PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UNA PLATAFORMA DE PROGRAMACIÓN TANGIBLE COMO HERRAMIENTA EDUCATIVA QUE PUEDA PERMITIR EL DESARROLLO DEL PENSAMIENTO COMPUTACIONAL EN NIÑOS DE LA EDUCACIÓN BÁSICA REGULAR

Tesis para optar el Título de Ingeniera Electrónica

AUTORA:

Maria Alejandra Manrique Zarate

ASESOR:

PhD(c) MSc. Pablo Cárdenas Cáceres

Lima, marzo del 2021

Resumen

El pensamiento computacional, que se puede entender como el conjunto de habilidades y capacidades para resolver problemas a través de las ciencias computacionales y la programación, permite desarrollar tres competencias del Currículo Nacional de Educación Básica que plantea el Ministerio de Educación (MINEDU): la Competencia 22: *"Diseña y construye soluciones tecnológicas para resolver problemas de su entorno"*, la Competencia 23: *"Resuelve problemas de cantidad"* y la Competencia 28: *"Se desenvuelve en los entornos virtuales generados por las Tecnologías de la información y comunicación (TIC)"*. Estas tres competencias impactan directamente en dos de los puntos principales del perfil de egreso de un estudiante de la Educación Básica Regular (EBR) que promueve el MINEDU. El primero de estos puntos es el aprovechamiento responsable de las TIC para interactuar con la información y gestionar su comunicación y aprendizaje, el cual se puede medir con el índice NRI *(Networked Readiness Index)* que revela la correlación casi perfecta entre el nivel de absorción de las TIC de un país y los impactos económicos y sociales que las Tecnologías de Información y Comunicación tienen en su economía y sociedad. El segundo de los puntos del perfil de egreso hace referencia a la interpretación de la realidad y toma decisiones a partir de conocimientos matemáticos que aporten a su contexto y se puede medir con el Programa para la Evaluación Internacional de Alumnos de la OCDE (PISA) en el rubro de matemáticas. Con respecto a ambos indicadores, el Perú se encuentra por debajo de la media mundial con puntajes que lo ubican en el nivel 1 de la evaluación PISA y en el puesto 90 del índice NRI. La programación puede mejorar estos resultados; sin embargo, la única iniciativa que existió en el Perú para promoverla, "Una laptop por niño", fracasó debido a la complejidad y altos costos de las herramientas que utilizaba. La programación tangible surge como alternativa para la enseñanza de la programación, ya que reduce la brecha de edad, facilita la enseñanza y el aprendizaje y no requiere de una computadora por usuario, por lo que abarata costos.

La presente tesis hace una revisión de las plataformas de programación tangible existentes y plantea el diseño de una nueva plataforma que tome en cuenta el Currículo Nacional de Educación Básica. Esta plataforma está compuesta por piezas de programación tangibles sin diseño electrónico en su interior, a través de las cuales se codifica una trayectoria, y un robot móvil que virtualiza el código creado a través de las piezas tangibles y realiza la trayectoria codificada. Finalmente, se obtienen resultados positivos en las simulaciones realizadas para la plataforma de programación tangible diseñada, resultando en un dispositivo de bajo costo capaz de seguir trayectorias programadas a través de piezas tangibles.

A mi madre, Saby, por ser mi ejemplo a seguir y amor incondicional.
A mi padre, Gustavo, por enseñarme la fortaleza.
A mi familia por apoyarme e inspirarme cada día.
A mis amigas de toda la vida, Mari, Andrea, Jimena y Alysson,
gracias por los buenos momentos.
A William por el infinito apoyo y amor.
A mis amigos de la universidad y a mi promoción de electrónica
por hacer de la etapa universitaria la mejor de mi vida.
A mi asesor por las enseñanzas y la motivación.
A mi mascota, Mojito, por acompañarme cada noche de estudio.
A todas las mujeres que hacen historia en las ciencias e ingeniería.

“La imaginación es la facultad del descubrimiento, preeminentemente. Es lo que penetra en los mundos nunca vistos a nuestro alrededor, los mundos de la ciencia.”

Ada Lovelace

Índice General

Índice de figuras

Índice de tablas

Introducción

El mundo en el que vivimos se va desarrollando tecnológicamente cada vez más en distintas áreas y la educación no es la excepción. En los últimos años, esta se ha complementado con las Tecnologías de Información y Comunicación (TIC) como herramientas metodológicas y educativas. Una de estas herramientas es la programación, la cual ayuda con la adquisición de habilidades de pensamiento computacional, sobre todo en escolares de Educación Básica Regular (EBR). El pensamiento computacional se puede definir como la habilidad y capacidad para resolver problemas utilizando la programación y los fundamentos de las ciencias computacionales [1].

Específicamente, en Perú, la enseñanza de la programación puede llegar a jugar un papel muy importante a la hora de ayudar con el cumplimiento del Currículo Nacional de Educación Básica, ya que permite el desarrollo de algunas de sus competencias, entre ellas la Competencia 22: *"Diseña y construye soluciones tecnológicas para resolver problemas de su entorno"*, la Competencia 23: *"Resuelve problemas de cantidad"* y la Competencia 28: *"Se desenvuelve en los entornos virtuales generados por las TIC"*. Estas competencias pueden ser contenidas en dos de los puntos principales del perfil de egreso de un estudiante de la EBR que promueve el Ministerio de Educación (MINEDU).

El primero de estos puntos es el aprovechamiento responsable de las TIC para interactuar con la información y gestionar su comunicación y aprendizaje [2]. Con respecto a este punto, según los últimos resultados del "Informe Global sobre Tecnologías de la Información: TICs para Crecimiento Inclusivo" presentados por el Foro Económico Mundial (FEM), el Perú se encuentra en el puesto número 90 a nivel mundial y en el puesto 14 en Latinoamérica. Cabe resaltar que dicho informe se basa en el índice NRI *(Networked Readiness Index)* que revela la correlación casi perfecta entre el nivel de absorción de las TIC de un país y los impactos económicos y sociales que las Tecnologías de Información y Comunicación tienen en su economía y sociedad [3].

El segundo de los puntos del perfil de egreso hace referencia a la interpretación de la realidad y toma decisiones a partir de conocimientos matemáticos que aporten a su contexto [2]. Sobre este último, se puede mencionar la posición actual del Perú según el Programa para la Evaluación Internacional de Alumnos de la OCDE (PISA), cuyo objetivo es determinar si los alumnos más próximos a culminar la educación obligatoria han adquirido algunos de los conocimientos y habilidades necesarios para desenvolverse en el mundo moderno. En el rubro específico de Matemáticas de esta evaluación, Perú

se encuentra muy por debajo de la media, específicamente en el nivel 1, que implica que los estudiantes son capaces de realizar tareas matemáticas directas y explícitas; sin embargo, no saben inferir o tomar decisiones secuenciales, tampoco aplicar algoritmos y mucho menos, crearlos [4].

Capítulo 1

Marco problemático

El Currículo Nacional de Educación Básica no contempla el pensamiento computacional pese a que promueve un perfil de egresado que, según lo mencionado anteriormente, no está siendo asimilado efectivamente por los estudiantes. La raíz de este problema se encuentra en los primeros años de la EBR (inicial y primaria), ya que al no existir una herramienta que les permita a los profesores introducir capacidades y conocimientos que desarrollen el pensamiento computacional desde temprana edad, este no es adquirido con facilidad resultando esta, la problemática de la tesis. Además, se sabe que, durante los primeros ocho años de vida, los niños aprenden más rápidamente que en cualquier otra época [5].

Se han desarrollado iniciativas en todo el mundo para poder introducir el pensamiento computacional a escolares de la EBR mediante la enseñanza de la programación. Una de estas, fue la introducción del *Scratch* (plataforma de programación orientada a niños) a escolares de entre 8 y 15 años de edad en el sector más vulnerable de Guayaquil. Con esta iniciativa, se comprobó que la inserción de la programación estimula el desarrollo de habilidades cognitivas, creativas y el pensamiento lógico matemático, pues el 60.55% obtuvo un mejor desenvolvimiento académico [6]. Además, en Italia, ingenieros de la *Università Politecnica delle Marche* aplicaron el proyecto piloto "Robotics In School" en el *Institute Comprensivo Largo Cocconi,* colegio ubicado en Roma. Este proyecto tuvo como objetivo la enseñanza de la programación mediante el kit robótico programable *Lego Mindstorms* a niños de la educación primaria. “Robotics In School” permitió concluir que los escolares envueltos pudieron desarrollar nuevas habilidades y potenciar las que ya tenían, entre ellas la resolución de problemas, el trabajo en equipo y la correcta utilización de la tecnología que tenían a su alcance [7].

En Perú, una iniciativa fue realizada en el año 2007 por el MINEDU con la implementación del programa "Una laptop por niño" junto a la organización sin fines de lucro que lleva el mismo nombre en inglés "*One Laptop per child*" (OLPC). Durante el programa, se entregaron cerca de más de un millón de laptops XO (computadoras diseñadas para niños) en diferentes escuelas a nivel nacional [8]. Adicionalmente, en el 2011, se inició también la repartición de 92 000 kits de robótica "*WeDo*" de la empresa Lego, con el fin de sentar una base en la enseñanza de la programación en niños de escuelas públicas. Los resultados de esta iniciativa se muestran en distintos trabajos de investigación [9], [10] que concluyen que el fracaso del programa se dio por causa de la poca preparación técnica que tenían los profesores y por la complejidad que ellos consideraban para enseñar programación computacional mediante los kits de robótica sin ser expertos en el tema. Otra de las desventajas fue que estos kits no se fabrican en Perú, por lo que los costos y tiempos de mantenimiento se veían incrementados.

Una plataforma de programación computacional tiene tres partes diferenciadas: el lenguaje de programación, que permite crear un código; el entorno de programación, que es donde se coloca el código; y el intérprete, que es el que finalmente ejecuta todas las instrucciones. Uno de los principales problemas de las plataformas virtuales, como *Scratch, Lego Mindsotrms* o *WeDo,* es que requieren de una computadora de escritorio. Este requerimiento limita la edad de uso, por lo que existen propuestas que optan por un código tangible en lugar de uno virtual. La programación tangible (TPL por sus siglas en inglés) fue denominada así por Suzuki y Kato en el año 1993, que en su investigación permitieron la comunicación con un intérprete virtual a través de la manipulación de objetos físicos [11]. La interacción con objetos cotidianos para el usuario hace la programación más intuitiva para principiantes [12]. Las ventajas de usar plataformas de programación totalmente tangibles como herramienta educativa son las siguientes:

- Tener la posibilidad de manipular el código con las manos y no depender de ninguna computadora adicional. Esto permite disminuir la brecha de edad para la introducción al mundo de la programación, ya que el niño se involucra más rápidamente cuando los elementos de aprendizaje forman parte del mundo real [11].

- Implementar dinámicas grupales y cooperativas, pues se pueden idear múltiples soluciones por problema o incluso soluciones mejor elaboradas [11]. Por otro lado, se logra una optimización de recursos, ya que la programación tradicional requiere de una computadora completa por cada usuario en el mejor de los casos, resultando en un aprendizaje más costoso.

• Detectar errores en el código. La depuración requiere de experticia y un perfecto conocimiento del código y el lenguaje en el que se está programando. Una plataforma de programación tangible permite visualizar con facilidad todo el código de manera física, brindándole a los escolares una oportunidad de mejorar su habilidad de autocorrección [12].

En base a lo antes mencionado, se plantearán el objetivo principal y los objetivos secundarios con el fin de proponer una solución a la problemática expuesta. Con este motivo, el objetivo general de la tesis será diseñar una herramienta electrónica de bajo costo que les permita a profesores la enseñanza de la programación en simultaneo a un salón de clases de niños de la EBR para facilitar la adquisición del pensamiento computacional usando la programación tangible. Además, los objetivos secundarios serán los siguientes:

- Diseñar un lenguaje de programación tangible de fácil entendimiento y manejo, que les permita a escolares de la EBR sin conocimientos previos en esta ciencia, programar de manera más intuitiva.
- Diseñar un entorno de programación que permita virtualizar el código, mediante una fotografía de este mismo, y mostrarlo a través de una aplicación móvil haciendo uso de un servidor web.
- Diseñar un robot móvil (intérprete) de bajo costo que ejecute instrucciones creadas mediante el código tangible.
- Diseñar y simular un sistema de control (usando el control de trayectorias) que le permita al intérprete la realización de instrucciones de movimiento con precisión.

Finalmente, cabe resaltar el alcance de la tesis, el cual abarcará el diseño de la plataforma de programación tangible, tomando en cuenta los objetivos previamente presentados, mas no se centrará en la validación pedagógica.

Capítulo 2

Marco de referencia

El capítulo 2 desarrollará tres marcos: el marco histórico, el marco conceptual y el marco metodológico. El marco histórico contempla investigaciones sobre plataformas de programación tangible y sus principales características, así como un cuadro comparativo de estas plataformas. El marco teórico describe las consideraciones necesarias para diseñar una plataforma de programación tangible. Finalmente, el marco metodológico expone la estrategia escogida para la realización de la tesis.

2.1 Marco histórico: Estado del arte

A continuación, se mostrará una breve revisión de cada una de las principales plataformas de programación tangible desarrolladas alrededor del mundo.

2.1.1 Algoblock

Es una plataforma de programación creada por H. Suzuki y H. Kato para mejorar las habilidades de diseño de soluciones mediante actividades grupales en escolares de primaria y secundaria [11]. Esta plataforma comprende un conjunto de bloques físicos que se pueden conectar entre sí manualmente para formar un programa. Cada uno de estos bloques representa un comando, que en conjunto forman un lenguaje con un entorno de programación netamente tangible. Algunas de estas instrucciones o comandos son “adelante”, “atrás”, “deslizar a la izquierda/derecha” y “girar” y, además, también existen comandos de control como “si-entonces-de lo contrario” y

"repetir hasta". Para que el lenguaje de programación pueda ser ejecutado, se necesita de una computadora, pues el intérprete es virtual: un submarino que se puede visualizar en un monitor y que acatará la lógica del código tangible previamente creado.

2.1.2 TactusLogic

Es una plataforma de programación, creada por Andrew Cyrus Smith, Heinrich Springhorn, Steven Bruce Mulligan, Ireyan Weber y Jackie Norris, con un entorno y lenguaje de programación tangibles. De manera similar a Algoblock, está compuesta por bloques de madera denominados codeBlocks, con la diferencia de que estos no llevan ningún diseño electrónico en su interior. Por otro lado, el intérprete es virtual, este traduce el código a lenguaje java mediante una foto de los bloques previamente ordenados por el usuario, haciendo uso del procesamiento de imágenes. La lógica completa que sigue el sistema TactusLogic se puede apreciar en la Figura 1. Además, proporciona asistencia al usuario a través de la detección de errores que realiza el compilador mediante comentarios y a través de la función "ayuda", la cual proporciona instrucciones sobre el uso del sistema [13]. Al no existir un intérprete del mundo real, no se pueden apreciar los resultados del código creado de manera interactiva, resultando una plataforma de programación no muy atractiva para niños.

Figura 1. Diagrama de bloques de TactusLogic

2.1.3 Algorithmic Bricks

Es una herramienta para introducir a novatos al mundo de la programación a través de TPL y fue creada por Dai-Young Kwon, Han-Sung Kim, Jae-Kwoun Shim, y Won-Gyu Lee. La investigación realizada por Kwon detalla que una plataforma de programación tangible debe tener cinco características, siendo la fácil detección de

errores a través de la depuración la primera de estas. La segunda es que deben existir instrucciones concretas, la tercera indica que los comandos deben ser sencillos de asimilar, la cuarta hace referencia a una implementación cooperativa o grupal y la última de estas características es que debe ser de bajo costo para que pueda ser fácilmente adquirida [12]. A diferencia de las plataformas anteriores, Algorithmic Bricks contiene un intérprete robot conformado por tres sensores ubicados en los lados frontal, izquierdo y derecho. Estos sensores al detectar una línea negra reaccionan ejecutando el código recibido previamente. Este código es creado mediante bloques tangibles denominados A-bricks y pueden ser manipulados de dos maneras: colocándolos uno sobre otro o de manera horizontal, ver Figura 2. Se realizó un estudio de eficacia para comparar esta plataforma y Scratch como herramientas útiles para enseñar la programación a escolares y, se comprobó que la programación tangible es mejor asimilada que la programación virtual como se puede observar en la Figura 3. Además, los escolares sometidos al estudio describieron a A-bricks como una plataforma de fácil uso y de mejor manejo que Scratch y, por otro lado, declararon que los puntajes bajos obtenidos en los niveles 9 y 10 se debían más a la difícil utilización del intérprete robot, mientras que los puntajes bajos de Scratch se daban por el mismo entorno y lenguaje de programación de dicha plataforma virtual [14].

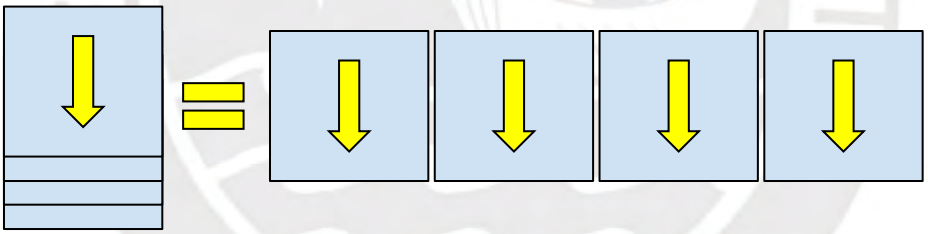


Figura 2. Formas equivalentes de la colocación de los bloques tangibles

2.1.4 Proteas

Proteas es un kit de programación creado por Theodosios Sapounidis y Stavros Demetriadis y consta de dos lenguajes de programación: T_Butterfly y T_ProRob, los cuales se pueden usar de manera complementaria. T_Butterfly, el primero de ellos, es tangible y permite guiar al intérprete, una mariposa virtual, a través de un laberinto. Está compuesto por bloques de tamaño regular para que puedan ser fácilmente sujetados por niños. Por otro lado, el lenguaje T_ProRob, que también es tangible, tiene como intérprete al robot Lego NXT y está compuesto por bloques tangibles de

menor tamaño que los anteriores. Además, este segundo lenguaje de programación permite la reutilización de la codificación y provee asistencias al usuario mediante la verificación del código a través de indicaciones visuales en cada bloque [15]. La desventaja de T_butterfly es que requiere de un intérprete virtual, por lo que requiere de una computadora, limitando la edad de uso y restringiendo el sector económico y, por otro lado, T_ProRob requiere de un intérprete costoso (robot Lego NXT), por lo que no es accesible para todos los interesados en adquirir el kit.

2.1.5 P-Cube

Fue desarrollado por Shun Kakehashi, Tatsuo Motoyoshi, Ken'ichi Koyanagi, Toru Ohshima y Hiroshi Kawakami como una respuesta a la falta de herramientas que permitan enseñar la programación a niños con discapacidades visuales. Consiste en un conjunto de bloques, un tablero de programación, un robot y una computadora [16]. Cada bloque representa un comando como "adelante", "girar a la derecha/izquierda", "alto", además existen dos bloques de control: "bucle" y "condicional". La limitación de esta plataforma es la necesidad de una computadora para poder cargar el código creado a una tarjeta de memoria SD y recién introducir esta tarjeta al robot que es el intérprete, haciendo muy engorroso y lento el proceso de ejecutar dicho código, ver Figura 3.

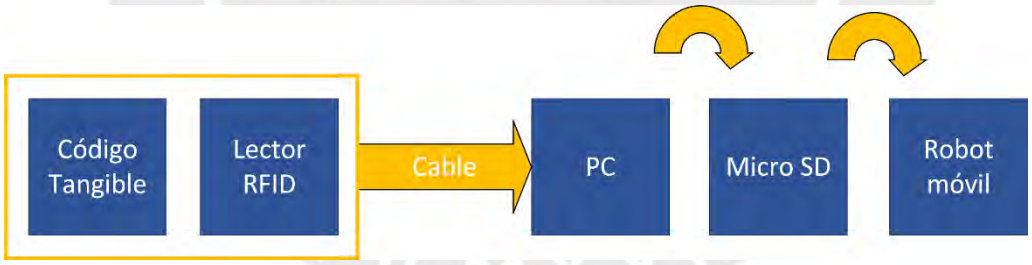


Figura 3. Componentes de la plataforma P-Cube

2.1.6 FYO

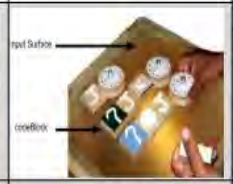
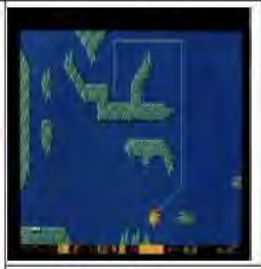
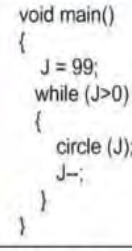
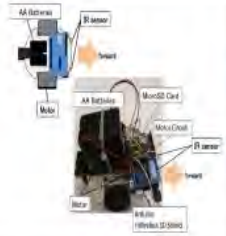
Es una plataforma de programación tangible desarrollada por Pablo Cárdenas que consiste en un tablero, bloques tangibles y un robot como intérprete; además, introduce conceptos de programación como la depuración y "llamar a una función". La principal motivación de la creación de esta plataforma fue la falta de una herramienta que

permita la enseñanza de la programación en países en pleno desarrollo como Perú, pues las personas aprenden a programar recién en la etapa universitaria y solo si es que se sigue una carrera orientada a la ingeniería y/o tecnología [12]. La limitación de este sistema es que, a pesar de ser diseñada para ser una herramienta de bajo costo, no está orientada a la enseñanza simultánea de la programación de un salón de clases completo, por lo que se necesitaba un tablero por cada alumno en el mejor de los casos, incrementando los precios y haciéndolo no accesible.

Como se ha visto, diferentes desarrolladores han hecho uso de la programación tangible como herramienta principal para la enseñanza de la programación a niños de cortas edades y sin experiencia previa en esta. Se encuentran algunas diferencias entre el tipo de intérprete que maneja cada plataforma, la sintaxis del lenguaje, el entorno de programación y algunas otras características principales que se mostrarán en la Tabla 1. Se puede observar que las plataformas desarrolladas logran reducir la brecha de edad para la enseñanza de la programación, pero ninguna tomó en cuenta los currículos nacionales de educación del país en el que fueron probadas, ni la enseñanza simultánea de esta ciencia computacional a un salón de clases completo, resultando muy costosas.

Tabla 1

Tabla comparativa de las diferentes plataformas de programación tangible

	PLATAFORMAS						
CARACTERÍTICAS	**ALGOBLOCK**	**TACTUSLOGIC**	**ALGORITHMIC BRICKS**	**PROTEAS**		**P-CUBE**	**FYO**
				T_BUTTERFLY	T_PROROB		
DESARROLLADORES	H. Suzuki y H. Kato	Andrew Cyrus Smith	Dai-Young Kwon	Theodosios Sapounidis y Stavros Demetriadis		Shun Kakehashi et Al.	Pablo Cárdenas
SINTAXIS DEL LENGUAJE DE PROGRAMACIÓN	**BLOQUES COMPUESTOS**	**BLOQUES DE COMANDOS Y PARÁMETROS**	**BLOQUES COMPUESTOS**	**BLOQUES COMPUESTOS**	**BLOQUES COMPUESTOS**	**BLOQUES COMPUESTOS**	**BLOQUES DE COMANDOS Y PARÁMETROS**
ENTORNO DE PROGRAMACIÓN							
	TANGIBLE	**TANGIBLE**	**TANGIBLE**	**TANGIBLE**	**TANGIBLE**	**TANGIBLE**	**TANGIBLE**
INTÉRPRETE							
	Virtual: submarino	**Virtual: java**	**Robot**	**Virtual: mariposa**	**Robot Lego NXT**	**Robot**	**Robot**
EDAD RECOMENDADA	**5+**	**6+**	**6+**	**4+**	**6+**	**5+**	**5+**

2.2 Marco Teórico

El Marco teórico abarcará las consideraciones necesarias para el diseño de una plataforma de programación tangible, la cual incluye al lenguaje de programación, el entorno de programación y el intérprete.

2.2.1 Lenguaje de programación

La creación de un nuevo lenguaje de programación con propósito educativo debe tomar en cuenta el análisis de los paradigmas de programación, los cuales son: programación imperativa procedural, programación funcional y programación orientada a objetos [17]. Además, algunos autores [18], [19] sugieren la programación basada en reglas como un paradigma atractivo para principiantes en las ciencias computacionales.

2.2.1.1 Programación imperativa procedural

Este paradigma es uno de los más utilizados y conocidos en el proceso de la programación y, como se puede intuir por el nombre, el usuario será capaz de desarrollar programas a través de procedimientos (conjuntos de bloques de código ejecutable). Además, es secuencial y utiliza conceptos muy cercanos al lenguaje máquina como el acceso a la memoria, lo cual lo hace eficiente pero complejo. Algunos lenguajes de programación de este estilo son C, Java y Pascal [20].

2.2.1.2 Programación funcional

El paradigma funcional está orientado a la construcción de funciones basadas en el cálculo matemático y cuya característica principal, a diferencia de la programación imperativa procedural, es que se suprimen los tipos de datos y utiliza como herramienta a la recursividad. A pesar de ser poco eficiente, una de sus ventajas es que es de fácil aprendizaje para aquellos con conocimientos matemáticos previos [21]. Clojure, Erlang y Haskell son lenguajes de programación que corresponden a este estilo [20].

2.2.1.3 Programación basada en reglas

La programación basada en reglas permite crear programas usando una base de datos compuesta por condiciones y acciones, que implica que las acciones se realicen solo si es que se ha cumplido algún antecedente condicional [18]. Por otro lado, al usar reglas y acciones definidas por el programador, se vuelve más intuitiva pues está basada plenamente en el razonamiento del usuario, haciéndose más personalizada.

2.2.2 Entorno de programación

El entorno de programación abarca las piezas tangibles que contendrán el lenguaje de programación, la captura de una fotografía a dichas piezas tangibles y la aplicación móvil, que permitirá visualizar el resultado de la compilación del código y la interacción con el usuario, ver Figura 4.

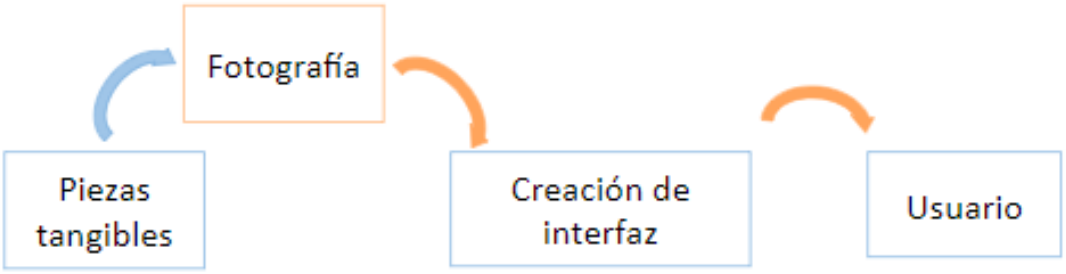


Figura 4. Entorno de programación

Figura 4. Entorno de programación

2.2.3 Intérprete

El diseño electrónico en el interior del intérprete el procesamiento de la imagen obtenida a través de la fotografía de las piezas se realizará mediante librerías acondicionadas para Python, lenguaje escogido por su portabilidad y potencia [22].

2.2.3.1 NumPy

Esta librería, fundamental para la computación científica, es usada para ejecutar operaciones matemáticas de alto nivel con soporte para matrices y arreglos. Una imagen puede ser vista como una matriz que contiene datos de los píxeles y, por lo tanto, operaciones básicas de la librería en cuestión como "cortar y "enmascarar" permiten variar los valores de dichos datos [23].

2.2.3.2 SciPy

Scipy, al igual que NumPy, es de uso científico y proporciona paquetes para poder realizar el procesamiento de imágenes a través de funciones como filtros lineales y no lineales, funciones de morfología binaria, mediciones de objetos, entre otras. Cabe resaltar que esta librería también hace uso de arreglos y matrices, de forma similar que NumPy, para trabajar con imágenes [24].

2.2.3.3 OpenCV-Python

Esta librería además de ser compatible con la Biblioteca de procesamiento de imágenes de Intel que contiene operaciones de bajo nivel (binarización, filtrado, estadísticas, etc.), permite implementar funciones como la calibración de imágenes, detección de características, seguimiento, análisis de forma y movimiento, entre otras. En adición, es considerada una de las librerías de procesamiento de imágenes más rápidas, dado que su background está escrito en C/C++ [25], [38].

Debido a que la interacción con el usuario se realizará a través de una aplicación móvil, se hará uso de un servidor web para poder acceder a esta a través de cualquier dispositivo con acceso a internet.

2.2.3.4 Servidor Web

Un servidor web permite almacenar diferentes tipos de archivos como texto, imágenes y videos y, finalmente, muestra estos mediante navegadores de Internet.

Se hace uso del protocolo HTTP (*Hipertext Transfer Protocol*). El espacio que brindan estos servidores para alojar una página web se denomina *hosting* y están compuestos por archivos de diferentes tipos, entre ellos están los siguientes: HTML, CSS, PHP.

En la Tabla 1 del Estado del Arte, se pudo observar que existe una gran variedad de intérpretes (robots móviles) para cada plataforma de programación propuesta y, una de las diferencias más notables entre estos fue que dicho intérprete sea tangible o virtual, resaltando los múltiples beneficios de que este sea tangible. Por otro lado, en base a diversos estudios sobre robots orientados a fines educacionales, se destaca que características como la no distinción de género y una morfología cuadrada son preferidas por los niños [26], así como características zoomórficas o antropomórficas, en ambos casos con presencia de extremidades, aunque estas solo sean simbólicas [27].

Sin embargo, la característica principal a tomar en cuenta a la hora de diseñar un robot móvil es el terreno sobre el que este se desplazará y, de acuerdo a este detalle y basándose en la clase de locomoción, se pueden clasificar a los robots móviles en tres principales tipos, robots móviles con ruedas, robots móviles con patas (bípedos, cuadrúpedos, etc.), y, finalmente, robots móviles tipo oruga [28].

2.2.3.5 Robots Móviles con ruedas

Esta es la opción más utilizada debido a su baja complejidad en el diseño mecánico y su alta eficiencia (menos potencia consumida con respecto a la distancia recorrida). Existen diferentes robots móviles de este tipo [29], [30] desarrollados con fines educacionales. La diferencia entre estos radica principalmente en el sistema de control, el número de actuadores y el número de sensores. Sin embargo, la eficiencia de estos sistemas se ve afectada cuando existen variaciones en el terreno inesperadas debido a la fricción o, si es que han sido probados en un terreno específico, y luego este es cambiado por uno con características diferentes.

2.2.3.6 Robots móviles con patas

Los robots móviles con patas, a diferencia de los robots con ruedas, superan el obstáculo de un terreno difícil o abrupto. Otra característica favorable de este tipo de sistemas es que representan de mejor forma características humanoides o zoomórficas. Se han desarrollado muchos robots de este tipo alrededor del mundo

[31], [32] y la principal dificultad que han encontrado los creadores ha sido el diseño mecánico debido a la cantidad de grados de libertad que requiere este tipo de locomoción.

2.2.3.7 Robots móviles tipo oruga

Estos sistemas hacen uso de pistas de deslizamiento, lo cual implica una mayor maniobrabilidad sobre terrenos abruptos y menos complejidad en el diseño mecánico con respecto a los robots móviles con patas, pues solo se necesitan actuadores de tracción o rudas [28]. Por otro lado, uno de los principales problemas de los robots móviles tipo oruga es que consumen mucha potencia por rozamiento, por lo que se debe analizar si es que el sistema con una locomoción tipo oruga es realmente necesario para el tipo de aplicación al que está destinado.

La tesis diseñará un intérprete robot con una locomoción basada en ruedas, por lo que a continuación se detallarán las diferentes configuraciones de ruedas que existen en esta categoría. Se revisarán la configuración Ackerman, la configuración omnidireccional y, finalmente, la configuración diferencial.

2.2.3.8 Configuración Ackerman

Esta es una de las configuraciones más usadas por su fácil implementación y es la que normalmente se observa en la mayoría de vehículos terrestres. Presenta cuatro ruedas, dos delanteras y dos traseras, donde las delanteras son las que permiten el giro sobre el eje y las traseras, aseguran la estabilidad, ver Figura 5. La principal desventaja de estos robots móviles es que presentan restricciones no holónomas; es decir, el número de grados de libertad total no es igual al número de grados controlable.

