UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

INFORME FINAL DEL TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

Diseño de un 'Kit de Aprendizaje IoT' físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de programación.

Trabajo de titulación previo a la obtención del título de Ingeniería en Telecomunicaciones

AUTOR:

Olmedo Moreno Jaime Alexander

DIRECTOR:

Suárez Zambrano Luis Edilberto

Ibarra - Ecuador 2025

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad. Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004209266		
APELLIDOS Y NOMBRES:	OLMEDO MORENO JAIME ALEXANDER		
DIRECCIÓN:	Atuntaqui, Calle Rio Amazonas y Las Vertientes		
EMAIL:	jeolmedom@utn.edu.ec/jaolmedo2@outlook.es		
TELÉFONO FIJO:		**TELF. MOVIL**	0990555388

DATOS DE LA OBRA	
TÍTULO:	Diseño de un 'Kit de Aprendizaje IoT' físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de programación
AUTOR:	Olmedo Moreno Jaime Alexander
FECHA:	2025/06/11
CARRERA/PROGRAMA:	Pregrado
TITULO	Ingeniero en Telecomunicaciones
DIRECTOR:	Msc. Luis Edilberto Suárez Zambrano

AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Olmedo Moreno Jaime Alexander, con cédula de identidad Nro. 1004209266, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de integración curricular descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

Ibarra, a los 11 días del mes de junio de 2025.

EL AUTOR:

Olmedo Moreno Jaime Alexander.

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 11 días del mes de junio de 2025.

EL AUTOR:

Olmedo Moreno Jaime Alexander.

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 11 de junio de 2025

Msc. Luis Edilberto Suárez Zambrano

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

.....

Msc. Luis Edilberto Suárez Zambrano

C.C.: 1002304291

APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificador del trabajo de Integración Curricular Diseño de un 'Kit de Aprendizaje IoT' físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de programación, elaborado por Jaime Alexander Olmedo Moreno, previo a la obtención del título de Ingeniero en Telecomunicaciones, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

Ing. Luis Edilberto Suárez Zambrano, Msc

C.C.: 1002304291

Ing. Fabián Geovanny Cuzme Rodríguez, Msc

C.C.: 1311527012

DEDICATORIA

Dedico este trabajo a los investigadores y profesionales cuyas ideas y contribuciones han sido fuente de inspiración a lo largo de este proceso. A todos aquellos que, con su esfuerzo y dedicación, han impulsado el avance del conocimiento y el desarrollo tecnológico, marcando indirectamente el camino que he seguido.

Este trabajo es una muestra del impacto que generan las nuevas ideas y soluciones en el campo del IoT y la educación. Gracias a su visión e investigaciones, encontré la motivación y las herramientas necesarias para llevar adelante este proyecto con compromiso y entusiasmo.

Jaime Alexander Olmedo Moreno

AGRADECIMIENTO

Agradezco profundamente a las instituciones que proporcionaron los recursos necesarios para llevar a cabo esta investigación. Su colaboración fue esencial para la materialización de mi proyecto, facilitando tanto los aspectos técnicos como logísticos.

No puedo dejar de mencionar a mi familia y amigos, quienes siempre estuvieron a mi lado, brindándome su amor y apoyo incondicional. A mis padres, por su constante confianza en mí, su paciencia y por enseñarme el valor del esfuerzo y la perseverancia.

A mis amigos, por sus palabras de aliento y por estar siempre allí cuando más los necesitaba. Este proyecto le pertenece tanto a usted como a mí, pues sin su apoyo emocional y moral no habría sido posible llegar hasta aquí.

Finalmente, a todas las personas que, de alguna manera, contribuyeron a este logro, ya sea con una palabra de aliento, una crítica constructiva o un gesto amable. Su colaboración, aunque a veces silenciosa, siempre fue apreciada y dejó una huella significativa en este proceso.

RESUMEN

El presente estudio tiene como finalidad el diseño de un "Kit de Aprendizaje IoT" físico que integra sensores y actuadores, con el propósito de fomentar el desarrollo del pensamiento lógico en los fundamentos de programación en estudiantes.

El objetivo general fue crear un sistema educativo accesible que permita a los estudiantes comprender y aplicar los conceptos de Internet de las Cosas y programación.

La metodología empleada incluyó el desarrollo de un prototipo modular juntamente con la integración de hardware y software, además se menciona la implementación de pruebas piloto en entornos educativos que permitieron evaluar el uso del kit mediante la interacción directa de estudiantes.

Los resultados más relevantes indicaron que los estudiantes lograron desarrollar proyectos funcionales de manera autónoma, con un alto nivel de interés y una mejor comprensión de los conceptos relacionados con IoT y programación.

Se concluye que el kit es una herramienta educativa eficaz para enseñar fundamentos de programación e IoT, de igual manera se recomienda mejorar la documentación, añadir nuevos módulos, mantenimiento de código y proporcionar formación continua a los docentes para maximizar su impacto educativo.

Palabras clave: Kit de aprendizaje, IoT, programación, educación, pensamiento lógico, tecnología educativa.

ABSTRACT

The present study aims to design a physical "IoT Learning Kit" that integrates sensors and actuators, with the purpose of promoting the development of logical thinking in the fundamentals of programming in students.

The general objective was to create an accessible educational system that allows students to understand and apply the concepts of the Internet of Things and programming.

The methodology employed included the development of a modular prototype along with the integration of hardware and software. It also mentions the implementation of pilot tests in educational environments, which allowed the evaluation of the kit through the direct interaction of students.

The most relevant results indicated that the students were able to develop functional projects autonomously, with a high level of interest and a better understanding of the concepts related to IoT and programming.

It is concluded that the kit is an effective educational tool for teaching fundamentals of programming and IoT. Likewise, it is recommended to improve the documentation, add new modules, maintain the code, and provide continuous training to teachers to maximize its educational impact.

Keywords: Learning kit, IoT, programming, education, logical thinking, educational technology.

LISTA DE SIGLAS

IoT - Internet of Things (Internet de las Cosas)

KIT - Kit de Aprendizaje

SW - Software

HW - Hardware

API - Application Programming Interface (Interfaz de Programación de Aplicaciones)

PCB - Printed Circuit Board (Placa de Circuito Impreso)

IDE - Integrated Development Environment (Entorno de Desarrollo Integrado)

GPIO - General Purpose Input/Output (Entrada/Salida de Propósito General)

Wi-Fi - Wireless Fidelity (Red Inalámbrica)

TCP/IP - Transmission Control Protocol/Internet Protocol (Protocolo de Control de Transmisión/Protocolo de Internet)

UDP - User Datagram Protocol (Protocolo de Datagrama de Usuario)

Wi-Fi SDK - Wi-Fi Software Development Kit (Kit de Desarrollo de Software para Wi-fi)

ESP32 - A 32-bit microcontroller with Wi-Fi and Bluetooth capabilities used in IoT applications.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I. ANTECEDENTES

Introducción

En la era digital actual, el Internet de las Cosas (IoT) se ha convertido en una parte integral de nuestra vida cotidiana, transformando la forma en que interactuamos con el mundo que nos rodea. Desde dispositivos domésticos inteligentes hasta aplicaciones industriales, el IoT ha abierto un vasto panorama de posibilidades tecnológicas. Sin embargo, el dominio efectivo de esta tecnología requiere no solo comprender sus fundamentos técnicos, sino también desarrollar habilidades de pensamiento lógico y programación.

En este contexto, el diseño de un 'Kit de Aprendizaje IoT' físico surge como una herramienta educativa innovadora, destinada a fomentar el desarrollo del pensamiento lógico y las habilidades de programación desde una edad temprana. Este kit no solo busca enseñar los principios fundamentales del IoT, sino también infundir habilidades cognitivas y de resolución de problemas que son esenciales en el mundo digital actual.

Este kit de aprendizaje proporcionará a los estudiantes una experiencia práctica y tangible en el mundo del IoT, permitiéndoles interactuar con sensores, actuadores, y dispositivos conectados de manera intuitiva y accesible. A través de proyectos y ejercicios diseñados en donde los estudiantes podrán explorar conceptos clave de programación, aprender a diseñar algoritmos, y comprender los principios subyacentes de la lógica computacional.

El objetivo principal de este kit es no solo proporcionar conocimientos técnicos, sino también cultivar habilidades de pensamiento crítico y creativo que son fundamentales en la sociedad digital actual y en el futuro.

1.1. Problema de investigación

La constante evolución de la tecnología muestra las necesidades del ser humano por encontrar nuevos campos los cuales requieran progreso, tal es el caso de la educación en donde se toma en consideración el artículo realizado por García (2022) en donde dice que: “en la actualidad la innovación es una aspiración y una condición determinante de nuestro tiempo, sustentada en una premisa de supervivencia: quien no innova, muere”.

Por esta razón es conveniente el desarrollo de habilidades que impulsen las demandas actuales.

Es así como Davila et al. (2021) menciona que en Ecuador "la carencia de diseño dentro de los materiales didácticos para desarrollar los temas a tratar hace que los aprendizajes sean de corta duración haciendo un obstáculo en la secuencia de su aprendizaje provocando vacíos en su conocimiento"

Gracias a estas premisas los conceptos en lógica de programación obtienen causas relevantes que afectan a la enseñanza de la lógica y sus posibles desarrollos tecnológicos,

Alcaraz et al. (2021) mencionan que "el aprendizaje de la programación es difícil y requiere un arduo trabajo por parte de los estudiantes", esto da paso a generar temor hacia el desarrollo de este tipo de habilidades.

Los métodos de enseñanza tradicionales de la lógica de programación se centran en la teoría y los conceptos abstractos. Esto puede ser desmotivador para los estudiantes que prefieren aprender de manera práctica donde se debería promover según el estudio realizado por Galván Cardoso & Siado Ramos (2021) en el cual se concluye diciendo que "se cree necesario que los alumnos visualicen a la escuela como una parte significativa en su vida". impulsando de esta manera la creatividad y la utilidad que en realidad refleja el desarrollo de habilidades lógico-matemáticas.

1.2. Justificación

El rápido avance tecnológico característico de las últimas décadas ha transformado profundamente la forma en que vivimos, trabajamos y nos relacionamos. En este contexto se ha generado las soluciones como la educación STEM en el boletín de prensa publicado por Senescyt (2018) detalla:" El término STEM que es abreviatura de los términos en inglés: Science, Technology, Engineering and Mathematics (Ciencia, Tecnología, Ingeniería y Matemáticas), busca que las ciencias no solo sean consideradas como materias, sino también como herramientas que permitan a los niños ser creativos, resolver problemas e innovar."

Este desarrollo acelerado de tecnologías innovadoras, como el Internet de las cosas (IoT), no solo redefine nuestra interacción con el entorno, sino que configura de manera

fundamental el panorama laboral del futuro, en donde el Ministerio de Educación (2021) en el artículo resalta que los trabajos del futuro están estrechamente ligados al proceso de programación.

Un estudio realizado por Raposo et al. (2022) menciona que "únicamente el 19,4% de los trabajos desarrolla proyectos puramente STEAM, siendo la robótica comercial de uso predominante." Esto proyecta al desarrollo de tecnología educativa, de igual manera se menciona que "el pensamiento computacional se presenta como una habilidad necesaria que puede desarrollarse desde niveles tempranos".

Para luego indagar según la malla curricular detalla por Ministerio de Educación (2017) para el bachillerato técnico en Informática, se enuncia que un objetivo es el "desarrollo de sistemas informáticos con lenguajes de programación" presentando el kit como un modelo de aprendizaje basado en práctica, que desarrolla creatividad e interés por la materia.

En este contexto, el diseño del kit físico surge como una respuesta estratégica para potenciar el material educativo con uso dela tecnología IoT, y que gracias a Domínguez & Pucha (2022) se menciona que " los docentes deben asumir la obligación y la responsabilidad de mantenerse actualizados aceptando los procesos innovadores dentro de la educación" presentando el proyecto como una integración de avances tecnológicos contemporáneos en la educación que pretende equipar a los estudiantes con habilidades no solo relevantes, sino también cruciales para su éxito en una sociedad cada vez más digitalizada.

1.3. Objetivos

1.3.1. Objetivo General

- Diseñar un "Kit de Aprendizaje IoT" físico el cual disponga de sensores y actuadores con el propósito de impulsar el desarrollo del pensamiento lógico en fundamentos de programación.

1.3.2. Objetivos Específicos

- Fundamentar teóricamente las temáticas sobre las necesidades y habilidades de aprendizaje de los jóvenes en relación con la programación y el IoT.

- Establecer requerimientos de diseño de software que permitan formar la arquitectura de un hardware modular empleando módulos inalámbricos.
- Desarrollar el contenido para el kit mediante la integración de hardware y software que permita accesibilidad y facilidad a los estudiantes.
- Realizar pruebas piloto del kit en un entorno educativo para evaluar la efectividad detallando los resultados en un informe final.

1.4. Alcance

Este proyecto sigue una metodología general en cascada, mientras que el desarrollo del kit se basa en un enfoque de prototipado evolutivo, lo que permite realizar ajustes y mejoras continuas para los estudiantes de Tercer año de Bachillerato Técnico en Informática.

El primer paso consistirá en investigar la enseñanza de la programación y los conceptos fundamentales, analizando los métodos pedagógicos actuales y los recursos disponibles. A partir de este análisis, se diseñará un kit de aprendizaje accesible, adaptado a los jóvenes y fundamentado en la metodología STEAM.

A continuación, se definirán los componentes clave del kit, destacando un dispositivo central que gestionará la comunicación con los sensores y actuadores. El desarrollo se llevará a cabo de manera iterativa, permitiendo ajustes progresivos a través de pruebas continuas y retroalimentación constante.

Finalmente, para validar la efectividad del kit, se realizarán pruebas piloto con los estudiantes, evaluando su participación, interés y comprensión de los conceptos de programación. Los resultados permitirán medir el impacto del kit en el proceso de enseñanza, destacando sus aportes al desarrollo educativo.

CAPITULO II. REVISIÓN BIBLIOGRÁFICA

El Internet de las Cosas (IoT) está revolucionando la educación al permitir un aprendizaje interactivo y personalizado a través de dispositivos conectados. Desde experimentos científicos en tiempo real hasta la monitorización del progreso estudiantil, el IoT facilita la recopilación de datos para adaptar la enseñanza a las necesidades de los alumnos.

Este capítulo analiza cómo la combinación de sistemas embebidos y tecnologías IoT para la enseñanza y el aprendizaje. Además, se revisará la literatura existente sobre conceptos clave como la programación, la lógica computacional, los lenguajes de programación visual, y las arquitecturas de redes y software que se utilizan en este campo, proporcionando una base sólida para la integración de estas tecnologías en el ámbito educativo.

2.1. Sistemas Electrónicos en la educación

Solano (2020) en el desarrollo de su tesis presenta como problemática la enseñanza y uso de un sistema electrónico Fig. 1, capaz de realizar enseñanza de idioma de manera creativa mostrando la oportunidad en el campo educativo, para motivar y resaltar interés a los estudiantes.

Figura 1

Prototipo electrónico

Fuente: Solano (2020) prototipo de juguete electrónico didáctico,

De igual manera Quilo & Javier (2023) señalan la necesidad de desarrollar nuevos contenidos educativos que generen interés en los estudiantes. Pues se consideran que, los alumnos pueden obtener una visión más clara de cómo funciona el mundo real al observar cómo la energía eléctrica se transforma en aplicaciones electrónicas. Su investigación se centra en la creación de un prototipo robótico Fig. 2, empleando la metodología STEAM y la Robótica Educativa para fomentar un aprendizaje más activo y contextualizado.

