firmeza a todo el chasis y sobre todo para dejar pasar las conexiones de los cables, que van desde el arduino hacia el protoboard.

Figura 45

Pieza encargada de dar firmeza al chasis

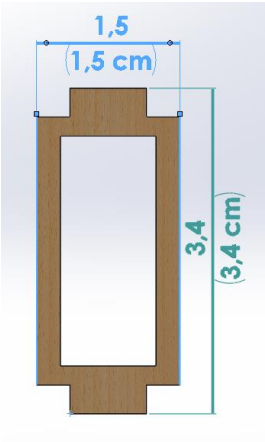


Nota. Elaboración Propia

La última pieza que conforma el chasis del robot se muestra en la figura 46 la cual es la encargada de dar firmeza a los sensores ultrasónicos y también sirve para dejar pasar los cables que van desde los sensores hacia la placa arduino y hacia el protoboard. Para mejor interpretación las piezas del chasis se encuentran detalladamente en los planos del anexo 1.

Figura 46

Pieza destinada a dar firmeza a los sensores ultrasónicos



Nota. Elaboración Propia

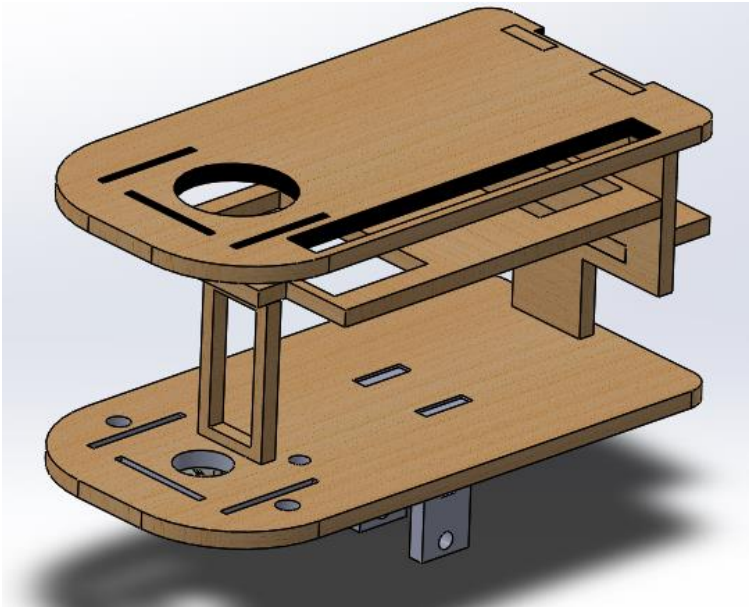
5.2.4. Ensamblaje del robot

En este apartado se habla de la forma como se ensambló el robot móvil diseñado para el presente proyecto de titulación, desde su parte mecánica hasta su sistema eléctrico. En la parte mecánica se describirá como utilizando pedazos de madera se edifica su carrocería o chasis, mientras que para el sistema eléctrico se dará a conocer los diferentes acondicionamientos de los sensores HC-SR04 utilizados, además de la forma de controlar los motores tanto de dirección como de tracción.

5.2.4.1. Sistema Mecánico. Una vez diseñadas las piezas del chasis y seleccionado todo el sistema mecánico como llantas traseras y delanteras se procede a ensamblar primeramente cada una de las piezas del chasis que la conforman como se muestra en la siguiente imagen.

Figura 47

Ensamblaje de las piezas que forman el chasis

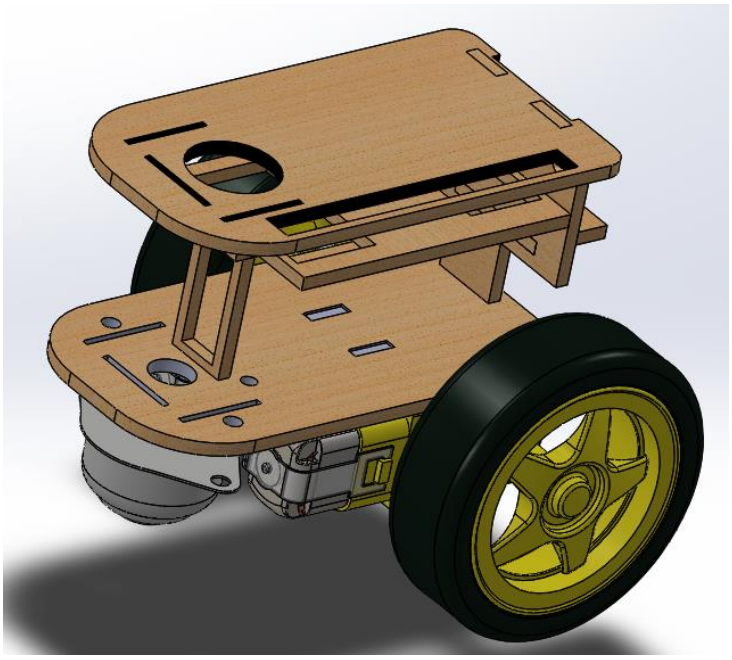


Nota. Elaboración Propia

Una vez que tenemos el chasis se procede a colocar los motores elegidos, sus dos ruedas laterales y la rueda loca que va en el centro como se puede apreciar en la siguiente imagen.

Figura 48

Implementación de los motores y ruedas en el chasis.

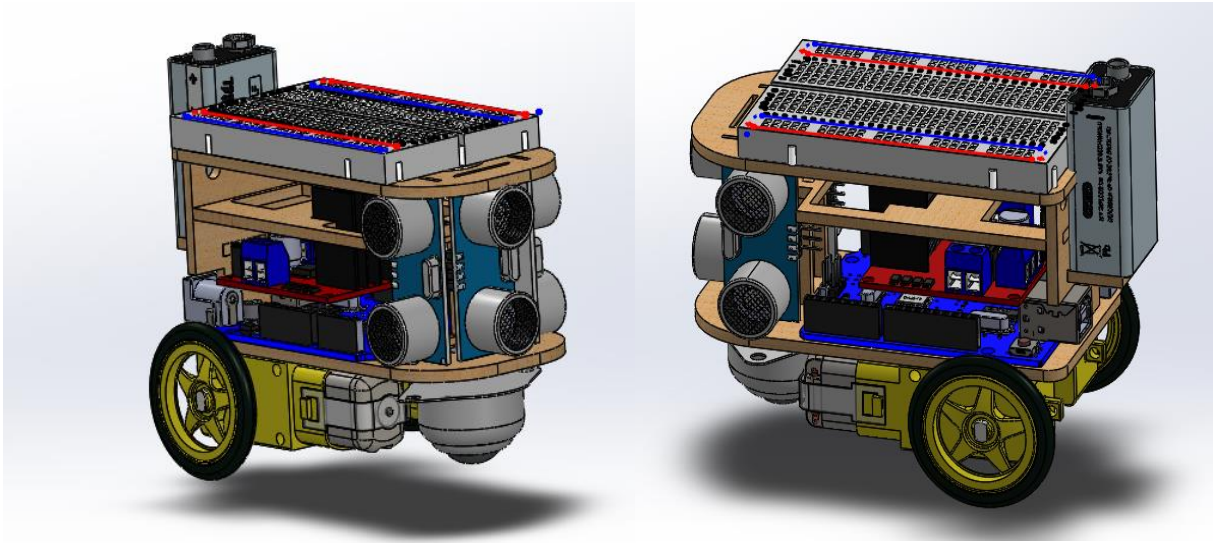


Nota. Elaboración Propia

Cuando se tiene todo el chasis armado y sus motores implementados, lo siguiente es colocar el resto de componentes tal como se puede observar en la figura 49, que son los tres sensores ultrasónicos, la placa arduino, un pequeño protoboard, y su fuente de alimentación en este caso una batería de 9 voltios.

Figura 49

Colocación de los componentes del robot en el chasis

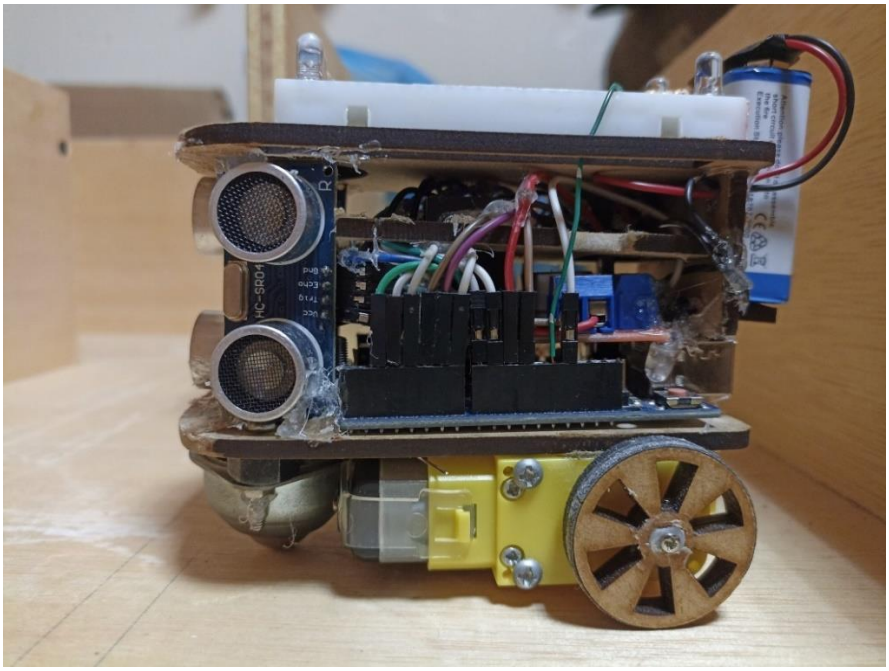


Nota. Elaboración Propia

5.2.4.2. Sistema eléctrico. En este apartado ya con todos los componentes colocados en el chasis, se procede a realizar las respectivas conexiones eléctricas como se puede apreciar en la figura 50 como son; desde la placa arduino a los diferentes sensores ultrasónicos, así mismo las conexiones que van en el driver L298D y los dos motores, igualmente se procede alimentar la placa arduino y el protoboard mediante la batería de 9 voltios, por último se debe conectar el interruptor y los tres diodos led.

Figura 50

Vista lateral izquierda del robot



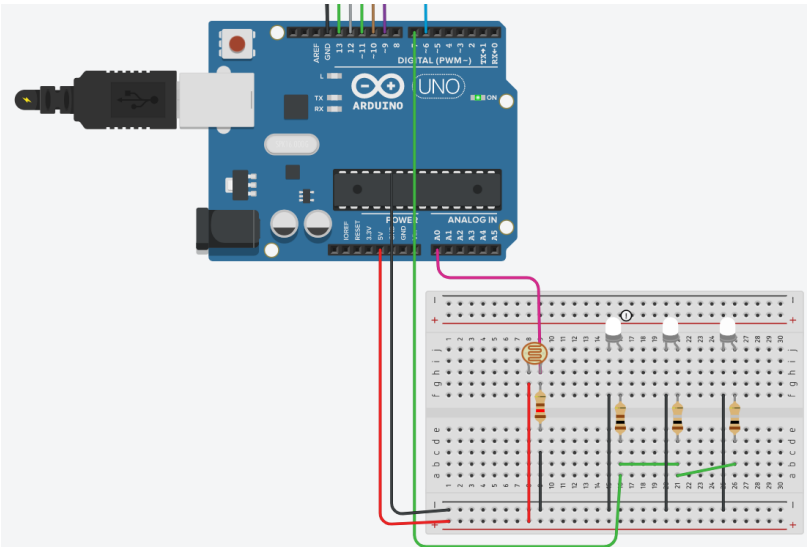
Nota. Elaboración Propia

Es importante mencionar que los sensores ultrasónicos necesitan de una buena iluminación para que funcionen correctamente, es por ello que mediante el programa de tinkercad se tuvo que diseñar un sistema automático de luces, que se enciendan cuando el ambiente en el que se encuentra el robot la luminosidad no sea tan buena, por ello se utilizó un sensor de luz o fotorresistor que se activará cuando la luminosidad del ambiente sea menos de 40 lux.

En la figura 51, se muestra el esquema y simulación desarrollado en tinkercad que demuestra el correcto funcionamiento de este sistema.

Figura 51

Diagrama y simulación del sistema de luces.

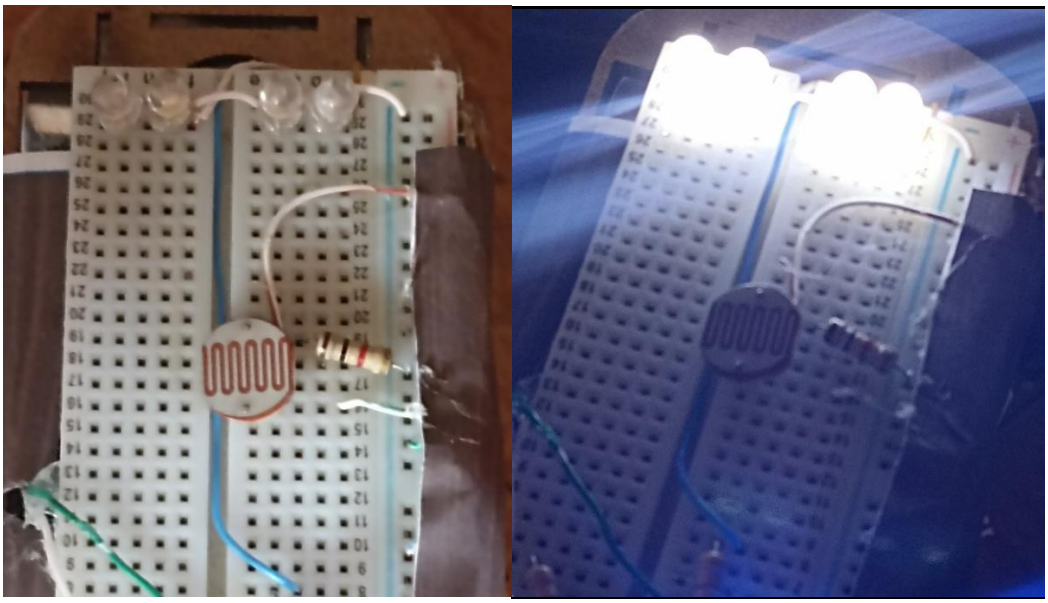


Nota. Elaboración Propia

En la siguiente figura se puede apreciar la implementación del sistema automático de luces en el robot, para este sistema se utilizaron diodos led color blanco, conectados en paralelo, también se utiliza el fotorresistor encargado de enviar la señal cuando detecte obscuridad.

Figura 52

Implementación del sistema de luces en el robot.



Nota. Elaboración Propia

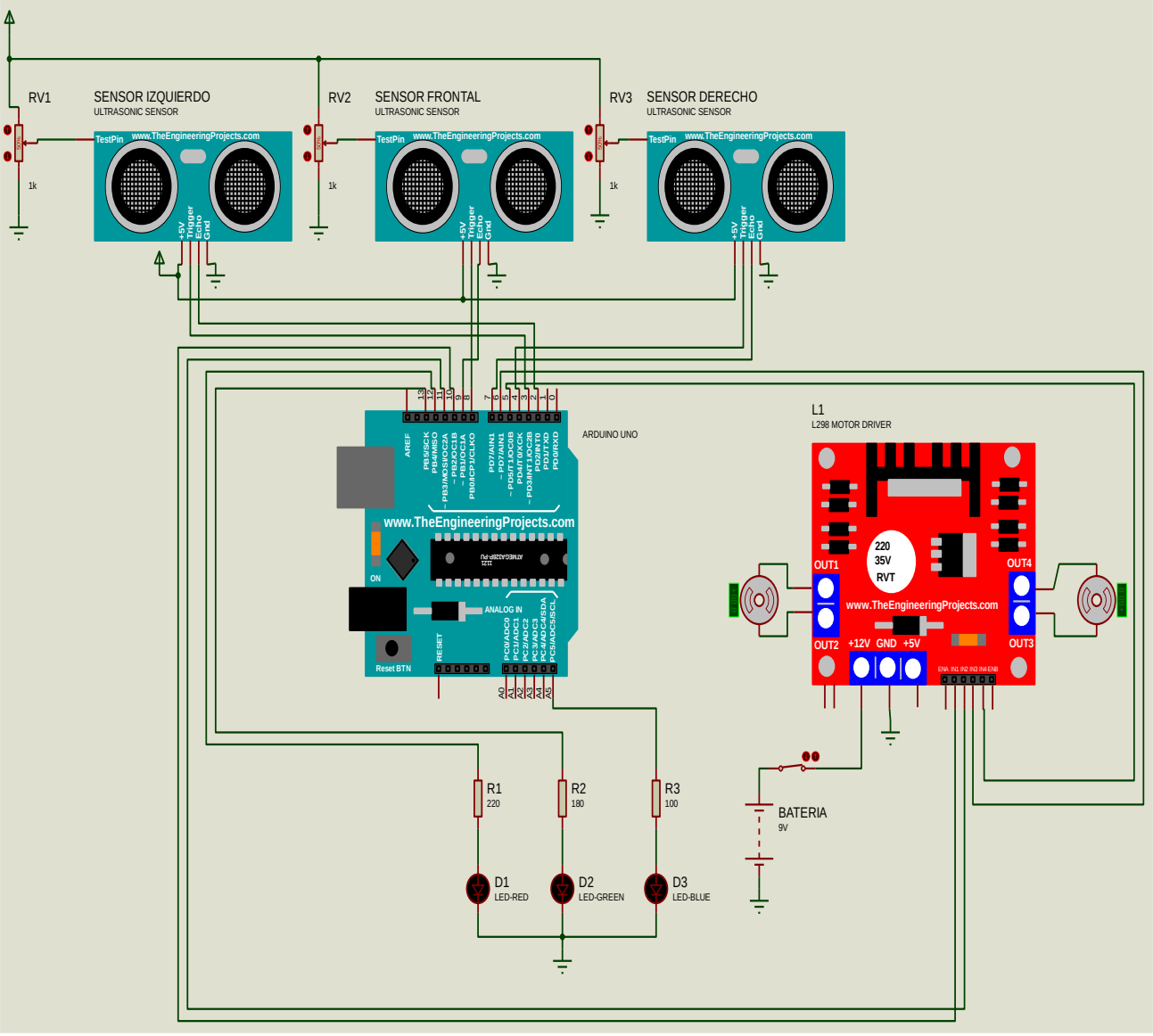
Mediante el software de Proteus se logró realizar un diagrama general de todas las conexiones eléctricas, que se utilizaron en la elaboración del robot, como se observa en la figura 53, en este esquema se utilizaron todos los componentes que se describieron anteriormente, como

son los sensores ultrasónicos, motores, diodos led, resistencias, batería, entre otros. Para realizar las respectivas conexiones en el robot se usaron cable puentes de diferentes medidas y colores, macho-hembra y macho-macho.