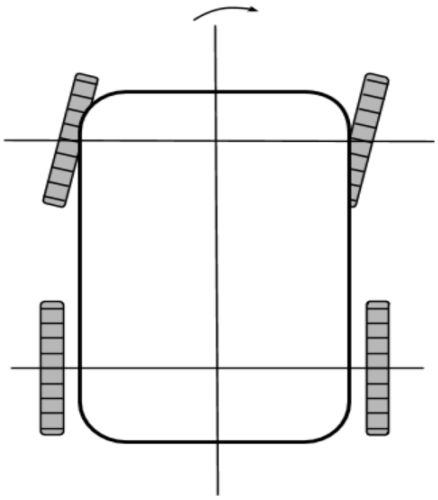


Figura 5. Robot móvil con configuración Ackerman

2.2.3.9 Configuración omnidireccional

A diferencia de la configuración anterior en la que el robot móvil usa ruedas convencionales, la configuración omnidireccional hace uso de dos o más omni-ruedas (ruedas con discos perpendiculares a la dirección de giro), que le otorgan un movimiento flexible de alta precisión, ver Figura 6. De esta manera, los robots de ruedas omnidireccionales pueden realizar movimientos complicados, pues se reducen las restricciones cinemáticas; sin embargo, no garantizan un movimiento en línea recta, siendo esta la principal desventaja [33].

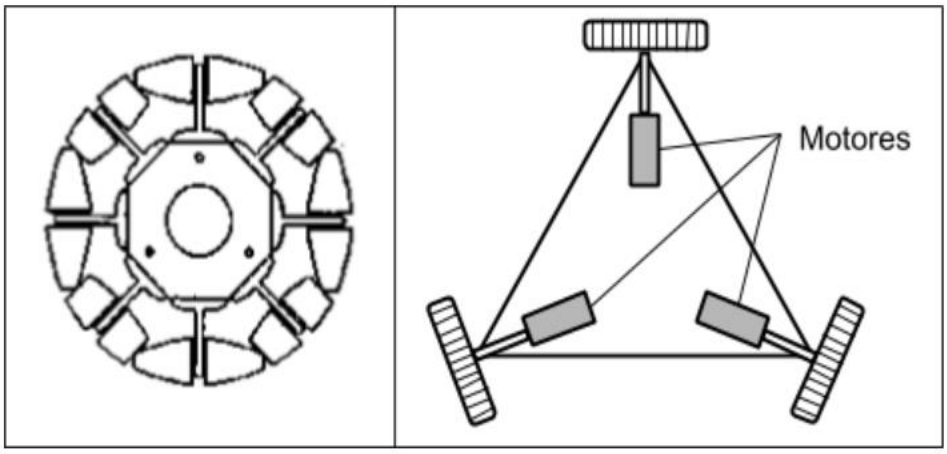


Figura 6. Vista superior de una omni-rueda y un robot móvil omnidireccional de 3 ruedas

2.2.3.10 Configuración diferencial

Los robots móviles con este tipo de configuración presentan tres ruedas, donde las dos primeras, las principales, sirven para manejar el movimiento del robot y la tercera, para garantizar su estabilidad, como se puede observar en la Figura 7. Asimismo, presenta dos grados de libertad y ambos motores existentes se encuentran alineados en un mismo eje [34]. La posición y la velocidad del robot se pueden controlar mediante *encoders*, por lo que la orientación del vehículo es una función del desplazamiento de ambas ruedas.

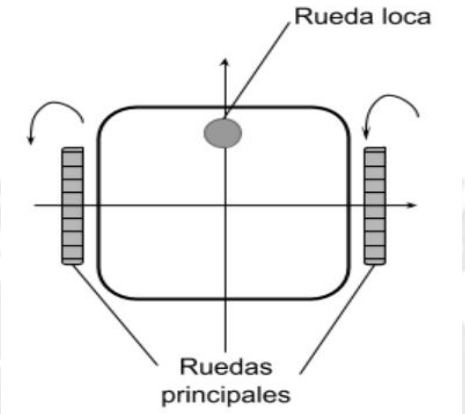


Figura 7. Robot móvil con configuración diferencial

Como se ha visto, cada configuración presenta ventajas y desventajas, pero de acuerdo a los requerimientos de la tesis y el análisis de las opciones mencionadas, se escogerá una configuración diferencial, ya que es la de menor complejidad en el diseño mecánico, garantiza el movimiento en línea recta y presenta el mejor performance para realizar un control de seguimiento de trayectoria [34]. Además, los robots móviles con configuración *ackerman*, debido a sus restricciones no holónomas, y con configuración omnidireccional, debido su dificultad para seguir movimientos en línea recta, dificultan el diseño del algoritmo de control de trayectorias.

El último de los objetivos específicos presentados en el Capítulo 1 es realizar el control de trayectorias del intérprete robot móvil con la mayor precisión posible, por lo que se estudiaron las diferencias existentes entre un control de desplazamiento y un control de trayectoria. Se debe recalcar que el desplazamiento hace referencia a la distancia lineal existente entre un punto de partida y uno de llegada, mientras que una trayectoria es la ruta o recorrido realizado para llegar al punto de llegada.

2.2.3.11 Estrategia de control de desplazamiento y orientación basada en el modelo cinemático del robot diferencial

Esta estrategia ha sido diseñada específicamente para un robot móvil diferencial con las características anteriormente mencionadas, el cual origina sus movimientos al girar a diferente velocidad cada motor correspondiente a cada una de las dos ruedas principales. A continuación, en la Figura 8, se puede observar la estructura del robot con respecto a un punto P ubicado en el punto medio del eje entre las dos ruedas.

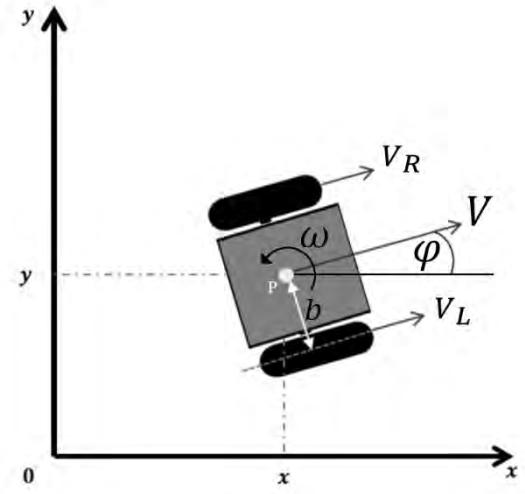


Figura 8. Estructura del robot móvil diferencial

$$V = \frac{V_R + V_L}{2} \tag{1}$$

$$\dot{\varphi} = \omega = \frac{V_R - V_L}{2b} \tag{2}$$

Donde, V es la velocidad lineal del robot, V_R y V_L son las velocidades lineales de las ruedas derecha e izquierda respectivamente, φ es la orientación angular del robot y ω es la aceleración angular del robot.

$$\dot{x} = Vcos(\varphi) \tag{3}$$
$$\dot{y} = Vsen(\varphi) \tag{4}$$
$$\dot{\varphi} = \omega \tag{5}$$

Donde, x e y representan la posición lineal del robot en el plano cartesiano.

Entonces, el sistema cinemático será visto de la siguiente manera, donde, ω_R y ω_L son las entradas y x, y y φ son las salidas.

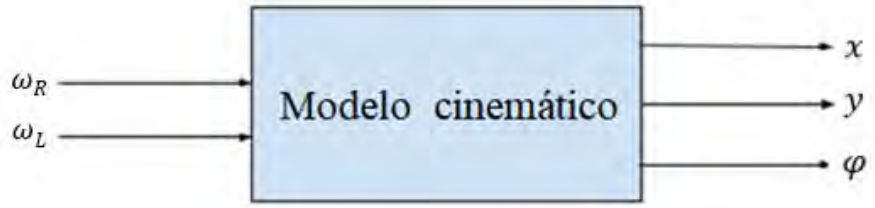


Figura 9. Representación de entradas y salidas del sistema cinemático

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \frac{r\cos(\varphi)}{2} & \frac{r\cos(\varphi)}{2} \\ \frac{r\,sen(\varphi)}{2} & \frac{r\,sen(\varphi)}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{6}$$

La ecuación número 6 representa el modelo cinemático matricial del robot diferencial en cuestión, que en base a un control por realimentación de estados permite definir la posición del robot en el plano cartesiano XY y su orientación a través de φ. La principal limitación de este tipo de control es que para poder definir el recorrido del intérprete se necesitará dividir la trayectoria deseada en múltiples desplazamientos lineales, por lo que un recorrido muy complicado sería engorroso de seguir. Esto se puede notar en la Figura 10, donde la curva 2 representa el recorrido original y la curva 1 representa el recorrido divido en varios desplazamientos lineales.

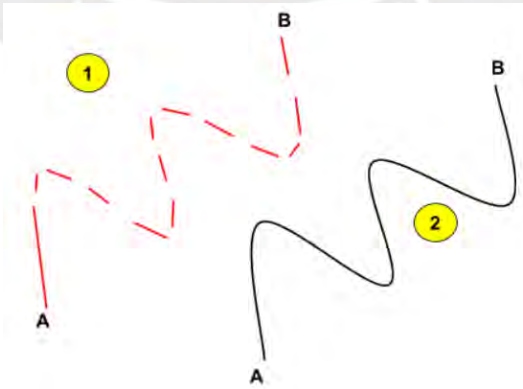


Figura 10. Diferenciación entre un controlador de desplazamiento y uno de trayectoria

2.2.3.12 Estrategia de control de trayectoria basada en el modelo cinemático del robot diferencial

Esta estrategia implica tomar como entradas del sistema cinemático a V y ω, por lo que se obtiene en el nuevo modelo matricial presentado a continuación [36].

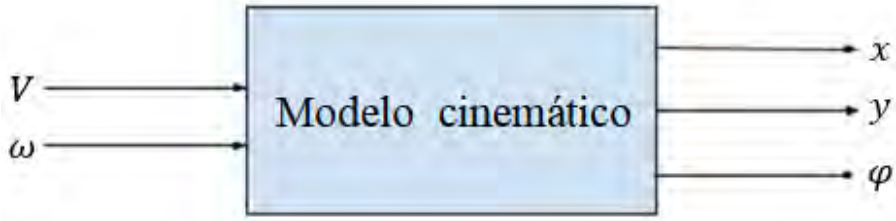


Figura 11. Nueva representación de entradas y salidas del sistema cinemático

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\,(\varphi) & 0 \\ sen(\varphi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{7}$$

El siguiente paso es mover el punto P una distancia a y se obtiene el esquema de la Figura 12.

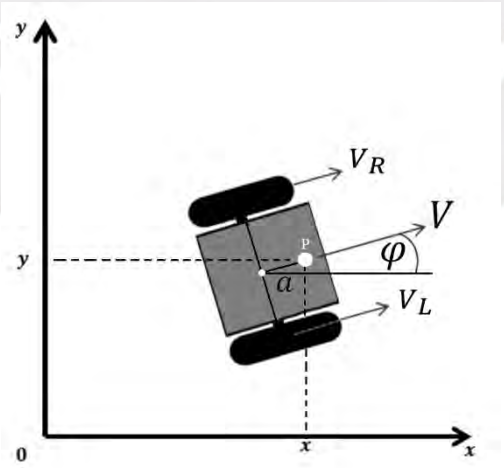


Figura 12. Estructura del robot diferencial con el punto P desplazada una distancia a

Cabe resaltar que este modelo cinemático realiza el control de trayectoria sobre el punto P ya desplazado, por lo que se obtiene el modelo matricial de la ecuación 8.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\end{bmatrix} = \begin{bmatrix}\cos(\varphi) & -asen(\varphi)\\ sen(\varphi) & a\cos(\varphi)\\ 0 & 1\end{bmatrix}\begin{bmatrix}V\\ \omega\end{bmatrix} \tag{8}$$

Como se desea realizar un control de trayectorias, no se tomará en cuenta la orientación del robot como salida del sistema. Entonces, el sistema expresado de forma matricial quedará de la siguiente manera:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = M\begin{bmatrix}V\\ \omega\end{bmatrix} \tag{9}$$

Donde,

$$M = \begin{bmatrix}\cos(\varphi) & -asen(\varphi)\\ sen(\varphi) & a\cos(\varphi)\end{bmatrix} \tag{10}$$

Finalmente, se propone la siguiente Ley de Control que permitirá eliminar los elementos no lineales de M:

$$\begin{bmatrix}V\\ \omega\end{bmatrix} = M^{-1}\begin{bmatrix}K_1 x_e + \dot{x}_d\\ K_2 y_e + \dot{y}_d\end{bmatrix} \tag{11}$$

Donde,

$$x_e = x_d - x \tag{12}$$

$$y_e = y_d - y \tag{13}$$

Cabe resaltar que $K_{1,2}$ son las ganancias del controlador, x_d e y_d representan la posición deseada en el plano XY y x_e e y_e representan el error de posición.

Se debe tener en cuenta que la distancia a debe ser diferente de 0, pues la inversa de la matriz M contiene elementos con denominadores iguales a dicha distancia.

$$M^{-1} = \begin{bmatrix}\cos(\varphi) & sen(\varphi)\\ -\frac{sen(\varphi)}{a} & \frac{cos(\varphi)}{a}\end{bmatrix} \tag{14}$$

Además, reemplazando la Ley de Control en el modelo cinemático, se obtiene la ecuación que permitirá hallar las constantes $K_{1,2}$.

$$\begin{bmatrix}\dot{x}_e\\ \dot{y}_e\end{bmatrix} = \begin{bmatrix}-K_1 & 0\\ 0 & -K_2\end{bmatrix}\begin{bmatrix}x_e\\ y_e\end{bmatrix} \tag{15}$$

En base a todo lo previamente planteado, se procederá a realizar el diseño del controlador que se documentará en el Capítulo 2 y cuyo esquema se muestra a continuación:

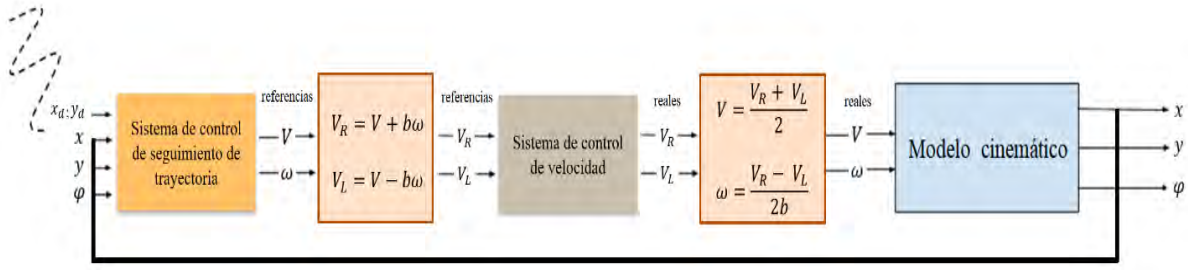


Figura 13. Esquema del controlador del robot móvil diferencial

2.3 Marco Metodológico

El diseño de la solución para el problema expuesto en la Introducción contempla como primer paso el análisis de las plataformas de programación tangibles desarrolladas en el Marco Histórico del Capítulo 1. Es necesario ubicar las ventajas y desventajas de cada plataforma para poder utilizar sus puntos fuertes y evitar los puntos débiles, así como desarrollar el Marco Teórico y Marco Metodológico. De esta manera, se puede llegar al paso dos que implica el diseño de una **Plataforma de Programación Tangible** que permita cumplir de manera exitosa el objetivo general planteado anteriormente.

Con respecto al diseño de esta plataforma, se tomará como referencia a la norma alemana VDI 2206 [37], la cual contempla las siguientes etapas: la elaboración de la lista de exigencias, el diseño del lenguaje de programación tangible, la elaboración de la estructura de funciones, la elaboración del concepto de solución y la elaboración del proyecto definitivo. Al completar el paso dos, se tendrán listos los diseños del lenguaje de programación, entorno de programación e intérprete que permitirán el diseño de dicha plataforma y quedarán plasmados en el Capítulo 2 de la tesis. El tercer paso, que implica el modelamiento matemático del sistema para obtener gráficas, simulaciones y el análisis del diseño final, estará contenido en el Capítulo 3. Finalmente, los resultados y observaciones finales estarán consideradas dentro del cuarto y último paso y, serán parte de la redacción del Capítulo 4. La tesis finiquitará con las conclusiones y recomendaciones del trabajo final.

Capítulo 3

Diseño de la plataforma de programación tangible

El presente trabajo de tesis, como se mencionó en el Marco Metodológico, toma como referencia a la norma de diseño alemana VDI 2206, la cual contempla como primera etapa a la elaboración de la lista de exigencias con las características principales que deberá tener la TPL. Las siguientes etapas serán el diseño del lenguaje de programación tangible, la elaboración de la estructura de funciones, el concepto de solución y, finalmente, la cuarta etapa es el proyecto definitivo.

3.1 Elaboración de la lista de exigencias

La función principal del proyecto está contenida en el objetivo principal de la tesis, el cual es diseñar una herramienta educativa para escolares de la Educación Básica Regular (EBR) que les permita desarrollar el pensamiento computacional a través de la programación tangible. La TPL deberá estar compuesta por tres partes principales: lenguaje, entorno e intérprete. El entorno de programación contempla a las piezas tangibles de programación, el dispositivo móvil y la interacción con el intérprete que será a través del procesamiento de la fotografía de estas piezas. El intérprete es un robot móvil con locomoción a ruedas de tipo diferencial que deberá seguir una trayectoria definida por las piezas tangibles. Este modelo de solución debe cumplir con una lista de requerimientos (planteada en el Anexo 1) y cuya descripción se encuentra en la Tabla 2.

Tabla 2
Lista de exigencias con las características principales

Características	Descripción
Geometría	De acuerdo a las diferentes TPL estudiadas en el Marco Histórico, el tamaño recomendado para el intérprete debe ser de máximo $25\ cm \times 25\ cm \times 25\ cm$ además tendrá morfología animal ya que es un producto orientado a niños. Las piezas tangibles serán cuadradas, sin bordes filosos y de $10\ cm \times 10ccm \times 1.5\ cm$ para evitar la dificultad visual y el peligro de ser ingeridas.
Cinemática	Se diseñará un robot móvil con locomoción basada en ruedas de tipo diferencial como se explicó en el Marco Tórico. La velocidad del intérprete debe ser la adecuada para que esta trayectoria pueda ser fácilmente visible, por lo que esta será 1 longitud/3 s (siendo 1 longitud = 0.25 m), velocidad tomada de [12].
Control	Se realizará un control de trayectoria basado en el modelo cinemático para un robot diferencial [36]. Este implica un error en estado estable igual a 0.
Electrónica (Hardware)	La autonomía planteada será de 3 horas como mínimo, lo cual es equivalente a 4 horas pedagógicas, cuya duración está establecida en el Capítulo V de Ley N° 24029, Ley del profesorado [40]. Esta autonomía garantiza por lo menos 2 sesiones de enseñanza haciendo uso contino de la plataforma. Por otro lado, las piezas tangibles no tendrán diseño electrónico en su interior.
Software	Se diseñará un lenguaje de programación basado en reglas, ya que como se mencionó en el Marco Teórico, este paradigma (con instrucciones atrás, adelante, derecha, izquierda) es el más atractivo para principiantes en las ciencias computacionales. Esto se comprueba en el Marco Histórico, ya que es el tipo de programación más usado.
Comunicaciones	La comunicación entre las piezas tangibles y el intérprete robot será inalámbrica. De este modo los bloques no tendrán diseño electrónico en su interior, lo que permitirá disminuir el costo de las piezas tangibles y facilitarles su manipulación a los usuarios (niños).

Seguridad	Se planteará el diseño del chasis del intérprete en base al uso del material PLA que, al ser compostable, permitirá un proceso de eliminación más eco-amigable cuando termine su ciclo de vida y se convierta en *e-waste* (desperdicio electrónico) [41]. Las piezas tangibles serán de madera para reducir el impacto medioambiental. Además, se respetará el Anexo IV del Reglamento de la Ley N° 28376, Ley que prohíbe y sanciona la fabricación, importación, distribución y comercialización de juguetes y útiles de escritorio tóxicos o peligrosos [39]. Este anexo lista los elementos que deben ser controlados de acuerdo a su concentración en la fabricación de juguetes.
Ergonomía	Se tomarán como referencia las recomendaciones del peso máximo que puede cargar un estudiante en una mochila brindadas por el Instituto Nacional de Salud [42]. Esta información plantea que un estudiante puede cargar como máximo el 15% de su peso para no ir en contra de la ergonomía, por lo que las piezas tangibles y el robot no deben superar los 3 kg y de este modo, puedan ser fácilmente manipulables.
Costos	La tesis plantea una alternativa de bajo costo, por lo que se tomará como referencia la segunda fase de la iniciativa "Una computadora por niño" que se llevó a cabo en el Perú. En este proyecto, se repartieron sets *WeDo* de la empresa Lego cuyo costo es de 214 dólares americanos por set [43]. Por este motivo, el diseño de la Plataforma de Programación Tangible contemplará materiales y componentes que cumplan los requerimientos previamente listados y que no superen este monto, incluyendo el costo del diseño.

3.2 Diseño del lenguaje de programación

En el Marco Teórico, se desarrollaron las características de los principales paradigmas de programación: imperativa procedural, funcional y basada en reglas. La tesis diseñará un lenguaje de programación basado en reglas, ya que muchos autores [11-16] han obtenido buenos resultados en plataformas de programación tangibles con este paradigma. Además, se introducirán las funciones, puesto que es básico para todo aquel que desee aprender a programar.

Se pueden adaptar los estudios realizados al lenguaje, propiamente dicho, para poder comprender la semiótica de los leguajes de programación [44]. Tomando como referencia a Morris, quien realizó un estudio de la semiótica en su obra "*Foundations of the Theory of Signs*", se pueden dividir a los signos en tres partes [45]. Estas partes serán definidas según la Real Academia de la Lengua Española [46] y se explicará cómo es que se relacionan con un lenguaje de programación, ver Tabla 3.

Tabla 3
Disciplinas de la semiótica aplicadas a los lenguajes de programación

Disciplina	**Área de estudio**	**Relación con un lenguaje de programación**
Sintaxis	Modo en que se combinan las palabras y los grupos que estas forman para expresar significados, así como las relaciones que se establecen entre todas esas unidades.	Formato de los programas y conjunto de reglas que deben seguirse para que estos se consideren correctos
Semántica	Significado de las unidades lingüísticas y de sus combinaciones.	Comportamiento de los programas.
Pragmática	Disciplina que estudia el lenguaje en su relación con los hablantes, así como los enunciados que estos profieren y las diversas circunstancias que concurren en la comunicación.	Técnicas empleadas para la construcción de programas.

El diseño del lenguaje de programación se realizará tomando en cuenta las disciplinas previamente mencionadas (sintaxis, semántica y pragmática) y, además, se tendrá como referencia los principios de diseño de un lenguaje de programación [44].

Es importante identificar estos principios y aplicarlos, ya que garantizarán un buen diseño. Por este motivo, los más importantes se listarán a continuación:

- Concisión notacional
  El lenguaje debe ser de ayuda para el programador desde antes de la codificación, claro, simple, conciso y con una sintaxis legible.

- Expresividad
  El usuario debe ser capaz de poder expresar sus intenciones sin caer en la falta de seguridad usuario-plataforma.

- Eficiencia
  El lenguaje debe permitir la creación de programas eficientes y, además, diferentes técnicas de optimización.

3.2.1 Diseño de la sintaxis y especificación de la semántica

Como se mencionó en la Tabla 3, el diseño de la sintaxis del lenguaje de programación permitirá describir su estructura y la especificación de la semántica definirá qué hace el programa y qué acciones realizará el intérprete de acuerdo a la sintaxis previamente diseñada.

Como se trata de un lenguaje de programación que será usado en una Plataforma de Programación Tangible, se debe tener en cuenta que este lenguaje le permitirá al usuario codificar a través de piezas tangibles. Cada una de estas piezas contendrá un comando, parámetro o instrucción a los que se llamarán "bloques". Tomando como referencia a los autores revisados en el Marco Histórico y la programación basada de reglas, se han definido los siguientes bloques:

- Bloques de comandos

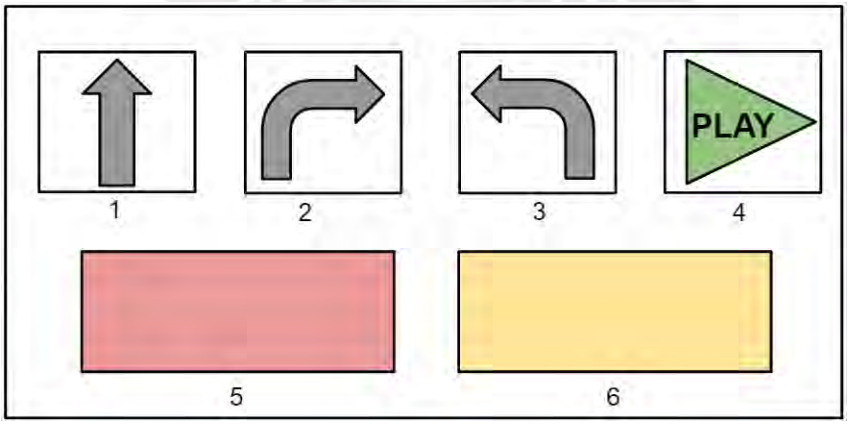


Figura 14. Representación gráfica de los bloques de comando

Como se puede ver en la Figura 14, existen seis bloques de comando: avanzar, girar a la derecha, girar a la izquierda, "*play*" para llamar a una función, un bloque rojo, denominado "end", el cual representará el fin de la función principal y un bloque amarillo, denominado "go to F1", que indicará el fin de una función secundaria. Estos dos últimos bloques no contendrán letras por fines didácticos, ya que servirán como un tope.

- Bloques de parámetros

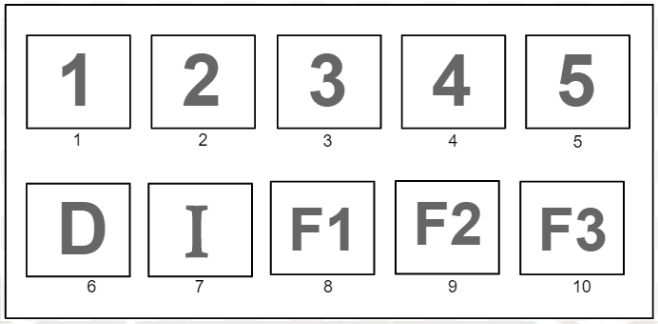


Figura 15. Representación gráfica de los bloques de parámetros

Los bloques de parámetros (Figura 15) son diez en total: números de 1 al 5 para limitar la distancia de recorrido (máximo 5 unidades) del intérprete robot, 2 bloques con las letras "D" e "I" para recalcar la dirección de giro y 3 bloques de funciones, donde F1 es la función principal.

Para poder crear un algoritmo, se considerarán las siguientes reglas:

- F1 se usará para llamar a la función principal y, los bloques F2 y F3 servirán para llamar a las funciones secundarias. No habrá diferenciación en la connotación entre la función principal y las funciones secundarias para facilitar la enseñanza de funciones a un novato en la programación.

- Una instrucción estará representada por un bloque de comando y un bloque de parámetro.

- El bloque de comando avanzar debe ir acompañado de un bloque de parámetro, un número entre 1 y 5, los cuales representan una distancia. El comando irá a la izquierda del parámetro como en el ejemplo de la Figura 16.

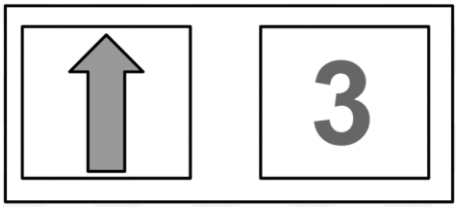


Figura 16. Instrucción que representa avanzar 3 unidades de distancia

- Los bloques de comando girar a la derecha y girar a la izquierda deben ir acompañados de la letra que representa la dirección de giro, los cuales representan girar y avanzar, simulando un cuarto de círculo. El ejemplo de la Figura 17 representa lo previamente mencionado.

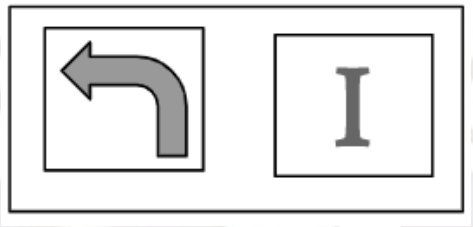


Figura 17. Instrucción que representa girar a la izquierda y avanzar

- Las instrucciones irán una debajo de la otra y ese será el orden en el que serán ejecutadas, ver Figura 18.

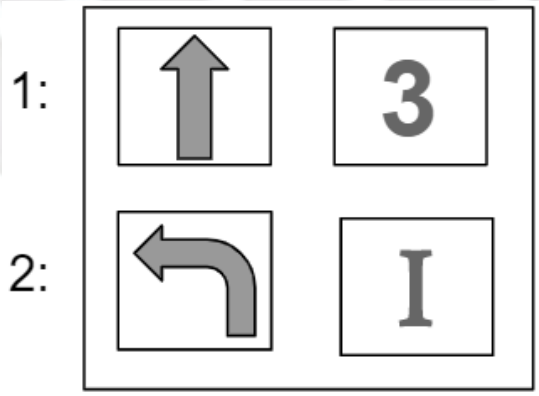


Figura 18. Porción de un programa que representa avanzar 3 unidades de distancia y luego girar a la izquierda

- Para llamar a una función, se usará el bloque de comando *"play"* y un bloque de parámetro (F1, F2, F3, ..., o F9) como en la Figura 19.

Figura 19. Instrucción que representa llamar a la función F2

En base a todas las reglas previamente mencionadas, la Figura 20 ejemplificará un programa completo y la Figura 21 representará cómo el intérprete robot ejecutaría dicho programa.

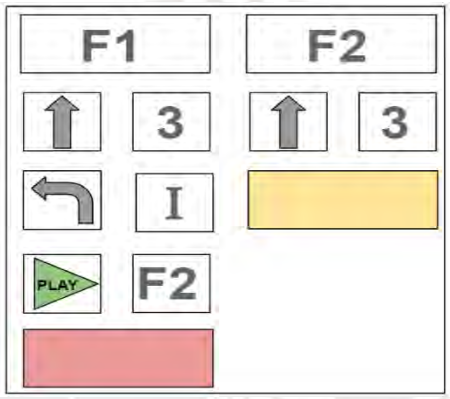


Figura 20. Programa que representa la función principal, F1, y la función secundaria, F2

20. Programa que representa la función principal, F1, y la función secundaria, F2

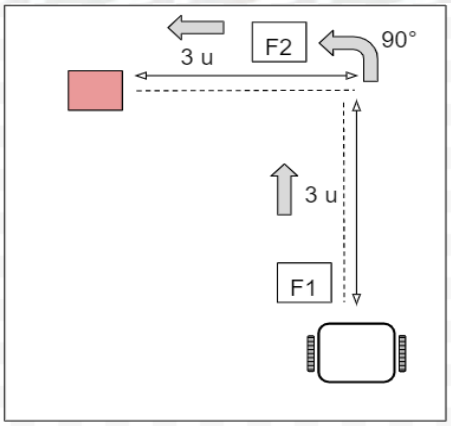
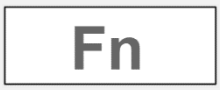
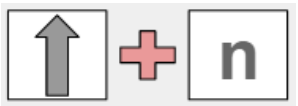
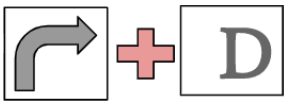
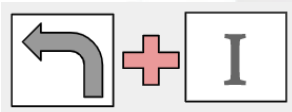
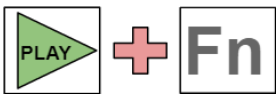


Figura 21. Trayectoria del robot al ejecutar al programa de la Figura 20

Además, el intérprete robot realizará una serie de acciones ante cada una de las diferentes instrucciones, las cuales se detallarán en la Tabla 4.

Tabla 4
Instrucciones y acciones del robot *móvil*

Instrucción	Acción
	- El intérprete no se mueve y espera otra instrucción. - Solo si n=1, se visualiza “Fn” en la pantalla y se emite un sonido.
	- El Intérprete robot se avanza en línea recta “n” unidades de distancia - No se visualizará la instrucción en la pantalla y no se emitirán sonidos.
	- El Intérprete robot gira a la derecha y avanza un cuarto de circunferencia. - No se visualizará la instrucción en la pantalla y no se emitirán sonidos.
	- El Intérprete robot gira a la izquierda y avanza un cuarto de circunferencia. - No se visualizará la instrucción en la pantalla y no se emitirán sonidos.
	- El intérprete no se mueve y salta a la instrucción “Fn”. - Se visualiza “Fn” en la pantalla y no se emiten sonidos.
	- El intérprete se detiene y no espera más instrucciones. - No se visualizará la instrucción en la pantalla, mas sí se emitirá un sonido.
	- El intérprete no se mueve y regresa a la línea en la que se quedó en F1. - No se visualizará la instrucción en la pantalla y no se emitirán sonidos.

3.3 Elaboración de la estructura de funciones

La elaboración de la estructura de funciones comprehende varias etapas que permitirán determinar las funciones que la Plataforma de Programación Tangible a diseñar deberá tener para que pueda cumplir con el objetivo principal de la tesis. A continuación, se listarán estas etapas: Abstracción, Determinación de los principios tecnológicos y la secuencia de operaciones, Fijación de procesos técnicos, determinación de la aplicación de los sistemas técnicos y sus limitaciones, agrupación de funciones, y, finalmente, la representación de la estructura de funciones.

