UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

PLATAFORMA ELECTRÓNICA, A NIVEL DE PROTOTIPO PARA LA APLICACIÓN DE LA PROGRAMACIÓN TANGIBLE EN EDUCACIÓN INICIAL

PRESENTADO POR:

Br. ROGER GROVAS LUNA

PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

ASESOR:

MSC. ING. FERNANDO TAGLE CARBAJAL

CUSCO – PERÚ

2024

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: "PLATAFORMA ELECTRONICA, A NIVEL DE PROTOTIPO PARA LA APLICACION DE LA PROGRAMACIÓN TANGIBLE EN EDUCACIÓN INICIAL"

presentado por: ROGER GROVAS LUNA con DNI Nro.: 41976430 presentado por: ..... con DNI Nro.: ..... para optar el título profesional/grado académico de INGENIERO ELETRÓNICO

.....

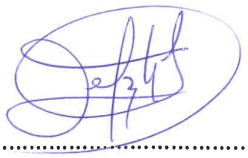
Informo que el trabajo de investigación ha sido sometido a revisión por 02 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 6 %.

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** la primera página del reporte del Sistema Antiplagio.

Cusco, 23 de diciembre de 2024



.....

Firma

Post firma: Fernando Tagle Carbajal

Nro. de DNI: 29585972

ORCID del Asesor: 0000-0003-0481-5277

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: **oid:** 27259:418502970

ROGER GROVAS LUNA

tesis grado.pdf

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:418502970

Fecha de entrega
23 dic 2024, 9:57 a.m. GMT-5

Fecha de descarga
23 dic 2024, 11:59 a.m. GMT-5

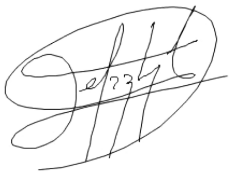
Nombre de archivo
tesis grado.pdf

Tamaño de archivo
4.6 MB

150 Páginas

27,875 Palabras

168,571 Caracteres



6% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 10 palabras)

Fuentes principales

4%	Fuentes de Internet
1%	Publicaciones
4%	Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

Caracteres reemplazados
111 caracteres sospechosos en N.º de páginas
Las letras son intercambiadas por caracteres similares de otro alfabeto.

Texto oculto
951 caracteres sospechosos en N.º de páginas
El texto es alterado para mezclarse con el fondo blanco del documento.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

PRESENTACION

En la actualidad, la tecnología y la programación son fundamentales en nuestra vida cotidiana. Por ello, es esencial que los niños comiencen a aprender conceptos básicos de programación desde una edad temprana, transformando el aprendizaje en un juego y fomentando su curiosidad natural. La programación tangible se presenta como una de las soluciones más innovadoras para enseñar a los niños a programar de manera lúdica, dinámica y práctica. Este enfoque utiliza objetos físicos de programación y robots actuadores, para impartir conocimientos sobre la lógica y mecánica de la programación. Al interactuar físicamente con estos elementos, los niños no solo desarrollan su creatividad, sino que también mejoran sus habilidades para resolver problemas, lo que les proporciona una comprensión más concreta del funcionamiento de la tecnología. La presente tesis se centra en el desarrollo de un prototipo electrónico de programación tangible diseñado para potenciar las habilidades cognitivas y sociales de los niños.

Se realizó un análisis de diversos sistemas de programación tangible con el objetivo de crear un prototipo electrónico adaptado a estudiantes de educación inicial en entornos educativos dinámicos y lúdicos. Al concluir la investigación, se realizaron pruebas piloto para evaluar la facilidad de uso como una herramienta alternativa y de apoyo en la enseñanza, cuyo diseño esta específicamente para ayudar a los niños de educación inicial a aprender programación de manera más accesible y efectiva.

Varios estudios han demostrado que los niños que interactúan con sistemas de programación tangible obtienen puntajes significativamente más altos en pruebas de pensamiento lógico y resolución de problemas. En resumen, la programación tangible se presenta como una alternativa efectiva, dinámica y entretenida para introducir a los niños en el mundo de la programación y la tecnología.

DEDICATORIA

Este trabajo está dedicado al Niño Jesús de Huarccoy, al Sr. de Qoyllurity y al Padre Urraca, por derramar su gracia a mi persona y permitir que concluya en esta etapa de mi vida.

A mi Mamita, Nemesia, por haber confiado siempre en mi a pesar de que muchas veces le fallé.

A mi Papá Marcelo y hermanos que siempre estuvieron orgullosos de mí.

A la persona que me ayudo en los momentos difíciles de mi vida y hoy forma parte de mi vida, mi esposa Luz y mi hijo Stefanno.

A las personas que me apoyaron y me acompañaron en las etapas más críticas de mi vida.

A todos mis amigos de código y facultad, que siempre los llevare presente.

Roger GROVAS LUNA.

AGRADECIMIENTO

A mi asesor el Ing. Fernando, TAGLE CARBAJAL, por toda la paciencia y el apoyo que tuvo durante el desarrollo de este trabajo.

Al Ing. Luis, JIMENEZ TRONCOSO por la disponibilidad y deferencia que tuvo con el aporte de sus consejos durante el desarrollo del trabajo de tesis.

Al Ing. Alex Jhon, QUISPE MESCO por su apoyo y sus palabras de superación.

A mi amigo George W. ORTIZ, por su motivación para culminar esta investigación.

A mi amigo Franco ROSA, por su paciencia y tiempo durante el proceso de la investigación.

Finalmente, a la plana docente de la Facultad de Ingeniería Electrónica de la UNSAAC, por impartir sus conocimientos durante el desarrollo de la carrera profesional.

RESUMEN

El Capítulo II establece el marco de referencia, analizando plataformas existentes y desarrollando el marco teórico sobre lenguaje, entorno de programación e intérprete. Se propone una investigación aplicada descriptiva para evaluar la funcionabilidad y su impacto de usabilidad en preescolares. La metodología incluye una revisión de literatura y casos de estudio para fundamentar el diseño de la plataforma. El Capítulo III detalla el diseño de la plataforma, partiendo de una lista de exigencias que define sus características. Se desarrolla un lenguaje basado en reglas para niños, implementado mediante piezas físicas, un tablero y un intérprete móvil. El Capítulo IV aborda la implementación, cubriendo el diseño electrónico, mecánico y software. Se seleccionaron componentes específicos, se elaboró el diagrama esquemático y se desarrolló software para el microcontrolador. El Capítulo V evalúa la plataforma a través de un estudio descriptivo. En cuanto a la funcionalidad, se logró un 80% de éxito en la detección, con 4 fallos en 20 interacciones. Asimismo, se realizó una prueba de usabilidad con estudiantes y tutores de 50499 Oropeza, utilizando la plataforma de programación tangible diseñada, con base en investigaciones previas de Smith, Ramos y Gallardo. Los resultados mostraron que el 85% de los estudiantes encontró la plataforma fácil de usar y el 90% quedó satisfecho con el resultado final. La interacción fue mayormente intuitiva, con un promedio de 5 minutos por sesión, y se registró un aumento del 30% en la participación. Sin embargo, la disponibilidad limitada de prototipos resultó en una interacción deficiente, con un 45 % de éxito y un 55 % de fracaso. Los resultados cuantitativos indican la necesidad de realizar mejoras en el diseño con el fin de optimizar la experiencia del usuario en el uso de la plataforma tangible, así como su viabilidad como herramienta de apoyo pedagógico.

Palabras Claves:

Programación Tangible, bloques programables, entorno de programación, usabilidad.

ABSTRACT

Chapter II establishes the frame of reference, analyzing existing platforms and developing the theoretical framework on language, programming environment, and interpreter. A descriptive applied research is proposed to evaluate functionality and its usability impact on preschoolers. The methodology includes a literature review and case studies to support the platform design. Chapter III details the platform design, starting from a list of requirements that defines its characteristics. A rule-based language for children is developed, implemented through physical pieces, a board, and a mobile interpreter. Chapter IV addresses implementation, covering electronic, mechanical, and software design. Specific components were selected, the schematic diagram was created, and software was developed for the microcontroller. Chapter V evaluates the platform through a descriptive study. Regarding functionality, an 80% success rate was achieved in detection, with 4 failures in 20 interactions. Additionally, a usability test was conducted with students and tutors from 50499 Oropeza, using the designed tangible programming platform, based on previous research by Smith, Ramos, and Gallardo. Results showed that 85% of students found the platform easy to use, and 90% were satisfied with the final result. Interaction was mostly intuitive, with an average of 5 minutes per session, and a 30% increase in participation was recorded. However, limited prototype availability resulted in poor interaction, with a 45% success rate and 55% failure rate. Quantitative results indicate the need for design improvements to optimize user experience in using the tangible platform, as well as its viability as a pedagogical support tool.

Keywords:

Tangible Programming, programmable blocks, programming environment, usability.

INTRODUCCION

El mundo en el que vivimos está experimentando un avance tecnológico significativo en diversas áreas, y la educación no es una excepción. En los últimos años, las Tecnologías de Información y Comunicación (TIC) se han integrado como herramientas metodológicas y educativas esenciales. Una de estas herramientas es la programación tangible, que ofrece una forma innovadora de introducir a los niños en el ámbito de la tecnología y la programación. Utilizando objetos físicos y manipulables, crea un entorno más lúdico, creativo e interactivo, facilitando la adquisición de habilidades como el pensamiento computacional, especialmente en estudiantes de Educación Básica Regular (EBR) [1]

La implementación de la programación tangible se alinea con las TIC en la educación de nuestro país y podría desempeñar un papel crucial en el cumplimiento del Currículo Nacional de Educación Básica. Esto permitiría el desarrollo de competencias clave dentro del marco normativo del Ministerio de Educación (MINEDU), tales como: Competencia 22 (diseña y construye soluciones tecnológicas para resolver problemas en su entorno), Competencia 23 (resuelve problemas de cantidad) y Competencia 28 (se desenvuelve en entornos virtuales generados por las TIC) [2].

A diferencia del uso tradicional de pantallas digitales y dispositivos electrónicos, la programación tangible emplea objetos físicos, como bloques o piezas geométricas, que los niños pueden manipular y ordenar para crear programas simples. Esto les permite comprender los principios básicos de la programación. Este planteamiento representa una metodología innovadora que busca acercar el mundo de la programación a los niños del nivel inicial de manera dinámica y entretenida, complementando así la enseñanza tradicional y fomentando habilidades como la creatividad, el pensamiento crítico y la resolución de problemas.

En este trabajo se ha desarrollado un prototipo electrónico de programación tangible que podría funcionar como una herramienta alternativa para apoyar el aprendizaje de los niños. Este prototipo electrónico utiliza objetos físicos para definir trayectorias, permitiendo que un robot actuador ejecute instrucciones de forma inalámbrica. Esto facilita que los niños comprendan la programación de manera sencilla y directa. Este enfoque no muy utilizado en educación inicial hace que el aprendizaje sea más accesible y atractivo, ayudando a los estudiantes a desarrollar competencias clave en un entorno educativo dinámico. Además de fomentar habilidades técnicas, esta metodología promueve habilidades sociales esenciales para el desarrollo integral de los niños, ya que les permite colaborar y resolver problemas juntos durante su proceso educativo. También contribuirá al perfil de egreso del estudiante en Educación Básica Regular, alineándose con los aprendizajes esperados que fomentan el uso responsable y reflexivo de las Tecnologías de Información y Comunicación (TIC) para interactuar con la información y gestionar su comunicación y aprendizaje [2].

Finalmente, se llevó a cabo simulaciones con la plataforma electrónica de programación tangible para validar su usabilidad. Se observó que este dispositivo, de bajo costo, es capaz de ejecutar trayectorias programadas y definidas mediante el uso de piezas físicas ordenadas por los niños. Posteriormente, se espera que esta plataforma electrónica sea mejorada y pueda así aplicarse como herramienta de aprendizaje alternativa en la educación inicial, contribuyendo al desarrollo integral de los niños en un entorno educativo dinámico.

INDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

CAPÍTULO I: ASPECTOS GENERALES

1.1. Título

"Plataforma Electrónica, a nivel de prototipo para la aplicación de la programación tangible en educación inicial".

1.2. Planteamiento del Problema

1.2.1. Problemática

El entorno en la que vivimos hoy en día va desarrollando la tecnología en un ritmo acelerado, los experimentos del cambio en la tecnología actual, guiado por los adelantos en las tecnologías digitales, se encuentran en todas partes. Simplemente con ver en los aparatos móviles y los sistemas informáticos cada vez más avanzados. Se afirma que la gran mayoría de los sectores son beneficiarios con las herramientas tecnológicas, el Sector Educación no es la excepción. En los últimos años, la Educación en nuestro país se está complementando mediante las herramientas Tecnológicas de Información y Comunicación (TIC) que son herramientas metodológicas aplicadas a los educandos [1]. De las tantas herramientas que actualmente existen con las Tecnologías de Información y Comunicación (TIC) resaltamos la programación, la cual nos ayuda con el desarrollo de habilidades del pensamiento computacional sobre todo en niños que se encuentran dentro de la etapa preoperacional (4 a 6 años de edad) según la clasificación en el Desarrollo Cognitivo de Jean William Fritz Piaget, por consiguiente, la programación podría ser aplicada a estudiantes a partir de los 5 años que cursan en nivel inicial de la Educación Básica Regular de nuestra región [2], [3].

En cuanto a la programación se podría decir que está relacionada directamente con el pensamiento computacional y uno de los conceptos más conocidos es la que formuló la primera persona en mecer el término, la ingeniera de la Nacionalidad de Estados Unidos Jeannette Wing

en que se define como “el proceso mental para formular un problema y sus soluciones de manera que se representan en una forma en que pueden ser llevadas a un agente de procesamiento de información”, y en una definición más adecuada seria “la destreza y capacidad de solucionar dificultades utilizando los fundamentos de la programación” [4].

En el Perú la Educación Básica Regular, por intermedio del Ministerio de Educación, emite un documento normativo denominado Currículo Nacional que es el instrumento marco de la política educativa en la educación básica regular y contiene el desarrollo de conocimientos que se espera que la población estudiantil logre en el transcurso de su formación básica, acorde a los fines y principios de la educación de nuestro país, objetivos de la educación básica y el Proyecto Educativo Nacional. Este instrumento establece el Perfil de egresado de la Educación Básica Regular, las Competencias Nacionales y sus evoluciones desde el principio hasta la conclusión de la educación básica regular, así como los niveles esperados por ciclo, nivel y modalidades [2].

Dentro de nuestro país, en la actualidad la enseñanza de la programación puede llegar a ser un instrumento muy importante en el cumplimiento del Currículo Nacional de Educación Básica Regular, pues permite el cumplimiento de 03 de la 29 Competencias Nacionales, así como se observa en la figura 1, y entre ellas están las siguientes:

- ✓ Diseña y construye soluciones tecnológicas para resolver problemas de su entorno.
- ✓ Resuelve problemas de cantidad y la numero.
- ✓ Se desenvuelve en los entornos virtuales generados por las TIC.

Figura 1

Perfil de egreso de estudiantes de Educación Básica Regular - EBR

Nota: Adaptado de *Curriculo Nacional de Educacion Basica*, por MINEDU, 2016.

Estas Competencias Nacionales son; la competencia 22, la competencia 23 y la competencia 28 respectivamente, las cuales comprenden dos principales objetivos del egresado de Educación Básica Regular que fomenta las instancias del Ministerio de Educación.

La programación podría incidir en la mejora del desarrollo del pensamiento computacional; sin embargo, la única vez que se promovió en el Perú fue con el programa, una portátil por niño/a u OLPC ("One Laptop per child") [5], que no resultó debido a los altos costos en su mantenimiento y complejidad de las herramientas que se usaba, y ante ese fracaso surge la

programación tangible como una opción y alternativa diferente para la enseñanza de la programación, ya que disminuye la edad para su uso, facilita y mejora la enseñanza y no es necesario una computadora por usuario, por lo que disminuye los costos.

El trabajo que se propone hace una revisión de las diferentes plataformas de programación tangible ya existentes con la finalidad de desarrollar e implementar *una Plataforma Electrónica, a nivel de prototipo para la aplicación de la programación tangible en educación inicial*, acorde a la realidad de nuestra región. La plataforma seleccionada estará compuesta por piezas de programación tangibles, a través de las cuales se define una trayectoria, y un intérprete actuador electrónico que virtualiza e interpreta el código creado a través de las piezas tangibles para finalmente ejecutar la trayectoria codificada.

1.2.2. Problema General

Por lo tanto, de lo anterior:

En la actualidad, el Currículo Nacional de Educación Básica Regular (EBR) no contempla el uso de una herramienta que les sea de gran ayuda a los profesores introducir capacidades y conocimientos que desarrollen el pensamiento computacional desde temprana edad, especialmente en los niveles de inicial y primaria, etapas fundamentales donde los niños presentan mayor capacidad de aprendizaje y desarrollo cognitivo durante sus primeros ocho años de vida [6].

A pesar de la amplia disponibilidad de componentes y tarjetas electrónicas en el mercado de la región, existe una notable ausencia en las instituciones educativas de nivel inicial la utilización del uso de plataformas electrónicas de programación tangible. Estas plataformas electrónicas, que permiten a los niños interactuar físicamente con elementos de programación sin necesidad de pantallas digitales, representan una herramienta pedagógica innovadora para desarrollar el pensamiento computacional desde temprana edad.

La carencia de estas plataformas electrónicas de programación tangible afecta directamente a los docentes, quienes no cuentan con recursos pedagógicos concretos para introducir conceptos básicos de programación y así desarrollar el pensamiento computacional de manera lúdica y adaptada a la edad de los estudiantes.

"Esta realidad hace evidente la necesidad de explorar y desarrollar un prototipo de plataforma electrónica de programación tangible, es decir, que los niños puedan tocar y manipular directamente. Esta herramienta tiene por objetivo fomentar el desarrollo del pensamiento computacional en estudiantes de nivel inicial, justamente durante la etapa donde su capacidad de aprendizaje está en su punto más alto en comparación con las otras etapas del desarrollo infantil".

1.2.3. Problemas Específicos

- ✓ **Inexistencia de Plataformas Adaptadas:** No hay suficientes plataformas de programación tangible disponibles que sean adecuadas para el contexto educativo de la región que faciliten el aprendizaje a través de métodos más accesibles y prácticos.
- ✓ **Desconexión del uso de la TIC:** La enseñanza de la programación no está alineada adecuadamente con las competencias nacionales establecidas en el Currículo Nacional de Educación Básica Regular, lo que limita su integración en el sistema educativo.
- ✓ **Desconocimiento sobre Pensamiento Computacional:** Existe una falta de comprensión sobre cómo la programación puede contribuir al desarrollo del pensamiento computacional en los estudiantes de etapa preescolar.
- ✓ **Complejidad de Herramientas:** Las herramientas tecnológicas actuales son complejas y no están diseñadas para ser utilizadas por niños del nivel inicial, lo que dificulta su aprendizaje.

- ✓ **Accesibilidad y Costo:** La enseñanza de la programación tradicional requiere recursos costosos, como computadoras, lo que limita su implementación en el aula.
- ✓ **Falta de Capacitación Docente:** Los docentes pueden no estar adecuadamente capacitados para enseñar programación a niños tan jóvenes, lo que afecta la calidad de la enseñanza.

Estos problemas específicos contribuyen un reto para implementar eficientemente el prototipo electrónico y más aún si se trata de la programación tangible en la educación básica regular, directamente en nuestro caso para educación inicial, limitando las oportunidades para desarrollar habilidades esenciales en los niños.

1.2.4. Justificación

A lo largo del mundo, se han desarrollado numerosas iniciativas para introducir la programación en el currículo escolar con el objetivo de fomentar el desarrollo del pensamiento computacional en los estudiantes de la Educación Básica Regular. Un ejemplo notable es la implementación de Scratch, una plataforma de programación orientada a niños, en un grupo de escolares de entre 8 y 15 años en las zonas más vulnerables de Guayaquil. Esta iniciativa demostró que la enseñanza de programación puede estimular significativamente el desarrollo de habilidades cognitivas, creativas y el pensamiento lógico-matemático, dado que el 60.55% de los participantes mostró una mejora notable en su rendimiento académico [7].

En Italia, los ingenieros de la Universidad Politécnica delle Marche llevaron a cabo el proyecto piloto “Robotics In School” en el Institute Comprensivo Largo Cocconi de Roma. Este proyecto tuvo como objetivo enseñar programación a través del kit robótico programable Lego Mindstorms a niños de educación primaria. Los resultados mostraron que los estudiantes

adquirieron nuevas habilidades y mejoraron las existentes, como la resolución de problemas, el trabajo en equipo y el uso eficiente de tecnologías disponibles [8].

En Perú, el Ministerio de Educación (MINEDU) implementó en 2007 el programa “Una laptop por niño” en colaboración con la ONG "One Laptop per Child" (OLPC). Se distribuyeron más de un millón de laptops XO a diversas escuelas a nivel nacional. Posteriormente, en 2011, se entregaron 92,000 kits de robótica “WeDo” de Lego con el objetivo de fomentar la enseñanza de la programación en escuelas públicas. Sin embargo, varios estudios concluyeron que el programa no alcanzó sus metas debido a la insuficiente preparación técnica de los docentes y la complejidad percibida en la enseñanza de programación con estos kits. Otro factor fue la falta de producción local de los kits, lo que incrementó los costos y tiempos de mantenimiento [9], [10].

El prototipo de la plataforma electrónica de programación tangible que se propone consta de los siguientes componentes: el lenguaje de programación (piezas tangibles), que permite crear un código; tablero de programación, que es donde se coloca las piezas tangibles; y el intérprete, que ejecuta las instrucciones. Este tipo de programación, denominada "programación tangible" (TLP, por sus siglas en inglés), fue conceptualizada por Suzuki y Kato en 1993, quienes demostraron la interacción con un intérprete virtual mediante la manipulación de objetos físicos [11]. La programación tangible ofrece varias ventajas educativas: permite manipular el código con las manos sin necesidad de una computadora adicional, favoreciendo así la rápida introducción de los niños al mundo de la programación. También facilita dinámicas grupales y cooperativas, optimiza recursos al no requerir una computadora por cada usuario y mejora la capacidad de autocorrección de los estudiantes al visualizar el código de manera física [11], [12].

1.3. Objetivos

1.3.1. Objetivo General

Diseñar una plataforma electrónica e implementar un prototipo para la aplicación de la programación tangible en la educación inicial.

1.3.2. Objetivos Específicos

- ✓ Diseñar un lenguaje de programación tangible dirigido a estudiantes del nivel inicial.
- ✓ Diseñar un entorno para la programación tangible, como interface de usuario para enviar instrucciones al interprete actuador electrónico (robot móvil).
- ✓ Diseñar el sistema electrónico que realice el procesamiento de la programación tangible.
- ✓ Diseñar un intérprete actuador electrónico tipo robot móvil, que ejecute las instrucciones programadas mediante el código tangible, para que se finalice con un control de trayectoria.
- ✓ Implementar el prototipo de la plataforma electrónica y realizar pruebas piloto, aplicada como mínimo en 01 institución educativa del nivel inicial.
- ✓ Aplicar encuesta de usabilidad del prototipo electrónico.

1.4. Alcances y Limitaciones

1.4.1. Alcances

1.4.1.1. Alcances Técnicos.

- ✓ **Desarrollo de hardware**
  - o Diseño e implementación de bloques físicos programables.
  - o Sistema de conexión entre bloques y reed switch mediante interfaces magnéticas.
  - o Sistema de alimentación y gestión de energía

- Prototipo funcional con capacidad para 06 bloques simultáneos.
- Autonomía de batería mínima de 3 horas de uso continuo.

✓ **Desarrollo de software**

- Interfaz de interpretación de bloques físicos.
- Interpretación en tiempo real de la secuencia de bloques.
- Sistema de traducción de configuraciones físicas a código.
- Soporte para estructuras básicas de programación como las secuencias lineales.

✓ **Interacción**

- Reconocimiento inmediato de conexiones.
- Retroalimentación visual mediante leds.
- Retroalimentación sonora para errores.
- Detección visual de orientación de bloques tangibles.

1.4.1.2. Alcances de implementación.

✓ **Desarrollo**

- Prototipo funcional completo.
- Pruebas de funcionamiento documentadas.

✓ **Validación**

- Pruebas de usabilidad con el grupo usuario.
- Verificación de funcionalidad.

1.4.2. Limitaciones

1.4.2.1. Limitaciones Técnicas

✓ **Hardware**

- Limitado a conexiones en superficie plana.

- Sin resistencia al agua o humedad.
- Sin capacidad de actualización remota.

✓ **Software**

- No se incluirán estructuras de programación avanzadas.
- Sin soporte para tipos de datos complejos.
- Sin capacidad de depuración paso a paso.
- Limitado a un solo lenguaje de programación.
- Sin interfaz gráfica avanzada.

1.4.2.2. Limitaciones Metodológicas.

✓ **Investigación**

- Limitada a 20 estudiantes.
- Sin grupo de control
- Limitada a una institución educativa del nivel inicial.
- Solamente se considera una variable única.
- La falta de formación y preparación técnica de los docentes puede limitar la eficacia del prototipo, se necesita capacitación adecuada para utilizar esta herramienta de manera efectiva.
- La evaluación de los beneficios a largo plazo del uso del prototipo puede ser compleja y requerir un seguimiento continuo, el cual no es parte de alguno de los objetivos de la presente investigación.

1.4.2.3. Limitaciones de Implementación.

✓ **Desarrollo**

- Prototipo limitado a ambiente controlado.

- Sin certificaciones ni validaciones por el MINEDU.
- Sin producción en masa.

✓ **Recursos**

- Presupuesto limitado para componentes.
- Herramientas de fabricación básicas.

1.4.3. Consideraciones Adicionales

✓ **Escalabilidad**

- Posibilidad de agregar nuevos bloques en futuras versiones.
- Arquitectura preparada para actualizaciones de firmware.
- Diseño modular para facilitar mejoras.

✓ **Mantenimiento**

- Componentes reemplazables individualmente.
- Calibración manual requerida periódicamente.
- Limpieza regular necesaria para óptimo funcionamiento.

✓ **Seguridad**

- Protecciones básicas contra cortocircuitos.
- Materiales no tóxicos.
- Bordes redondeados para seguridad.
- Voltajes de operación seguros.

CAPÍTULO II: MARCO DE REFERENCIA

2.1. Marco de Referencia

En esta etapa se desarrollará en 3 marcos: marco histórico, marco teórico y 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, y finalmente, el marco metodológico expone la estrategia escogida para la realización de la tesis.

2.1.1. 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.

2.1.1.1. 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 educación 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.

2.1.1.2. TactusLogic. Es una plataforma de programación, creada por Heinrich Springhorn, Andrew Cyrus Smith, Ireygan Weber, Steven Bruce Mulligan, 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 2. 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 2

Diagrama de bloques de TactusLogic

***Nota*:** Etapas de TactusLogic: Programación utilizando objetos físicos.

2.1.1.3. Algorithmic Bricks. Es una herramienta para introducir a novatos al mundo de la programación a través de TPL (Tangible Programing Languages) y fue creada por Han Sung Kim, Dai Young Kwon, Won Gyu Lee y Jae Kwoun Shim. La investigación realizada por Kwon detalla que una plataforma de programación tangible debe tener cinco características, la primera es la fácil detección de errores a través de la depuración. La segunda es que deben existir

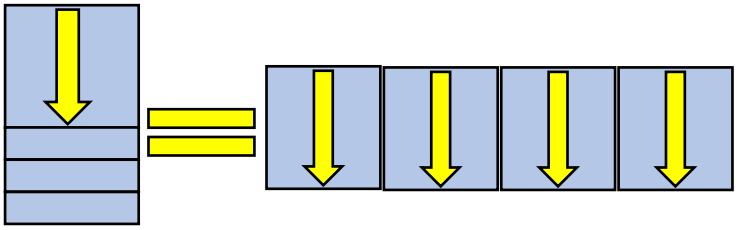
instrucciones concretas, la tercera indica que los comandos deben ser sencillos de asimilar, la cuarta hace referencia a una implementación cooperativa o grupal y la última de estas características es que debe ser de bajo costo para que pueda ser fácilmente adquirida [12].

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 3. 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.

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 3

Equivalencia en las formas para la colocación de los bloques tangibles



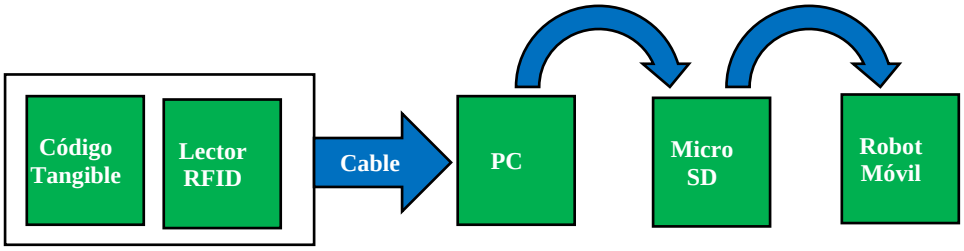
Nota: Colocación de bloques tangibles apilados uno sobre otros o colocados horizontalmente.

2.1.1.4. 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 NAT 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 NAT), por lo que no es accesible para todos los interesados en adquirir el kit.

2.1.1.5. P-Cube. Fue desarrollado por Tatsuo Motoyoshi, Shun Kakehashi, Ken´ichi Koyanagi, Hiroshi Kawakami y Toru Ohshima, 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 4.

Figura 4

Componentes de la programación P-Cube



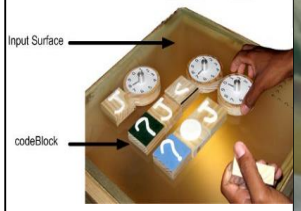
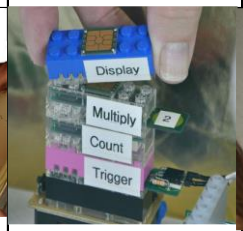
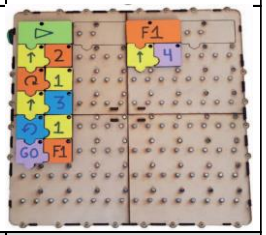
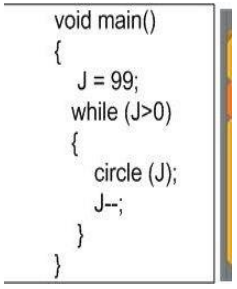
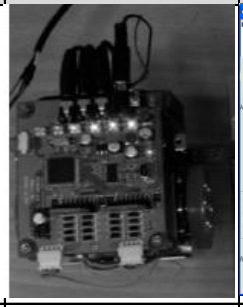
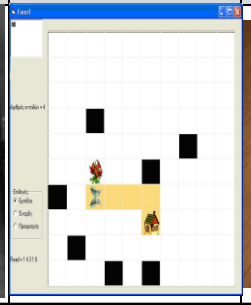
Nota: Etapas de P-Cube: Programación utilizando objetos físicos con etiquetas de radiofrecuencia.