Asimismo, en el anexo 7 se puede apreciar un esquema eléctrico más a detalle de las conexiones eléctricas realizadas en este robot.

Figura 53

Esquema de conexiones del micro robot



Nota. Elaboración propia desarrollada en Proteus 8.

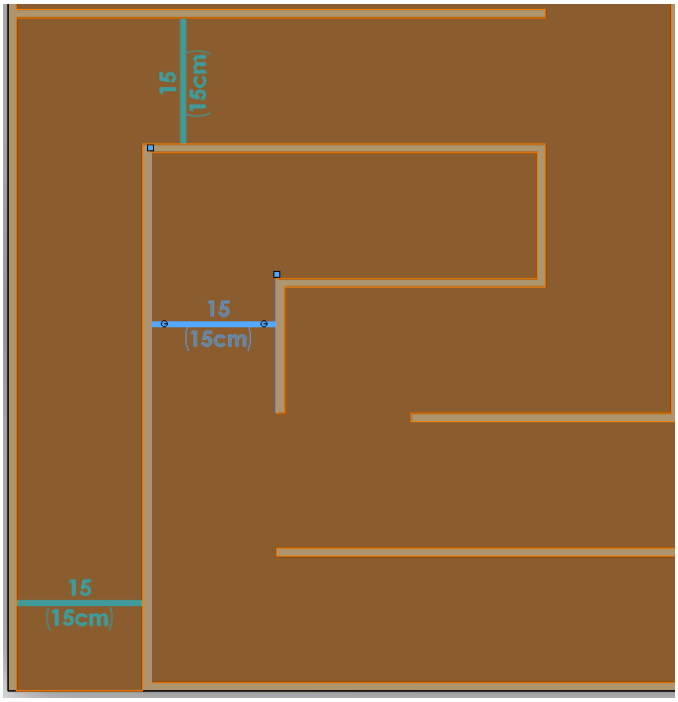
5.2.5. Diseño del Laberinto

Las primeras pruebas con el robot se realizan con simples tablones de madera, probando los diferentes problemas con los que se podría encontrar el robot dentro de un laberinto. Una vez que se obtienen satisfactoriamente los resultados esperados se procede al diseño y construcción del laberinto con el fin de programar y ajustar debidamente el robot para que logre el objetivo principal de este proyecto.

Por ello se procedió a diseñar el laberinto en el programa de SolidWorks, como el robot tiene una medida de 11cm, la anchura del camino o carril se determinó que sea de 15cm como se observa en la figura 54 para que el robot puede circular libremente y evite chocarse en las paredes.

Figura 54

Medidas de cada carril del laberinto

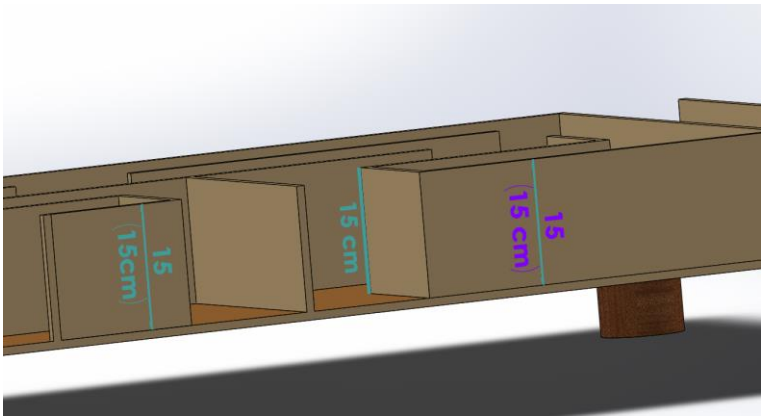


Nota. Elaboración Propia

Ahora la altura del robot es de 12 cm por lo que el laberinto debe tener una altura mayor para que los sensores ultrasónicos puedan funcionar perfectamente, tomando en cuenta esta observación la altura ideal sería de 15cm como se indica en la figura 55.

Figura 55

Altura del laberinto

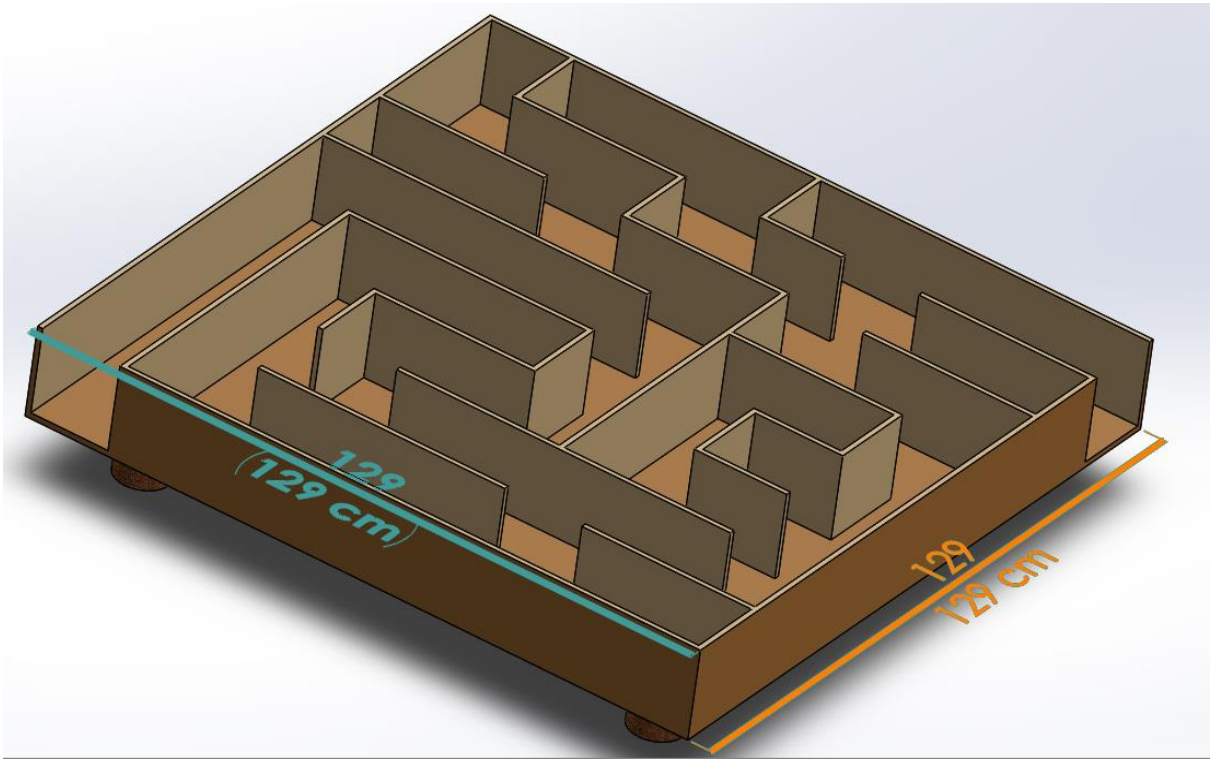


Nota. Elaboración Propia

Finalmente tomando en cuenta todas estas observaciones de altura y distancias se decide construirlo con madera MDF, su configuración es en forma matricial de 8 x 8 dándonos como medidas un total de 129 cm de largo y 129 cm de ancho como se puede apreciar en la figura 56.

Figura 56

Medidas totales del laberinto

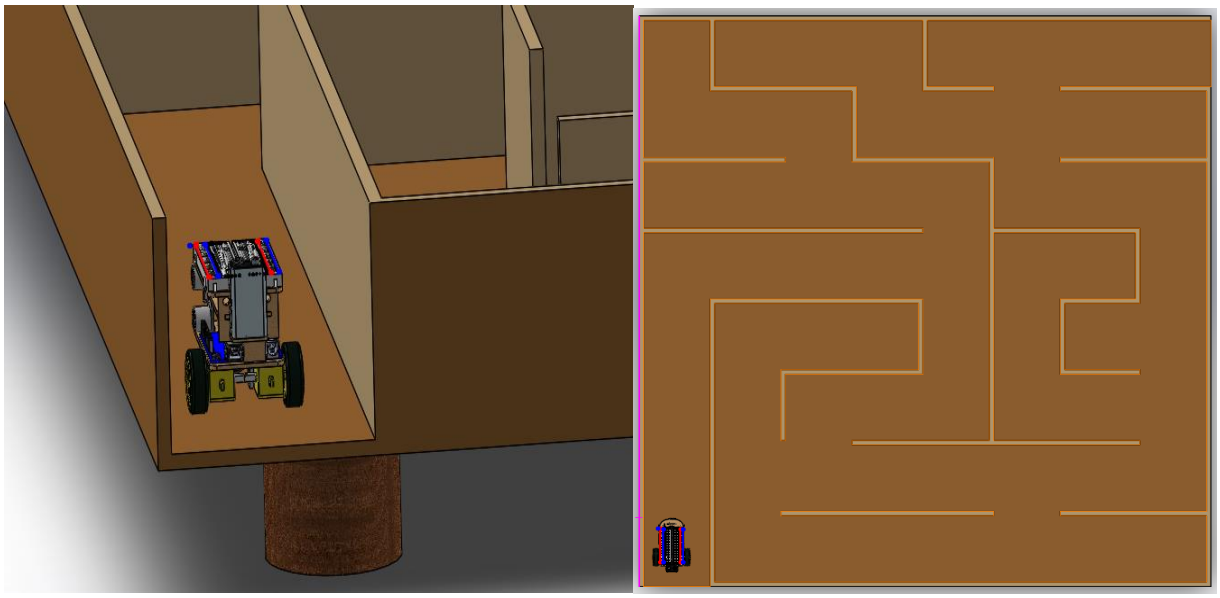


Nota. Elaboración Propia

Por último, como se observa en la figura 57 el robot construido cabe y circula perfectamente en los caminos el laberinto, como en sus curvas, intersecciones y rectas.

Figura 57

Implementación del robot en el laberinto diseñado.



Nota. Elaboración Propia

5.2.6. Pruebas Realizadas

En este apartado se presenta todo lo referente a las pruebas realizadas sobre el robot móvil tanto en el hardware como en el software y su funcionamiento en conjunto, las pruebas se pueden dividir en tres partes que son pruebas de hardware, de software puesto que solo de esa manera se puede probar y demostrar el funcionamiento global, además del algoritmo seleccionado sobre el entorno de pruebas (laberinto), de los cuales se consiguieron diferentes resultados que son presentados y detallados a continuación:

5.2.6.1. Pruebas de Software. La preparación del programa IDE sufrió muchas modificaciones antes de conseguir óptimos resultados para que el robot pueda moverse satisfactoriamente dentro del laberinto y van a la par de la variación del hardware, debido a que el aumento de sensores involucra también un cambio en la programación, pero lo más relevante fueron las pruebas con los diferentes algoritmos de desplazamiento que se conocen y que se intentaron implementar, hasta conseguir el que mejor se adecuó al sistema del robot.

5.2.6.2. Pruebas sobre el laberinto. Una vez finalizada la creación del robot móvil con las partes necesarias como los circuitos de acondicionamiento de los sensores ultrasónicos, control de motores y con el programa de control cargado, se procede a probar el funcionamiento de dicho robot con lo que se sacaron muchas conclusiones que fueron satisfactorias en base a los resultados que se buscaban alcanzar.

Al inicio se trabajaba con una alimentación conectada directo de la placa arduino, tanto para el control como para la potencia lo que provocaba que al momento en que los motores arrancaban la placa de control se resetee, esto era debido al consumo de corriente que se daba en el arranque, por lo que la mejor solución para corregir este problema fue separar la alimentación mediante conexiones en el protoboard.

Ya puesto en funcionamiento, ahora si el robot sigue una trayectoria lo suficientemente recta como para no colisionar contra las paredes del laberinto y se desplaza sin problemas dentro del mismo. Superadas con éxito todas las pruebas realizadas se considera que el proyecto ejecuta los requerimientos primordiales exigidos previamente, así como los objetivos determinados antes de iniciar el proyecto, por lo que se considera que el robot es apto para su uso.

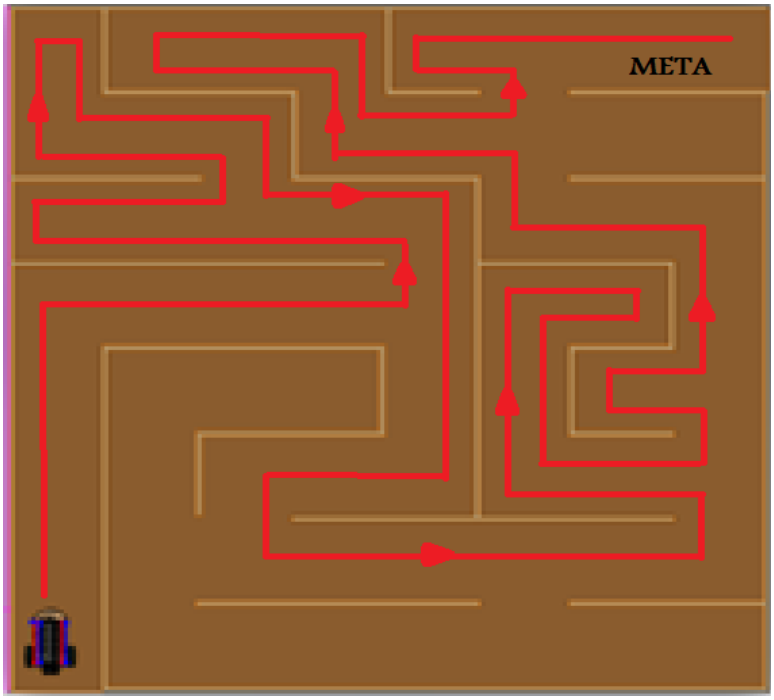
5.2.7. Análisis de los resultados obtenidos

Para esta evaluación se describen los resultados obtenidos en el entorno de pruebas (laberinto), así como el tiempo que le toma al micro robot resolver el laberinto y tratar de mejorar ese tiempo.

5.2.7.1. Primera Prueba. Para la primera prueba, el prototipo estaba cargado con el algoritmo de la mano izquierda, lo que le tomaba 3 minutos en resolver el laberinto, para este algoritmo el prototipo se desplazó por todo el laberinto siguiendo la pared izquierda incluyendo caminos sin salida, a continuación, se muestra el trayecto que transitó el robot durante el primer recorrido.

Figura 58

Recorrido durante la primera prueba



Nota. Elaboración Propia

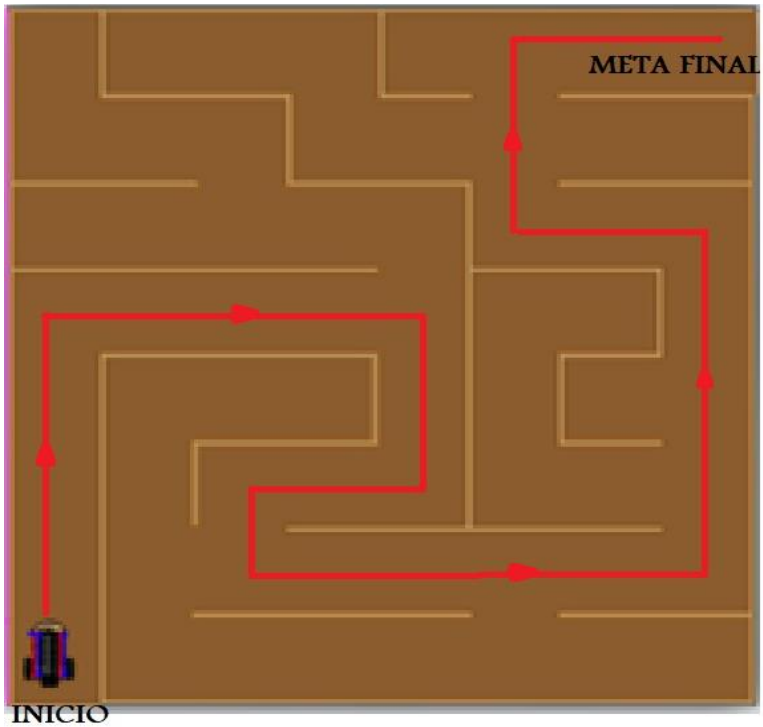
Como se puede observar en la figura 58, el robot encontró la salida del laberinto, pero el prototipo entra a 6 caminos sin salida haciendo que el tiempo sea demasiado largo, por lo que este algoritmo de la mano izquierda sea descartado.