3.3.1 Abstracción: Caja Negra

La etapa de abstracción tiene como objetivo expandir la mente del diseñador, con el fin de que explore todas las soluciones posibles antes de llegar al diseño final. Esta etapa plantea que cualquier función se puede representar en base tres magnitudes: señales (datos, impulsos de control, información), energía (mecánica, eléctrica, óptica, etc.) y materia (insumos, piezas, objetos de todo tipo). Se desarrollarán dos cajas negras, una para el entorno de programación y otra para el intérprete robot móvil.

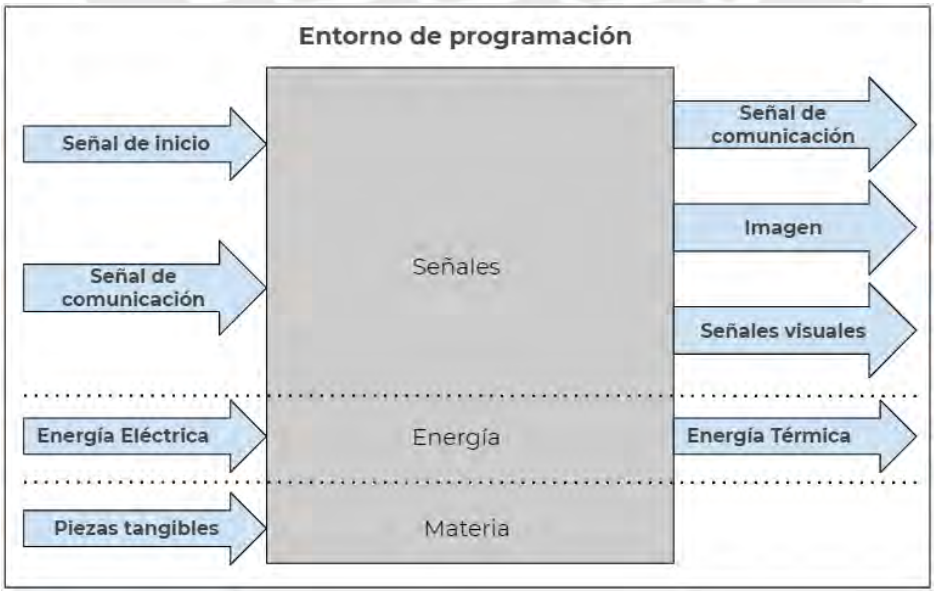


Figura 22. Caja negra del entorno de programación

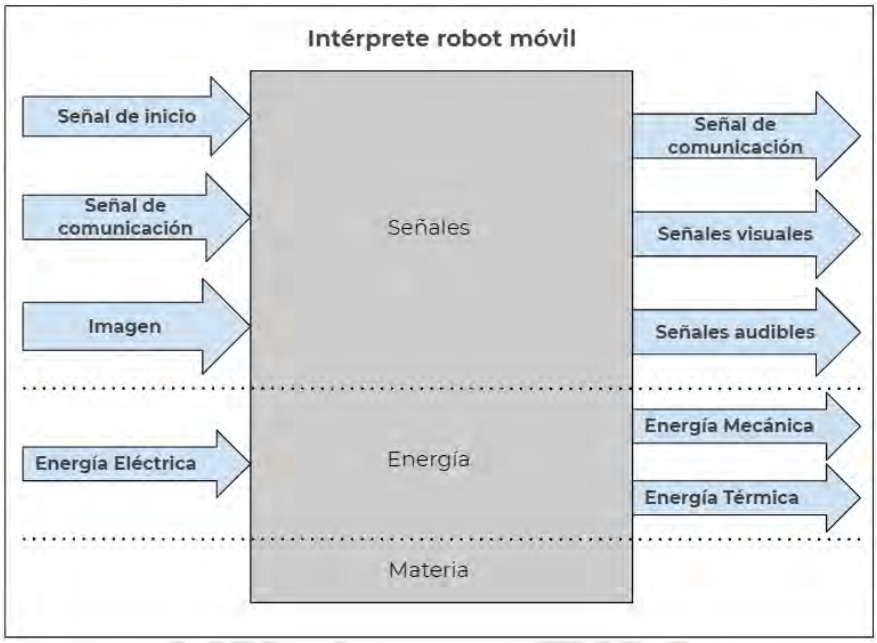


Figura 23. Caja negra del intérprete robot móvil

3.3.2 Principios tecnológicos y secuencia de operaciones

Los principios tecnológicos permitirán transformar las entradas de la caja negra obtener las salidas y de esta manera, completar el diseño de la Plataforma de Programación Tangible (TPL). Por este motivo, en el Marco Teórico se realizó una investigación sobre todas las tecnologías que serán de utilidad para poder cumplir con lo antes mencionado y, entre estas, destacan el procesamiento de imágenes, el diseño electrónico, servidores web y el diseño mecánico de un robot móvil.

La TPL que se desea diseñar seguirá la secuencia de procesos que se mencionará a continuación. En primer lugar, se prenderá el robot móvil y se enseñarán las piezas tangibles que no tienen diseño electrónico en su interior al usuario. Este procederá programar a través de estas piezas y, de ese modo, creará una trayectoria. Seguidamente, otro usuario (o el mismo) procederá a tomarle una fotografía desde la vista de planta con algún dispositivo móvil y la "cargará" en una aplicación móvil. Esta aplicación móvil enviará la foto a un servidor web, el cual se encargará de procesar la fotografía y obtener una trayectoria en base a esta. Después, comunicará la trayectoria al robot móvil y este ejecutará las instrucciones recibidas. Cabe resaltar que robot móvil contendrá en su interior una microcomputadora que levantará el servidor web y se encargará del control de su trayectoria.

3.3.3 Fijación de procesos técnicos

En base a la descripción previamente expuesta, se puede detallar la secuencia de operaciones que se desea que siga la Plataforma de Programación Tangible:

- **Etapa previa**

    1. Verificar el encendido del robot móvil
    2. Dar instrucciones de cómo deben ser utilizadas las piezas tangibles
    3. Explicar la sintaxis del lenguaje de programación al usuario
    4. Programar una trayectoria a través de las piezas tangibles
    5. Verificar la correcta colocación de las piezas tangibles

- **Etapa de ejecución**

    1. Fotografiar piezas tangibles
    2. Generar interfaz de usuario
    3. Enviar fotografía
    4. Recibir fotografía
    5. Procesar imagen
    6. Identificar instrucciones
    7. Generar datos de trayectoria
    8. Realizar el control de trayectoria
    9. Censar variables de movimiento
    10. Generar movimiento
    11. Emitir señales visuales
    12. Emitir señales audibles

- **Etapa final**

    1. El robot vuelve a la fase de espera de alguna instrucción

3.3.4 Aplicación de los sistemas técnicos y sus limitaciones

Los procesos técnicos mencionados anteriormente pueden ser ejecutados por el hombre o por un sistema técnico y, por esto es necesarios clasificarlos para poder hallar las limitaciones y los límites de la capacidad humana, ver Tabla 5.

Tabla 5
Clasificación y limitaciones de los procesos técnicos

N°	Proceso técnico	Clasificación	Limitaciones externas
F1	Verificar del encendido del robot móvil	Proceso Manual	Incapacidad humana
F2	Dar instrucciones de cómo deben ser utilizadas las piezas tangibles	Proceso manual	Incapacidad humana
F3	Explicar de la sintaxis del lenguaje de programación al usuario	Proceso Manual	Incapacidad humana
F4	Programar una trayectoria a través de las piezas tangibles	Proceso Manual	Incapacidad humana
F5	Verificar la correcta colocación de las piezas tangibles	Proceso Manual	Incapacidad humana
F6	Fotografiar piezas tangibles	Proceso Manual	Incapacidad humana Cámara del dispositivo móvil puede no ser óptima
F7	Generar interfaz de usuario	Proceso Automático	Ninguna
F8	Enviar fotografía	Proceso Manual	Puede fallar la comunicación
F9	Recibir fotografía	Proceso Automático	Puede fallar la comunicación
F10	Procesar imagen	Proceso Automático	La imagen puede no ser óptima
F11	Identificar instrucciones	Proceso Automático	Ninguna
F12	Generar datos de trayectoria	Proceso Automático	Ninguna
F13	Realizar el control de trayectoria	Proceso Automático	Superficie puede no ser óptima

F14	Censar variables de movimiento	Proceso Automático	Ninguna
F15	Generar movimiento	Proceso Automático	Ninguna
F16	Emitir señales visuales	Proceso Automático	Ninguna
F17	Emitir señales audibles	Proceso Automático	Ninguna

3.3.5 Agrupación de funciones

Es necesario relacionar los procesos técnicos para poder hallar la estructura de funciones óptima. Estos procesos técnicos pueden ser realizados uno tras otro o de manera simultánea, por lo que conocer el orden exacto en el que estos serán ejecutados es de suma importancia. En la Figura 24 y en la Figura 25 se pueden observar las agrupaciones de las funciones pertenecientes a los procesos manuales y los procesos automáticos.

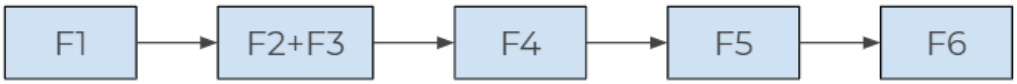


Figura 24. Agrupación de funciones para los procesos manuales

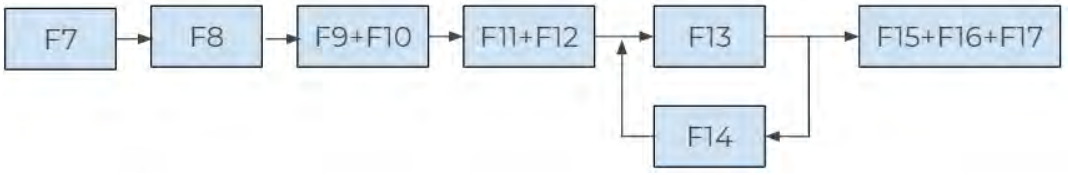


Figura 25. Agrupación de funciones para los procesos automáticos

3.3.6 Representación de la estructura de funciones

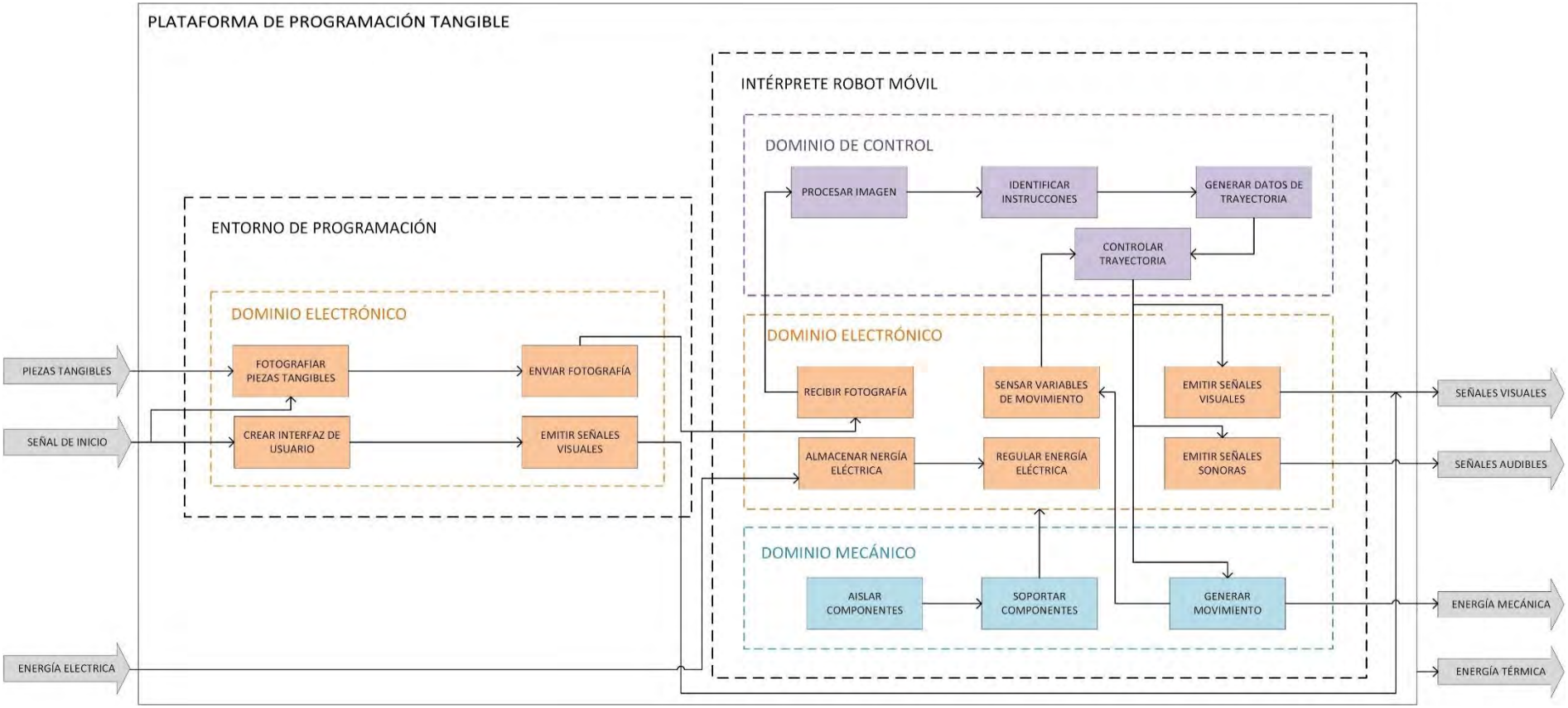


Figura 26. Estructura de funciones

La estructura de funciones presentada en la Figura 26 permite representar la interacción entre las entradas y salidas de las cajas negras del entorno de programación e intérprete planteadas en la Figura 22 y la Figura 23.

3.4 Concepto de solución

El concepto de solución permitirá hallar el diseño del proyecto definitivo, por lo cual, en base a la estructura de funciones descrita anteriormente, se puede plantear la matriz morfológica y, a partir de esta, describir alternativas de solución que combinen diferentes tecnologías y componentes que cumplan con los requerimientos y especificaciones. Finalmente, se evaluarán estas alternativas y se escogerá la más adecuada.

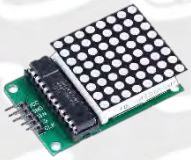
3.4.1 Matriz morfológica

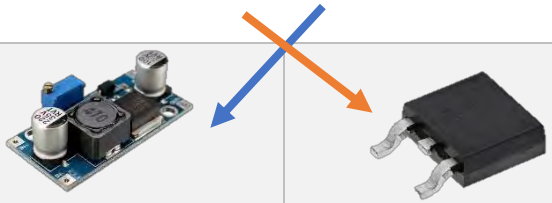
El entorno de programación abarcará las piezas tangibles, la fotografía de estos bloques (a través de un dispositivo móvil) y la interacción con el usuario, la cual se dará a través de una aplicación móvil.

Una matriz morfológica permitirá construir un sistema de soluciones para cumplir con el objetivo principal, haciendo uso de funciones parciales, la cual será desarrollada para el intérprete, ver Tabla 6. La primera columna representa las ya mencionadas funciones parciales y, sus respectivas filas describen las opciones (portadores de funciones) que permitirán que estas se realicen exitosamente.

Tabla 6
Matriz morfológica del intérprete móvil

N°	Función	Alternativa 1	Alternativa 2
1	Enviar/Recibir fotografía	Bluetooth	Wifi

3	Procesar imagen	 Microcomputadora	 Microcontrolador
6	Realizar el control de trayectoria	 Microcomputadora	 Microcontrolador
7	Censar variables de movimiento	 Girosocopio + Encoder	 Giroscopio
8	Generar movimiento	 Servomotor	 Motor DC
9	Emitir señales visuales	 Matriz de LEDs	 LCD 128x64
10	Emitir señales audibles	 Altavoz	 Buzzer
11	Almacenar energía eléctrica	 Baterías Li-Ion	 Baterías de Ni/MH

12	Regular energía eléctrica	 Step up	LDO

3.4.2 Descripción de las alternativas de solución

A partir de la matriz morfológica, se pudieron llegar a dos alternativas de solución (azul y naranja), las cuales serán descritas y evaluadas en los siguientes puntos para poder determinar cuál es la más óptima y de este modo, cumplir con el objetivo principal de la tesis.

3.4.2.1 Alternativa de solución azul

La alternativa de solución azul plantea piezas tangibles con diferentes combinaciones de colores (códigos de colores) para representar a cada bloque. El usuario será capaz de realizar un programa con estos bloques y, de esta manera, poder representar una trayectoria. Este programa se capturará en una imagen a través de una fotografía hecha con el dispositivo móvil y esta será enviada al intérprete a través una aplicación web utilizando la conexión inalámbrica Bluetooth.

El intérprete recibirá la imagen y realizará el procesamiento de la imagen a través de una microcomputadora, la cual también permitirá obtener una trayectoria. Estos datos serán procesados y permitirán realizar el control de trayectoria. Este robot, que como se mencionó en el Capítulo 1, es un robot con locomoción diferencial, hará uso de servomotores y tendrá morfología cuadrada sin distinción de género, basándose en la referencia [26]. Usará una matriz de LEDs y un *buzzer* para emitir señales visuales y audibles.

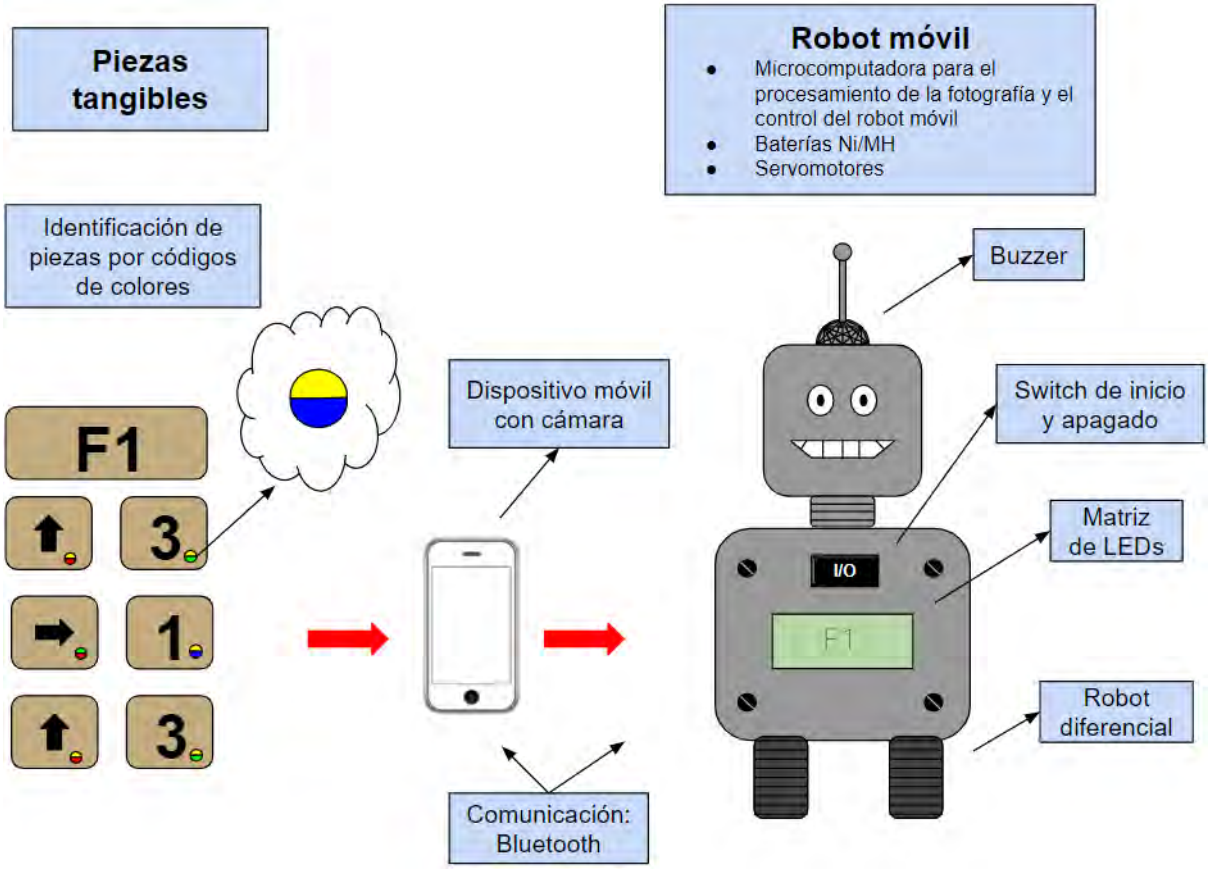


Figura 27. Boceto de la alternativa de solución azul

3.4.2.2 Alternativa de solución naranja

La alternativa de solución naranja plantea piezas tangibles con códigos QR en cada una de estas, con las cuales el usuario codificará un programa para representar una trayectoria. Se fotografiarán las piezas con un dispositivo móvil y, de este modo, poder obtener una imagen y enviarla al intérprete haciendo uso de la comunicación inalámbrica, la cual será Wifi.

El intérprete al recibir la imagen, la procesará a través de una microcomputadora, la cual enviará un archivo con esta información vía *bluetooh* a un microcontrolador. Este último será el encargado de realizar el control de trayectoria y de censar los movimientos del intérprete robot móvil con locomoción diferencial. Además, el robot tendrá morfología animal, basándose en la referencia [27] y podrá emitir señales sonoras y audibles a través de una pantalla LCD y un altavoz.

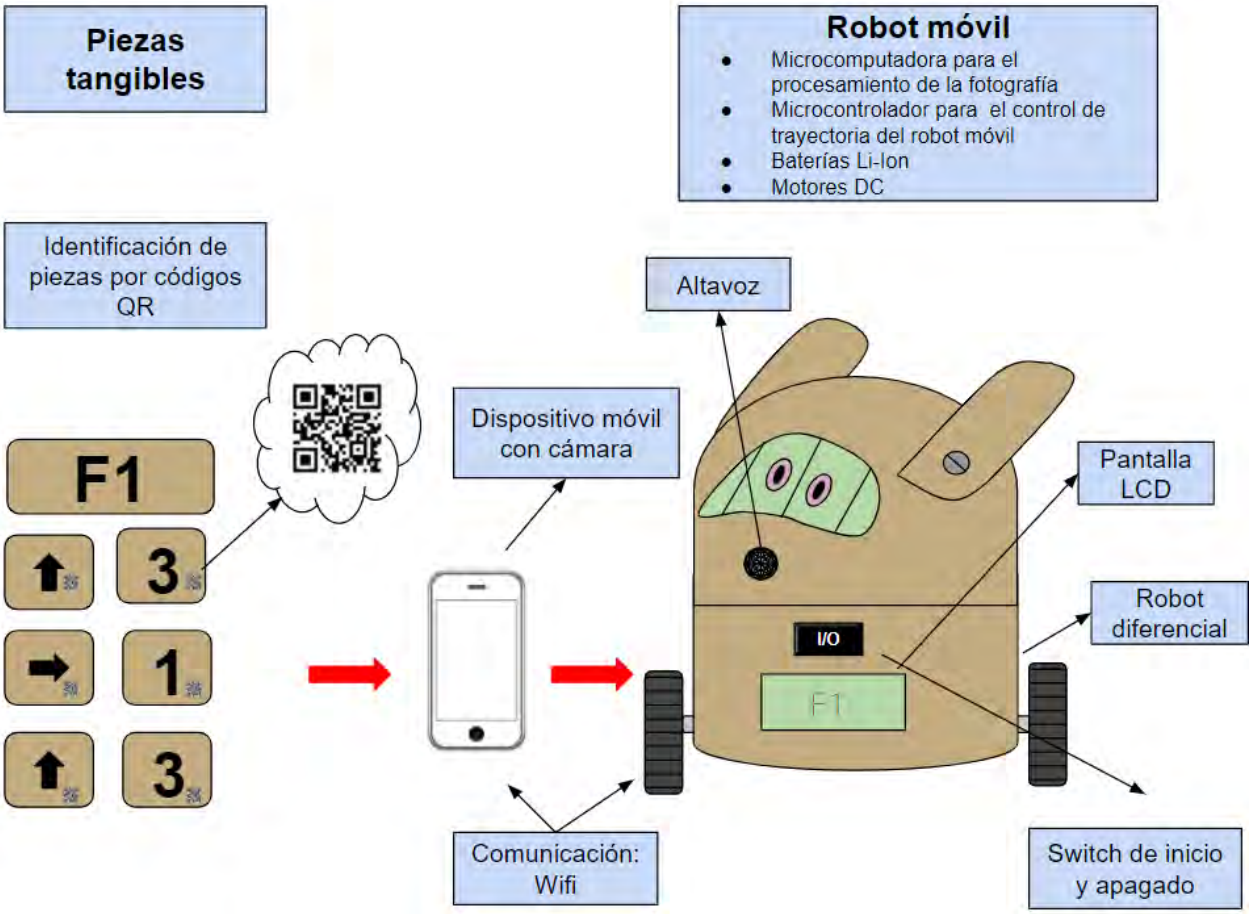


Figura 28. Boceto de la alternativa de solución naranja

Figura 28. Boceto de la alternativa de solución naranja

3.4.2.3 Evaluación técnico-económica de las alternativas de solución

En esta etapa, se deberán evaluar las alternativas naranja y azul, para lo cual se deben formular criterios que permitan una comparación útil y, de esta manera, obtener la variante óptima. Se puntuará cada criterio, donde 0 significa que no satisface, 1 significa que es aceptable, 2 significa que es suficiente, 3 significa bien y 4 significa muy bien.

Tabla 7
Evaluación técnico-económica de las alternativas naranja y azul

N°	Criterios técnicos y económicos	Alternativas de solución		
		Azul	Naranja	Óptima
1	Seguridad	2	3	4
2	Rapidez	2	3	4
3	Estabilidad	3	3	4
4	Facilidad de manejo	2	3	4
5	Confiabilidad	2	4	4
6	Lista de exigencias	3	4	4
7	Número de piezas	3	2	4
8	Fácil adquisición de materiales de fabricación	3	3	4
9	Costos diversos	3	2	4
10	Fácil montaje	3	2	4
	Suma Total	**26**	**31**	**40**

Una alternativa que satisfaga aceptablemente con la lista de exigencias y el objetivo de la tesis necesitaría un puntaje de 20 puntos. En la Tabla 7, se puede observar que las dos alternativas descritas a anteriormente sobrepasan este puntaje, pero la que resulta más óptima es la alternativa naranja.

3.5 Proyecto definitivo

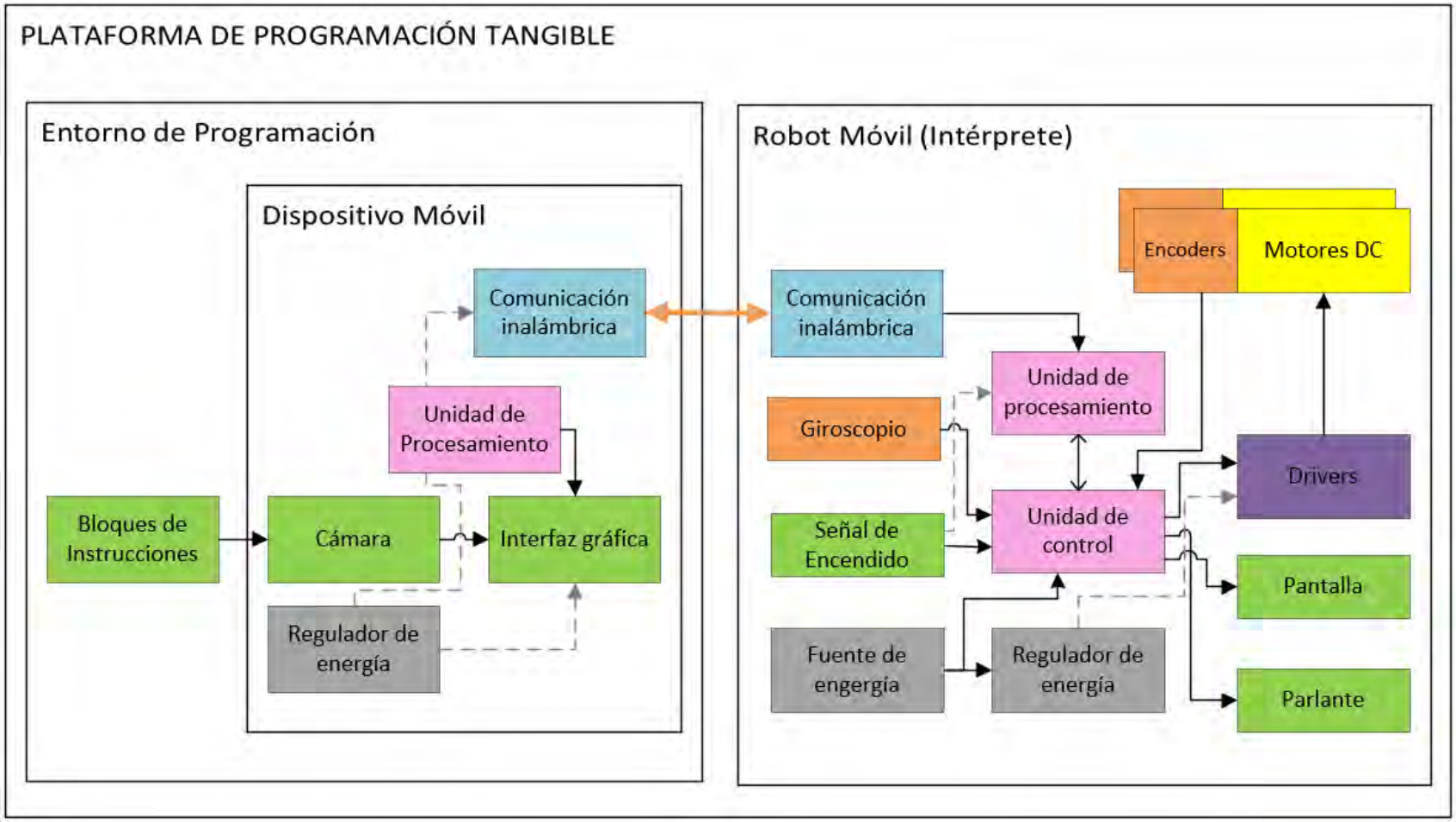


Figura 29. Modelo de solución

Tomando en cuenta el análisis de las alternativas a través de la matriz morfológica y la evaluación realizada a estas, se pudo obtener el diagrama de bloques del proyecto definitivo, ver Figura 29. Se puede notar que existen dos partes diferenciadas: el entorno de programación y el intérprete móvil que necesitan ser diseñadas.

3.5.1 Diseño del intérprete robot móvil

El diseño del intérprete robot móvil contempla tres partes principales: el diseño electrónico, el diseño mecánico, el diseño del algoritmo de control y el diseño de su respectivo software. El diseño de estas partes se hará por separado, pero deben guardar concordancia, ya que son dependientes entre sí.

3.5.1.1 Diseño electrónico del robot móvil

El diseño electrónico del robot móvil abarca la elección de componentes que cumplan con los requerimientos de la alternativa de solución Naranja y el diagrama esquemático.

3.5.1.1.1 Elección de componentes

Para realizar la elección de componentes, se analizan los requerimientos básicos de estos y las diferentes opciones existentes en el mercado actual.

- **Unidad de procesamiento**

En cuanto a la unidad de procesamiento, cuya función será el levantamiento del servidor web y el procesamiento de la fotografía enviada por el usuario, se realizará una comparación entre diferentes microcomputadoras que cumplan con esta función. Debido a las características y a su bajo precio, se optará por una Raspberry PI Zero W, además, esta también se encuentra en el mercado local (aunque con un precio más elevado), ver Tabla 8.

Tabla 8

Alternativas de microcomputadoras

		Alternativas		
Característica	**Requerimientos**	NanoPi NEO Air V1.1	Raspberry PI Zero W	Onion Omega2+
Procesador	-	Allwinner H3 Quad Core A7	Broadcom BCM2835	MT7688 SoC
Velocidad de reloj	>500 MHz	1.2 GHz	1 GHz	580 MHz
Memoria interna	>128 MB	512 MB	512 MB	128 MB
Comunicación inalámbrica	Bluetooth	Bluetooth 4.0 802.11n wireless LAN	Bluetooth 4.0 802.11n wireless LAN	802.11n wireless LAN
Dimensiones (mm)	-	40 x 40	65 x 30	42.9 X 26.4
Voltaje de alimentación	-	5 V	5 V	3.3 V
Disponibilidad	-	Importación	Importación Mercado local	Importación
Precio	-	$ 26.99 + envío	$ 10.00 + envío S/ 100.00	$ 13.00 + envío

Es necesario resaltar que, para realizar los cálculos en la elección de componentes, se usarán como datos las máximas medidas y los máximos pesos, aunque estos no sean usados en el diseño final, salvo que la precisión sea un requerimiento.