2.1.1.6. FYO (Follow Your Objective). 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 sí 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.

Diversos desarrolladores han utilizado la programación tangible como herramienta principal para enseñar a niños sin experiencia previa. Las plataformas existentes presentan diferencias en el intérprete, sintaxis del lenguaje, entorno de programación y características principales, aunque estas plataformas reducen la barrera etaria para la enseñanza de programación, ninguna consideró los currículos nacionales de educación ni la enseñanza simultánea a un salón completo, resultando ser soluciones costosas y poco adaptables al contexto educativo.

Figura 5

Comparación de diferentes plataformas de programación tangible

	PLATAFORMAS						
CARACTERÍSTICAS	**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 Stayros Demetriadis			
SINTAXIS DEL LENGUAJE DE PROGRAMACIÓN	BLOQUES COMPUESTOS	BLOQUE DE COMANDOS Y PARÁMETROS	BLOQUES COMPUESTOS	BLOQUES COMPUESTOS	BLOQUES COMPUESTOS	BLOQUES COMPUESTOS	BLOQUE 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+

2.1.2. Marco Teórico

En esta sección se abarcará los estudios necesarios para diseñar una plataforma electrónica de programación tangible, el cual incluye un lenguaje de programación, un entorno de programación y el intérprete actuador electrónico.

2.1.2.1. Lenguaje de Programación. Durante el diseño y para el correcto funcionamiento se realizará la creación de un nuevo lenguaje de programación con propósito educativo que debe tomar en cuenta el análisis de los principios 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 una muestra didáctica para principiantes en las ciencias computacionales.

2.1.2.1.1. *Programación Imperativa Procedural.* Este principio 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].

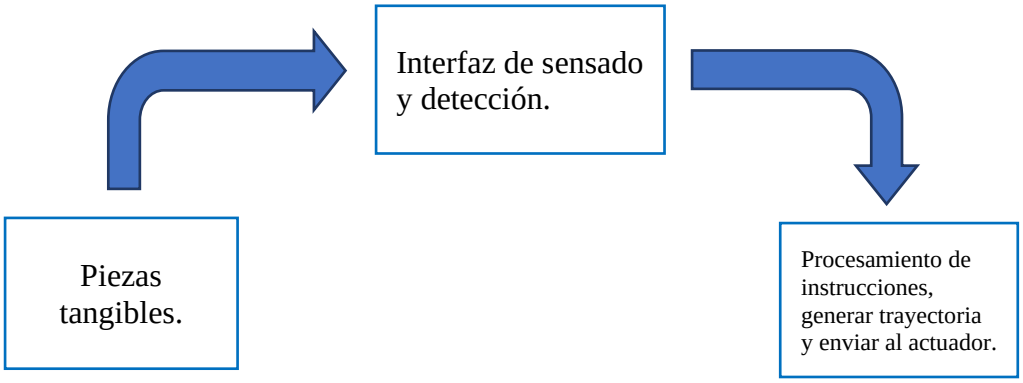
2.1.2.1.2. *Programación Funcional.* El principio funcional está orientado a la construcción de funciones basadas en el cálculo matemático y cuya característica principal, a diferencia de la programación imperativa procedural, es que se suprimen los tipos de datos y utiliza como herramienta a la recursividad. 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].

2.1.2.1.3. *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 sí 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.

2.1.2.2. Entorno de Programación. En cuanto al entorno de programación, esta abarcará las piezas tangibles que contendrán el lenguaje de programación, la interfaz de sensado utiliza la electrónica analógica mediante Reed Switch para la detección las piezas tangibles y finalmente mediante la conversión analogía digital permitirá la compilación del código y enviar al actuador, consecuentemente interactuando de manera directa con el usuario, ver Figura 6.

Figura 6

Entorno de programación



2.1.2.3. Intérprete Actuador Electrónico. El diseño electrónico en el interior del intérprete se encarga de recepcionar y procesar las instrucciones compiladas previamente enviadas por el entorno de programación a partir del ordenamiento de las piezas tangibles para finalmente realizar una trayectoria definida por las mismas.

En la figura 5 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 [22], así como características zoomórficas o antropomórficas, en ambos casos con presencia de extremidades, aunque estas solo sean simbólicas [23].

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 [24].

2.1.2.3.1. ***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 [25], [26] 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, sí es que han sido probados en un terreno específico, y luego este es cambiado por uno con características diferentes.

2.1.2.3.2. ***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 zoomórtficas. Se han desarrollado muchos robots de este tipo alrededor del mundo [27], [28] y la

principal dificultad que han encontrado los creadores ha sido el diseño mecánico debido a la cantidad de movimientos o grados de libertad que requiere este tipo de movimiento traslacional.

2.1.2.3.3. ***Robots Móviles Tipo Oruga.***

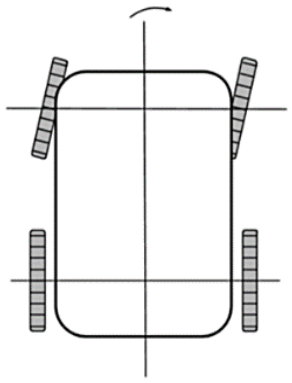
Los sistemas de robots móviles con orugas ofrecen mayor maniobrabilidad en terrenos abruptos y simplicidad mecánica en comparación con robots con patas, utilizando únicamente actuadores de tracción [24]. Sin embargo, presentan la desventaja de un alto consumo de potencia por rozamiento, lo que requiere evaluar su necesidad según la aplicación específica.

La investigación desarrollará un intérprete robot con locomoción basada en ruedas, para lo cual se explorarán tres configuraciones principales: Ackerman, omnidireccional y diferencial, cada una con características distintas para su potencial implementación.

2.1.2.3.4. ***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 7. La principal desventaja de estos robots móviles es que presentan restricciones no holónomas; es decir, la cantidad de movimientos o grados de libertad total no es igual al número de grados manipulable o controlable.

Figura 7

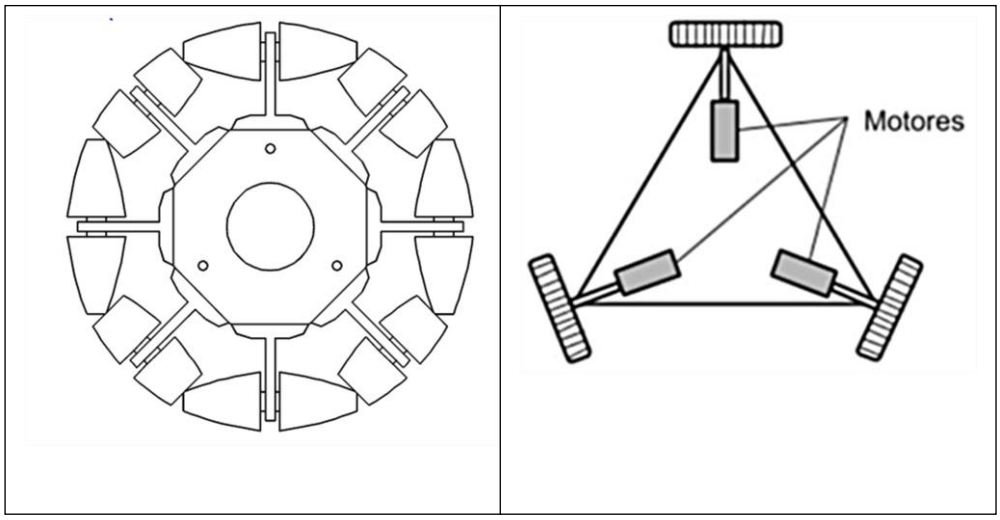
Robot móvil con configuración Ackerman



2.1.2.3.5. ***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 omniruedas (ruedas con discos perpendiculares a la dirección de giro), que le otorgan un movimiento flexible de alta precisión, ver Figura 8. 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 [29].

Figura 8

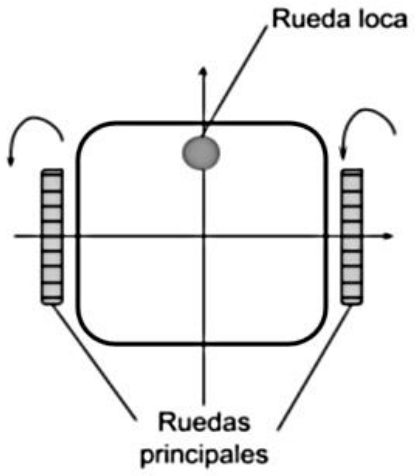
Vista de planta de una rueda omnidireccional y un robot movil de 3 ruedas



2.1.2.3.6. ***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 que se mantenga estable, así como se observa en la Figura 9. Asimismo, presenta dos grados de libertad y ambos motores existentes se encuentran alineados en un mismo eje [30]. La posición y la velocidad del robot se pueden ser controlables, por lo que la orientación del vehículo es una función del desplazamiento de dos de sus llantas.

Figura 9

Robot móvil mediante la configuración diferencial

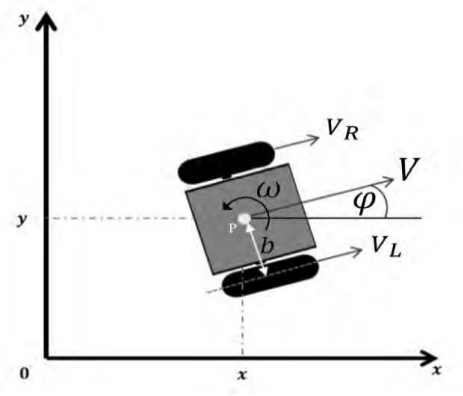


Como se ha visto, cada configuración presenta ventajas y desventajas, pero de acuerdo a los requerimientos de la tesis y el análisis de las opciones mencionadas, se escogerá una configuración diferencial, ya que es la de menor complejidad en el diseño mecánico, garantiza el movimiento en línea recta y presenta el mejor performance para realizar un control de seguimiento de trayectoria [30]. 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. En los objetivos específicos presentados en el capítulo anterior 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.

2.1.2.3.7. ***Estrategia Mediante Control de Desplazamiento y Orientación en Base al Modelo Cinemático del Robot en Configuración Diferencial.*** Esta estrategia ha sido diseñada específicamente para el 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, la Figura 10, se observa la parte estructural del robot con respecto a un punto P ubicado en el punto medio del eje entre las dos ruedas.

Figura 10

Estructura del robot móvil diferencial



$$V = \frac{V_R + V_L}{2} \quad (1)$$

$$\dot{\varphi} = \omega = \frac{V_R - V_L}{2b} \quad (2)$$

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.

$$\dot{x} = V\cos(\varphi) \quad (3)$$

$$\dot{y} = V\mathrm{sen}(\varphi) \quad (4)$$

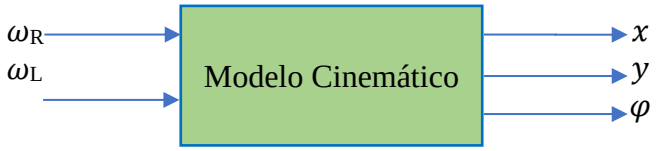
$$\dot{\varphi} = \omega \quad (5)$$

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 11

Modelamiento de entradas y salidas de un sistema cinemático

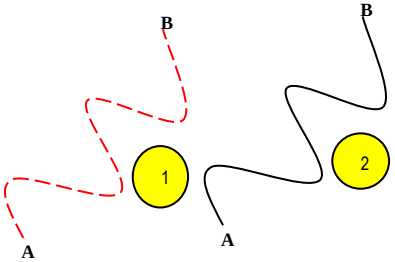


$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \frac{r\cos(\varphi)}{2} & \frac{r\cos(\varphi)}{2} \\ \frac{r\sin(\varphi)}{2} & \frac{r\sin(\varphi)}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \quad (6)$$

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 el posicionamiento del robot en las coordenadas cartesianas *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 12, donde la curva 2 representa el recorrido original y la curva 1 representa el recorrido divido en varios desplazamientos lineales.

Figura 12

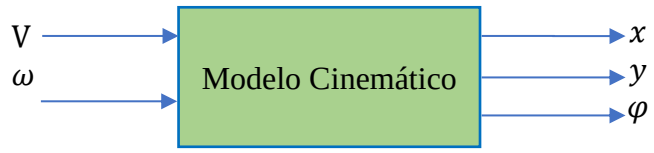
Diferencia del controlador de desplazamiento y de trayectoria



2.1.2.3.8. ***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 [31].

Figura 13

Representación modificada de entradas y salidas de un sistema cinemático

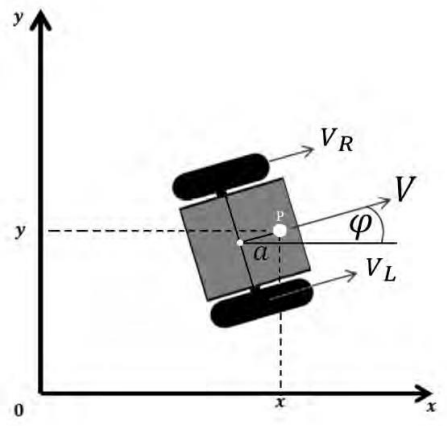


$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \quad (7)$$

El siguiente paso es mover el punto P una distancia a y se obtiene el esquema de la Figura 14.

Figura 14

Configuración diferencial con desplazamiento del punto P a 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.

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

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:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = M\begin{bmatrix}V\\ \omega\end{bmatrix} \quad (9)$$

Donde,

$$M = \begin{bmatrix}\cos\varphi & -a\sin\varphi\\ \sin\varphi & a\cos\varphi\end{bmatrix} \quad (10)$$

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

$$\begin{bmatrix}V\\ \omega\end{bmatrix} = M^{-1}\begin{bmatrix}K_1 x_e + \dot{x_d}\\ K_2 y_e + \dot{y_d}\end{bmatrix} \quad (11)$$

Donde:

$$x_e = x_d - x \quad (12)$$

$$y_e = y_d - y \quad (13)$$

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.

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

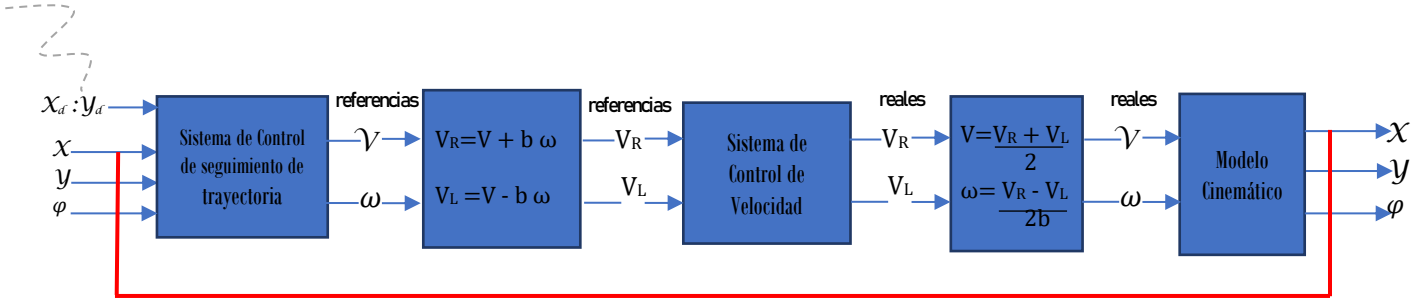
Además, reemplazando la Ley de Control en el modelo cinemático, se obtiene la ecuación que permitirá hallar las constantes $K_{1,2}$.

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \end{bmatrix} = \begin{bmatrix} -k_1 & 0 \\ 0 & -k_2 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \end{bmatrix} \quad (15)$$

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 15

Controlador del robot móvil mediante configuración diferencial



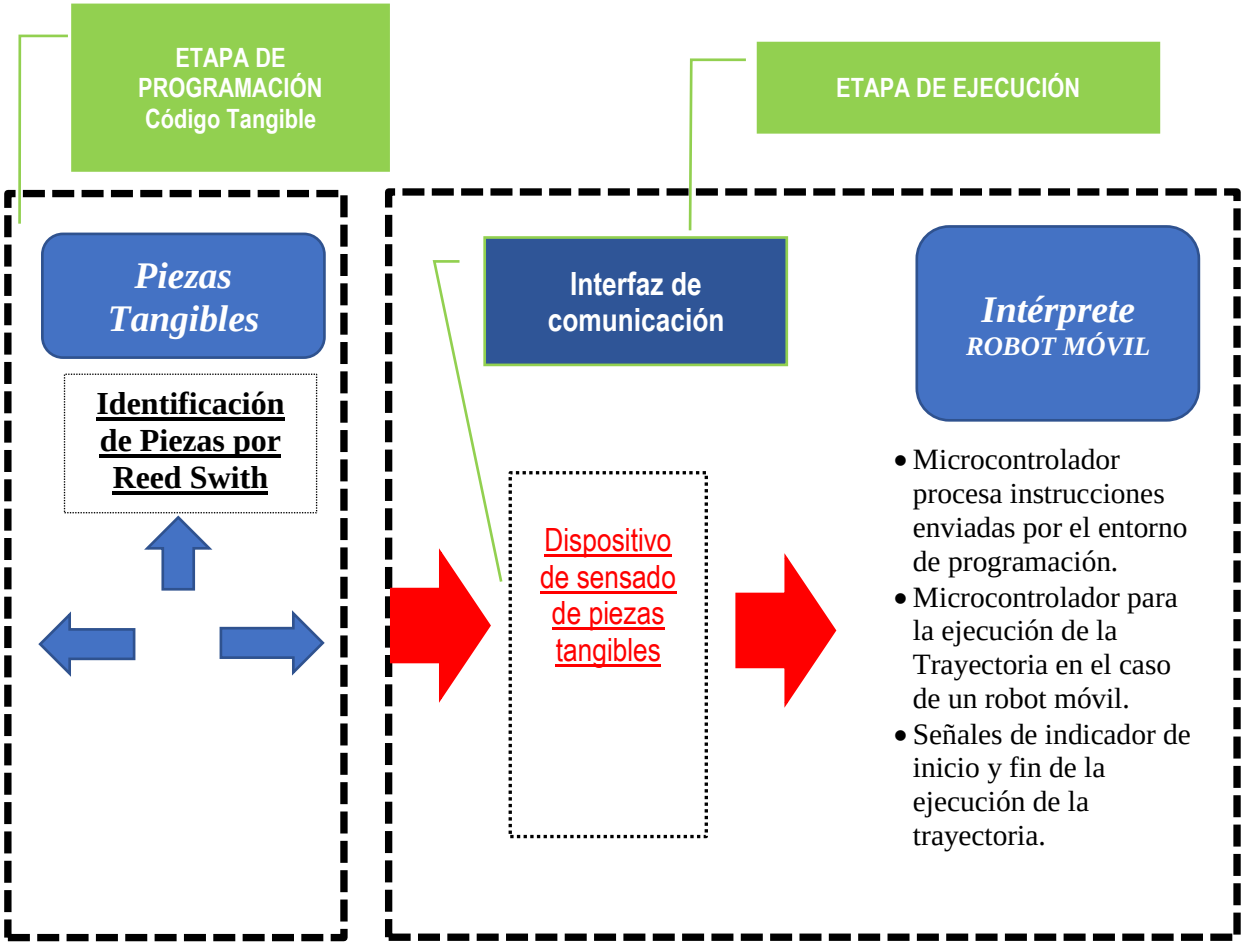
2.1.2.3.9. *Parámetros del prototipo electrónico.* Los parámetros son características que el prototipo electrónico debe contar una vez concluido el diseño y el ensamblaje final, para dar una respuesta optima al momento de realizar funcionamiento respectivo, para ello mencionamos los siguientes parámetros:

- ✓ **Dinamicidad. –** Es la capacidad de la actividad o movimiento que tiene el prototipo electrónico para ejecutar la trayectoria codificada mediante la programación tangible.
- ✓ **Desempeño. –** Referido al rendimiento del prototipo de la plataforma electrónica, es decir al tiempo de respuesta que demanda la ejecución de la trayectoria codificada desde su inicialización.
- ✓ **Fidelidad. –** Es referido a la exactitud y la precisión del prototipo electrónico frente a la ejecución de la trayectoria codificada.

- ✓ **Confiabilidad. –** La confiabilidad del prototipo de la plataforma electrónica es la taza o porcentaje de error que se presenta durante el tiempo demandado en ejecutar la trayectoria codificada.

Figura 16

Diagrama de bloques del prototipo electrónico



2.1.3. Marco Metodológico

La metodología para desarrollar la solución propuesta se estructura en las siguientes fases:

Fase 1: Investigación y Análisis Preliminar:

- ✓ Estudio comparativo de plataformas de programación tangibles existentes (marco histórico).
- ✓ Identificación de fortalezas y debilidades de cada plataforma.
- ✓ Desarrollo del marco teórico y metodológico.

Fase 2: Diseño del Prototipo Electrónico:

- ✓ Elaboración y definición de un listado de exigencias mínimas.
- ✓ Diseño del lenguaje de programación tangible.
- ✓ Elaboración de la estructura de funciones.
- ✓ Desarrollo del concepto de solución.

Fase 3: Desarrollo e implementación del Prototipo electrónico:

- ✓ La implementación del prototipo que incluye:
  - o Lenguaje de programación.
  - o Entorno de programación.
  - o Intérprete actuador electrónico.

Fase 4: Validación y Documentación:

- ✓ Realización de pruebas de usabilidad con usuarios finales.
- ✓ Análisis de resultados.
- ✓ Elaboración de conclusiones y recomendaciones.

Cada fase será documentada en los capítulos correspondientes de la tesis, asegurando una progresión coherente desde el análisis inicial hasta la implementación y validación final del prototipo.

2.1.3.1. Tipo de Investigación. El trabajo que se presenta es una investigación:

Aplicada descriptiva: Esta investigación es **aplicada** por que busca innovar pedagógicamente mediante el uso de las TIC, específicamente una plataforma de programación tangible, para mejorar el aprendizaje en nivel inicial, con un diseño **descriptivo** que evaluará la percepción de usabilidad entre niños de 4 a 6 años y docentes tutores sobre esta innovación tecnológica adaptada al contexto escolar.

2.1.3.2. Variables e Indicadores.

2.1.3.2.1. ***Variable Independiente.*** Se considera una variable independiente en un contexto de investigación aplicada descriptivo debido a las siguientes razones:

- ✓ Aplicada:
- Busca resolver un problema práctico concreto.
- Tiene como objetivo generar una solución inmediata (en este caso, mejorar el aprendizaje).
- Aplica conocimientos teóricos a una situación específica (innovación con programación tangible).
- Pretende implementar una herramienta tecnológica con impacto directo.
- ✓ Descriptiva:
- Describe y caracteriza un fenómeno (percepción de usabilidad de niños y docentes).
- No busca modificar la realidad, sino comprenderla.
- Recolecta información detallada sobre la innovación tecnológica.

- Evalúa percepciones y características de los participantes.
- Permite conocer cómo se manifiesta el fenómeno estudiado en su contexto natural.

En resumen, la plataforma electrónica de programación tangible es una variable independiente en un estudio descriptivo cuya variable dependiente será en nivel de percepción de usabilidad entre niños y docentes ya que permite manipular y evaluar su impacto en un contexto educativo real. Esto proporciona una base sólida para investigar a largo plazo cómo esta herramienta pedagógica que podría mejorar el aprendizaje y desarrollo del pensamiento computacional entre los estudiantes, ver la tabla 1.

Tabla 1

Variable

Variables	Definición Conceptual	Indicadores	Técnicas e instrumentos
Variable Independiente Plataforma Electrónica de programación Tangible.	Es una forma de interacción hombre - máquina, mediante un hardware electrónico que incorpora in microcontrolador reprogramable y que permite la utilización de objetos físicos como componentes de programación y en donde cada objeto tiene características y funcionalidades específicas que al secuenciarles o colocar en un orden determinado es posible generar un programa, para que finalmente sea interpretado mediante un actuador electrónico.	Funcionabilidad.	✓ Porcentaje de funciones de la plataforma que se implementan correctamente. (Observación directa). ✓ Tasa de errores o fallos en la plataforma durante las pruebas. (Registro de errores).
Variable Dependiente Percepción de Usabilidad.	La percepción de usabilidad se define como la valoración subjetiva de los usuarios (niños de 4-6 años y docentes) respecto a la facilidad, eficiencia, comprensión y satisfacción en el uso de la plataforma de programación tangible, evaluando la interacción, la intuitividad de la interfaz, el esfuerzo requerido para realizar tareas, la satisfacción emocional y la capacidad de aprendizaje de la herramienta tecnológica en el contexto educativo.	Usabilidad.	✓ Tiempo promedio que los usuarios tardan en completar una tarea en la plataforma. (Registro de tiempo). ✓ Puntuación promedio de satisfacción del usuario. (Encuestas).
		Interacción.	✓ Numero de interacciones o manipulaciones realizadas por los usuarios durante la sesión. (Registro de uso). ✓ Frecuencia de uso de diferentes características de la plataforma. (Registro de Uso).

Nota: Elaboración propia.

CAPÍTULO III: DISEÑO DEL PROTOTIPO DE LA PLATAFORMA ELECTRONICA DE PROGRAMACION TANGIBLE

El prototipo de la plataforma electrónica de programación tangible debe ser diseñada de manera que sea fácil de usar y permita a los educandos interactuar con ella de forma intuitiva y natural, y para lograr lo se mencionó en el marco metodológico, se tomará como referencia la norma alemana para el proceso de diseño mecatrónico VDI 2206 [32], en donde se tiene lo siguiente:

- ✓ **Primera Etapa**. - Es la elaboración de la lista de exigencias con las características principales que deberá tener la plataforma electrónica de programación tangible.
- ✓ **Segunda Etapa**. - Diseño del lenguaje de programación tangible.
- ✓ **Tercera Etapa**. - la elaboración de la estructura de funciones y el concepto de solución.
- ✓ **Cuarta Etapa**. - 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 electrónica a nivel de prototipo con objetivos educativos para la aplicación de la programación tangible en escolares del nivel inicial, el cual sirva como una herramienta de enseñanza alternativa de manera lúdica y dinámica, con la finalidad de que sea considerada como una metodología pedagógica que facilite el aprendizaje de la programación en niños de corta edad para poder ayudar en el desarrollo del pensamiento computacional, pensamiento crítico y la resolución de problemas a través de la programación tangible. La plataforma electrónica de programación tangible deberá estar compuesta por tres partes principales: lenguaje, entorno e intérprete. El lenguaje contempla a las piezas tangibles de programación, en donde cada una de las piezas contienen reglas y condiciones para la

programación, asimismo la interacción con el intérprete será a través del sensado y procesamiento de las piezas tangibles en el tablero programable o entorno de programación, debidamente ordenadas acorde a una trayectoria definida. El intérprete es un robot móvil con locomoción a ruedas de tipo diferencial que deberá seguir una trayectoria definida por las piezas tangibles o lenguaje de programación. 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 *Plataformas de Programación Tangible* estudiadas en el Marco Histórico, el tamaño recomendado para el intérprete debe ser de máximo 25 cm X 25 cm X 25 cm, además tendrá morfología animal ya que es un producto orientado a niños. Las piezas tangibles serán cuadradas, sin bordes filosos y de 5 cm X 4 cm X 1 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 Teórico. 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 = 0.12 m), velocidad tomada de [12].
✓ Control	Se realizará un control de trayectoria basado en el modelo cinemático para un robot diferencial [33].
✓ 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 [34]. 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, consecuentemente para los niños. 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 electrónico robot móvil 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 y las piezas tangibles en base al uso del material MDF y PLA que, al ser compostable, permitirá un proceso de eliminación más eco-amigable y así colaborar con la reducción en el impacto medio ambiental. 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 [35].
✓ 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 [36]. 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 2 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 [37]. 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.

Se pueden adaptar los estudios realizados al lenguaje, propiamente dicho, para poder comprender la semiótica de los leguajes de programación [38]. 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 [39]. Estas partes serán definidas según la Real Academia de la Lengua Española y se explicará cómo es que se relacionan con un lenguaje de programación, ver Tabla 3.

Tabla 3

Disciplinas de la semiótica aplicada 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 [38].

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 (claro, simple, conciso y con una sintaxis legible) para el programador desde antes de la codificación.

- 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.

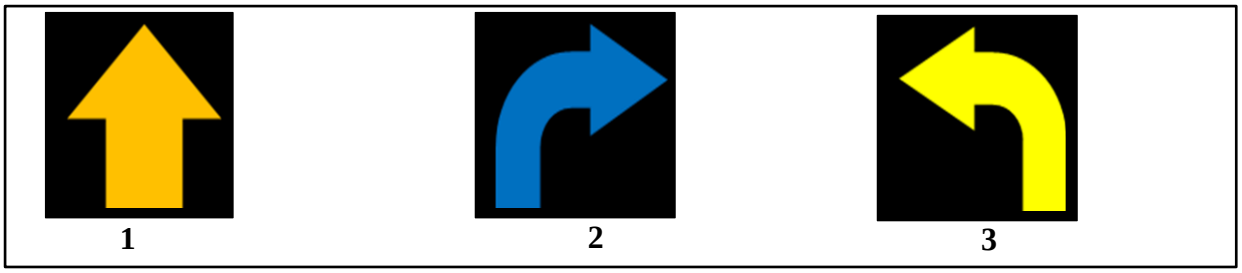
3.2.1. Diseño de la Sintaxis y Especificación de la Semántica

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 el siguiente bloque:

Figura 17

Representación gráfica de los bloques de comando

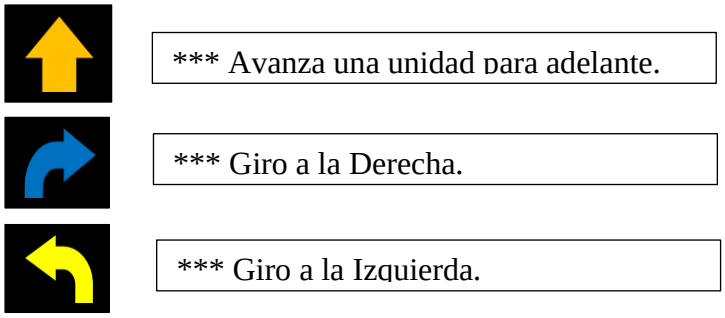


Como se puede ver en la Figura 17, existen tres bloques de comando: avanzar, girar a la derecha y girar a la izquierda.