Figura 2

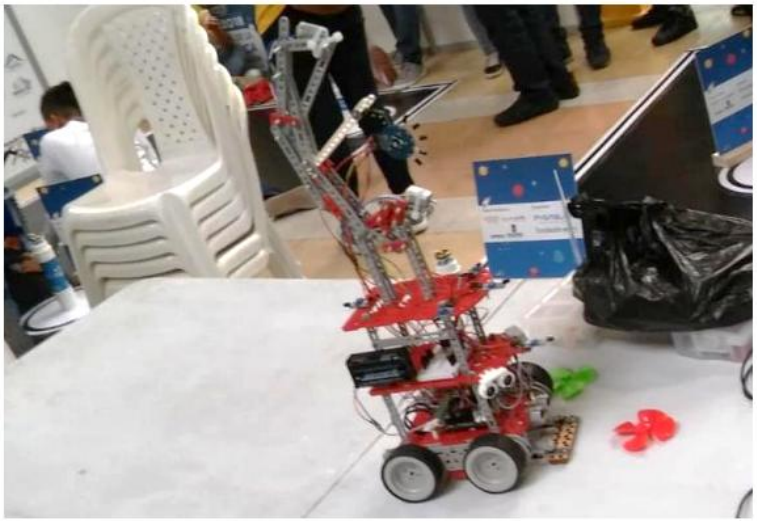
Prototipo interactivo presentado

Fuente: Quilo & Javier (2023) Material didáctico para la enseñanza de circuitos eléctricos (ley de Ohm)

Finalmente, estos conceptos pueden ser adaptados a sistemas mucho más complejos que se integran en prototipos robóticos, proporcionando una base sólida para el desarrollo de tecnologías avanzadas; la metodología STEM juega un papel crucial en este proceso, ya que fomenta un enfoque interdisciplinario que combina habilidades técnicas con la creatividad. Este enfoque estimula el pensamiento crítico y la innovación, habilidades esenciales en la robótica moderna. Como lo menciona Ramírez (2020) en su trabajo final donde la integración de estos conceptos fue clave en el desarrollo del prototipo robótico MALU Fig. 3, un ejemplo claro de cómo la metodología STEM permite la creación de sistemas robóticos avanzados que responden a desafíos educativos.

Figura 3

Robot presentado



Fuente: Ramírez (2020) Malú, primer prototipo de System Robotics

2.2. Internet de las Cosas (IoT)

En la recomendación UIT-T (2012) define como al IoT como una "Infraestructura mundial al servicio de la sociedad de la información que propicia la prestación de servicios avanzados mediante la interconexión (física y virtual) de las cosas" en cual permite el desarrollo de soluciones que propongan un avance tecnológico, y que por tal motivo el mismo desarrollo proponga un método alternativo de enseñanza y apoyo hacia el docente.

El alcance del Internet de las cosas (IoT) es verdaderamente amplio y está transformando rápidamente la forma en que interactuamos con el mundo que nos rodea. Este paradigma tecnológico abarca la conexión y comunicación de dispositivos físicos a través de la red, permitiendo la recopilación, intercambio y análisis de datos en tiempo real es así como Gallardo et al. (2023) señalan que este fenómeno no solo impacta industrias y aspectos cotidianos, sino que también desempeña un papel significativo en el ámbito educativo, revolucionando la forma en que se enseña y aprende.

El IoT ofrece oportunidades innovadoras en el campo educativo al proporcionar herramientas interactivas y experiencias de aprendizaje personalizadas ya que la integración de dispositivos conectados permite a los educadores crear entornos de

aprendizaje más dinámicos y participativos siendo ejemplos pizarras interactivas, sensores en laboratorios y hasta dispositivos portátiles en donde su base es un microcontrolador que puede ser adaptado para cualquier aplicación tal como se ilustra en la Fig. 4 y de esta manera fomentan la colaboración en el aula, el IoT facilita el acceso a información en tiempo real y promueve la adquisición de habilidades técnicas relevantes para la era digital.

Figura 4

Disposición de Tecnología

Fuente: Tomado de Servicio AI DALL-E

2.3. Arquitectura del Internet de las Cosas (IoT)

La arquitectura de redes inalámbricas es un proceso muy necesario para en la creación de infraestructuras de comunicación de red, donde al igual que una arquitectura física, el entorno inalámbrico requiere una cuidadosa planificación, consideración del entorno y atención a los detalles técnicos, por esta razón se vuelve necesario la selección de la tecnología adecuada hasta la distribución estratégica de puntos de acceso, el diseño de redes inalámbricas busca proporcionar cobertura confiable, alto rendimiento y seguridad.

2.3.1. Arquitectura General

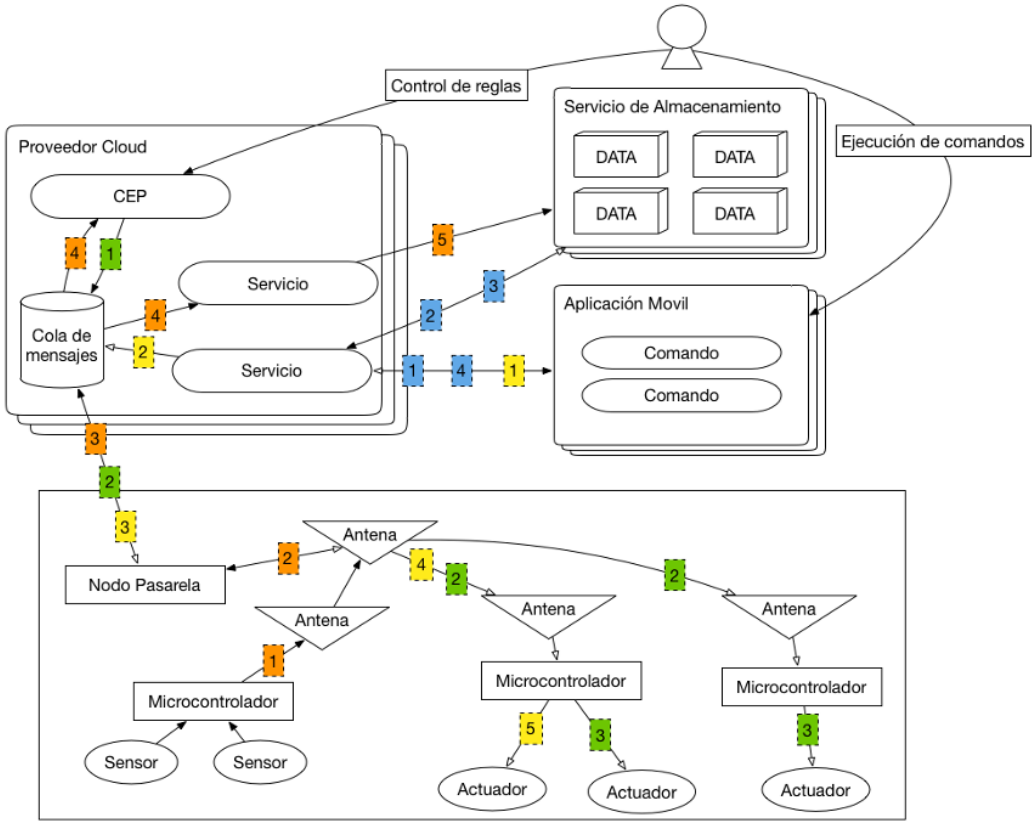
La arquitectura mostrada por Domínguez (2016) muestra la integración de un conjunto de sensores que se enlazan con uno o varios microcontroladores los mismos

están vinculados a dispositivos que facilitan la comunicación bidireccional con un nodo pasarela, el cual también dispone de un dispositivo inalámbrico conectado a él Fig. 5.

Una vez que el nodo pasarela recopila los datos provenientes de los sensores, estos son transmitidos a la nube para su almacenamiento y disponibilidad mostrando como objetivo final es no solo adquirir datos de los sensores, sino también poder activar otros dispositivos conectados a los microcontroladores.

Figura 5

Arquitectura IoT General



Fuente: Domínguez (2016) Arquitectura IoT

2.3.2. Arquitectura de Capas

Según Domínguez (2016) se disponen tres capas en la arquitectura general de un dispositivo de IoT, siendo representado en la Fig. 6

Figura 6

Arquitectura por capas IoT

Fuente: El autor

Adicionalmente, el autor presenta los siguientes conceptos para las capas propuestas

Capa percepción: Este estrato se encarga de adquirir las propiedades físicas de los objetos, como temperatura, humedad y ubicación, a través de sensores, y luego transformar esta información en señales digitales para su transmisión a través de la red.

Capa de red: La principal función de esta capa es enviar los datos recolectados por el nivel de percepción hacia su destino mediante la red, utilizando tecnologías como 3G, 4G, WiFi, Bluetooth, ZigBee, entre otras.

Capa de Aplicación: La tarea principal de la capa de aplicación consiste en desarrollar diversas aplicaciones según los objetivos y los datos adquiridos por la capa de percepción

2.4. Programación y Pensamiento Lógico

El pensamiento lógico y la programación son fundamentales en el desarrollo humano pues apoyan a la resolución de problemas en diversas áreas y que gracias a estos conocimientos se obtiene la capacidad de analizar, organizar y deducir información de manera coherente, lo cual es crucial en la toma de decisiones, planificación y generación de futuras soluciones.

2.4.1. Pensamiento Lógico

Según Naranjo et al. (2016) menciona que “El pensamiento lineal o lógico, es la manera en la cual las personas aprenden a pensar desde edades tempranas o a inicios de la vida escolar” en donde se resalta la importancia para la resolución de problemas para la vida futura, y se agrega según Castro (2023) que “el desarrollo del pensamiento lógico requiere de proposiciones, conceptos y razonamientos” que esta manera genera criterios de verdad denominados lógicos.

2.4.2. Fundamentos de Programación

La programación se refiere al proceso de diseñar e implementar un conjunto de instrucciones que permiten realizar una tarea específica o resolver un problema particular. Estas instrucciones, conocidas como código de programación, se escriben utilizando un lenguaje de programación que siguen una sintaxis y estructura específicas.

El objetivo fundamental de la programación es proporcionar instrucciones necesarias para realizar operaciones y resolver problemas de manera eficiente. Los programas pueden variar en complejidad desde simples instrucciones hasta aplicaciones y sistemas operativos completos.

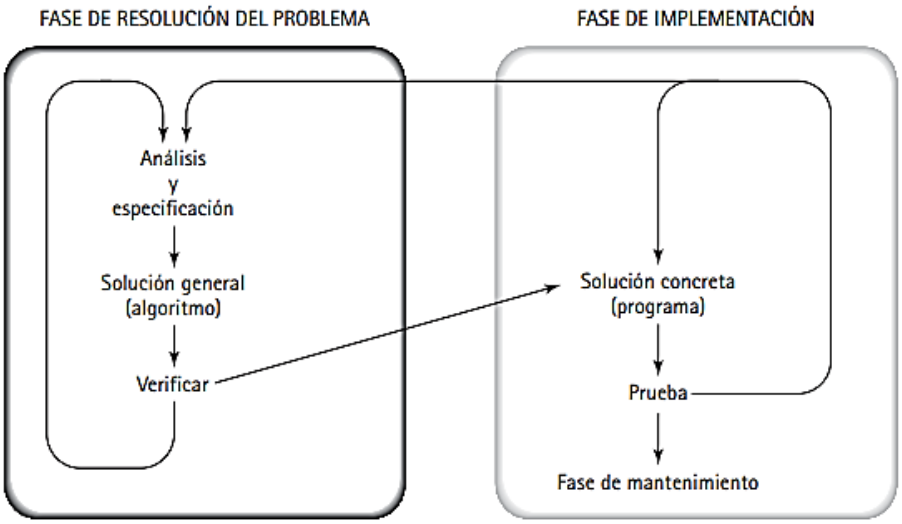
El proceso de programación sigue un enfoque sistemático como se menciona en Dale & Weems (2007) en donde el proceso comienza con el análisis y especificación del problema, definiendo claramente los requisitos y paso seguido, se desarrolla un algoritmo como solución general, acompañado de la verificación para asegurar que aborda efectivamente el problema.

En la fase de implementación, se traduce el algoritmo a un programa concreto, se realiza una prueba para confirmar el correcto funcionamiento y se corrigen posibles errores.

Finalmente, durante la fase de mantenimiento, se optimiza y actualiza el programa según sea necesario para adaptarse a cambios y mejorar su rendimiento, completando así el ciclo de desarrollo de software tal como muestra la Fig. 7.

Figura 7

Ciclo de desarrollo de software



Fuente: D*ale & Weems (2007)*

2.5. Paradigmas y Lenguajes de Programación

La programación abarca diversos enfoques y paradigmas que determinan cómo se estructuran y ejecutan las instrucciones dentro de un sistema informático. Los paradigmas de programación proporcionan conceptos que guían la solución de problemas mediante diferentes métodos. A su vez, los lenguajes de programación son herramientas que implementan estos paradigmas, permitiendo a los programadores escribir código de manera eficiente de tal manera se exploran los principales paradigmas y los tipos de lenguajes de programación que permiten desarrollar aplicaciones informáticas, desde los más cercanos al hardware hasta los de más alto nivel de abstracción.

Programación Imperativa:

La programación imperativa se basa en especificar cómo realizar operaciones paso a paso, pues dan las instrucciones necesarias al procesador para realizar un cambio inmediato de estado.

Programación Estructurada

Este paradigma permite implementar la programación imperativa con ciertos criterios de control, que mejoran la legibilidad y comprensión de las instrucciones, y además David et al. (2011) menciona que en sus estudios tres estructuras de control son esenciales, la secuencial, condicional e iterativa.

Programación Orientada a Objetos (POO):

La POO se caracteriza por una abstracción que contiene información esencial, la cual puede estar compuesta por variables y funciones que definen el comportamiento del objeto, adicionalmente se dice que la relación entre objetos puede darse con diversos métodos.

Lenguajes de Bajo Nivel

Los lenguajes de bajo nivel son aquellos que están estrechamente vinculados a la arquitectura y características específicas de la computadora.

Estos lenguajes proporcionan una representación directa de las instrucciones ejecutadas por la unidad de procesamiento central (CPU) y están altamente orientados al hardware, donde algunos ejemplos de lenguajes de bajo nivel incluyen el lenguaje ensamblador y el lenguaje máquina.

Los mismos ofrecen un mayor control sobre los recursos del sistema, pero requieren un conocimiento detallado de la arquitectura del hardware y son menos expresivos desde el punto de vista humano.

Lenguajes de Alto Nivel

Los lenguajes de alto nivel son aquellos que están diseñados para ser más accesibles y comprensibles para los programadores, proporcionando un mayor nivel de abstracción y alejándose de la complejidad de la arquitectura de hardware.

Estos lenguajes se centran en la productividad del programador y en la expresividad del código algunos ejemplos de lenguajes de alto nivel incluyen C, C++, Java, Python y muchos otros.

Estos lenguajes permiten una programación más eficiente y menos propensa a errores, facilitan la portabilidad del código entre diferentes plataformas y suelen contar con características de gestión automática de memoria y estructuras de control más avanzadas.

2.6. Lenguaje Visual de Programación (LVP)

Se menciona en Tonche García (2010) que la Programación Visual (LVP) es un paradigma de programación que se basa en el uso de representaciones gráficas e iconos en lugar de texto para expresar la lógica y las instrucciones de un programa por tal motivo en lugar de escribir líneas de código en un lenguaje de programación textual, se utilizan elementos visuales, como bloques, diagramas o iconos, para crear algoritmos y definir la lógica de los programas.

En un Lenguaje Visual de Programación (LVP), se propone emplear iconos generalizados que poseen una representación dual, abarcando tanto una parte lógica, que define el significado o la función, como una parte física, que se traduce en la imagen o símbolo visual correspondiente.

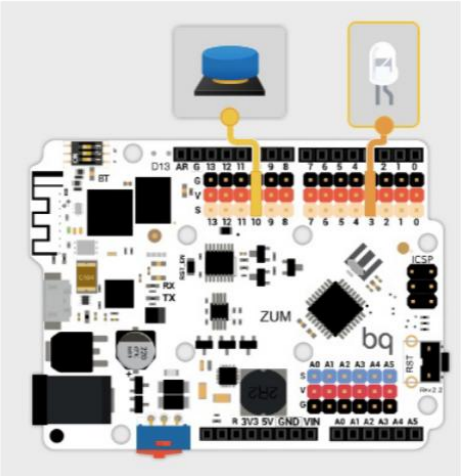
Este enfoque no solo facilita la comprensión de la lógica de programación, sino que también contribuye a la accesibilidad para principiantes al ofrecer una representación gráfica que simplifica la comprensión de la estructura y el flujo del programa.

2.6.1. Estado de Arte

Lara Bermúdez (2018) menciona en su trabajo la forma que da la adopción de estos lenguajes de programación visuales y explica que los mismos pueden ser integrados a microcontroladores Fig. 8 para la programación y construcción de “Smart cities” lo que proporciona un punto de partida, en gran medida señala que estos lenguajes permiten el desarrollo e integración de la electrónica y software.