- **Motores DC con caja reductora**

El robot móvil, al tener locomoción diferencial, requiere controlar dos ruedas, por lo que serán necesarios dos motores idénticos para generar el movimiento de estas. Se deben identificar los criterios de diseño para la elección de los motores, los cuales serán el torque, la velocidad nominal y las dimensiones (para cumplir con los requerimientos de geometría).

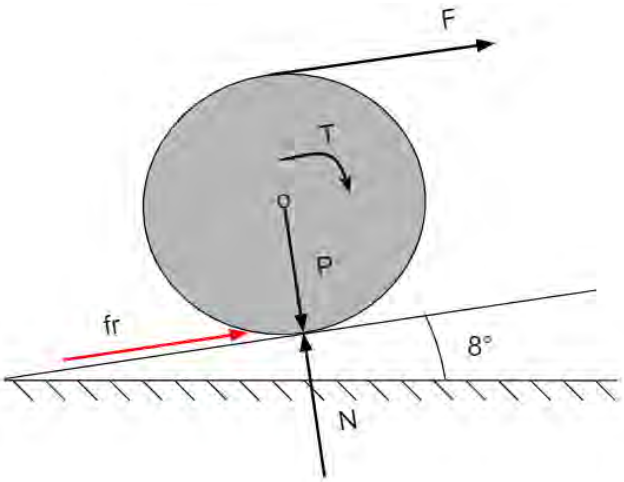


Figura 30. Diagrama dinámico de una rueda del robot móvil

En la Figura 30, se puede observar la representación de las fuerzas que interactúan en una de las ruedas del robot móvil. En la Tabla 2, se expuso como exigencia de ergonomía que la Plataforma de Programación Tangible no debe superar los 3 Kg. Por este motivo, el robot no deberá superar los 2.5 Kg y esta será la masa asumida para realizar cálculos. Además, se plantea una aceleración de 1 m/s^2 promedio para la cinemática del intérprete.

Es sabido que,

$$\sum F_x = ma$$

Donde,

m: Masa que recae sobre una de las ruedas (1.25 Kg)
a: Aceleración deseada (1 m/s^2)

Por lo que obtiene:

$$fr - P_x = ma \tag{16}$$

Además,

$$f_r = \frac{T}{R} \tag{17}$$

Donde,

R: Radio máximo de una de las ruedas (0.05 m)

Finalmente,

$$T_{o\,TOTAL} = \mathrm{n} \times R \times m \times (a + g \times sen(8°)) \quad (18)$$

Donde,

n: eficiencia asumida de 70%
g: fuerza de la gravedad (9.8 m/s^2)

De este modo, se puede obtener que $T = 0.10342\,Nm$ o $T = 1.0546\,Kg/cm$ el cual será el torque mínimo requerido.

La segunda característica a determinar es la velocidad de revolución de las ruedas del motor. La Tabla 2 indica que el tamaño máximo del intérprete móvil debe ser de 0.25 x 0.25 x 0.25 m y su velocidad de 1 longitud por 3 segundos, por lo que se tomarán estos datos como referencias. Además, al ser ruedas de 0.05 m de radio, se estima que recorrerán 0.314 m, ver Figura 31.

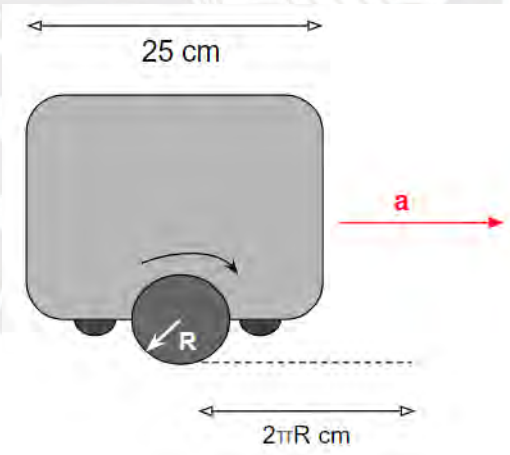


Figura 31. Representación de la máxima distancia que puede recorrer el robot móvil en 3 s

Finalmente, la fórmula 16 permite el cálculo de rpm:

$$RPM = \frac{l}{td} \times 60 \quad (19)$$

Donde,

l: Longitud máxima del robot móvil (0.25 m)
t: Tiempo que tarda en recorrer dicha longitud (3 s)
d: Distancia que recorren las ruedas al girar 1 vez (0.314 m)

De la Ecuación (19), se puede obtener que la velocidad mínima requerida para el motor será de 16 rpm, tomando un FS de 2, se necesitará un motor con una velocidad de 32 rpm.

En la Tabla 9, se pueden observar algunos motores que cumplen con los requerimientos previamente descritos, pero se escogerá el motor del modelo JGA25-370-78K por su precio y su disponibilidad. Además, este incorpora un *encoder* en cuadratura (efecto hall) que facilita la lectura y control de las revoluciones del motor.

Tabla 9
Alternativas de motorreductores

		Alternativas		
Característica	**Requerimiento**	JGA25-370-78K	RobotShop	ACP
Velocidad nominal	32 rpm	77 rpm	165 rpm	100 rpm
Torque	1.0546 Kg/cm	1.3 Kg/cm	6.5 Kg/cm	0.19 Kg/cm
Voltaje nominal	5 V	6 V	6 V	3 V
Corriente de trabajo	-	0.5 A	3 A	0.23 mA
Encoder	SÍ	SÍ	SÍ	SÍ
Dimensiones (mm)	-	31 x 24.4 x 24.4	24 x 50 x 8	12 x 10 x 15.1
Disponibilidad	-	Exportación	Exportación	Exportación
Precio	-	$ 9.66 + envío	19.72 € + envío	$ 10 + envío

- **Driver para motor**

La elección del driver del motor dependerá de la corriente necesaria para alimentar los motores de la Tabla 9. Se escogerá el driver DRV8871 porque cumple con este requerimiento y por su bajo precio.

Tabla 10
Alternativas de drivers para el control de motores

		Alternativas		
Característica	**Requerimiento**	RS PRO 417-9728	Pololu Dual MC33926	DRV8871
Voltaje de entrada	-	6—15 V	5—28 V	6.5—45 V
Corriente máxima	>0.5 A	3 A	3 A	3.6 A
Dimensiones (mm)	-	62 x 41 x 40	45.7 x 27.9 x 6.1	24 x 20 x 10
Disponibilidad	-	Importación	Importación	Importación
Precio (2 motores)	-	2 x 20.13 €	$ 29.95 + envío	2 x $ 7.50 + envío

- **Sensor de variables de movimiento**

En cuanto el sensor de variables de movimiento se usará el MPU9250 por la cantidad de grados de libertad y sus opciones variadas para la comunicación con la unidad de control, ver Tabla 11.

Tabla 11
Alternativas de sensores de variables de movimiento (Parte A)

		Alternativas	
Característica	**Requerimiento**	MPU9250	MPU6050
Voltaje nominal	-	3 V	3 V
Corriente máxima	-	3.4 mA	3.6 mA
Comunicación	SPI	I^2C SPI	I^2C

Tabla 11
Alternativas de sensores de variables de movimiento (Parte B)

		Alternativas	
Característica	**Requerimiento**	MPU9250	MPU6050
Resolución	±250°/S, ±500°/S, ±1000°/S, ±2000°/S	±250°/S, ±500°/S, ±1000°/S, ±2000°/S	±250°/S ±500°/S, ±1000°/S, ±2000°/S
Grados de libertad	9	9	6
Dimensiones (mm)	-	20 x 16 x 3	20 x 16 x 3
Disponibilidad	-	Mercado local	Mercado local
Precio	-	S/. 40.00	S/. 20.00

- **Pantalla gráfica**

Las pantallas gráficas mostradas en la Tabla 12 tienen similitudes en cuanto las especificaciones técnicas (comunicación, resolución y voltaje nominal), pero se escogerá el modelo ST920 por disponibilidad y precio.

Tabla 12
Alternativas de pantallas gráficas

Tabla 12 Alternativas de drivers para la pantalla gráfica

		Alternativas		
Característica	**Requerimiento**	CFAG128128A-TMI-TZ	ST7920	ERM12864-7
Voltaje de alimentación	-	5 V	5 V	5 V
Comunicación	-	Paralela	Paralela SPI	Paralela
Resolución	≥128 x 64 pixeles	128 x 128 pixeles	128 x 64 pixeles	128 x 64 pixeles
Dimensiones (mm)	-	86 x 100 x 14.5	93 x 70 x 13.5	70 x 76 x 12.5
Disponibilidad	-	Importación	Local	Importación
Precio	-	$ 47.73 + envío	S/ 40.00	$ 7.23 + envío

- **Parlante**

La selección del parlante también se basará en la disponibilidad y precio, por lo que se escogerá la opción de Naylamp, ver Tabla 13. Además, la intensidad de sonido de dicha opción será la siguiente:

$$I = \frac{P}{4\pi d^2} \tag{20}$$

Con una atenuación por distancia definida por:

$$A = 20 \log (d) \tag{21}$$

Donde,

P: Potencia en watts (0.5 w)
d: Distancia (10 m)

La intensidad de sonido resulta $3.98 \times 10^{-4}\ \mathrm{W/m^2}$ o 86 dB y la atenuación por distancia de 20 dB, por lo que la intensidad de sonido real será de aproximadamente 66 dB y será necesario utilizar un amplificador (transistor BJT).

Tabla 13
Alternativas de altavoz

		Alternativas	
Característica	**Requerimiento**	Naylamp Sen-Sound	Eiechip
Rango de frecuencias	> 20 Hz	0.6—10 KHz	No especifica
Potencia	≥ 0.5 W	0.5 W	1 W
Impedancia	-	8 Ω	8 Ω
Dimensiones	-	28 mm de diámetro	50 mm de diámetro
Disponibilidad	-	Mercado local	Exportación
Precio	-	S/ 6.00	$ 2.46 + envío

- **Unidad de control**

La cantidad de periféricos que estarán conectados a la unidad de control son los ya antes mencionados. Para organizar esta información, se realizará la Tabla 14, en la cual se especificarán la cantidad de pines necesarios por cada periférico.

Tabla 14
Periféricos

Periférico	N° de entradas y salidas	Entradas y salidas que requieren un N° de canales PWM
Driver de motor derecho	-	2
Driver de motor izquierdo	-	2
Enconder de motor derecho	-	2
Enconder de motor izquierdo	-	2
Pantalla gráfica	8	-
Sensor de variables de movimiento	2	-
Altavoz	1	-
Modulo bluetooth	2	-
Total	13	8

Se necesitan 13 pines disponibles para entradas y salidas digitales y analógicas, además de 8 pines disponibles para PWM, este es un dato necesario para la elección de la unidad de control. Además, se debe tener en cuenta que se necesitan pines disponibles en el caso de que surjan inconvenientes. Bajo las consideraciones de la Tabla 15, se optará por un Arduino Mega 2560 REV 3. Las especificaciones técnicas de este dispositivo cumplen con los requerimientos de diseño y el precio es el más bajo.

Tabla 15

Alternativas de unidades de control

		Alternativas		
Característica	**Requerimiento**	TIVA	Arduino Mega 2560 REV 3	EFM32WG
Procesador	-	TM4C123GH6PM	ATmega2560	Cortex M4
Velocidad de reloj	> 10 MHz	80 MHz	16 MHz	48 MHz
Memoria Interna	-	256 KB	256 KB	256 KB
Comunicación inalámbrica	Bluetooth	NO	NO	NO
Entradas analógicas	-	16	16	16
Entrada y salidas digitales	> 12	28	54	93
Voltaje de alimentación	-	4.75—5.25 V	7—9 V	1.98—3.8 V
Dimensiones (mm)	-	50 x 57.15	101.52 x 53.3	No especifica
Disponibilidad	-	Importación	Mercado local	Importación

- **Módulo bluetooth**

La unidad de control necesitará comunicarse con la unidad de procesamiento, por lo que se optará por una conexión inalámbrica, Bluetooth. Se optará por el módulo HC-05, ya que presenta la versatilidad de ser usado como maestro y esclavo, ver Tabla 16.

Tabla 16

Alternativas del módulo bluetooth (Parte A)

		Alternativas	
Característica	**Requerimientos**	HC-05	HC-06
Voltaje de operación	-	3.6—6 V	No especifica

Tabla 16

Alternativas del módulo bluetooth (Parte B)

		Alternativas	
Característica	**Requerimientos**	HC-05	HC-06
Comunicación	UART, SPI	UART	UART
Velocidad de transmisión	> 500 bps	1200 bps—1.3 Mbps	1200 bps—1.3 Mbps
Modo de uso	Maestro y esclavo	Maestro y esclavo	Esclavo
Dimensiones (mm)	-	27 x 12.7	27 x 12.7
Disponibilidad	-	Mercado local	Mercado local
Precio	-	S/ 30.00	S/ 25.00

- **Fuente de voltaje**

La elección de la fuente de voltaje dependerá del cálculo aproximado de consumo de corriente de cada componente, el cual está especificado en cada hoja de datos. La Figura 32 muestra el consumo de corriente total aproximado.

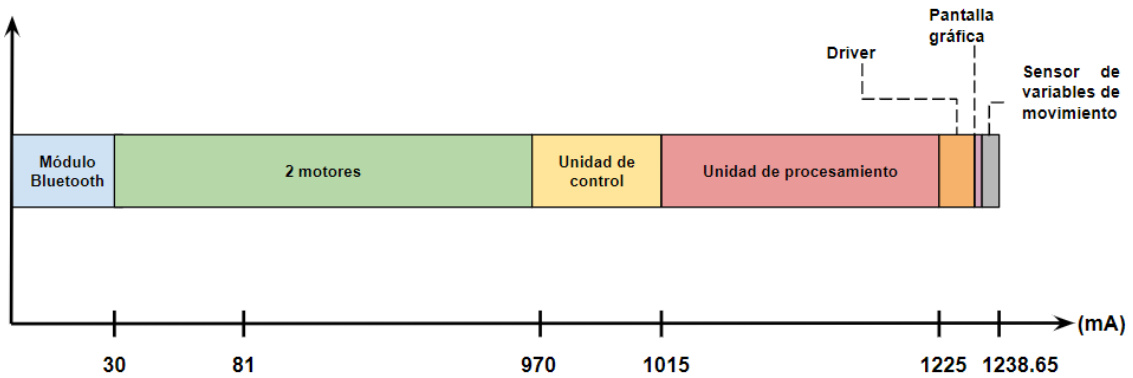


Figura 32. Consumo aproximado de corriente

La Tabla 17 muestra las principales características de la batería elegida por ser de Li-Ion y estar dentro de los costes previstos.

Tabla 17

Características de las baterías

Característica	Samsung INR18650-26J M
Voltaje	3.6 V
Capacidad	2600 mAh
Peso	45.5 g
Dimensiones	18.33 x 64.85 mm
Disponibilidad	Mercado local
Precio	S/ 22.00

Para poder calcular la autonomía de las baterías, se usa la siguiente ecuación:

$$\frac{W_B}{W_C} = \frac{I_B \times V_B}{I_C \times V_B} = H \tag{21}$$

Donde,

W_B: Potencia de la batería
W_C: Potencia consumida
V_B: Voltaje de la batería
I_B: Intensidad de la batería
I_C: Intensidad consumida

Aplicando la ecuación 21 para un juego de 2 baterías en serie (7.2 V), la potencia W_B es de 14976 mW, considerando una eficiencia de 80% y la potencia W_C es de 8918.28 mW. Esto implica que dos baterías en serie durarían un aproximado de 1.7 horas, por lo que se usarán 2 juegos de baterías en serie agrupadas en paralelo.

Si se vuelve a realizar el cálculo para estas 4 baterías, la duración de las baterías en el robot es de, aproximadamente, 3.4 horas (mayor a 3 h) lo cual cumple con la lista de exigencias de la Tabla 2.

- **Regulador de voltaje**

El driver DRV8871 necesita 6.5 V como mínimo y, debido a que a se usarán 2 baterías (6-7.2 V), se fijará 7 V como tensión de trabajo. El regulador de voltaje AP7381 permite regular voltajes entre 3.3-40 V y fijarlos en 7 V.

3.5.1.1.2 Diagrama esquemático del robot móvil

El diagrama esquemático se realizó con el *software* Eagle y, se basa en todos los componentes seleccionados después de haber realizado el análisis previamente visto, ver Anexo 2. Además, también se muestra el diagrama de conexiones junto al diseño del circuito impreso en el Anexo 3, cuyo modelo 3D se observa en la Figura 34.

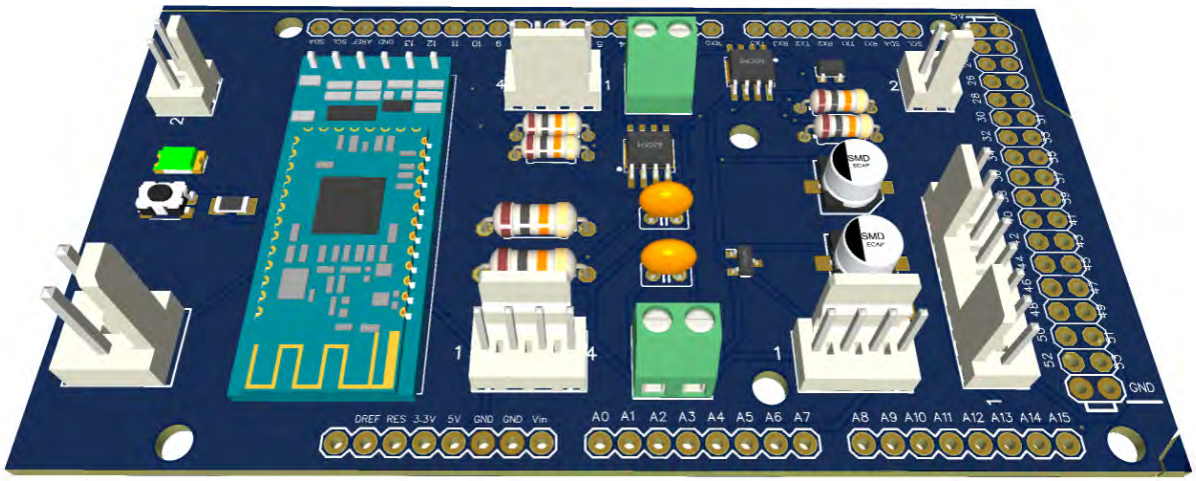


Figura 33. Modelo 3D del PCB del robot móvil

3.5.1.2 Diseño mecánico del robot móvil

Se realizó el modelado del chasis del intérprete robot móvil en base al boceto de la alternativa de solución electa (naranja). Además, respetando los requerimientos de seguridad, el material más adecuado para la impresión 3D del chasis deberá ser PLA. También se cumplen con los requerimientos de geometría con respecto

a las dimensiones máximas y los bordes no filosos. El plano del chasis del intérprete se encuentra en el Anexo 4.

Figura 34. Modelado 3D del chasis del intérprete robot móvil

Figura 34. Modelado 3D del chasis del intérprete robot móvil

3.5.1.3 Diseño del algoritmo de control del robot móvil

La estrategia de control, como ya se explicó en el Marco teórico, estará basada en el modelo cinemático del robot diferencial [36], para lo cual se seguirá el esquema planteado en la Figura 13. En primer lugar, se realizó la representación del modelo cinemático del mismo en la herramienta Simulink del *software* Matlab, ver Figura 35.

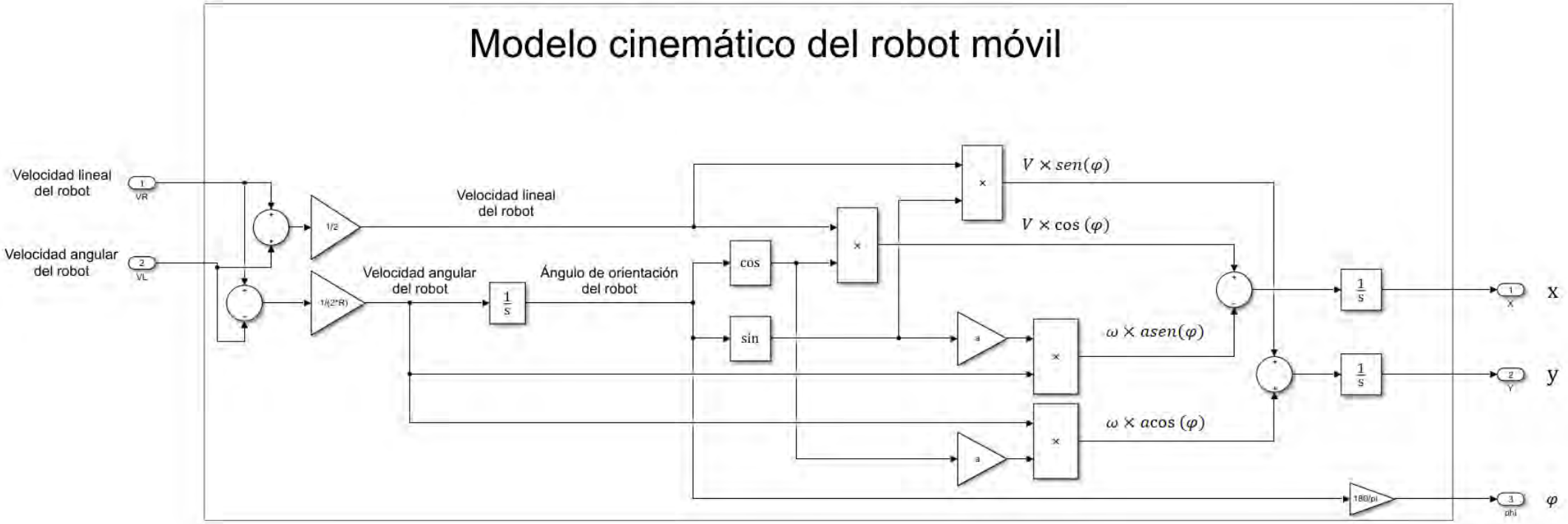


Figura 35. Diagrama de bloques del robot móvil realizado con el uso de Simulink

En la Figura 36 (a), se muestra el ensayo de un motor en grados sexagesimales con características muy similares a las del motor electo cuando se le aplica una entrada escalón de 2.25 V.

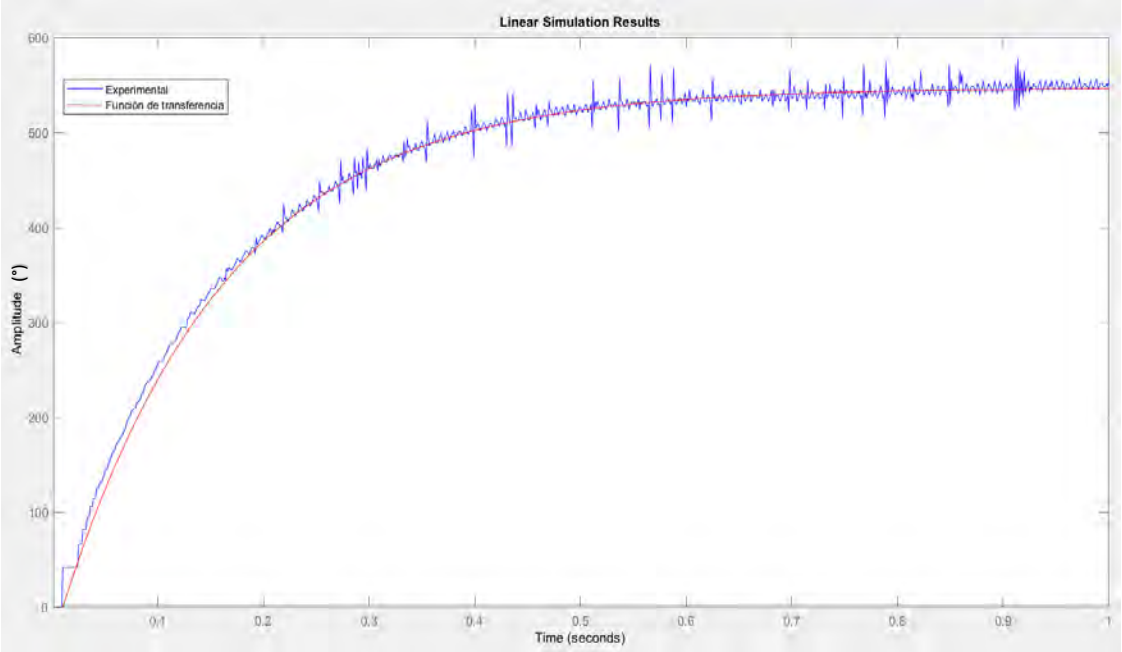


Figura 36. Modelo matemático de los motores derecho e izquierdo

La función de transferencia del motor DC se puede aproximar a una función de transferencia de primer orden, representada en la ecuación 22, donde τ es 0.156.

$$G_m(s) = \frac{1559}{s + 6.413} \tag{22}$$

Siguiendo con el esquema de la Figura 13, es necesario diseñar dos controladores de velocidad para los motores derecho e izquierdo. Como la función de transferencia de los motores DC es de primer grado y sin un polo en cero que garantice un error en estado estable igual a 0, se diseñará un controlador proporcional integral (PI) usando el método de lugar geométrico de raíces (LGR) que sigue el esquema de la Figura 37, donde b es la distancia entre las ruedas (0.089 m) y R es el radio de las ruedas (0.0475 m).

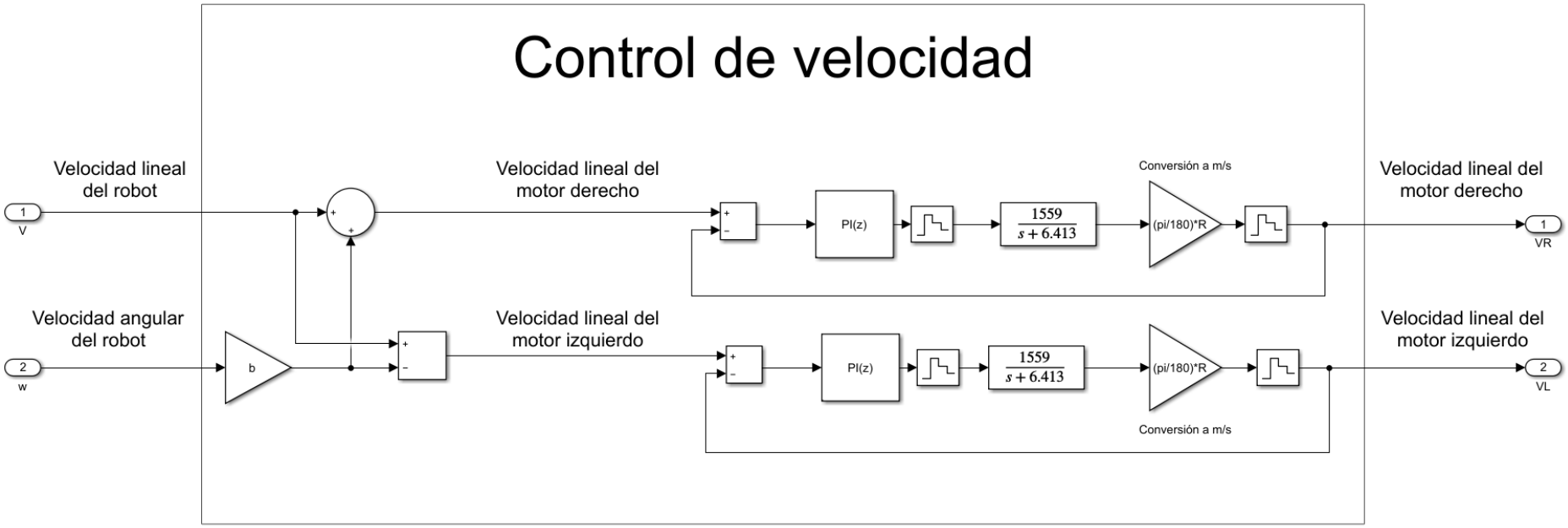


Figura 37. Controlador PI de velocidad discreto de los motores

Como primer paso para el diseño del controlador de velocidad, se escogió un tiempo de establecimiento (t_s) igual a 1 s y un máximo sobreimpulso (M_p) de 0 %, por lo que, de acuerdo a las ecuaciones 23, 24 y 25, el polo deseado es s = -4.

$$\sigma = \frac{4}{t_s} \tag{23}$$

$$\omega_d = \frac{-\pi\sigma}{\ln(M_p)} \tag{24}$$

$$s_{1,2} = (-\sigma;\ \pm\omega_d) \tag{25}$$

Como segundo paso, se adecuó la fórmula de un controlador PI (ecuación 26) y se reemplazó K_i/K_p por la constate b = 6, de modo que el polo deseado s = -4 esté dentro del LGR de $G_{s\prime}$ en lazo abierto.

$$G_c(s) = K_p + \frac{K_i}{s} = \frac{K_p(s+K_i/K_p)}{s} = \frac{K_p(s+b)}{s} \tag{26}$$

$$G_s(s) = K_p \times \frac{1559(\frac{R\pi}{180})(s+6)}{s(s+6.413)} = K_p \times G_{s\prime} \tag{27}$$

Las adecuaciones previamente mencionadas se realizaron con el fin de que $G_{s\prime}$ se asemeje a un sistema que necesita un controlador proporcional y, además, se agrega la conversión de unidades (ver ecuación 27) que se observa en la Figura 37 para poder obtener las velocidades lineales a la salida del controlador, donde R es el radio de las ruedas (0.0475 m).

Como último paso, se graficó el LGR de $G_{s\prime}$ en lazo abierto (ver Figura 38) y se halló la ganancia (K_p) en el polo deseado (s = -4), resultando K_p = 3.74 y K_i=22.44, ver ecuación 28.

$$b = \frac{K_i}{K_p} \tag{28}$$

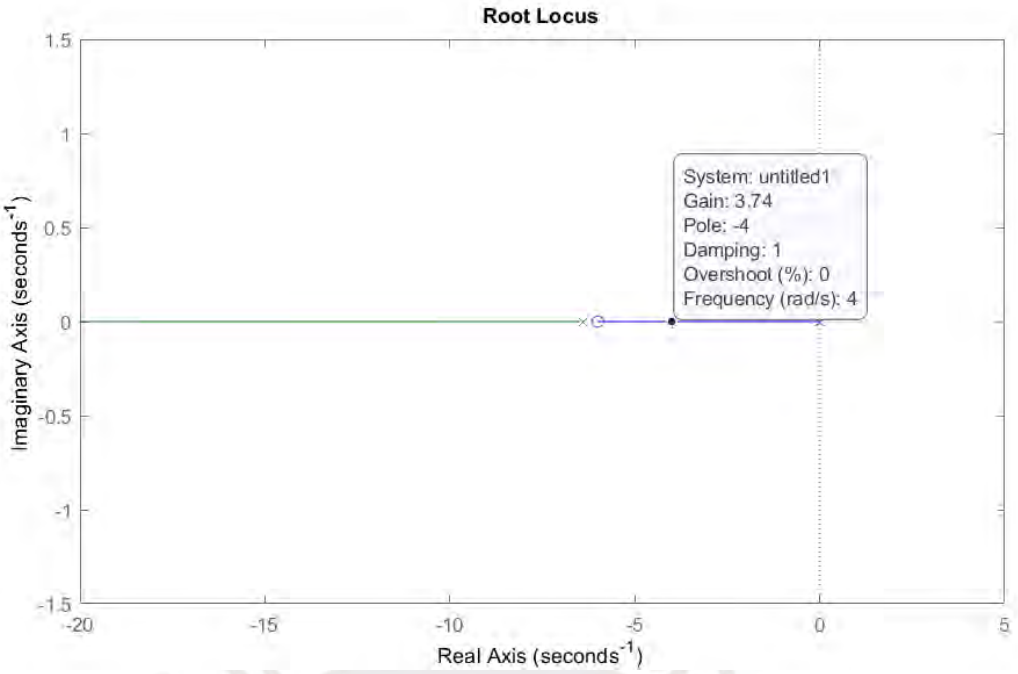
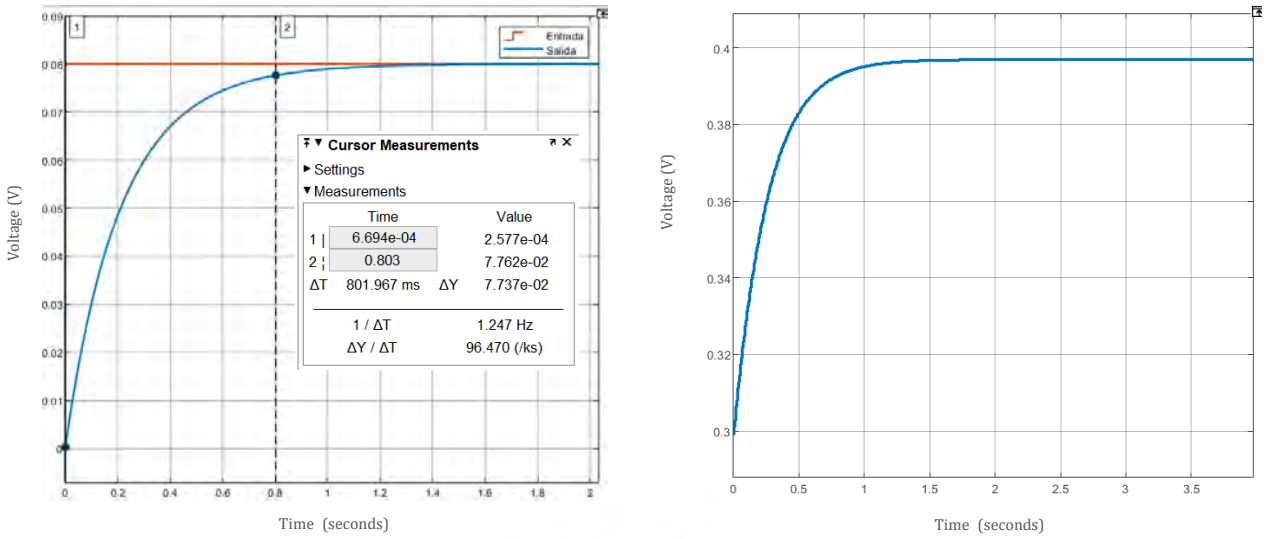


Figura 38. Lugar geométrico de raíces de $G_{s\prime}$ en lazo abierto

Finalmente, se realizó la discretización, ya que se necesita un controlador digital y se realizó el diagrama de simulación con la ayuda de la herramienta Simulink, ver Figura 38. El ensayo simula un sistema de control de velocidad lineal para una de las ruedas, el cual se realizó aplicando una entrada de 0.08 m/s (siendo esta la velocidad deseada para el intérprete) y se logró un error en estado estable (e_{ss}) igual a 0 y un tiempo de establecimiento (t_s) igual a 0.803 s, ver Figura 38 (a).