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

Figura 18

Reglas del bloque de comandos para crear un algoritmo



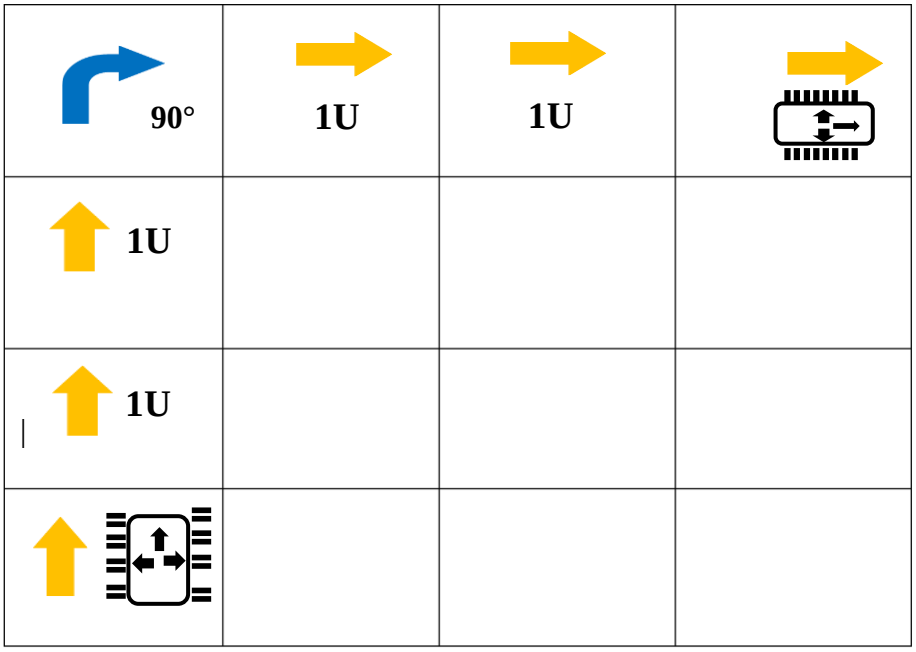
En base a todas las reglas previamente mencionadas, la figura 19 ejemplificará un programa completo y la Figura 20 representará cómo el intérprete robot ejecutaría dicho programa.

Figura 19

Programa que representa un algoritmo (trayectoria)

Figura 20

Trayectoria del robot al ejecutar el algoritmo programado de la Figura 19



3.3. Elaboración de la Estructura de Funciones

La creación de la estructura de funciones implica diversas etapas que facilitarán la identificación de las funciones necesarias para que la Plataforma de Programación Tangible a desarrollar cumpla con el objetivo principal establecido en 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 busca expandir la creatividad del diseñador explorando soluciones desde tres perspectivas fundamentales: señales (datos, información), energía (mecánica, eléctrica) y materia (insumos, piezas). Se desarrollarán dos cajas negras: una para el entorno de

programación y otra para el intérprete robot móvil, permitiendo una visualización conceptual de sus componentes y funcionalidades.

Figura 21

Caja negra del entorno de programación

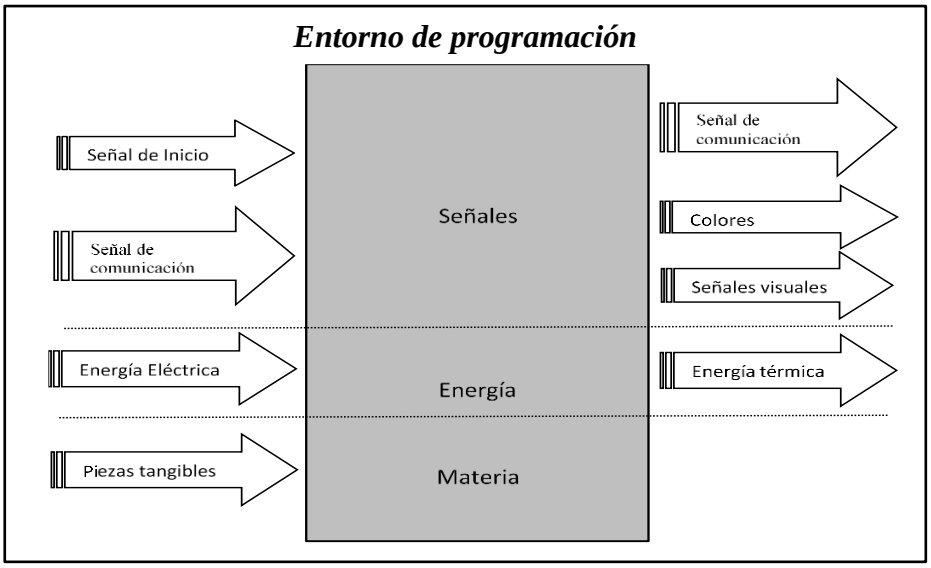
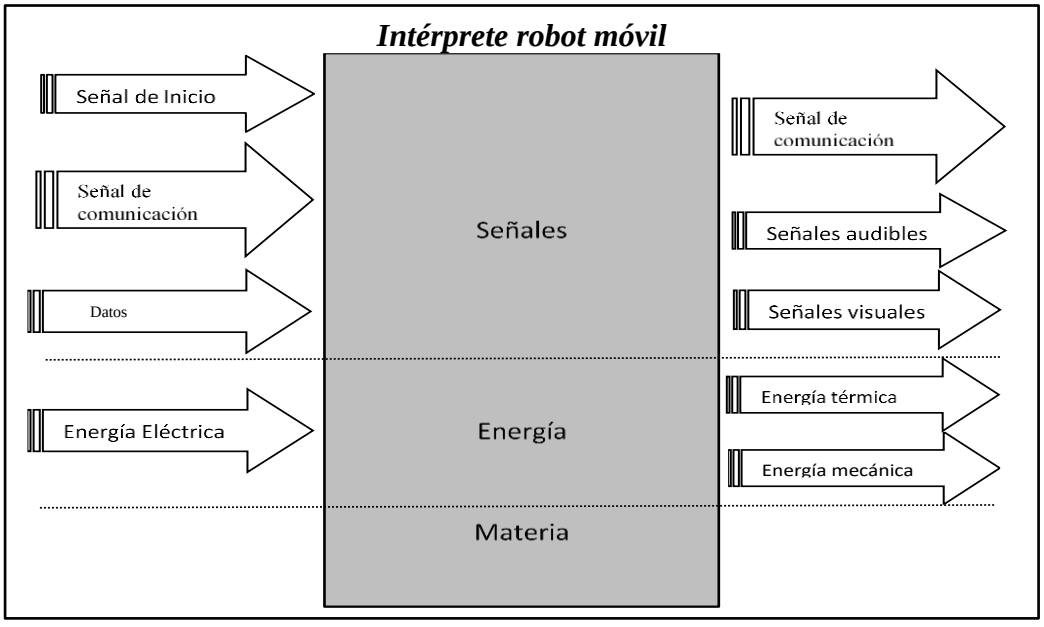


Figura 22

Caja negra del interprete robot móvil



3.3.2. Principios Tecnológicos y Secuencia de Operaciones

Para crear una plataforma de programación tangible, es esencial entender cómo convertir las acciones físicas en instrucciones que el sistema pueda ejecutar. Este proceso funciona como una "caja negra", donde las acciones del usuario al manipular los bloques de programación son las entradas, y las respuestas del sistema, como el movimiento del robot móvil, son las salidas. Para lograr esta transformación, nuestra investigación teórica se ha enfocado en tres tecnologías fundamentales: el procesamiento de señales, que permite interpretar las interacciones con los bloques; el diseño electrónico, necesario para crear los circuitos y sistemas de control; y el diseño mecánico, crucial para desarrollar un robot móvil que responda adecuadamente a las instrucciones programadas.

La plataforma de programación tangible funciona siguiendo una secuencia clara y estructurada. Para comenzar, el usuario debe encender tanto el robot móvil como el tablero programable. Luego, el estudiante crea un algoritmo colocando las piezas tangibles en el tablero, definiendo así la trayectoria que seguirá el robot.

Una vez iniciado el sistema, este realiza dos procesos principales:

- ✓ Identificación y procesamiento:
  - El arreglo de Reed Switch y el conversor analógico-digital (ADC) detectan las piezas tangibles.
  - El microcontrolador identifica la codificación de cada pieza.
  - El sistema procesa la secuencia según el orden establecido.
  - Se genera la trayectoria basada en el algoritmo creado.
- ✓ Ejecución de movimientos:
  - La trayectoria se transmite al robot móvil mediante conexión Bluetooth.

- El microprocesador interno del robot recibe las instrucciones.
- El robot ejecuta la secuencia de movimientos programada (adelante, izquierda y derecha), así como se observa en las figuras 19 y 20.

Este diseño permite una interacción intuitiva entre el estudiante y la programación, traduciendo elementos físicos en movimientos reales del robot.

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 sobre el tablero programable.
5. Verificar la correcta colocación de las piezas tangibles en el tablero programable.

- Etapa de ejecución:

1. Sensar piezas tangibles, es decir identificar las piezas tangibles mediante el arreglo de Reed Switch y del conversor analógico digital ADC para obtener el código y/o instrucción a través del microcontrolador.
2. Generar datos de la trayectoria obtenida del tablero programable y las piezas tangibles.
3. Realizar el control de trayectoria.
4. Generar movimiento.

5. Emitir señales visuales y 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 4.

Tabla 4

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 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	Inicialización	Proceso manual.	Incapacidad humana.
F7	Sensar piezas tangibles.	Proceso Automático.	Reed Switch dañado.
F8	Identificar instrucciones.	Proceso Automático.	Ninguna.
F9	Generar datos de trayectoria.	Proceso Automático	Ninguna.
F10	Realizar el control de trayectoria.	Proceso Automático.	Superficie irregular.
F11	Generar movimiento.	Proceso Automático.	Superficie irregular.
F12	Emitir señales visuales.	Proceso Automático.	Ninguna.
F13	Emitir señales audibles.	Proceso Automático.	Ninguna.

3.3.5. Agrupación de Funciones

La optimización de la estructura funcional requiere una integración cuidadosa de los procesos técnicos del sistema, los cuales pueden ejecutarse de forma secuencial o simultánea. Las figuras 23 y 24 muestran una clasificación de las funciones en dos categorías: procesos manuales, que requieren intervención directa del usuario y siguen una secuencia predefinida, y procesos automáticos, que se ejecutan sin intervención del usuario y pueden ocurrir simultáneamente en respuesta a las entradas manuales.

Figura 23

Agrupación de funciones para los procesos manuales

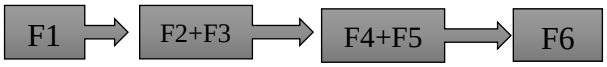


Figura 24

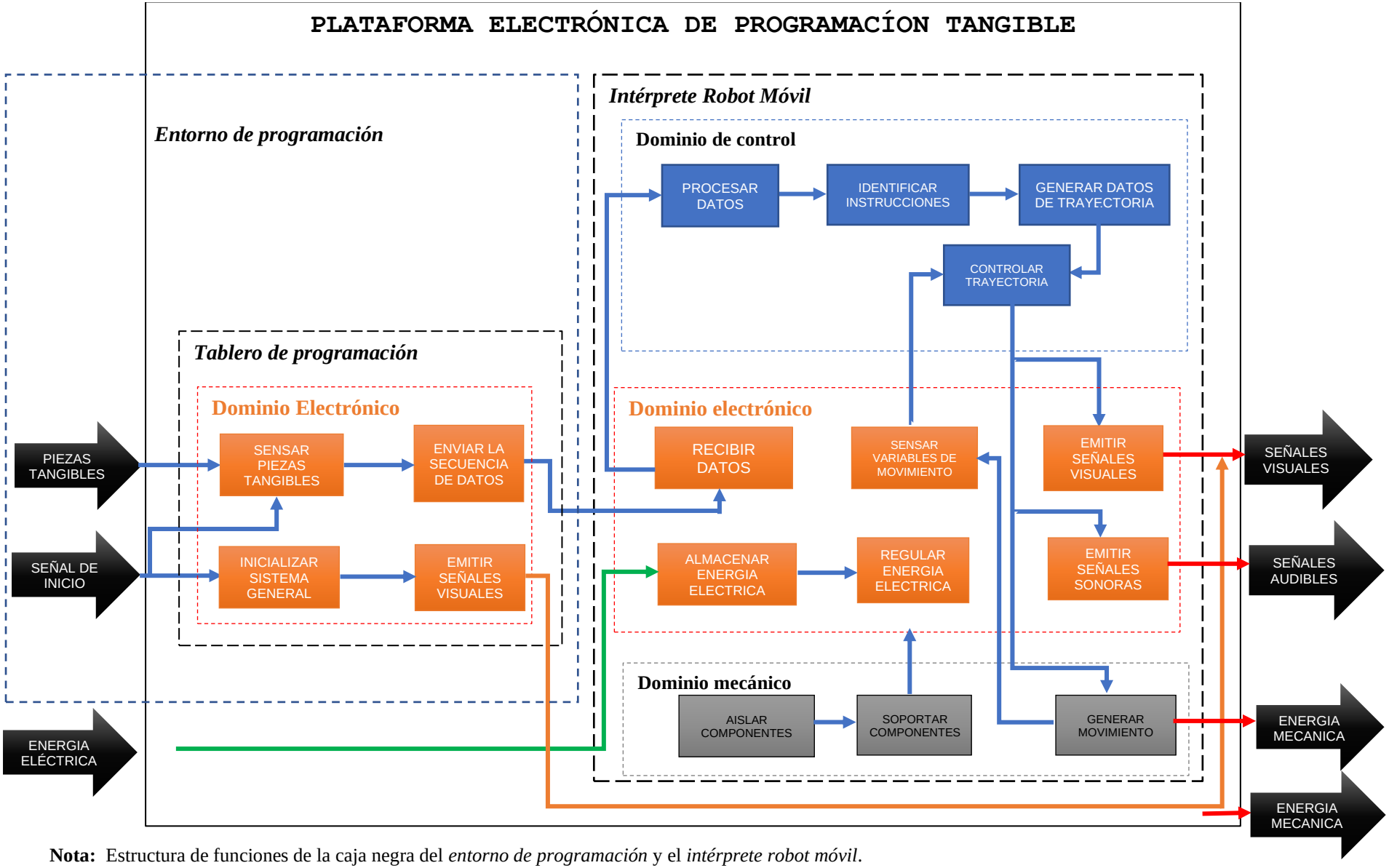
Agrupación de funciones para los procesos automáticos

3.3.6. Representación de la Estructura de Funciones

La Figura 25 muestra un diagrama de estructura de funciones que ilustra cómo se relacionan los diferentes componentes del sistema. Este diagrama es especialmente importante porque visualiza la interacción entre las entradas y salidas de dos elementos fundamentales: el entorno de programación y el intérprete, previamente representados como cajas negras en las Figuras 21 y 22. Esta representación nos permite entender cómo fluye la información a través del sistema y cómo cada componente se comunica con los demás para lograr el funcionamiento deseado.

Figura 25

Estructura de funciones



Nota: Estructura de funciones de la caja negra del *entorno de programación* y el *intérprete robot móvil*.

3.4. Concepto de Solución

Para llegar al diseño definitivo del proyecto, se seguirá un proceso estructurado de toma de decisiones. Partiendo de la estructura de funciones ya establecida, se desarrollará una matriz morfológica que permitirá visualizar diferentes opciones tecnológicas y componentes disponibles. Esta matriz ayudará a generar diversas alternativas de solución, cada una combinando distintas tecnologías y componentes que cumplan con los requerimientos y especificaciones del proyecto. Posteriormente, se evaluará cada alternativa según criterios específicos, lo que permitirá seleccionar la solución más adecuada y viable para implementar.

3.4.1. Matriz Morfológica

Para desarrollar el prototipo electrónico de programación tangible, necesitamos definir claramente los componentes del entorno de programación, que consta de dos elementos principales:

- Las piezas tangibles.
- El tablero programable con tecnología Reed Switch.

La matriz morfológica, presentada en la tabla 5, es una herramienta fundamental que nos ayuda a visualizar y organizar todas las posibles soluciones para cada función del sistema. Esta matriz está estructurada de la siguiente manera:

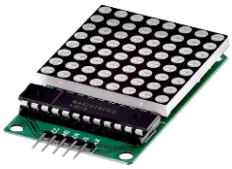
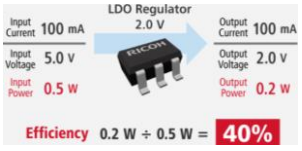
- En la primera columna se listan las funciones parciales del sistema.
- En las filas correspondientes se presentan las diferentes opciones o "portadores de funciones" que podrían cumplir con cada función.

Esta organización nos permite evaluar sistemáticamente las diferentes combinaciones posibles para encontrar la solución óptima que cumpla con el objetivo principal del proyecto.

Tabla 5

Matriz morfológica del interprete móvil

N°	Función	Alternativa 1	Alternativa 2
1	Enviar / Recibir secuencia de datos.	Wi-Fi	Bluetooth®
2	Codificación de piezas tangibles	**Tarjeta NFC RFID** 	**Imán de Neodimio** 
3	Sensado y/o decodificación de piezas tangibles.	**Modulo Lector RFID 13.56Mhz RC522** 	**Interruptor / Sensor MAGNETICO Reed Switch** 
4	Procesar y Sensar las Piezas Tangibles.	**Microcomputadora** 	**Microcontrolador** 
5	Realizar control de trayectoria.	**Microcomputadora** 	**Microcontrolador** 

		Giroscopio	**Giroscopio + Encoder**
6	Sensar variables de movimiento.		
		Servo Motor	**Motor paso a paso**
7	Generar movimiento.		
		Matriz LED	**Cursores Iluminados**
8	Emitir señales visibles.		
		Altavoz	**Buzzer**
9	Emitir señales audibles.		
		Baterías Ni/MH	**Baterías de Li-Ión**
10	Almacenar energía.		
		TC 4056 witch Step Up	**TP4056**
11	Cargador de Batería de Li.		
		LDO	**STEP UP**
12	Regulador de energía eléctrica.		

3.4.2. Descripción de las Alternativas de Solución

A partir de la matriz morfológica, se pudieron llegar a dos alternativas de solución 1 y 2, las cuales serán descritas y evaluadas a continuación para poder determinar cuál es la más óptima y de este modo, cumplir con el objetivo principal de la tesis.

3.4.2.1. Solución de la Alternativa 1. La primera propuesta se basa en el uso de tarjetas especiales (que contienen tecnología NFC y RFID) que funcionan como bloques de instrucciones para programar. Cada tarjeta representa una instrucción diferente que el robot debe seguir, permitiendo crear una secuencia de movimientos o una ruta específica. El proceso funciona así: primero, un lector RFID identifica las tarjetas que el usuario ha seleccionado; luego, mediante el convertidor ADC de decodifica esta información; finalmente, la microcomputadora (Raspberry) envía estas instrucciones de manera inalámbrica por WiFi al robot móvil para que las ejecute.

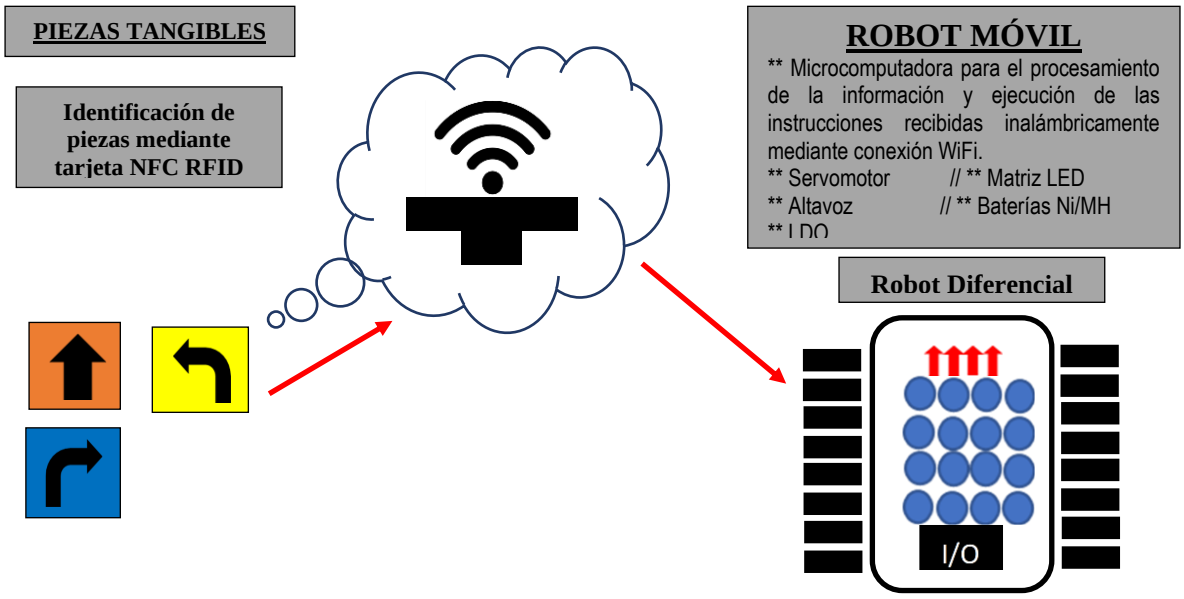
El robot móvil (que actúa como intérprete) recibe la secuencia de instrucciones a través de una conexión WiFi. Estas instrucciones son procesadas por otra microcomputadora (Raspberry) que lleva incorporada el robot. Esta computadora se encarga de convertir las instrucciones recibidas en movimientos específicos, permitiendo que el robot siga la ruta o trayectoria que fue programada usando las tarjetas.

Una vez procesadas las instrucciones, el robot móvil las ejecutará para moverse siguiendo la trayectoria definida. Se trata de un robot con forma cuadrada, que se mueve utilizando un sistema de locomoción diferencial, lo que significa que tiene dos motores de precisión (servomotores), uno a cada lado, que le permiten avanzar y girar con exactitud. Para interactuar con su entorno, el robot móvil está equipado con una matriz LED que puede mostrar señales visuales, y también cuenta con un altavoz incorporado que le permite emitir señales audibles. Estas características le permiten

al robot no solo moverse según las instrucciones programadas, sino también dar retroalimentación visual y auditiva durante su funcionamiento

Figura 26

Solución - alternativa 1



3.4.2.2. Solución de la Alternativa 2. La segunda propuesta utiliza un sistema de programación basado en piezas que contienen imanes especiales (de neodimio). Cada pieza representa una instrucción diferente, y el usuario puede organizarlas para crear una trayectoria específica, el sistema funciona de la siguiente manera:

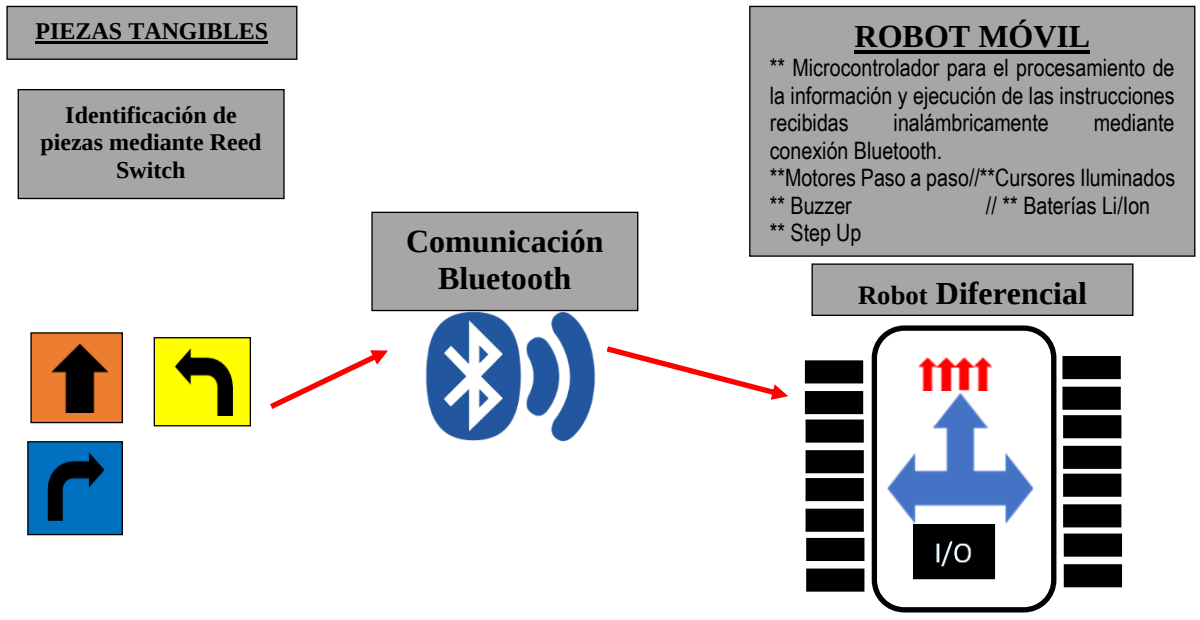
- El usuario coloca las piezas magnéticas sobre el tablero de programación en el orden deseado para crear un programa es decir definir la trayectoria.
- El tablero de programación consta de un conjunto de sensores especiales (llamados Reed Switch) detecta la presencia y posición de los imanes.
- Un convertidor (ADC) transforma estas señales magnéticas en señales digitales.

- Un microcontrolador Arduino procesa toda esta información controlar los movimientos del robot para que siga exactamente la ruta que fue programada usando las piezas magnéticas.
- Finalmente, las instrucciones se envían al robot móvil mediante una conexión Bluetooth y esta ejecuta las instrucciones generándose así una trayectoria a partir de las piezas magnéticas.

El robot tiene una apariencia que imita a un animal y utiliza motores paso a paso para moverse con precisión mediante un sistema de dos ruedas (locomoción diferencial). Para hacer más interactivo su funcionamiento, el robot cuenta con indicadores luminosos que muestran la dirección hacia donde se está moviendo y un pequeño altavoz (buzzer) que puede emitir sonidos para señalar diferentes acciones o estados.

Figura 27

Solución - alternativa 2



3.4.2.3. Evaluación Técnico – Económica de las Alternativas de Solución. Para elegir la mejor opción entre las alternativas 1 y 2, necesitamos evaluarlas usando un sistema de puntuación con criterios específicos de comparación. La evaluación usará la siguiente escala de puntuación:

- 0 puntos = No satisface
- 1 punto = Aceptable
- 2 puntos = Suficiente
- 3 puntos = Bien
- 4 puntos = Muy bien

Este sistema de evaluación nos permitirá comparar ambas alternativas de manera objetiva y determinar cuál es la más adecuada para nuestro proyecto.

Tabla 6

Evaluación técnico-económica de las alternativas de solución

N°	Criterios Técnicos y Económicos	Alternativas de Solución		
		01	02	Óptima
1	Seguridad	2	3	4
2	Rapidez	2	3	4
3	Estabilidad	3	4	4
4	Fidelidad	2	3	4
5	Dinamicidad	2	3	4
6	Facilidad de Manejo	3	3	4
7	Confiabilidad y taza de error	3	3	4
8	Lista de exigencias	3	3	4
9	Número de piezas	3	3	4
10	Fácil adquisición de materiales de fabricación	3	3	4
11	Costos diversos	3	3	4
12	Fácil montaje	3	3	4
	Total, de puntos	32	37	48

Según nuestros criterios de evaluación, para considerar que una alternativa es aceptable y cumple con los requisitos del proyecto y los objetivos de la tesis, debe alcanzar al menos 20 puntos en total.

Al revisar los resultados mostrados en la Tabla 6, podemos ver que tanto la alternativa 1 como la alternativa 2 superaron este puntaje mínimo requerido. Sin embargo, después de comparar ambas opciones, la alternativa 2 (que usa piezas con imanes de neodimio) obtuvo una puntuación más alta, lo que la convierte en la opción más adecuada para este proyecto.

3.5. Proyecto Definitivo

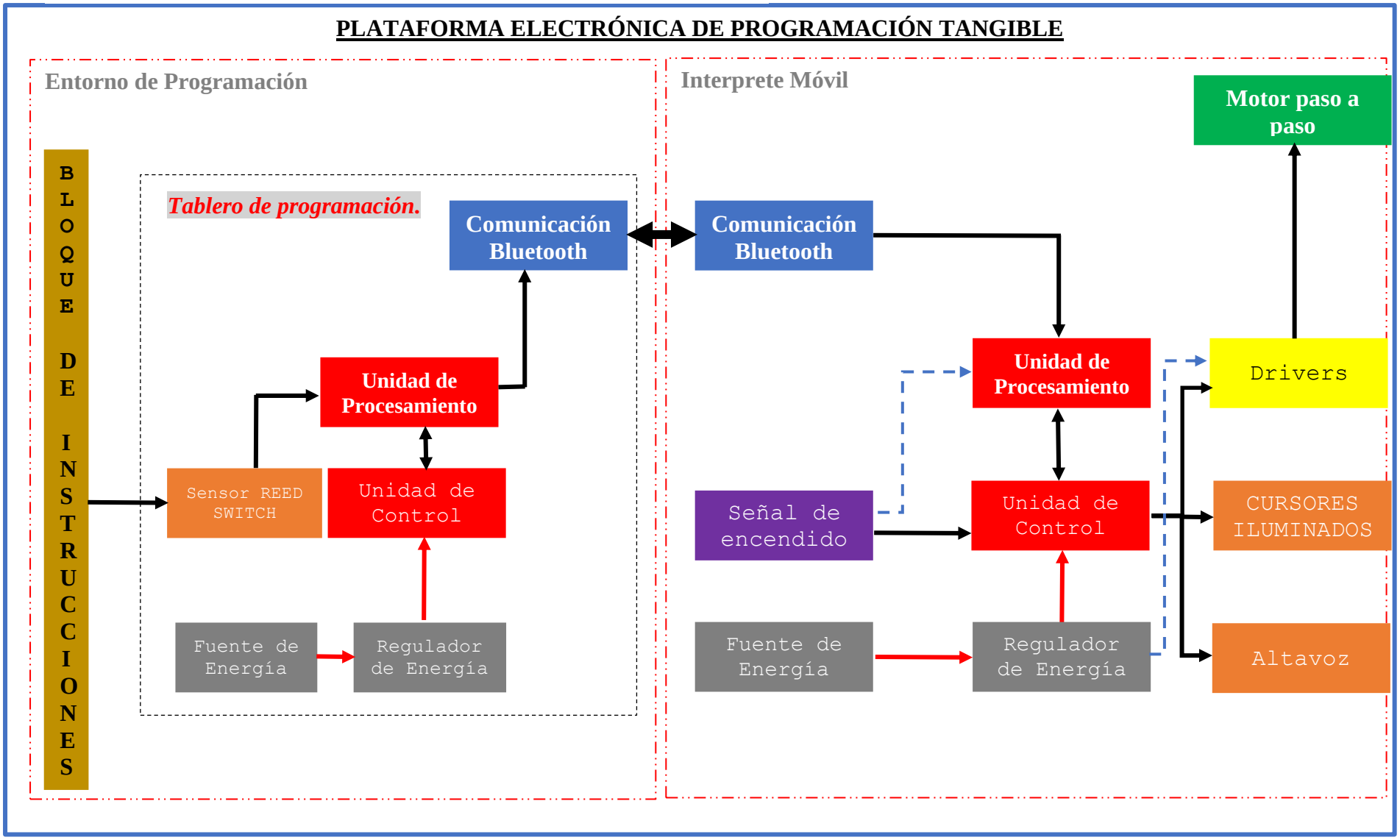
Después de analizar las diferentes alternativas usando una matriz morfológica y evaluar cada opción, hemos podido crear un diagrama de bloques que muestra la estructura final del proyecto (que se puede ver en la Figura 28). Este diagrama revela que el proyecto se divide en dos partes principales que necesitan ser diseñadas:

- ✓ El entorno de programación: donde el usuario interactúa con las piezas para crear las instrucciones.
- ✓ El intérprete móvil: que es el robot que ejecuta las instrucciones programadas.