5.2.7.2. Segunda Prueba. Durante la segunda prueba el tiempo que le tomó al robot en solucionar el laberinto mejoró a 50 segundos, por lo que este algoritmo es una muy buena opción y nos garantizará resolver cualquier laberinto desconocido, eso debido a que se utilizó el algoritmo de la mano derecha como se puede apreciar en la figura 61, si bien es un tiempo muy bueno comparado con la primera prueba, se decidió realizar una tercera prueba utilizando un algoritmo que mejore aún as el tiempo en resolver el laberinto**.**

5.2.7.3. Tercera Prueba. En esta última prueba el robot encontró la salida del laberinto en 35 segundos, para ello se aplicó un algoritmo que sigue en línea recta mientras el sensor frontal tenga una distancia mayor a 7 cm, pero cuando el sensor frontal detecta pared, el robot hace una comparación de distancias entre el sensor derecho e izquierdo y elige la distancia mayor y sigue su recorrido evitando así caminos sin salida.

Figura 59

Recorrido durante la tercera prueba



Nota. Elaboración Propia

Como se puede apreciar en la figura 59 este algoritmo es el que realiza la trayectoria más corta de todos los algoritmos, si bien este sería la mejor opción de todas, no nos garantizará resolver cualquier laberinto desconocido, a excepción del que tenemos construido por lo que esta opción también queda descartada.

6. Resultados

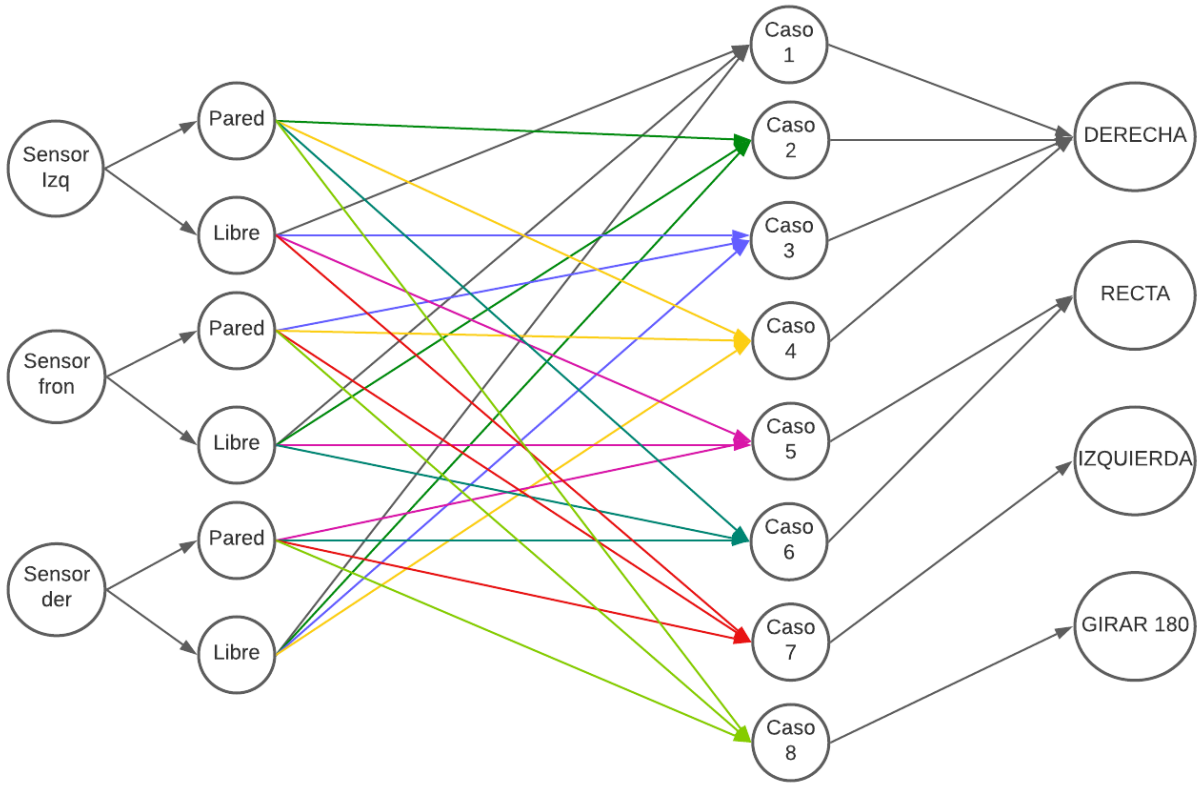
En el presente capítulo se describirá el producto final como es el robot móvil objeto de cumplimiento del presente proyecto, teniendo en cuenta tanto el apartado físico, como el de software. Además, se explicarán todos los aspectos que se consideren precisos para su correcta comprensión, apoyándose tanto en referencias al código como en flujogramas o esquemas. En cuanto al apartado físico, para describirlo correctamente, se utilizarán tanto imágenes del mismo como referencias a los planos.

6.1. Algoritmo Implementado

El algoritmo implementado en el prototipo es el de la mano derecha que se observa en la figura 60, este realiza el seguimiento de la pared que se encuentra a la derecha. Para el algoritmo escogido, el robot siempre seguirá la pared derecha mientras la tenga, si no existe la pared derecha el robot dará un giro de 90 grados a la derecha, este proceso se realizará de forma recursiva hasta encontrar la salida del laberinto.

Figura 60

Red neuronal del algoritmo implementado



Nota. Elaboración Propia

Caso 1: Después de detectar camino libre al frente, a la izquierda y a la derecha, el robot gira 90 grados a la derecha siguiendo la lógica del algoritmo de la mano derecha.

Caso 2: Tras detectar camino libre al frente y a la derecha y un obstáculo o pared a la izquierda el robot deberá girar obligatoriamente hacia la derecha.

Caso 3: El robot tomará la decisión de girar hacia la derecha cuando los sensores ultrasónicos detecten un obstáculo en la parte frontal, pero a la izquierda y a la derecha descubra camino libre.

Caso 4: Cuando los sensores ultrasónicos detecten un obstáculo tanto en la parte frontal como en la izquierda y a la derecha descubra camino libre el robot tomará la decisión de girar hacia su derecha.

Caso 5: Después de detectar camino libre al frente y a la izquierda, pero a la derecha el sensor detecta un obstáculo o pared el robot seguirá necesariamente en línea recta mientras este caso se cumpla.

Caso 6: Una vez que los sensores laterales de la izquierda y derecha revelen obstáculos, pero el sensor frontal detecte camino libre, el robot seguirá obligatoriamente en línea recta mientras este caso se cumpla.

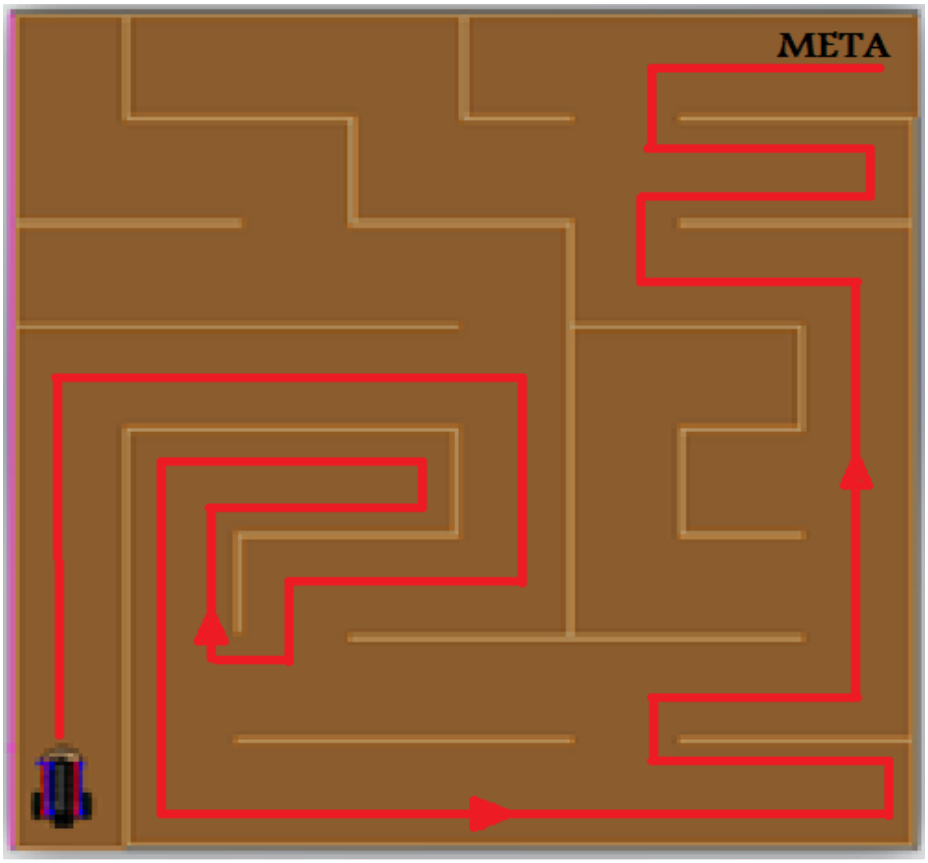
Caso 7: El robot sólo deberá girará hacia la izquierda cuando los dos sensores tanto el frontal como el derecho detecten un obstáculo, pero a la izquierda el sensor descubra camino libre.

Caso 8: En este caso el robot obligatoriamente deberá dar un giro de 180 grados debido a que todos los sensores ultrasónicos tanto el frontal, el izquierdo y el derecho van a detectar obstáculos o paredes.

La solución que se obtiene utilizando el algoritmo de la mano derecha se puede observar en la figura 61, el tiempo que le tomó al robot recorrer este camino es de 50 segundos.

Figura 61

Camino recorrido con el algoritmo de la mano derecha



Nota. Elaboración Propia

6.2. Programación

La programación como se puede apreciar en el anexo 2, recoge todos aquellos códigos necesarios para que el robot funcione correctamente en el entorno de pruebas y pueda llevar a cabo las funciones descritas anteriormente y necesarias para encontrar la salida del laberinto.

6.2.1. Control del Robot

En este apartado se explicará el código principal del proyecto, que es el encargado de hacer funcionar a los diferentes componentes que conforman al robot para que este funcione correctamente y alcance sus objetivos. Aunque no esté desarrollado en ese orden exactamente, se explicará el código siguiendo un orden diferente que se considera el adecuado para su mejor entendimiento.

- ✓ Declaración de la librería Ultrasonic.h (figura 62), que nos servirá para controlar nuestros sensores de ultrasonido de una manera más eficaz y precisa.

Figura 62

Librería utilizada para controlar los sensores ultrasónicos

```
#include <Ultrasonic.h>
```

Nota. Elaboración propia

- ✓ Se definen los pines PWM de la placa arduino con los sentidos de giro del motor (figura 63), el pin 5 es para controlar el giro hacia adelante del motor derecho, el pin 6 es para controlar el giro hacia atrás del motor derecho, el pin 10 es para controlar el giro hacia adelante del motor izquierdo y el pin 11 es para controlar el giro hacia atrás del motor izquierdo.

Figura 63

Pines PWM seleccionados para controlar los motores

```
#define MDER_ADELANTE 5 //derecho
#define MDER_ATRAS 6
#define MIZQ_ADELANTE 10// izquierdo
#define MIZQ_ATRAS 11
```

Nota. Elaboración propia

- ✓ Declaramos los pines de los sensores ultrasónicos tanto para la señal Tring como para la señal Echo, e ingresamos valores enteros como se puede apreciar en la figura 64, estos valores nos sirven para controlar la velocidad en las curvas y rectas que va a estar expuesto el robot, igualmente ingresamos valores de las distancias que van a detectar los sensores ultrasónicos.

Figura 64

Declaración de pines para sensores y valores de distancia

```
Ultrasonic front(8, 9); //(Tring PIN.Echo PIN)
Ultrasonic left(3, 2); //(Trig PIN.Echo PIN)
Ultrasonic right(4, 7); // (Trig PIN,Echo PIN)
int distancia_minima = 7;
int distancia_media = 10;
int distancia_max = 120;
int distancia_terminar = 100;
int led = 12;
int val=A0;
int vg = 50;
int v = 125;
int vf = 150;
int leftS, rightS, frontS;
```

Nota. Elaboración propia

- ✓ Se declara los pines de salida que se va a utilizar tanto para el control de los motores como para los 3 diodos led (figura 65), que nos van a indicar cuando el robot gira a la izquierda o a la derecha.

Figura 65

Pines de salida de diodos led y motores

```
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
pinMode(5, OUTPUT);
pinMode(6, OUTPUT);
pinMode(led, OUTPUT);
```

Nota. Elaboración propia

- ✓ A continuación, en la figura 66 se puede observar una de las partes más importante del programa utilizado en el robot, porque se va a estar ejecutando como un bucle hasta que nuestro robot encuentre la salida del laberinto.
- ✓ Lo primero que ejecuta nuestra función void loop es la acción de “Reads ( );” que hace referencia a leer los tres sensores ultrasónicos constantemente. Luego ejecuta “GoFront (

);" hace que el robot se mueva hacia adelante bajo ciertas condiciones, utilizando sentencias if para que nuestro robot haga una cosa u otra, asimismo se utiliza digitalWrite (led,HIGH); para encender el diodo led color verde, azul y rojo que nos indica que el robot se mueve a la izquierda, derecha, o cuando encuentra la salida del laberinto respectivamente. Por último, se tiene la subrutina Turn ( ); hace que si el sensor ultrasónico frontal detecta una distancia menor o igual a cinco, el robot se detendrá por 2 milisegundos y ejecuta un giro sea a la izquierda o derecha dependiendo de otras condiciones que se verán más adelante.

Figura 66

Función Void Loop de nuestro código

```
void loop() {
  Serial.println("Iniciando");
  delay(3000);
  Serial.println("Listo empezamos");
  delay(3000);
  while (true) {
    ReadS();
    GoFront();
    if (frontS >= distancia_minima && frontS < distancia_max) {
      ReadS();
      Ajust();
       digitalWrite(led,LOW);
    }
    if (frontS <= 5) {
      ReadS();
      STOP();
      //delay(2000);
      ReadS();
      Turn();
    }
        if (frontS > distancia_max && rightS < distancia_media && leftS < distancia_media) {
      ReadS();
      STOP();
      digitalWrite(led,HIGH);
      delay(10000);
  }
  }
}
```

Nota. Elaboración propia

- ✓ En este apartado se procederá a dar lectura de los 3 sensores gracias a la librería antes mencionada como se observa en la figura 67, tanto el sensor derecho (rightS), frontal (frontS) e izquierdo (leftS), además, se los imprimirá para saber el valor que están obteniendo en centímetros.

Figura 67

Subrutina para dar lectura de los sensores ultrasónicos

```
void ReadS () {
  Serial.print("\n Adelante: " + String(frontS) + "cm");
  Serial.print("\n Derecha: " + String(rightS) + "cm");
  Serial.println("\n Izquierda: " + String(leftS) + "cm");
  leftS = left.read();
  rightS = right.read();
  frontS = front.read();
}
```

Nota. Elaboración propia

- ✓ Mediante sentencias if (figura 68), se efectuarán giros a la izquierda o a la derecha cuando el sensor frontal detecte distancias menores a 5cm, para ello hay cuatro casos y se explicarán a continuación:
    - 1. Si el sensor izquierdo detecta distancias menores a 15cm y el sensor derecho detecta distancias mayores a 15cm, entonces la acción que el robot deberá tomar es girar hacia la derecha.
    - 2. Si el sensor izquierdo detecta distancias mayores a 15cm y el sensor derecho detecta distancias mayores a 15cm, entonces la acción que el robot deberá tomar es girar hacia la derecha.
    - 3. Si el sensor izquierdo detecta distancias mayores a 15cm y el sensor derecho detecta distancias menores a 15cm, entonces la acción que el robot deberá tomar es girar hacia la izquierda.
    - 4. Si el sensor izquierdo detecta distancias menores a 15cm y el sensor derecho también detecta distancias menores a 15cm, entonces el robot deberá realizar un giro de 180 grados porque se encuentra en un camino sin salida.

Figura 68

Función void Turn para realizar los giros del robot

```
void Turn() {
   if (leftS <= 15 && rightS > 15 ) {
    GoRight();
  }
  if (leftS > 15 && rightS > 15 ) {
    GoRight();
  }
  if (rightS <= 15 && leftS > 15 ) {
    GoLeft();
  }
if (rightS < 15 && leftS < 15 && frontS < 15 ) {
    GoBack();
  }
}
```

Nota. Elaboración propia

- ✓ Para prevenir que el robot choque con las paredes del laberinto, utilizamos < *void Ajust ( )* > como se puede apreciar en la figura 69, logrando que si el sensor ultrasónico derecho detecta distancias menores a 3 centímetros entonces el motor derecho girará hacia adelante mientras que el motor izquierdo girará hacia atrás, ahora si el sensor ultrasónico izquierdo detecta distancias menores a 2 centímetros entonces el motor derecho girará hacia atrás, mientras que el motor izquierdo girará hacia adelante.