Figura 8

Ejemplo de uso de programación visual

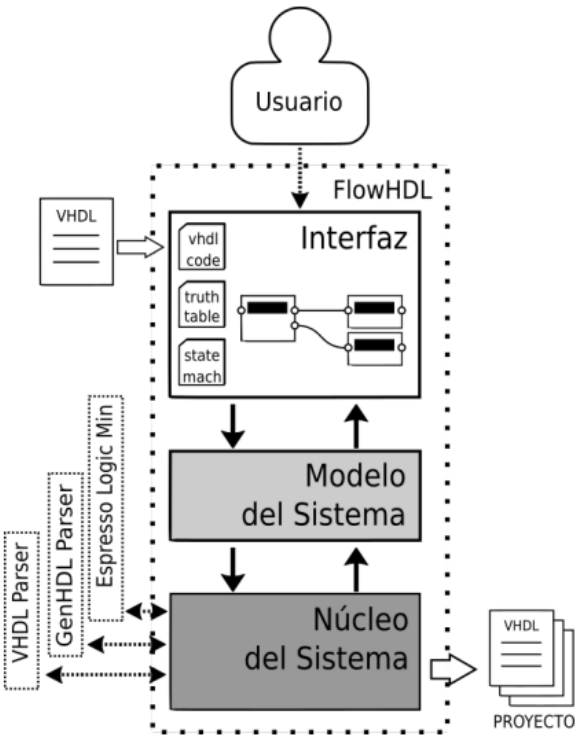


Fuente: Lara Bermúdez (2018) Conexión de componentes digitales

Antonelli & Gayoso (2021) en su artículo presenta una arquitectura de Software llamada "FLOWHDL" Fig. 9 en la cual muestra el proceso de interacción de programación visual con bloques, desde el core hasta el usuario final.

Figura 9

Arquitectura de Software FLOWHDL



Fuente: Antonelli & Gayoso (2021)

2.7. Redes Inalámbricas en IoT

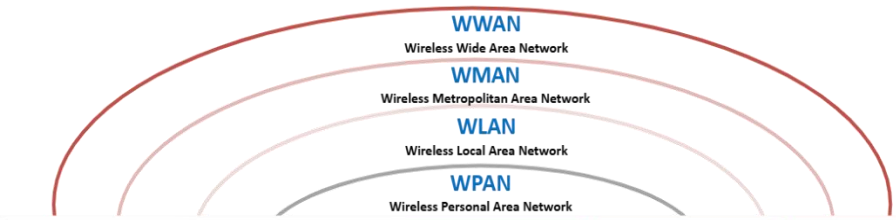
Vásconez Acuña (2014) menciona en su trabajo de tesis que una Red inalámbrica es un término que se utiliza en informática para designar la conexión de nodos sin necesidad de una conexión física, en donde dispositivos o nodos pueden compartir información empleando protocolos y aplicaciones que utilicen puertos.

2.7.1. Tipos de Redes Inalámbricas

Dentro del mismo documento se detallan cuatro tecnologías las cuales forman la parte representativa de este tipo en particular de redes inalámbricas, así la Fig. 10 muestra rápidamente las características para las cuales son desarrolladas.

Figura 10

Tipos de Redes Inalámbricas



	WPAN	WLAN	WMAN	WWAN
Estándares	Bluetooth	802,11 a/b/g/n	802,16 MMDS, LDS	GSM 2,5-3G HSDPA - HSUPA
Velocidad	<1Mbps	2-110 Mbps	22 Mbps	< 25 Mbps
Alcance	Corta	Medio	Medio-Largo	Largo
Aplicaciones	Peer-to-Peer	Redes Privadas	bucle de abonado final	acceso celular

Fuente: Adaptado https://farm8.staticflickr.com/7832/31911125777_b13ab7469e_b.jpg

2.7.2. Redes WPAN (Wireless Personal Area Network):

Las WPAN son redes de área personal inalámbricas diseñadas para la comunicación entre dispositivos cercanos, como Bluetooth y Zigbee, facilitando la conexión de dispositivos personales de corto alcance.

2.7.3. Redes WLAN (Wireless Local Area Network):

Las WLAN son redes de área local inalámbricas que permiten la conexión de dispositivos a una red mediante la tecnología Wi-Fi, como los estándares 802.11, proporcionando conectividad sin la necesidad de cables físicos.

2.7.4. WMAN (Wireless Metropolitan Area Network):

Un WMAN es una red de área metropolitana inalámbrica que abarca una ciudad o área geográfica extensa, proporcionando conectividad a alta velocidad a través de tecnologías como WiMAX, facilitando la conexión a larga distancia en entornos urbanos.

2.7.5. Redes WWAN (Wireless Wide Area Network):

Las WWAN son redes de área extensa inalámbricas que permiten la conexión de dispositivos a larga distancia, siendo ejemplos las redes móviles (3G, 4G, 5G) que ofrecen conectividad en áreas geográficas extensas.

2.8. Modos de Operación de Redes Inalámbricas

Según Vásconez Acuña (2014) dice que existen dos tipos de operaciones para una red inalámbrica, los cuales son los siguientes:

- Ad-Hoc
- Infraestructura

2.8.1. Ad-Hoc

Se menciona dentro del mismo documento que una red Ad-Hoc puede establecer comunicación directa entre los nodos existentes, es decir una conexión punto a punto, y que además añade que, si uno de estos dispone de acceso a Internet, el mismo puede

extender la funcionalidad hacia el resto de topología, en donde no existe una infraestructura detalla tal como se muestra en la Fig. 11 a.

2.8.2. Infraestructura

A diferencia del enfoque ad hoc, donde no existe un elemento central, en el modo de infraestructura se introduce un elemento de "coordinación", representado por un punto de acceso o estación base en donde ca dispositivo inalámbrico se conecta a este punto de acceso mediante el enlace inalámbrico, tal como muestra la Fig. 11 b.

Figura 11

Modos de Operación

Fuente: https://www.redesinalambricas.es/wp-content/uploads/2019/03/modos-inalambricos.jpg

2.9. Topologías de Redes Inalámbricas

Las topologías inalámbricas son configuraciones de redes que se basan en la comunicación a través de ondas electromagnéticas en lugar de cables físicos. Estas topologías son fundamentales en el mundo de las comunicaciones modernas, permitiendo la conectividad flexible y móvil en una amplia gama de dispositivos y entornos, de esta manera Fig. 12 muestra visualmente la estructura entre nodos IBM (2024).

2.9.1. Topología de Estrella:

Es la topología de red más común y estándar en entornos inalámbricos. En este diseño, todos los dispositivos se conectan a un punto central, como un concentrador o un enrutador, facilitando la gestión y el control de la red. Cada dispositivo se comunica directamente con el punto central, lo que simplifica la administración de la red.

2.9.2. Topología de Bus:

Esta topología no suele aplicarse generalmente, pero en el contexto inalámbrico, equivale a una red de malla completa que opera en un único canal. En este esquema, todos los dispositivos comparten el mismo canal de comunicación, lo que puede dar lugar a colisiones y limitaciones en el rendimiento de la red.

2.9.3. Topología de Árbol:

Esta topología se utiliza comúnmente por los Proveedores de Servicio de Internet inalámbricos. En ella, los dispositivos se organizan jerárquicamente, con nodos secundarios conectados a nodos principales. Este diseño facilita la expansión de la red y la gestión eficiente de grandes infraestructuras inalámbricas.

2.9.4. Topología de Anillo:

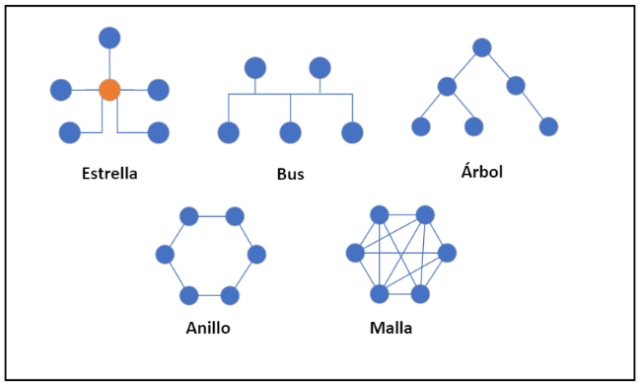
Esta topología puede utilizarse, en ella los dispositivos están conectados en forma de anillo, donde cada dispositivo está conectado al menos a otros dos. La comunicación se realiza en un solo sentido a lo largo del anillo, lo que puede simplificar el enrutamiento, pero también puede presentar desafíos si un nodo falla.

2.9.5. Topología de Malla:

Se refiere a una red donde cada dispositivo está conectado a múltiples nodos, permitiendo rutas redundantes y mayor robustez. Esta topología es flexible y resistente a fallos, ya que la comunicación puede seguir diferentes caminos. Su uso es común en aplicaciones críticas donde la disponibilidad y la fiabilidad son fundamentales, como en redes de sensores inalámbricos.

Figura 12

Topologías de red



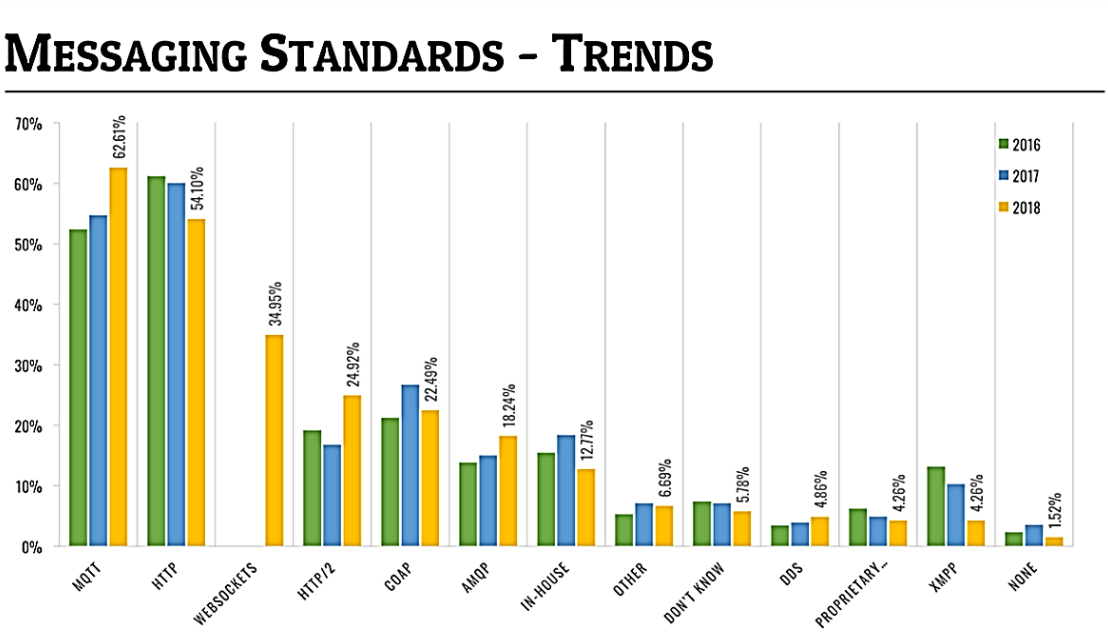
Fuente: adaptado de IBM (2024)

2.10. Protocolos de Comunicación en IoT

Orzuza (2022) en su trabajo muestra las tendencias de los principales protocolos dentro de la capa Aplicación en donde protocolos como MQTT, HTTP y COAP son las tecnologías más empleadas Fig. 13:

Figura 13

Tendencias de los Protocolos IoT



Fuente: Orzuza (2022)

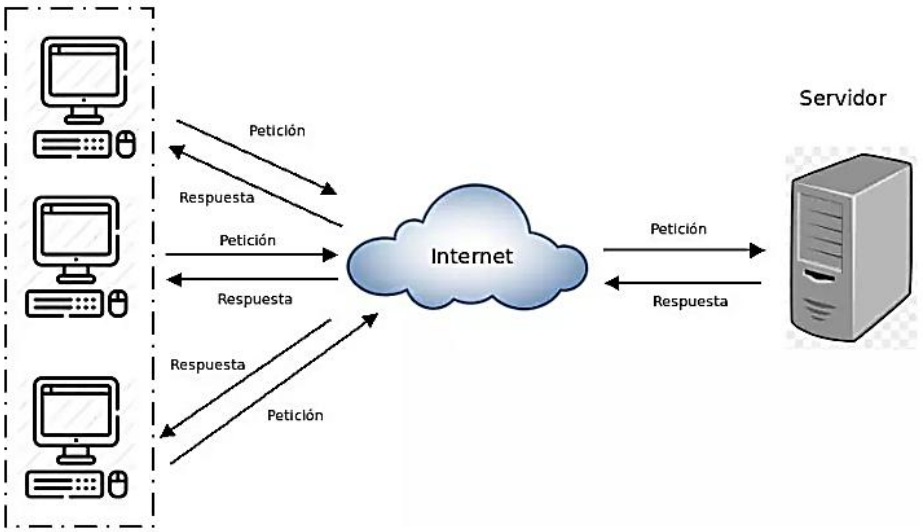
Por tal motivo se toma se referencia los mismos ya que de igual manera proponen soluciones al trabajo de integración planteado.

2.10.1. Arquitectura Cliente/Servidor

Según el documento revisado se tiene que la arquitectura cliente-servidor es un modelo de diseño en el cual las responsabilidades se dividen entre servidores y clientes, los clientes realizan solicitudes al servidor, que responde a estas peticiones como se evidencia en la Fig. 14.

Figura 14

Arquitectura Cliente Servidor



Fuente: https://infimg.com/bimg/2019/02/diagrama-cliente-servidor.jpeg

Para este modelo se evidencia las siguientes características

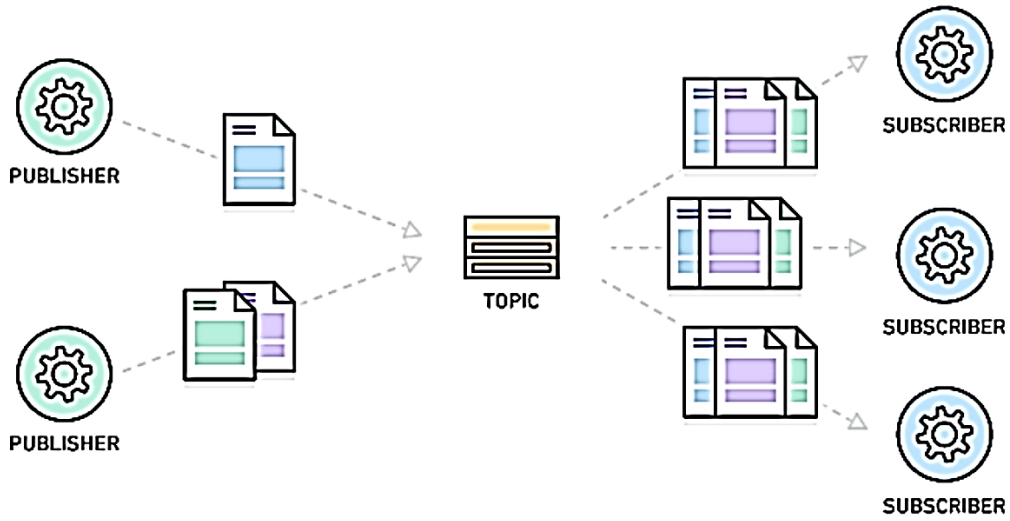
Interacción Cliente-Servidor: Tanto cliente y servidor pueden actuar como una sola entidad o de manera independiente e incluso en plataformas separadas por tanto cada plataforma es escalable de forma independiente, permitiendo actualizaciones o reemplazos tecnológicos transparentes.

2.10.2. Arquitectura Publicador/Suscriptor

En los entornos de IoT, que se basan principalmente en la nube, es crucial emplear la mensajería asincrónica para lograr un desacoplamiento eficiente entre los diversos componentes del sistema y permitir que estos intercambien información en respuesta a eventos como muestra la Fig. 15, además estos dispositivos tienen tendencia a disponer siempre de una conexión en línea.

Figura 15

Arquitectura Publicador/Subscriptor



Fuente: Amazon Web Services

En el modelo de publicación/suscripción (pub/sub), cualquier mensaje publicado sobre un tema es recibido de inmediato por todos los suscriptores de este. Este enfoque se utiliza en arquitecturas orientadas a eventos o para desacoplar aplicaciones, mejorando aspectos como el rendimiento, la confiabilidad y la escalabilidad. Varias instancias clientes pueden conectarse a intermediarios (brokers) y suscribirse a temas específicos, facilitando una comunicación eficiente y distribuida en entornos IoT.

2.10.3. HTTP (Hypertext Transfer Protocol):

HTTP es el protocolo estándar utilizado para la comunicación en la World Wide Web. Su función principal es posibilitar la transferencia de datos entre un cliente y un servidor.

Las interacciones entre ambos se basan en el intercambio de mensajes de texto y siguen el modelo de solicitud/respuesta, operando sobre el protocolo de transporte TCP.