Esta estructura nos permite visualizar y organizar mejor el desarrollo del proyecto, trabajando en cada parte de manera específica y cómo se conectan entre sí.

Figura 28

Modelo de solución final planteado



CAPÍTULO IV: IMPLEMENTACIÓN DEL PROTOTIPO DE LA PLATAFORMA ELECTRONICA DE PROGRAMACION TANGIBLE

La implementación del prototipo de programación tangible es un proceso que convierte el diseño inicial en un dispositivo electrónico funcional. Este capítulo incluye la selección de los componentes electrónicos necesarios (así como se describió con anterioridad que sean componentes y tarjetas electrónicas del mercado local), el diseño del circuito que los conectará, el montaje físico de las piezas, la programación del cerebro electrónico (microcontrolador), y la unión del hardware con el software. Todo esto se somete a pruebas iniciales y finales para garantizar que funcione correctamente. Finalmente, se protege todo el sistema en una carcasa resistente, lo que permite tener una primera versión del producto que se puede mejorar con el tiempo.

4.1. Diseño del Hardware del Interprete Robot Móvil

El diseño del intérprete robot móvil contempla las siguientes partes principales: el diseño electrónico, el diseño mecánico 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.

4.1.1. Diseño Electrónico de Robot Móvil y Elección de Componentes

El diseño electrónico del robot móvil abarca la elección de componentes que cumplan con los requerimientos de la solución de la alternativa 2 y el diagrama esquemático, y para realizar la elección de componentes, se analizaron los requerimientos básicos de estos en las diferentes opciones existentes en el mercado actual.

4.1.1.1. Unidad de procesamiento y control. La unidad de procesamiento y control es la parte operativa del sistema que gestiona tres funciones principales: la recepción e interpretación de datos vía protocolo Bluetooth provenientes del entorno de programación tangible; el procesamiento de estas instrucciones que representan la secuencia definida por el usuario mediante las piezas tangibles en el tablero de programación; y la ejecución del control de movimiento que traduce estas instrucciones en señales para los actuadores (motores) y los indicadores audiovisuales. Este sistema integrado permite la transformación de comandos físicos en una trayectoria de movimiento predefinida, coordinando la activación sincronizada de los elementos de salida según la programación tangible realizada por el usuario.

La selección del microcontrolador se basará en una tabla comparativa que especifica la cantidad de pines necesarios para cada periférico y las características técnicas requeridas, permitiendo una evaluación y comparación entre diferentes opciones disponibles en el mercado, ver tabla 8.

Tabla 7

Periféricos

Periférico	**Nro de entradas y salidas.**
Driver de motor derecho	4
Driver de motor izquierdo	4
Indicador de nivel de batería	3
Entrada de señal de nivel de batería	1
Cursores Ilimunados	3
Led de encendido general	1
Altavoz	1
Modulo Bluetooth	2
Total	**19**

El microcontrolador seleccionado debe disponer de un mínimo de 19 pines para gestionar las entradas/salidas digitales y analógicas del sistema, más un margen adicional de pines de reserva

para futuras expansiones o contingencias. Este requerimiento técnico es un criterio fundamental para el proceso de selección del controlador.

Tabla 8

Comparación de microcontroladores adecuados

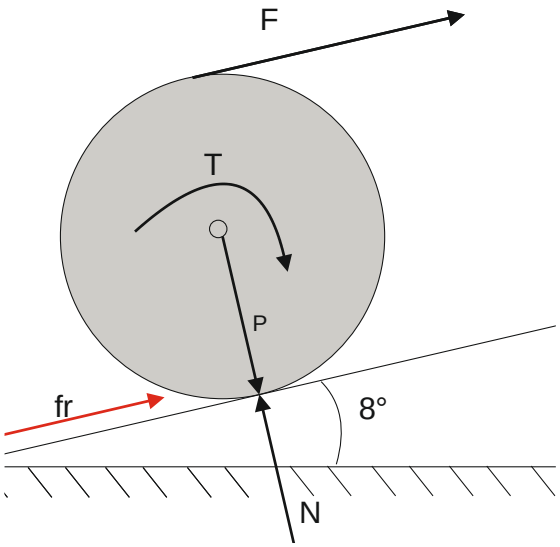
Características	Requerimientos	Alternativas	
		Arduino UNO	**Arduino NANO**
Procesador	----------	ATMEGA 328	ATMEGA 328
Velocidad de reloj	>= 16MHz	16MHz	16MHz
Memoria interna	>=32Kb	32Kb	32Kb
Comunicación inalámbrica	Bluetooth	No	No
Dimensiones (mm)	<=50x10mm (largo X ancho)	68x53mm	45x18mm
Voltaje de alimentación	----------	7-12 Volt	7-12 Volt
Disponibilidad	----------	Mercado local	Mercado local
Precio	----------	**S/. 129.00**	**S/.91.00**

De acuerdo a la tabla 9 y estimando un análisis de diseño conservador se valora las condiciones máximas de operación (medidas y pesos), se ha seleccionado el microcontrolador Arduino Nano como unidad de control óptima. Esta elección se fundamenta en dos criterios principales: sus especificaciones técnicas satisfacen los requerimientos de diseño establecidos, incluyendo el número necesario de pines como la capacidad de procesamiento; y presenta la mejor relación costo-beneficio entre las alternativas evaluadas.

4.1.1.2. Motores Paso a Paso. Para el intérprete móvil (robot), al tener una 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 29

Diagrama dinámico de una rueda del robot móvil



En la Figura 29, se puede observar la representación de las fuerzas que interactúan en una de las ruedas del robot móvil. En la Tabla 3, se expuso como exigencia de ergonomía que la Plataforma de Programación Tangible no debe superar los 0.8 Kg. Por este motivo, el robot no deberá superar los 1 Kg y esta será la masa asumida para realizar cálculos, de la misma manera se asume un diámetro de rueda de 0.075m de diámetro de la rueda. Además, se plantea una aceleración de 1 m/s2 promedio para la cinemática del intérprete, asimismo asumimos un ángulo promedio de inclinación de 8° porque toda superficie no es llana, se podrían presentar muchas otras superficies diferentes al ideal.

Es sabido que:

$$\sum Fx = ma$$

Donde:

m: Masa que recae sobre todas las ruedas (0.5 Kg)

a: Aceleración deseada ($1m/s^2$)

por lo que se obtiene:

$$fr - Px = ma \quad (16)$$

Además,

$$fr = \frac{T}{R} \quad (17)$$

Donde:

R: Radio máximo de una de las ruedas (0.0375 m)

Finalmente

$$T_{oTOTAL} = n \times R \times m \times (a + g \times \text{sen}(8°)) \quad (18)$$

Donde:

n: eficiencia asumida de 70%

g: fuerza de la gravedad ($9.8m/s^2$)

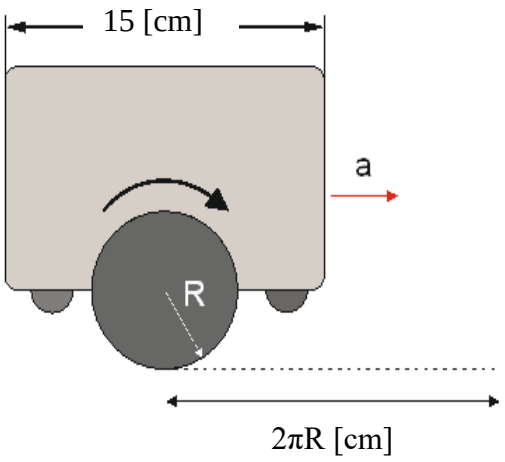
$T_{oTOTAL} = 0.7 \times 0.0375 \times 0.5 \times (1 + 9.8 \times \text{sen}(8°))$

De este modo, se puede obtener que ***T=0.03103N.m*** o haciendo que F=ma, tenemos que; ***T= 0.3163 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 3 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, sin embargo, en el diseño propuesto se está considerando que el tamaño del *interprete será de 0.10m x 0.10m x 0.15 m y su velocidad será de 1 longitud por 3 segundos*, para concluir estos datos se tomarán estos datos como referencias. Además, al ser ruedas de 0.0375 m de radio, se estima que recorrerán 0.23562 m, ver Figura.

Figura 30

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



Finalmente, para el cálculo de rpm:

Sabemos que:

$v = w * r$ (Velocidad lineal en función a la velocidad angular y radio de la rueda)

$w = \frac{v}{r}$ (rad/seg.)

$v = \frac{e}{t}$

$w = \frac{e}{t*r}$ (rad/seg.)

en conclusión:

$RPM = \frac{e}{t*r} * \frac{60}{2\pi}$ (revoluciones por minuto) (19)

Donde:

e: Longitud máxima del robot móvil (0.15 m)

t: Tiempo que tarda en recorrer dicha longitud (3 s)

r: radio de la rueda (0.0375 m)

Reemplazando en la Ecuación (19), se puede obtener un valor de 12.73 RPM, concluyendo que la velocidad mínima requerida para el motor será de 13 rpm, en la Tabla 9, se pueden observar

algunos motores que cumplen con los requerimientos previamente descritos, pero se escogerá el motor del modelo *24BYJ48 DC 5V* (es un motor DC paso a paso) por su precio y su disponibilidad.

Tabla 9

Alternativas de motores DC

Características	**Requerimientos**	**Alternativas**		
		JGA25-370-78K	**RobotShop**	**24BYJ48**
Velocidad nominal	**13 rpm**	77 rpm	165 rpm	199rpm
Torque	**0.3163 Kg/cm**	1.3 Kg/cm	6.5 Kg/cm	0.34 Kg/cm
Voltaje nominal	**5 V**	6 V	6 V	5 V
Corriente de trabajo	**-**	0.5 A	3 A	200 mA
Encoder	**-**	SI	SI	NO
Dimensiones mm	**-**	31x24.4x24.4	24x50x8	42x32x30
Disponibilidad	**-**	Exportación	Exportación	Mercado Local
Precio	**-**	**S/. 23.66 + envío**	**S/. 19.72 + envío**	**S/. 18**

4.1.1.3. 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 ULN 2803 porque cumple con este requerimiento y por su bajo precio.

Tabla 10

Alternativas de drivers para el control de motores

Características	**Requerimientos**	**Alternativas**		
		RS PRO 417-9728	**Pololu Dual MC33926**	**ULN2803**
Voltaje de entrada	**-**	6-15 V	5-28 V	5-12 V
Corriente máxima	**>0.5 A**	3 A.	3 A.	500mA.
Dimensiones (mm)	**-**	62x41x40	45.7x27.9x6.1	22.2x6.1x3.56
Disponibilidad	**-**	Importación	Importación	Mercado Local
Precio (2 motores)	**-**	**2x20.13 dólar**	**S/. 20.95 + envío**	**S/. 2x7.50 + envío**

4.1.1.4. Cursores Iluminados. En cuanto para las señales visibles se ve por conveniente utilizar Cursores Iluminados de LED, con un consumo de 20mA por LED, y para ello utilizaremos un arreglo de 03 LED con la finalidad de indicar la dirección de movimiento y/o acción acorde al orden de las piezas tangibles y la trayectoria definida por el usuario, asimismo indicar que se utilizará Cursores Iluminados de LED por que será más atractivo para los usuarios que en nuestro caso serán niños de 4 a 6 años de edad.

Tabla 11

Especificaciones técnicas de los Cursores Iluminados LED

Características	
Voltaje nominal	1.7 – 3.4 V
Corriente nominal	20 mA
Tipo	Led difuso
Comunicación	SPI
Dimensiones (mm)	10 mm
Disponibilidad	Mercado local
Precio	S/. 3.00

4.1.1.5. Buzzer. La selección del Buzzer también se basará en la disponibilidad y precio, por lo que se escogerá la opción de Naylamp, ver Tabla 12. Además, la intensidad de sonido de dicha opción será la siguiente:

$$I = \frac{P}{4\pi d^2} \qquad (20)$$

Con una atenuación por distancia definida por:

$$A = 20\log(d) \qquad (21)$$

P: Potencia en Watts (0.15 W)

d: Distancia (5 m)

dB = 10 x log (I/I_0)

$I_0 = 10^{-12}$ W/m2 intensidad mínima de audición.

El análisis matemático indica que la intensidad sonora calculada es de 4.77 x 10-4 W/m2 (equivalente a 86.78 dB), que al considerar la atenuación por distancia de 13.98 dB, resulta en una intensidad real de 73 dB. Debido a esta pérdida de potencia sonora, se requiere implementar un amplificador (transistor) para mantener niveles audibles adecuados.

Tabla 12

Alternativas de altavoz

Características	Requerimientos	Alternativas	
		Naylamp Sen - Sound	Buzzer Zumbador
Rango de frecuencias	**> 20 Hz**	0.6 - 10 KHz	2 KHz
Potencia	**>=0.15 W**	0.5 W	0.099-0.15 W
Impedancia	-	8 ohm	8 ohm
Dimensiones (mm)	-	28 diámetro	12 diámetro
Disponibilidad	-	Mercado Local	Mercado Local
Precio	-	**S/. 10.00**	**S/. 6.00 + envío**

4.1.1.6. Unidad de 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 la tabla 13.

Tabla 13

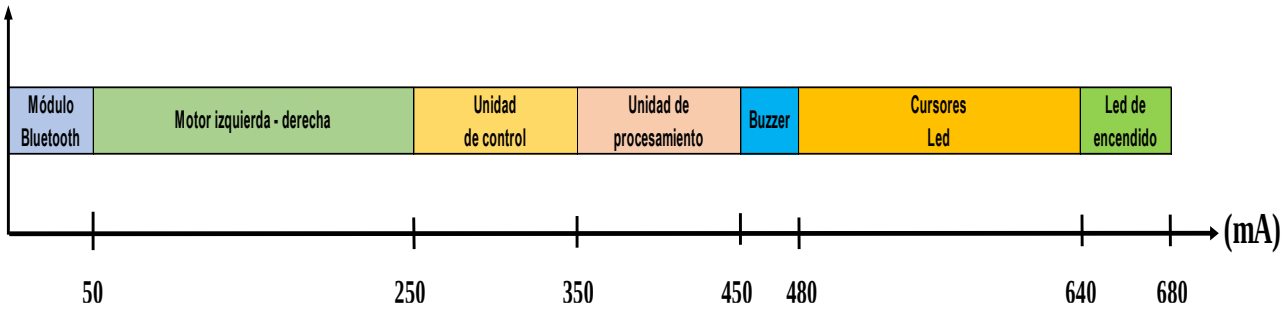
Alternativas de modulo bluetooth

Características	Requerimientos	Alternativas	
		HC-05	HC-06
Voltaje de operación	-	3.3 - 6 V	3.3 - 6 V
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)	-	27x12.7	27x12.7
Disponibilidad	-	Local	Local
Precio	-	**S/. 30.00**	**S/. 25.00**

4.1.1.7. Fuente de Voltaje. El sistema requiere una fuente de voltaje que se determinará basándose en el consumo total de corriente de todos los componentes. Para calcular esto, se revisó el consumo individual de cada componente según sus hojas de especificaciones técnicas (datasheets), y la suma total de estos consumos se presenta de manera gráfica en la Figura 31. Este cálculo es crucial para garantizar que la fuente de alimentación pueda suministrar la energía necesaria para el funcionamiento óptimo de todo el sistema.

Figura 31

Consumo de corriente del actuador robot movil



La batería seleccionada para el sistema es el modelo 18650 de iones de litio, ampliamente disponible en el mercado local. Como se detalla en la tabla 14, las especificaciones técnicas de esta batería cumplen satisfactoriamente con los requerimientos de consumo de corriente calculados anteriormente (mostrados en la figura 31). Las características principales de esta batería la hacen ideal para alimentar todos los componentes del sistema de manera óptima.

Tabla 14

Muestra las principales características de la batería

Características	**Modelo 18650.**
Voltaje	3.7 V.
Corriente	1980 mAh.
Peso	45.5 gr.
Dimensiones (mm)	18.33x68.45
Disponibilidad	Mercado local

Precio	**S/. 40.00**

En cuanto al cálculo de la autonomía de las baterías, que usamos:

$$\frac{W_B}{W_C} = \frac{I_B x V_B}{I_C x V_C} = H \qquad (21)$$

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 una batería (3.7 V), la potencia W_B es de 7,326 mW, considerando una eficiencia de 80% y la potencia W_C es de 5,170 mW. Esto implica que dos baterías en serie durarían un aproximado de 1.41 horas, en conclusión, necesitamos 2 baterías independientes para realizar el cambio.

Si se vuelve a realizar el cálculo para estas 2 baterías, la duración de las baterías en el robot es de, aproximadamente, 2.82 horas (aproximado a 3 h) lo cual cumple con la lista de exigencias de la Tabla 3.

4.1.1.8. Regulador de Voltaje. El sistema utiliza un conversor de voltaje DC-DC Step Up modelo MT3608, que cumple dos funciones importantes:

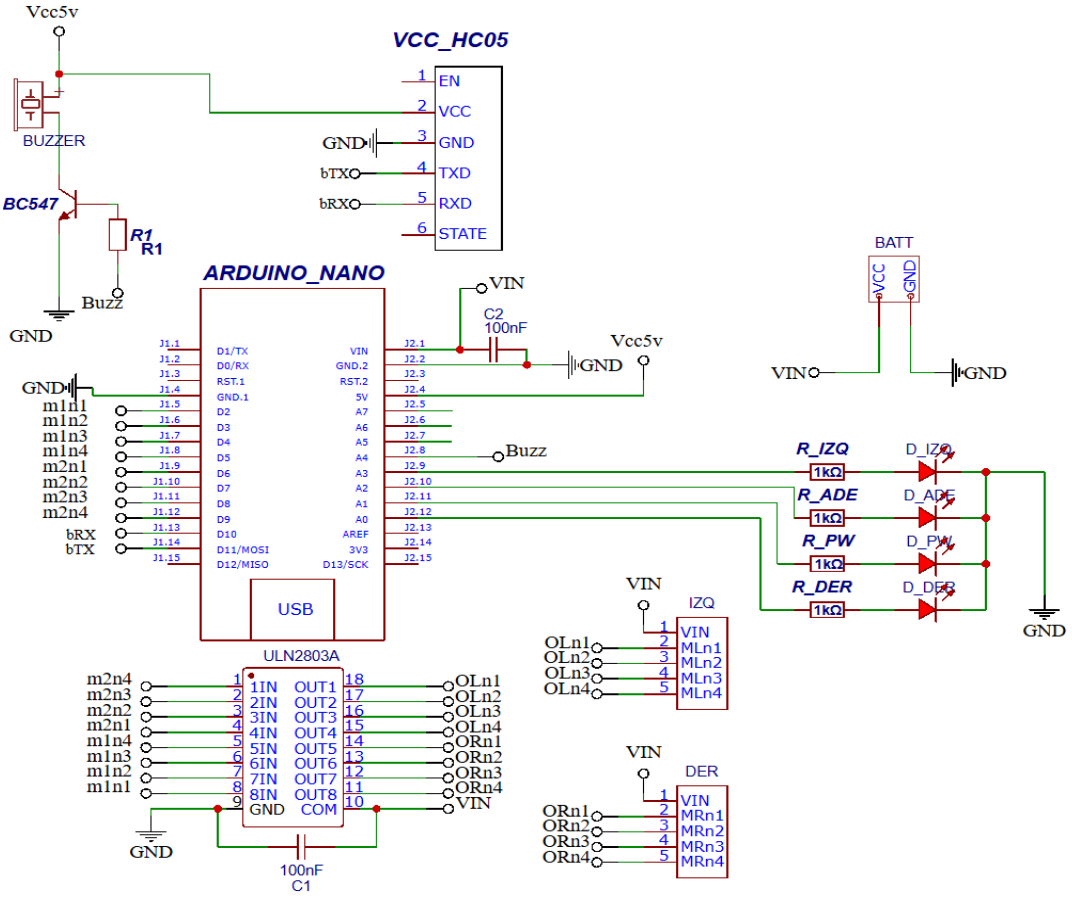
- ✓ Eleva el voltaje de la batería de 3.7V (proporcionado por una batería 18650) a un voltaje estable de 5V, que es el voltaje de trabajo requerido.
- ✓ Es compatible con el sistema ya que puede operar desde un voltaje mínimo de entrada de 2.0V y puede regular voltajes en un rango de 5V hasta 28V, en nuestro caso fijarlo en 5.0V.

Este componente es esencial para garantizar que todos los elementos del sistema reciban el voltaje adecuado y constante de 5V para su correcto funcionamiento.

4.1.1.9. Diagrama esquemático del robot móvil. El diseño del circuito PCB del sistema fue desarrollado utilizando el software EasyEDA. Este proceso incluyó la creación del diagrama esquemático con todos los componentes previamente seleccionados y analizados, el cual puede ser consultado en el Anexo A. Adicionalmente, se elaboraron dos elementos clave: el diagrama de conexiones y el diseño del circuito impreso, disponibles en el Anexo B y Anexo C respectivamente. Para una mejor visualización del resultado final, se presenta un modelo en 3D del circuito en la Figura 32.

Figura 32

Diagrama del circuito electrónico del robot móvil

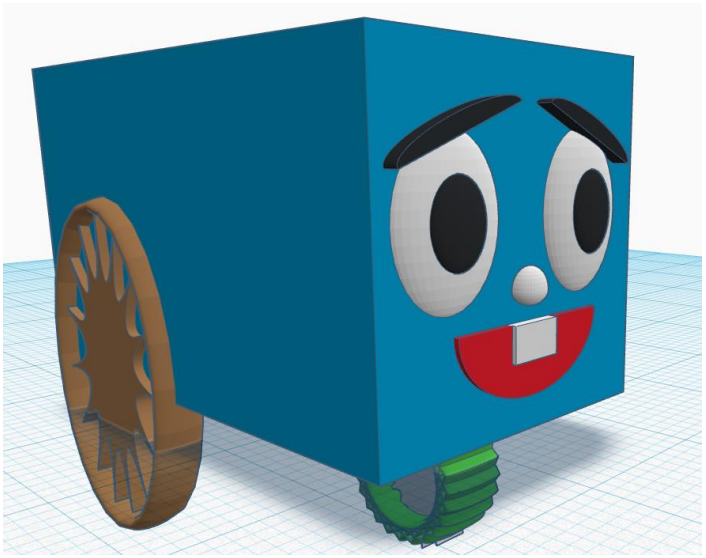


4.1.2. Diseño mecánico del robot móvil.

El chasis del robot móvil intérprete fue modelado siguiendo el diseño establecido en el boceto de la alternativa 2, que fue la solución seleccionada. Se determinó usar PLA como material para la impresión 3D, ya que cumple con los requerimientos de seguridad necesarios. El diseño final respeta todas las especificaciones técnicas establecidas, como las dimensiones máximas permitidas y la ausencia de bordes filosos, asegurando así un chasis seguro para la manipulación de los estudiantes.

Figura 33

Modelado 3D del chasis del interprete robot móvil

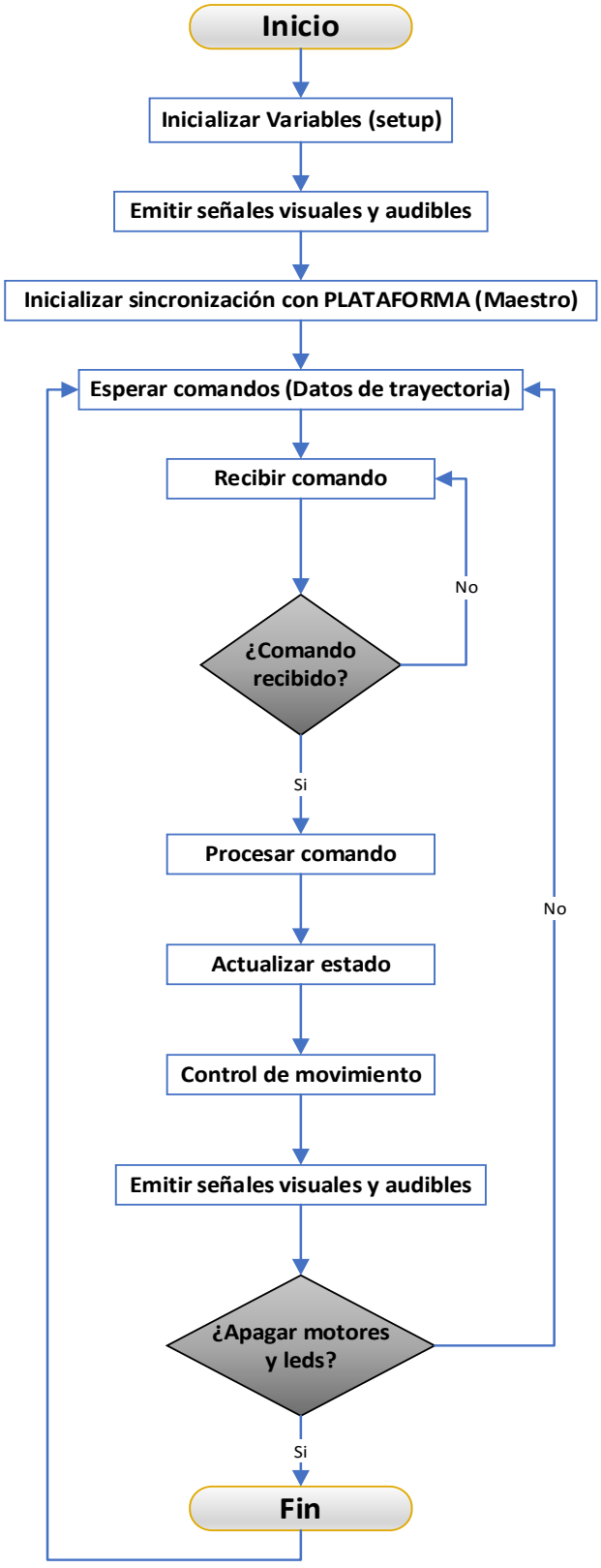


4.1.3. Diseño del software del interprete Robot Móvil.

El software del interprete robot móvil es desarrollado para un microcontrolador arduino nano, utilizando un lenguaje de programación basado en C++. El funcionamiento completo del programa se puede visualizar en el diagrama de flujo presentado en la Figura 34, donde se detalla la lógica y secuencia de operaciones que ejecutará el robot.

Figura 34

Diagrama de flujo del interprete robot móvil



4.2. Diseño del hardware entorno de programación.

En esta sección, es importante destacar que el diseño del entorno de programación tangible consta de dos partes; un tablero programable, que actúa como interfaz física; y piezas tangibles que representan las instrucciones. El diseño electrónico del tablero permitirá la interacción con las piezas y la comunicación con el software. Las piezas tangibles, a través de su diseño mecánico y codificación, proporcionan una interfaz intuitiva para el usuario. El software, por su parte, es responsable de traducir las acciones físicas del usuario (manipulación de las piezas) en comandos digitales para el robot móvil, cerrando así el ciclo de interacción.

4.2.1. Diseño electrónico del tablero programable

El diseño electrónico del tablero de programación implica una selección rigurosa de componentes que satisfagan las especificaciones técnicas exigidas por la solución propuesta en la alternativa 2. Esta selección abarca desde el microcontrolador hasta los sensores, actuadores y fuentes de alimentación, considerando factores como el rendimiento, tamaño, consumo energético.

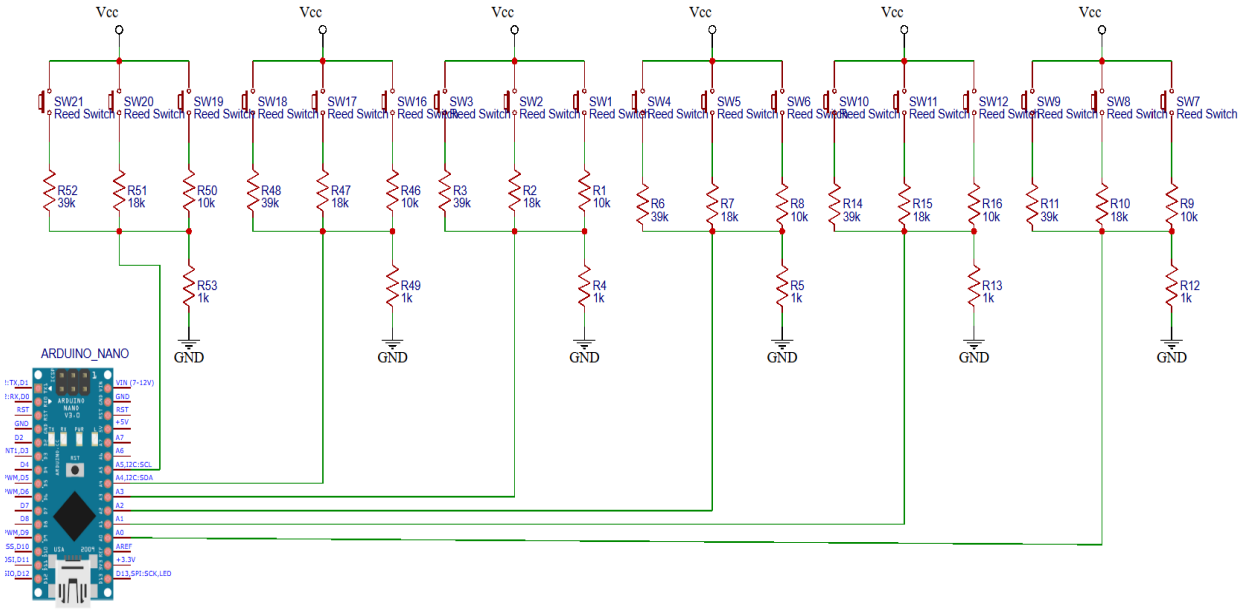
4.2.1.1. Slots y/o lector de piezas tangibles de programación. El sistema de programación tangible implementa una interfaz física que consta de 6 slots de entrada, cada uno integrando Reed Switches como sensores de detección magnética para las piezas de programación. La arquitectura electrónica incorpora un arreglo de divisores de tensión configurados con resistencias conectadas a los Reed Switches, cuyas salidas analógicas son procesadas por el convertidor analógico-digital (ADC) interno del microcontrolador.