Figura 69

Función void Ajust, para corregir los choques del robot

```
void Ajust() {
  Serial.println("front");
  GoFront();
  if (rightS <= 3) {
    analogWrite(MDER_ADELANTE, vf + 5);// derecho
    digitalWrite(MDER_ATRAS, LOW);
    analogWrite(MIZQ_ATRAS, 10);
    digitalWrite(MIZQ_ADELANTE, LOW);
  }
  if (leftS <= 2) {
    analogWrite(MDER_ADELANTE, LOW);// derecho
    digitalWrite(MDER_ATRAS, 5);
    analogWrite(MIZQ_ATRAS, LOW);
    digitalWrite(MIZQ_ADELANTE, 5);
  }
 }
```

Nota. Elaboración propia

- ✓ Para que el robot se mueva hacia adelante necesitamos utilizar esta subrutina (figura 70), que controla los sentidos de giro del motor en este caso los motores izquierdo y derecho van a girar hacia adelante.

Figura 70

Control de giro de motores hacia adelante

```
void GoFront() {
  digitalWrite(MDER_ATRAS, LOW);
  analogWrite(MDER_ADELANTE, vf);
  analogWrite(MIZQ_ADELANTE, vf-11);
  digitalWrite(MIZQ_ATRAS, LOW);
}
```

Nota. Elaboración propia

- ✓ Void GoBack ( ), se utilizará cuando el robot se encuentre en un camino o callejón sin salida (figura 71), para ello los motores van a girar en sentido contrario el motor izquierdo va a girar hacia adelante mientras que el motor derecho va a girar hacia atrás, hasta que el robot de un giro de 180 grados.

Figura 71

Control de giro del robot de 180 grados

```
void GoBack() {
STOP();
  analogWrite(MDER_ATRAS, 120);
  digitalWrite(MDER_ADELANTE, LOW);
  digitalWrite(MIZQ_ATRAS, LOW);
  analogWrite(MIZQ_ADELANTE, 100);
  delay(500);
}
```

Nota. Elaboración propia

- ✓ En la figura 72 se puede apreciar la función Void GoLeft ( ), que se utilizará cuando el robot se encuentre en un camino donde los sensores frontal y derecho se encuentren obstruidos, y el sensor izquierdo libre. Para ello los motores van a girar en sentido contrario y a una velocidad diferente, el motor izquierdo va a girar hacia adelante mientras que el motor derecho va a girar hacia atrás, hasta que el robot de un giro de 90 grados.

Figura 72

Control de giro del robot hacia la izquierda

```
void GoLeft() {
  STOP();
  analogWrite(MDER_ADELANTE, v+15);
  digitalWrite(MDER_ATRAS, LOW);
  analogWrite(MIZQ_ATRAS, vg);
  digitalWrite(MIZQ_ADELANTE, LOW);
  delay(500);
}
```

Nota. Elaboración propia

- ✓ La función Void GoRight ( ), que se observa en la figura 73 se utilizará cada vez que el sensor derecho detecte distancias mayores a 10 centímetros para ello los motores van a girar en sentido contrario y a una velocidad diferente, el motor izquierdo va a girar hacia atrás mientras que el motor derecho va a girar hacia adelante, hasta que el robot de un giro de 90 grados.

Figura 73

Control de giro del robot hacia la derecha

```
void GoRight() {
  STOP();
  digitalWrite(MDER_ADELANTE, LOW);
  analogWrite(MDER_ATRAS, vg);
  digitalWrite(MIZQ_ATRAS, LOW);
  analogWrite(MIZQ_ADELANTE, v);
  delay(500);
}
```

Nota. Elaboración propia

- ✓ Por último, en la figura 74 se muestra la función Void STOP ( ), esta se utilizará cuando el robot encuentre la salida del laberinto, esto se logra a partir que el sensor frontal detecte distancias mayor a 150 centímetros para ello los motores van a dejar de girar y un diodo LED se encenderá indicando que el robot encontró la salida.

Figura 74

Motores apagados al encontrar la salida del laberinto

```
void STOP() {
  digitalWrite(MDER_ATRAS, LOW);
  digitalWrite(MDER_ADELANTE, LOW);
  digitalWrite(MIZQ_ATRAS, LOW);
  digitalWrite(MIZQ_ADELANTE, LOW);
}
```

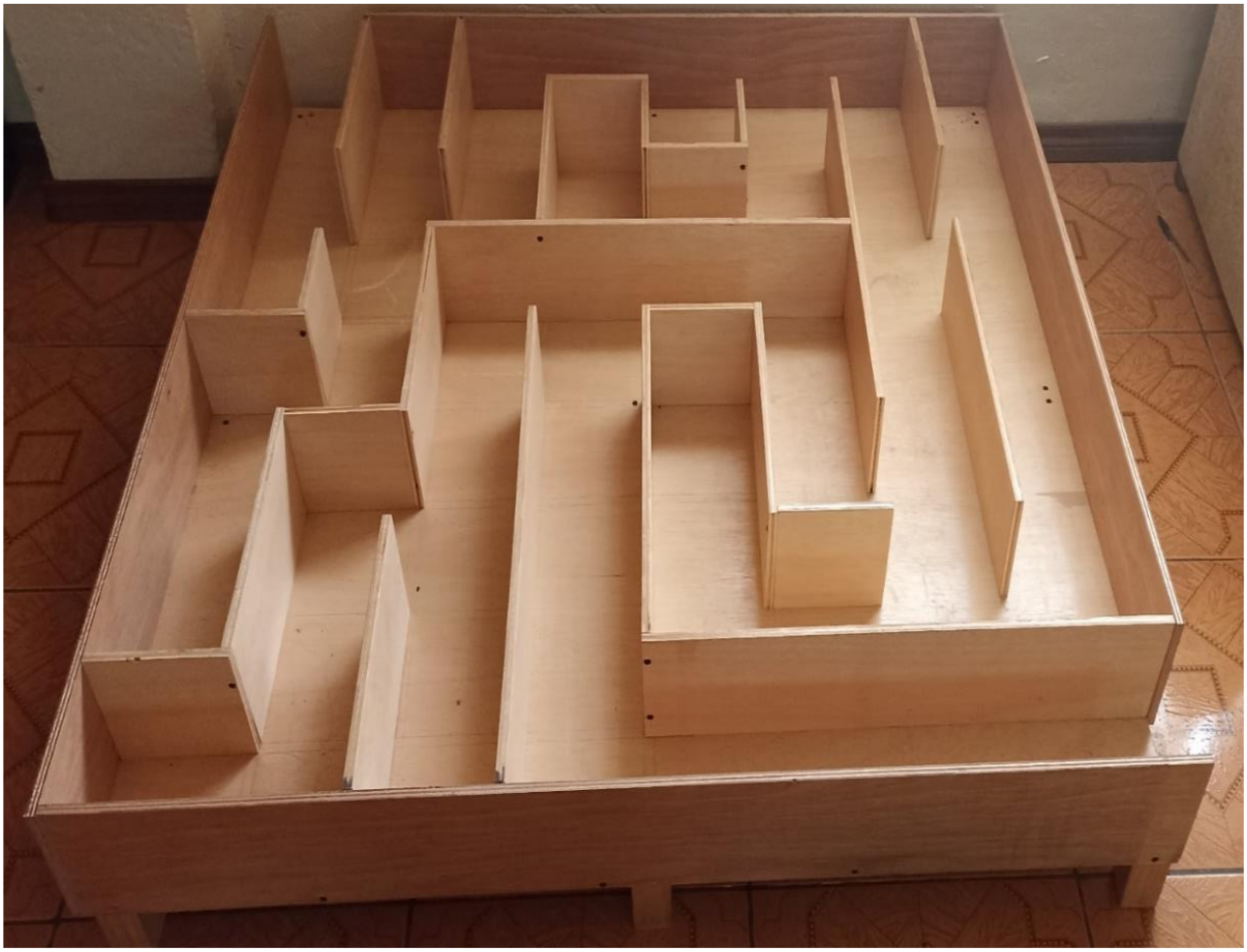
Nota. Elaboración propia

6.3. Construcción del Laberinto

El laberinto fue construido con las respectivas medidas diseñadas en SolidWorks, anteriormente mencionadas en su apartado, el material utilizado es madera MDF de 10 milímetros de espesor empleadas en las paredes y en su base como se observa en la figura 75, además se utilizaron listones de madera en la parte inferior del laberinto para dar soporte a toda la estructura, por ultimo las paredes del laberinto están sujetas con tornillos laminex N6, para que su desmontaje sea más fácil y se pueda reconfigurar el laberinto moviendo las piezas según se desee.

Figura 75

Laberinto construido con madera MDF



Nota. Elaboración propia

6.4. Costos del Proyecto

Los gastos totales que conlleva la elaboración de este proyecto como son la construcción del micro robot y el laberinto, así como los gastos en ingeniería son de $ 1167,25 dólares. En la tabla 6 se describen estos costos.

Tabla 6

Costos totales del proyecto

Descripción	Cantidad	Valor Unitario ($)	Valor Total ($)
Costos que conlleva la construcción del micro robot	1	103,75	103,75

Costos que conlleva la construcción del laberinto	1	63,50	63,50
Gastos de ingeniería	1	1000	1000
TOTAL ($)			**1167,25**

Nota. Elaboración propia

6.4.1. Costos que conlleva la construcción del micro robot

En la Tabla 7 se presentan todos los gastos que implicó la construcción del prototipo del robot móvil, como son la adquisición de la plataforma móvil, costos por la construcción de las placas de los circuitos de control y acondicionamiento, compra de los sensores, servomotor, además de diversos elementos que se consideraron en la construcción, también se toma en cuenta los gastos de ingeniería que es todo lo concerniente al diseño, adaptación de los diferentes mecanismos, creación del programa de control e implementación de la plataforma móvil así como también el tiempo que se tardó en el perfeccionamiento del robot.

Tabla 7

Lista de elementos y costos que conlleva la construcción del micro robot

Descripción	Cantidad	Valor Unitario ($)	Valor Total ($)
Placa de Arduino UNO	1	25	25
Resistencias	10	0,10	1
Chasis	1	6	6
Motorreductores	2	7	14
Conectores	20	0,15	3
Diodos LED	5	0,15	0,75
Pulsadores	2	0,25	0,50
Tarjeta de control de motores	1	12	12
Sensores Ultrasónicos	3	5	15
Baterías	2	11	22
Protoboard	1	4,50	4,50
TOTAL ($)			**103,75**

Nota. Elaboración propia

6.4.2. Gastos de Ingeniería

Sobre los gastos de ingeniería se han tomado en cuenta numerosos factores como son, la investigación sobre el sistema que controla al micro robot, conocer el software que permitirá la programación de dichos componentes y todas sus herramientas de soporte, además de las horas efectivas que se ha trabajado sobre el prototipo en cuanto al diseño, construcción y adaptación de nuevos sistemas para permitir el funcionamiento de la plataforma móvil y perfeccionamiento del programa de control para que de esta manera su trabajo en conjunto sea óptimo, tomando en cuenta todos estos factores los gastos por ingeniería son de $1000 dólares, tomando en cuenta un costo por hora efectiva de trabajo de 10 USD.

6.4.3. Costos de construcción del laberinto

En la tabla 8 se exponen todos los gastos que involucró la construcción del entorno de pruebas (laberinto), como son el material de las paredes, los tornillos de sujeción, los soportes, entre otros materiales.

Tabla 8

Lista de elementos y costos que conlleva la construcción del laberinto

Descripción	Cantidad	Valor Unitario ($)	Valor Total ($)
Plancha PLYWOOD de 4x8	1	55	55
Tornillo Laminex Nº6	50	0,05	2,50
Lija para madera grano 80	2	0,75	1,50
Liston de madera	1	4,50	4,50
TOTAL ($)			**63,50**

Nota. Elaboración propia

7. Discusión

El presente proyecto de tesis se desarrolló para conocer el trabajo y aplicaciones que ejerce un robot resuelve laberintos en esta era tecnológica y en la industria, muy poco explotadas en nuestro ciudad y universidad, es por esto que se planteó el diseño y la construcción de un prototipo económico y fácil de construir en nuestro medio local.

Para el desarrollo y elaboración de este tipo de robot resultó necesaria la revisión e investigación de los diferentes métodos y algoritmos de solución de laberintos en que se basa el desarrollo de este proyecto, se analizaron cada una de las condiciones técnicas dentro de la cual se desenvuelve, así mismo se diseñó un sistema electrónico y mecánico que permita desarrollarlo y aplicarlo de manera funcional y viable.

En el bosquejo y construcción de este tipo de robots móviles se pudo conocer y establecer los parámetros de diseño, eléctrico, mecánico y de programación que se debe tener presente en el diseño del prototipo, de esta manera se implementa un sistema de locomoción tipo triciclo un sistema muy útil encargado del movimiento del robot, haciendo posible cambiar su orientación sin movimientos de traslación. Con esto se anhela dar un referente para iniciar investigaciones dentro de este sistema y así mejorar los diseños.

Al realizar las simulaciones de algunos algoritmos de solución de laberintos en el programa de Karel, se pudo determinar que el algoritmo usado en nuestro proyecto de la mano derecha, nos garantizará resolver cualquier laberinto desconocido, si no se hubiese realizado una simulación la complejidad de la selección y programación de cualquier algoritmo hubiese aumentado de manera considerable, puesto que programar cada algoritmo e implementarlo en el prototipo el gasto económico hubiese aumentado, igual que el tiempo en desarrollar todas estas opciones de algoritmo que existen en la solución de laberintos.

Los resultados obtenidos en este proyecto son muy satisfactorios, debido a que cumple su objetivo de resolver los caminos posibles para solucionar el laberinto en 50 segundos, por otro lado se considera que esta propuesta de robot laberinto es innovadora, porque cuenta con medidas muy pequeñas, ajustadas a las dimensiones de los componentes electrónicos, mecánicos y eléctricos escogidos en el proyecto. Además, existen proyectos que plantean utilizar cuatro motores, a diferencia del diseño que se propone que son de dos motores y una rueda loca, asimismo el prototipo maneja tres sensores de ultrasonido para anticiparse a la toma de decisiones en cada encrucijada e intersección, puesto que utilizar solo un sensor de ultrasonido hace que el robot se

demore más en encontrar la salida del laberinto, debido a que tiene que girar 180 grados en cada toma de decisión, por último el robot cuenta con luces led para indicarnos cuando el robot gira a la izquierda, derecha y también cuando encuentra la salida.

Sabiendo cada uno de los materiales y componentes que se van a utilizar en el proyecto y sus costos, se estableció el presupuesto del prototipo dando un total de $1167,25 USD. En el mercado ecuatoriano no se encuentran robots móviles dedicados a solucionar laberintos, sin embargo, en Estados Unidos se encuentran robots parecidos de similares características que superan los $200,00 dólares. El prototipo construido es de tamaño más pequeño en comparación a estos robots móviles, esto es debido a que a la hora de diseñar nuestro prototipo predominó el tamaño de sus componentes, ocupando cada lugar de manera eficaz con esto se logró abaratar un poco los costos de construcción.

A nivel nacional existe un desinterés en esta línea de investigación dando como resultado que nuestro país se convierta en uno de los menos tecnológicos, sin embargo, existen algunos prototipos funcionales que cuenta con características muy diferentes especialmente en el tamaño, cantidad de sensores ultrasónicos y configuración de los mismos, haciendo que el algoritmo varíe en cada prototipo.

En la Universidad Nacional de Loja por parte de los estudiantes se han desarrollado muy pocos proyectos que involucren micro robots móviles, y menos aún robots dedicados a resolver laberintos, por último, en nuestro medio local se desconoce que existan competencias que permitan fomentar el desarrollo de este tipo de robots y tecnologías.

8. Conclusiones

- El modelo matemático que nos garantizará resolver cualquier laberinto desconocido, es el algoritmo de la mano derecha, usado en el prototipo.
- Mediante la construcción del micro robot y del laberinto se logró validar el algoritmo de la mano derecha, por lo que el robot necesitó un tiempo de 50 segundos para atravesar y encontrar la salida del laberinto, el mismo que también se simuló en los programas de Karel y Tinkercad.
- El laberinto que se construyó tiene medidas de 1.3 metros de ancho y de 1.3 metros de largo, el cual está elaborado con madera MDF de 1 centímetro de espesor y tiene un diseño en forma matricial de 8 x 8, dándonos como medidas en cada carril de 15 centímetros de ancho durante todo el recorrido del laberinto.
- Los costos juegan un papel muy importante a la hora de tomar decisiones por ello luego de haber analizado las características de cada componente seleccionado como la capacidad en mAh de la batería, cantidad de puentes h en el controlador L298N, el torque de los motores y el material en la elaboración del laberinto, el costo total del proyecto asciende a los $1167,25 USD.