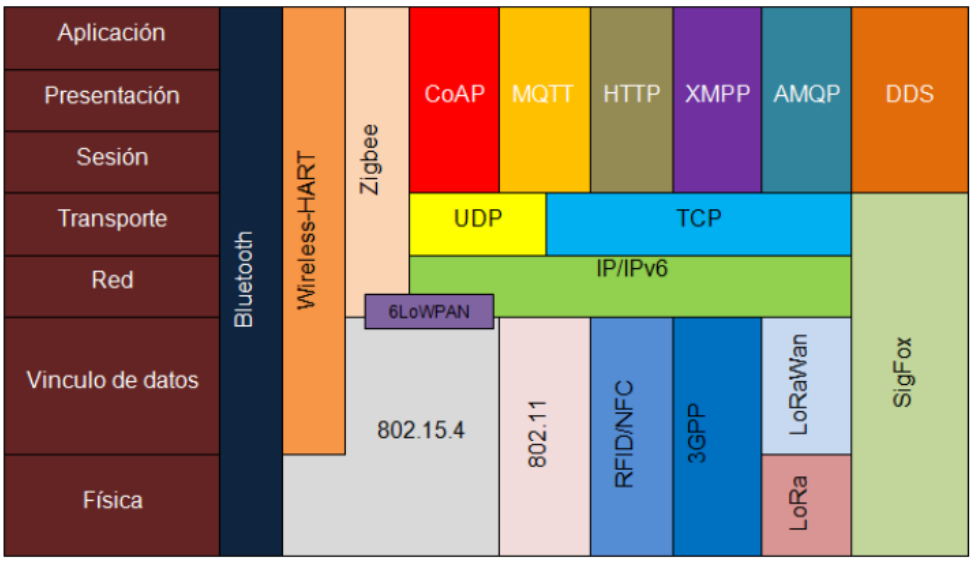
2.10.4. CoAP (Constrained Application Protocol):

Es un protocolo diseñado específicamente para entornos con recursos limitados, como dispositivos IoT y redes de sensores. Orientado a la eficiencia, CoAP sigue el modelo de solicitud/respuesta y opera sobre el protocolo de transporte UDP. Ofrece funcionalidades similares a HTTP, adaptándose especialmente a dispositivos con restricciones de memoria y ancho de banda.

Finalmente, la Fig. 16 muestra el despliegue de las tecnologías y su correspondencia con el modelo, OSI

Figura 16

Correspondencia Protocolos en la capa OSI



Fuente: González García et al. (2017)

2.11. Hardware para IoT

González García et al. (2017) Señalan que "sensores y actuadores son piezas fundamentales para IoT y posibilitan que objetos de la vida cotidiana interactúen entre ellos y con los seres humanos a través de Internet o redes dedicadas, recopilando

información del entorno o interactuando con él". Esta interacción genera una creciente necesidad de control eficiente en los sistemas IoT, lo que ha dado lugar a una variedad de plataformas que facilitan la gestión y el procesamiento de datos provenientes de estos dispositivos. Entre estas plataformas, los microcontroladores se destacan como una opción versátil y eficiente, ofreciendo una amplia gama de características que se adaptan a diferentes necesidades y aplicaciones. A continuación, se presentan algunas de las plataformas más utilizadas en este contexto:

2.11.1. Microcontroladores

Aníbal (2007) dispone en su trabajo que microcontrolador es un circuito integrado que funciona como un pequeño computador, diseñado para realizar tareas específicas dentro de un sistema embebido. Incluye un procesador, memoria (RAM y ROM), y periféricos de entrada/salida en un solo chip. Su bajo costo, tamaño reducido y capacidad para controlar sensores, actuadores y otras interfaces lo hacen ideal para aplicaciones de IoT, automatización, dispositivos electrónicos y sistemas de control, según esto se obtiene tres posibles plataformas Arduino, Raspberry PI y ESP-32 la Tabla 1 considera algunas características de estos:

Tabla 1
Plataformas consideradas

Plataforma	Descripción	Conectividad	Procesamiento	Interfaces de Comunicación	Aplicaciones
Arduino	Plataforma de hardware y software de código abierto para prototipos rápidos de IoT	Depende del modelo (algunas con Wi-Fi, Bluetooth)	Microcontroladores de 8 o 32 bits	UART, SPI, I2C, GPIO	Prototipos, enseñanza de electrónica y programación
Raspberry Pi	Computadora compacta de bajo costo con capacidad para sistemas operativos Linux	Wi-Fi, Ethernet, Bluetooth	Procesador ARM de 32 o 64 bit	UART, SPI, I2C, GPIO, USB	Proyectos complejos, servidores, IoT, multimedia
ESP32	Microcontrolador económico y potente con conectividad inalámbrica integrada	Wi-Fi, Bluetooth/BLE	Procesador de doble núcleo Tensilica Xtensa LX6	UART, SPI, I2C, GPIO	IoT industrial, dispositivos inteligentes, educación

2.11.2. Sensores

Un sensor es un dispositivo que detecta y mide cambios en su entorno físico o químico y convierte esta información en señales eléctricas o digitales. Estos dispositivos son esenciales, ya que proporcionan datos sobre variables ambientales como temperatura, humedad, luz, movimiento, entre otros.

La información recopilada por los sensores se utiliza para monitorear condiciones, tomar decisiones y realizar acciones específicas en respuesta a cambios en el entorno.

2.11.3. Actuadores

Un actuador es un dispositivo que realiza una acción física o mecánica en respuesta a señales eléctricas o digitales recibidas, los actuadores son fundamentales para ejecutar acciones en el mundo físico basándose en la información proporcionada por los sensores.

Pueden realizar tareas como activar o desactivar dispositivos, controlar el movimiento de motores, ajustar la intensidad de una luz, entre otras acciones físicas.

Los actuadores permiten que los sistemas IoT no solo recopilen datos del entorno, sino que también respondan y afecten activamente ese entorno.

CAPITULO lll. DISEÑO

En este capítulo se presenta el proceso de diseño y desarrollo del prototipo, donde se detallan los requerimientos necesarios de hardware y software que aseguran su funcionamiento adecuado en el entorno educativo. Luego, se explica el diseño físico, incluyendo los componentes seleccionados y su interconexión. Finalmente, se detalla el diseño del software, que abarca la comunicación, la estructura y el control del Backend y Frontend.

3.1. Metodología

Este proyecto sigue un enfoque de prototipado evolutivo, que permite desarrollar el prototipo de manera iterativa, realizando mejoras continúas basadas en pruebas y retroalimentación. Así, el prototipo se ajustará según las necesidades del entorno educativo y los estudiantes.

El proceso comienza con la selección de los componentes de hardware y software necesarios, siguiendo el estándar para la especificación de requisitos. Esto asegura que el diseño y la implementación estén alineados con los objetivos del proyecto.

A medida que se avanza en el desarrollo, se realizan iteraciones en las que se ajusta el diseño del hardware y la integración del software según los resultados obtenidos. Este enfoque permite que el sistema sea flexible, intuitivo y adaptándose a las necesidades de los estudiantes.

A continuación, se detalla el Modelo de Prototipado Evolutivo y su aplicación en este proyecto.

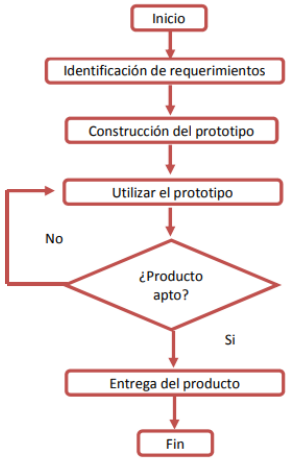
3.2. Modelo de Prototipado Evolutivo

El Modelo de Prototipado Evolutivo es una metodología de desarrollo iterativa donde el sistema se construye mediante sucesivas versiones, incorporando mejoras basadas en la retroalimentación de los usuarios por tal motivo el modelo es ideal para proyectar requisitos que pueden evolucionar a lo largo del tiempo, como es el caso del prototipo IoT de aprendizaje, en el cual la interacción con los usuarios (estudiantes) permite ajustar y optimizar sus funcionalidades.

A diferencia de modelos tradicionales, donde el diseño y desarrollo son estáticos, en el prototipado evolutivo se crean versiones funcionales del sistema receptando los requerimientos de estudiantes para desarrollar una primera etapa que facilita una evaluación temprana y continua tal como se evidencia en la Fig. 17.

Figura 17

Flujo del proceso de prototipado evolutivo

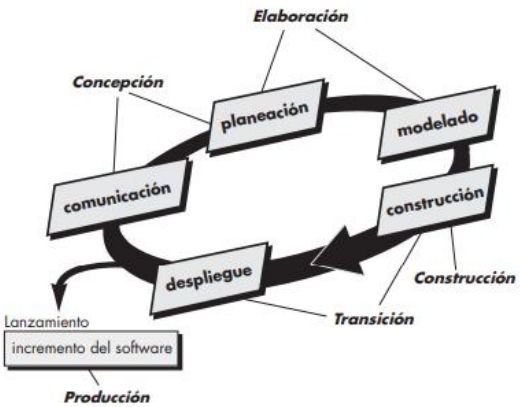


Fuente: Tomado de Mamani Vilca et al. (2019)

Según Pressman (2010) la metodología de prototipado evolutivo consta de cinco fases, las cuales se ilustran en la Fig. 18. Estas fases se han adaptado y aplicado al proceso general del presente trabajo, tal como se describe a continuación:

Figura 18

Fases de metodología de prototipado evolutivo



Fuente: Pressman (2010) pag.47

Fase de Concepción:

En esta fase, se definen se identifican los requisitos iniciales de hardware y software, se añade de igual manera el análisis de los recursos necesarios y disponibles en la unidad educativa.

Fase de Elaboración:

Se trabaja en el diseño detallado del sistema y se define la integración de los componentes y cómo se despliegan las funcionalidades clave que así mismo se provee la resolución de problemas técnicos iniciales y se realizan pruebas preliminares del prototipo inicial.

Fase de Construcción:

En esta fase, se empieza a codificar e implementar el sistema real. Esto incluye programar la interfaz visual y la, integración de funcionalidades en la plataforma de desarrollo y se propone la integración con el Backend se adiciona pruebas mientras se construye para asegurarse el correcto funcionamiento.

Fase de Transición:

Cuando el sistema está listo, se implementa mediante el diseño de estructura modela y se ajustan detalles correspondientes al diseño.

Fase de Producción:

El sistema se despliega para su uso real en donde se pretende monitorear el comportamiento real, lo resultados se recopilan para hacer mejoras o agregar nuevas funciones en el futuro.

3.3. Situación Actual

El presente proyecto se llevó a cabo en la Unidad Educativa Alberto Enríquez, la cual ofrece el Bachillerato Técnico en Informática. Para la validación del sistema propuesto, se realizaron pruebas con la participación de estudiantes y del docente de la materia.

Como parte del proceso, se llevó a cabo una inspección de la infraestructura tecnológica de la institución, donde se identificó la disponibilidad de computadoras Intel NUC (Next Unit of Computing) con soporte para conectividad inalámbrica.

Además, se impartió una clase introductoria en la que se evaluaron los conocimientos de los estudiantes sobre programación, electrónica e integración de microcontroladores con capacidades inalámbricas. Esta evaluación permitió analizar la base conceptual de los participantes en relación con la electrónica y su aplicación en el proyecto.

La Fig. 19 muestra la infraestructura disponible además del grupo de treinta y dos estudiantes que participó en la prueba.

Figura 19

Infraestructura y Grupo Participante

Fuente: El Autor

3.4. Concepción

En esta fase se evalúan los requerimientos de los usuarios, los cuales son fundamentales para definir la interacción con el sistema. A partir de estos requerimientos, se consideran los aspectos clave de un sistema embebido, que incluyen la integración de hardware y software. Estos componentes son la base sobre la cual el estudiante podrá interactuar con el sistema de manera intuitiva y eficiente.

3.4.1. Definición de abreviaturas

Para poder reducir la complejidad de los términos empleados por requerimientos se establece emplear el uso de acrónimos que detallen el tipo de requerimiento, la **Tabla 2** muestra la descripción de estos.

Tabla 2
Definición de abreviaturas

Abreviatura	Descripción
RqqU	Requerimiento especificado para usuario
RqqS	Requerimiento especificado de Software
RqqH	Requerimiento especificado de Hardware

3.4.2. Requerimientos de los Usuarios

Para el diseño del kit se consideran perfiles de usuarios que permiten evaluar la efectividad como herramienta educativa, por esta razón se especifican los siguientes actores involucrados para el proyecto:

Tabla 3
Actores Involucrados al sistema

Parte Interesada	Descripción	Interés
Estudiantes	Usuarios principales del kit, principalmente de bachillerato-universidad, interesados en aprender programación y electrónica aplicada al IoT.	Aprender de manera práctica sobre programación y electrónica de esta manera se recibe retroalimentación para depuración del Kit.
Docentes	Profesores de materias relacionadas con tecnología, informática o ciencias que pueden usar el kit como herramienta educativa.	Facilitar el aprendizaje de los estudiantes en programación y electrónica.
Aficionados	Personas interesadas en aprender IoT de manera autodidacta o recreativa, sin necesariamente ser estudiantes.	Adquirir conocimientos sobre IoT y explorar aplicaciones prácticas.
Desarrolladores de Software	Encargados de la creación y mantenimiento del software asociado al kit.	Desarrollar una plataforma confiable y educativa.
Fabricantes de Hardware	Proveedores de los componentes electrónicos (sensores, actuadores, microcontroladores, etc.) utilizados en el kit.	Proveer hardware de calidad que se integre fácilmente con el sistema

Gracias a la encuesta realizada al docente de la unidad educativa se obtuvo cierto conocimiento sobre los requerimientos base para los estudiantes, los datos de dicha encuesta están reflejados en el **ANEXO A: Formato de Entrevista para Docente** del presente trabajo.

Finalmente, realizando un análisis de los resultados obtenidos por dicha encuesta, se presentan a continuación los requerimientos de usuarios Tabla 4 que guiarán el desarrollo del kit educativo.

Tabla 4
Requerimientos de Usuario

Nomenclatura	Requerimiento	Descripción	Prioridad
RqqU_01	Lenguaje de Programación	El sistema debe integrar un lenguaje adecuado el cual facilite depuración de código rápidamente.	Alta
RqqU_02	Desarrollo visual	El sistema debe permitir la creación de programas de forma visual, facilitando el aprendizaje de la programación.	Alta
RqqU_03	Interfaz amigable	El software debe tener una interfaz fácil de usar para facilitar el aprendizaje de los usuarios.	Alta
RqqU_04	Documentación y tutoriales	El sistema debe incluir recursos educativos como tutoriales y documentación clara para guiar a los usuarios.	Media
RqqU_05	Conexión con hardware	El sistema debe interactuar con sensores, actuadores y microcontroladores, permitiendo actividades prácticas.	Media
RqqU_06	Costo	El sistema desarrollado debe mantener un costo bajo como herramienta educativa	Alta
RqqU_07	Compatibilidad Wi-Fi	El sistema debe permitir la conexión inalámbrica para su uso en dispositivos como computadoras, tabletas o smartphones.	Media

3.4.3. Requerimientos del Software

Para el software del kit, se adopta un enfoque basado en sistemas embebidos, lo que permite que el código se ejecute directamente sobre el microcontrolador sin necesidad de un sistema operativo completo. Aunque se dispone de computadoras, estas tienen una

capacidad limitada de memoria RAM, lo que puede afectar el rendimiento, un sistema embebido optimiza la gestión de recursos, garantizando que el kit sea autónomo, eficiente y capaz de integrar funciones adicionales sin comprometer su rendimiento debido a las restricciones de memoria.

Para su implementación, se evalúan diversos lenguajes de programación con el objetivo de optimizar el rendimiento y la eficiencia en el uso de recursos del microcontrolador, además de facilitar el desarrollo, la Tabla 5 presenta algunos lenguajes de programación más empleados con sus ventajas para su implementación en un microcontrolador:

Tabla 5

Lenguajes de Programación para microcontroladores

Lenguaje de Programación	Ventajas	Limitaciones
Micro Python	Entorno de desarrollo rápido y fácil.	Menos eficiente en la gestión de recursos (memoria y velocidad).
	Ideal para principiantes, con buena documentación.	Limitado en aplicaciones que requieren alta eficiencia.
C/C++	Control preciso sobre el hardware y los recursos del sistema.	Requiere experiencia avanzada en programación y arquitectura del sistema.
	Mejor optimización de memoria y rendimiento.	Curva de aprendizaje pronunciada, especialmente para principiantes.
JavaScript (Node.js en IoT)	Fácil integración con plataformas web y interfaces visuales.	Menos eficiente en recursos en comparación con C/C++.
	Compatible con entornos web y servicios en la nube.	No es la opción óptima para sistemas embebidos con recursos limitados.

Para componer el sistema se propone utilizar un intérprete en lugar de un compilador ya que este enfoque permite realizar modificaciones rápidas en el funcionamiento del sistema sin necesidad de compilar nuevamente el programa.