Este sistema digitaliza las señales en una codificación de 3 bits (000-111), donde cada combinación binaria representa una instrucción específica en el protocolo de control. Los datos procesados son transmitidos mediante comunicación inalámbrica Bluetooth al sistema actuador, el cual decodifica la secuencia de instrucciones para ejecutar el control cinemático de los motores y

la activación sincronizada de los periféricos audiovisuales según la secuencia programada físicamente en el tablero. A continuación, se muestra la figura 35 en donde se muestra la conexión entre el microcontrolador y el arreglo de divisores de tensión con resistencias y Reed Switch.

Figura 35

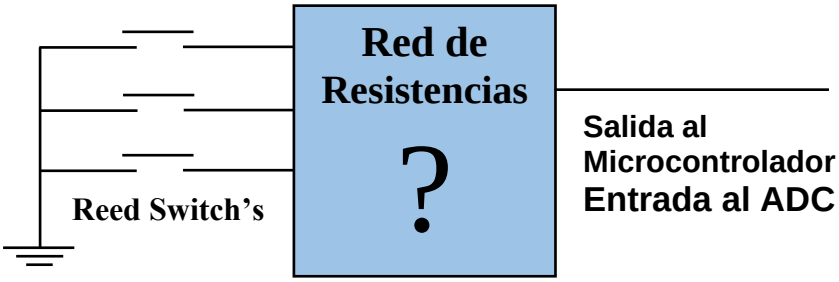
Arreglo de resistencias y Reed Switch con el microcontrolador para 6 slot



4.2.1.2. Cálculo de valores de las resistencias y diseño del ADC. El diseño implementa una red de divisores de tensión que integra una matriz de 3 Reed Switches por pieza de programación. La configuración genera 2^3 (8) combinaciones, donde cada combinación representa una instrucción única. La arquitectura del circuito utiliza un arreglo de resistencias que, en combinación con el estado binario de los Reed Switches (activados por los imanes de las piezas tangibles), produce voltajes de referencia equidistantes. Estos niveles analógicos son digitalizados por el ADC del microcontrolador, permitiendo una alta resolución entre estados y una codificación precisa de las instrucciones programadas físicamente, optimizando la fiabilidad en la detección de comandos y eliminación de ruidos.

Figura 36

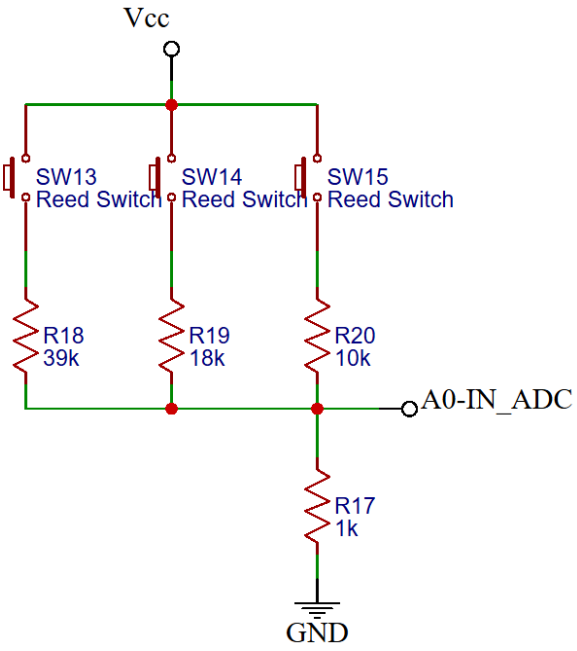
Arreglo de resistencias y Reed Switch con el microcontrolador



Nota: Bloque que representa el arreglo de resistencias conjuntamente con los Reed Swtich solamente para una entrada ADC, en nuestro caso se usaran 4 bloques para cuatro piezas tangibles de programación.

Figura 37

Representación Eléctrica del arreglo de resistencias y Reed Switch



Como se puede observar en la figura 37, la red de divisores de tensión, compuesta por resistencias y Reed Switches, genera diferentes voltajes que son convertidos por el ADC del microcontrolador entre 0 y 1023 niveles discretos, permitiendo identificar cada instrucción de programación de manera única y precisa.

Y como se mencionó anteriormente, la finalidad es lograr que estas combinaciones estén espaciadas uniformemente entre VCC y GND. Esta distribución uniforme implica que las combinaciones deben ser equitativas y, en la medida de lo posible, lineales. Esto significa que los niveles de voltaje deben variar de manera constante a lo largo del rango.

En la imagen de referencia implementamos un diseño modular que replica un arreglo estandarizado de tres Reed Switches con sus respectivos divisores de tensión para cada entrada del ADC. Esta configuración única está optimizada para detectar una sola pieza tangible de programación por slot, permitiendo una lectura precisa de las instrucciones codificadas en cada slot del tablero. Las tres Reed Switch, junto con las resistencias en serie, forman parte del circuito, y la resistencia R17 completa el divisor de tensión necesario para la conversión, del mismo modo el voltaje de referencia para el ADC se establece en 1.1 voltios según las especificaciones técnicas para microcontrolador arduino nano, esto permite aprovechar la resolución completa de 10 bits (2^10 = 1024), pero como empezamos desde 0, se dice que el valor entero toma valores entre 0 a 1023 de 1024 valores del ADC. Se ha observado que al utilizar un valor bajo para R17 (en este caso, 1 kΩ), se mejora la linealidad de los resultados, aunque esto implica una reducción en la oscilación del voltaje real. Esta configuración asegura una medición más precisa y estable de las señales generadas por los Reed Switch al ser activados por un campo magnético.

A continuación, se observa las siguientes figuras, donde se realizó una tabla mediante un cálculo manual para ver la linealidad de la lectura ADC, para realizar esta tabla se utilizó la una proporción de conversión para la lectura del ADC y que a continuación se indica:

$$\frac{Resolucion\ del\ ADC}{V\_referencia} = \frac{Lectura\ del\ ADC}{V\_medido}$$

Donde;

- Resolución del ADC; considerando que es para 10 bit para el Arduino entonces es de 1023 valores posibles en el rango de 0 a 5 voltios.
- V_referencia; para los procesadores ATMega 328 se considera la referencia interna de 1.1 voltios, sin embargo, también se podría utilizar el voltaje de referencia externa utilizando resistencias para su respectivo divisor de tensión, pero para nuestro caso se utiliza la referencia interna propio del microprocesador.
- V_medido; es el valor obtenido de divisor de tensión de la red resistiva y los Reed Swtiches, es decir la tensión analógica de salida.
- Lectura del ADC; es el valor del ADC equivalente y proporcional del valor del voltaje analógico medido.

  Entonces para hallar el valor de la Lectura del ADC tenemos:

$$Lectura\ del\ ADC = \frac{1023}{V\_referencia}\ x\ V\_medido$$

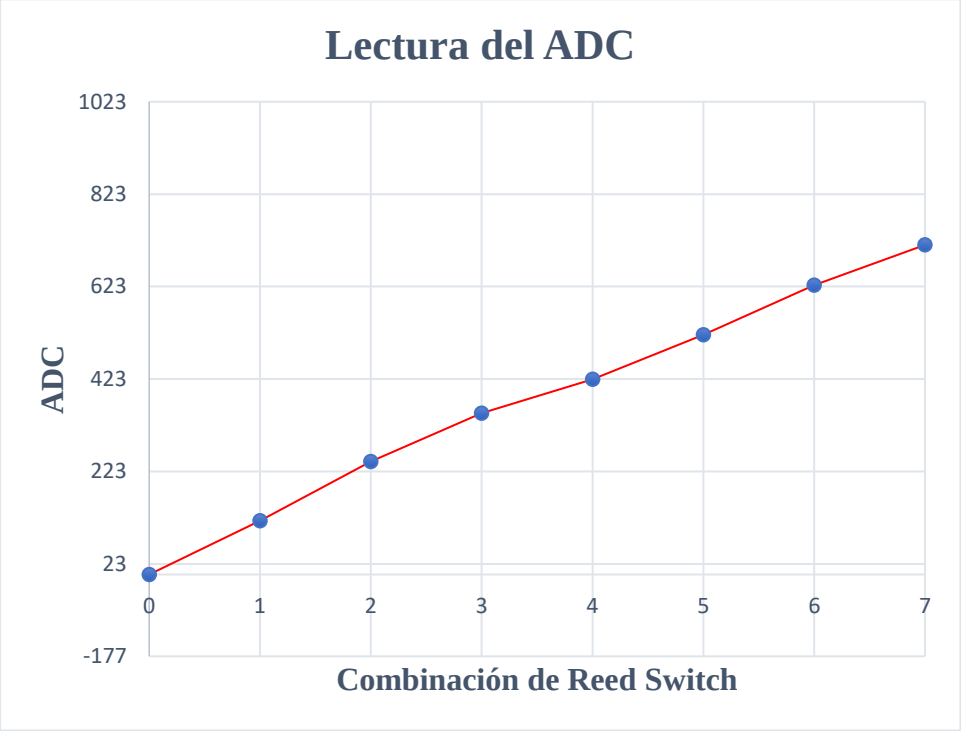
Figura 38

Hoja de calculo del ADC en funcion a los valores de Resistencias

Combinacion Reed Switch	R20	R19	R18	R17	R_Equival	Vcc	V_Out	V_Ref_Inter	Resolu	Lectura del ADC
	10k	18k	39 k	1k	Reed Switch	5 Voltios	Valor Medido	1.1 Arduino	ADC	
0	0	0	0	1.00	Abierto	5.00	0.000	1.1	10	0
1	0	0	1	1.00	39	5.00	0.125	1.1	10	116
2	0	1	0	1.00	18	5.00	0.263	1.1	10	245
3	0	1	1	1.00	12.32	5.00	0.375	1.1	10	349
4	1	0	0	1.00	10	5.00	0.455	1.1	10	423
5	1	0	1	1.00	7.96	5.00	0.558	1.1	10	519
6	1	1	0	1.00	6.43	5.00	0.673	1.1	10	626
7	1	1	1	1.00	5.52	5.00	0.767	1.1	10	713

Figura 39

Estados de los Reed Switch frente a los valores o combinaciones



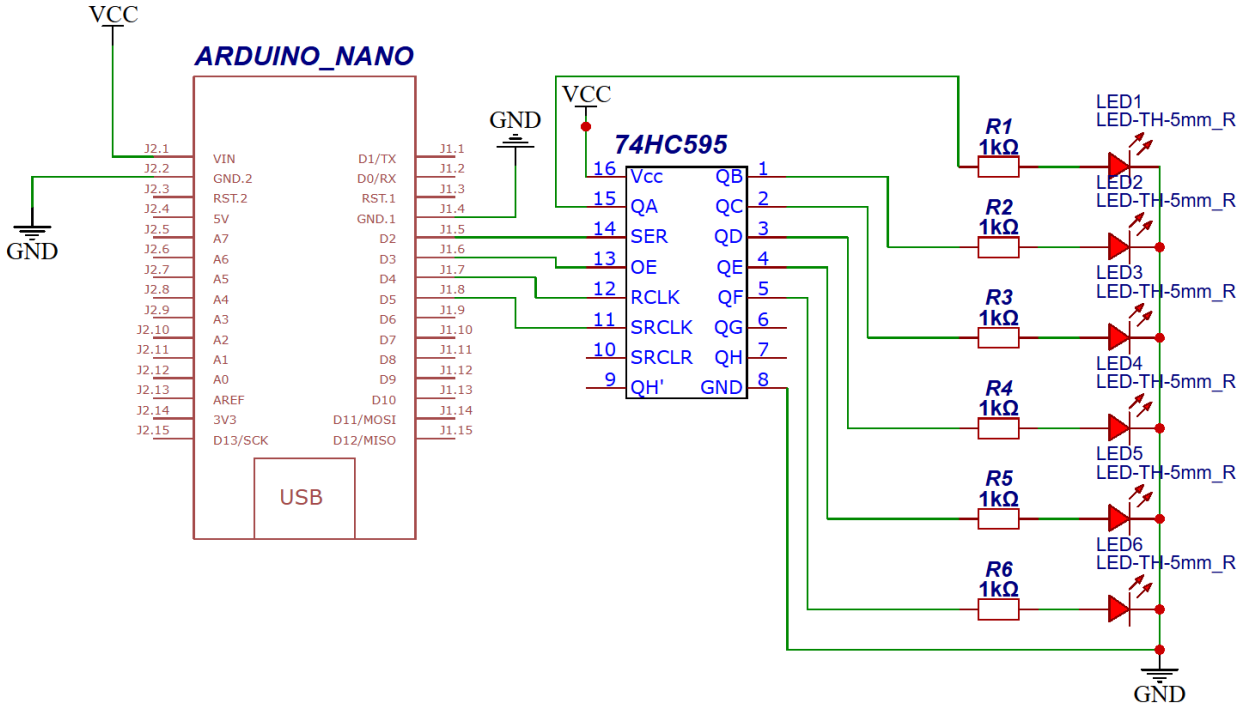
Nota: Se puede observar la respuesta lineal y esto es muy bueno para la correcta distribución de los valores en el ADC.

En el diseño del circuito, se han considerado los valores de las resistencias R17, R18, R19 y R20 basándose en el estudio realizado por HUGHSON, L. (2015). *Read multiple switches using ADC*, https://www.edn.com/read-multiple-switches-using-adc/, que aborda la lectura de múltiples interruptores utilizando un convertidor analógico a digital (ADC). Este enfoque permite que cada interruptor Reed switch contribuya a un divisor de tensión, asegurando que las combinaciones de resistencias generen voltajes uniformemente distribuidos entre VCC y GND. La elección de resistencias adecuadas es de suma importancia para lograr una linealidad óptima en las lecturas del ADC, facilitando así la detección precisa de los estados de los interruptores. La implementación de este método asegura que se maximice el rango de entrada del ADC y se minimicen los errores en la conversión.

4.2.1.3. Panel indicador de lectura de piezas tangibles de programación. El circuito del indicador de lectura de las piezas tangibles está diseñado para proporcionar una señal luminosa mediante un panel de LEDs, que indica qué tarjeta o pieza tangible se está leyendo durante el proceso de decodificación. El circuito interconecta el microcontrolador Arduino nano con un registro de desplazamiento 74HC595, que desplaza los datos a medida que se ejecuta la lectura de cada pieza tangible. Las señales luminosas acompañan el procedimiento de decodificación, indicando el slot y la tarjeta en proceso de lectura, facilitando así la interacción y el control del sistema, la figura 40 grafica la interconexión entre en microcontrolador y el CI de desplazamiento.

Figura 40

Circuito indicador de lector de 06 piezas tangibles de programación

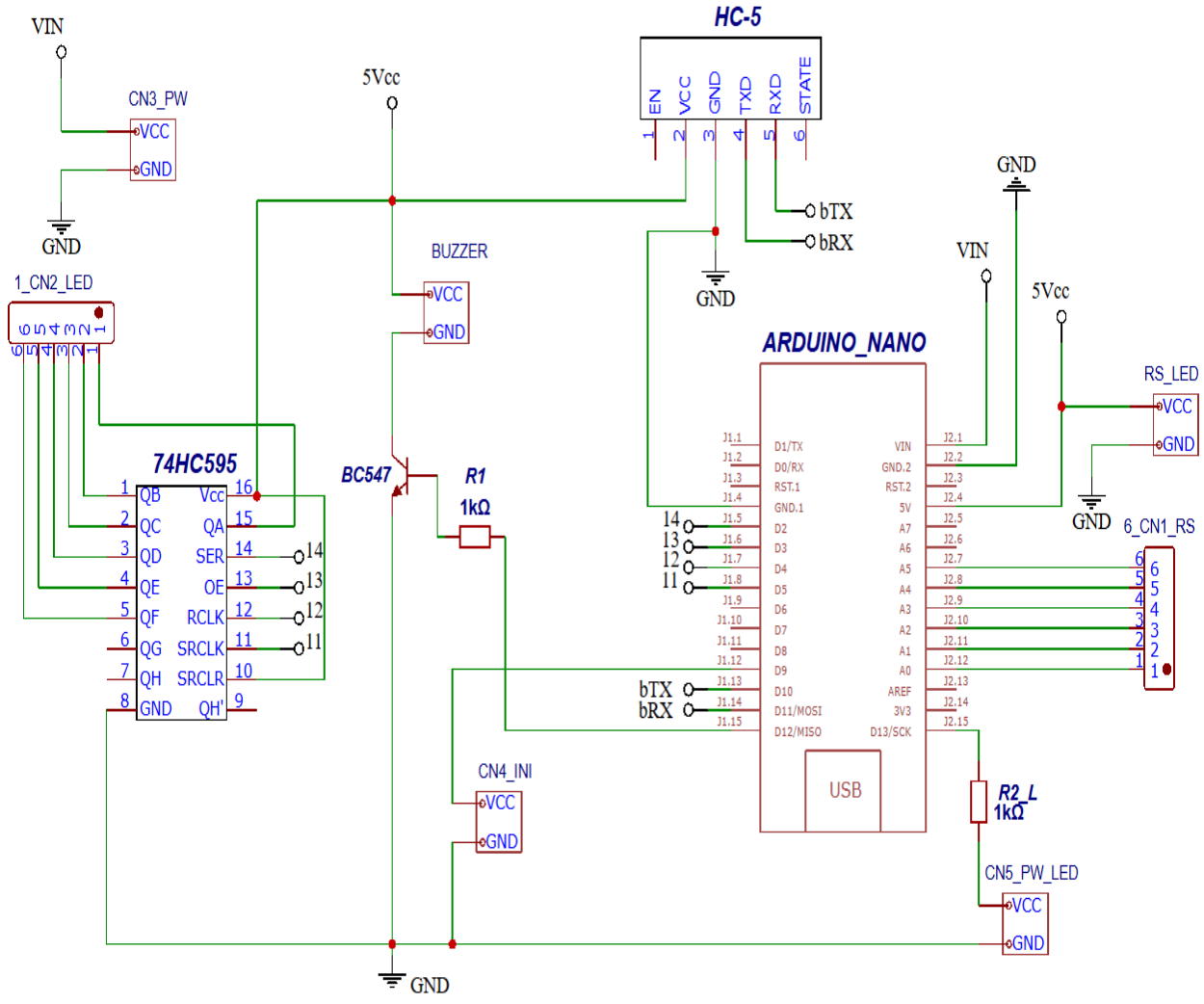


El diagrama esquemático final integra para cada uno de los 6 slots: los arreglos de divisores de tensión con Reed Switches para la detección de piezas, el sistema de LEDs indicadores

controlados por el registro 74HC595, y el microcontrolador que procesa las señales y transmite las instrucciones vía Bluetooth al robot móvil, ver figura 41.

Figura 41

Circuito final del tablero programable



4.2.2. Diseño mecánico de las piezas de programación tangible

El diseño de las piezas de programación tangible implementa una mecánica utilizando imanes de neodimio (NdFeB) de 5mm x 3mm como elementos clave de codificación. Cada pieza utiliza una configuración específica de tres imanes que interactúan con los Reed Switches del tablero, emulando un código binario de 3 bits (permitiendo 2^3 combinaciones posibles).

El principio de funcionamiento se basa en la interacción electromagnética entre:

- Los imanes de neodimio, que generan un campo magnético debido a su aleación de neodimio-hierro-boro
- Los Reed Switches, que contienen láminas ferromagnéticas que se cierran en presencia del campo magnético, actuando como interruptores que abren y cierran el circuito del divisor de tensión.

Esta configuración permite codificar cada instrucción de manera única mediante diferentes patrones de colocación de los imanes, donde la presencia o ausencia del imán en cada posición representa un valor como resultado del divisor de tensión, que posteriormente es interpretado por el microcontrolador Arduino como una instrucción específica para ser enviada al robot actuador.

Figura 42

Codificación de la pieza tangible de programación

CODIFICACION DE LAS PIEZAS TANGIBLES

Valor Decimal	Valor Binario			Codificación o Instrucción
0	0	0	0	SIN VALOR
1	0	0	1	ADELANTE
2	0	1	0	SIN VALOR
3	0	1	1	SIN VALOR
4	1	0	0	DERECHA
5	1	0	1	IZQUIERDA
6	1	1	0	SIN VALOR
7	1	1	1	SIN VALOR

De la misma manera en la siguiente figura se muestra el diseño de la pieza tangible de programación en 3D, en donde:

Vista Frontal; muestra la interfaz o cara principal de la pieza de programación que interactúa con el usuario. En esta superficie se puede identificar visualmente qué instrucción representa la pieza (por ejemplo, avanzar, girar, etc.).

Vista Posterior; Se observan las cavidades específicamente diseñadas para alojar los imanes de neodimio de 5mm x 3mm. La ubicación de estos imanes sigue un patrón preciso que genera un código binario de 3 bits único, el cual será leído por los Reed Switches del tablero para identificar la instrucción correspondiente.

Figura 43

Diseño 3D de la codificación de la pieza tangible de programación

4.2.3. Diseño electrónico final del entorno de programación y elección de componentes

Una vez completado el análisis y el diseño electrónico, el siguiente paso es la selección de los componentes. Posteriormente, se presentará el diagrama electrónico final y el diseño mediante una representación en 3D del entorno de programación.

La elección de los componentes se basó en los requisitos establecidos en la Tabla 3. Se analizaron los requerimientos básicos de cada componente y se evaluaron las diferentes opciones disponibles en el mercado actual.

4.2.3.1. Unidad de procesamiento. La unidad de procesamiento tendrá la función de controlar el flujo de datos de las entradas analógicas y procesarlos de acuerdo a la secuencia de las piezas tangibles de programación ordenadas por el usuario, se utiliza un microcontrolador Arduino cuya función es de interpretar, procesar y enviar una secuencia de instrucciones codificadas, y para llevar a cabo estas tareas, el microcontrolador empleará un conversor analógico-digital (ADC) interno, este conversor ADC procesará los valores de tensión obtenidos a través de un arreglo de resistencias configuradas como divisor de tensión.

Los valores de tensión serán generados por los sensores Reed Switch. Estos sensores detectarán las piezas tangibles colocadas en los slots del tablero de programación, permitiendo el paso de corriente y generando diferentes valores en el divisor de tensión.

Finalmente, la unidad de procesamiento interpretará estas diferentes tensiones del divisor de tensión y generará una secuencia de instrucciones. Estas instrucciones se enviarán de manera inalámbrica (vía Bluetooth) al intérprete (robot móvil), acompañadas de señales visuales y audibles. Para este proyecto, se realizará una comparación entre diferentes microcontroladores que puedan cumplir con esta función. La cantidad de periféricos conectados a la unidad de control se especificará en la Tabla 15, donde se detallará el número de pines necesarios para cada periférico.

Tabla 15

Periféricos para 4 piezas tangibles de programación

Periférico	**Nro de entradas y salidas.**
Registro de desplazamiento LED	4
Lector de piezas tangibles ADC	4
Pulsador de inicialización	1
Led de encendido general	1
Altavoz	1
Modulo Bluetooth	2
Indicador de nivel de batería	3
Entrada de señal de nivel de batería	1
Total	**17**

Se necesitan 17 pines disponibles para entradas y salidas digitales y analógicas, 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.

Tabla 16

Comparación de microcontroladores adecuados

Características	Requerimientos	Alternativas	
		Arduino UNO	**Arduino NANO**
Procesador	----------	ATMEGA 328	ATMEGA 328
Velocidad de reloj	>= 16MHz	16MHz	16MHz
Memoria interna	>=32Kb	32Kb	32Kb
Comunicación inalámbrica	Bluetooth	No	No
Dimensiones (mm)	<=50x10mm (largo X ancho)	68x53mm	45x18mm
Voltaje de alimentación	----------	7-12 Volt	7-12 Volt
Disponibilidad	----------	Mercado local	Mercado local
Precio	----------	**S/. 129.00**	**S/.91.00**

Bajo las consideraciones de la Tabla 16, se optará por un *Microcontrolador Arduino Nano*. Las especificaciones técnicas de este dispositivo cumplen con los requerimientos de diseño y el precio es el más bajo. 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.

4.2.3.2. Registro de desplazamiento. El registro de desplazamiento se implementa para expandir número de salidas digitales del microcontrolador. El sistema está diseñado para manejar 6 piezas de programación tangible, requiriendo por tanto 6 salidas digitales adicionales. Debido a las limitaciones de pines del microcontrolador, se utiliza un registro de desplazamiento de 6 bits que convierte una entrada serial en salidas paralelas, permitiendo controlar los 6 indicadores LED correspondientes a cada slot de lectura de las piezas tangibles de programación.

Tabla 17

Comparación de registros de desplazamiento adecuados

Características	Requerimientos	Alternativas	
		74HC595	**74F675**
Número de Bits	= 6 bits	8 bits	16 bits
Velocidad de reloj	>= 100 MHz	200 – 450 Mhz	100 – 450 Mhz
Función	Serie a Paralelo	SIPO	SIPO
Tipo Lógico	CMOS	CMOS	CMOS
Corriente de alimentación	----------	5 mA	5 mA
Voltaje de alimentación	5 V	2-6 V	4.5 – 5.5 V
Disponibilidad	----------	Mercado local	Mercado local
Precio	----------	**S/. 5.00**	**S/.9.00**

4.2.3.3. Sensores Reed Switch. El Reed Switch es un interruptor electromagnético que consiste en dos láminas ferromagnéticas encapsuladas en un tubo de vidrio sellado. Su funcionamiento se basa en el siguiente principio:

Estado Normal:

- Las láminas permanecen separadas
- El circuito está abierto (no conduce)

Operación Magnética:

- Al detectar un campo magnético suficiente, las láminas se magnetizan
- Las láminas se atraen entre sí, cerrando el circuito
- Al retirar el campo magnético, las láminas retornan a su posición inicial

Este componente actúa como sensor binario en el sistema, detectando la presencia de los imanes de neodimio en las piezas de programación tangible, permitiendo la codificación de las instrucciones mediante la activación selectiva de múltiples Reed Switches.

Las características técnicas específicas del Reed Switch utilizado se detallan en la tabla mencionada.

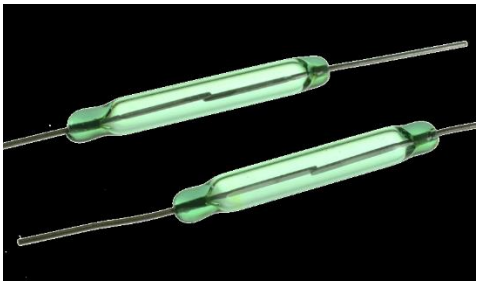
Tabla 18

Especificaciones técnicas del sensor Reed Switch

Características	Tipo Reed Switch
Mecanismo	Magnético Normalmente Abierto
Configuración	SPST Unipolar de un solo tiro
Voltaje nominal	5 V
Rompimiento de voltaje máximo	250 V
Corriente Máxima	500 mA
Campo Magnético necesario	10-30 mT
Tiempo de operación	0.6 ms como máximo
Tiempo de liberación	0.1 ms como máximo
Vida útil	1 a 200 millones de operaciones
Longitud de vidrio	27 mm
Disponibilidad	Mercado Local
Precio	**S/. 50.00**

Figura 44

Reed Switch



4.2.3.4. Buzzer. En cuanto a la selección del Buzzer se considera las mismas características ya explicadas y calculadas en la tabla 13 de la parte del actuador robot móvil.

Tabla 19

Alternativas de altavoz

Características		Buzzer Zumbador
Rango de frecuencias	**> 20 Hz**	2 KHz
Potencia	**>= 0.15 W**	0.099-0.15 W
Impedancia	**-**	8 ohm
Dimensiones (mm)	**-**	12 diámetro
Disponibilidad	**-**	Mercado Local
Precio		**S/. 10.00**

4.2.3.5. Unidad de 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 la tabla 20.

Tabla 20

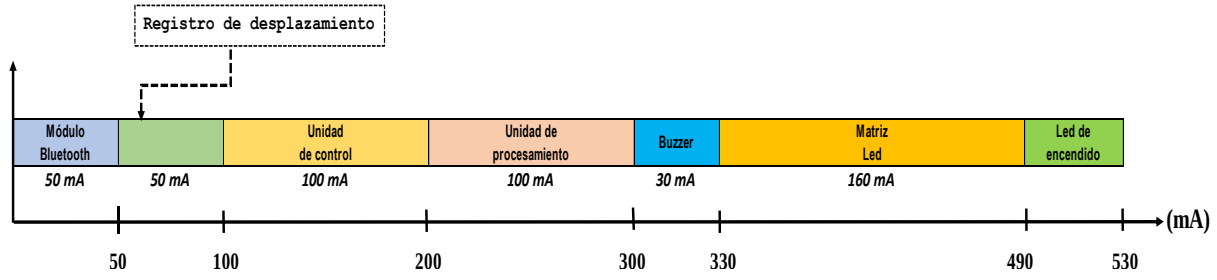
Alternativas de modulo bluetooth

Características	Requerimientos	Alternativas	
		HC-05	**HC-06**
Voltaje de operación	-	3.3 - 6 V	3.3 - 6 V
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)	-	27x12.7	27x12.7
Disponibilidad	-	Local	Local
Precio	-	**S/. 30.00**	**S/. 25.00**

4.2.3.6. 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 45 muestra el consumo de corriente total aproximado.

Figura 45

Consumo de corriente del entorno de programación



De la misma manera mostramos a continuación la tabla 22 en donde se menciona las características de la fuente o batería que utilizaremos para el circuito global del entorno de programación.

Tabla 21

Muestra las principales características de la batería

Características	Modelo 18650
Voltaje	3.7 V
Corriente	1980 mAh
Peso	45.5 gr.
Dimensiones (mm)	18.33x68.45
Disponibilidad	Importación
Precio	S/. 40.00 + envío.