9. Recomendaciones

- ✓ Se recomienda el uso de sensores de distancia que sean infrarrojos por su tiempo de respuesta fiabilidad en la medición, permitiendo al prototipo tener una mejor captura de datos y detectar correctamente todos los casos.
- ✓ Es importante tomar en cuenta el lugar donde se van a colocar los sensores ultrasónicos, por ello se recomienda ubicarlos en la parte frontal del robot, de forma diagonal apuntando hacia la derecha, izquierda y al frente para saber de manea anticipada a que caso se aproxima y de esta manera evitar choques del robot con la pared.
- ✓ Se recomienda realizar programas sencillos para probar los sensores ultrasónicos al igual que los motores, antes de colocarlos en nuestro prototipo, debido a que algunos sensores y motores pueden tener inconvenientes de fábrica.
- ✓ Se recomienda utilizar baterías recargables mínimo de 800mAh puesto que, al trabajar con dos motores, las tarjetas de control y algunos sensores, provocan un gran consumo de corriente lo que hace que las baterías se descarguen rápidamente y se corre el riesgo de que en algún momento el voltaje entregado el sistema de control no sea el suficiente para hacer que este trabaje regularmente.
- ✓ Para un trabajo futuro, con el objetivo de que la velocidad no disminuya en el prototipo mientras la batería se descarga, se sugiere diseñar e implementar un control PID para mantener la velocidad constante del robot en función del voltaje que la batería disponga.
- ✓ Utilizar algún material como lija número 80 en la circunferencia de las ruedas, para tener un mejor agarre en la pista del laberinto que es fabricado con madera MDF y evitar que el prototipo patine en las curvas.
- ✓ Se sugiere que se diseñe la PCB del circuito con esto se reduce aún más las medidas y tamaño del robot, además esto permite disminuir peso y da mayor agilidad al robot dentro del laberinto.
- ✓ Se propone que en un futuro el robot sea construido con otra tarjeta microcontroladora, de mejores prestaciones que la tarjeta de arduino UNO, como por ejemplo la placa raspberry pi 4 modelo b, para almacenar en su memoria los diferentes recorridos que el robot puede seguir, para ello se puede utilizar el programa de control que se encuentra en el anexo 6, este programa contiene vectores para guardar las trayectorias que sigue el robot cuando navega por el laberinto y mediante la lógica de aprendizaje escoger exclusivamente la ruta

correcta, por lo cual cada vez que se ingrese en una ruta o a un camino sin salida esta deberá ser eliminada del Arreglo de Estados.

10. Bibliografía

Abellàn, J. (21 de Febrero de 2016). *Rincòn de Maxwell.* Obtenido de Modulaciòn por ancho de pulso (PWM): https://elrincondemaxwell.wordpress.com/2016/02/21/modulacion-por-ancho-de-pulso-pwm/

Acuña Regalado, C. O., & Paredes Escobar, E. P. (30 de Agosto de 2016). DISEÑO Y CONSTRUCCIÓN DE UN ROBOT MÓVIL QUE PUEDA DESPLAZARSE DENTRO DE UN LABERINTO. Quito, Pichincha, Ecuador: ESCUELA POLITÉCNICA NACIONAL.

Aguilar Castillo, W. G., & Enriquez Astudillo , P. D. (2008). *DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO MÓVIL CON CAPACIDAD DE APRENDIZAJE PARA LA NAVEGACIÓN EN LABERINTOS.* Quito: Escuela Politécnica del Ejército.

Andaluz Ortiz, G. M. (2011). *Automatizaciòn Industrial Sistemas de Control.* Quito: scuela Politècnica Nacional.

Apaza Condori, D. (2017). *Microcontrolaores PIC fundamentos y aplicaciones un enfoque didàctico.* Arequipa: Universidad Autònoma San Francisco.

Arduino. (23 de Octubre de 2020). *Arduino.cl*. Obtenido de Software de Arduino: https://arduino.cl/programacion/

Atria Innovation. (22 de Octubre de 2019). *Qué son las redes neuronales y sus funciones*. Obtenido de Atria Innovation: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

Basogain, X. (2008). *REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES.* Bilbao, España: Escuela Superior de Ingenierìa de Bilbao.

Baturone, A. O. (2001). *ROBÒTICA Manipuladores y robots mòviles.* Barcelona, España: Marcombo Boixareu Editores.

Berger, R. (7 de Marzo de 2019). *Industria 4.0 para LUPEON*. Obtenido de LUPEON: https://lupeon.com/2019/03/industria-4-0-para-lupeon-la-impresion-3d-como-herramienta-fundamental-en-la-cuarta-revolucion-industrial/

Cepeda , C. (13 de Abril de 2015). *Olimpiada Mexicana de Informática.* Obtenido de Tutorial de Karel el Robot: https://www.olimpiadadeinformatica.org.mx/OMI/OMI_Primaria/material/EntrenamientoOMIKarel/Varios/Tutorial%20Karel.pdf

Compromiso Social Bancaja. (4 de Abril de 2011). *El laberinto de Creta.* Obtenido de Por laberintos: https://drive.google.com/file/d/10uek1BcmhfxNxR3k3HuXPLBr8kMDKSlQ/view

Costa Rodriguez, J. (Septiembre de 2017). ESCUELA UNIVERSITARIA POLITÉCNICA. *OPTIMIZACIÓN DEL ALGORITMO DE UN ROBOT TIPO LABERINO*. Coruña, España.

Cruz Ruiz, I. O., Lara Velázquez, P., & Guitierrez Andrade, M. (23 de Mayo de 2019). Un algortmo estocástico para resolver laberintos. *REVISTA DE MATEMÁTICA: TEORÍA Y APLICACIONES*, 20. doi:https://doi.org/10.15517/rmta.v26i2.38322

Dìaz, E. (21 de Enero de 2020). *Quanomy.* Obtenido de Cómo salir de un laberinto de forma eficaz: https://quonomy.com/como-salir-de-un-laberinto-de-forma-eficaz-metodos-infalibles#mactodos-para-escapar-de-un-laberinto

Dìaz, J. L., Garcìa, A., & Rìos, E. Y. (2009). IMPLEMENTACION DE UNA MODULACION PWM OPTIMIZADA PARA EL CONTROL DE UN MOTOR TRIFASICO DE INDUCCION. *Revista Colombiana de Tecnologìas de Avanzada, I*(13), 9.

Dièquez, L. (19 de Mayo de 2020). *INTRODUCCIÓN AL MOTOR DC O MOTOR DE CORRIENTE CONTINUA.* Obtenido de Kolwidi: https://kolwidi.com/blogs/blog-kolwidi/intriduccion-al-motor-dc-o-motor-de-corriente-continua

Electronics Projects. (4 de Diciembre de 2018). *PJRC Electronics Projects Components Available Worldwide*. Obtenido de Placa de desarrollo USB Teensy: https://www.pjrc.com/store/teensy40_pins.html

Enegon. (13 de Diciembre de 2020). *Enegon*. Obtenido de Baterìas Recargables: https://energizer.lat/Ecuador/product/energizer-max-baterias-9v/

Grupo Velasco. (29 de Septiembre de 2018). *Grupo Velasco*. Obtenido de Productos cable protoboard: http://www.velasco.com.ec/velasco/producto.php?id=4093

Guerra Carmenate, J. (2 de Marzo de 2021). *Programar facil*. Obtenido de Programar ESP32 con Arduino IDE: https://programarfacil.com/esp8266/programar-esp32-ide-arduino/

Irwin, D. (2004). *Anàlisis Bàsico de circuitos de Ingenierìa* (Quinta ed.). Nueva York: Prentice Hall Hispanoamèrica.

Keyence. (15 de Marzo de 2019). *Fundamentos del sensor.* Obtenido de ¿Que es un sensor ultrasònico?: https://www.keyence.com.mx/ss/products/sensor/sensorbasics/ultrasonic/info/

Mardones, J. L., & Gonzàlez, A. (7 de Abril de 2019). *Robòtica educativa, electrònica divertida.* Obtenido de Como usar el mòdulo sensor infrarrojos: https://www.web-robotica.com/arduino/conceptos-basicos-arduino/como-usar-el-modulo-sensor-de-infrarrojos-ir-fc-51-para-evitar-obstaculos-con-robot-arduinogenuino

Martín del Rey, Á. (2018). El algoritmo de Tarry o cómo salir de un laberinto a la primera. *ABC CIENCIA*, 2-6.

Neal, A. J. (20 de Enero de 2010). *SERVO MAGAZINE.* Obtenido de Tips for selecting DC motors for your mobile robot: https://www.servomagazine.com/magazine/article/tips_for_selecting_dc_motors_for_your_mobile_robot

Ocampo, M., & Catarina. (2018). *Inteligencia Artificial.* Ciudad de mèxico: Foro consultivo FCCyT.

Olmeda, O. (2016). *Trabajo de Microrrobots: SENSORES DE MEDIDA POR CONTACTO.* Ciudad de Mèxico: Ingenierìa en Telecomunicaciòn.

Ponce, J. C., Torres, A., & Silva , A. (2014). *Inteligencia Artificial* (Primera ed.). Madrid: Iniciativa Latinoamericana de Libros de Texto abiertos(LATIn).

Rodriguez, M., Pozo, D., Morales, L., Rosero, J., & Rosales, A. (Octubre de 2014). MAPEO DE LABERINTOSY BÚSQUEDA DE RUTAS CORTAS MEDIANTETRES MINI ROBOTS COOPERATIVOS. *Revista EPN, 34*(1), 6.

Rus, D. (2019). Robótica: una década de transformaciones. *OpenMind*, 6-16.

Solectro. (24 de Noviembre de 2020). *Curso de Raspberry Pi desde cero.* Obtenido de Solectro: https://solectroshop.com/img/cms/Productos%20DICIEMBRE/Gu%C3%ADa%20de%20Raspberry%20Pi%20desde%20cero%20-%20Solectro.pdf

Suarez, P., Velasco, V., & Bricaire, A. (2 de Agosto de 2006). *Discretizaciòn exacta de un robot mòvil con retardo de transporte.* Obtenido de Ciencias e Ingeniería Neogranadina: https://www.redalyc.org/pdf/911/91116104.pdf

Val Romàn, J. L. (27 de Octubre de 2016). *Conferencia de asesores y decanos de ingenierìa informàtica.* Obtenido de Induatria 4.0: la transformaciòn digital de la industria: http://coddii.org/wp-content/uploads/2016/10/Informe-CODDII-Industria-4.0.pdf

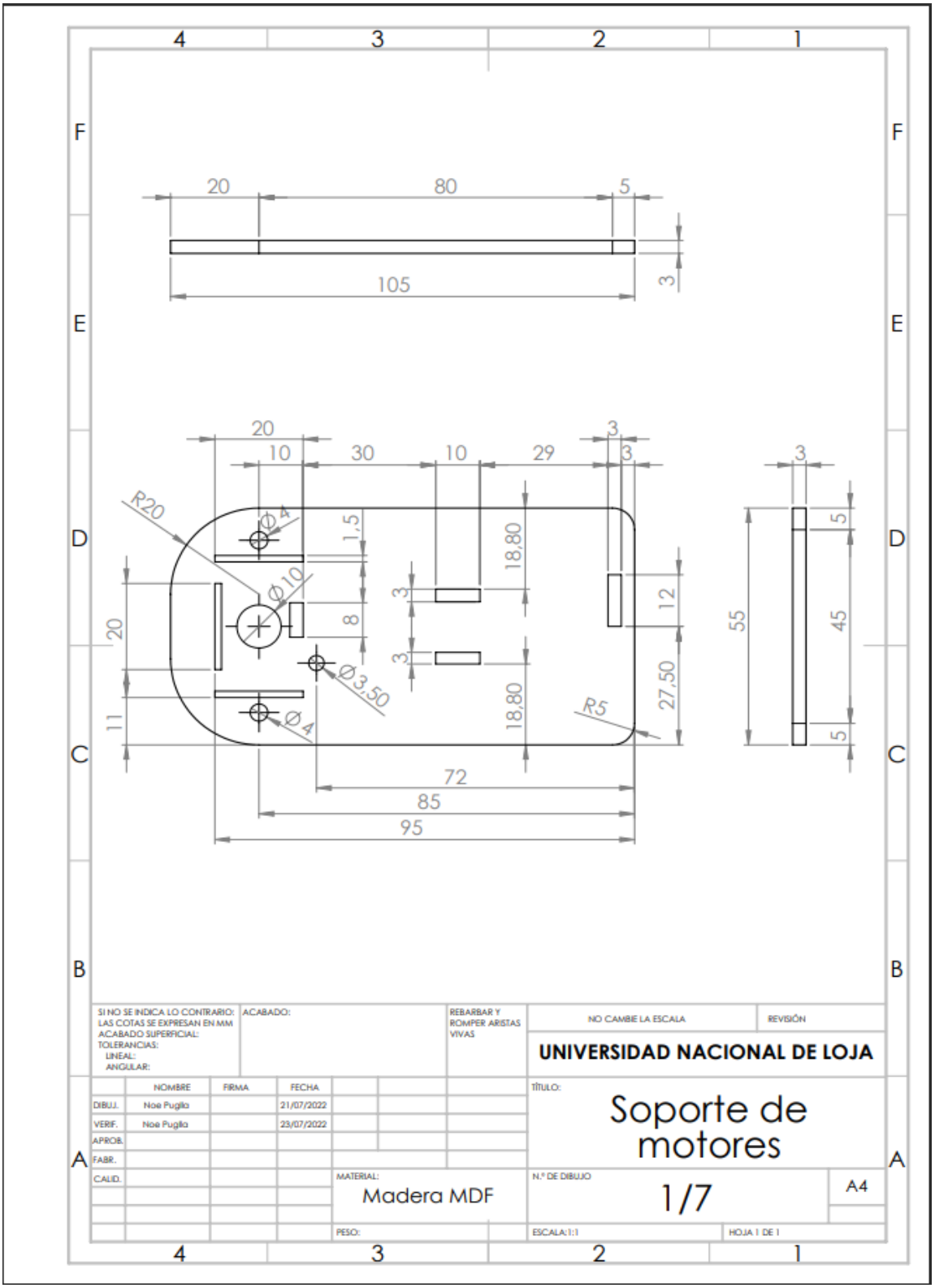
Vicuña Novillo, J., Rojas Hernàndez, D., Olivo Mazòn, B., Rìos Molina, J., & Villavicencio Càrdenas, O. (2018). *Arduino y el Internet de las Cosas* (Primera ed.). Alicante: Área de Innovación y Desarrollo,S.L. doi:http://dx.doi.org/10.17993/IngyTec.2018.45

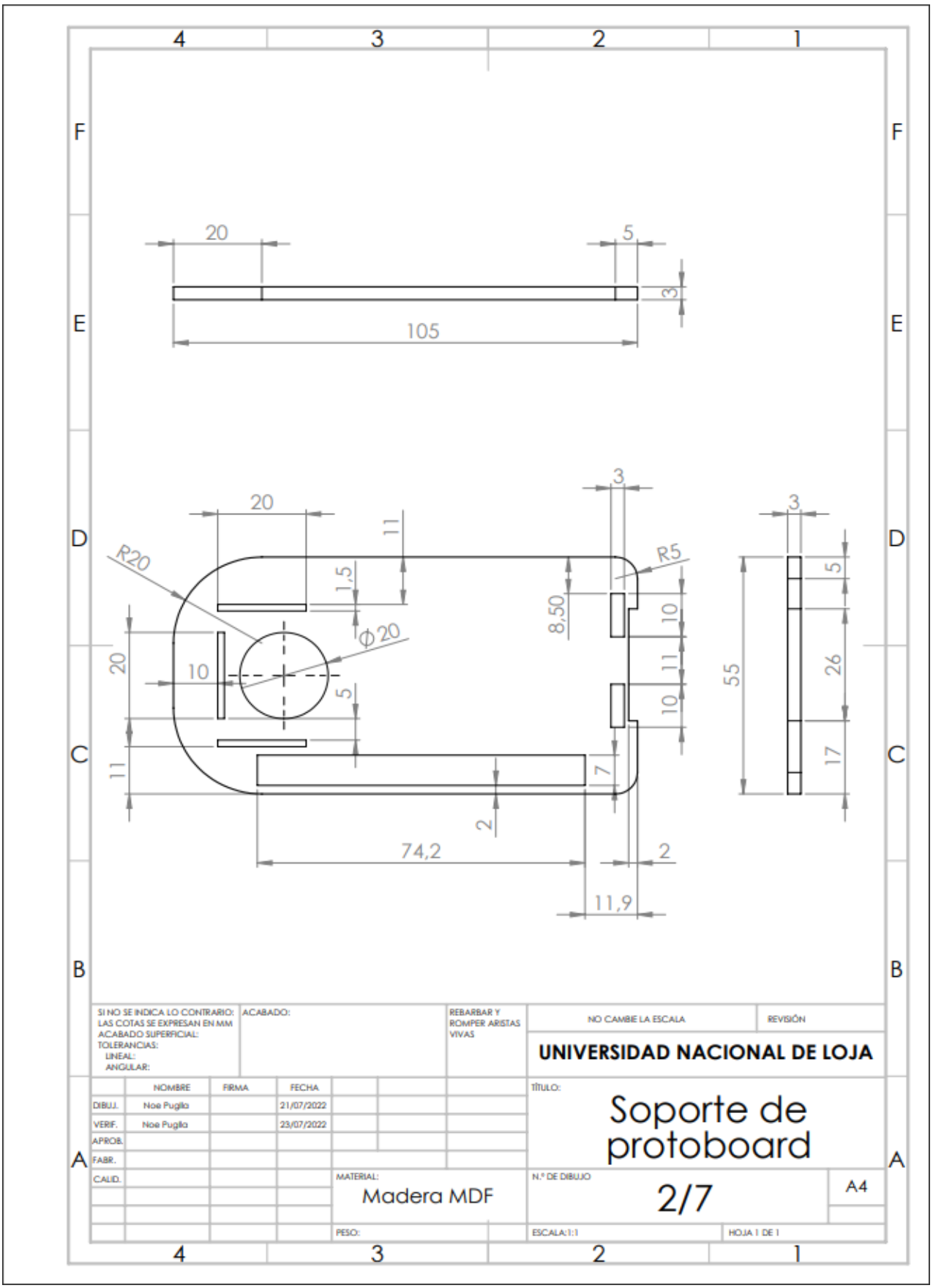
Vildósola, E. (2008). *ACTUADORES.* Santiago: Soltex Chile S.A.