Además, se analizó el nivel de voltaje que provee la ley de control y si es que este puede ser correctamente manejado por los drivers de cada motor. Se graficó la curva de este voltaje con respecto a la simulación previa, resultando un máximo 0.3967 V, el cual es menor al voltaje con el que trabajan los drivers (7 V), ver Figura 38 (c). Es necesario que exista dicha diferencia, ya que el modelo cinemático no contempla posibles perturbaciones como la fricción de las ruedas y el peso del intérprete.



(a) (b)

Figura 39. Simulación de la respuesta del controlador de velocidad ante un escalón y de la ley de control

El controlador de trayectoria tendrá constantes $k_{1,2}$ que serán establecidas como 5, ya que, al reemplazar la ley de control en el modelo cinemático, estas constantes quedan dispuestas en la diagonal principal y representan directamente los polos del controlador (ver ecuación 15), por lo que su tiempo de establecimiento será de 0.8 s. El diagrama de simulación realizado con la herramienta Simulink se observa en la Figura 40 y se basa en la ecuación 11 del Marco Teórico.

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & sen(\varphi) \\ -\frac{sen(\varphi)}{a} & \frac{cos(\varphi)}{a} \end{bmatrix} \begin{bmatrix} K_1 x_e + \dot{x}_d \\ K_2 y_e + \dot{y}_d \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix} = \begin{bmatrix} -K_1 & 0 \\ 0 & -K_2 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \end{bmatrix} \tag{15}$$

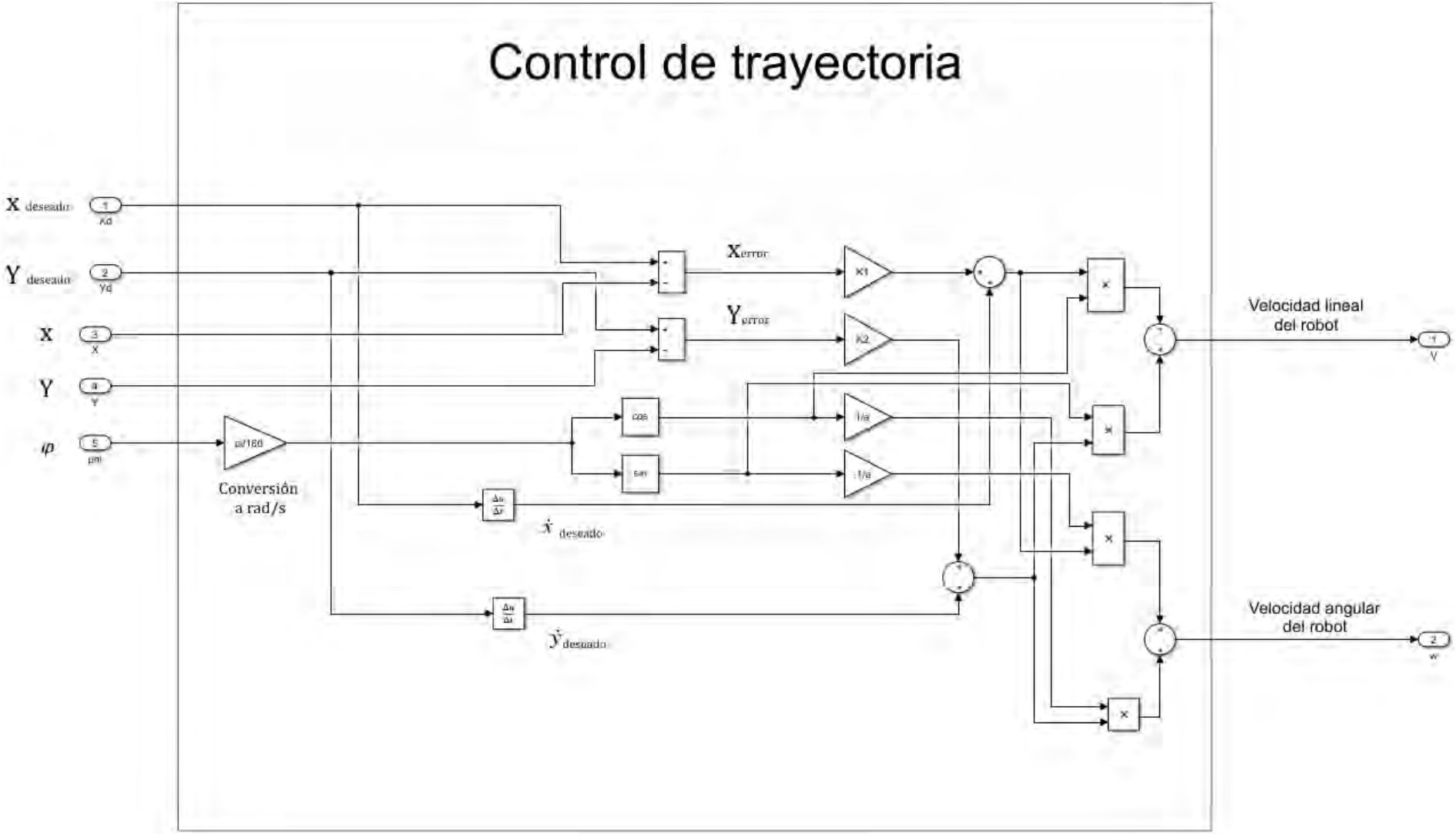


Figura 40. Controlador de trayectoria

El controlador completo integra al modelo cinemático del robot móvil, el bloque de control de velocidad, que abarca los controladores de velocidad para los motores izquierdo y derecho, y el bloque de control de trayectoria, ver Figura 41. Como se puede observar, las entradas son la posición deseada en el plano XY (X_d, Y_d), la posición actual en el plano XY (X, Y) y la orientación del robot móvil (φ).

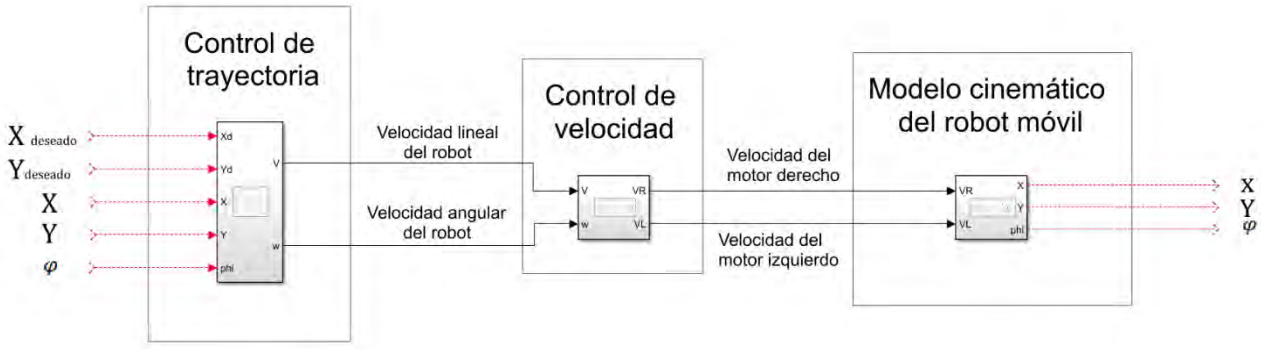


Figura 41. Sistema de control del intérprete robot móvil

Figura 41. Sistema de con robot móvil

3.5.1.4 Diseño del *software* del robot móvil

El diseño del software del intérprete robot móvil se muestra en la Figura 42 y se pueden encontrar dos partes diferenciadas: el de la microcomputadora, que se codificará en lenguaje Python, y el del microcontrolador, que se programará en su propio lenguaje basado en C++.

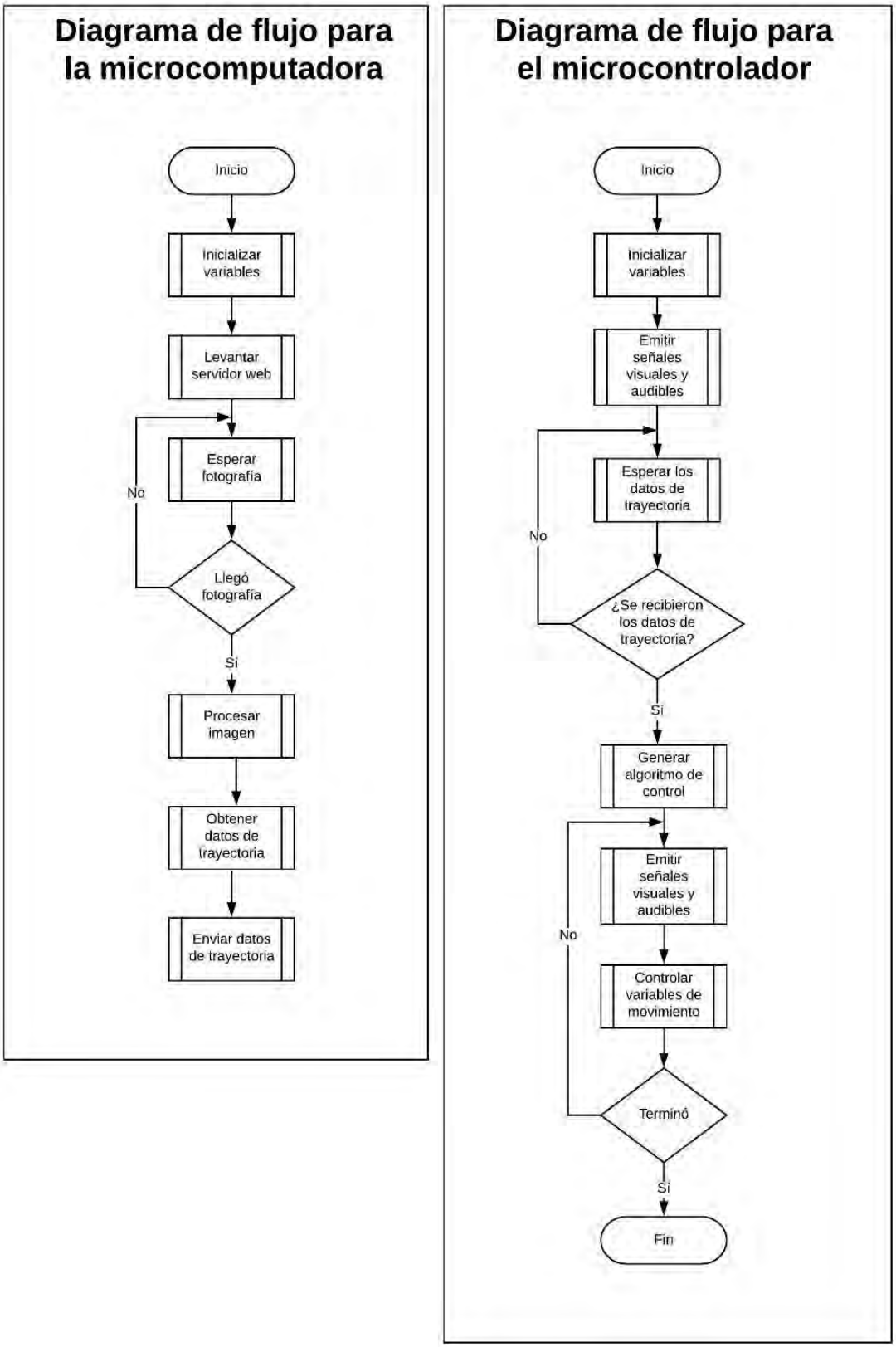


Figura 42. Diagramas de flujo para el intérprete

3.5.2 Diseño del entorno de programación

El diseño del entorno de programación abarca el diseño mecánico de las piezas tangibles y el diseño del software que permitirá el procesamiento de la fotografía tomada al código construido con las piezas y la interacción con el usuario.

3.5.2.1 Diseño mecánico del entorno de programación

La cantidad total de piezas tangibles es 15 (7 bloques de comandos y 8 bloques de parámetros) y estas se pueden representar a través de 5 diseños diferentes, ver Anexo 5. La unión de estas piezas se representa a través de la Figura 43.

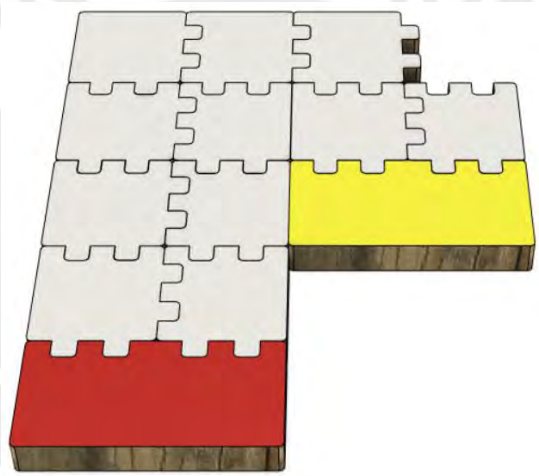


Figura 43. Representación de la unión de las piezas tangibles

3.5.2.2 Diseño del *software* del entorno de programación

El *software* del entorno de programación tiene como primera función la interacción con el usuario, por lo que será necesario el levantamiento de un servidor web a través de la microcomputadora (Raspberry Pi Zero W). Para poder realizar esta tarea, se simulará el entorno de dicha microcomputadora con la herramienta Oracle VM VirtualBox, en la que se integrará el sistema operativo Raspbian.

El servidor web se levantará a través del programa servidor *Apache Server* y el envío de la imagen se realiza a través de una aplicación web, programada en los lenguajes html y php, que tiene las siguientes funciones:

- Cargar un archivo desde el dispositivo móvil (celular) en el que se está usando dicha aplicación.
- Ejecutar el script que realiza el procesamiento de la imagen.

Cabe resaltar que la función del usuario es únicamente programar a través de las piezas tangibles, fotografiar este código y cargar la imagen a través de la aplicación web.

La segunda función del *software* del entorno de programación es el procesamiento de la fotografía subida al servidor web se realizará a través de un archivo en lenguaje Python que utilizará las librerías Open CV y Zbar. Esta última librería permite la decodificación de varios códigos QR en una misma imagen. La información de los códigos QR será ordenada a través de un algoritmo para poder crear la trayectoria que deberá seguir el intérprete robot móvil. El diagrama de flujo se puede observar en la Figura 44.

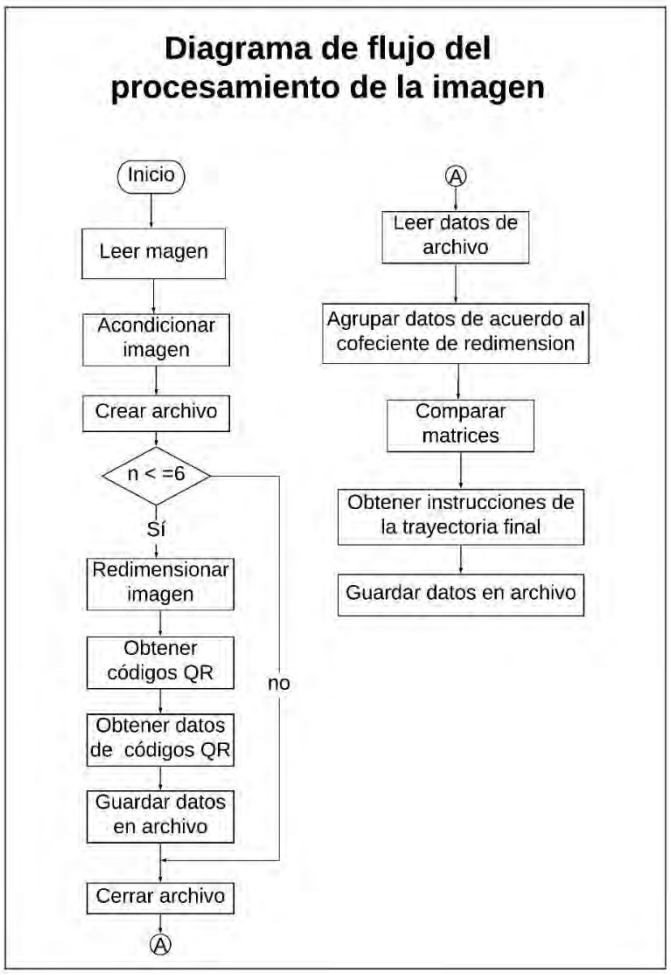


Figura 44. Diagrama de flujo del entorno de programación

Capítulo 4

Análisis y simulaciones

El Capítulo 4 documentará los resultados y simulaciones de los circuitos, algoritmos y softwares diseñados en el Capítulo 3 para poder validarlos y, de esta manera, poder medir el cumplimiento de los objetivos de la tesis.

4.1 Validación del entorno de programación

La validación del entorno de programación implica corroborar los diseños del *software* que realiza el procesamiento de la imagen y del *software* del servidor web, para comprobar que este sea correctamente levantado y la aplicación web sirva como medio de interacción con el usuario.

4.1.1 Validación del procesamiento de la imagen

Se procesaron imágenes de diferentes tamaños y calidades con el software diseñado y se observó que, para una redimensión de la imagen, el software reconocía ciertos códigos QR y otros no, pero para otra redimensión, reconocía los códigos QR que para la anterior redimensión no había reconocido, ver Figura 45.

Figura 45. Códigos obtenidos bajo diferentes redimensiones

La información obtenida a través de los códigos QR para cada una de las 6 redimensiones realizadas en la imagen, de acuerdo al diagrama de flujo de la Figura 44, fue correctamente ordenada y almacenada en matrices en base a su posición en la fotografía para que las instrucciones se ejecuten en el orden deseado por el usuario. Finalmente, se realizó una comparación entre estas matrices para obtener la información final y está fue guardada exitosamente en un archivo .txt como se muestra en la Figura 46 y y en la Figura 47.

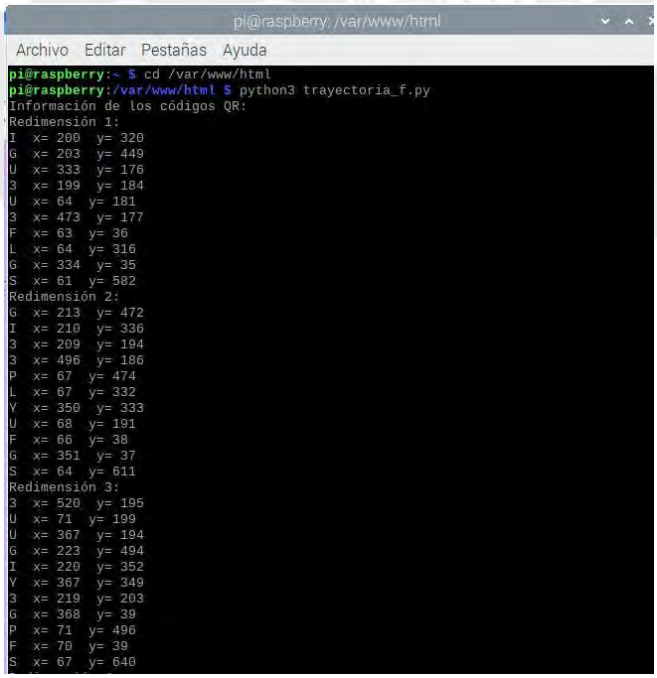


Figura 46. Información de los códigos obtenidos en cada redimensión e información final ordenada después de la comparación entre estas (Parte A)

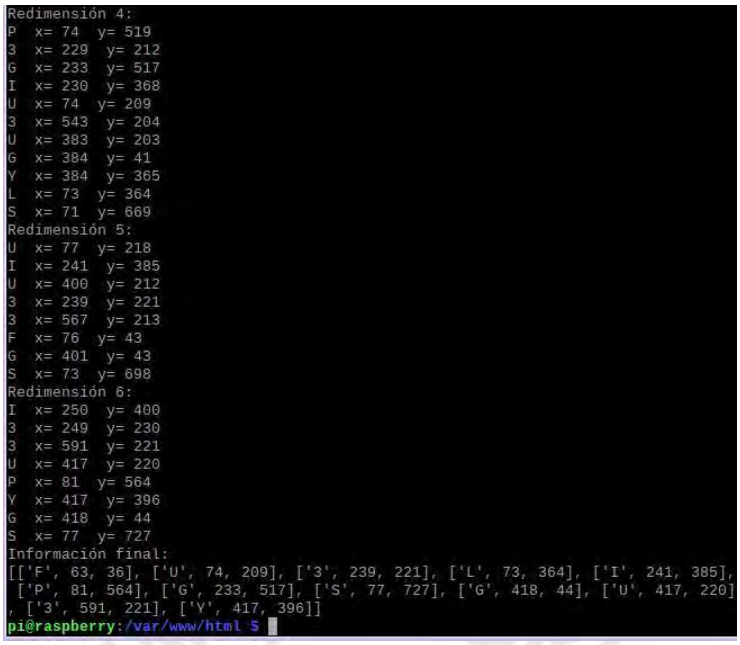


Figura 47. Información de los códigos obtenidos en cada redimensión e información final ordenada después de la comparación entre estas (Parte B)

Además, las simulaciones hechas con fotografías obtenidas desde distintos ángulos, posiciones y dispositivos móviles permiten fijar los requerimientos de la imagen a procesar, ver Tabla 18.

Tabla 18

Requerimientos de a imagen obtenida a través del dispositivo móvil

Característica	Requerimiento
Tamaño (pixeles)	3000 x 3600
Ángulo	Vista de planta
Tamaño del archivo	$<=$ 10 MB
Resolución (ppp)	$>=$ 72

Una vez validado el código que realiza el procesamiento de la imagen, se procedió a integrar este código a la aplicación web. La aplicación web tiene 2 únicas funciones, permitirle al usuario cargar una fotografía de sus archivos a través de un botón denominado “Choose file” y ejecutar el archivo con lenguaje Python que se encarga de

procesar la imagen a través de otro botón denominado “Subir archivo”, tomando como parámetro de entrada la fotografía subida por el usuario, ver Figura 48.

Figura 48. Aplicación web abierta desde el navegador Chromium

4.2 Validación del modelo teórico del intérprete robot móvil

Los diseños realizados para el software y algoritmo de control necesitan ser validados para comprobar su correcto funcionamiento. Se realizaron simulaciones y pruebas con cada uno de estos y los resultados se muestran en los siguientes puntos.

4.2.1 Validación del algoritmo de control de trayectoria

Se realizó la simulación del algoritmo de control de trayectorias, haciendo uso de la herramienta Simulink de Matlab. El vector “I” simula las instrucciones obtenidas del procesamiento de la imagen ingresada por el usuario que realiza la microcomputadora. Este vector es la entrada al bloque “Generador de trayectorias”, el cual contiene un algoritmo que genera tramos que el intérprete robot móvil deberá seguir en base a su orientación. Se graficaron las funciones matemáticas que representan cada uno de los posibles tramos codificados mediante los bloques tangibles, ya que al existir únicamente giros de $\pi/2$ rad las orientaciones censadas por el giroscopio serán de $n\pi/2$ rad siendo n un número entero, ver Tabla 19.

Tabla 19

Funciones matemáticas de los tramos (Parte A)

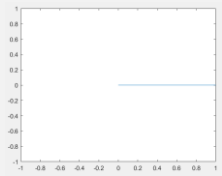
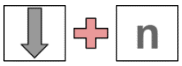
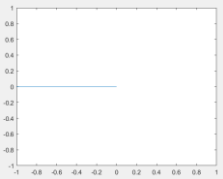
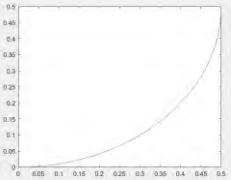
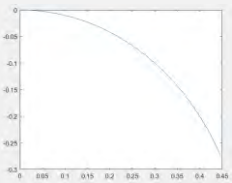
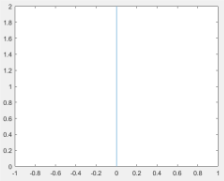
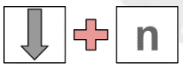
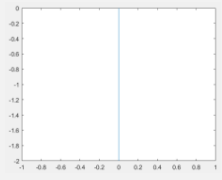
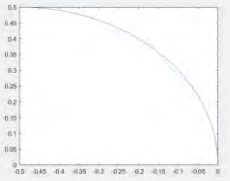
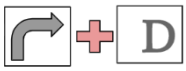
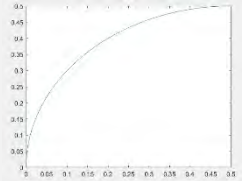
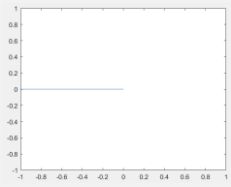
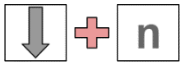
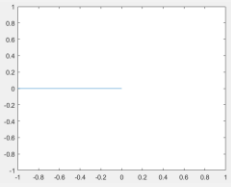
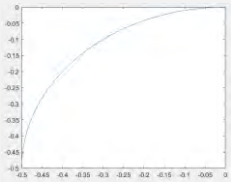
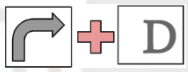
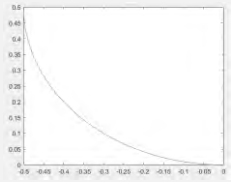
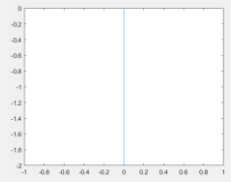
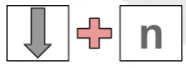
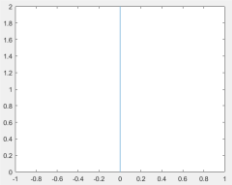
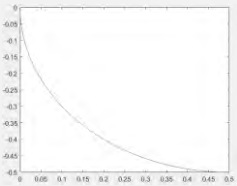
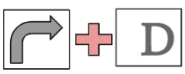
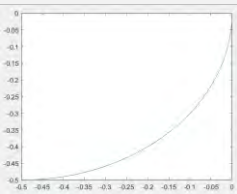
Orientación (rad)	Instrucción	Funciones matemáticas	Gráfica
0		$x = x_a + vt$ $y = y_a$	
		$x = x_a - vt$ $y = y_a$	
		$x = x_a + 0.5sen(vt)$ $y = -0.5cos(-vt) + 0.5$	
		$x = x_a + 0.5sen(vt)$ $y = 0.5cos(vt) - 0.5$	
$\pi/2$		$x = x_a$ $y = y_a + vt$	
		$x = x_a$ $y = y_a - vt$	
		$x = 0.5cos(vt) - 0.5$ $y = y_a + 0.5sen(vt)$	
		$x = -0.5cos(vt) + 0.5$ $y = y_a + 0.5sen(vt)$	

Tabla 20

Funciones matemáticas de los tramos (Parte B)

Orientación (rad)	Instrucción	Funciones matemáticas	Gráfica
$\boldsymbol{\pi}$		$x = x_a - vt$ $y = y_a$	
		$x = x_a + vt$ $y = y_a$	
		$x = x_a - 0.5sen(vt)$ $y = 0.5cos(vt) - 0.5$	
		$x = x_a + 0.5sen(-vt)$ $y = -0.5cos(vt) + 0.5$	
$\boldsymbol{3\pi/2}$		$x = x_a$ $y = y_a - vt$	
		$x = x_a$ $y = y_a + vt$	
		$x = -0.5cos(-vt) + 0.5$ $y = y_a - 0.5sen(vt)$	
		$x = 0.5cos(vt) - 0.5$ $x = y_a + 0.5sen(-vt)$	

.

El bloque “Generador de trayectorias” tiene como salida a la trayectoria deseada representada a través de funciones matemáticas. Finalmente, las entradas del sistema de control de trayectorias: X, Y y φ se realimentan, ya que son las variables medidas y las entradas X_d e Y_d son las salidas del “Generador de trayectorias”. Lo anteriormente especificado se puede observar en la el Diagrama de Simulación de la Figura 49. basado en el diseño del algoritmo de control realizado en el capítulo 2 (ver Figura 41). Además, se puede evidenciar que los bloques “Control de trayectoria” y “Control de Velocidad” se simularon a través de *Matlab functions* y de este modo, poder comprobar que el código ingresado en el microcontrolador funcione correctamente.

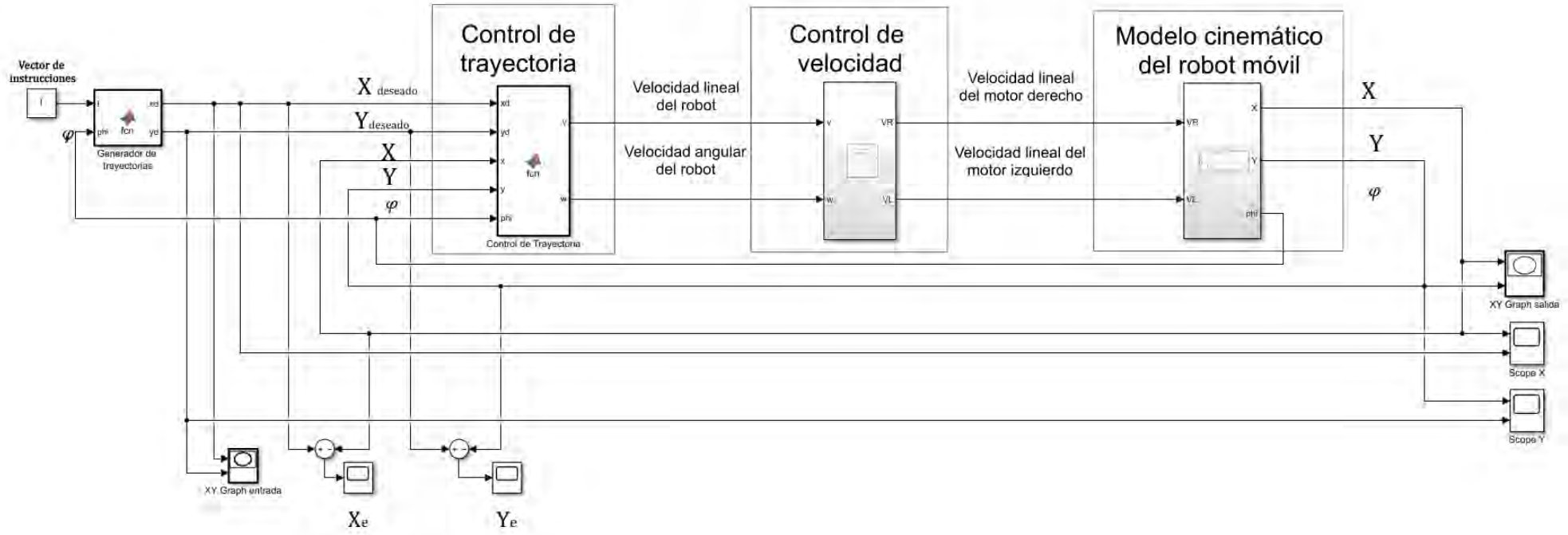


Figura 49. Diagrama de simulación del algoritmo de control de trayectorias

Como ejemplo para el análisis, se tomará una simulación que ejemplifica una trayectoria de 3 tramos, donde X_d e Y_d representan las posiciones en el plano XY deseadas. El primer tramo corresponde a la función avanzar 3 unidades de 0.25 m, el segundo tramo representa un giro a la izquierda que equivale a un cuarto de circunferencia y el tercer tramo representa avanzar nuevamente 3 unidades de 0.25 m. La Figura 4950 muestra a trayectoria deseada y la trayectoria obtenida por el controlador.