En cuanto al cálculo de la autonomía de las baterías, que usamos:

$$\frac{W_B}{W_C} = \frac{I_B x V_B}{I_C x V_C} = H \qquad (21)$$

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 una batería (3.7 V), la potencia W_B es de 7,326 mW, considerando una eficiencia de 80% y la potencia W_C es de 2,225 mW. Esto implica que dos baterías en serie durarían un aproximado de 3.29 horas (equivalente a 4 horas pedagógicas considerando la hora pedagógica 45 min.), en conclusión, necesitamos solamente 01 batería para

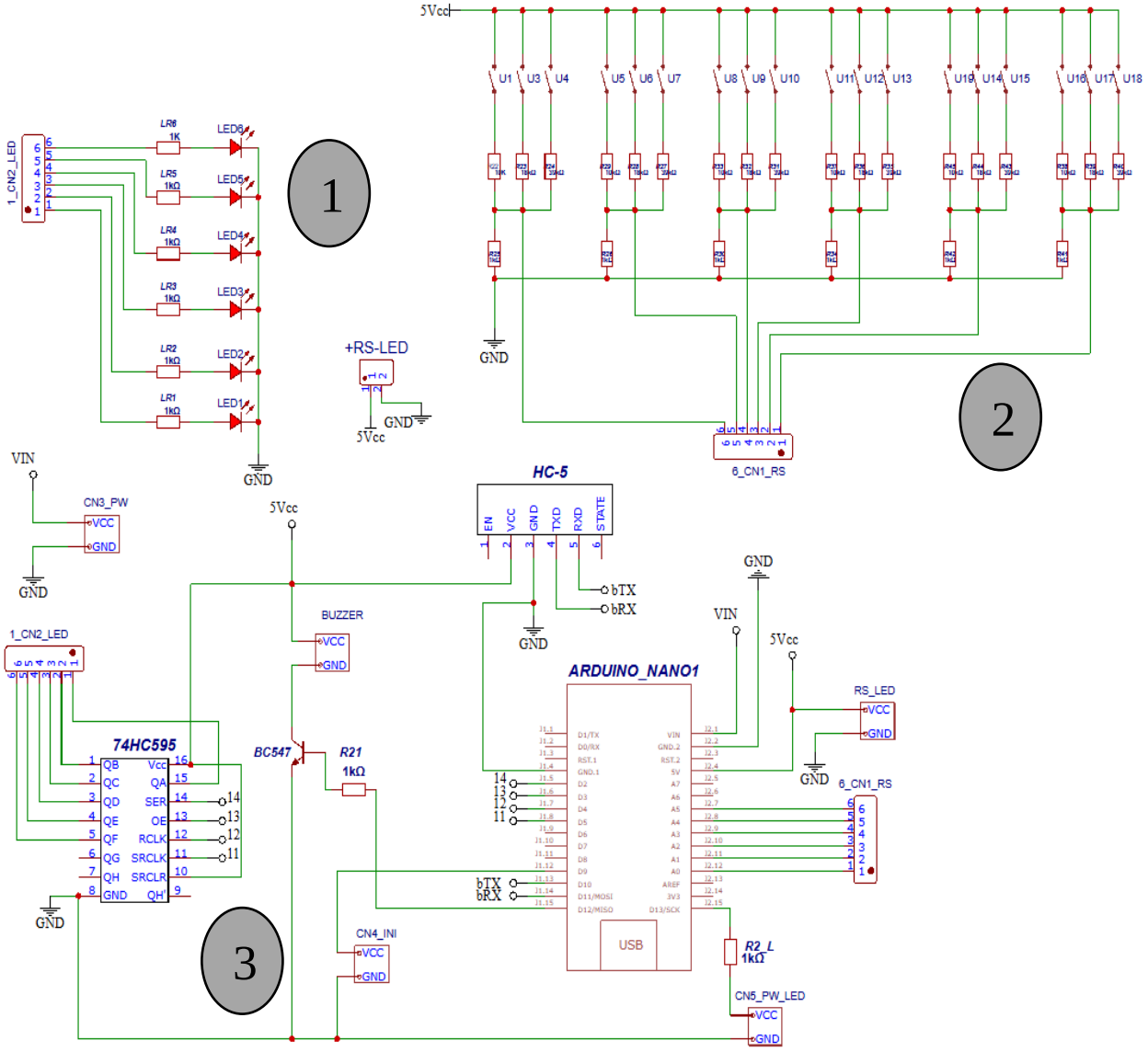
el sistema electrónico del entorno de programación, asimismo indicamos que se cumple con la lista de exigencias de la Tabla 3.

4.2.3.7. Regulador de voltaje. El conversor DC-DC Step Up MT3608 necesita 2.0 V como mínimo y, debido a que se usará 1 batería (3.7 V), se fijará 5 V como tensión de trabajo. Este regulador de voltaje MT3608 permite regular voltajes entre 5 - 28 V y fijarlos en 5 V.

4.2.3.8. Diagrama esquemático final del entorno de programación. El diseño electrónico completo se desarrolló utilizando el software EasyEDA, incorporando todos los componentes previamente seleccionados y analizados. La documentación completa del sistema incluye el diagrama esquemático detallado en el Anexo A, el diagrama electrónico final del entorno de programación en la Figura 51, el diagrama de conexiones en el Anexo D y el diseño del circuito impreso en el Anexo E. Adicionalmente, se generaron visualizaciones 2D y 3D del sistema completo, presentadas en las Figuras 46, 47 y 48, que muestran la implementación física final del circuito.

Figura 46

Diagrama electrónico final del entorno de programación



Nota:

- 1: Bloque matriz Led indicador
- 2: Bloque de detección de piezas tangibles
- 3: Bloque de procesamiento y control

Figura 47

Mod. 2D del PCB, tablero programable-indicador de lectura -ADC y Reed Switch

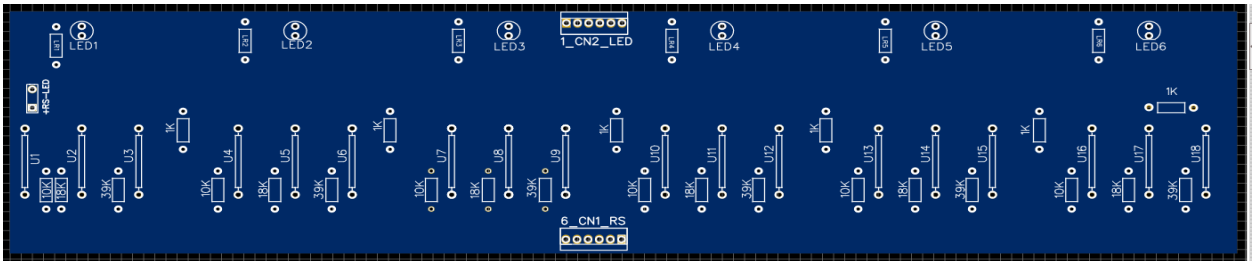


Figura 48

Mod. 3D del PCB, tablero programable-indicador de lectura -ADC y Reed Switch

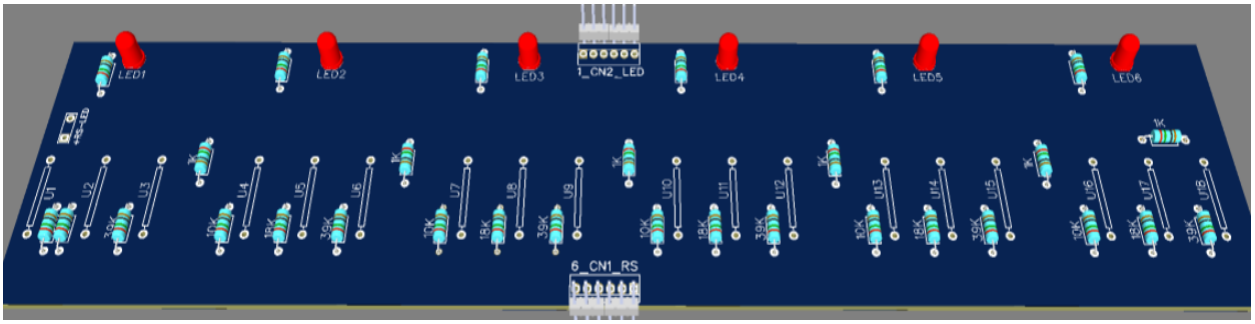
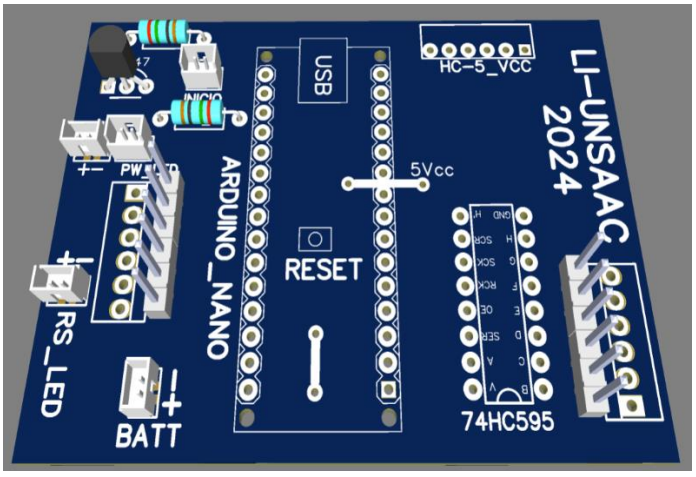


Figura 49

Modelo 3D del PCB del entorno de programación

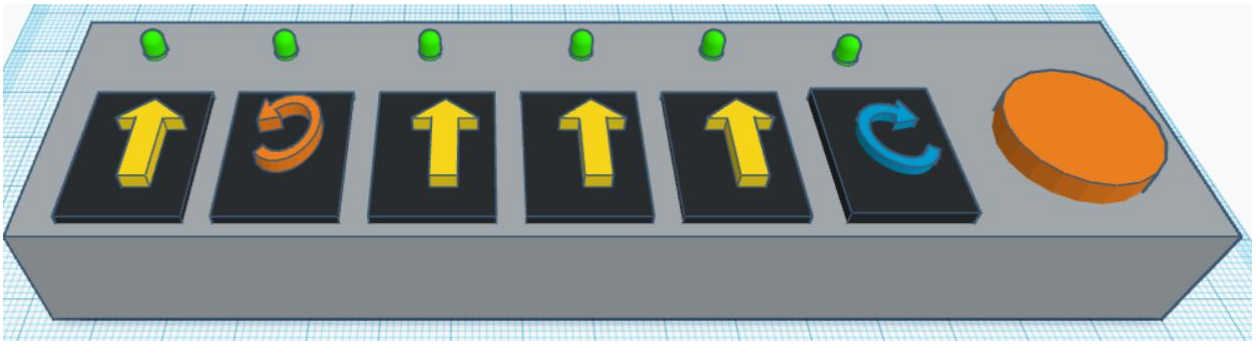


4.2.4. Diseño mecánico del entorno de programación.

Se realizó el modelado 3D del chasis del entorno de programación, 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, ver la figura 50.

Figura 50

Representación 3D del entorno de programación

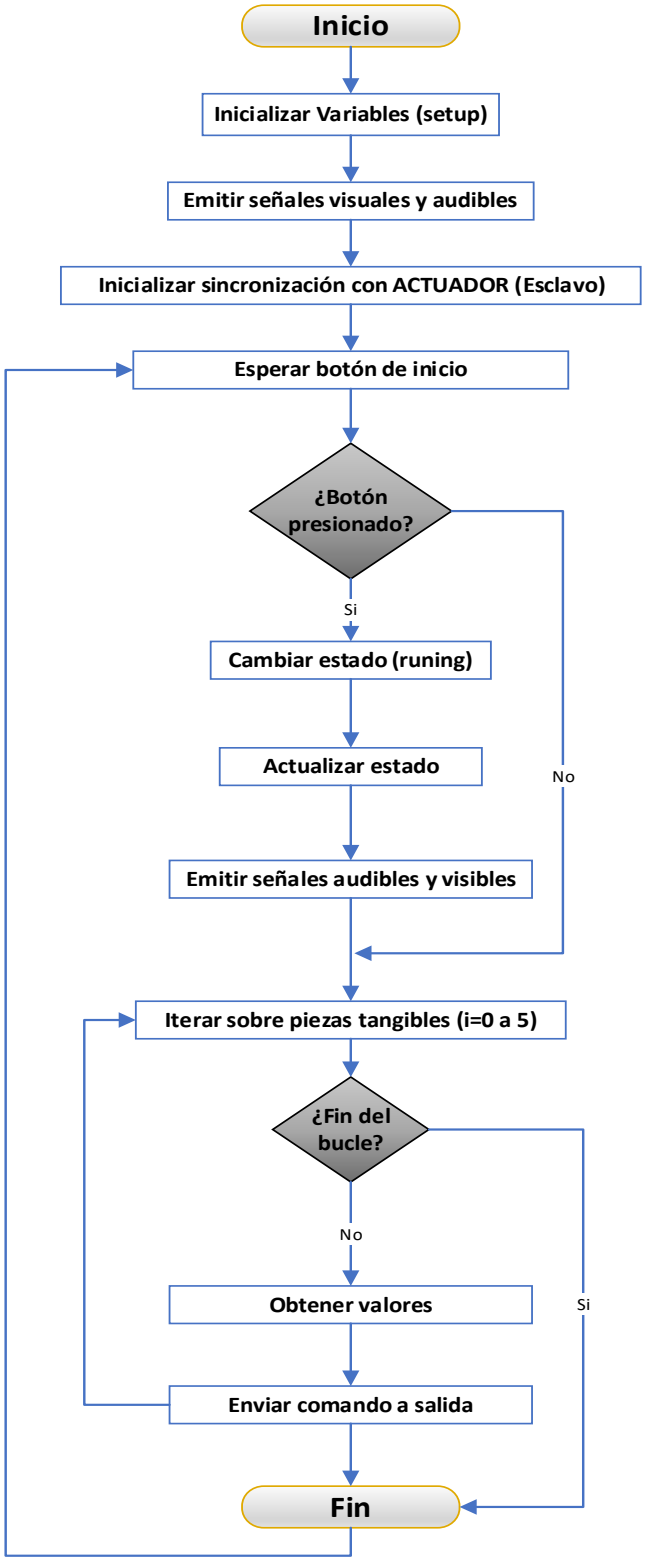


4.2.5. Diseño del software del entorno de programación

El software del entorno de programación tiene como función principal ejecutar los procesos establecidos en la plataforma tangible, facilitando una interacción dinámica con los usuarios. Para lograr esto, el microcontrolador Arduino nano procesa el orden de las piezas tangibles colocadas en el tablero, obteniendo los valores de los Reed Switch a través de un convertidor analógico-digital (ADC). Esto permite generar acciones como AVANZAR, GIRO A LA DERECHA y GIRO A LA IZQUIERDA, que se envían al actuador mediante tecnología Bluetooth. La función del usuario es programar utilizando las piezas tangibles y presionar un botón de inicio para que el robot móvil ejecute la tarea programada. Además, el software crea una trayectoria para el robot basándose en los algoritmos establecidos en el microcontrolador. El diagrama de flujo que ilustra este proceso se puede observar en la Figura 51.

Figura 51

Diagrama de flujos del tablero de programación



CAPÍTULO V: VALIDACIÓN

5.1. Análisis

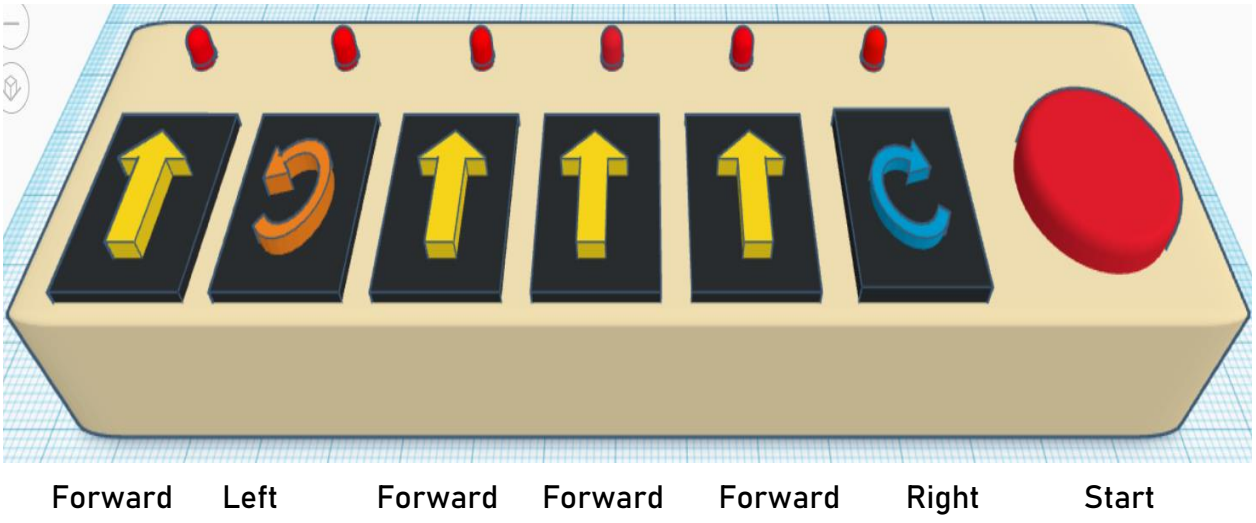
En los capítulos anteriores se diseñó un Lenguaje de Programación Tangible (LPT) simple y accesible para un prototipo electrónico de programación tangible, con el objetivo de utilizarlo como una herramienta pedagógica alternativa que facilite la enseñanza de programación y desarrollo del pensamiento computacional en niños de educación inicial cuyas características principales de este LPT son:

- Utilización de recursos de bajo costo y disponibles en el mercado local.
- Sintaxis basada en símbolos gráficos intuitivos, sin requerir lecto-escritura.
- Cada símbolo se ubica en el centro de una pieza tangible de programación.

Este diseño busca crear un entorno de programación que resulte familiar y accesible para los niños, permitiéndoles aprender conceptos de programación de una manera más práctica y lúdica. La figura 52 muestra un ejemplo de las piezas tangibles con los símbolos gráficos que representan las instrucciones del LPT.

Figura 52

Ejemplo de un programa en LPT



El Lenguaje de Programación Tangible (LPT) diseñado utiliza símbolos gráficos de flechas con diferentes direcciones para representar las acciones básicas que puede ejecutar el intérprete robot móvil. Estas acciones se desarrollan sobre un tablero dividido en cuadrículas, donde cada cuadricula representa una unidad o un movimiento o acción a ejecutar por el robot móvil.

Las tres acciones iniciales definidas para el robot móvil son: avanzar, girar a la izquierda y girar a la derecha. Sin embargo, el LPT permite agregar más instrucciones y subfunciones a medida que aumenta la complejidad o se adapta a diferentes grupos etarios, y en nuestro caso se podría incrementar hasta 2^3 (8) instrucciones posibles.

Un programa tangible (PT) en este lenguaje se construye secuencialmente colocando las piezas físicas con los símbolos gráficos de izquierda a derecha. Cuando este PT se ejecuta, el robot móvil interpretará la secuencia de instrucciones y realizará los movimientos correspondientes en el mundo virtual.

Por ejemplo, en la Figura 52 se muestra un PT compuesto por 6 piezas tangibles que indican al robot móvil: avanzar 1 unidad, girar a la izquierda, avanzar 3 unidades y finalmente girar a la derecha.

Luego de definir el Lenguaje de Programación Tangible (LPT), se diseñó la arquitectura del entorno de programación tangible. Esta arquitectura permite clasificar los componentes del sistema y definir las relaciones entre ellos, los componentes de esta arquitectura son:

- ✓ **Modelo:**

- Lenguaje de Programación Tangible (LPT): Está compuesto por símbolos impresos en piezas físicas, cuya semántica se corresponde con las acciones para manipular el robot móvil.

- Lenguaje Intermedio: Es un lenguaje que actúa como intermediario entre el LPT y el mundo. Su objetivo es desacoplar el LPT de la forma en que se representa el mundo (interfaz de usuario). Se utiliza el lenguaje C++ para programar los microcontroladores, ya que es una notación simple y muy utilizada.

✓ **Vista (Interfaz de Usuario):** La interacción del usuario con el prototipo de la plataforma de programación tangible se realiza a través del tablero de programación y las piezas tangibles. Esta interfaz tiene dos funciones:

- Permitir ordenar secuencialmente las piezas tangibles de programación, definiendo así una trayectoria de movimiento.
- Interpretar el orden y/o secuencia de las piezas tangibles para ejecutar las acciones del robot móvil.

✓ **Controlador:** El controlador tiene tres funcionalidades principales:

- Recibir la secuencia de datos a través de los sensores Reed Switch y el conversor analógico-digital (ADC), y generar el Programa Tangible (PT) en base a ello.
- Traducir el Programa Tangible (PT) al Lenguaje Intermedio (LI).
- Interpretar el Lenguaje Intermedio (LI) para ejecutar las instrucciones en la interfaz de usuario.

El proceso de implementación del Programa Tangible (PT) implica digitalizar el programa utilizando sensores Reed Switch y imanes de neodimio para leer y codificar las piezas tangibles. Se interpreta el contenido del PT y se traduce a un lenguaje intermedio (LI), implementando posteriormente los actuadores del robot móvil para mostrar los movimientos derivados de estas acciones. Este trabajo busca establecer una base para futuras investigaciones, permitiendo extensiones o modificaciones sin cambios complejos en la arquitectura del sistema.

5.2. Validación de la Plataforma Electrónica de Programación Tangible

La validación de la plataforma electrónica de programación tangible es fundamental para evaluar su efectividad en el contexto educativo de la educación inicial tal como se delimito que este estudio será aplicado a estudiantes de edad pre escolar.

La presente investigación es de tipo **aplicativo** con un diseño **descriptivo**, en la cual se considera una única variable independiente de estudio: la "Plataforma Tangible de Programación" y una variable dependiente que es pa percepción de usabilidad en los usuarios. Los indicadores evaluados fueron:

- ✓ **Funcionalidad**: Verificar que la plataforma cumpla con los requerimientos establecidos y realice las operaciones y funciones para las cuales fue diseñada.
- ✓ **Usabilidad**: Evaluar la facilidad de uso y comprensión de la plataforma por parte de los usuarios (niños de educación inicial).
- ✓ **Interacción**: Analizar la interacción entre los usuarios (niños) y la plataforma tangible de programación, observando la experiencia y aceptación.

Para validar estos aspectos, se llevó a cabo un proceso de evaluación en dos etapas:

5.2.1. Proceso de validación

Se implementó un proceso de validación de la plataforma electrónica de programación tangible en la IE. N° 50499 Justo Barrionuevo Álvarez de Oropeza, con 25 alumnos de 6 años separados en grupos de 5 (A, B, C, D, E).

Se aplicó, no sólo la observación directa si no también cuestionarios verbales antes y después del uso del prototipo para medir los indicadores de funcionalidad, usabilidad e interacción.

- ✓ **Cuestionario Previo:** Evaluación del conocimiento previo sobre programación.
- ✓ **Cuestionario Posterior:** Evaluación de funcionalidad, usabilidad e interacción.

5.2.1.1. Evaluación de la funcionabilidad. Se realizaron pruebas para verificar que cada pieza tangible interactuara correctamente con el sistema. Se observó, que, al colocar una ficha de programación en el slot del tablero de programación, se esperaba que el LED correspondiente se encendiera, indicando que la instrucción había sido reconocida. Durante las pruebas iniciales, se registró una tasa de éxito del 80% en la detección de las piezas. Se documentaron fallos en 4 de las 20 interacciones realizadas, lo que llevó a ajustes en el diseño del circuito.

5.2.1.2. Medición de la usabilidad. Se implementó un modelo de test de usabilidad exploratorio y evaluativo, en el cual se solicitó a los participantes que compartieran ideas, opiniones e impresiones emocionales sobre la plataforma tangible. Este modelo de encuesta de usabilidad, que se detalla en el anexo H, se basa en investigaciones previas sobre programación tangible, como el artículo "Tangible Cubes as Programming Objects" de Andrew C. Smith [40]. Asimismo, otros autores, como C. Ramos, J. Rodríguez y L. Cecchi (2020), en su trabajo titulado Plataforma para la programación tangible de la Universidad Nacional de La Plata [41], y Gallardo, D., Julià, C. F., y Jordà, S. (2008) con su artículo "TurTan: Un lenguaje de programación tangible para la exploración creativa" [42].

El test se aplicó a estudiantes y educadores al finalizar la sesión sobre la aplicación del prototipo de programación tangible. La encuesta incluía preguntas sobre la facilidad de uso y la satisfacción general, y los resultados mostraron que el 85% de los estudiantes consideraron que la plataforma era fácil de usar, mientras que el 90% expresó satisfacción con la experiencia. Además, durante las sesiones se realizaron observaciones que indicaron que los estudiantes podían manipular las piezas sin necesidad de ayuda adicional tras una breve introducción y un trabajo colaborativo en grupo.

5.2.1.3. Observaciones de la interacción. Se midió el tiempo promedio que los estudiantes dedicaron a interactuar con las piezas tangibles durante cada sesión. En promedio, los grupos de estudiantes pasaron 5 minutos manipulando las piezas y programando instrucciones. Se registró un aumento del 30% en el número de interacciones por uso a medida que avanzaba el estudio, se determinó dos oportunidades para cada grupo, lo que indica un creciente compromiso con la plataforma. Además, se observó que los estudiantes debatían entre sí sobre cómo programar sus instrucciones, lo que refleja una interacción social positiva.

5.2.2. Resultados cuantitativos.

Los resultados del cuestionario posterior se presentan a continuación:

Indicador	Grupo A	Grupo B	Grupo C	Grupo D	Grupo E	Promedio
Funcionalidad	3	4	3	2	4	3.2
Usabilidad	3	3	4	2	3	3
Interacción	4	4	5	3	4	4

Escala de Medición:
1 = Muy Bajo
2 = Bajo
3 = Regular
4 = Alto
5 = Muy Alto

Figura 53

Gráfico de barras; Resultados de encuestas por indicador

5.2.3. Errores identificados en el prototipo.

Durante la implementación y validación del prototipo, se documentaron varios errores que impactaron su rendimiento:

- **Funcionalidad Incompleta:**
  - ✓ Descripción: Algunas funciones clave no estaban disponibles o no operaban como se esperaba.
  - ✓ Impacto: Esto llevó a confusión entre los alumnos y limitó su capacidad para completar las tareas.
  - ✓ Frecuencia: Reportado por el 30% de los grupos.

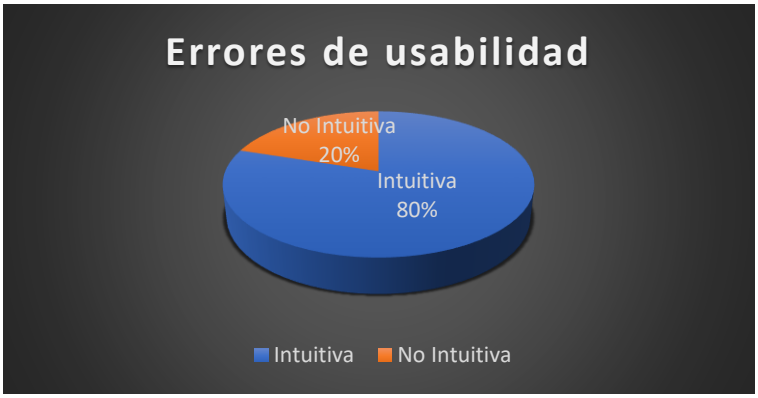
Figura 54

Diagrama circular sobre errores de funcionabilidad

- **Problemas de Usabilidad:**
  - ✓ Descripción: La interfaz para algunos se le hacía difícil insertar correctamente las piezas dentro del slot del tablero de programación.
  - ✓ Impacto: Los estudiantes enfrentaron dificultades con el tablero y piezas de programación porque no tenía en cuenta la orientación física de las coordenadas, sino que se basaba en la percepción del usuario (izquierda, derecha, adelante), siempre que el tablero de programación estuviera frente al operador. Esto generó confusión y provocó pérdida de tiempo para el resto del grupo, afectando así su experiencia general.
  - ✓ Frecuencia: Observado en el 80% de las interacciones es intuitiva.

Figura 55

Diagrama circular sobre errores de usabilidad



- **Interacción Deficiente:**
  - ✓ Descripción: La limitada disponibilidad de un único prototipo impidió la interacción simultánea de todos los grupos, restringiendo su capacidad de experimentación y comprometiendo el aprendizaje colaborativo.
  - ✓ Impacto: Los estudiantes experimentaron frustración al enfrentarse a nuevos elementos tecnológicos en el entorno escolar, donde la falta de familiaridad y comprensión de su funcionamiento generó desánimo y obstaculizó su proceso de aprendizaje.
  - ✓ Frecuencia: Experimentado por el 45% de los alumnos.

Figura 56

Diagrama circular sobre errores de interacción

CAPÍTULO VI: COSTOS Y PRESUPUESTO

En la Tabla 22, se puede observar el listado de todos los componentes necesarios para la construcción de la plataforma de programación tangible seleccionados en la etapa de diseño. El precio final es de 601.00 soles, sin considerar los costes de diseño, lo cual cumple con el requerimiento de costos, pues el precio es menor en un 24% al costo del set WeDo del proyecto "una laptop por niño" cuyo costo fue de 214 dólares americanos (791.00 soles). 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/.)
Microprocesador Arduino Nano	02	0.009	60.00
Batería Recargable Li/Ion Icr18650 - 3300mAh y 3.7V	02	0.195	88.00
Motor PaP 28BYJ-48 5V	02	0.090	36.00
Driver ULN2803A	02	0.022	15.00
Cursor Iluminado, 20 mA.	01	0.004	15.00
Buzzer	02	0.001	12.00
Módulo Bluetooth HC-05	02	0.008	60.00
Regulador de Voltaje Step Up MT3608	02	---	30.00
Resistencias, diodos etc	---	---	50.00
Hardware Robot móvil	01	0.200	80.00
Hardware Tablero de programación	01	0.300	110.00
Tablero de desplazamiento	01	0.200	45.00
Total		1.029	601.00

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 lo menos 150 unidades para que el costo de diseño unitario no exceda los 54 soles.

Tabla 23

Costos de Ingeniería

Componente	**Horas**	**Precio/Hora S/.**	**Precio Total S/.**
Diseño de Hardware	50	30.00	1,500.00
Diseño Mecánico	50	30.00	1,500.00
Diseño Electrónico	48	30.00	1,440.00
Diseño de Software	60	30.00	1,800.00
Total			**6,240.00**

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

De acuerdo con lo desarrollado en la tesis, se pueden extraer las siguientes conclusiones:

- En cuanto a la viabilidad y costo-efectividad, se demostró que es factible diseñar una plataforma de programación tangible como herramienta educativa para la enseñanza de programación en el nivel inicial de la educación básica regular. El análisis económico reveló que el prototipo final tiene un costo de S/. 601.00 (US$ 162.43), el cual podría reducirse significativamente mediante la producción en serie. Este precio resulta competitivo, siendo 24% menor que el robot WeDo utilizado en la segunda etapa de la iniciativa 'Una laptop por niño'.
- El lenguaje de programación diseñado cumple con el primer objetivo específico, permitiendo crear programas simples y apropiados para niños de educación inicial. Se alinea con las competencias 22, 23 y 28 del Currículo Nacional de Educación Básica, introduciendo además conceptos del paradigma de programación funcional de manera accesible y sencilla.
- Se logró diseñar un entorno de programación que permite al usuario crear y definir trayectorias mediante la colocación y ordenamiento de piezas tangibles, procesando estas acciones a programas intermedios ejecutables por un robot móvil. Esta aproximación tangible facilita la comprensión y aplicación de conceptos de programación de manera intuitiva y práctica.
- Se logro diseñar el sistema electrónico que realiza el procesamiento de la programación tangible; así mismo el diseño electrónico del robot móvil cumple con

los requerimientos establecidos en el Anexo A y satisface el objetivo de ser un intérprete capaz de seguir trayectorias.