Considerando que el objetivo del sistema es idear una herramienta visual que ayuden con procesos lógicos se consideran las herramientas de la Tabla 6 para que el proceso de enseñanza sea: didáctico, practico y despierte más interés por parte de estudiantes.

Tabla 6

Herramientas de Software Visual

Herramienta	Ventajas	Limitaciones
Blockly	Basado en bloques, fácil de usar para principiantes.	Requiere implementación manual para integrarse con hardware específico.
	Personalizable e integrable en aplicaciones web.	No es una plataforma completa, sino una herramienta para construir entornos de programación visual.
	Genera código en varios lenguajes como JavaScript, Python y C++.	Algunos lenguajes deben ser realizados por el desarrollador para ajustar a la solución
Node-RED	Especializado en IoT y automatización.	Requiere conocimientos previos en IoT y flujos de datos.
	Permite la programación visual con flujos de datos.	No es una opción adecuada para programación estructurada tradicional.
	Integración con dispositivos IoT y APIs externas.	La integración de APIs externas suele ser complicada al usuario
Open Roberta (NEPO)	Enfocado en educación STEM con soporte para robots y hardware educativo.	No es tan personalizable como Blockly para otros proyectos.
	Compatible con micro:bit, Arduino, Raspberry Pi y más.	No tan extendido como Scratch en educación general.

Mediante el análisis de los conocimientos anteriores se realiza el análisis para extraer los requerimientos necesarios, que debe alcanzar el software en el desarrollo del prototipo y así integrar los requerimientos de usuarios Tabla 7.

Tabla 7
Requerimientos de Software

Nomenclatura	Requerimiento	Descripción	Importancia
		Sistema Base	
RqqS_01	Interprete en tiempo real	El sistema debe responder a la interacción del usuario en tiempo real con sensores y actuadores	Alta
RqqS_02	Facilidad de desarrollo y depuración	Se debe proporcionar un sistema base accesible para realizar cambios en el código sin procesos complejos de compilación y carga.	Alta
RqqS_03	Sintaxis sencilla y legible	El software debe permitir la ejecución de scripts que puedan modificarse sin reiniciar el sistema, facilitando la actualización de funcionalidades.	Alta
RqqS_04	Compatibilidad con hardware embebido	El sistema debe poder adaptarse a limitaciones de hardware optimizando recursos	Alta
		Soporte de entorno Visual	
RqqS_05	Soporte para programación visual	El sistema debe ser compatible con herramientas de programación visual que faciliten el aprendizaje de los usuarios.	Alta
RqqS_06	Interfaz Amigable	La aplicación debe ser accesible al usuario sin la necesidad de software adicional	Media
RqqS_07	Fácil Integración	La herramienta permite la integración sencilla tanto en el desarrollo base del microcontrolador como desarrollo visual para interfaz de usuario	Media

3.4.4. Selección de Software

Para la selección de Software se consideran los requerimientos previamente mencionados, en donde se propone separar la selección en dos procesos tanto del sistema base como el soporte de la herramienta visual y así evaluarlos, de tal manera que 0 indica un valor que el requisito no cumple, mientras que un valor de 1 indica que sí Tabla 8

Tabla 8
Elección de Software (Sistema base)

Requisito	Micro Python	C/C++	JavaScript (Node.js en IoT)
RqqS_01	1	0	1
RqqS_02	1	0	0

RqqS_03	1	1	1
RqqS_04	0	1	0
TOTAL	**3**	**2**	**2**

Elección: El lenguaje que mejor se adapta a la propuesta es Micro Python, ya que, aunque no está altamente optimizado para sistemas embebidos, permite obtener resultados aceptables gracias a su ejecución mediante intérprete lo que permite fácil depuración y ejecución.

Para determinar la herramienta de programación visual se consideran los tres requerimientos restantes, la Tabla 9 indica la valoración para elección de Software

Tabla 9

Elección de Software (Herramienta Visual)

Requisito	Blocky	Node-Red	Open Roberta (NEPO)
RqqS_05	1	1	1
RqqS_06	1	0	0
RqqS_07	1	0	1
TOTAL	**3**	**1**	**2**

Elección: La herramienta de programación visual seleccionada es Blockly, ya que facilita la integración con sistemas embebidos de manera sencilla. Además, su alto nivel de personalización y flexibilidad en el desarrollo la hacen una opción versátil, permitiendo adaptar y expandir sus funcionalidades según las necesidades del proyecto.

3.4.5. Requerimientos del Hardware

Para los requerimientos de hardware se realizó una encuesta dirigida hacia los estudiantes, para comprobar los conocimientos placas embebidas y componentes electrónicos **ANEXO B: Formato de Encuesta para Estudiantes**, la cual mediante el análisis de los datos se observó que solo el treinta por ciento de estos habían trabajado con microcontroladores.

Adicionalmente, el costo del hardware influye directamente a la selección de dispositivo por lo que debe considerarse al momento de su selección ya que de esta manera el prototipo puede ser accesible para una amplia gama de estudiantes y entornos educativos sin comprometer la calidad de los componentes y la experiencia de aprendizaje.

Tomando en cuenta las consideraciones de usuario y software la Tabla 10 evidencia la comparación con las respectivas placas.

Tabla 10

Comparativa de Microcontroladores

Característica	ESP-32	BeagleBone Black	Raspberry Pi Pico	ESP-8266
Conectividad	Wi-Fi y Bluetooth	Ethernet, Wi-Fi (con módulo adicional)	No tiene Wi-Fi ni Bluetooth integrados	Solo Wi-Fi
Rendimiento	Procesador de doble núcleo, 240 MHz	Procesador ARM Cortex-A8, 1 GHz	Microcontrolador de doble núcleo, 133 MHz	Procesador de 80 MHz
GPIOs Totales	34 GPIOs (dependiendo de la versión)	65 GPIOs	26 GPIOs	17 GPIOs
Entradas Analógicas (ADC)	18 canales ADC (12 bits)	7 canales ADC (12 bits)	3 canales ADC (12 bits)	1 canal ADC (10 bits)
Salidas PWM	Hasta 16 pines con PWM	Soporte de PWM en varios pines	Hasta 26 pines con PWM	Hasta 8 pines con PWM
Interfaces de Comunicación	SPI, I2C, UART, CAN	SPI, I2C, UART, CAN, USB	SPI, I2C, UART	SPI, I2C, UART
Costo Aproximado	$5 - $8 USD	$50 - $60 USD	$4 - $6 USD	$3 - $5 USD
Soporte de Lenguajes	C, C++, MicroPython, JavaScript (NodeMCU), Arduino	C, C++, Python, JavaScript, Bash	C, C++, MicroPython	C, C++, MicroPython, Arduino

Con los datos previos se realiza una síntesis de los requerimientos que cubren los aspectos para la elaboración del sistema Tabla 11.

Tabla 11

Requerimientos de Hardware

Nomenclatura	Requerimiento	Descripción	Importancia
RqqH_01	Modularidad	El hardware debe contar con 10 entradas de tipo analógico y digital (GPIO) para poder emplear diversos sensores y actuadores	Alta

RqqH_02	Escalabilidad	Debe poseer procesador, memoria y almacenamiento suficiente para ejecutar programas más complejos a medida que los estudiantes avanzan en su aprendizaje	Media
RqqH_03	Conectividad	Debe incluir Wi-Fi para la comunicación con plataformas externas.	Media
RqqH_04	Soporte	Lenguajes compatibles para control e integración de aplicaciones en el dispositivo, así como documentación.	Alto
RqqH_05	Gestión de Energía	Funcionamiento con baterías o alimentación externa	Medio
RqqH_06	Costo	Costo de la plataforma equilibrado con su rendimiento	Alto

3.4.6. Selección de Hardware

Para la selección de Hardware se consideran los requerimientos previamente mencionados y se evalúan las plataformas previamente mencionadas en donde 0 indica un valor de que la plataforma no cumple mientras que un valor de 1 indica que si cumple la Tabla 12 muestra la valoración para la selección de Hardware.

Tabla 12

Selección de Hardware

Requisito	ESP-32	BeagleBone Black	Raspberry Pi Pico	ESP-8266
RqqH_01	1	0	0	0
RqqH_02	1	1	1	0
RqqH_03	1	0	0	1
RqqH_04	1	1	1	1
RqqH_05	0	0	0	0
RqqH_06	1	0	1	1
TOTAL	**5**	**2**	**3**	**3**

Elección: La plataforma de desarrollo selecciona es la ESP-32 ya que la misma ofrece un procesador doble núcleo a 240 (MHz), 520 KB de SRAM y 4 MB de almacenamiento, de igual manera soporta hasta 18 canales canales de GPIO con ADC para sensores o actuadores implementa también comunicación Wifi y Buetooth finalmente su costo es relativamente bajo comparado con otros dispositivos.

3.5. Elaboración

Para abordar los requisitos previamente mencionados, el microcontrolador ESP-32 funciona como núcleo de la solución pues en su memoria se albergará una aplicación web que funcionará como Frontend, permitiendo a los usuarios interactuar con el sistema a través de una interfaz gráfica y el Backend Micro Python el cual gestionará las entradas y salidas del microcontrolador, permitiendo el control y monitoreo de los dispositivos conectados. La comunicación entre los nodos se manejará mediante sockets, lo que garantiza una transmisión eficiente de datos entre los dispositivos del sistema.

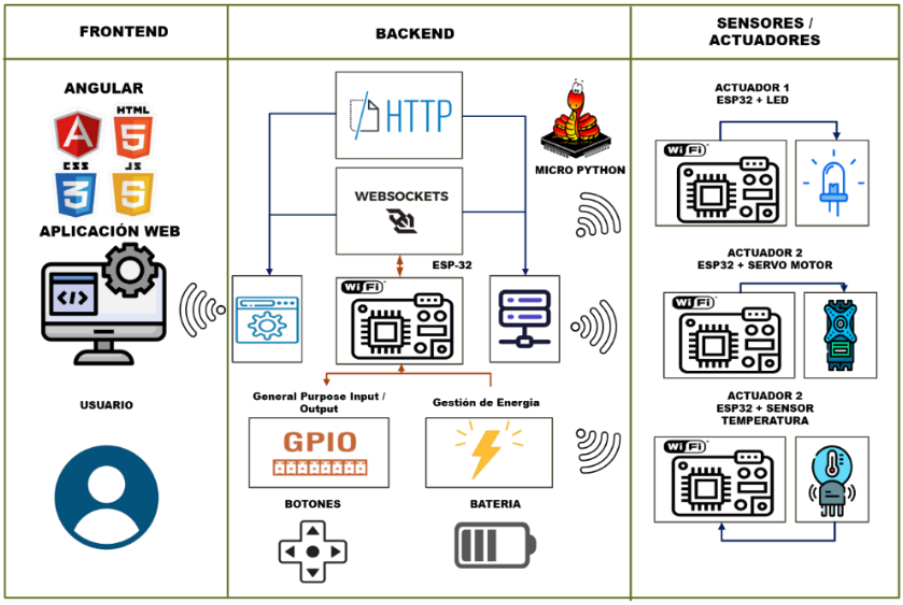
3.5.1. Arquitectura

La arquitectura propuesta para este sistema basado en ESP-32 realiza múltiples roles dentro de la solución, integrando tanto el Frontend como el Backend en un solo dispositivo como lo muestra la Fig. 20.

Este enfoque unificado no solo simplifica el diseño y reduce los costos, sino que también asegura un sistema autónomo, escalable y de fácil mantenimiento. La utilización de Micro Python como intérprete permite realizar ajustes y actualizaciones de código sin la necesidad de recompilar o reiniciar el sistema, dando al sistema flexibilidad y rapidez en la modificación de los proyectos.

Figura 20

Arquitectura General de Prototipo



Fuente: El Autor

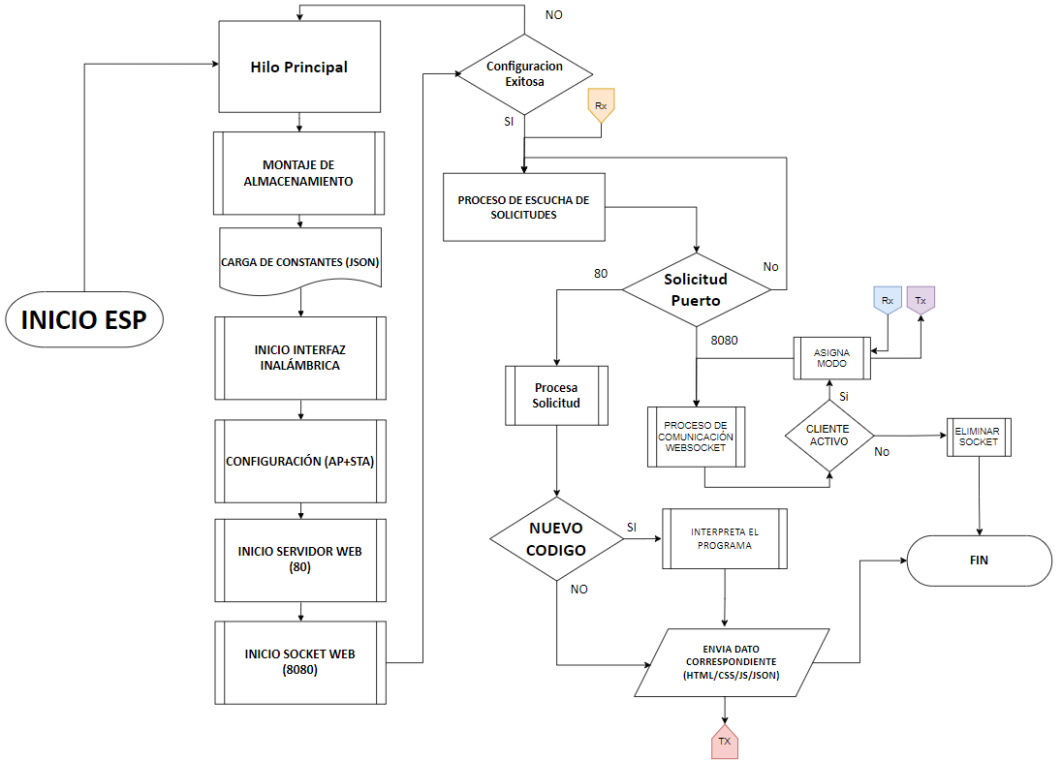
Procesos en el Microcontrolador

La Fig. 21 muestra los procesos necesarios en el microcontrolador. En el diagrama se detallan los pasos previos a la inicialización del sistema, por lo que el microcontrolador debe contar con los archivos de configuración iniciales. Además, se incluyen los servicios fundamentales para la comunicación: servidor web en el puerto 80 y un web socket en el puerto 8080 para comunicación bidireccional en tiempo real.

Las solicitudes procesadas a través de la interfaz inalámbrica en el puerto 80 pueden contener el código a interpretar. En ese caso, el código es cargado en la memoria principal y ejecutado en un bucle asíncrono, finalmente el microcontrolador prepara una respuesta que debe ser enviada para finalizar la interacción.

Figura 21

Diagrama de flujo para microcontrolador

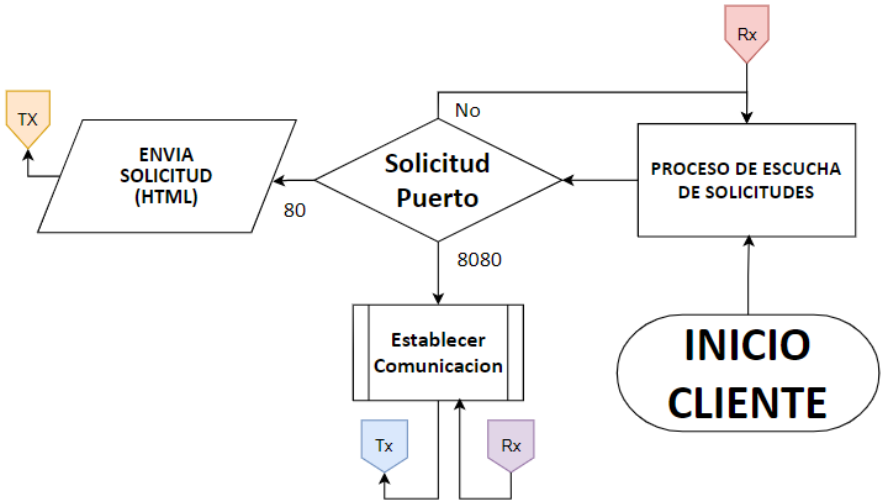


Fuente: El Autor

La Fig.22 muestra el diagrama de interacción con el cliente.

