PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Estudio del 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

TRABAJO DE INVESTIGACIÓN PARA LA OBTENCIÓN DEL GRADO DE BACHILLER EN CIENCIAS CON MENCIÓN EN INGENIERÍA ELECTRÓNICA

AUTOR

Maria Alejandra Manrique Zarate

ASESOR:

Pablo Cárdenas Cáceres

Lima, agosto, 2020

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 TIC"*. Estas tres competencias impactan directamente en dos de los puntos principales del perfil de egreso de un estudiante de la Educación Básica Regular 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 y 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 muy por debajo de la media mundial, con puntajes muy bajos que lo ubican en el nivel 2 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. El presente trabajo de investigación hace una revisión de las plataformas de programación tangible existentes y los conceptos teóricos necesarios para el diseño de una nueva plataforma que tome en cuenta el Currículo Nacional de Educación Básica.

ÍNDICE

CAPÍTULO 1:

MARCO PROBLEMÁTICO 7

1.1 Motivación 7

1.2 Justificación 9

MARCO DE REFERENCIA 11

2.1 Marco histórico: Estado del arte 11

2.1.1 Algoblock 11

2.1.2 TactusLogic 12

2.1.3 Algorithmic Bricks 12

2.1.4 Proteas 13

2.1.5 P-Cube 14

2.1.6 FYO 14

2.2 Marco Teórico 17

2.2.1 Lenguaje de programación 17

2.2.1.1 Programación imperativa procedural 17

2.2.1.2 Programación funcional 17

2.2.1.3 Programación basada en reglas 18

2.2.2 Entorno de programación 18

2.2.3 Intérprete 18

2.2.3.1 NumPy 18

2.2.3.2 SciPy 19

2.2.3.3 OpenCV-Python 19

2.2.3.4 Servidor Web 19

2.2.3.5 Robots Móviles con ruedas 20

2.2.3.6 Robots móviles con patas 20

2.2.3.7 Robots móviles tipo oruga 20

2.2.3.8 Configuración Ackerman 21
2.2.3.9 Configuración omnidireccional 21
2.2.3.10 Configuración diferencial 22
2.2.3.11 Estrategia de control de desplazamiento y orientación basada en el modelo cinemático del robot diferencial 23
2.2.3.12 Estrategia de control de trayectoria basada en el modelo cinemático del robot diferencial 25
CONCLUSIONES 28
RECOMENDACIONES Y TRABAJOS FUTUROS 29
BIBLIOGRAFÍA 30

ÍNDICE DE FIGURAS

Figura 1.Diagrama de bloques de TactusLogic.....12
Figura 2. Formas equivalentes de la colocación de los bloques tangible.....13
Figura 3. Componentes de la plataforma P-Cube 14
Figura 4. Entorno de programación.....18
Figura 5. Robot móvil con configuración Ackerman 21
Figura 6. Vista superior de una omni-rueda y un robot móvil omnidireccional de 3 ruedas 22
Figura 7. Robot móvil con configuración diferencial 22
Figura 8. Estructura del robot móvil diferencial 23
Figura 9. Representación de entradas y salidas del sistema cinemático.....24
Figura 10. Diferenciación entre un controlador de de.....24
Figura 11. Nueva representación de entradas y salidas del sistema cinemático.....25
Figura 12. Estructura del robot diferencial con el punto P desplazada una distancia a 25
Figura 13. Esquema del controlador del robot móvil diferencial.....27

ÍNDICE DE TABLAS

Tabla 1. Tabla comparativa de las diferentes plataformas de programación tangible16

CAPÍTULO 1
MARCO PROBLEMÁTICO

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].

1.1 Motivación

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].

Actualmente, 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. 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.

1.2 Justificación

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].

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. Y, finalmente, el marco teórico describe las consideraciones necesarias para diseñar una plataforma de programación tangible.

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].

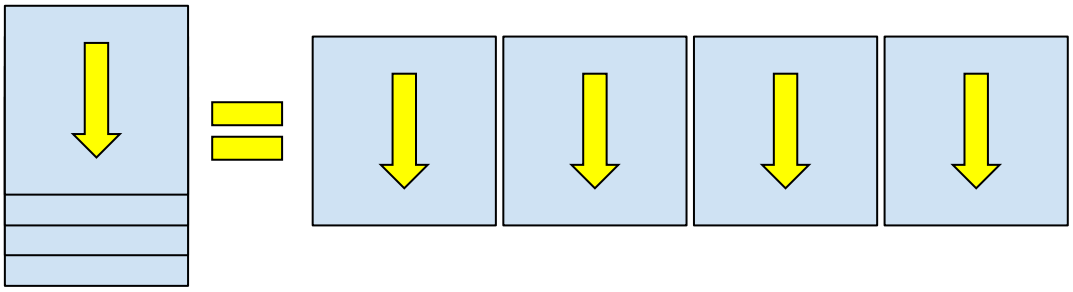


Figura 2. Formas equivalentes de la colocación de los bloques tangibles

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.