- El prototipo implementado se logró probar en la institución educativa N° 50499 Justo Barrionuevo Álvarez de Oropeza y posteriormente se aplicaron encuestas de usabilidad a los niños y profesores; obteniéndose resultados satisfactorios.
- La elección de los componentes se fundamentó en cálculos y fórmulas teóricas para un robot de locomoción diferencial, garantizando así su funcionalidad y eficiencia en la ejecución de los programas desarrollados por los usuarios.
- En términos de funcionalidad, se logró una tasa de éxito del 80% en la detección de piezas, lo que indica una interacción efectiva entre las piezas tangibles y el sistema, aunque se identificaron áreas de mejora que llevaron a ajustes en el diseño del circuito.
- En cuanto a la usabilidad, el 85% de los estudiantes consideró que la plataforma era fácil de usar y el 90% expresó satisfacción con la experiencia, lo que sugiere una aceptación positiva del sistema por parte de los usuarios, además, se observó un aumento del 30% en las interacciones a medida que avanzaba el estudio, lo que refleja un creciente compromiso y una interacción social activa entre los estudiantes. Estos resultados sugieren que la implementación de la herramienta ha sido efectiva y bien recibida, aunque también resalta la importancia de continuar refinando el diseño para maximizar su eficacia y usabilidad.

Recomendaciones

Aunque el enfoque principal de la tesis no es un análisis pedagógico exhaustivo, se recomienda encarecidamente realizar ejercicios experimentales con niños de educación inicial de esta plataforma electrónica. Estos ejercicios proporcionarían valiosos insights sobre la interacción real de los niños con la plataforma y su eficacia en el aprendizaje de conceptos de programación.

El diseño actual de las piezas tangibles, que se asemeja a un rompecabezas, cumple efectivamente con el objetivo didáctico y fomenta un enfoque lúdico y dinámico en la programación. Esta característica es un punto principal del diseño y debe mantenerse en futuras iteraciones.

Se recomienda enfatizar la importancia de una colocación precisa y centrada de las piezas tangibles en los slots del tablero programable, esto podría lograrse mediante:

- ✓ Mejoras en el diseño físico de las piezas y slots para facilitar una alineación correcta.
- ✓ Inclusión de guías visuales claras en el tablero.
- ✓ Desarrollo de un sistema de retroalimentación que indique cuando una pieza está correctamente insertada.

Desarrollar instrucciones claras y adaptadas a la edad de los usuarios sobre cómo colocar correctamente las piezas. Estas podrían incluir elementos visuales y ejemplos prácticos.

Considerar la implementación de un sistema de detección y corrección de errores leves en la colocación de las piezas, para minimizar el impacto en el movimiento del robot móvil sin comprometer la precisión del aprendizaje.

Incorporar sesiones de práctica guiadas al inicio del uso de la plataforma, donde los niños puedan familiarizarse con la correcta colocación de las piezas bajo supervisión.

Establecer un proceso de evaluación continua y retroalimentación durante el uso de la plataforma en entornos educativos reales, permitiendo ajustes y mejoras basados en la experiencia práctica.

Estas recomendaciones buscan potenciar la eficacia educativa de la plataforma, asegurando que cumpla con su objetivo de introducir conceptos de programación de manera accesible y efectiva a niños de educación inicial, mientras se mantiene la precisión necesaria para el funcionamiento óptimo del sistema.

BIBLIOGRAFIA

[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://dialnet.unirioja.es/servlet/articulo?codigo=6868305

[2] Ministerio de Educación, "Currículo Nacional de Educación Básica', Perú, 2016. Disponible en: https://hdl.handle.net/20.500.12799/4551

[3] Mota. (julio 4, 2020). Jean Piaget y las Etapas del Desarrollo Cognitivo. Disponible en: https://valentinamota.com/jean-piaget-y-las-etapas-del-desarrollo-cognitivo/

[4] Moreno León. (marzo 23, 2014). Pensamiento Computacional. Disponible en: http://surl.li/ckdmzu

[5] Giulia, LEGO WeDo and OLPC Peru: national collaboration, One Laptop Per Child, (february 12, 2011). Disponible en: http://surl.li/lvgwys

[6] División de Desarrollo Humano, Desarrollo infantil, (febrero 22, 2021). Disponible en: http://surl.li/ryfvmb

[7] Sangacha y Ortiz, "Estrategia de enseñanza para el desarrollo de habilidades a través de la programación empleando herramientas interactivas', Espirales: Revista Multidisciplinaria de Investigación, (diciembre 11, 2017). Disponible en: http://surl.li/jsmhev

[8] Scaradozzi, Sorbi, Pedale, Valzano y Vergine, "Teaching robotics at the primary school: an innovative approach", Procedia - Social and Behavioral Sciences, (febrero, 2015). Disponible en: https://doi.org/10.1016/j.sbspro.2015.01.1122

[9] Laura y Bolívar, Una Laptop Por Niño en escuelas rurales del Perú: un análisis de las barreras y facilitadores, (diciembre 2009). Disponible en: https://hdl.handle.net/20.500.12799/800

[10] Paz O., María; Martínez O., María F. "¿Qué ha pasado para que el programa ULPN en el Perú sea (hasta ahora) un fracaso?,(2012). Disponible en: https://wiki.laptop.org/go/OLPC_Peru

[11] Suzuki y Kato, "AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning," (junio 14, 2016). Disponible en: http://surl.li/stbuwr

[12] Caceres, Venero and Cuellar, "Tangible programming mechatronic interface for basic induction in programming," 2018 IEEE Global Engineering Education Conference (EDUCON), (abril, 2018), pp. 183-190. Disponible en: https://doi.org/10.1109/educon.2018.8363226

[13] Smith, springhorn, Bruce, Weber and Norris, "TactusLogic: Programming using physical objects," IST-Africa 2011 Conference Proceedings, (january, 2011). Disponible en: https://www.researchgate.net/publication/241625640

[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,

(noviembre, 2012).
Disponible en: https://doi.org/10.1109/te.2012.2190071

[15] TT. Sapounidis and S. Demetriadis, "Touch Your Program with Hands: Qualities in Tangible Programming Tools for Novice," 2011 15th Panhellenic Conference on Informatics, (november, 2011), pp. 363-367.
Disponible en: https://doi.org/10.1109/pci.2011.5

[16] S. Kakehashi, 'T. Motoyoshi, K. Koyanagi, 'T. Ohshima and H. Kawakami, "PCUBE: Block Type Programming Tool for Visual Impairments," 2013 Conference on 87 Technologies and Applications of Artificial Intelligence, Taipei, (december 2013), pp. 294-299.
Disponible en: https://doi.org/10.1109/taai.2013.65

[17] N. N. Dummel, 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, (abril, 2019).
Disponible en:https://doi.org/10.1109/educon.2019.8725073

[18] M. Y. Ímamoglu and D. Cetinkaya, "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.
Disponible en: https://doi.org/10.1109/ickea.2017.8169904

[19] T. McNerney, “Tangible Programming Bricks: an approach to making programming accessible to everyone”, 2000. ‘Tesis de Maestría, MIT, Cambridge, Massachusetts.
Disponible en: http://hdl.handle.net/1721.1/62094

[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, (october, 2017), vol. 60, no. 10, pp. 91-100.
Disponible en: https://doi.org/10.1145/3126905

[21] Trejos Buriticá, ‘Aprovechamiento de los tipos de pensamiento matemático en el aprendizaje de la programación funcional’, Tecnura, (2018), vol. 22, no. 56, pp. 29-39.
Disponible en: https://doi.org/10.14483/22487638.12807

[22] V. Cietto, C. Gena, I. Lombardi, C. Mattutino and C. Vaudano, "Co-designing with kids an educational robot," IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), Genova, (september, 2018).
Disponible en: https://doi.org/10.1109/arso.2018.8625810

[23] G. Trovato, F. Cuellar y M. Nishimura, “Introducing ‘theomorphic robots”, 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), (november, 2016), pp. 1245-1250.
Disponible en: https://doi.org/10.1109/humanoids.2016.7803429

[24] R. González, R. Rodríguez & Guzmán, “Robots Móviles con Orugas Historia, Modelado, Localización y Control”, Revista Iberoamericana de Automática e Informática Industrial, vol. 12, (2015) no. 1, pp. 3-12.
Disponible en: https://doi.org/10.1016/j.riai.2014.11.001

[25] G. Perez-Paina, E. J. Guizzo, I. Torres, D. Gonzalez-Dondo, C. Paz and EF. Trasobares, "Open hardware wheeled mobile robot for educational purposes," 2018 Ninth Argentine Symposium

and Conference on Embedded Systems (CASE), Cordoba, 2018, pp. 13-18.
Disponible en: https://doi.org/10.23919/sase-case.2018.8542162

[26] F. Mondada et al., "Bringing Robotics to Formal Education: The Thymio OpenSource Hardware Robot," IEEE Robotics & Automation Magazine, (march 2017).
Disponible en: https://doi.org/10.1109/mra.2016.2636372

[27] 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.
Disponible en: https://doi.org/10.1109/lra.2019.2896465

[28] 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.
Disponible en: https://doi.org/10.1109/iconda.2017.8270392

[29] 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 (ICASYMP), Bangkok, Thailand, (january, 2019).
Disponible en: https://doi.org/10.1109/ica-symp.2019.8646041

[30] 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.
Disponible en: https://doi.org/10.1109/intercon.2018.8526366

[31] S. Mora, "Diseño y desarrollo de controles de robot mediante métodos numéricos basados en teoría de algebra lineal", trabajo fin de grado, Universidad Politécnica de Valencia, 2017.
Disponible en: http://hdl.handle.net/10251/106808

[32] D. Belategui, “Análisis mecatrónico de accionamiento de máquinas herramienta teniendo en cuenta la flexibilidad de la estructura”, Tesis Doctoral, Universidad Euskal Herriko del país de vasco, (may, 2017), pp. 41-45.
Disponible en: http://hdl.handle.net/10810/25648

[33] L. Guzman, M. Molina & E. Rodriguez, “Seguimiento de trayectorias con un robot móvil de configuración diferencial”, Art. De investigación, Universidad Militar Nueva Granada, (junio, 2014), pp. 26-34.
Disponible en: https://doi.org/10.21500/20275846.298

[34] Ley N° 24029, Ley del Profesorado.
Disponible en: http://surl.li/vhghiv

[35] 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° 517, 2008.
Disponible en: www.digesa.minsa.gob.pe/norma_consulta/RM517-2008-MINSA.pdf

[36] Mochila escolar no debe exceder del 15% del peso del estudiante, Instituto Nacional de Salud, 2018.
Disponible en: http://surl.li/hajwoj

[37] LEGO® Education WeDo 2.0 Core Set, Lego Education.
Disponible en: http://surl.li/ckgpil

[38] J. Labra Gayo, Cueva Lovelle, R. Castanedo, A. Juan, M. Luengo, EF. Ortin, Intérpretes y Diseño de Lenguajes de Programación, (noviembre, 2003).
Disponible en: http://surl.li/wlxatt

[39] H. Jia, "Foundations of the Theory of Signs" by Charles William Morris, 1938, Article in Chinese Semiotic Studies, (february, 2019).
Disponible en: http://dx.doi.org/10.1515/css-2019-0001

[40] Smith, AC. 2006. Tangible cubes as programming objects. 16th International conference on artificial reality and telexistence, 29 Nov- 1 Dec 2006, pp157-161.
Disponible en: https://doi.org/10.1109/icat.2006.121

[41] Ramos, Celeste; Rodríguez, Jorge; Cecchi, Laura. (2020). Plataforma para la programación tangible. Universidad Nacional de La Plata.
Disponible en: http://sedici.unlp.edu.ar/handle/10915/103745

[42] Gallardo, D., Julia, CF, y Jorda, S. (2008). TurTan: Un lenguaje de programación tangible para la exploración creativa. 2008 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems.
Disponible en: https://doi.org/10.1109/TABLETOP.2008.4660189

Anexo A – Lista de exigencias

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

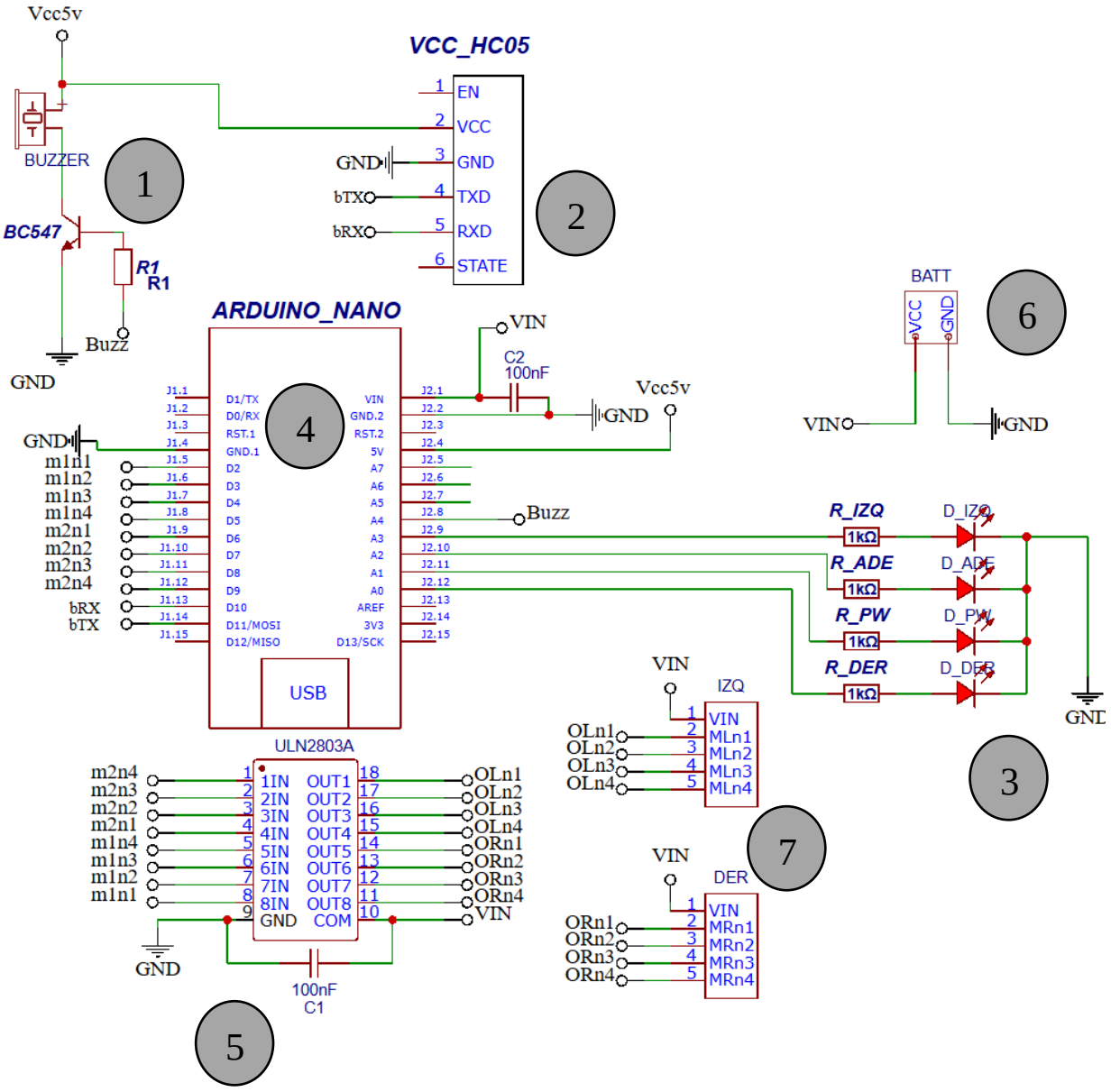
Tabla 24

Lista de requerimientos

	Características	Requerimientos	Deseo o Exigencia
✓	Geometría	Volumen de Robot Móvil **≤ 25 x 25 x 25 cm.** Volumen de las piezas **≤ 5 cm X 4 cm X 1 cm.**	**D**
✓	Cinemática	La velocidad del intérprete será una **longitud del mismo por cada 3 segundos.**	**E**
✓	Control	Conseguir que siga la trayectoria y alcance la posición final con un error ≤ 0.05 unidades.	**E**
✓	Electrónica (Hardware)	La autonomía ≥ 3 horas. Locomoción del interprete: diferencial.	**D**
✓	Software	Lenguaje de programación: paradigma basado en reglas.	**E**
✓	Comunicaciones	La comunicación entre entorno e interprete: inalámbrica.	**D**
✓	Seguridad	Material del interprete robot móvil: PLA. Material de las piezas tangibles: madera.	**E**
✓	Ergonomía	Peso máximo de la plataforma de programación tangible ≤ 2 Kg.	**E**
✓	Costos	Costo de 1 set ≤ 214 dólares americanos.	**E**

Anexo B – Diagrama esquemático del interprete robot móvil

El diagrama esquemático del intérprete robot móvil.

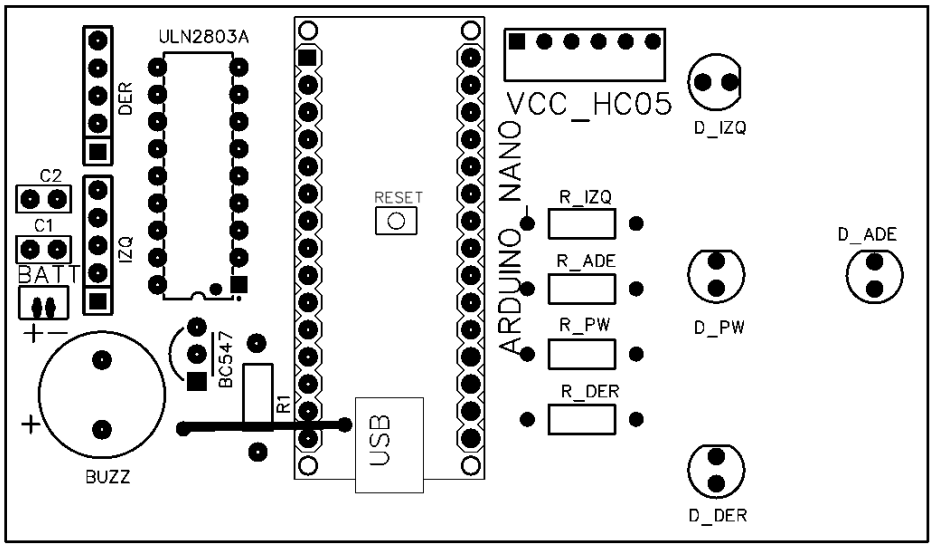


1. Altavoz.
2. Comunicación serial inalámbrica – Bluetooth.
3. Cursores de LED, indicador de dirección y encendido.
4. Microprocesador.
5. ULN2803A Driver motores.
6. Regulación y alimentación.
7. Conector de motores.

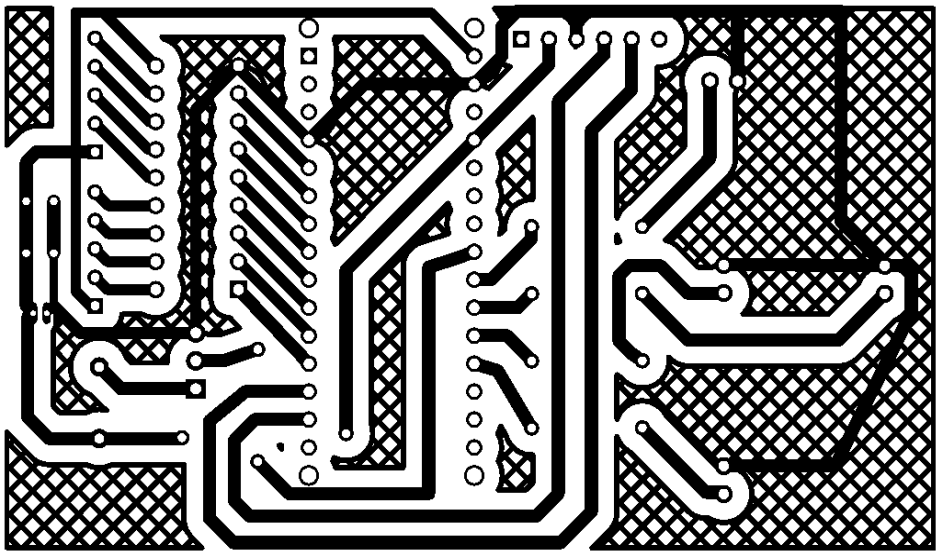
Anexo C – Diseño del circuito impreso PCB

El diseño del circuito impreso (PCB) que ira acoplada al microcontrolador se realizó a través del software EasyEDA, a continuación, se muestran sus 2 capas.

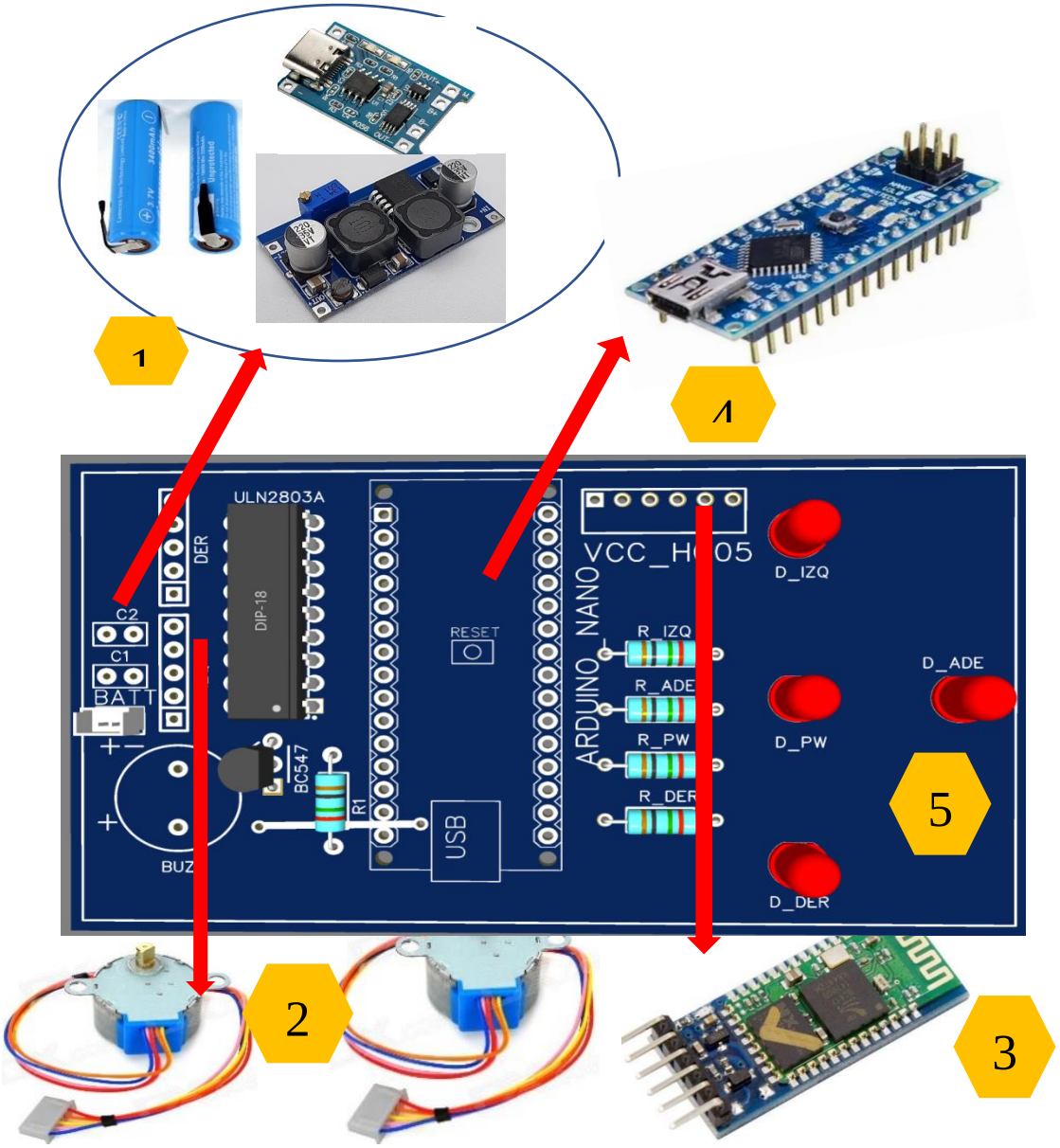
Capa superior



Capa inferior



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

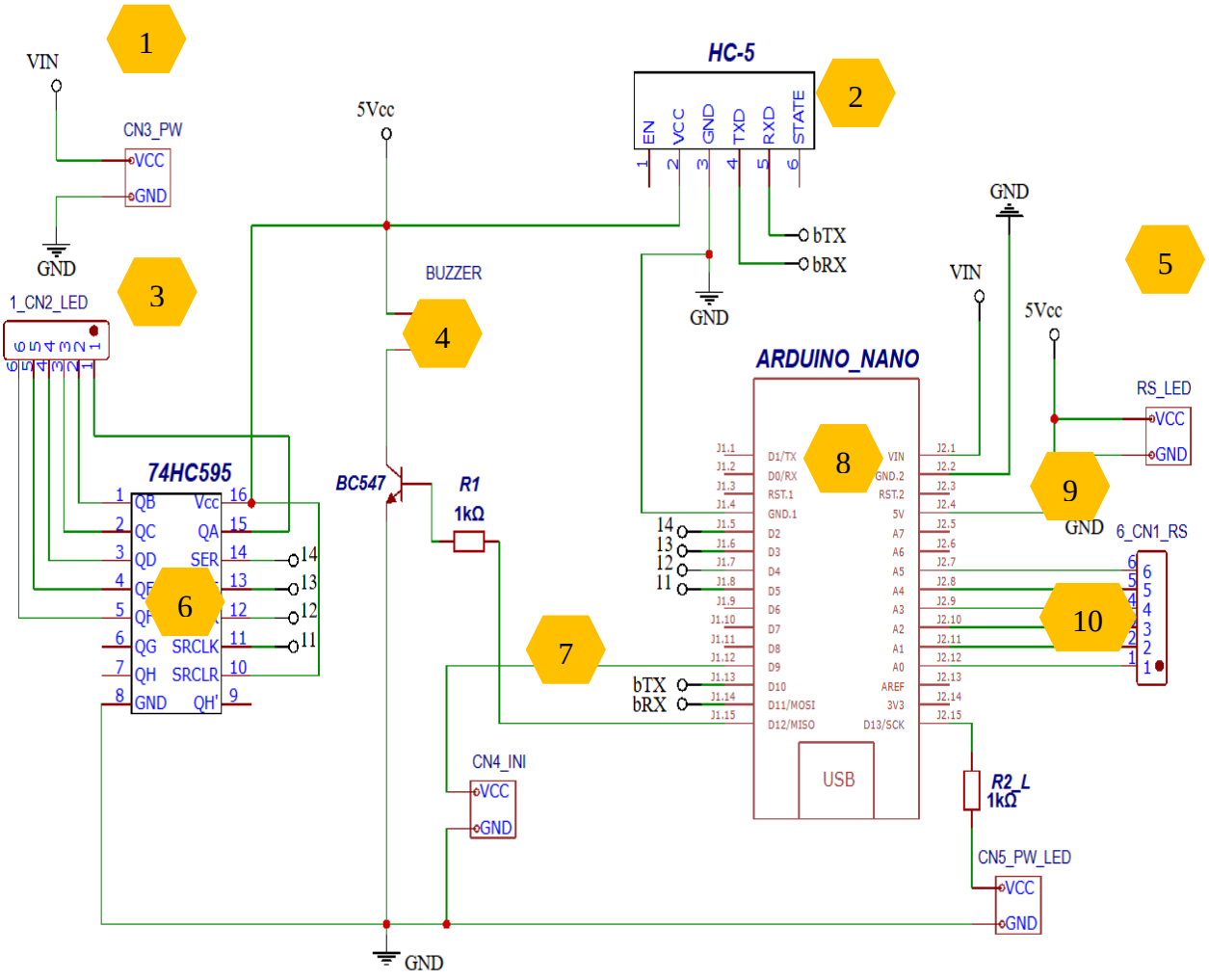


1. Fuente de alimentación (Batería, módulo de carga y módulo de regulación).
2. Motores, Izquierdo – Derecho.
3. Módulo de Comunicación Inalámbrica (BLUETOOTH).
4. Microcontrolador.
5. Cursor Iluminado LED.

Anexo D – Diagrama esquemático del entorno de programación

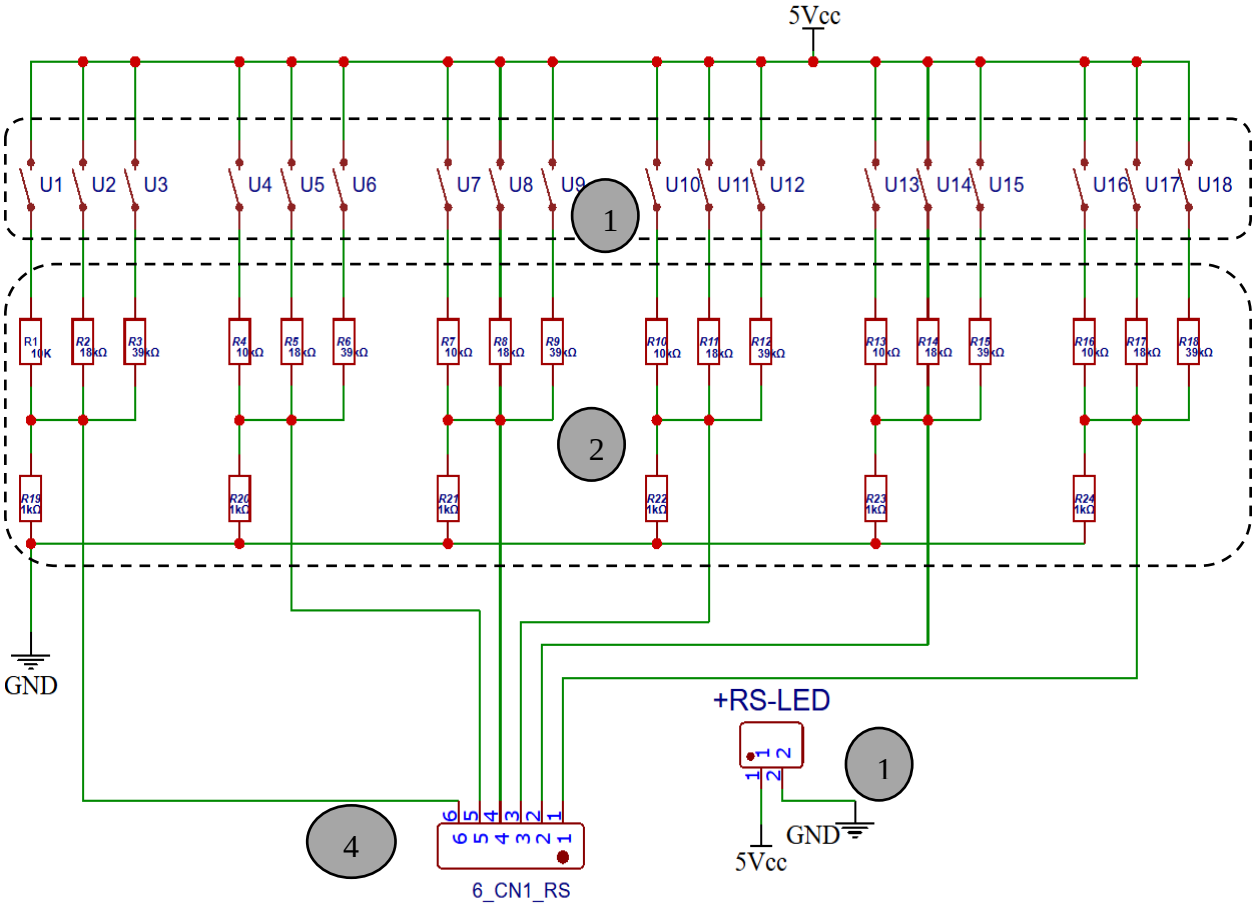
El diagrama esquemático del entorno de programación.