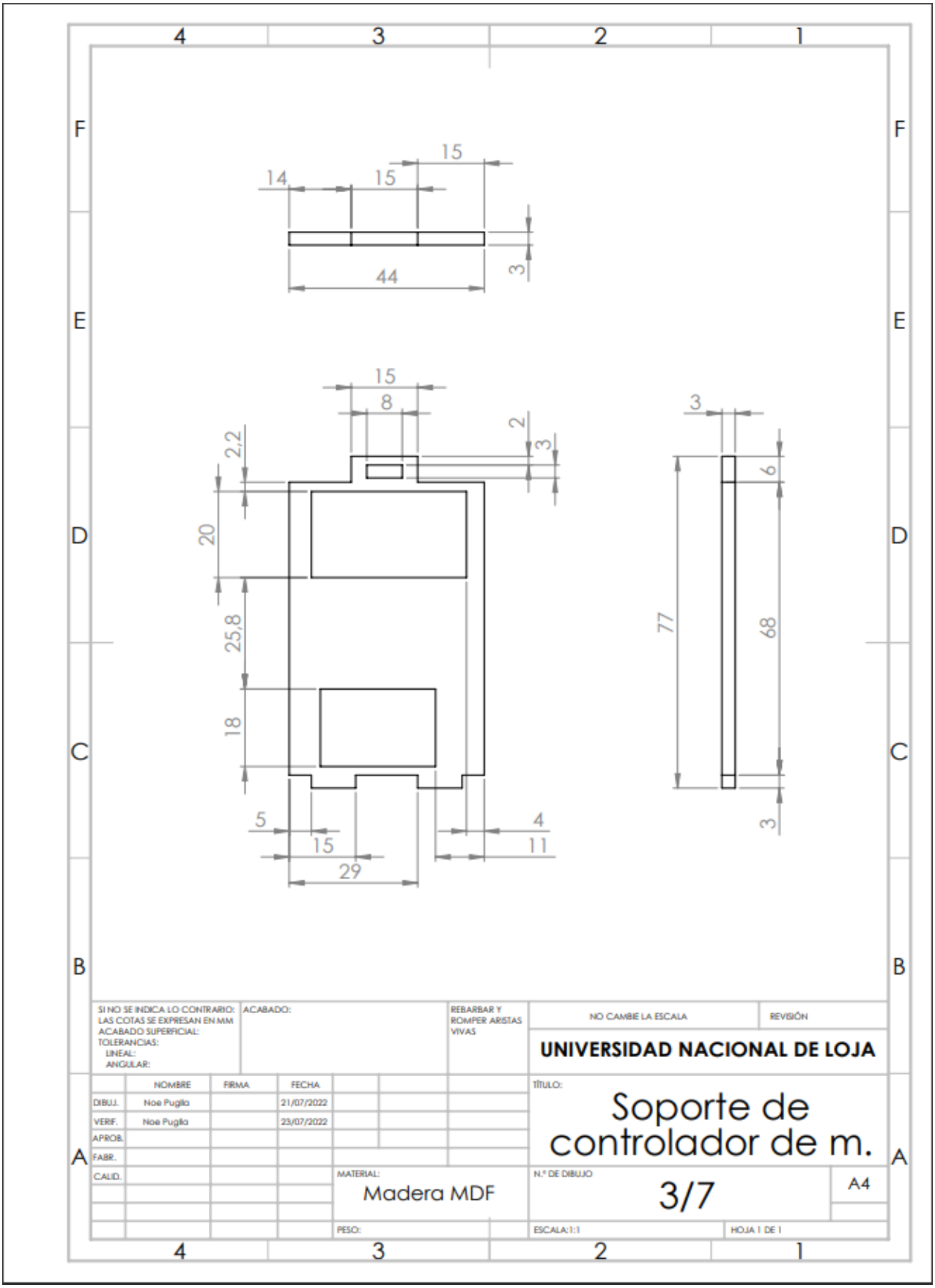
Yepes, L. (18 de Agosto de 2019). *Ingenierìa Bàsica.* Obtenido de Como salir de un laberinto. Mètodo de la mano derecha: https://ingenieriabasica.es/como-salir-de-un-laberinto/

11. Anexos

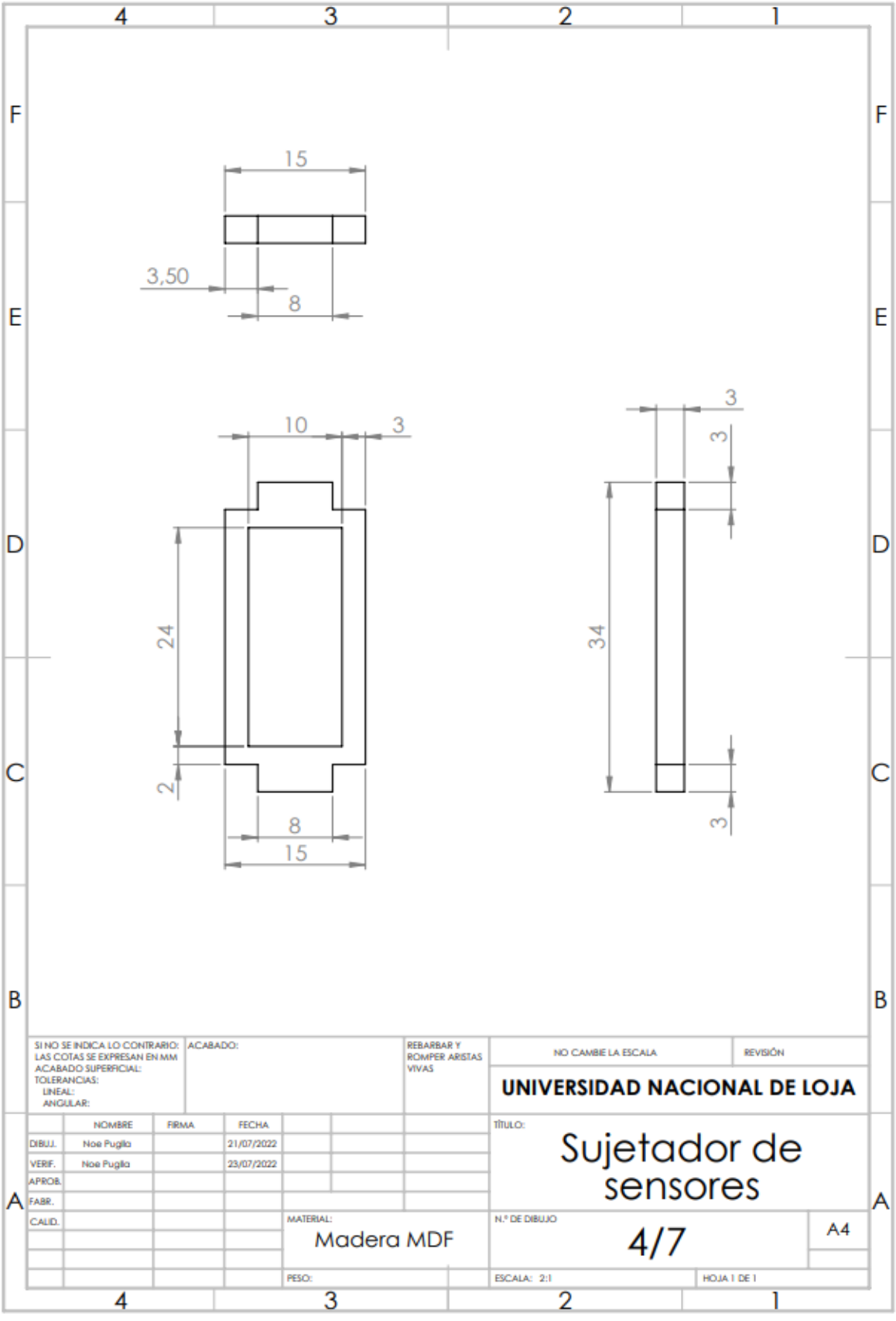
Anexo 1. Planos de las piezas del chasis



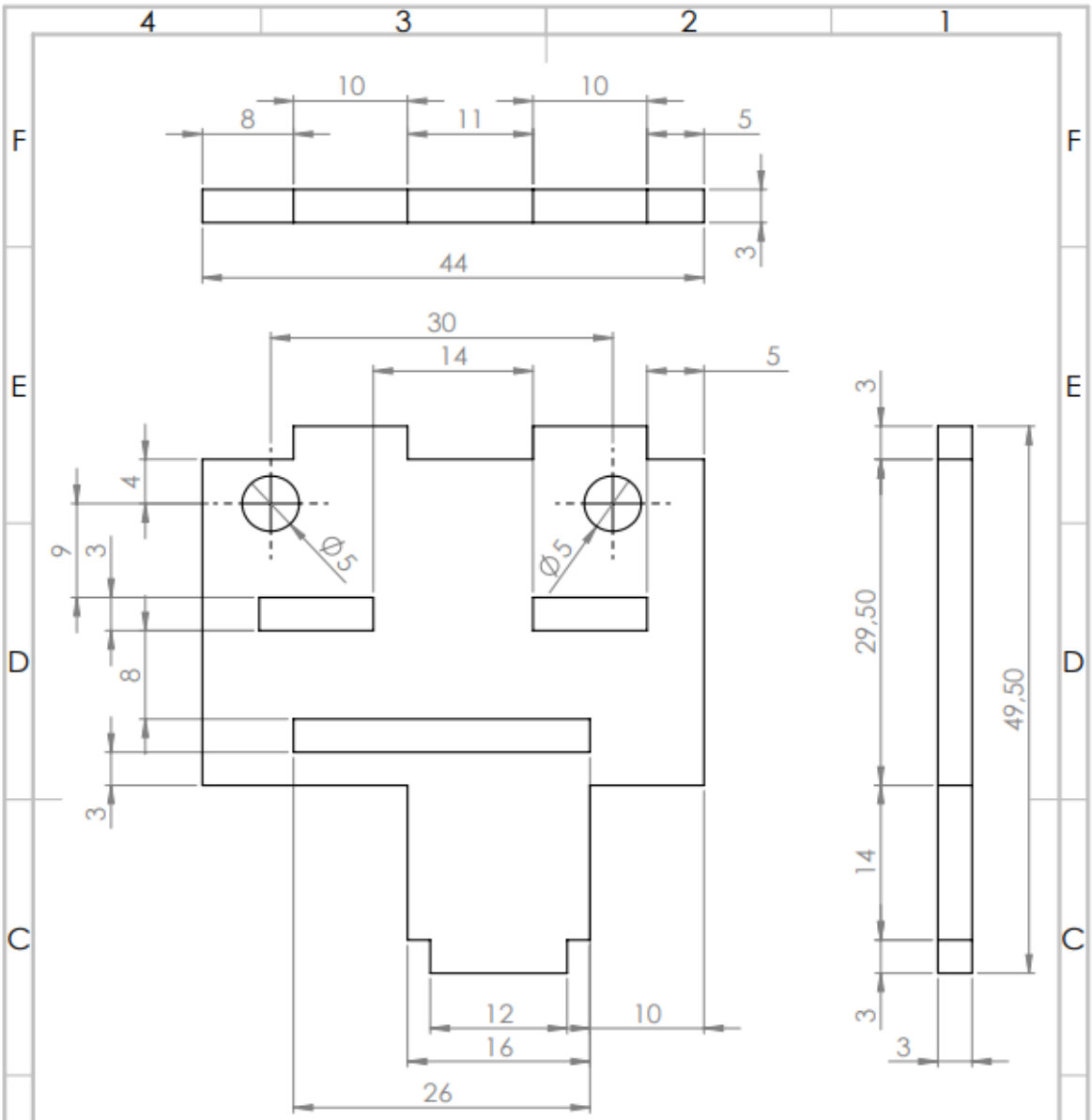

20
5
3
105
20
11
3
R20
1,5
R5
8,50
10
Φ20
20
10
5
11
10
11
7
2
74,2
2
11,9
3
5
55
26
17
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:
ACABADO:
REBARBAR Y ROMPER ARISTAS VIVAS
NO CAMBIE LA ESCALA
REVISIÓN
UNIVERSIDAD NACIONAL DE LOJA
NOMBRE
FIRMA
FECHA
DIBUJ. Noe Puglla 21/07/2022
VERIF. Noe Puglla 23/07/2022
APROB.
FABR.
CALID.
TÍTULO:
Soporte de protoboard
MATERIAL:
Madera MDF
N.º DE DIBUJO
2/7
A4
PESO:
ESCALA:1:1
HOJA 1 DE 1



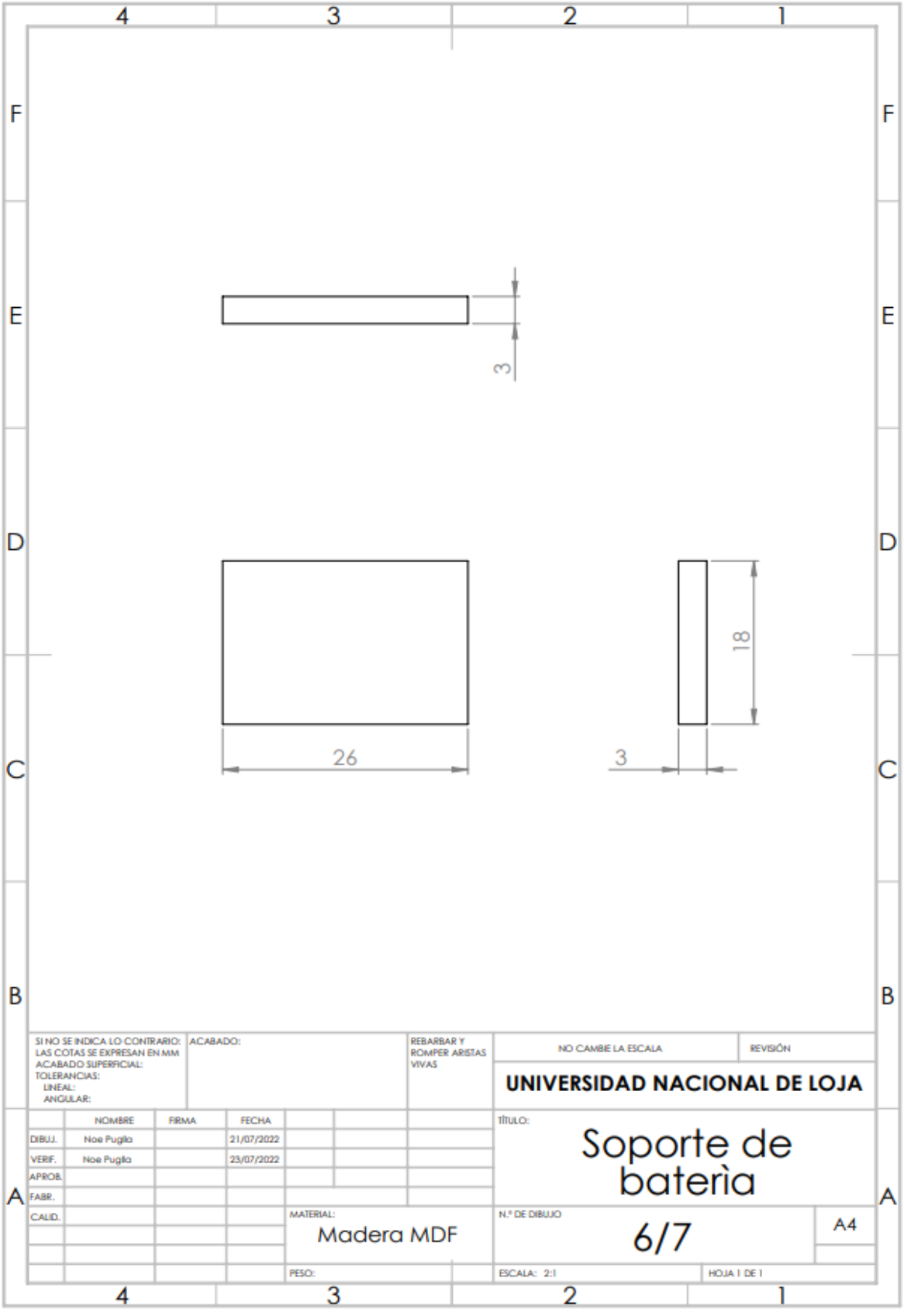
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
						UNIVERSIDAD NACIONAL DE LOJA	
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	Noe Puglla		21/07/2022			Soporte de controlador de m.	
VERIF.	Noe Puglla		23/07/2022				
APROB.							
FABR.							
CALID.				MATERIAL: Madera MDF		N.º DE DIBUJO 3/7	A4
				PESO:		ESCALA:1:1	HOJA 1 DE 1