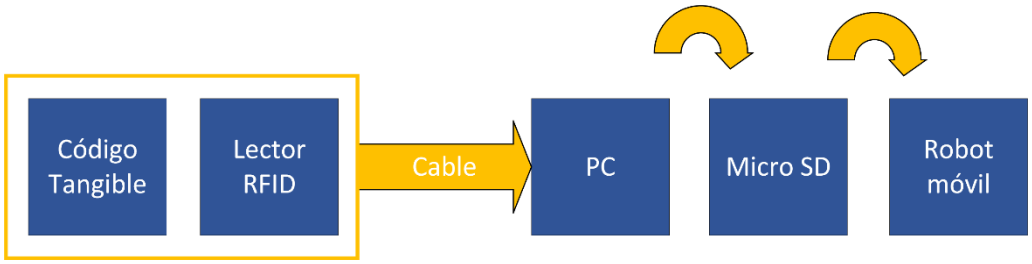


Figura 3. Componentes de la plataforma P-Cube

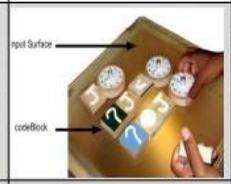
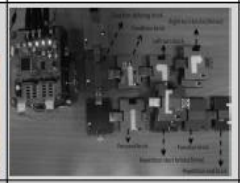
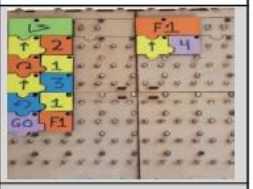
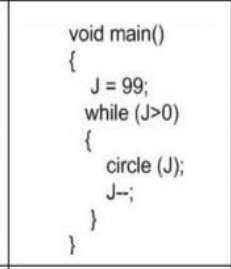
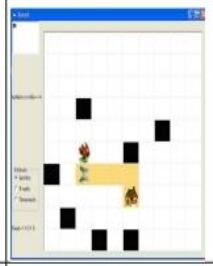
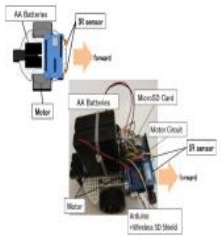
2.1.6 FYO

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**
DESARROLLADORES	H. Suzuki y H. Kato	Andrew Cyrus Smith	Dai-Young Kwon	**T_BUTTERFLY**	**T_PROROB**	Shun Kakehashi et Al.	Pablo Cárdenas
				Theodosios Sapounidis y Stavros Demetriadis			
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.

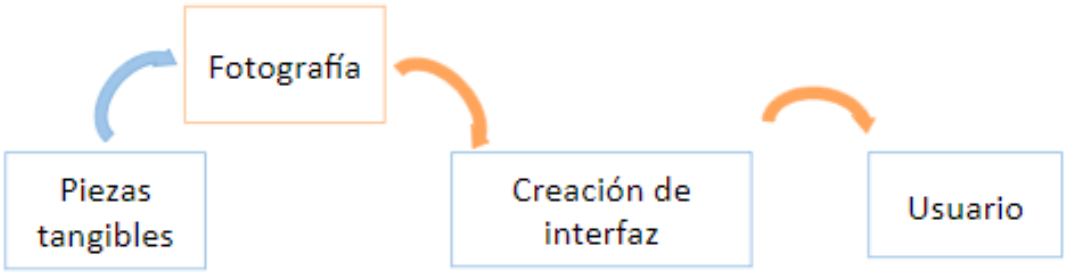


Figura 4. Entorno de programación

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, pues 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.

El estudio plantea las bases para el diseño de 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.

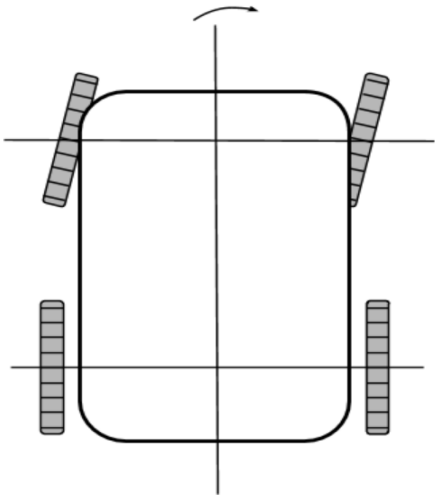


Figura 5. Robot móvil con configuración Ackerman

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].