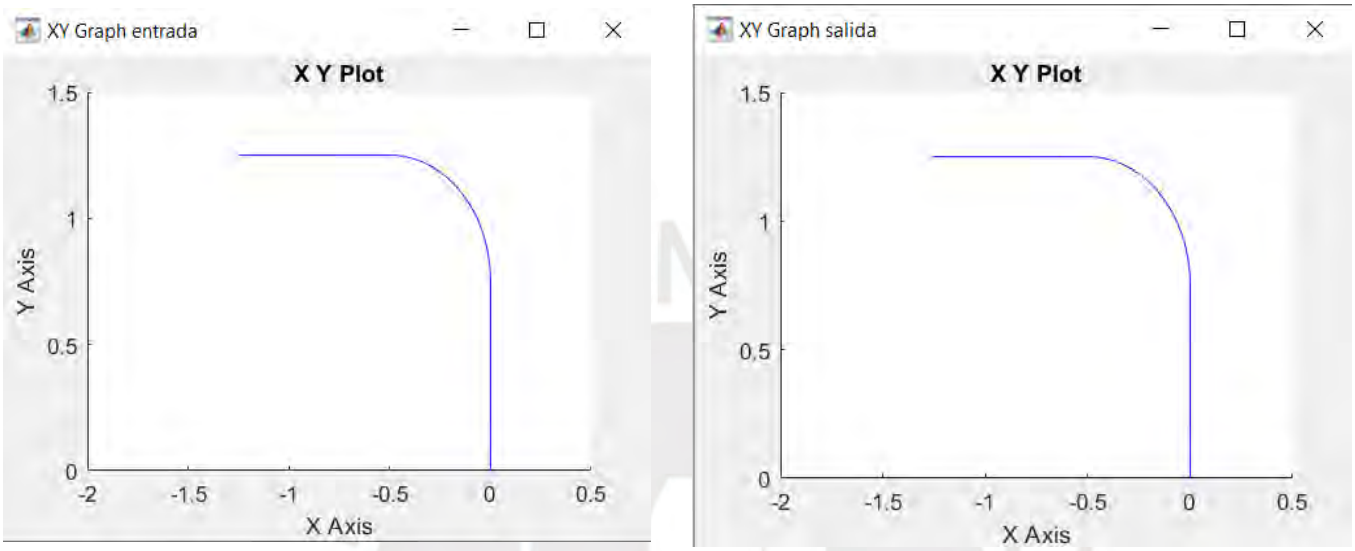


Figura 50. Gráficas de la entrada y salida del controlador

Asímismo, se muestran las gráficas de las variables X, X_d e Y, Y_d, donde se aprecia que no existe una diferencia significativa entre las posiciones deseadas y las ejecutadas por el controlador, ver Figura 51 y Figura 52.

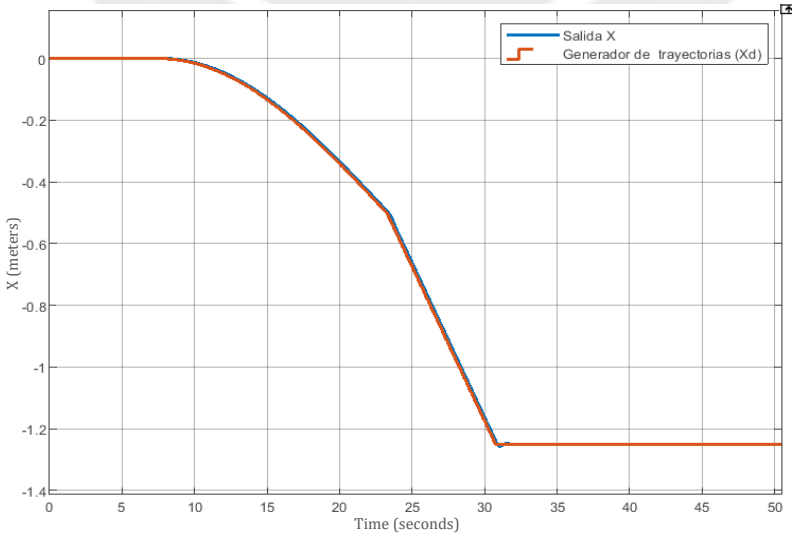


Figura 51. Gráficas de la variable censada X para la simulación

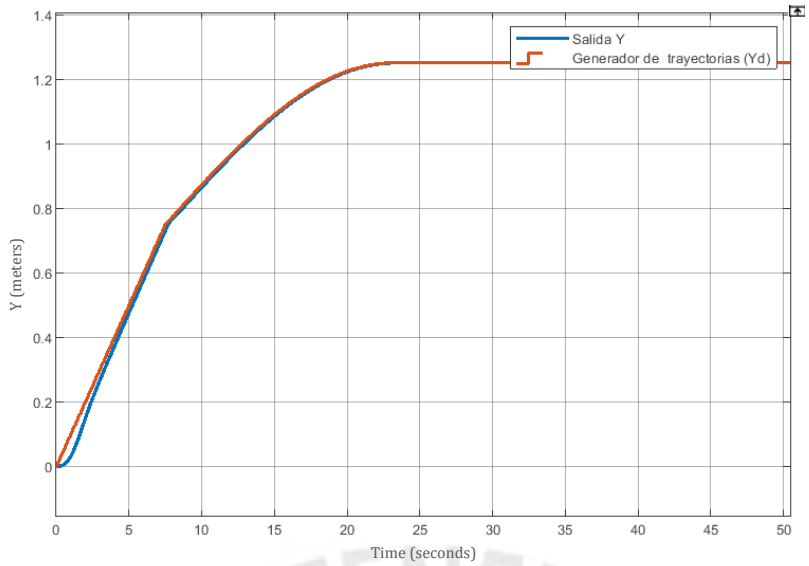


Figura 52. Gráficas de la variable censada Y para la simulación

Además, se muestran las gráficas de los errores, x_e y x_e, donde se observa que el error de la posición con respecto a X se encuentra en el rango, $[-16.56 \times 10^{3}, 6.313 \times 10^{-3}]$, y que el error de la posición con respecto a Y se encuentra en el rango $[-1.199 \times 10^{-14}, 7.079 \times 10^{-2}]$, ambos rangos muy cercanos a 0, ver Figura 52.

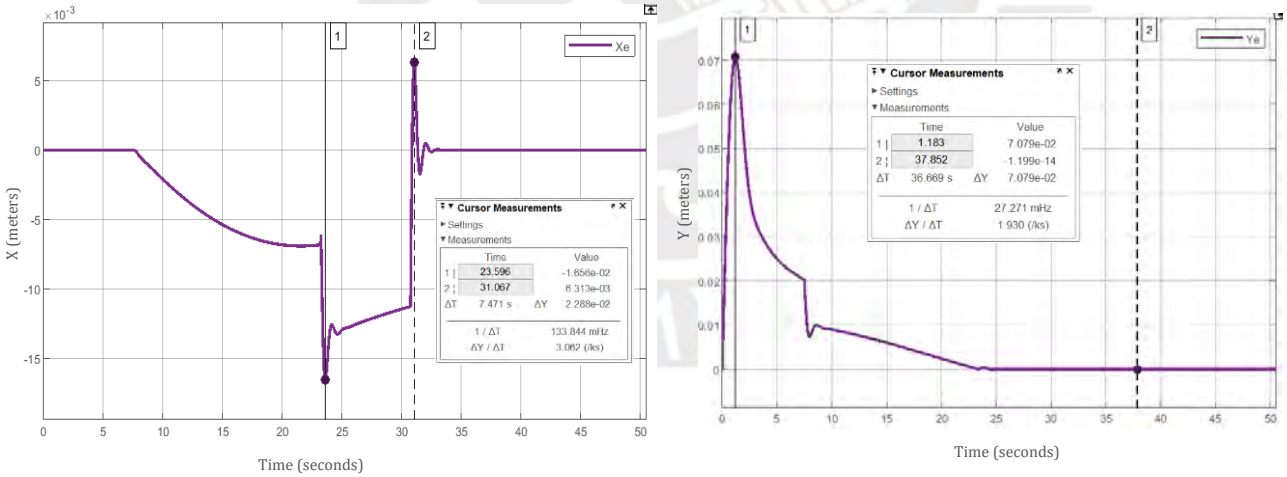


Figura 53. Gráficas de los errores para X e Y (X_e e Y_e)

Cabe mencionar que se tomaron en cuenta las ecuaciones del Marco Teórico:

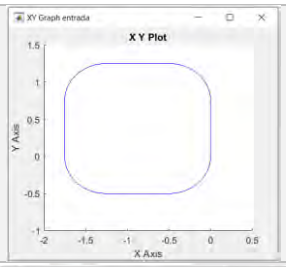
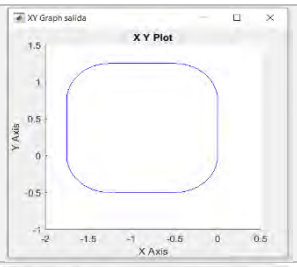
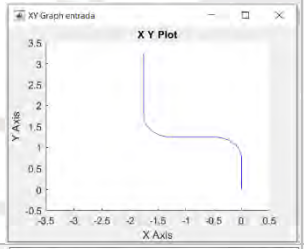
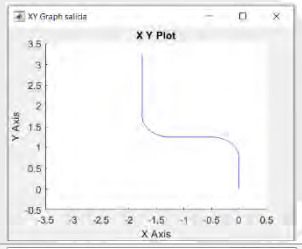
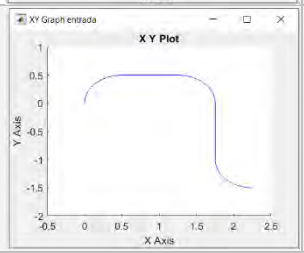
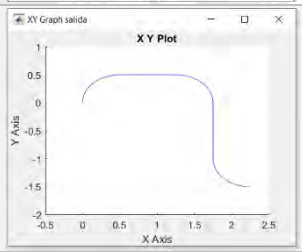
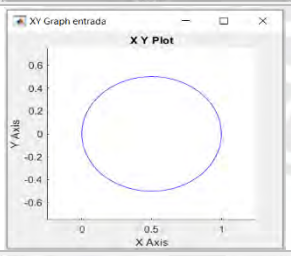
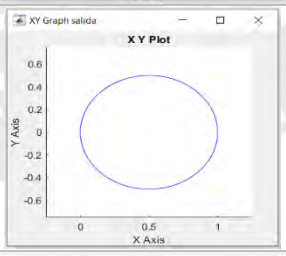
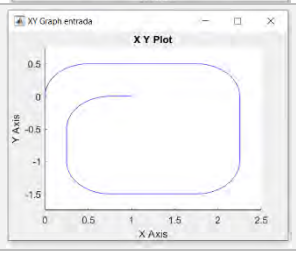
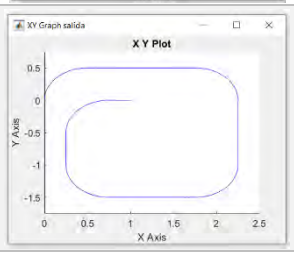
$$x_e = x_d - x \tag{12}$$

$$y_e = y_d - y \tag{13}$$

Finalmente, se muestra la Tabla 21, la cual contiene diferentes trayectorias simuladas y el rango de error entre cada una de estas trayectorias deseadas y las que fueron obtenidas por el sistema de control.

Tabla 21

Simulaciones de diferentes trayectorias

Vector de entrada	Trayectoria deseada	Trayectoria obtenida	Rango de error
I = [70, 85, 51, 76, 73, 85, 51, 76, 73, 85, 51, 76, 73, 85, 51, 76, 73, 83]			$X_e \in [-16.56 \times 10^{-3}, 11.14 \times 10^{-3}]$ $Y_e \in [-12.83 \times 10^{-3}, 7.079 \times 10^{-2}]$
I = [70, 85, 51, 76, 73, 85, 51, 82, 69, 85, 53, 83]			$X_e \in [-16.56 \times 10^{-3}, -4.97 \times 10^{-8}]$ $Y_e \in [-6.631 \times 10^{-3}, 7.079 \times 10^{-2}]$
I = [70, 82, 69, 85, 51, 82, 69, 85, 52, 76, 73, 83]			$X_e \in [-3.174 \times 10^{-3}, 2.243 \times 10^{-2}]$ $Y_e \in [-1.11 \times 10^{-14}, 4.005 \times 10^{-2}]$
I = [70, 82, 69, 82, 69, 82, 69, 82, 69, 83]			$X_e \in [-6.075 \times 10^{-3}, 10.82 \times 10^{-3}]$ $Y_e \in [-7.368 \times 10^{-3}, 4.005 \times 10^{-2}]$
I = [70, 82, 69, 85, 53, 82, 69, 85, 52, 82, 69, 85, 52, 82, 69, 85, 50, 82, 69, 85, 49, 83]			$X_e \in [-11.64 \times 10^{-3}, 22.53 \times 10^{-3}]$ $Y_e \in [-13.58 \times 10^{-3}, 4.005 \times 10^{-2}]$

4.3 Validación económica

En la Tabla 22, se puede observar el listado de todos los componentes necesarios para la construcción de la Plataforma de Programación Tangible electos en la etapa de diseño. El precio final es de 555.84 soles o 161.46 dólares americanos[1] sin considerar los costes de diseño, lo cual cumple con el requerimiento de costos, pues el precio es menor a 214 dólares americanos. Además, el peso total de todos los componentes no excede los 2.5 Kg planteados en la lista de requerimientos como peso máximo.

Tabla 22

Lista de precios previstos

Componente	Cantidad	Peso unitario (kg)	Precio unitario (S/)
Microcomputadora Raspebrry Pi Zero W	1	0.009	38.81
Batería recargable YSN-05480	1	0.195	80.27
Motor DC con caja reductora	2	0.090	33.88
Driver DRV8871	2	0.022	43
Giroscopio MPU9250	1	0.00272	20
Pantalla LCD ST7920	1	0.9	40
Altavoz	1	0.01	6
Módulo bluetooth HC-05	1	0.008	30
Regulador de voltaje	1	-	18
Arduino MEGA 2560 REV 3	1	0.037	58
Resistencias, diodos, etc.	-	-	20
Baterías INR18650-26J M	4	0.046	22
Caja porta-baterías	1	0.015	7
PLA	1	0.5	30
Interruptor de chasis	1	0.015	5
Tornillos con tuerca M3x10	-	0.01	5
Listones de madera			20
Total		**2.10972**	**599.84**

En la Tabla 23, se aprecian los costes de ingeniería calculados en el periodo de un año, que fue el tiempo utilizado para el desarrollo de la tesis. En necesario recalcar que estos costes son totalitarios (no por unidad), por lo que se requiere de una producción de por

[1] Tasa de conversión: 1 dólar americano = 0.29 soles peruanos (Revisado 2/03/2021)

lo menos 150 unidades para que el costo de diseño unitario no exceda los 20 dólares americanos.

Tabla 23
Costes de ingeniería

Componente	Horas	Precio/Hora (S/)	Precio total (S/)
Análisis de Viabilidad	70	20	2100
Diseño del software	100	30	3000
Diseño mecánico	50	30	1500
Diseño electrónico	50	30	1500
Diseño del algoritmo de control	100	30	3000
Total			**11100**

Conclusiones

De acuerdo a lo desarrollado en la tesis, es factible diseñar una plataforma de programación tangible que le pueda servir de herramienta educativa a los profesores para la enseñanza de la programación a niños de la educación básica regular a través de la programación tangible y que, además, sea de bajo costo. El análisis económico arroja un precio final de 555.84 soles o 158.53 dólares americanos, como se observa en la Tabla 22 y, que, además, puede reducirse con la producción en masa, siendo este menor al precio del robot usado en la segunda etapa de la iniciativa "Una laptop por niño".

Se concluye que el lenguaje de programación diseñado permite realizar programas simples y concisos que podrían ser aptos para niños de la Educación Básica Regular, cumpliendo con el primer objetivo específico, debido a que se tomó en cuenta el Currículo Nacional de Educación Básica y se contemplaron las siguientes competencias: Competencia 22: *"Diseña y construye soluciones tecnológicas para resolver problemas de su entorno"*, la Competencia 23: *"Resuelve problemas de cantidad"* y la Competencia 28: *"Se desenvuelve en los entornos virtuales generados por las TIC"* como parte del contexto para su desarrollo. Además, se puede aplicar el concepto de funciones de manera sencilla y directa, lo cual implica que el usuario estará aprendiendo, al mismo tiempo, conceptos del paradigma de programación funcional.

Se concluye que se diseñó un entorno de programación que cumple con el segundo objetivo específico, ya que le permite al usuario ingresar una imagen de sus archivos para tomarla como parámetro del *script* que procesa imágenes y, a partir de este procesamiento, obtener una trayectoria (ver Figura 45 - Figura 47). El *software* que permite procesar la fotografía es eficiente, ya que no se limita a un solo tamaño de imagen; sin embargo, se desconocen todos los dispositivos móviles que podrían ser usados para obtener esta fotografía. El código fue probado con imágenes de diferentes características y logró obtener trayectorias cuando estas tuvieron dimensiones desde 661 pixeles x 789 pixeles hasta 3000 pixeles x 3600 pixeles y con una resolución de, mínimo, 72 ppp.

Se concluye que se logró un diseño electrónico funcional del robot móvil (ver Anexo 2), ya que cubre los requerimientos listados en el Anexo 1 y, además, satisface el objetivo específico de ser un intérprete que pueda seguir trayectorias, puesto que la elección de componentes realizada en el diseño electrónico del intérprete en el Capítulo 2 (ver Tabla 13 – Tabla 17) tomó en cuenta cálculos y fórmulas teóricas para un robot con locomoción diferencial con las mismas características del que se diseñó.

Finalmente, se concluye que el algoritmo de control funciona correctamente ya que logra seguir una trayectoria deseada ingresada por el usuario y, además, tiene un error (diferencia entre la trayectoria deseada y la trayectoria alcanzada) menor a 0.05 unidades (1 unidad = 0.25 m) como se observa en la Tabla 21, pese a que se realizó únicamente un control de trayectoria, mas no de orientación, cumpliendo con el último objetivo específico.

Recomendaciones

A pesar de que el objetivo principal de la tesis no es realizar un análisis pedagógico de la interacción de niños con la plataforma de programación tangible, es recomendable realizar ejercicios experimentales con el público a la que la tesis está destinada, así como consultar especialistas en educación básica regular.

Las piezas tangibles han sido diseñadas de tal forma que, al ser utilizadas para programar, se asemejen a un rompecabezas, lo cual cumple con el fin didáctico y, además, permite ordenar los códigos QR que pertenecen a cada pieza. Sin embargo, se recomienda que la fotografía de estas piezas esté correctamente centrada en estas y con la mínima resolución especificada previamente.

Se han procesado diferentes imágenes que cumplen con los requerimientos especificados en las simulaciones del Capítulo 3; sin embargo, se recomienda realizar más ensayos para probar la eficiencia del algoritmo de procesamiento de imágenes y su alcance.

Bibliografía

[1] García-Valcarcel Muñoz-Repiso, A. and Caballero-Gonzalez, Y. Robótica para desarrollar el pensamiento computacional en Educación Infantil. In: Comunicar, vol 27, pp.63-72, 2019. Disponible en:
https://doi.org/10.3916/C59-2019-06 [Accedido 10-sep, 2019]

[2] Ministerio de Educación, "Currículo Nacional de Educación Básica", Perú, 2016. Disponible en:
http://www.minedu.gob.pe/curriculo/pdf/curriculo-nacional-de-la-educacion-basica.pdf [Accedido 10-sep, 2019]

[3] World Economic Forum, "Global Information Technology Report", 2016. [On line]. Available in:
http://reports.weforum.org/global-information-technology-report-2016/ [Accessed: 10-sep, 2019]

[4] Ministerio de Educación, "El Perú en PISA 2015: Informe nacional de resultados", Oficina de Medición de la Calidad del Aprendizaje, 2017. Disponible en:
http://umc.minedu.gob.pe/wp-content/uploads/2017/04/Libro_PISA.pdf [Accedido 10-sep, 2019]

[5] UNICEF, UNESCO, OMS y Banco Mundial, Para la vida, 3ra edición. Nueva York, N.Y.: UNICEF, 2002, pp. 21-37.

[6] L. Sangacha and J. Ortiz, "Estrategia de enseñanza para el desarrollo de habilidades a través de la programación empleando herramientas interactivas", Espirales: Revista Multidisciplinaria de Investigación, 2017.

[7] D. Scaradozzi, L. Sorbi, A. Pedale, M. Valzano and C. Virginie, "Teaching robotics at the primary school: an innovative approach", Procedia - Social and Behavioral Sciences, vol 174, pp. 3838–3846, Elsevier Ltd., 2015.

[8] MIT Media Lab, "LEGO WeDo and OLPC Peru: national collaboration", One Laptop Per Child, 2011. Available in:
http://blog.laptop.org/2011/02/12/lego-wedo-olpc-peru/#.XYFK3ChKjIW [Accedido 16-sep, 2019]

[9] Quispe, L., David, C., & Bolívar Díaz, E. J. Una Laptop Por Niño en escuelas rurales del Perú: un análisis de las barreras y facilitadores. CIES, 2009. Disponible en: http://repositorio.minedu. gob.pe/handle/123456789/800 [Accesed 17-sep, 2019]

[10] Paz O., María; Martínez O., María F. "¿Qué ha pasado para que el programa ULPN en el Perú sea (hasta ahora) un fracaso?" en 14 Encuentro latinoamericano: Comunicación e industria digital: Tendencias, escenarios y oportunidades, Lima, Perú, 2012. Disponible en: http://catalogo.ulima.edu.pe/conferencias/felafacs2012/eje3/26.pdf [Accedido 17-sep, 2019]

[11] H. Suzuki y H. Kato, "AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning," Proceedings of 4th European Logo Conference, pp. 297-303, Athens, 1993.

[12] P. C. Caceres, R. P. Venero and F. C. Cordova, "Tangible programming mechatronic interface for basic induction in programming," 2018 IEEE Global Engineering Education Conference (EDUCON), Tenerife, 2018, pp. 183-190. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8363226&isnumber=8363090 [Accesed 21-sep, 2019]

[13] A. C. Smith, "Tangible Cubes as Programming Objects," 16th International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06), Hangzhou, 2006, pp. 157-161. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=4089231&isnumber=4089191 [Accesed 21-sep, 2019]

[14] D. Kwon, H. Kim, J. Shim and W. Lee, "Algorithmic Bricks: A Tangible Robot Programming Tool for Elementary School Students," in IEEE Transactions on Education, vol. 55, no. 4, pp. 474-479, Nov. 2012. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=6172535&isnumber=6341131 [Accesed 21-sep, 2019]

[15] T. Sapounidis and S. Demetriadis, "Touch Your Program with Hands: Qualities in Tangible Programming Tools for Novice," 2011 15th Panhellenic Conference on Informatics, Kastonia, 2011, pp. 363-367. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=6065115&isnumber=6065036 [Accesed 22-sep, 2019]

[16] S. Kakehashi, T. Motoyoshi, K. Koyanagi, T. Ohshima and H. Kawakami, "P-CUBE: Block Type Programming Tool for Visual Impairments," 2013 Conference on

Technologies and Applications of Artificial Intelligence, Taipei, 2013, pp. 294-299. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=6783884&isnumber=6783819 [Accesed 22-sep, 2019]

[17] N. Dümmel, B. Westfechtel and M. Ehmann, "Work in Progress: Gathering Requirements and Developing an Educational Programming Language," 2019 IEEE Global Engineering Education Conference (EDUCON), Dubai, United Arab Emirates, 2019, pp. 1-4. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8725073&isnumber=8725024 [Accesed 22-sep, 2019]

[18] M. Y. İmamoğlu and D. Çetinkaya, "A rule-based decision support system for programming language selection," 2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA), London, 2017, pp. 71-75. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8169904&isnumber=8169886 [Accesed 23-sep, 2019]

[19] T. McNerney, “Tangible Programming Bricks: an approach to making programming accessible to everyone”, 2000. Tesis de Maestría, MIT, Cambridge, Massachusetts.

[20] Ray, B, Posnett, DP, Devanbu, P & Filkov, V. A Large-Scale Study of Programming Languages and Code Quality in GitHub. Communications of the ACM, 2017, vol. 60, no. 10, pp. 91–100, viewed 22 October 2019. Available in: http://search.ebscohost.com.ezproxybib.pucp.edu.pe:2048/login.aspx?direct=true&db=iih&AN=125351984&lang=es&site=eds-live&scope=site [Accesed 23-sep, 2019]

[21] Trejos Buriticá, OI 2018, ‘Aprovechamiento de los tipos de pensamiento matemático en el aprendizaje de la programación funcional’, Tecnura, vol. 22, no. 56, pp. 29–39, viewed 22 October 2019. Available in: http://search.ebscohost.com.ezproxybib.pucp.edu.pe:2048/login.aspx?direct=true&db=fua&AN=131635927&lang=es&site=ehost-live [Accesed 24-sep, 2019]

[22] J. Ranjani, A. Sheela and K. P. Meena, "Combination of NumPy, SciPy and Matplotlib/Pylab -a good alternative methodology to MATLAB - A Comparative analysis," 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), CHENNAI, India, 2019, pp. 1-5. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8741475&isnumber=8741349 [Accesed 29-sep, 2019]

[23] NumFOCUS, 2019. [Online]. Available in:
https://numpy.org/ [Accesed 22-sep, 2019]

[24] NumFOCUS, 2019. [Online]. Available in:
https://docs.scipy.org/doc/scipy/reference/tutorial/ndimage.html#correlation-and-convolution [Accesed 22-sep, 2019]

[25] X. Yan, G. Jing, M. Cao, C. Zhang, Y. Liu and X. Wang, "Research of Sub-Pixel Inner Diameter Measurement of Workpiece Based on OpenCV," 2018 International Conference on Robots & Intelligent System (ICRIS), Changsha, 2018, pp. 370-373. Available in:
http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8410307&isnumber=8410169 [Accesed 28-sep, 2019]

[26] V. Cietto, C. Gena, I. Lombardi, C. Mattutino and C. Vaudano, "Co-designing with kids an educational robot," 2018 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), Genova, Italy, 2018, pp. 139-140. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8625810&isnumber=8625716 [Accesed 22-sep, 2019]

[27] G. Trovato, F. Cuellar y M. Nishimura, “Introducing ‘theomorphic robots”, 016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), Cancún, 2016, pp. 1245-1250.

[28] González, R, Rodríguez, F & Guzmán, JL 2015, ‘Robots Móviles con Orugas Historia, Modelado, Localización y Control’, Revista Iberoamericana de Automática e Informática Industrial, vol. 12, no. 1, pp. 3–12. Disponible en:
http://search.ebscohost.com.ezproxybib.pucp.edu.pe:2048/login.aspx?direct=true&db=edselp&AN=S1697791214000788&lang=es&site=eds-live&scope=site [Accedido 29-sep, 2019]

[29] G. Perez-Paina, E. J. Guizzo, I. Torres, D. Gonzalez-Dondo, C. Paz and F. Trasobares, "Open hardware wheeled mobile robot for educational purposes," 2018 Ninth Argentine Symposium and Conference on Embedded Systems (CASE), Cordoba, 2018, pp. 13-18. Available in:
http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8542162&isnumber=8542158 [Accesed 01-oct, 2019]

[30] F. Mondada et al., "Bringing Robotics to Formal Education: The Thymio Open-Source Hardware Robot," in IEEE Robotics & Automation Magazine, vol. 24, no. 1, pp. 77-85, March 2017. Available in:

http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=7859350&isnumber=7886370 [Accesed 02-oct, 2019]

[31] K. Kaneko et al., "Humanoid Robot HRP-5P: An Electrically Actuated Humanoid Robot With High-Power and Wide-Range Joints," in IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1431-1438, April 2019. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8630006&isnumber=8581687 [Accesed 02-oct, 2019]

[32] S. B. A. Kashem, M. Tabassum and M. Chai, "A novel design of an amphibious robot having webbed feet as duck," 2017 International Conference on Computer and Drone Applications (IConDA), Kuching, 2017, pp. 17-21. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8270392&isnumber=8270381 [Accesed 02-oct, 2019]

[33] K. D. H. Thi, M. C. Nguyen, H. T. Vo, V. M. Tran, D. D. Nguyen and A. D. Bui, "Trajectory tracking control for four-wheeled omnidirectional mobile robot using Backstepping technique aggregated with sliding mode control," 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 2019, pp. 131-134.

[34] J. Cornejo, J. Magallanes, E. Denegri and R. Canahuire, "Trajectory Tracking Control of a Differential Wheeled Mobile Robot: a Polar Coordinates Control and LQR Comparison" 2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Lima, 2018, pp. 1-4. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8526366&isnumber=8526365 [Accesed 05-oct, 2019]

[35] L. Fan, Y. Zhang and S. Zhang, "Dynamic Trajectory Tracking Control of Mobile Robot," 2018 5th International Conference on Information Science and Control Engineering (ICISCE), Zhengzhou, 2018, pp. 728-732. Available in: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8612654&isnumber=8612498 [Accesed 05-oct, 2019]

[36] C. De La Cruz, "Control de Formación de robots", tesis doctoral, Universidad Nacional de San Juan, Argentina, 2007.

[37] J. Gausemeier, S. Moehringer, “VDI 2206- A New Guideline for the Design of Mechatronic Systems”, IFAC Proceedings Volumes,Volume 35, Issue 2, 2002, pp. 785-790. Available in:

http://www.sciencedirect.com/science/article/pii/S1474667017340351 [Accesed 15-oct, 2019]

[38] OpenCV team, 2019. [Online]. Available in:
https://opencv.org [Accesed 11-sep, 2019]

[39] Reglamento de la Ley N° 28376, Ley que prohíbe y sanciona la fabricación, importación, distribución y comercialización de juguetes y útiles de escritorio tóxicos o peligrosos. Resolución Ministerial N° 1020, 2007. Disponible en: http://www.digesa.minsa.gob.pe/DEPA/juguetes_utiles/pdf/juguetes.pdf [Accedido 10-jan, 2019]

[40] Ley Nº 24029, Ley del Profesorado. Disponible en:
http://www.minedu.gob.pe/normatividad/leyes/Ley24029.php [Accedido 10-jan, 2019]

[41] P. Kumar, S. Dwari, Utkarsh, S. Singh J. Kumar, 'Investigation and Development of 3D Printed Biodegradable PLA as Compact Antenna for Broadband Applications', IETE Journal of Research, vol. 66, no. 1, pp. 53–64. Available in:
http://search.ebscohost.com.ezproxybib.pucp.edu.pe:2048/login.aspx?direct=true&db=iih&AN=141750931&lang=es&site=ehost-live [Accesed 15-jan, 2019]

[42] Mochila escolar no debe exceder del 15% del peso del estudiante, Instituto Nacional de Salud, 2018. Disponible en:
https://web.ins.gob.pe/es/prensa/noticia/mochila-escolar-no-debe-exceder-del-15-del-peso-del-estudiante [Accedido 10-jan, 2019]

[43] LEGO® Education WeDo 2.0 Core Set, Lego Education. Available in:
https://education.lego.com/en-us/products/lego-education-wedo-2-0-core-set/45300 [Accesed 15-jan, 2019]

[44] J. Labra Gayo, Cueva Lovelle, R. Castanedo, A. Juan, M. Luengo, F. Ortin, Intérpretes y Diseño de Lenguajes de Programación, 2003.

[45] Foundations of the Theory of Signs. Charles William Morris, 1938.

ANEXOS

Anexo 1

A continuación, se muestra la lista de exigencias descrita en el Capítulo 3.

Tabla A1
Lista de requerimientos

Proyecto: Plataforma de programación tangible		
Responsable: Maria Alejandra Manrique Zarate		
Características	**Requerimientos**	**Deseo o exigencia**
Geometría	Volumen del robot $\leq 25 \times 25 \times 25$ cm Volumen de las piezas $\leq 10 \times 10 \times 1.5$ cm	D
Cinemática	Velocidad del intérprete será de una longitud del mismo por cada 3 segundos.	E
Control	Conseguir que se siga la trayectoria y se alcance la posición final con un error menor a 0.05 unidades.	E
Electrónica (Hardware)	Autonomía $\geq$ 3 horas Locomoción del intérprete: diferencial	D
Software	Lenguaje de programación: paradigma basado en reglas	E
Comunicaciones	Comunicación entre entorno e intérprete: inalámbrica	D
Seguridad	Material del intérprete robot móvil: PLA Material de las piezas tangibles: madera	E
Ergonomía	Peso máximo de la TPL = 3 kg	E
Costos	Costo de 1 set $\leq$ 214 dólares americanos	E

Anexo 2

El diagrama esquemático del intérprete robot móvil es que el que se muestra en la Figura A1.