Figura 22

Diagrama de flujo para cliente



Fuente: El Autor

3.5.2. Diseño General de Interfaz

Para el diseño del sistema es necesario asegurar tanto Backend como Frontend en el mismo dispositivo en donde se proponen áreas que permiten reutilización y despliegue sobre tecnologías HTML y que a continuación se muestra un boceto rápido del Frontend previsto Fig. 23.

Barra de Control: Permite gestionar acciones como cargar, guardar o ejecutar el código por medio de solicitudes http.

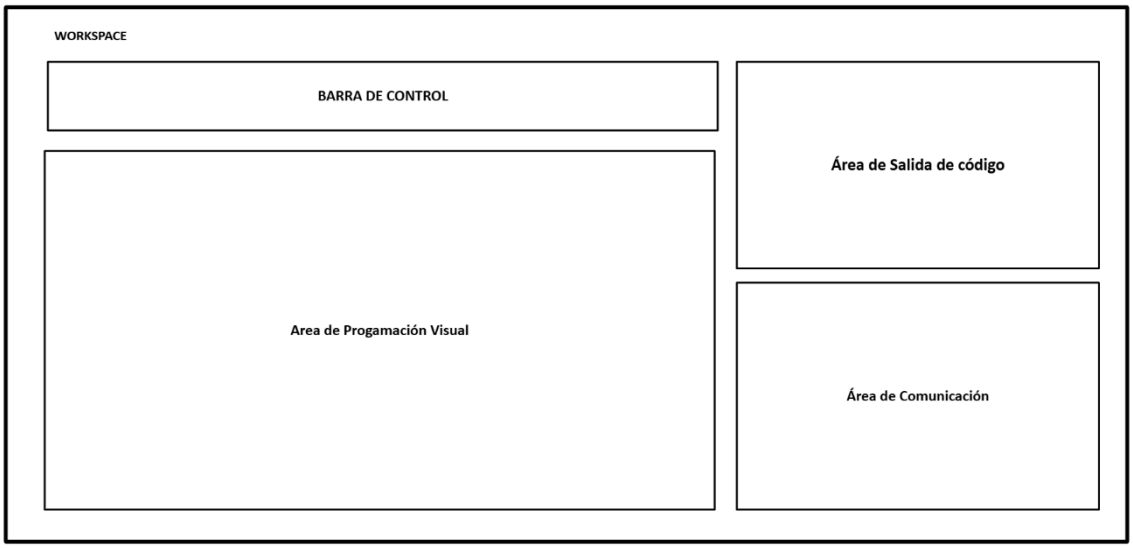
Área de Programación Visual: En esta área se puede construir objetos visuales que faciliten la impostación de bloques y estructuras de control visualmente.

Área de salida de Código: En esta área se evidencia la salida del lenguaje propio para el sistema, este lenguaje será usado por el intérprete cuando se ejecutan acciones como Ejecutar desde la barra de control.

Área de Comunicación: En esta área se define las salidas por medio de web sockets lo que permite interacciones cliente-servidor en tiempo real

Figura 23

Diseño General Frontend



Fuente: El Autor

Frontend. - La aplicación con la interfaz general se ejecuta en el microcontrolador de esta manera el usuario podrá visualizar el estado de los dispositivos conectados, controlar actuadores, leer valores de sensores además de ejecutar comandos en tiempo real el despliegue sobre tecnología HTML permite ser accesible desde cualquier navegador sin necesidad de instalar software adicional.

Control de Entradas / Salidas. -La gestión de entradas/salidas pueden incluir datos provenientes de sensores (por ejemplo, temperatura, humedad, presencia) que el ESP32 deberá leerlos de forma periódica. Las salidas incluyen el control de dispositivos como luces, motores o relés y al mismo tiempo debe procesar las solicitudes recibidas desde el Frontend y ejecutar las acciones correspondientes sobre el hardware enviando los resultados o estados de vuelta al usuario.

Comunicación. - Para garantizar una comunicación entre el ESP32 y Frontend, se emplean sockets ya que gestionados correctamente permiten responder y gestionar correctamente la interacción del usuario hacia el microcontrolador como se mencionó previamente.

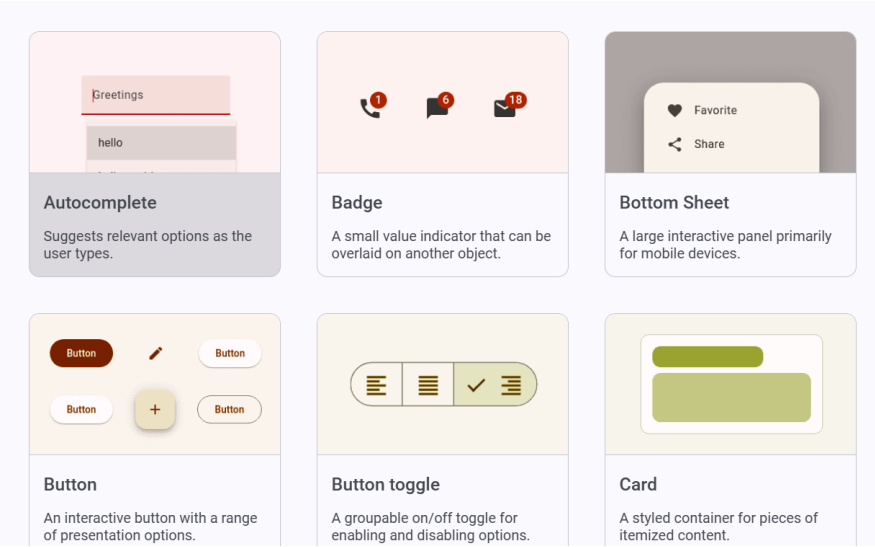
3.5.3. Frontend Aplicación Web

El Frontend se desarrollará utilizando Angular y TypeScript, lo que permite crear una interfaz web dinámica y estructurada. Esta interfaz incluye componentes mencionados en el diseño general siendo cubiertos por el Framework de Angular.

Componentes de Angular: La aplicación se organiza en componentes que gestionan diferentes funcionalidades de la interfaz, como la programación visual, la consola de salida y el editor de código. Ya que junto con el uso de TypeScript permite aprovechar tipado estático y características orientadas a objetos, lo que facilita la gestión y escalabilidad del código en el Frontend la Fig. 24 muestra algunos ejemplos de Framework a emplear.

Figura 24

Componentes de Angular

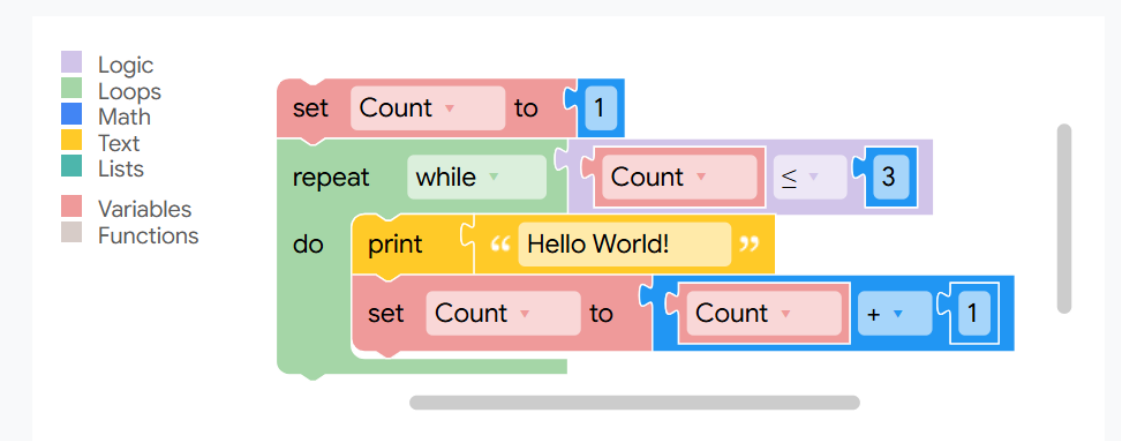


Fuente: Tomado de: https://material.angular.io/components/categories

Blockly: El Frontend integra Blockly, una herramienta de programación visual que permite crear código mediante bloques gráficos en lugar de escribir código manualmente. Este enfoque facilita la comprensión de conceptos de programación para usuarios sin experiencia previa la Fig. 25 muestra un ejemplo de la interfaz visual de Blockly creada por Google.

Figura 25

Programación visual con Blocky



Fuente: Tomado de https://developers.google.com/blockly?hl=es-419

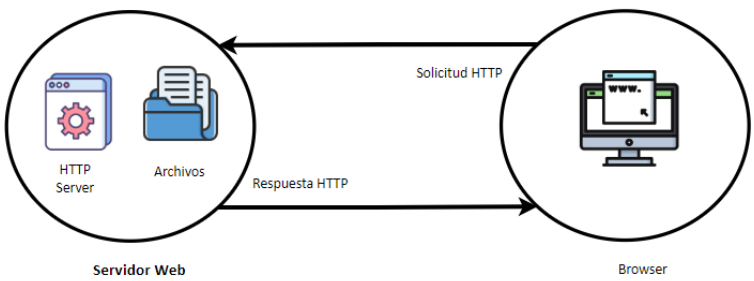
3.5.4. Diseño de Backend

El diseño de Backend está compuesto por varios servicios, incluyendo un servidor web, un servidor de sockets y un cliente socket por tal se describe el funcionamiento cada uno de estos elementos:

Servidor Web. **-** El dispositivo debe un servidor web embebido, que atiende las solicitudes HTTP del Frontend. Este servidor no solo se encarga de servir los archivos estáticos del Frontend (HTML, CSS, JavaScript), sino también de procesar las solicitudes que realiza el usuario a través de la interfaz web.

Figura 26

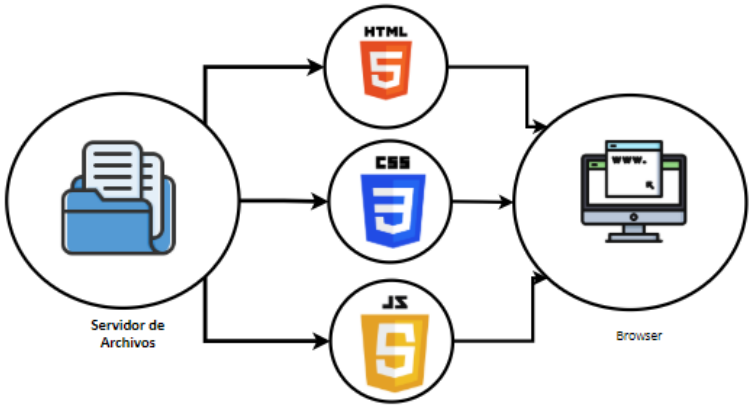
Servicios de Servidor Web



Fuente: El Autor

Servidor de Archivos. - Cuando el cliente se conecta, el servicio de archivos entrega los documentos necesarios para que el navegador cargue la aplicación web como se muestra en la Fig. 27, si no los encuentra se devuelve una respuesta con estructura HTML en formato plano.

Figura 27
Servicio de archivos estáticos



Fuente: El Autor

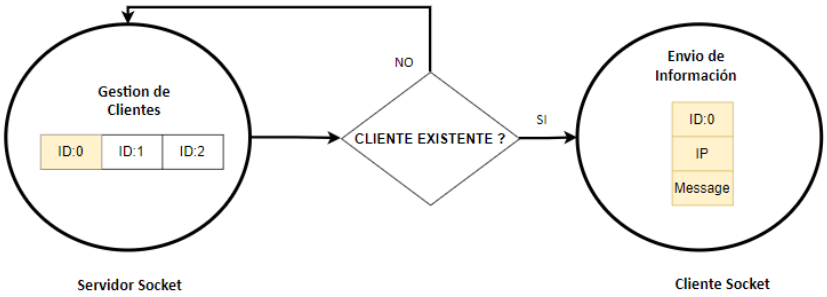
Servidor de Sockets. - El servidor de sockets Fig. 28 establece y mantiene conexiones de comunicación bidireccional y se mantiene escuchando lo que permite una comunicación en tiempo real sin la latencia a diferencia de las solicitudes HTTP tradicionales.

Conexión bidireccional. - A través de los WebSockets, el ESP32 puede enviar datos al Frontend, como actualizaciones de sensores o estados de dispositivos. Además, el Frontend puede enviar comandos al Backend sin necesidad de volver a cargar la página o realizar una nueva solicitud HTTP.

Asincronía. - Las conexiones de WebSocket son gestionadas de manera asíncrona, lo que permite al Backend procesar múltiples solicitudes al mismo tiempo sin bloquear el flujo de ejecución, lo cual es crucial para la eficiencia del sistema.

Figura 28

Diseño de Servidor Web Socket

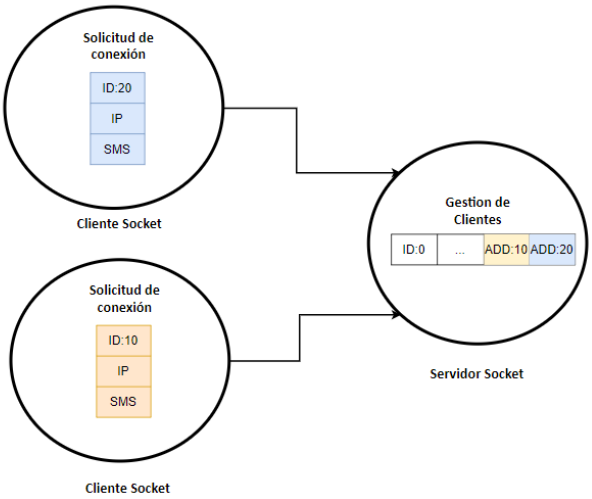


Fuente: El Autor

***Cliente Socket.* -** El cliente socket Fig.29 dentro del Backend permite al ESP32 establecer conexiones con otros dispositivos en la red. Este cliente se utiliza para enviar y recibir datos a través de conexiones de red, lo que permite al ESP32 comunicarse con otros nodos o dispositivos IoT, por ejemplo, para compartir datos o recibir datos de otros microcontroladores.

Figura 29

Estructura de cliente web socket



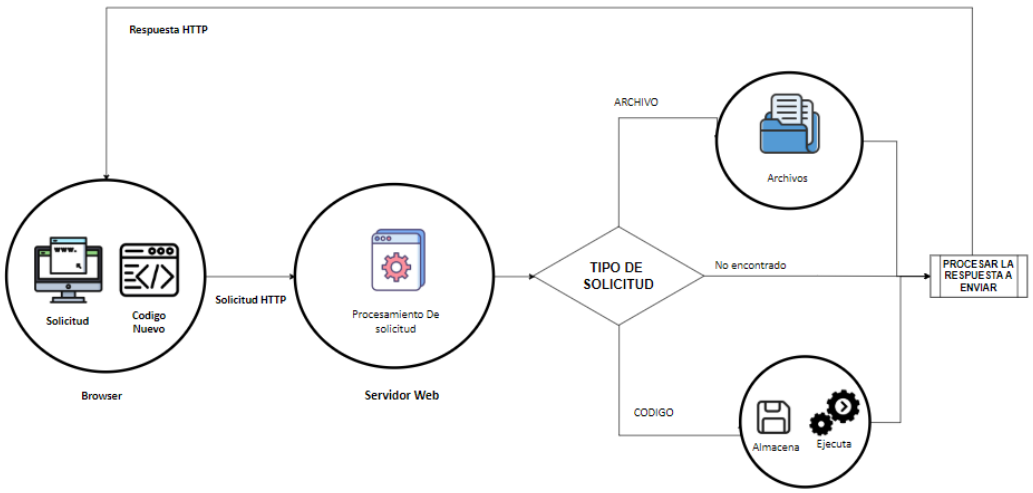
Fuente: El Autor

***Ejecución de código.* -** Una parte importante del Backend es la ejecución de las variables globales setup y loop, que son enviadas desde el Frontend. Estas variables

definen el comportamiento del sistema y se ejecutan de manera asincrónica utilizando uasyncio, un módulo que permite la ejecución de tareas concurrentes en el ESP32 sin bloquear el proceso principal la Fig. 30 evidencia la recepción de la solicitud HTTP con la información para el procesamiento de código del usuario.