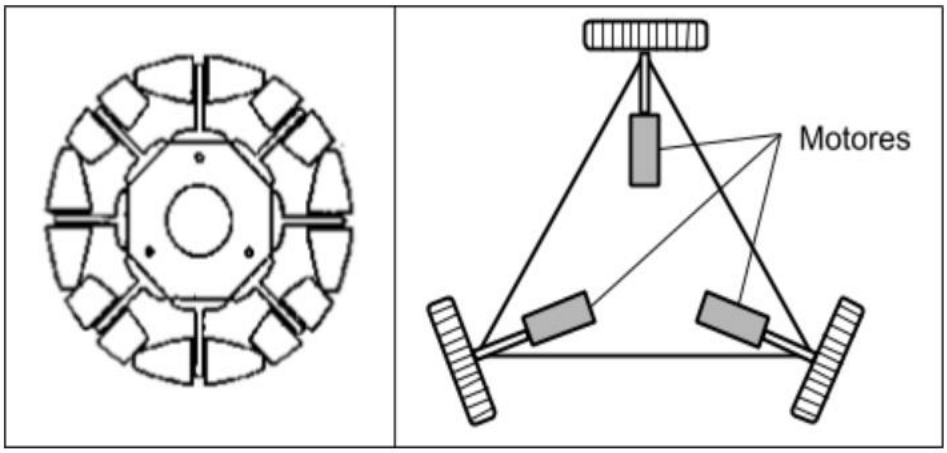


Figura 6. Vista superior de una omni-rueda y un robot móvil omnidireccional de 3 ruedas

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.

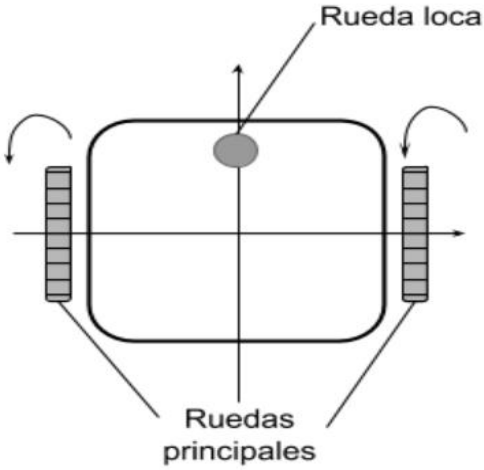


Figura 7. Robot móvil con configuración diferencial

Como se ha visto, cada configuración presenta ventajas y desventajas, pero de acuerdo a los requerimientos del presente trabajo y el análisis de las opciones mencionadas, se escogerá una configuración diferencial. 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.

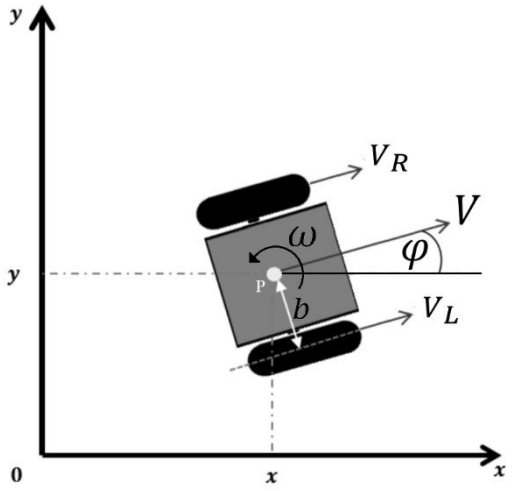


Figura 8. Estructura del robot móvil diferencial

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

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

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.

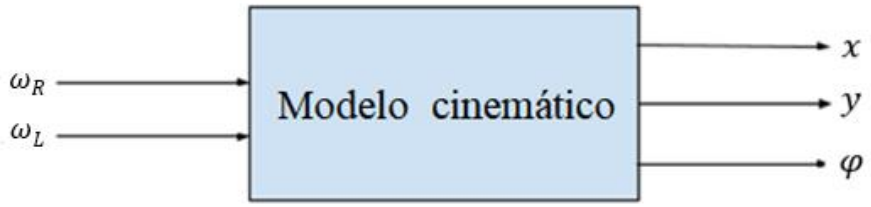


Figura 9. Representación de entradas y salidas del sistema cinemático

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

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.

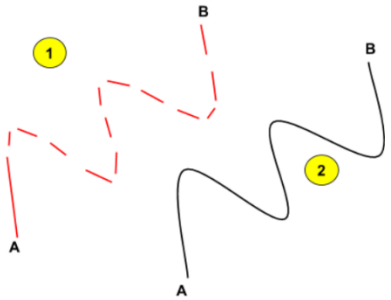


Figura 10. Diferenciación entre un controlador de desplazamiento y uno de trayectoria

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].