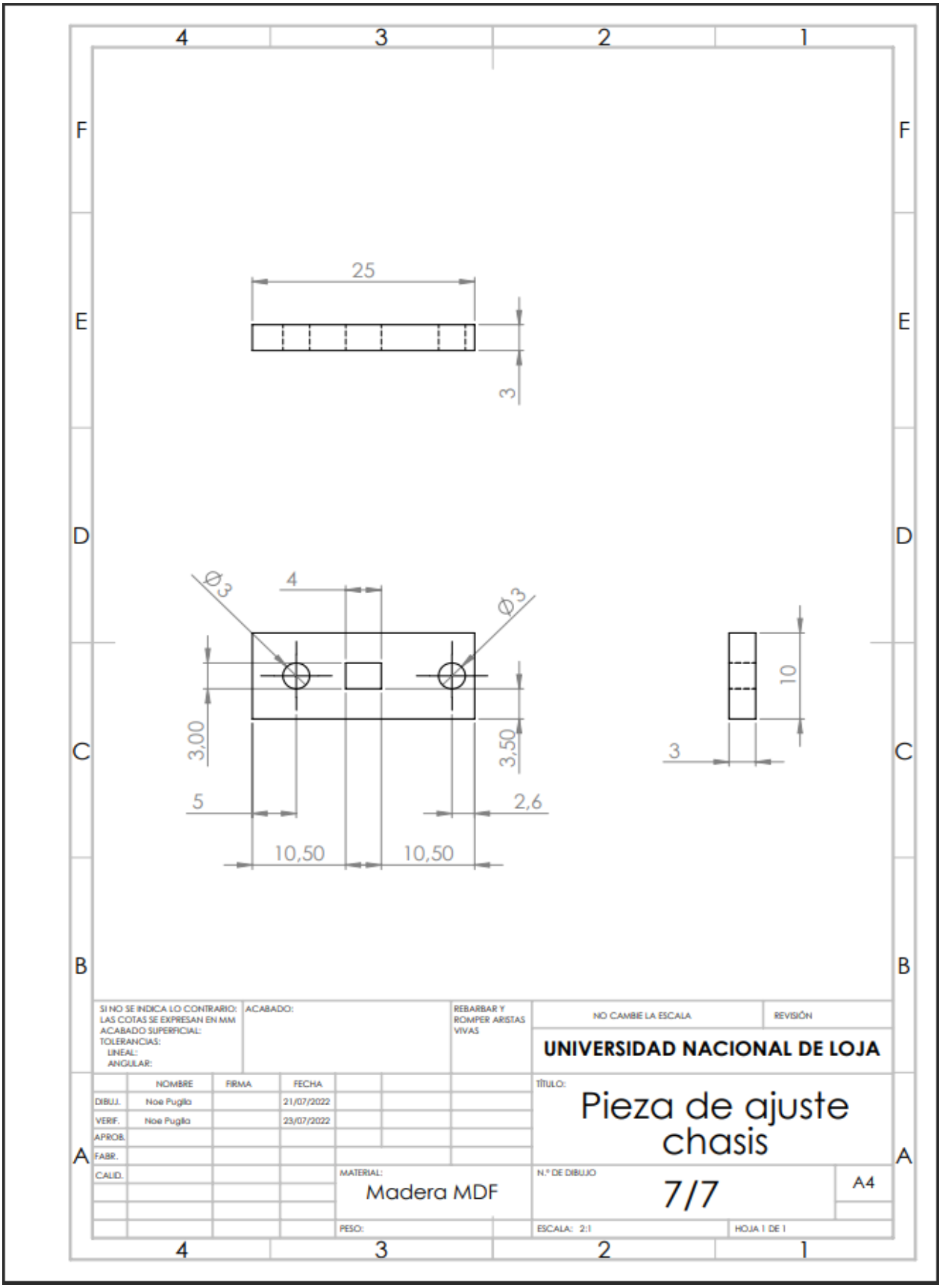


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN	
					UNIVERSIDAD NACIONAL DE LOJA		
	NOMBRE	FIRMA	FECHA		TÍTULO: Sujetador de sensores		
DIBUJ.	Noe Puglla		21/07/2022				
VERIF.	Noe Puglla		23/07/2022				
APROB.							
FABR.							
CALID.				MATERIAL: Madera MDF	N.º DE DIBUJO 4/7	A4	
				PESO:	ESCALA: 2:1	HOJA 1 DE 1	



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN	
				UNIVERSIDAD NACIONAL DE LOJA		
	NOMBRE	FIRMA	FECHA	TÍTULO: Soporte de placa arduino		
DIBUJ.	Noe Puglla		21/07/2022			
VERIF.	Noe Puglla		23/07/2022			
APROB.						
FABR.						
CALID.						
MATERIAL: Madera MDF				N.º DE DIBUJO 5/7	A4	
PESO:				ESCALA: 2:1	HOJA 1 DE 1	


4
3
2
1
F
E
D
C
B
A
3
26
3
18
SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:
ACABADO:
REBARBAR Y ROMPER ARISTAS VIVAS
NO CAMBIE LA ESCALA
REVISIÓN
UNIVERSIDAD NACIONAL DE LOJA
NOMBRE
FIRMA
FECHA
DIBUJ.
Noe Puglla
21/07/2022
VERIF.
Noe Puglla
23/07/2022
APROB.
FABR.
CALID.
TÍTULO:
Soporte de batería
MATERIAL:
Madera MDF
N.º DE DIBUJO
6/7
A4
PESO:
ESCALA: 2:1
HOJA 1 DE 1


4
3
2
1
F
E
D
C
B
A
25
3
Ø3
4
Ø3
3,00
3,50
5
2,6
10,50
10,50
10
3
SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:
ACABADO:
REBARBAR Y ROMPER ARISTAS VIVAS
NO CAMBIE LA ESCALA
REVISIÓN
UNIVERSIDAD NACIONAL DE LOJA
NOMBRE
FIRMA
FECHA
DIBUJ.
Noe Puglla
21/07/2022
VERIF.
Noe Puglla
23/07/2022
APROB.
FABR.
CALID.
TÍTULO:
Pieza de ajuste chasis
MATERIAL:
Madera MDF
N.º DE DIBUJO
7/7
A4
PESO:
ESCALA: 2:1
HOJA 1 DE 1

Anexo 2. Código del programa desarrollado

```
#include <Ultrasonic.h>
#define MDER_ADELANTE 5 //derecho
#define MDER_ATRAS 6
#define MIZQ_ADELANTE 10// izquierdo
#define MIZQ_ATRAS 11
//modificado 3
Ultrasonic front(8, 9); //(Tring PIN.Echo PIN)
Ultrasonic left(3, 2); //(Trig PIN.Echo PIN)
Ultrasonic right(4, 7); // (Trig PIN,Echo PIN)
int distancia_minima = 7;
int distancia_media = 10;
int distancia_max = 120;
int distancia_terminar = 100;
int led = 12;
int val=A0;
int vg = 50;
int v = 125;
int vf = 150;
int leftS, rightS, frontS;
void setup() {
  Serial.begin(9600);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  Serial.println("Iniciando");
  delay(3000);
  Serial.println("Listo empezamos");
  delay(3000);
  while (true) {
    ReadS();
    GoFront();
    if (frontS >= distancia_minima && frontS < distancia_max) {
      ReadS();
      Ajust();
```

```
        digitalWrite(led,LOW);
    }
    if (frontS <= 5) {
      ReadS();
      STOP();
      //delay(2000);
      ReadS();
      Turn();
    }
        if (frontS > distancia_max && rightS < distancia_media && leftS < distancia_media) {
      ReadS();
      STOP();
      digitalWrite(led,HIGH);
      delay(10000);
  }
  }
}
void ReadS () {
  Serial.print("\n Adelante: " + String(frontS) + "cm");
  Serial.print("\n Derecha: " + String(rightS) + "cm");
  Serial.println("\n Izquierda: " + String(leftS) + "cm");
  leftS = left.read();
  rightS = right.read();
  frontS = front.read();
}
void Turn() {
  if (leftS < rightS ) {
    GoRight();
  }
  if (leftS > rightS ) {
    GoLeft();
  }
  if (rightS <= 15 && leftS <= 15 && frontS <= 15) {
    GoBack();
  }
}

void Ajust() {
  Serial.println("front");
```

```
  GoFront();
  if (rightS <= 3) {
    analogWrite(MDER_ADELANTE, vf + 5);// derecho
    digitalWrite(MDER_ATRAS, LOW);
    analogWrite(MIZQ_ATRAS, 10);
    digitalWrite(MIZQ_ADELANTE, LOW);
  }
  if (leftS <= 2) {
    analogWrite(MDER_ADELANTE, LOW);// derecho
    digitalWrite(MDER_ATRAS, 5);
    analogWrite(MIZQ_ATRAS, LOW);
    digitalWrite(MIZQ_ADELANTE, 5);
  }
 }
void GoFront() {
  digitalWrite(MDER_ATRAS, LOW);
  analogWrite(MDER_ADELANTE, vf);
  analogWrite(MIZQ_ADELANTE, vf);
  digitalWrite(MIZQ_ATRAS, LOW);
}
void GoBack() {
  analogWrite(MDER_ATRAS, vf);
  digitalWrite(MDER_ADELANTE, LOW);
  digitalWrite(MIZQ_ATRAS, LOW);
  analogWrite(MIZQ_ADELANTE, vf);
}
void GoLeft() {
  STOP();
  analogWrite(MDER_ADELANTE, v);
  digitalWrite(MDER_ATRAS, LOW);
  analogWrite(MIZQ_ATRAS, vg);
  digitalWrite(MIZQ_ADELANTE, LOW);
  delay(500);
}
void GoRight() {
  STOP();
  digitalWrite(MDER_ADELANTE, LOW);
  analogWrite(MDER_ATRAS, vg);
  digitalWrite(MIZQ_ATRAS, LOW);
  analogWrite(MIZQ_ADELANTE, v);
  digitalWrite(led,LOW);
  delay(500);
}
void STOP() {
  digitalWrite(MDER_ATRAS, LOW);
  digitalWrite(MDER_ADELANTE, LOW);
  digitalWrite(MIZQ_ATRAS, LOW);
  digitalWrite(MIZQ_ADELANTE, LOW);
}
```

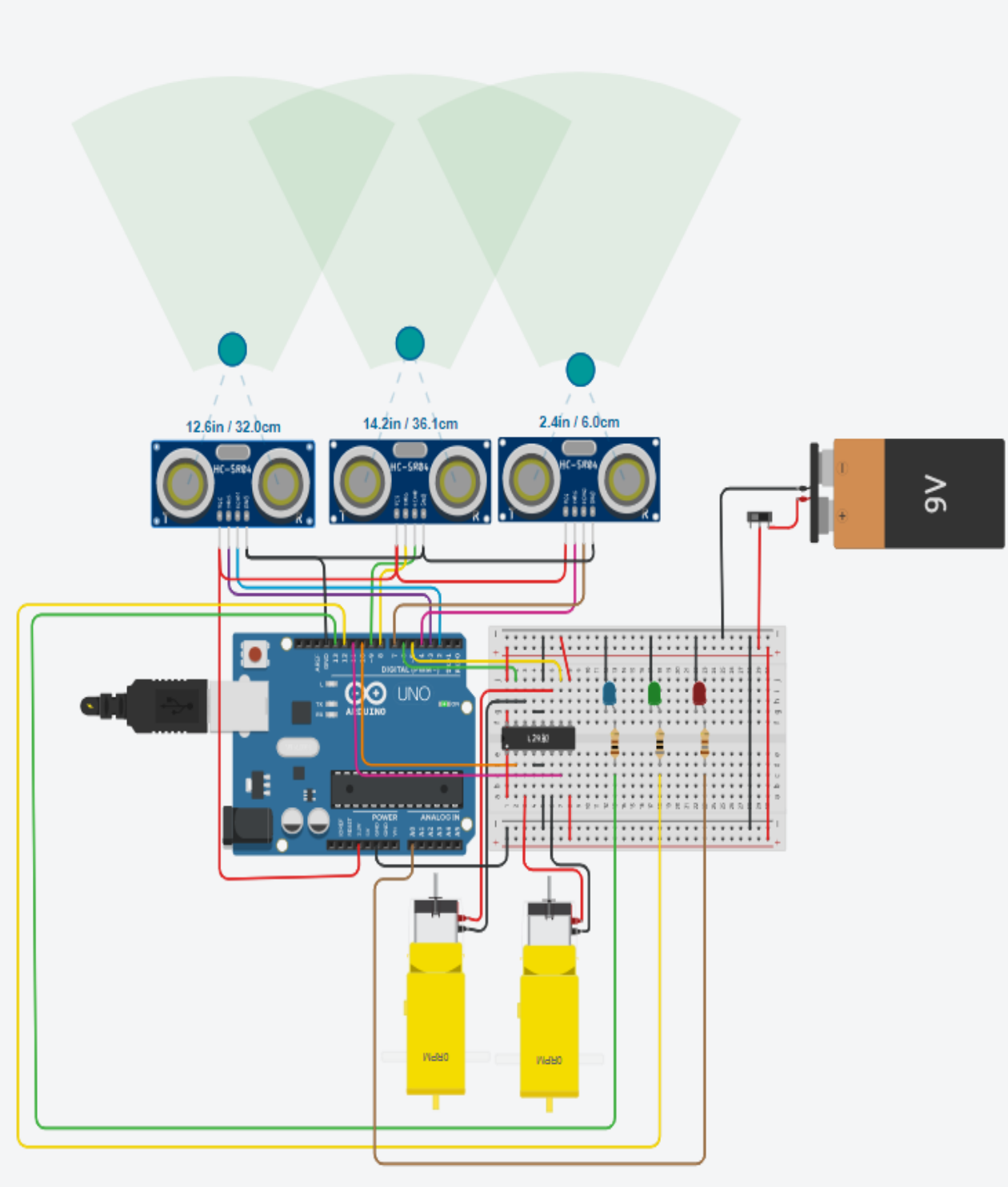
Anexo 3. Código del programa para verificar el funcionamiento de sensores de distancia hc-sr04

```
#define Pecho 6
#define Ptrig 7
long duracion, distancia;

void setup() {
  // put your setup code here, to run once:
Serial.begin (9600);
pinMode(Pecho,INPUT);
pinMode(Ptrig,OUTPUT);
pinMode(13,1);
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(Ptrig,LOW);
delayMicroseconds(2);
digitalWrite(Ptrig,HIGH);
delayMicroseconds(10);
digitalWrite(Ptrig,LOW);

duracion=pulseIn(Pecho,HIGH);
distancia=(duracion/2)/29;

if(distancia >= 500 || distancia <= 0){
Serial.println("---");
}
else{
  Serial.print(distancia);
  Serial.println("cm");
  digitalWrite(13,0);
}

if(distancia <= 10 && distancia >= 1){
  digitalWrite(13,1);
  Serial.println("Alarma.......");
}
delay(400);
}
```

Anexo 4. Código del programa para verificar funcionamiento de motores

```
int v = 100;
void setup() {
  Serial.begin(9600);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {
  Serial.println("Iniciando");
IRDERECHA();
}


void IRADELANTE() {
  digitalWrite(10, LOW);
  analogWrite(11, v);
  analogWrite(5, v);
  digitalWrite(6, LOW);
}
void IRATRAS() {
  analogWrite(10, v);
  digitalWrite(11, LOW);
  digitalWrite(5, LOW);
  analogWrite(6, v);
}
void IRIZQUIERDA() {
  analogWrite(10, v);
  digitalWrite(11, LOW);
  analogWrite(5, v);
  digitalWrite(6, LOW);
}
void IRDERECHA() {
  digitalWrite(10, LOW);
  analogWrite(11, v);
  digitalWrite(5, LOW);
  analogWrite(6, v);
}
```