Etapa de procesamiento de las instrucciones adquiridas mediante los sensores Reed Switch.



1. Regulación y alimentación.
2. Comunicación serial inalámbrica – Bluetooth.
3. Conexión a panel LED indicador de desplazamiento.
4. Altavoz.
5. Alimentación panel LED.
6. Registro de desplazamiento.
7. Pulsador de inicialización.
8. Microcontrolador.
9. Conexión a panel de Reed Switch.
10. LED de encendido.

Etapa de adquisicion de datos mediante los sensores Reed Switch



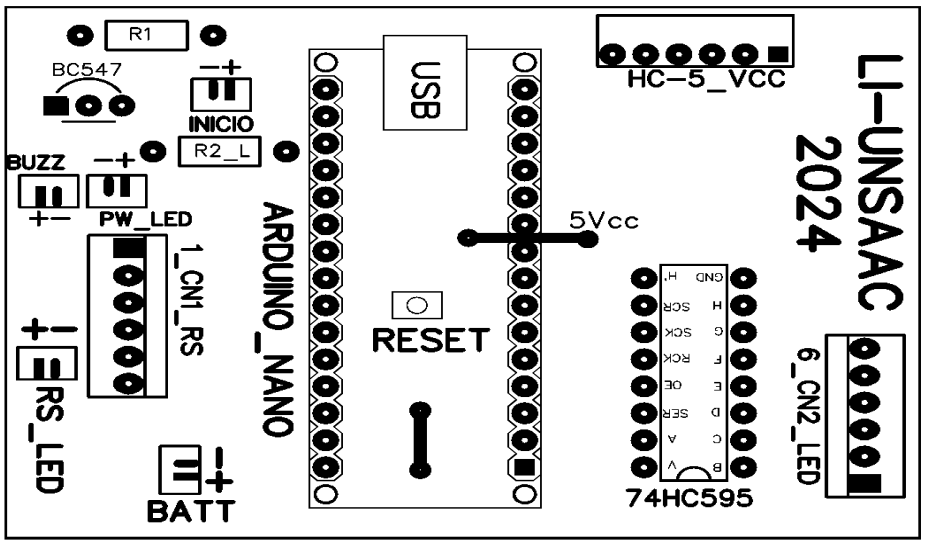
1. Alimentación.
2. Matriz de divisor de tensión para ADC.
3. Matriz Reed Switch.
4. Molex de conexión a la placa principal (lectura de datos o valores).

Anexo E – Diseño del circuito impreso PCB

Diseño del circuito impreso (PCB) del entorno de programación el cual ira acoplada al microcontrolador se realizó a través del software EasyEDA, a continuación, se muestran sus 2 capas, de las dos etapas.

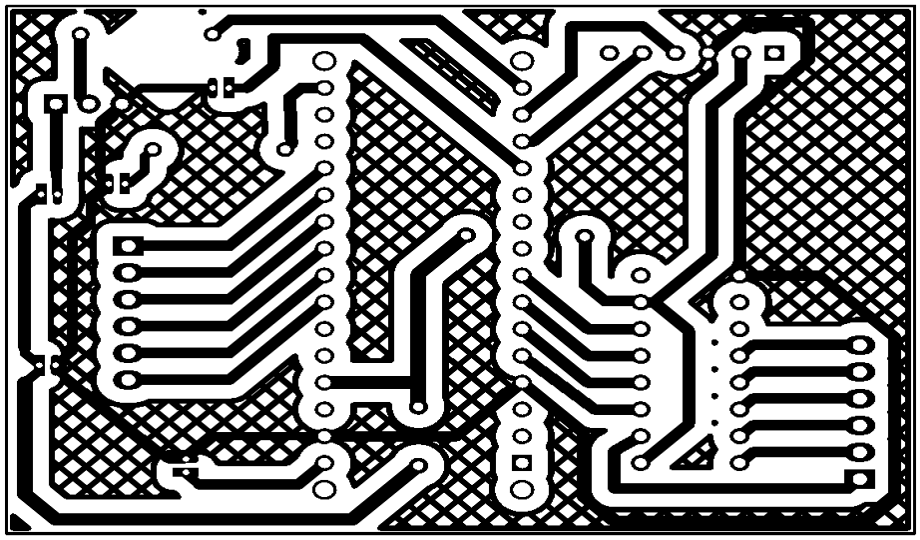
Capa superior

Procesamiento de datos.



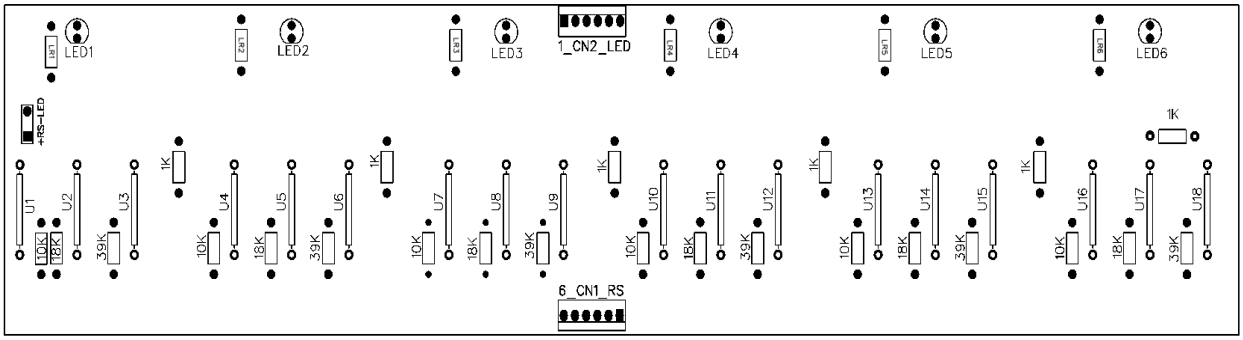
Capa inferior

Procesamiento de datos.



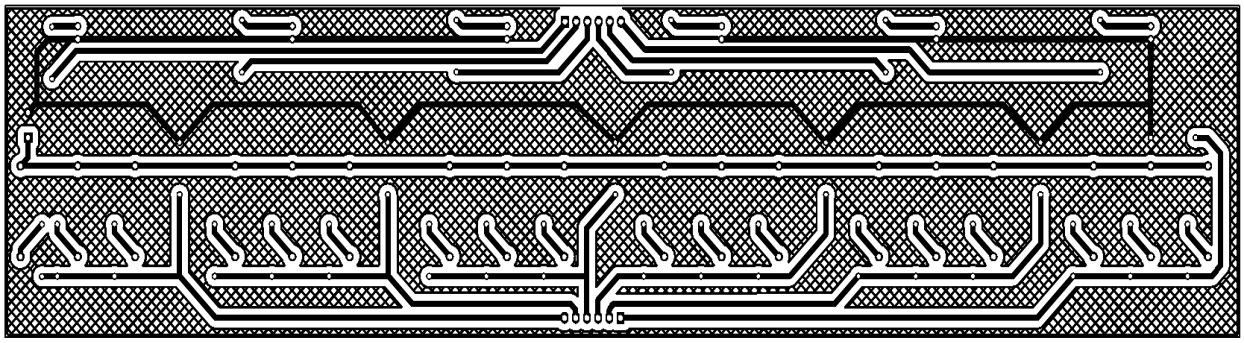
Capa superior

Panel de adquisición de datos mediante Reed Switch e indicador de pizas de programación.

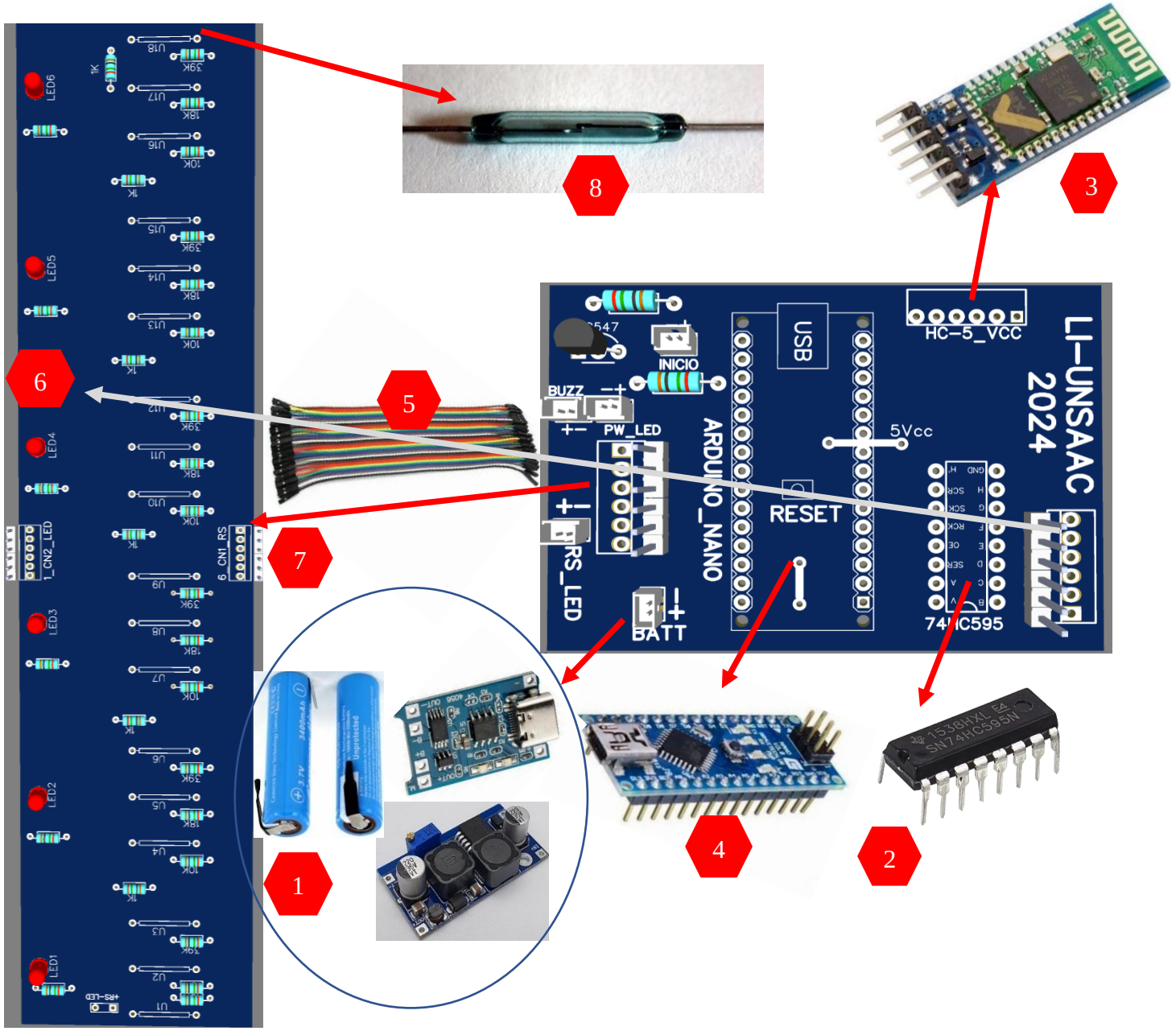


Capa inferior

Panel de adquisición de datos mediante Reed Switch e indicador de pizas de programación.



Asimismo, se muestra el diagrama de conexiones del PCB del chasis del tablero programable junto a sus periféricos.


8
3
6
5
7
1
4
2
HC-5_VCC
LI-UNSAAC
2024
USB
INICIO
BUZZ
PW_LED
ARDUINO_NANO
5Vcc
RESET
RS_LED
BATT
74HC595
LED6
LED5
LED4
LED3
LED2
LED1
1_CN2_LED
6_CN1_RS

1. Fuente de alimentación (Batería, módulo de carga y módulo de regulación).
2. Circuito integrado para el REGISTRO DE DESPLAZAMIENTO.
3. Módulo de Comunicación Inalámbrica (BLUETOOTH).
4. Microcontrolador.
5. Cables Jumper para las conexiones de los periféricos.
6. Tarjeta indicadora de lector de piezas tangibles.
7. Tarjeta de adquisición de datos mediante Reed Switch - ADC
8. Sensor magnético REED SWITCH

Anexo F – Código de programación

- **Código de programación del tablero de programable**

```
#define led 13

#define data 2
#define clock 4
#define latch 3

#define i0 A0
#define i1 A1
#define i2 A2

#define button A3
#define buzzer A4
#define cards 4
#define reeds 3

int values[cards][reeds] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0,
0}};

bool running = false;
bool a = false;
int i = 0;
int card[] = {
    0b00000010, 0b00000100, 0b00001000, 0b00010000,

    // 0b11111101,
    // 0b11111011,
    // 0b11110111,
    // 0b11101111,
};
int clear = 0b00000000;
// int clear = 0b11111111;

int val = 0;

bool getval(int i) {
  // values[i][0] = digitalRead(i1);
  // values[i][1] = digitalRead(i0);
  // values[i][2] = digitalRead(i2);
  // return values[i][0] > 500 || values[i][1] > 500 ||
values[i][2] > 500;

  Serial.print(!digitalRead(i1));
  Serial.print(!digitalRead(i0));
  Serial.print(!digitalRead(i2));
```

```
  Serial.println();
}

void print_reeds() {
  for (int i = 0; i < cards; i++) {
    print_reed(i);
  }
}

void print_reed(int i) {
  Serial.print("card ");
  Serial.print(i);
  Serial.print(": ");
  Serial.print(values[i][0]);
  Serial.print(" ");
  Serial.print(values[i][1]);
  Serial.print(" ");
  Serial.print(values[i][2]);
  Serial.println();
}

void beep() {
  digitalWrite(buzzer, HIGH);
  digitalWrite(led, HIGH);
  delay(50);
  digitalWrite(buzzer, LOW);
  digitalWrite(led, LOW);
}

void setup() {
  Serial.begin(9600);
  Serial.println("... start");

  pinMode(led, OUTPUT);
  pinMode(latch, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(data, OUTPUT);

  pinMode(enable, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  pinMode(i0, INPUT_PULLUP);
  pinMode(i1, INPUT_PULLUP);
  pinMode(i2, INPUT_PULLUP);

  digitalWrite(enable, LOW);
  digitalWrite(buzzer, LOW);
  for (int i = 0; i < 5; i++) {
```

```
    beep();
    delay(100);
  }
}

void clear_register() {
  digitalWrite(latch, LOW);
  shiftOut(data, clock, MSBFIRST, clear);
  digitalWrite(latch, HIGH);
}

void loop() {
  if (!digitalRead(button)) {
    running = !running;
    beep();
    print_reeds();
    delay(200);
  }

  for (int i = 0; i < 4; i++) {
    digitalWrite(latch, LOW);
    shiftOut(data, clock, MSBFIRST, card[i]);
    digitalWrite(latch, HIGH);

    getval(i);
    // print_reed(i);
    delay(5000);
  }
  // clear_register();
  // print_reeds();
  // delay(500);
}
```

- **Código de programación del actuador robot móvil**

```
#include <SoftwareSerial.h>
#include <TimerOne.h>

#define led 13

#define buzzer A4
#define led_lft A3
#define led_fwd A2
#define led_pwr A1
#define led_rgt A0

#define txPin 10
#define rxPin 11

#define m1n1 5
#define m1n2 4
#define m1n3 3
#define m1n4 2

#define m2n1 6
#define m2n2 7
#define m2n3 8
#define m2n4 9

// Vuelta entera 2048 Pasos
// Radio de llanta 6.9
// Longitud cuadradito 12cm
static int move = 1135;
static int turn = 512 + 8 * 14;

int dir_1 = 0;
int dir_2 = 0;
int seq_1 = 0;
int seq_2 = 0;
int move_c = 0;
int turn_c = 0;
int on = 0;

enum state_t { st_idle, st_bck, st_fwd, st_right, st_left };

state_t state = st_idle;

SoftwareSerial btSerial(rxPin, txPin);

void beep() {
  digitalWrite(buzzer, HIGH);
```

```
  digitalWrite(led, HIGH);
  delay(30);
  digitalWrite(buzzer, LOW);
  digitalWrite(led, LOW);
}

void beep_beep(int t) {
  for (int i = 0; i < t; i++) {
    beep();
    delay(50);
  }
}

void off() {
  digitalWrite(m1n1, LOW);
  digitalWrite(m1n2, LOW);
  digitalWrite(m1n3, LOW);
  digitalWrite(m1n4, LOW);

  digitalWrite(m2n1, LOW);
  digitalWrite(m2n2, LOW);
  digitalWrite(m2n3, LOW);
  digitalWrite(m2n4, LOW);

  digitalWrite(led_pwr, HIGH);
}

void off_led() {
  digitalWrite(led_fwd, LOW);
  digitalWrite(led_lft, LOW);
  digitalWrite(led_rgt, LOW);
}

void process_command() {
  char c;
  while (btSerial.available()) {
    c = btSerial.read();
    Serial.println(c);
    beep_beep(2);

    switch (c) {
    case 'f':
      if (state != st_fwd)
        off();
      state = st_fwd;
      dir_1 = 0;
      dir_2 = 0;
      move_c = 0;
      break;
```

```
    case 'b':

      if (state != st_bck)
        off();
      state = st_bck;
      dir_1 = 1;
      dir_2 = 1;
      move_c = 0;
      break;
    case 'l':

      if (state != st_left)
        off();
      state = st_left;
      dir_1 = 1;
      dir_2 = 0;
      turn_c = 0;

      break;
    case 'r':

      if (state != st_right)
        off();
      dir_1 = 0;
      dir_2 = 1;
      turn_c = 0;
      state = st_right;

      break;
    case 's':
      state = st_idle;
      off();
      break;

    default:
      state = st_idle;
      off();
      break;
    }
  }
}

int sequence[] = {0b00001100, 0b00000110, 0b00000011, 0b00001001,};

void stepper() {
  seq_1 = dir_1 ? seq_1 + 1 : seq_1 - 1;
  seq_2 = dir_2 ? seq_2 + 1 : seq_2 - 1;
  if (seq_2 > 3)
    seq_2 = 0;
```

```
  if (seq_1 > 3)
    seq_1 = 0;
  if (seq_2 < 0)
    seq_2 = 3;
  if (seq_1 < 0)
    seq_1 = 3;
  digitalWrite(m1n1, (sequence[seq_1] & 1));
  digitalWrite(m1n2, (sequence[seq_1] >> 1) & 1);
  digitalWrite(m1n3, (sequence[seq_1] >> 2) & 1);
  digitalWrite(m1n4, (sequence[seq_1] >> 3) & 1);

  digitalWrite(m2n1, (sequence[seq_2] & 1));
  digitalWrite(m2n2, (sequence[seq_2] >> 1) & 1);
  digitalWrite(m2n3, (sequence[seq_2] >> 2) & 1);
  digitalWrite(m2n4, (sequence[seq_2] >> 3) & 1);
}

void blink() {
  // digitalWrite(led, !digitalRead(led));

  switch (state) {
  case st_idle:
    off();
    off_led();
    break;

  case st_fwd:
    digitalWrite(led_fwd, HIGH);
    digitalWrite(led_pwr, LOW);
    move_c++;
    if (move_c >= move) {
      state = st_idle;
    }
    stepper();
    break;

  case st_bck:
    move_c++;
    if (move_c >= move) {
      state = st_idle;
    }
    stepper();
    break;

  case st_left:
    digitalWrite(led_lft, HIGH);
    digitalWrite(led_pwr, LOW);

    turn_c++;
```

```
    if (turn_c >= turn) {
      state = st_idle;
    }
    stepper();
    break;
  case st_right:
    digitalWrite(led_rgt, HIGH);
    digitalWrite(led_pwr, LOW);

    turn_c++;
    if (turn_c >= turn) {
      state = st_idle;
    }
    stepper();
    break;
  default:
    break;
  }
}

void setup() {

  pinMode(led, OUTPUT);

  pinMode(m1n1, OUTPUT);
  pinMode(m1n2, OUTPUT);
  pinMode(m1n3, OUTPUT);
  pinMode(m1n4, OUTPUT);

  pinMode(m2n1, OUTPUT);
  pinMode(m2n2, OUTPUT);
  pinMode(m2n3, OUTPUT);
  pinMode(m2n4, OUTPUT);

  pinMode(led_lft, OUTPUT);
  pinMode(led_fwd, OUTPUT);
  pinMode(led_pwr, OUTPUT);
  pinMode(led_rgt, OUTPUT);

  off();
  Timer1.initialize(1800); // uS
  Timer1.attachInterrupt(blink);
  btSerial.begin(9600);
  Serial.begin(115200);
  beep_beep(3);
  Serial.println("...Start car");
}

void loop() { process_command(); }
```

Anexo G – Manual de usuario de la plataforma electrónica de programación tangible

Introducción

Este manual está diseñado para guiar a los usuarios, especialmente a educadores y padres, en el uso de una plataforma de programación tangible destinada a niños de 4 a 6 años. La plataforma permite a los niños aprender conceptos básicos de programación mediante la manipulación de piezas físicas, fomentando su creatividad, y habilidades de resolución de problemas.

Requisitos Previos

Antes de comenzar a utilizar la plataforma, asegúrese de cumplir con los siguientes requisitos:

- Encendido: El tablero de programación, así como el intérprete robot móvil deben encenderse al mismo tiempo para que pueda iniciar la conectividad Bluetooth inalámbrica entre ambos equipos.
- Tiempo de Espera: La plataforma requiere un tiempo de espera de 60 segundos antes de que se pueda iniciar su uso. Este tiempo permite que todos los sistemas se inicialicen correctamente (Verifica que el LED azul dentro del actuador del robot móvil deje de parpadear rápidamente).

Diseño del Tablero

El tablero del recorrido puede ser diseñado con un máximo de 6x6 unidades cuadráticas (de 12x12 cm), esto permite una variedad de configuraciones y desafíos para los niños.

Uso Correcto de las Piezas

Para garantizar el correcto funcionamiento del sistema:

Encaje Correcto: Asegúrese de que las piezas utilizadas encajen bien en los slots del tablero programable. Un mal encaje puede resultar en un funcionamiento incorrecto o en la incapacidad del robot para seguir la trayectoria programada.

Pasos para Iniciar

1. Esperar 50 Segundos:

   Una vez encendido ambos equipos, espere 50 segundos antes de proceder a utilizar.

2. Diseñar el Tablero:

   Utilice las piezas tangibles para crear un recorrido en el tablero, asegurándose de que no exceda las dimensiones de 6x6 unidades.

3. Programar el Recorrido:

   Coloque las piezas en los slots correspondientes, asegurándose de que cada pieza esté bien encajada.

4. Iniciar la Ejecución:

   Una vez que el recorrido esté diseñado y todas las piezas estén correctamente colocadas, inicie la ejecución del programa desde la interfaz del dispositivo conectado.

Consejos Adicionales

Supervisión: Se recomienda la supervisión de un adulto durante las actividades para asegurar un ambiente seguro y educativo.

Exploración Creativa: Anime a los niños a experimentar con diferentes configuraciones y recorridos para fomentar su curiosidad y creatividad.

Resolución de Problemas: Si el robot no sigue la trayectoria como se esperaba, verifique el encaje de las piezas y asegúrese de que todas estén correctamente alineadas.

Anexo H – Modelo de Test de Usabilidad aplicado

A continuación, se muestra el test de usabilidad para el prototipo de programación tangible, aplicado en la II.EE.

A. PROTOCOLO DE PRUEBA

1. **Introducción (5 minutos):**

- Darle la bienvenida al niño y explicar de manera sencilla el propósito de la prueba.
- Obtener el consentimiento de los padres/tutores y asegurar la comodidad del niño.

2. **Familiarización (05 minutos):**

- Presentar los bloques físicos y sus funciones básicas al niño.
- Permitir que el niño explore libremente los bloques y se familiarice con ellos.

3. **Tarea 1: Mover el robot en línea recta (05 minutos por grupo):**

- Explicarle al niño que debe crear un programa para mover el robot en línea recta.
- Observa cómo el niño selecciona y conecta los bloques para lograr el objetivo.
- Brindar ayuda mínima si el niño lo solicita o se atasca.

4. **Tarea 2: Girar el robot 90 grados (05 minutos por grupo):**

- Pedir al niño que modifique el programa para hacer que el robot gire 90 grados.
- Observa cómo el niño se adapta al programa y si encuentra la solución.
- Proporcionar asistencia si es necesario.

5. **Tarea 3: Seguir un patrón de movimiento (05 minutos por grupo):**

- Solicitar al niño que cree un programa para que el robot siga un patrón de movimiento predefinido.
- Observa cómo el niño planifica y construye el programa.
- Brindar apoyo si el niño lo requiere.

6. **Cuestionario post-prueba (05 minutos):**

- Aplicar un breve cuestionario con preguntas sencillas sobre la experiencia del niño.
- Utilizar una escala de caras para evaluar la satisfacción del usuario.
- Recopilar comentarios y sugerencias del niño.

7. **Cierre (05 minutos):**

- Agradecer al niño por su participación y entregarle un pequeño obsequio.
- Asegurarse de que el niño se sienta cómodo y responder a cualquier pregunta.

Consideraciones adicionales

- Realizar las pruebas en un entorno cómodo y libre de distracciones.
- Contar con un moderador capacitado para guiar al niño de manera amable y paciente.
- Utilizar cámaras de grabación para registrar las interacciones y expresiones verbales.
- Analizar los datos recopilados para identificar problemas de usabilidad y áreas de mejora.
- Generar un informe detallado con los hallazgos y recomendaciones para el prototipo.

Este enfoque permite evaluar de manera efectiva la usabilidad del prototipo de programación tangible para niños de 4 a 6 años, obteniendo valiosa retroalimentación que ayudará a refinar y mejorar el diseño final.

B. Cuestionario Para Docentes y Niños en una Prueba de Usabilidad de un Prototipo de Programación Tangible

Cuestionario para docentes

1. ¿Qué tan familiarizado está sus alumnos con dispositivos tecnológicos y juguetes educativos?

a) Muy familiarizado

b) Algo familiarizado

c) Poco familiarizado

d) Nada familiarizado

2. ¿Crees que el prototipo es apropiado para el nivel de desarrollo y habilidades de los niños de 4 a 6 años?

a) Sí, es muy adecuado

b) En general es adecuado

c) No es muy adecuado

d) No es adecuado en absoluto

3. ¿Qué aspectos del prototipo le parecieron más atractivos e interesantes para sus alumnos?

a) Diseño y apariencia

b) Funcionalidad y desafíos

c) Instrucciones y guía

d) Otros: _____

4. ¿Qué aspectos del prototipo le parecieron menos claros o difíciles de usar para sus alumnos?

a) Diseño y apariencia

b) Funcionalidad y desafíos

c) Instrucciones y guía

d) Otros: _____

5. ¿Cree que el prototipo ayudó a sus alumnos a aprender conceptos básicos de programación?

a) Sí, aprendió mucho

b) Aprendió algunos conceptos

c) Aprendió poco

d) No aprendió nada

6. ¿Le gustaría que su hijo/a siguiera usando el prototipo en casa o en la escuela?

a) Sí, definitivamente

b) Probablemente sí

c) Probablemente no

d) No, definitivamente no

7. ¿Tiene alguna sugerencia o comentario adicional sobre el prototipo?

a) Respuesta abierta

---

8. ¿Qué tan motivados y entusiasmados parecían los niños durante la prueba?

a) Muy motivadores

b) Motivados

c) Neutral

d) Poco motivados

e) Nada motivado

9. ¿Qué aspectos del prototipo considera que necesitan mejoras o adaptaciones para este grupo de edad?

a) Diseño y apariencia

b) Funcionalidad y desafíos

c) Instrucciones y guía

d) Otros: _____

10. ¿Recomendarías el uso de este prototipo en actividades de programación para niños de 4 a 6 años?

a) Definitivamente sí

b) Probablemente sí

c) Neutral

d) Probablemente no

e) Definitivamente no

Cuestionario para niños

1. ¿Te gustó jugar con el prototipo?

a) Sí, me gustó mucho

b) Me gustó

c) No me gustó mucho

d) No me gustó nada

2. ¿Qué fue lo que más te gustó del prototipo?

a) Diseño y apariencia

b) Juegos y desafíos

c) Aprender cosas nuevas

d) Otros: _____

3. ¿Qué fue lo que menos te gustó del prototipo?

a) Diseño y apariencia

b) Juegos y desafíos

c) Instrucciones y guía

d) Otros: _____

4. ¿Entendiste cómo usar el prototipo?

a) Sí, entendí todo

b) Entendí la mayoría

c) Entendí poco

d) No entendí nada

5. ¿Aprendiste algo nuevo sobre programación al jugar con el prototipo?

a) Sí, aprendí mucho

b) Aprendí algunas cosas

c) Aprendí poco

d) No aprendí nada

6. ¿Te gustaría seguir jugando con el prototipo en casa o en la escuela?

a) Sí, me gustaría mucho

b) Me gustaría

c) No me gustaría mucho

d) No me gustaría nada

7. ¿Tienes algún otro comentario o sugerencia sobre el prototipo?

a) Respuesta abierta

---

8. ¿Te gustó crear programas para mover al robot?

a) Me gustó mucho

b) Me gustó

c) Neutral

d) No me gusto

e) No me gustó nada

¿Qué fue lo que más te gustó del juego de los bloques mágicos? (Respuesta abierta)

---

9. ¿Quieres seguir jugando con los bloques mágicos?

a) Sí, mucho

b) Si

c) Neutral

d) No

e) No, para nada

Estos cuestionarios permiten recopilar información valiosa sobre la usabilidad y la experiencia de los usuarios (docentes y niños) con el prototipo de programación tangible. Las preguntas están adaptadas al nivel de comprensión de los niños y buscan obtener una retroalimentación cualitativa y cuantitativa.