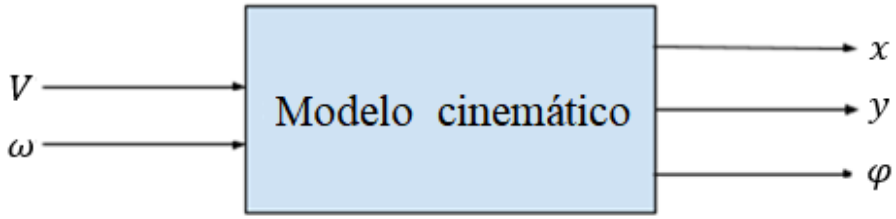


Figura 11. Nueva representación de entradas y salidas del sistema cinemático

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

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

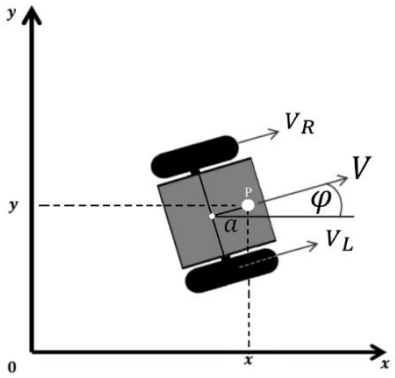


Figura 12. Estructura del robot diferencial con el punto P desplazada una distancia a

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,

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

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:

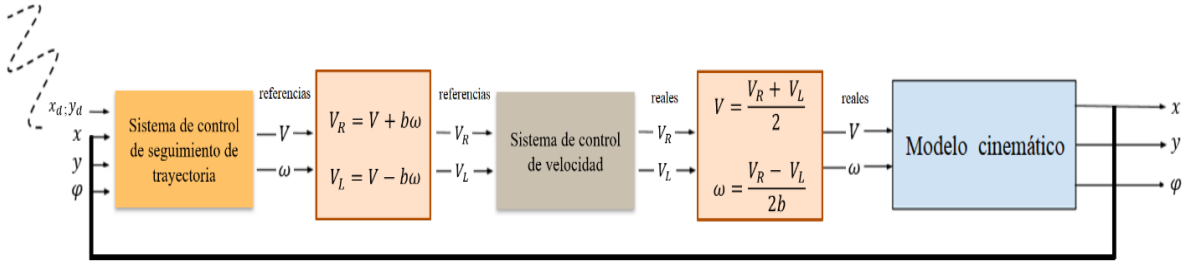


Figura 13. Esquema del controlador del robot móvil diferencial

CONCLUSIONES

Se puede concluir que el lenguaje de programación con un paradigma basado en reglas es el más adecuado para los principiantes en las ciencias computacionales. Lo mencionado previamente se comprueba con la revisión del Marco Histórico, ya que es el paradigma más usado por los autores.

Se puede concluir que el entorno de programación debe permitir virtualizar el código y abaratar costos de producción, por lo que un servidor web y piezas tangibles sin diseño electrónico en su interior como medios de interacción con el usuario logran este objetivo.

En base a la revisión del Marco Teórico, se puede concluir que un intérprete robot móvil con locomoción diferencial es el más adecuado para seguir las trayectorias codificadas por el usuario. Además, este tipo de locomoción simplifica el diseño mecánico.

Es factible concluir que se puede realizar un control de trayectorias basándose en el esquema de la Figura 13. Cabe resaltar que este esquema no realiza un control de orientación, pero esta debe ser censada para que el control de trayectorias funcione correctamente.

RECOMENDACIONES Y TRABAJOS FUTUROS

Se recomienda diseñar un lenguaje de programación que se adecúe a los movimientos previstos para el intérprete y al tipo de control de trayectorias que este realizará. Las instrucciones deben poder ser fácilmente entendidas por el usuario y realizables por el robot móvil.

Se recomienda usar patrones o códigos para reconocer cada pieza tangible y, de esta manera, simplificar el procesamiento de la fotografía del ensamble de las piezas. Además, será necesario plantear requerimientos de imagen, puesto que no todos los dispositivos móviles obtienen fotografías con las mismas características.

Se recomienda plantear una lista de requerimientos para el robot móvil, en cuanto a su costo, material, cinemática, dimensiones, ergonomía, entre otros. Además, se sugiere tomar como referencia de diseño a la norma alemana VDI 2206, la cual contempla diferentes etapas que permiten comparar tecnologías y escoger la opción más adecuada.

Se recomienda usar el software Matlab para simular el sistema de control planteado. Además, se sugiere simular cada módulo de este sistema separándolo del resto y después unir todos los módulos para evitar errores.

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

[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

[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

[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

[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

[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

[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

[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

[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

[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

[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. Availablein: http://ieeexplore.ieee.org.ezproxybib.pucp.edu.pe:2048/stamp/stamp.jsp?tp=&arnumber=8725073&isnumber=8725024

[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

[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

[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

[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

[23] NumFOCUS, 2019. [Online]. Available in: https://numpy.org/

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

[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

[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

[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

[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

[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

[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

[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

[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

[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

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