Anexo 5. Simulación del robot en el programa de tinkercad



Anexo 6. Programa de control algoritmo de dijkstra

```
#define vel_motor_izq 6 // control de velocidad de motor izquierdo
#define vel_motor_der 9 // control de velocidad de motor derecho
int led=7;
int val=A0;
int trig_frente = A4; // control o impulso enviado de sensor frontal
int echo_frente = A5; // control o impluso recibido de sensor frontal

int trig_izq = A2; // control o impulso enviado de sensor izquierdo
int echo_izq = A3; // control o impluso recibido de sensor izquierdo

int trig_der = 8; // control o impulso enviado de sensor derecho
int echo_der = A1; // control o impluso recibido de sensor derecho

const int m = 35;// maximo de elementos del vector
int pasos[m]; // pasos que debe realizar el robot para terminar
              //  el laberinto
int contador;
int alter[m]; // alternativa del camino tomado
int decision[m]; // puntos dicisivos

int repcir = 10;//pulsador repetir circuito
byte est1 = HIGH; // estado del pulsador repcir
int tiempo_espera1 = 10;//tiempo de espera para eliminar rebote
unsigned long tiempo_pulsador1;

int t_espera = 1500;// tiempo de espera para ejercutar el laberinto
int ledverde = 13; // indicador de funcionamiento de boton

int borrar = 11;//pulsador borrar memoria
int esta = HIGH; // estado del pulsador borrar
int tiempo_espera2 = 10;//tiempo de espera para eliminar rebote
unsigned long tiempo_pulsador2;

int ledrojo = 12; //indicador de borrar almacenamiento
```

```
int distancia_minima = 4;
int distancia_media = 7;
int distancia_max = 80;
int distancia_terminar = 100;

void setup() {
 pinMode(led,OUTPUT);
  pinMode(trig_frente, OUTPUT);
  pinMode(echo_frente, INPUT);
  pinMode(trig_izq, OUTPUT);
  pinMode(echo_izq, INPUT);
  pinMode(trig_der, OUTPUT);
  pinMode(echo_der, INPUT);
  pinMode(repcir, INPUT_PULLUP);
  digitalWrite(repcir, HIGH);
  pinMode(borrar, INPUT_PULLUP);
   digitalWrite(borrar, HIGH);
  pinMode(ledverde, OUTPUT);
  pinMode(ledrojo, OUTPUT);
  Serial.begin(9600);
  delay(5000);

  analogWrite(vel_motor_izq, 0); // inicializacion de motores con pulso 0;
  analogWrite(vel_motor_der, 0);
}

long readadelante(int trig_frente, int echo_frente) {
  long duracion_frente, frente;


  digitalWrite(trig_frente, LOW); // SENSOR FRONTAL
  delayMicroseconds(4);
  digitalWrite(trig_frente, HIGH);
  delayMicroseconds(10);
```

```
  delayMicroseconds(10);
  digitalWrite(trig_frente, LOW);
  duracion_frente = pulseIn(echo_frente, HIGH);
  frente = duracion_frente *10/ 292/2;
  return frente;
}

long readderecha(int trig_der, int echo_der) {
  long duracion_der, derecha;


  digitalWrite(trig_der, LOW); // SENSOR ULTRASONICO DERECHO
  delayMicroseconds(4);
  digitalWrite(trig_der, HIGH); //
  delayMicroseconds(10);
  digitalWrite(trig_der, LOW);
  duracion_der = pulseIn(echo_der, HIGH);
  derecha = duracion_der *10/ 292/2;
  return derecha;
}

long readizquierda(int trig_izq, int echo_izq) {
  long duracion_izq, izquierda;


 digitalWrite(trig_izq, LOW); // SENSOR ULTRASONICO IZQUIERDO
  delayMicroseconds(4);
  digitalWrite(trig_izq, HIGH); //
  delayMicroseconds(10);
  digitalWrite(trig_izq, LOW);
  duracion_izq = pulseIn(echo_izq, HIGH);
  izquierda = duracion_izq *10/ 292/2;
  return izquierda;
}
```

```
void avance(int frente, int derecha, int izquierda) {
  analogWrite(vel_motor_izq, 0); // inicializacion de motores con pulso 0;
  analogWrite(vel_motor_der, 0);
  if (frente > distancia_media)
    //correcion choque de pared
    if (derecha <= distancia_minima && frente > distancia_media) {
      analogWrite(vel_motor_izq, 130);//OJO
      analogWrite(vel_motor_der, 150);
    }
    if (izquierda >= distancia_max && derecha >= distancia_max) {
      for (int i = m - 1; i >= 0; i--) {
        if (decision[i] == 2) {
          if (pasos[i] == 2) {
            analogWrite(vel_motor_izq, 255);
            analogWrite(vel_motor_der, 30);
            pasos[contador] = 2;
            alter[contador] = 1;
          } else {
            analogWrite(vel_motor_izq, 255);
            analogWrite(vel_motor_der, 30);
            pasos[contador] = 1;
            alter[contador] = 2;
          }
        }
      }
      if (frente >= distancia_max) {
        decision[contador] = 2;
      } else {
        decision[contador] = 0;
      }
      contador++;
    }
    if (derecha >= distancia_max ) {
```

```
    analogWrite(vel_motor_izq, 255);
    analogWrite(vel_motor_der, 30);
    pasos[contador] = 2;
    alter[contador] = 3;
    if (frente >= distancia_max) {
      decision[contador] = 1;
    } else {
      decision[contador] = 0;
    }
    contador++;
  }
  if (izquierda <= distancia_minima && frente > distancia_media) {
    analogWrite(vel_motor_izq, 150);
    analogWrite(vel_motor_der, 130);
  }
  if (izquierda >= distancia_max) {
    analogWrite(vel_motor_izq, 60);
    analogWrite(vel_motor_der, 255);
    pasos[contador] = 1;
    alter[contador] = 3;
    if (frente >= distancia_max) {
      decision[contador] = 1;
    } else {
      decision[contador] = 0;
    }
    contador++;
  }
}
if (derecha < distancia_media && izquierda < distancia_media && frente < distancia_media) {
  do {
    analogWrite(vel_motor_izq, 255);
    analogWrite(vel_motor_der, 10);
    digitalWrite(led,HIGH);
  } while ( readadelante(triq frente, echo frente) < distancia media);
```

```
  for (int i = m - 1; i >= 0; i--) {
    if (decision[i] == 1) {
      pasos[i] = alter[i];
    }
    if (decision[i] == 2) {
      pasos[i] = alter[i];
    }
  }
}
```

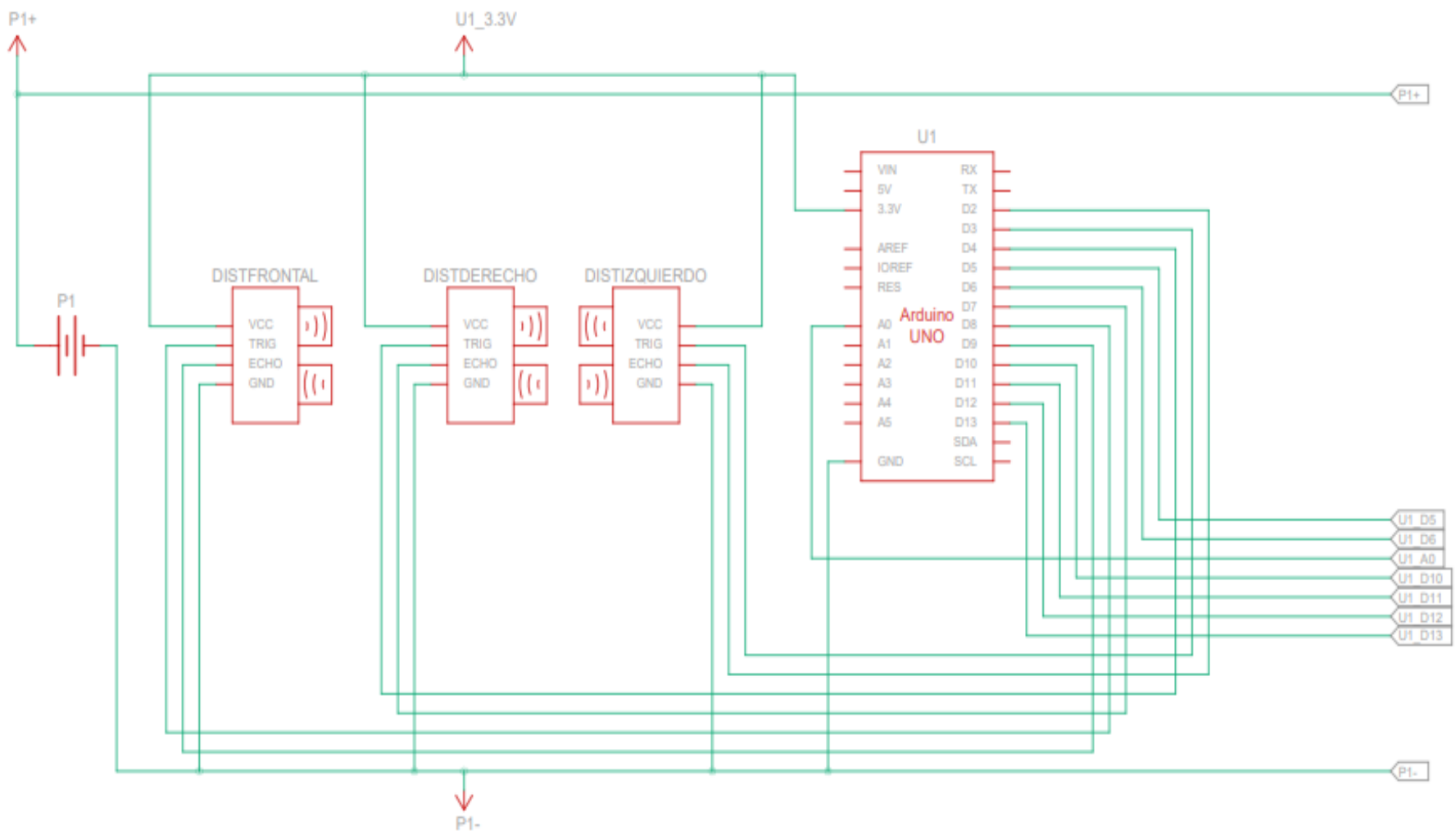
```
if (derecha >= distancia_max && izquierda >= distancia_max && frente >= distancia_max) {
  analogWrite(vel_motor_izq, 0);// aquí encuentra la salida y llega a la meta
  analogWrite(vel_motor_der, 0);
  Serial.println("\n Ha finalizado el laberinto");
  Serial.println("\n Desea repetir el circuito con la ruta mas corta?");
  int estado1 = digitalRead(repcir);
  if (estado1 != est1) {
    if (millis() - tiempo_pulsador1 >= tiempo_espera1) {
      tiempo_pulsador1 = millis();
      est1 = estado1;
    }
    if (estado1 == LOW) {
      Serial.println("\n Se procedera a repetir el circuito con la ruta mas corta");
      delay(t_espera);
      digitalWrite(ledverde, HIGH); // COMIENZA A RECORRER LABERINTO POR MEMORIA
      //void avance_controlado();
      int a = 0;
      do {
        if (frente > distancia_media) {
          //correcion choque de pared
          if (derecha <= distancia_minima && izquierda > distancia_minima && izquierda < distancia_media) {
            analogWrite(vel_motor_izq, 130);
            analogWrite(vel_motor_der, 150);
```

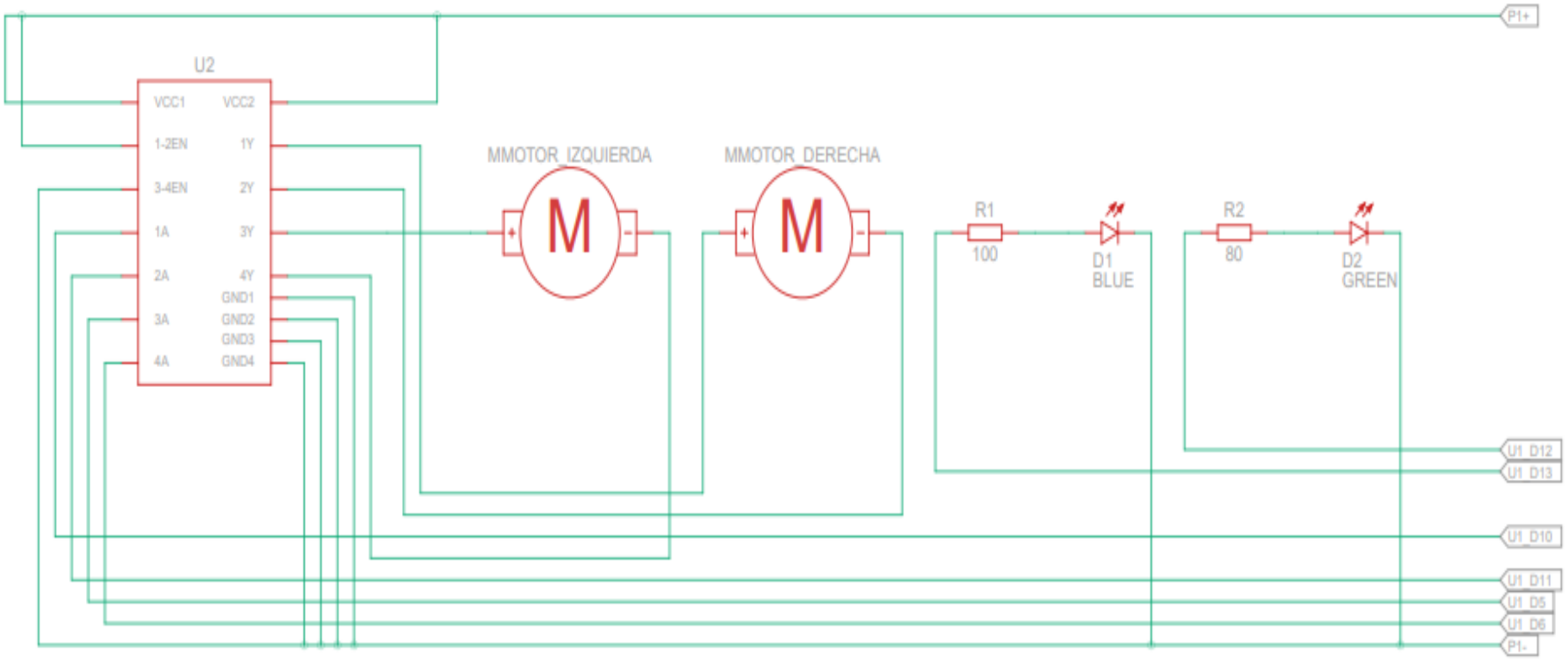
```
}
if (izquierda >= distancia_max && derecha >= distancia_max) {
  if (pasos[a] == 2) {
    analogWrite(vel_motor_izq, 255);
    analogWrite(vel_motor_der, 60);
  }
  if (pasos[a] == 1) {
    analogWrite(vel_motor_izq, 60);
    analogWrite(vel_motor_der, 255);
  }
}

if (derecha >= distancia_max) {
  if (pasos[a] == 2) {
    analogWrite(vel_motor_izq, 255);
    analogWrite(vel_motor_der, 60);
  }
  if (pasos[a] == 3) {
    analogWrite(vel_motor_izq, 150);
    analogWrite(vel_motor_der, 150);
  }
}
if (izquierda <= distancia_minima && derecha > distancia_minima && derecha < distancia_media) {
  analogWrite(vel_motor_izq, 150);
  analogWrite(vel_motor_der, 120);
}
if (izquierda >= distancia_max) {
  if (pasos[a] == 1) {
    analogWrite(vel_motor_izq, 60);
    analogWrite(vel_motor_der, 255);
  }
  if (pasos[a] == 3) {
    analogWrite(vel_motor_izq, 150);
    analogWrite(vel motor der, 150);
```

```
            }
          }
        }
        a++;
      } while (a <= contador);
      digitalWrite(ledverde, LOW); // FINALIZA LABERINTO
    }
    Serial.println("\n Desea borrar memoria?");
    int estado2 = digitalRead(borrar);
    if (estado2 != esta) {
      if (millis() - tiempo_pulsador2 >= tiempo_espera2) {
        tiempo_pulsador2 = millis();
        esta = estado2;
      }
      if (estado2 == LOW) {
        Serial.println("\n Se procedera a borrar memoria");
        delay(t_espera);
        digitalWrite(ledrojo, HIGH); // COMIENZA A BORRAR MEMORIA
        for (int b = 0; b <= m; b++) {
          pasos[b] = 0;
          alter[b] = 0;
          decision[b] = 0;
        }
        contador = 0;

        digitalWrite(ledrojo, LOW); // FINALIZA BORRAR MEMORIA
      }
    }
  }
}

void loop() {
  val=analogRead(A0);
```

```
  if (val > 400){
    digitalWrite(led,HIGH);
  }
  else{
    digitalWrite(led,LOW);
    }

  long frente = readadelante(trig_frente, echo_frente);
  Serial.print("\n Adelante: " + String(frente) + "u");
  long izquierda = readizquierda(trig_izq, echo_izq);
  Serial.print("\n Izquierda: " + String(izquierda) + "u");
  long derecha = readderecha(trig_der, echo_der);
  Serial.print("\n Derecha: " + String(derecha) + "u");
  if (frente < distancia_terminar) {
    avance(frente, derecha, izquierda);
  }
}
```

Anexo 7. Diagrama de conexión del robot




U2
VCC1
VCC2
1-2EN
1Y
3-4EN
2Y
1A
3Y
2A
4Y
GND1
3A
GND2
GND3
4A
GND4
MMOTOR_IZQUIERDA
MMOTOR_DERECHA
M
M
R1
100
D1
BLUE
R2
80
D2
GREEN
P1+
U1_D12
U1_D13
U1_D10
U1_D11
U1_D5
U1_D6
P1-

Anexo 8. Certificación de traducción del resumen

FINE-TUNED ENGLISH LANGUAGE INSTITUTE
Líderes en la Enseñanza del Inglés

Ing. María Belén Novillo Sánchez.
ENGLISH TEACHER- FINE TUNED ENGLISH CIA LTDA.

CERTIFICA:

Que el documento aquí compuesto es fiel traducción del idioma español al idioma inglés del resumen de tesis **"Diseño y construcción de un micro robot para resolución de laberintos"** autoría del señor **Noe Ronaldo Puglla Morocho** con número de cédula **1150339248**, egresado de la carrera de Ingeniería electromecánica de la Universidad Nacional de Loja.

Lo certifico en honor a la verdad y autorizo al interesado hacer uso del presente en lo que a sus intereses convenga.

Loja, 3 de octubre del 2022

Ing. María Belén Novillo Sánchez.
ENGLISH TEACHER- FINE TUNED ENGLISH CIA LTDA.