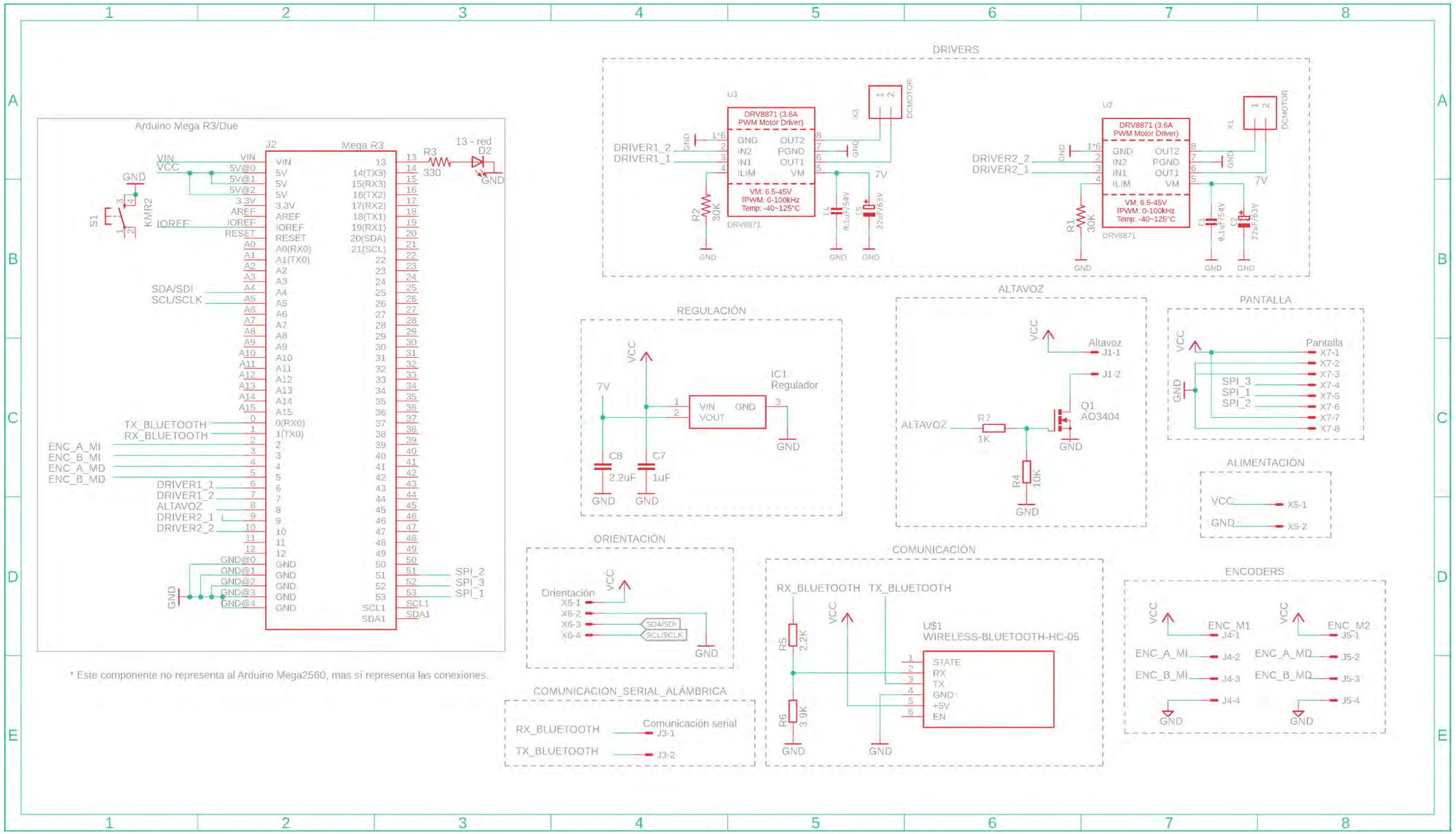


Figura A1. Diagrama esquemático del robot móvil

Anexo 3

El diseño del circuito impreso (PCB) que ira acoplado al microcontrolador se realizó a través del software Eagle y sus 2 capas se muestran a continuación.

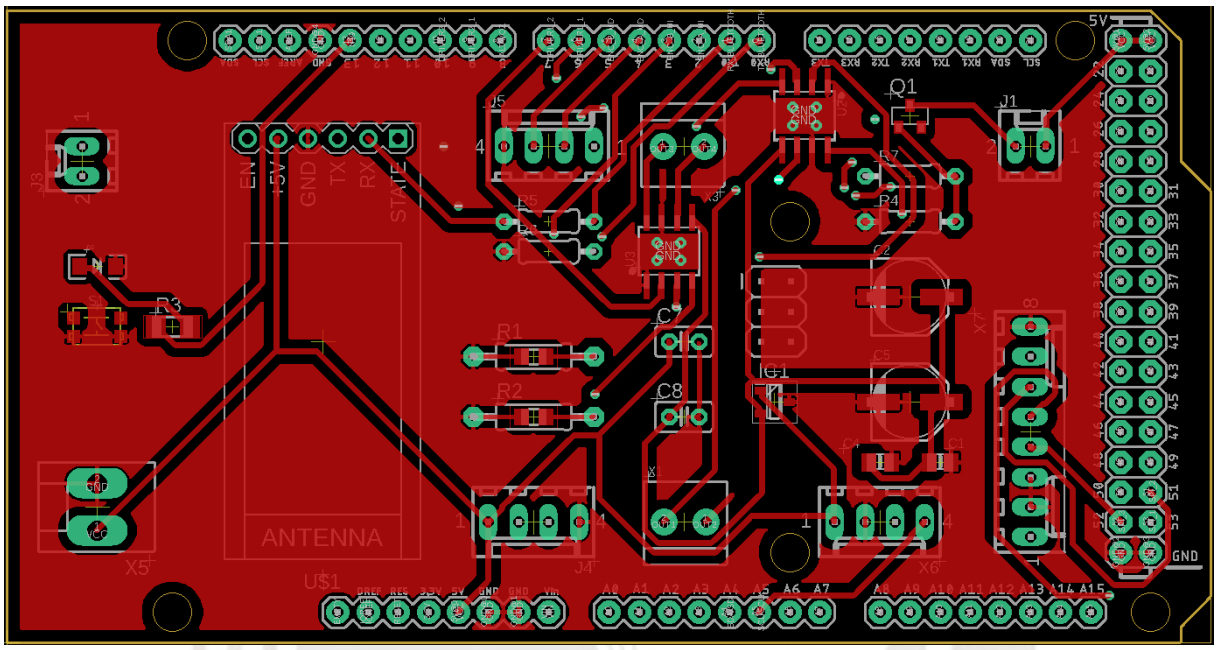


Figura A2. Capa 1 del PCB diseñado mediante Eagle

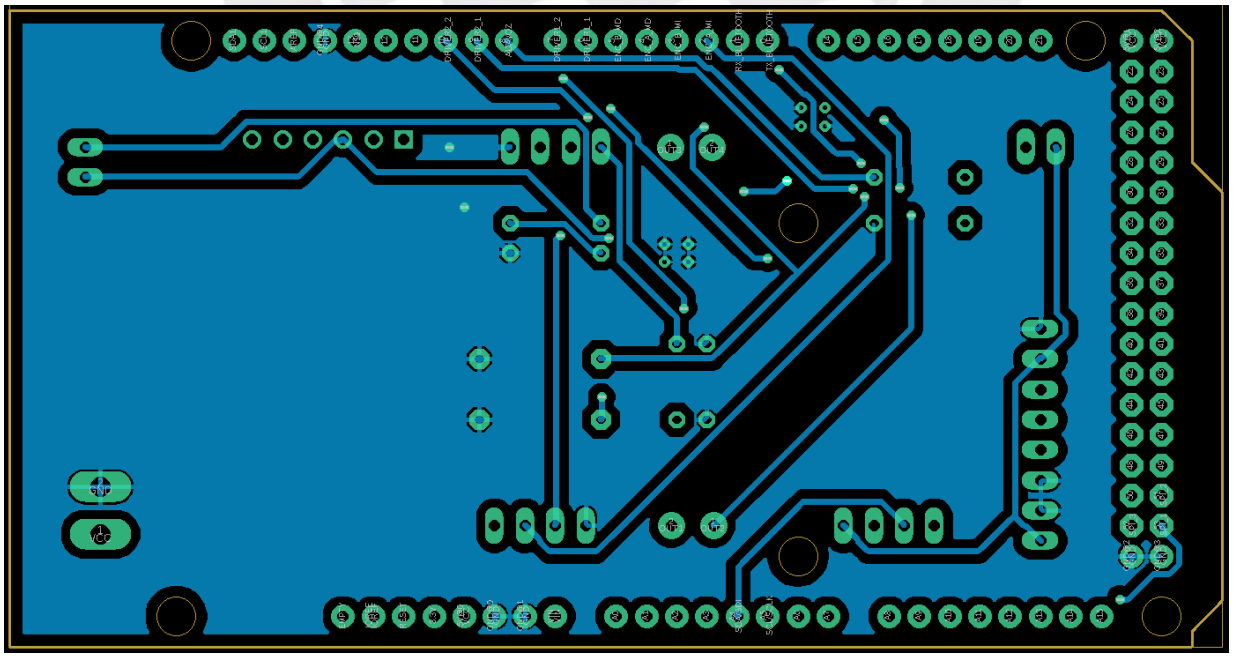


Figura A3. Capa 2 del PCB diseñado mediante Eagle

Además, se muestra el diagrama de conexiones del PCB junto a los periféricos que van ubicados en el chasis de intérprete robot móvil.

Tabla A2
Listado de periféricos

N°	Periférico
1	Baterías
2	Comunicación serial alámbrica
3	Motor izquierdo
4	Motor derecho
5	Giroscopio
6	Pantalla
7	Altavoz
8	Módulo bluetooth
9	Microcomputadora
10	Batería recargable

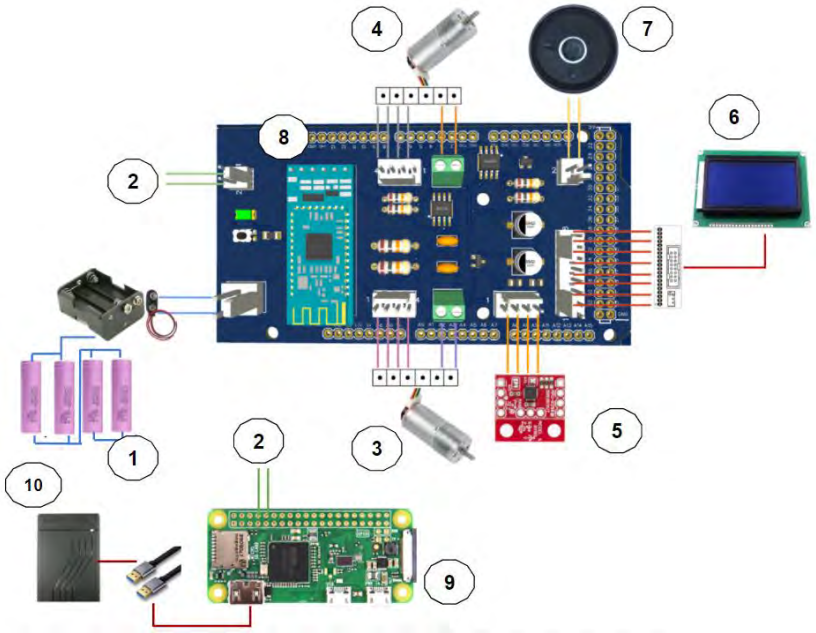


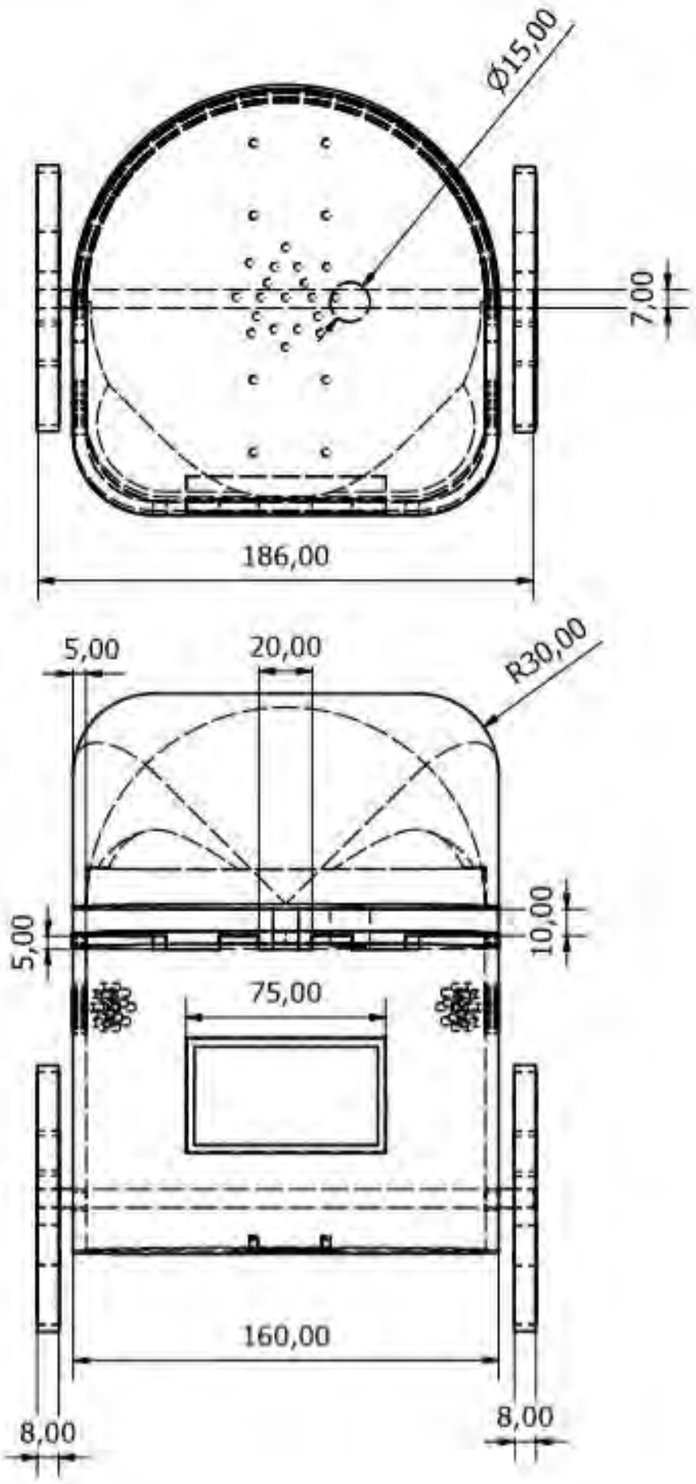
Figura A4. Diagrama de conexiones

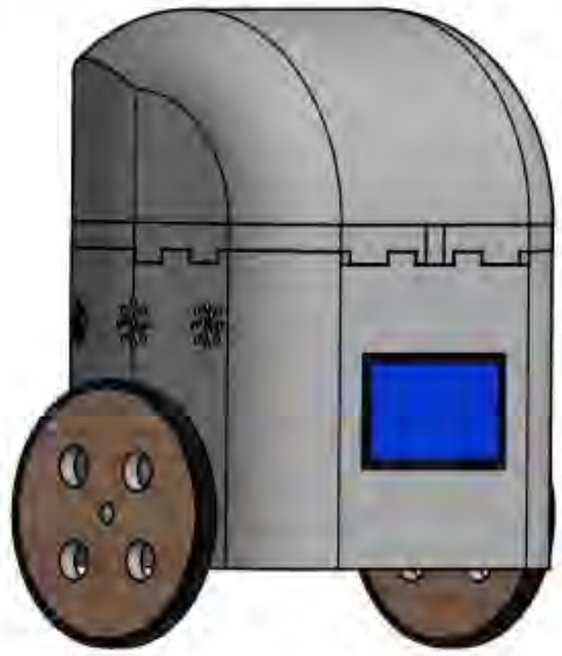
Anexo 4

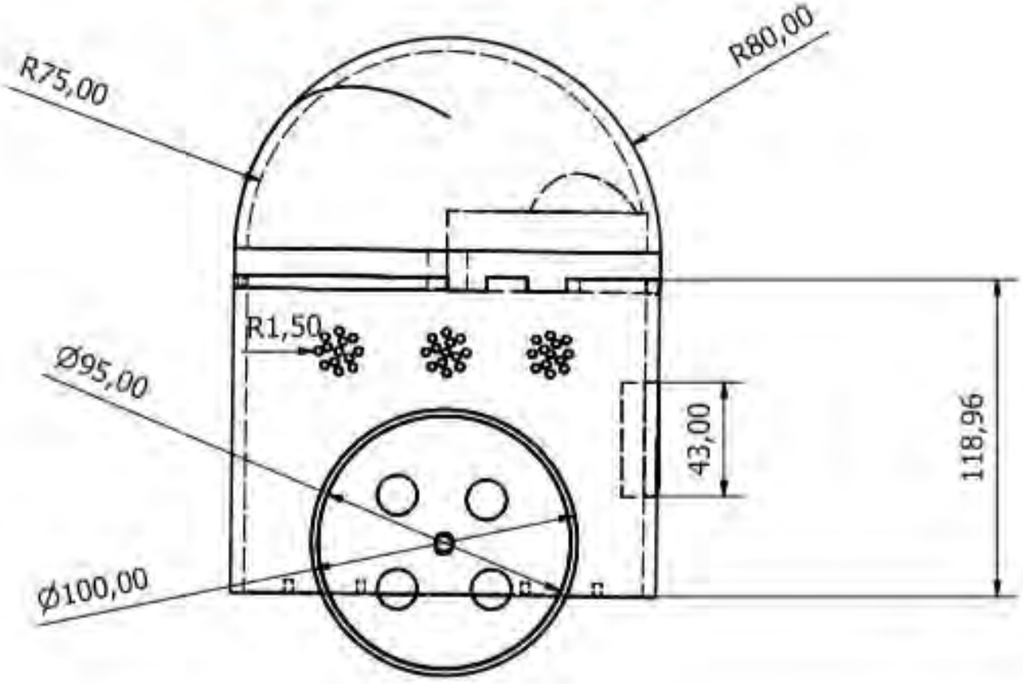
Los planos mecánicos del intérprete robot móvil se encuentran organizados en la Tabla A3.

Tabla A3
Listado de planos del intérprete robot móvil

Plano	Descripción
A1	Intérprete robot móvil – Vista general
A2	Intérprete robot móvil – Vista explosiva
A3	Cuerpo del robot móvil
A4	Jebe de rueda
A5	Rueda
A6	Tapa
A7	Cabeza
A8	Soporte de rueda loca

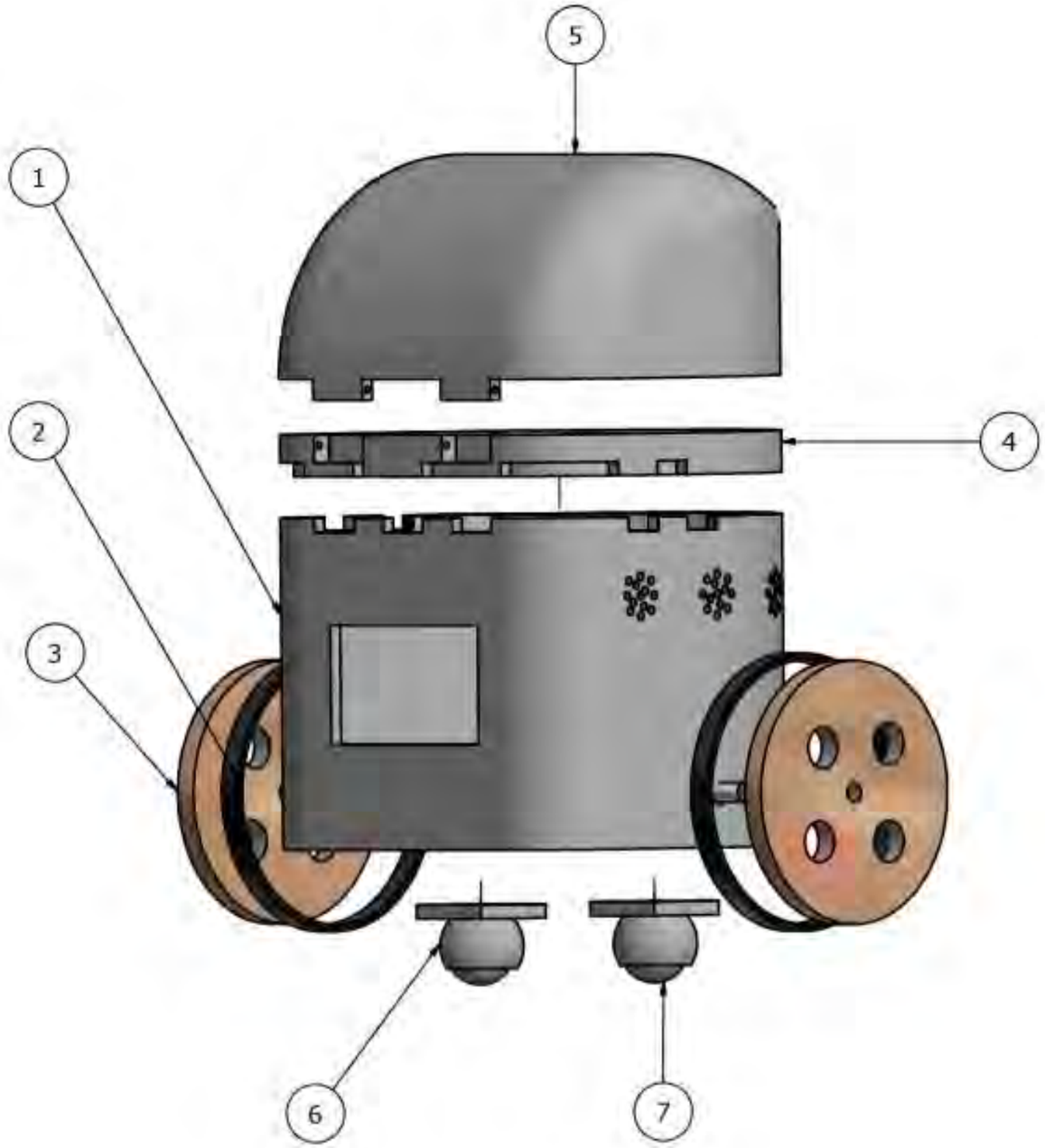






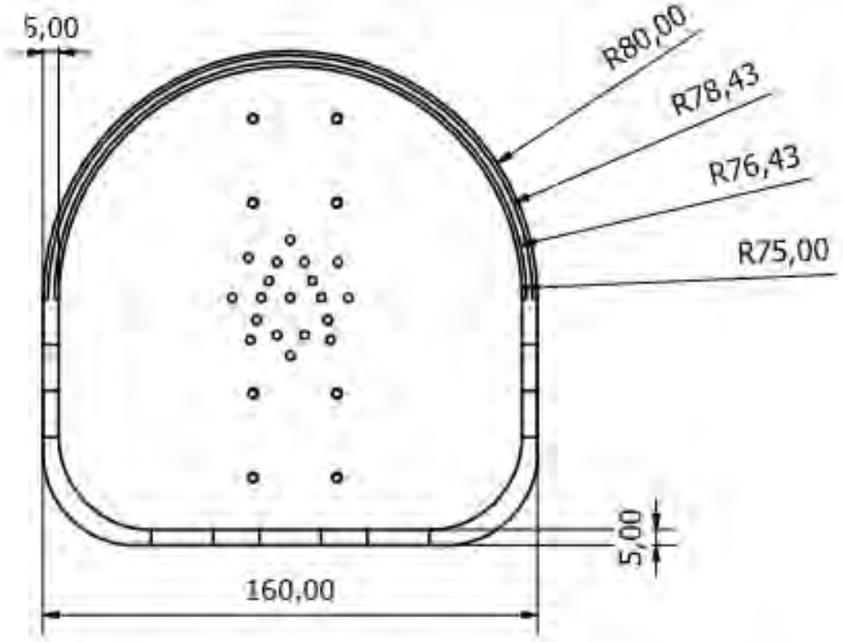
Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
Alejandra Manrique					13/06/2020	A1
Chasis del robot móvil		Trabajo de tesis				
		Vista general_robot móvil		Edición	Hoja 1 / 1	

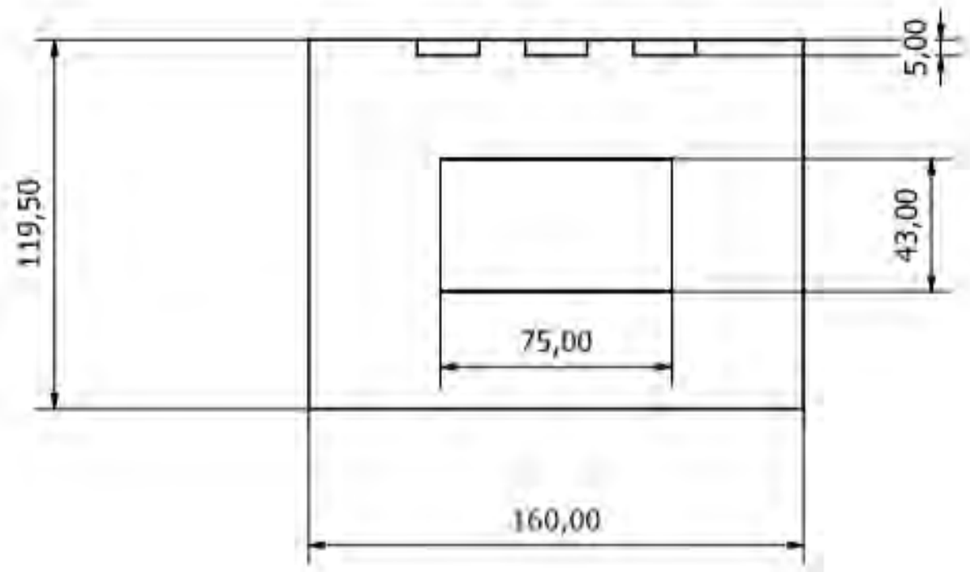
LISTA DE PIEZAS		
ELEMENTO	CTDAD	NOMBRE DE PIEZA
1	1	Cuerpo
2	2	Jebe de rueda
3	2	Rueda
4	1	Tapa
5	1	Cabeza
6	2	Soporte de rueda loca
7	2	Rueda loca

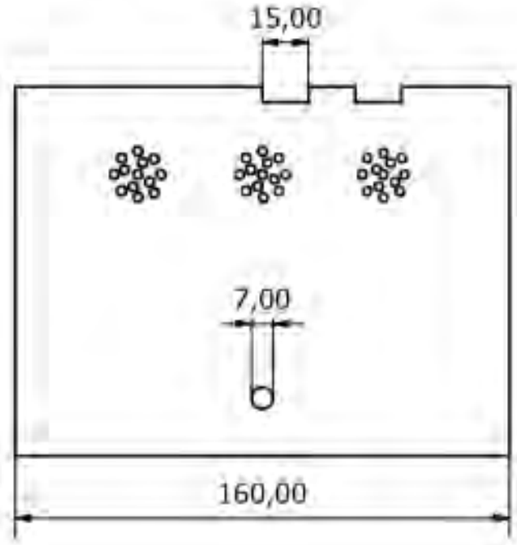


Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
Alejandra Manrique					26/06/2020	A2

Intérprete robot móvil	Trabajo de tesis		
	Vista explosiva_robot móvil	Edición v1.0	Hoja 1 / 1

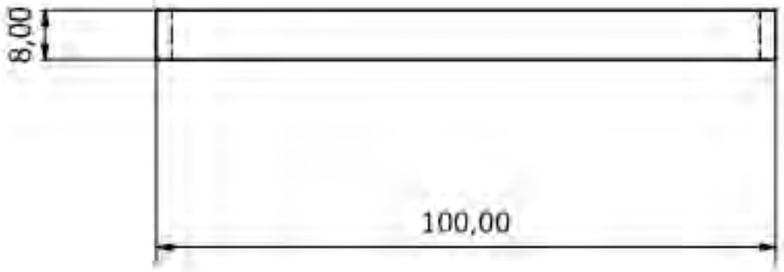


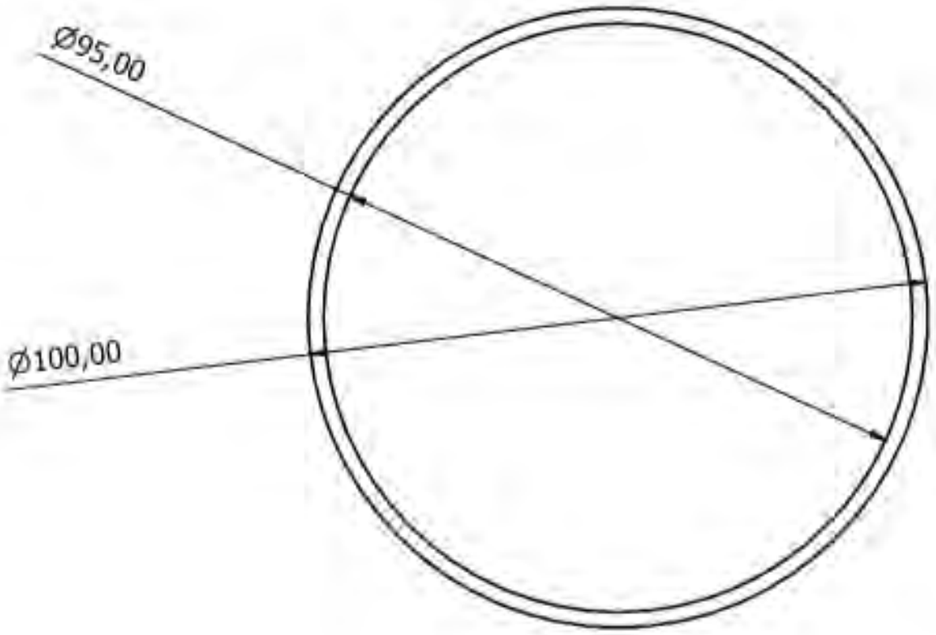


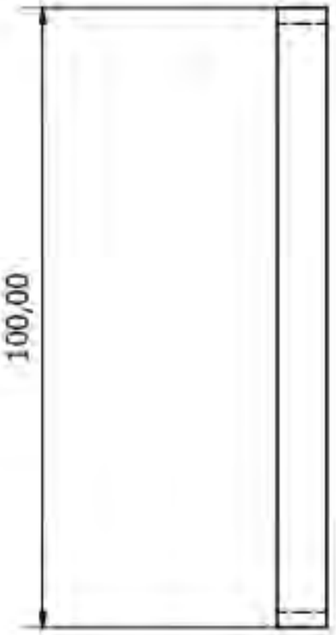


Diseño de	Revisado por:	Aprobado por:	Fecha		Fecha	
Alejandra Manrique					26/06/2020	A3

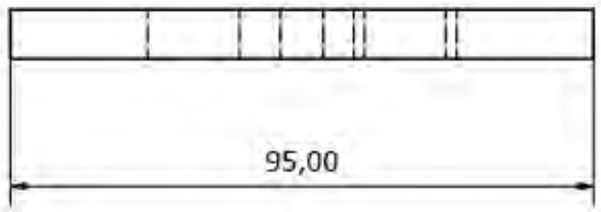
Cuerpo del robot móvil	Trabajo de tesis		
	Cuerpo	Edición	Hoja 1 / 1