Figura 30

Procesamiento de Solicitud HTML con código de ejecución



Fuente: El Autor

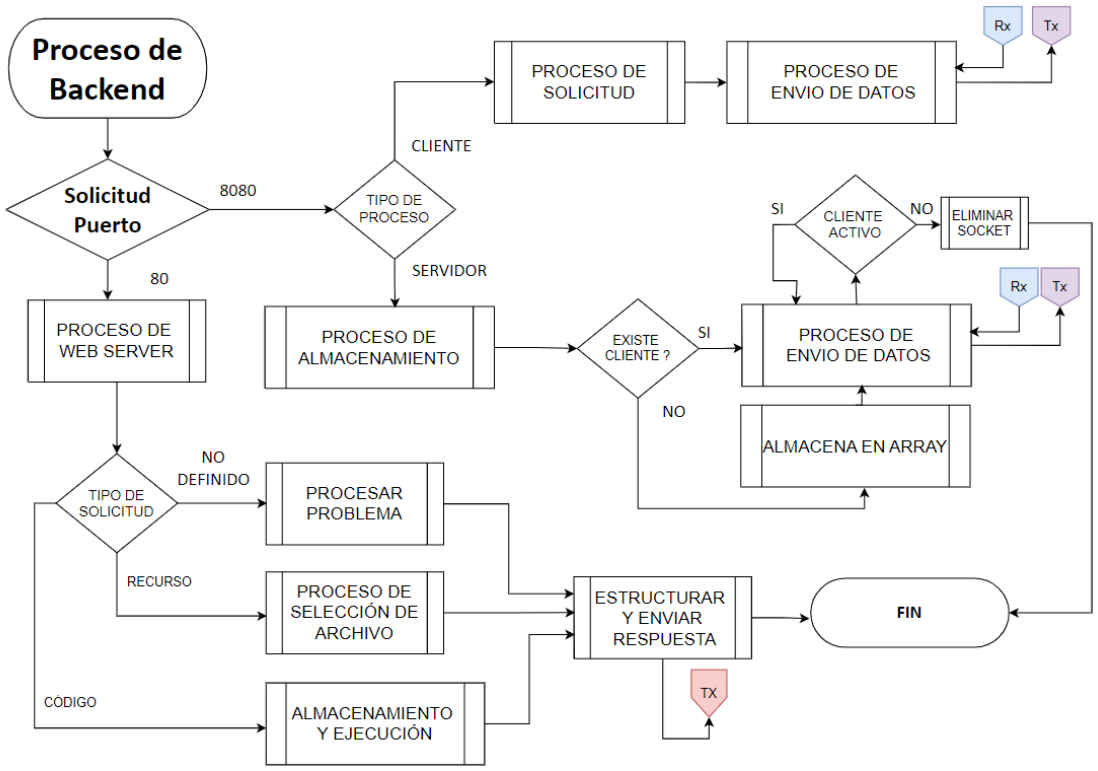
Setup. **-** Contiene la configuración inicial del sistema, como la inicialización de sensores, actuadores y configuraciones de red. Esta función se ejecuta una sola vez al iniciar el sistema.

Loop. **-** Contiene la lógica de ejecución continua del sistema, que se ejecuta de forma repetitiva. Se encarga de la lectura de sensores, el monitoreo de eventos y la gestión de salidas. Gracias a la ejecución asincrónica, el loop puede operar de manera eficiente sin bloquear otros procesos.

El Backend gestiona estos procesos de manera que puede recibir nuevas configuraciones o comandos desde el Frontend sin interrumpir otras operaciones del sistema y finalmente la estructura básica de los procesos se sintetiza en la Fig. 31.

Figura 31

Procesos a ejecutar en Backend



Fuente: El Autor

3.5.5. Diseño Físico

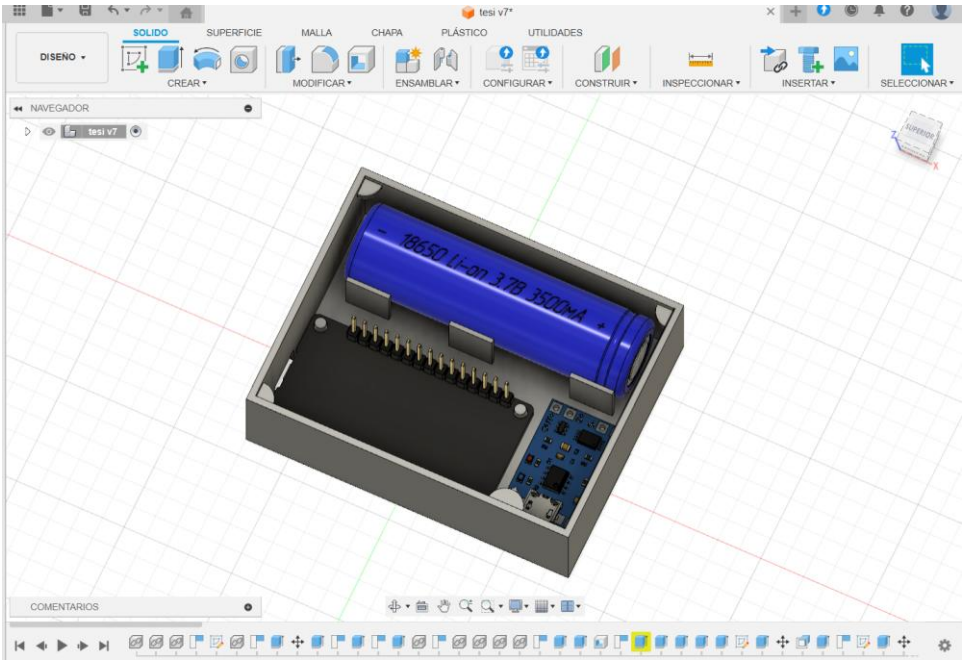
El diseño físico se realizó con ayuda del Software Fusión 360 y de esta manera se considera la carcasa en dos partes, el panel frontal, que almacena potenciómetros pulsadores, conectores led y switch, mientras que back-cover tiene; una batería, modulo para gestión de carga TP-4056 y el microcontrolador ESP 32.

3.5.6. Back Cover

Para implementar para la elaboración del back-cover como muestra la Fig. 32 se dispone de los elementos como modelos 3D lo que permite disponerlos y optimizar el espacio disponible.

Figura 32

Back cover de sistema



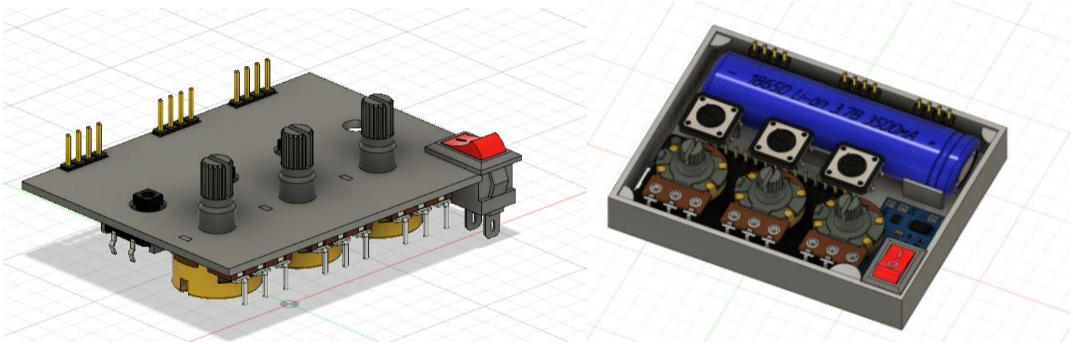
Fuente: El Autor

3.5.7. Panel Frontal

Para el Panel frontal se consideran tres potenciómetros y tres pulsadores y un diodo led RGB, el mismo será utilizado el practicas posteriores tal como se introduce en la Fig. 33 a**,** de igual manera gracias a los componentes prefabricados estos pueden ser mostrados y acoplados en una disposición final Fig. 33 b.

Figura 33

Distribución de elementos

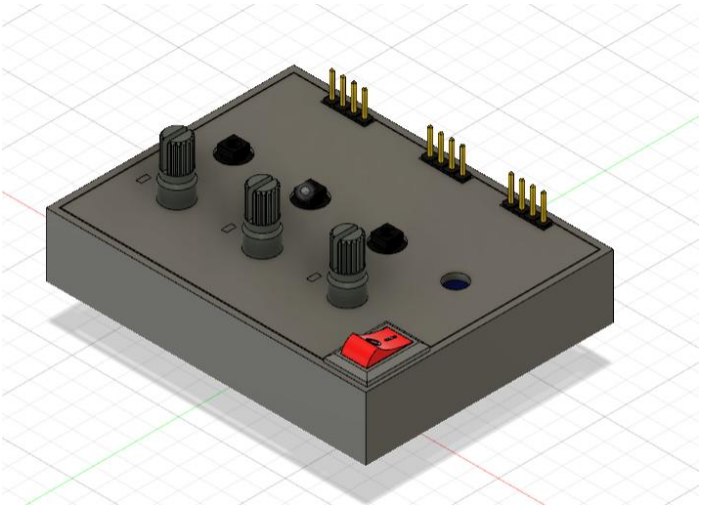


a) Panel Frontal b) Disposición Final

Fuente: El Autor

Finalmente, la estructura completa se evidencia en la Fig. 34 la cual muestra una vista previa mientras que el panel frontal permite a su vez conexión de sensores y actuadores mediante los conectores frontales que tiene distribución VCC, GND y 4 GPIO para el control de dispositivos externos.

Figura 34
Diseño Completo



Fuente: El Autor

3.6. Construcción

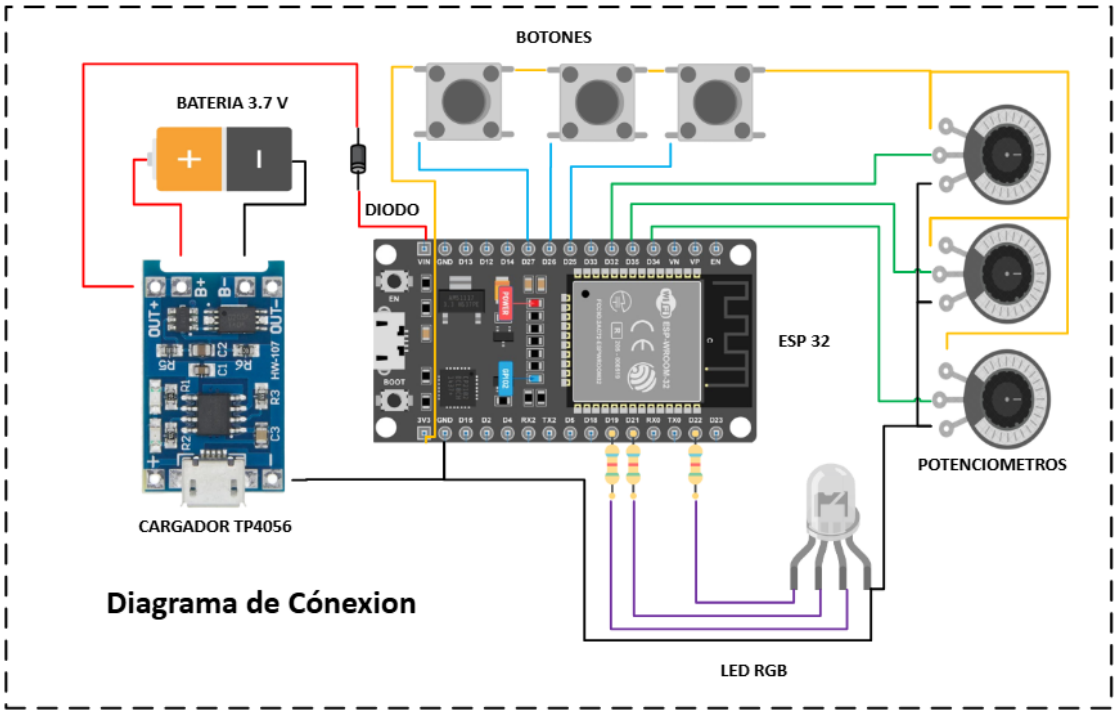
Para la fase de construcción se considera los conceptos previamente abordados y se desarrolla la versión base para el prototipo en la cual se designa el uso de componentes correspondientes a electrónica discreta, así como su suministro de energía y de esta manera los mismos serán fáciles de conseguir, disponer o reemplazar.

3.6.1. Componentes Externos de Hardware

El microcontrolador ESP32 actúa como el base principal del hardware, gestionando entradas, salidas y comunicación para interactuar con diversos componentes y está diseñado para demostrar conceptos de control, sensores y comunicación dentro del entorno IoT educativo tal como muestra la Fig. 35.

Figura 35

Esquema de conexión Hardware



Fuente: El Autor

Componentes Principales

ESP32. - Es el componente principal y controla todo el sistema. Se encarga de recibir entradas de los sensores, botones, disponer entradas como salidas y al mismo tiempo gestionar la comunicación inalámbrica.

Módulo TP4056. - Este módulo se utiliza para cargar la batería del sistema de manera segura. Está conectado a una batería que alimenta a la ESP32, permitiendo portabilidad y autonomía en el dispositivo.

Botones. - Se usan como dispositivos de entrada manual para permitir la interacción directa del usuario con el sistema.

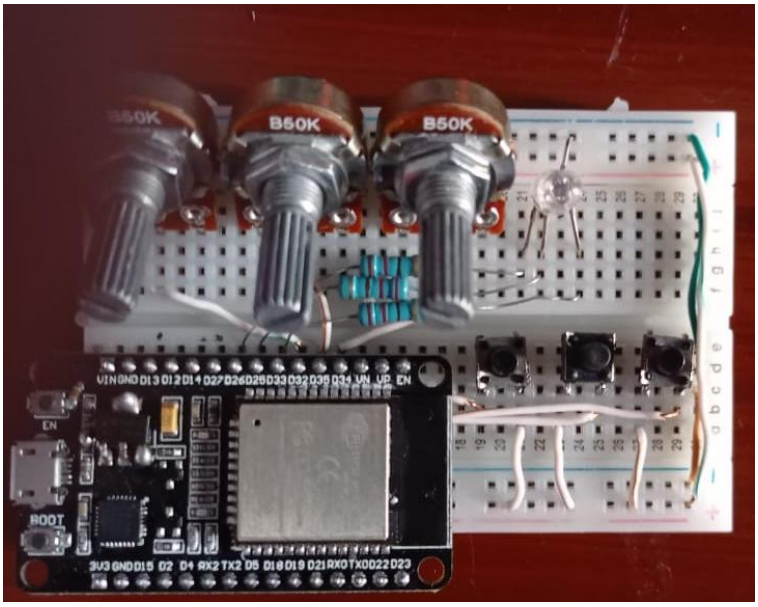
Potenciómetros. - son utilizados para ajustar parámetros o para simular entradas analógicas variables en el sistema.

LEDs. - Actúan como indicadores visuales para mostrar el estado del sistema o retroalimentar al usuario.

Gracias al modelo de prototipado rápido se puede preparar la estructura de Hardware a utilizar mediante un protoboard de esta manera gracias a esquema de conexión realizado permite comprobar conexiones y componentes externos Fig. 36.

Figura 36

Primer prototipo rápido



Fuente: El Autor

3.6.2. Programación de Backend (Micro Python)

Para iniciar la programación del entorno base, se considera que la plataforma cuenta con recursos limitados. Por ello, la implementación de asincronía resulta fundamental para gestionar eficientemente los procesos definidos en el diseño, permitiendo su ejecución sin afectar el rendimiento del sistema.

La Fig. 37 muestra los procesos de configuración de la interfaz inalámbrica seguido de los procesos de escucha para servidor web y servidor de web socket empleando métodos asíncronos para no saturar el microcontrolador.

Figura 37

Código Principal de inicio de procesos.

```
# Configuración del AP
SSID = 'ESP-S2'
PASSWORD = '123456789'
start_access_point(SSID, PASSWORD)

# Iniciar el bucle asincrónico
try:
    loop = asyncio.get_event_loop()

    loop.create_task(start_web_server(80))  # Servidor HTTP en puerto 80
    loop.create_task(start_websocket_server(8080))  # Servidor WebSocket en puerto 8080
    loop.run_forever()
except Exception as e:
    print(f"Error general en el sistema: {str(e)}")
```

Fuente: El Autor

La definición de la Fig. 38 muestra el proceso de servidor web el cual previamente fue declarado en un puerto específico sobre la interfaz inalámbrica, como se menciona anteriormente el servidor queda en escucha o espera a nuevas conexiones, para no bloquear el tiempo de procesamiento de CPU es necesario que el proceso sea asíncrono y no bloqueante esto permite ejecutar otros procesos simultáneamente.