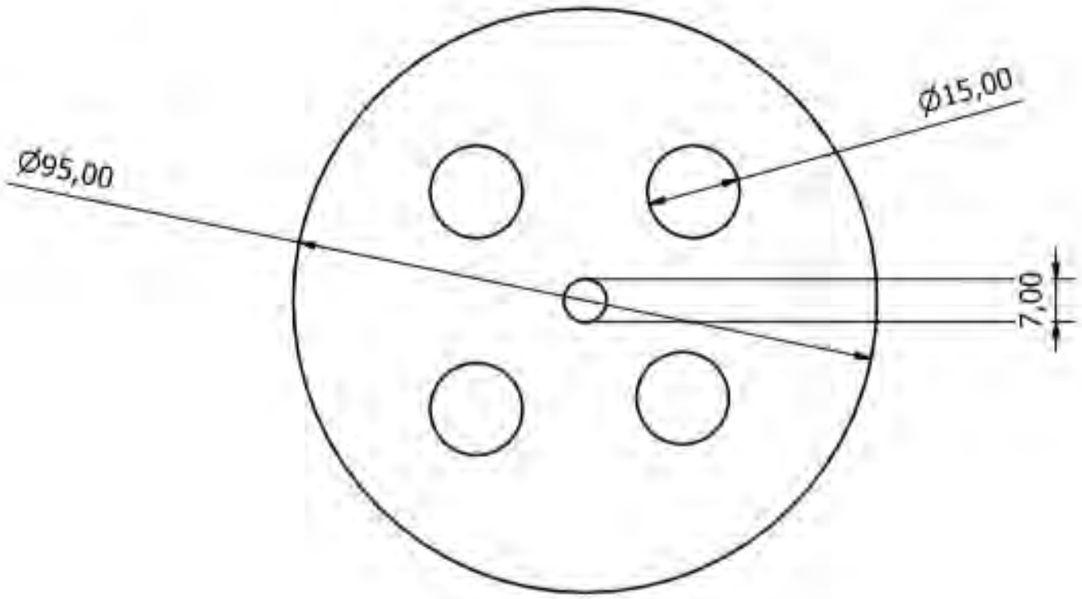


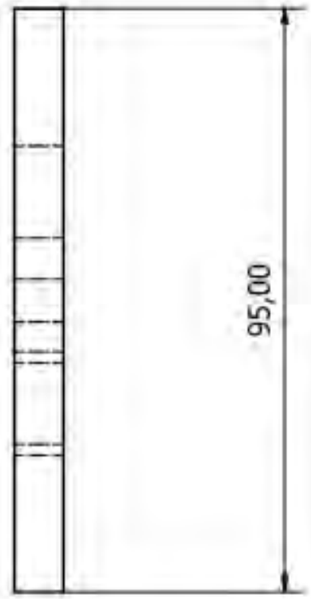




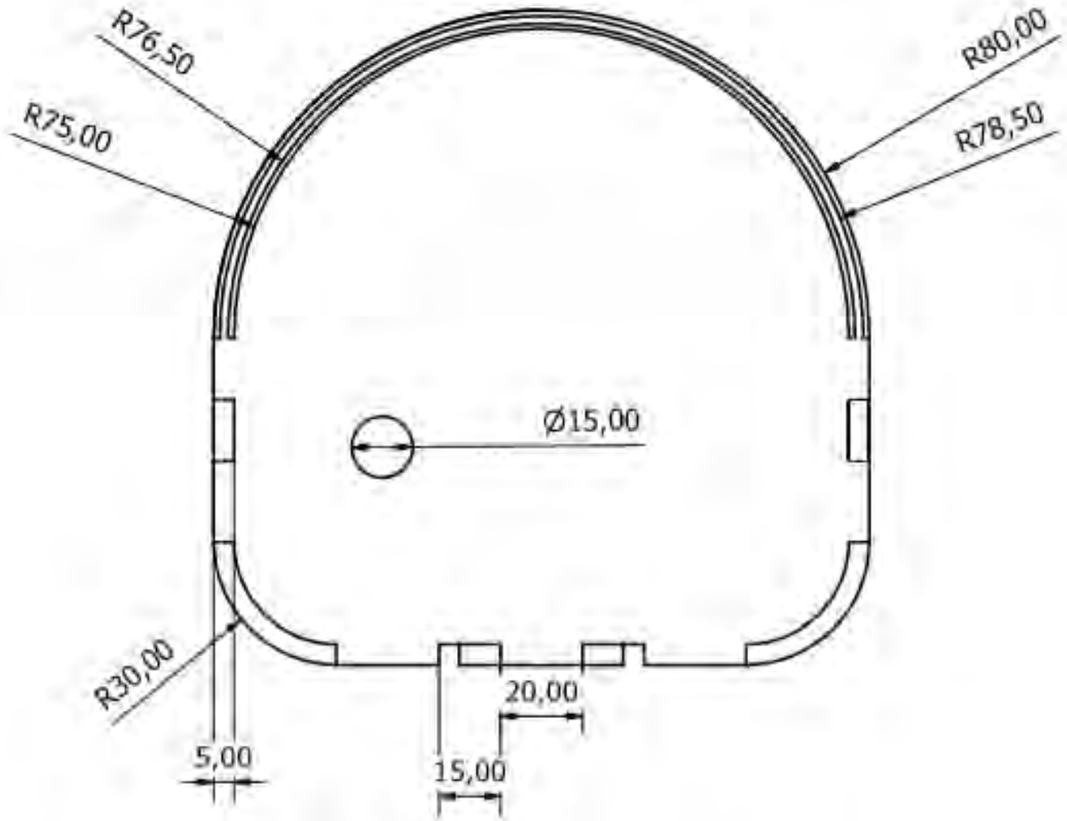
Diseño de	Revisado por	Aprobado por	Fecha		Fecha	A4
Alejandra Manrique					26/06/2020	
Jebe de rueda del robot móvil			Trabajo de tesis			
			Jebe		Escala	Hoja 1 / 1

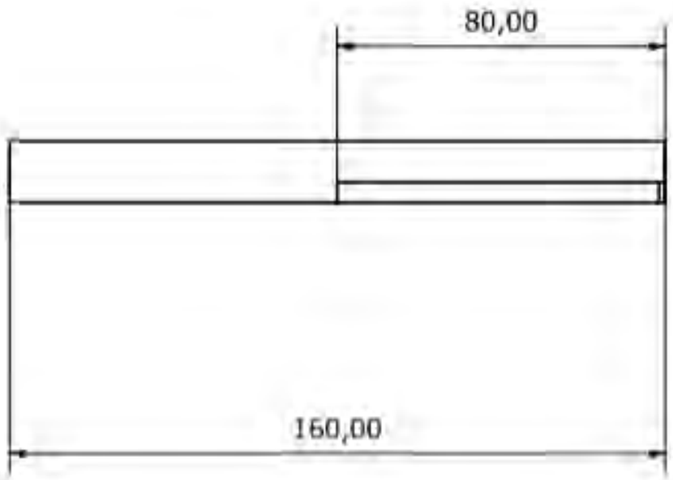


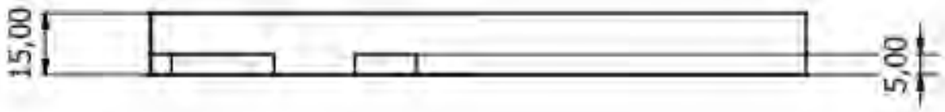




Diseñó de	Revisado por	Aprobado por	Fecha		Fecha	A5
Alejandra Manrique					26/06/2020	
Rueda del robot móvil				Trabajo de tesis		
				Rueda	Edición	Hoja 1 / 1

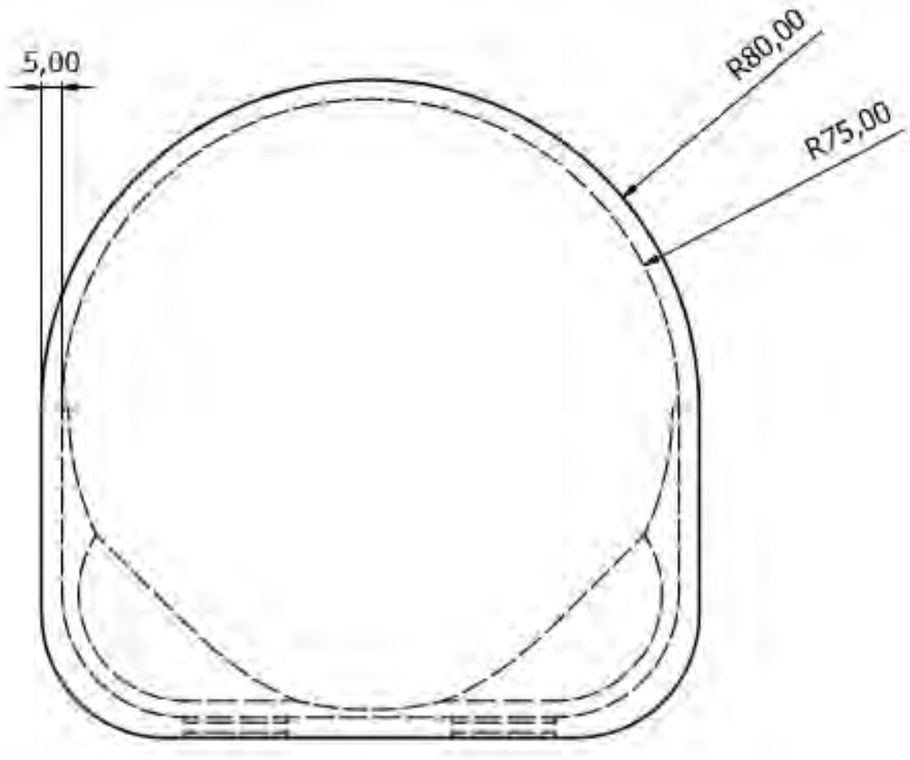


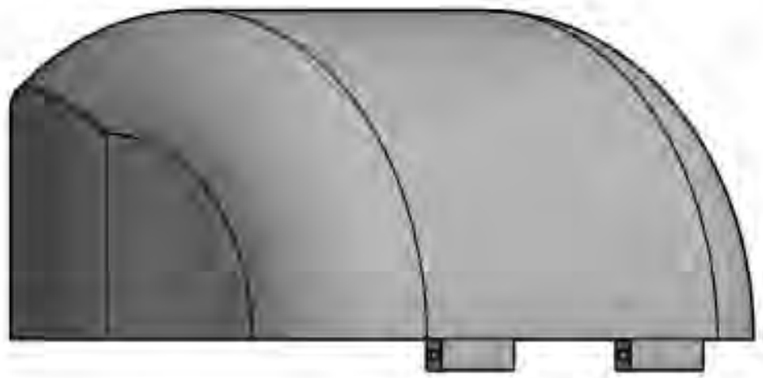


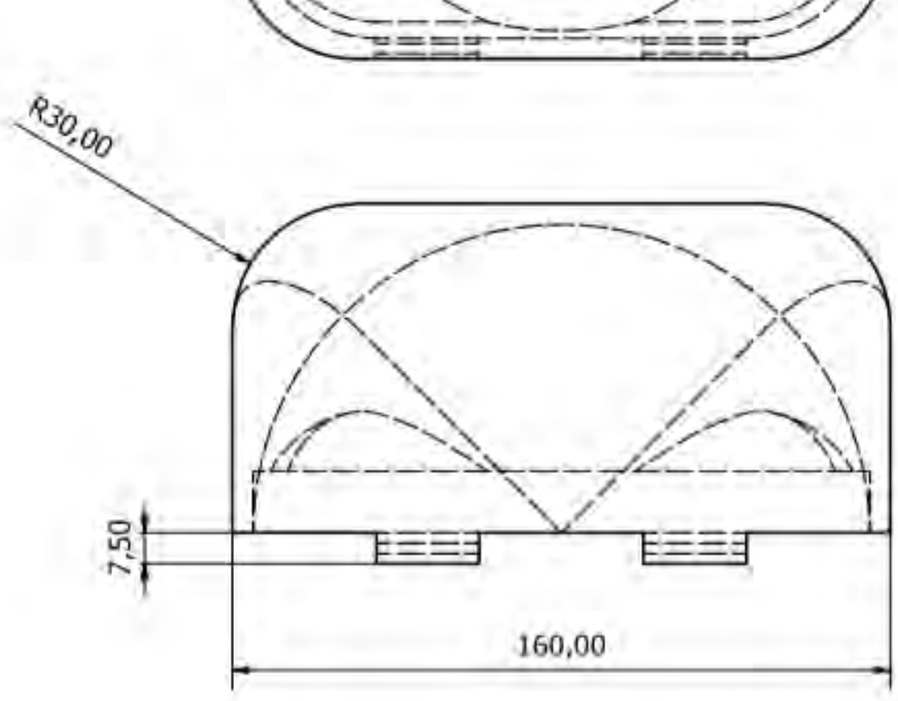


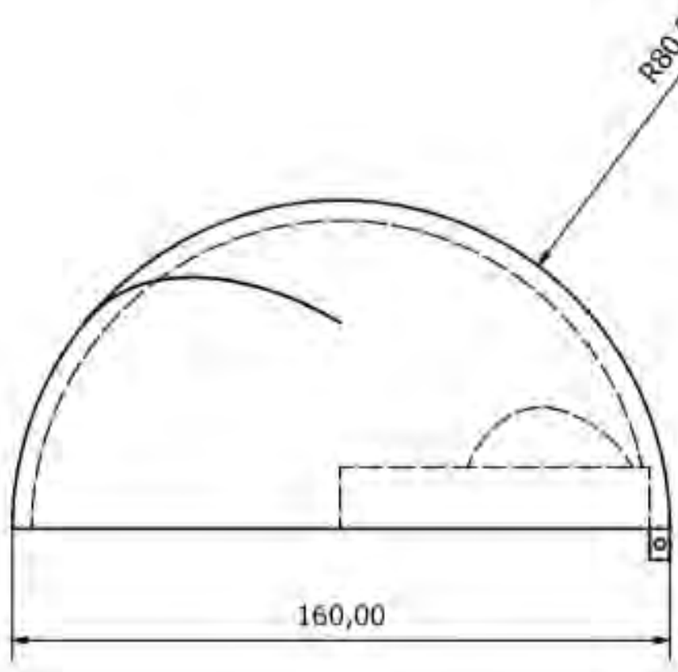
Diseño de	Revisado por	Aprobado por	Fecha		Fecha	A5
Alejandra Manrique					26/06/2020	

Tapa del robot móvil	Trabajo de tesis		
	Tapa	Edición	Hoja 1 / 1

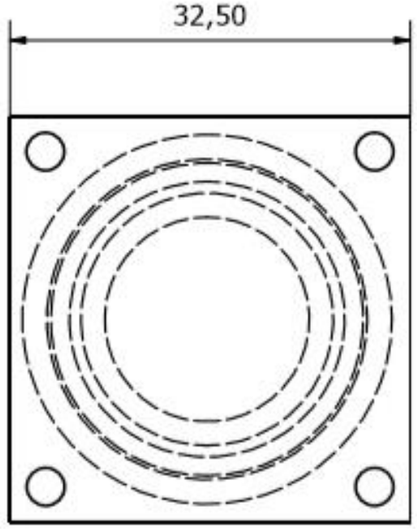


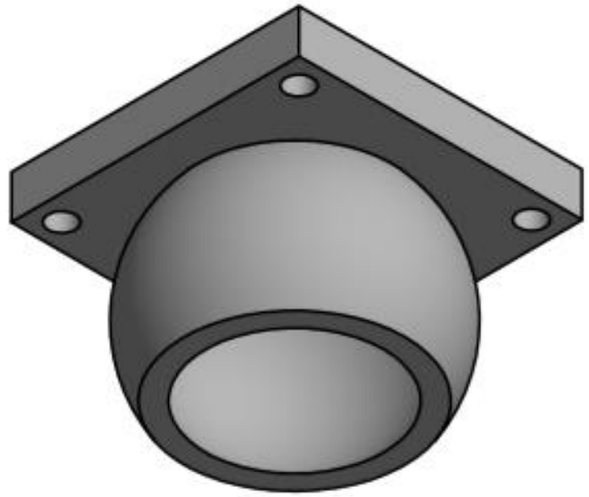


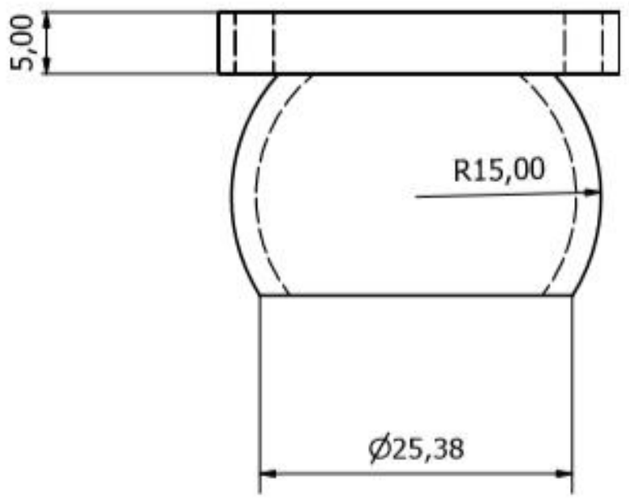


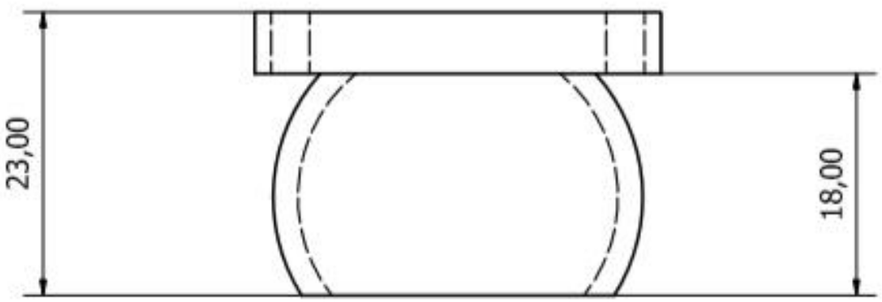


Diseño de: Alejandra Manrique	Revisado por:	Aprobado por:	Fecha:	Fecha: 26/06/2020	A7
Cabeza del robot móvil		Trabajo de tesis			
		Cabeza		Edición	Hoja 1 / 1









Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
Alejandra Manrique					26/06/2020	A7
Soporte de rueda loca del robot móvil				Trabajo de tesis		
				Soporte de rueda	Edición	Hoja 1 / 1

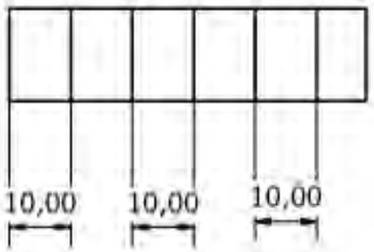
Anexo 5

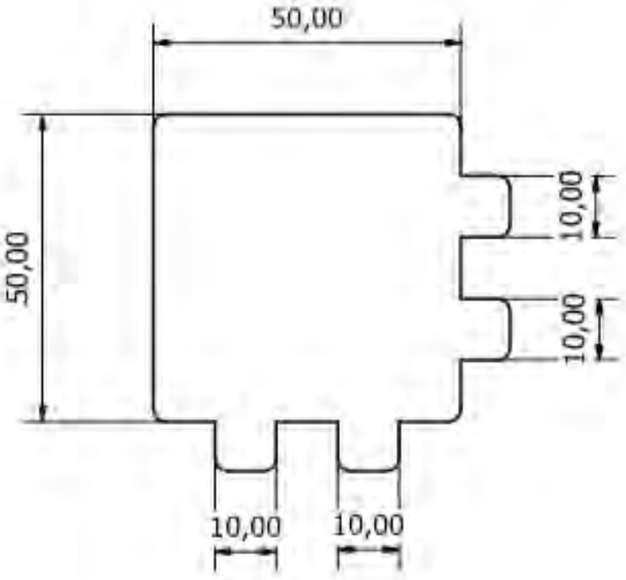
Los planos mecánicos de las piezas tangibles se encuentran organizados en la Tabla A4.

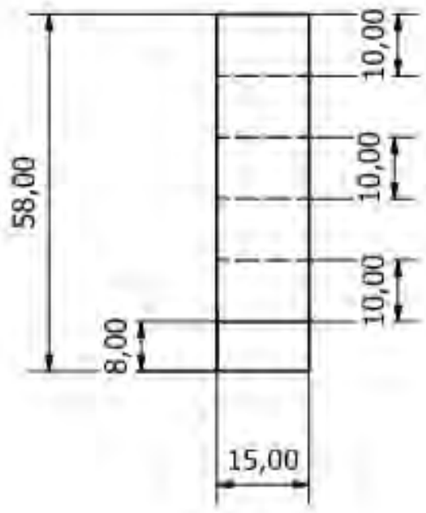
Tabla A4

Listado de planos de las piezas tangibles

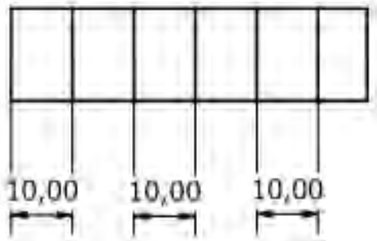
Plano	Descripción
B1	Pieza 1
B2	Pieza 2
B3	Pieza 3
B4	Pieza 4
B5	Pieza 5
B6	Piezas conjugadas

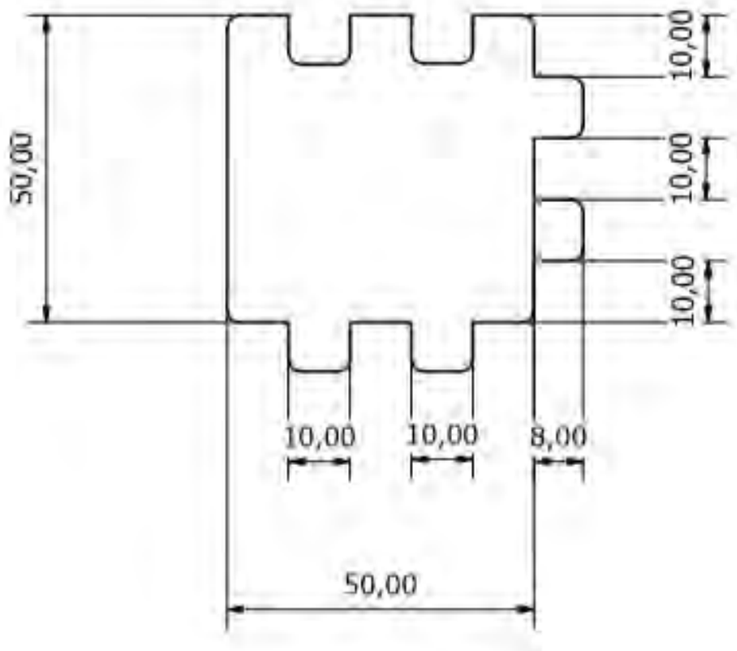


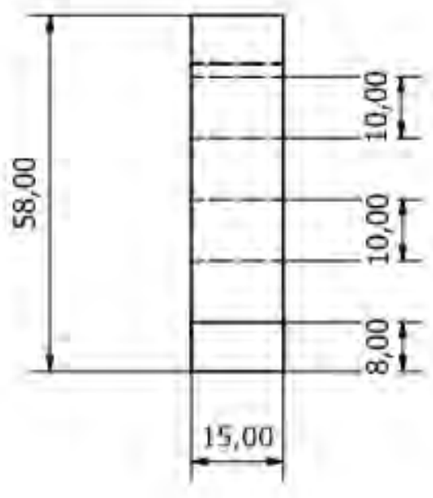




Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
Alejandra Manrique					22/05/2020	B1
Piezas tangibles				Trabajo de tesis		
				Pieza1-1	Escala	Hoja 1 / 1

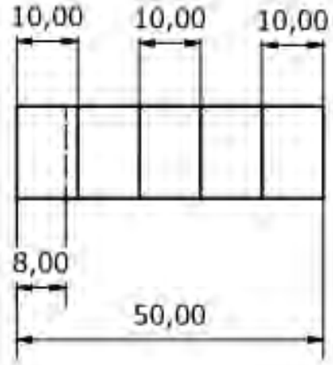


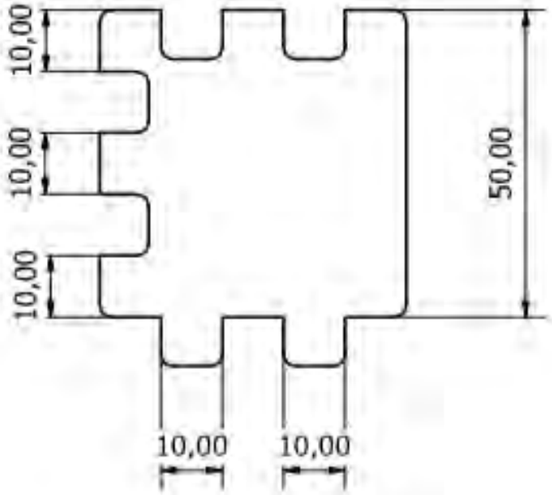


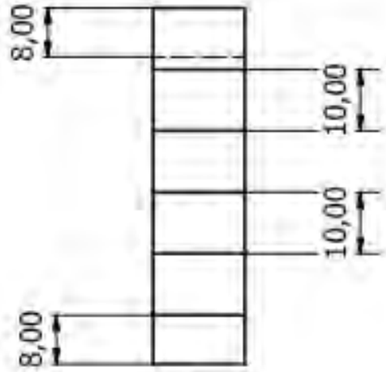


Diseño por	Revisado por	Aprobado por	Fecha		Fecha	
Alejandra Manrique					22/05/2020	B2

Piezas tangibles	Trabajo de tesis		
	Pieza 1-2	Escala	Hoja 1 / 1

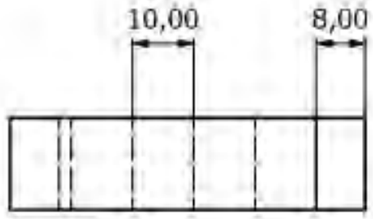


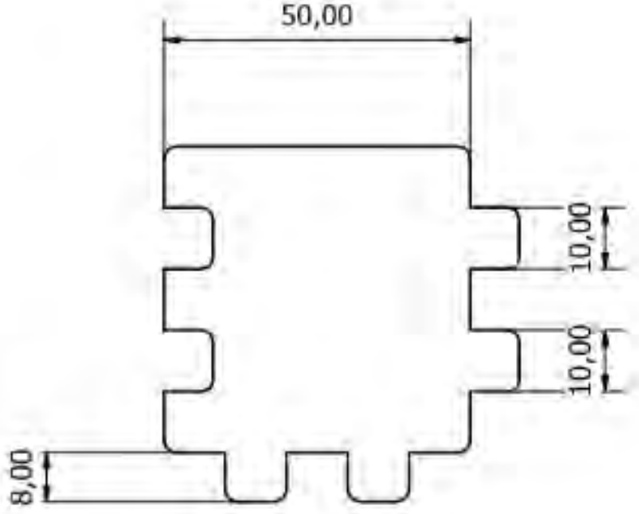


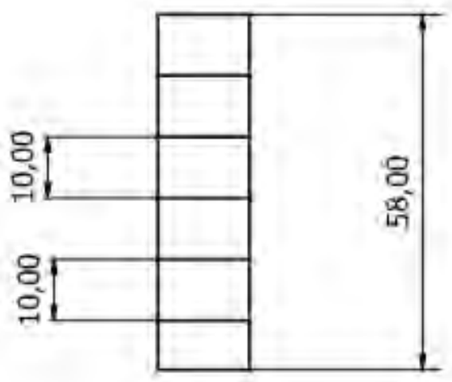


Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
Alejandra Manrique					23/05/2020	B3

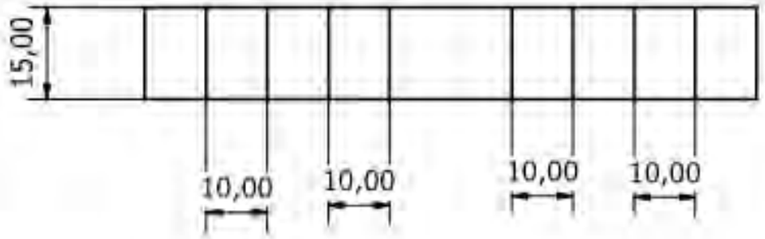
Piezas tangibles	Trabajo de tesis		
	Pieza1-3	Edición	Hoja 1 / 1

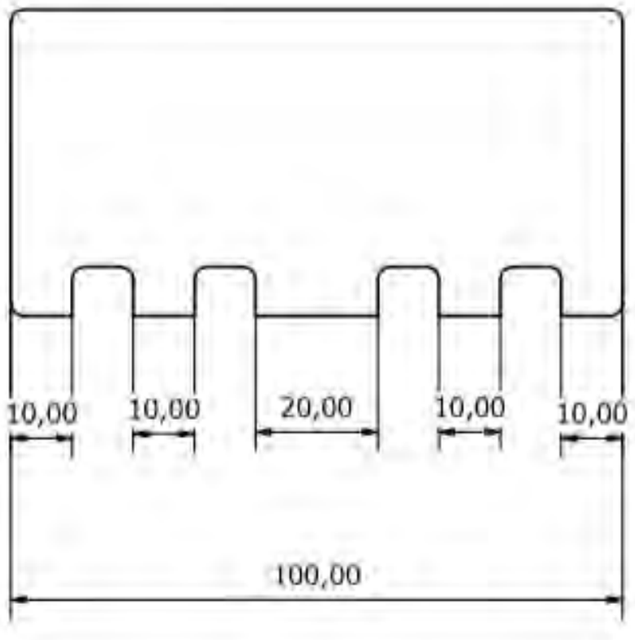


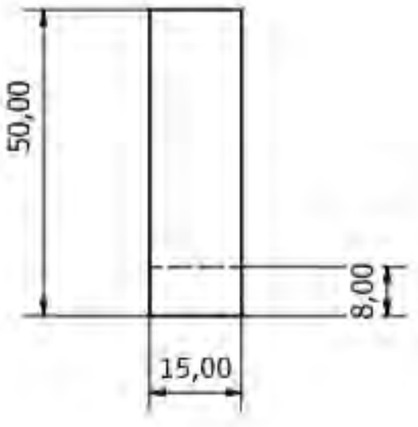




Diseño de: Alejandra Manrique	Revisado por:	Aprobado por: Fecha		Fecha 26/06/2020	B4
Piezas tangibles			Trabajo de tesis		
			Pieza 1-4	Edición	Hoja 1 / 1

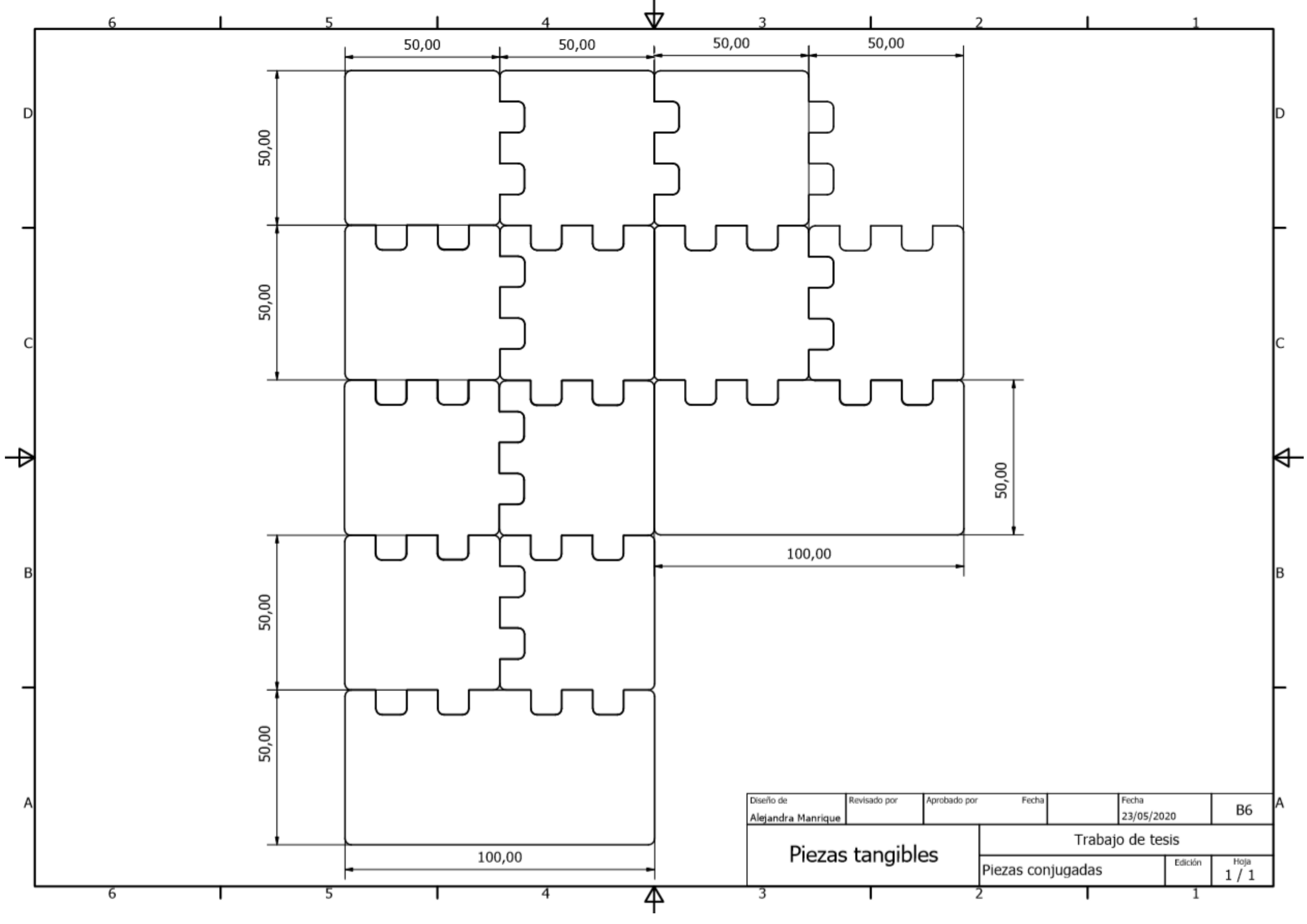






Diseño de	Revisado por	Aprobado por	Fecha	Fecha	B5
Alejandra Manrique				22/05/2020	

Piezas tangibles	Trabajo de tesis		
	Pieza 1-5	Escala	Hoja 1 / 1



Diseño de	Revisado por	Aprobado por	Fecha		Fecha	
Alejandra Manrique					23/05/2020	B6

Piezas tangibles	Trabajo de tesis		
	Piezas conjugadas	Edición	Hoja 1 / 1