Figura 38

Definición para servidor web

```
# Inicia servidor HTTP en el puerto indicado
async def start_web_server(port):
    addr = socket.getaddrinfo('0.0.0.0', port)[0][-1]  # Dirección de escucha
    server_socket = socket.socket()
    server_socket.bind(addr)              # Asocia socket a la dirección
    server_socket.listen(5)               # Escucha hasta 5 conexiones
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  # Reutiliza puerto
    print(f"Servidor web corriendo en el puerto {port}")

    await accept_connections(server_socket)  # Acepta conexiones entrantes

# Acepta conexiones HTTP entrantes
async def accept_connections(server_socket):
    global dispo_ips
    server_socket.setblocking(False)  # No bloqueante
    while True:
        try:
            client, addr = server_socket.accept()  # Acepta cliente
            if addr[0] not in dispo_ips:
                dispo_ips.append(addr[0])  # Registra IP si es nueva
            print('Conexión desde:', addr)
            await handle_client(client)  # Atiende cliente
        except Exception:
            await asyncio.sleep(0.1)  # Espera si no hay conexión
```

Fuente: El Autor

Después del inicio del servidor web se ejecuta la definición del servidor de web socket en el puerto 8080 que permite la comunicación en tiempo real, se observa en la

Fig. 39 que hay un array que almacena los clientes, así como un subproceso que controla su estado o tiempo de actividad.

Figura 39

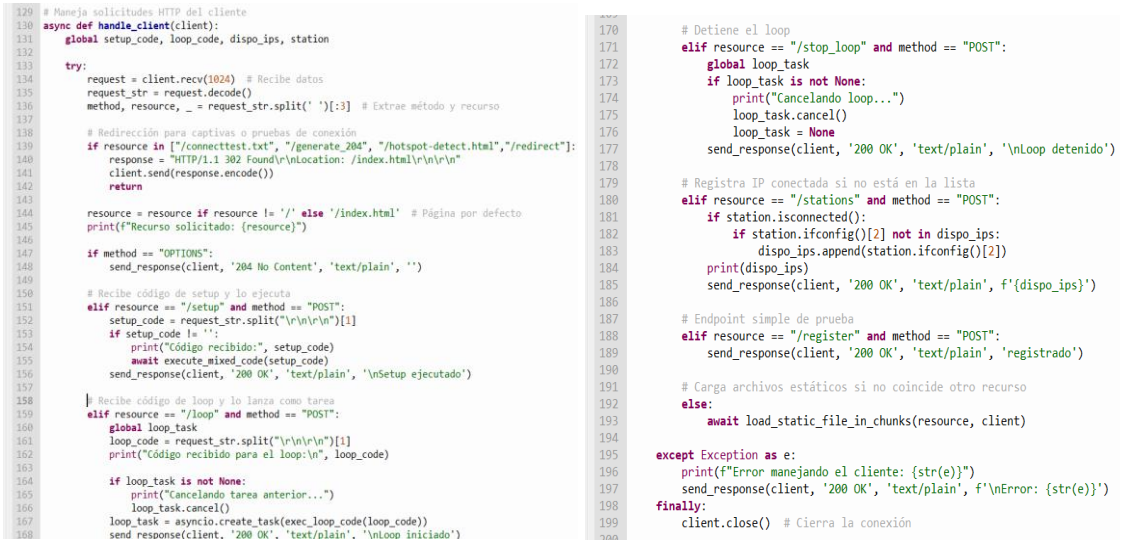
Definición para servidor web socket

```python
# Inicia el servidor WebSocket
async def start_websocket_server(port):
    global server_sockets, server_ws_task  # Usa variables globales

    addr = socket.getaddrinfo('0.0.0.0', port)[0][-1]  # Dirección de escucha
    ws_server_socket = socket.socket()
    ws_server_socket.bind(addr)           # Asocia el socket
    ws_server_socket.listen(1)            # Escucha conexiones
    ws_server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  # Reutiliza puerto
    ws_server_socket.setblocking(False)   # No bloqueante

    print(f"Servidor WebSocket corriendo en el puerto {port}")

    server_ws_task = asyncio.create_task(SocketTask())  # Inicia tarea asincrónica

    while True:
        try:
            client_socket, addr = ws_server_socket.accept()  # Acepta conexión
            client_socket.setblocking(False)

            ip_actual = addr[0]  # IP del cliente

            # Verifica si ya existe la IP conectada
            ip_ya_conectada = any(ip_actual == ip for _, (ip, _), _, _, _ in server_sockets)

            if ip_ya_conectada:
                print(f"La IP {ip_actual} ya fue registrada.")
                del_client(addr[0])  # Elimina cliente duplicado
            else:
                print(f'Cliente WebSocket conectado desde: {addr}')
                ws_server_handshake(client_socket)  # Handshake WebSocket
                server_sockets.append((client_socket, addr, time.time(), None, None))  # Registra cliente
        except OSError as e:
            if e.errno == 11:  # No hay conexión disponible aún
                await asyncio.sleep(0.1)
            else:
                print(f"Error aceptando conexión: {str(e)}")
```

Fuente: El Autor

El manejo de usa solicitud Web al puerto 80 está asociado en “endpoints” específicos los cuales permiten enviar diversas respuestas a las solicitudes las mismas están controladas por la definición mostrada en la Fig. 40.

Figura 40
Manejo de solicitudes puerto 80



```python
# Maneja solicitudes HTTP del cliente
async def handle_client(client):
    global setup_code, loop_code, dispo_ips, station

    try:
        request = client.recv(1024)  # Recibe datos
        request_str = request.decode()
        method, resource, _ = request_str.split(' ')[:3]  # Extrae método y recurso

        # Redirección para captivas o pruebas de conexión
        if resource in ["/connecttest.txt", "/generate_204", "/hotspot-detect.html","/redirect"]:
            response = "HTTP/1.1 302 Found\r\nLocation: /index.html\r\n\r\n"
            client.send(response.encode())
            return

        resource = resource if resource != '/' else '/index.html'  # Página por defecto
        print(f"Recurso solicitado: {resource}")

        if method == "OPTIONS":
            send_response(client, '204 No Content', 'text/plain', '')

        # Recibe código de setup y lo ejecuta
        elif resource == "/setup" and method == "POST":
            setup_code = request_str.split("\r\n\r\n")[1]
            if setup_code != '':
                print("Código recibido:", setup_code)
                await execute_mixed_code(setup_code)
            send_response(client, '200 OK', 'text/plain', '\nSetup ejecutado')

        # Recibe código de loop y lo lanza como tarea
        elif resource == "/loop" and method == "POST":
            global loop_task
            loop_code = request_str.split("\r\n\r\n")[1]
            print("Código recibido para el loop:\n", loop_code)

            if loop_task is not None:
                print("Cancelando tarea anterior...")
                loop_task.cancel()
            loop_task = asyncio.create_task(exec_loop_code(loop_code))
            send_response(client, '200 OK', 'text/plain', '\nLoop iniciado')

        # Detiene el loop
        elif resource == "/stop_loop" and method == "POST":
            global loop_task
            if loop_task is not None:
                print("Cancelando loop...")
                loop_task.cancel()
                loop_task = None
            send_response(client, '200 OK', 'text/plain', '\nLoop detenido')

        # Registra IP conectada si no está en la lista
        elif resource == "/stations" and method == "POST":
            if station.isconnected():
                if station.ifconfig()[2] not in dispo_ips:
                    dispo_ips.append(station.ifconfig()[2])
            print(dispo_ips)
            send_response(client, '200 OK', 'text/plain', f'{dispo_ips}')

        # Endpoint simple de prueba
        elif resource == "/register" and method == "POST":
            send_response(client, '200 OK', 'text/plain', 'registrado')

        # Carga archivos estáticos si no coincide otro recurso
        else:
            await load_static_file_in_chunks(resource, client)

    except Exception as e:
        print(f"Error manejando el cliente: {str(e)}")
        send_response(client, '200 OK', 'text/plain', f'\nError: {str(e)}')
    finally:
        client.close()  # Cierra la conexión
```

Fuente: El Autor

Para no saturar el microcontrolador con algún archivo pesado se utilizan chuncks que permiten fragmentar la información y enviarla a través de la interfaz inalámbrica, que de igual manera gracias al servidor de archivos localiza el recurso según su extensión y lo devuelve según su “endpoint” concreto, como muestra la Fig. 41 los códigos de error son considerados y estructurados en la respuesta a la solicitud.

Figura 41
Servidor de archivos

```python
# Función para servir archivos grandes en fragmentos
async def load_static_file_in_chunks(filename, client):
    content_type = 'text/plain'
    if filename.endswith('.html'):
        content_type = 'text/html'
    elif filename.endswith('.js'):
        content_type = 'application/javascript'
    elif filename.endswith('.css'):
        content_type = 'text/css'
    elif filename.endswith('.ico'):
        content_type = 'image/x-icon'

    if file_exists(filename):
        print(f"Archivo encontrado: {filename}")
        send_response(client, '200 OK', content_type, '')
        with open(filename, 'rb') as f:
            while chunk := f.read(1024):
                client.sendall(chunk)
    else:
        print(f"Archivo no encontrado: {filename}")
        send_response(client, '404 NOT FOUND', 'text/plain', 'Recurso no encontrado\n')

# Esta función verifica si el archivo existe
def file_exists(filename):
    try:
        with open(filename, 'r'):
            return True
    except OSError:
        return False
```

Fuente: El Autor

El Servidor web socket ejecuta un proceso mediante una definición que permite realizar handshake con sus clientes y mantener la conexión la Fig. 42 muestra este proceso.

Figura 42
Hanshake servidor Web Socket

```python
# Handshake inicial del WebSocket (lado servidor)
def ws_server_handshake(client_socket):
    request = client_socket.recv(1024)  # Recibe solicitud del cliente
    headers = {}

    # Procesa los encabezados HTTP
    for line in request.split(b'\r\n')[1:]:
        if not line:
            break
        key, value = line.split(b': ', 1)
        headers[key] = value

    # Obtiene la clave del cliente
    webkey = headers[b'Sec-WebSocket-Key']
    accept_key = webkey + b"258EAFA5-E914-47DA-95CA-C5AB0DC85B11"  # UUID fijo
    hashed = hashlib.sha1(accept_key).digest()  # Aplica SHA-1
    webaccept = ubinascii.b2a_base64(hashed).strip()  # Codifica en base64

    # Envía respuesta de protocolo WebSocket
    client_socket.send(b'HTTP/1.1 101 Switching Protocols\r\n'
                       b'Upgrade: websocket\r\n'
                       b'Connection: Upgrade\r\n'
                       b'Sec-WebSocket-Accept: ' + webaccept + b'\r\n\r\n')
```

Fuente: El Autor

La definición de la Fig. 43 muestra cual es proceso del cliente que ejecuta continuamente cuando ha sido registrado en el array de sockets.

Figura 43
Definición para procesamiento de cliente web socket

```python
# Tarea principal para gestionar clientes WebSocket
async def SocketTask():
    global server_sockets
    timeout_limit = 15  # Tiempo máximo de inactividad

    try:
        while True:
            for i, (client_socket, addr, last_active, client_type, sms) in enumerate(server_sockets):
                try:
                    message = await ws_server_receive(client_socket)  # Espera mensaje

                    if message is None:
                        # Si no hay mensaje, verifica inactividad
                        if time.time() - last_active > timeout_limit:
                            print(f"Cliente en {addr} inactivo")
                            del_client(addr[0])
                        continue

                    # Cliente cerró conexión o error al decodificar
                    if message in ("CONNECTION_CLOSED", "DECODE_ERROR"):
                        print(f"Cliente desconectado o error de decodificación desde {addr}")
                        del_client(addr[0])
                    else:
                        # Identifica cliente como frontend
                        if message == "Frontend":
                            server_sockets[i] = (client_socket, addr, time.time(), message, None)
                        # Si es un mensaje válido, lo guarda y responde
                        elif message is not None and message != '':
                            server_sockets[i] = (client_socket, addr, time.time(), None, message)
                            ws_server_send(client_socket, "OK")

                        # Si el mensaje está vacío, solo renueva el tiempo
                        elif message == '':
                            server_sockets[i] = (client_socket, addr, time.time(), None, None)
                            print(f'Mantener: {addr[0]}')
                            ws_server_send(client_socket, "OK")

                except OSError as e:
                    print(f"Error en cliente {addr}: {str(e)}")
                    del_client(addr[0])

            await asyncio.sleep(0.1)  # Breve pausa entre iteraciones

    except OSError as e:
        print(f"Error en SocketTask: {str(e)}")
```

Fuente: El Autor

La definición de la Fig. 44 muestra el proceso correspondiente al modo cliente para poder realizar una solicitud hacia otro nodo instanciando la ip y puerto destino, y tener comunicación en tiempo real esta será utilizada luego mediante Frontend.

Figura 44

Definición para modo cliente WebSocket

```python
# Conecta con un servidor WebSocket y realiza handshake
async def con_websocket(host, port):
    global client_sockets
    try:
        # Verifica si ya hay una conexión con esa IP
        ip_ya_registrada = any(host == ip for _, ip, _ in client_sockets)

        if ip_ya_registrada:
            print(f"La IP {host} ya fue registrada.")
            return True  # Ya está conectado
        else:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Crea socket TCP
            sock.connect((host, port))  # Conecta al servidor

            # Si handshake es exitoso, guarda el socket
            if ws_client_handshake(sock, host):
                client_sockets.append((sock, host, True))
                print("Clientes conectados:", client_sockets)
                return True

        return False  # Falló la conexión o el handshake

    except Exception as e:
        return f"Error al conectar Cliente ws: {e}"
```

Fuente: El Autor

Finalmente, El proceso de control de para ejecutar códigos que llegan desde una solicitud html es considerado gracias al manejo correspondiente de las palabras reservadas “/setup” y “/loop” esto permite simular el comportamiento de IDEs más conocidos como Arduino.

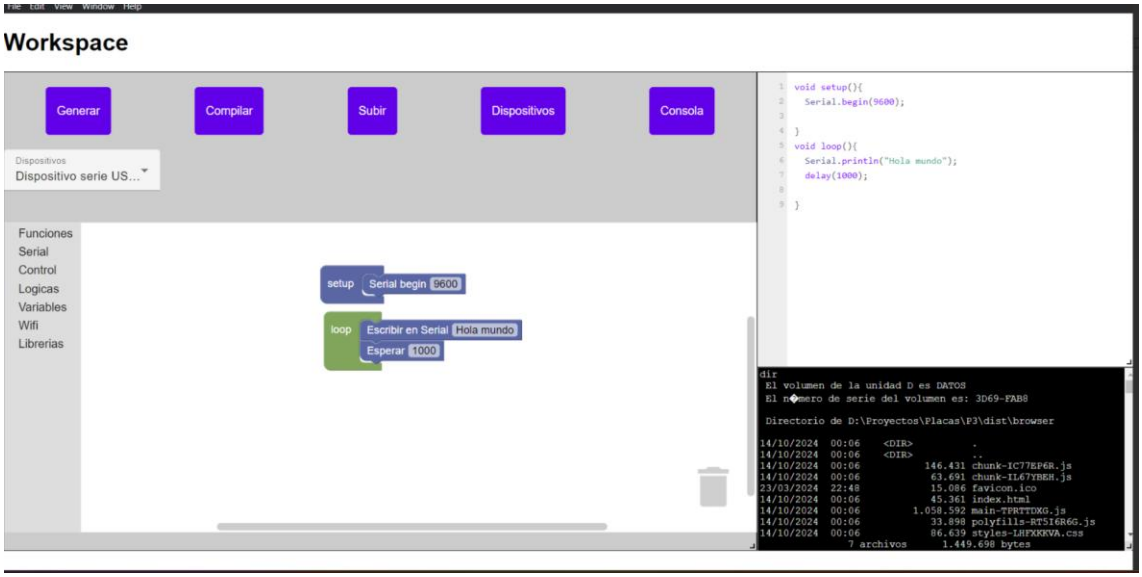
3.6.3. Programación de Frontend (Angular)

Aprovechando la estructura basada en componente se diseña el primer prototipo rápido de la solución en donde los usuarios pueden arrastrar y soltar bloques de código que representan funciones, estructuras de control, variables, entre otros. Los bloques están diseñados para interactuar directamente con las funcionalidades del ESP32, como leer sensores o controlar actuadores.

La Fig. 45 evidencia la primera versión de la aplicación web destina a ser empleada en el microcontrolador.

Figura 45

Primer despliegue Interfaz de Usuario



Fuente: El Autor

Traducción a Micro Python. - Una vez que el usuario completa su programación visual, el Frontend traduce el flujo de bloques a código Micro Python. Este código es enviado al Backend del ESP32 para su ejecución la Fig. 46 muestra el proceso de conversión a código.

Figura 46

Traducción de Frontend código micro Python

```
a = 0
b = machine.Pin(0, machine.Pin.OUT)
def setup():
b.on()
def loop():
for i in range(0,1,1):
 disp("Hola")
time.sleep(1)
```

Fuente: El Autor

Servicio de Socket Cliente. - El Frontend incluye un servicio que actúa como cliente de WebSocket y esta desarrollado con Visual Studio Code, permitiendo la comunicación en tiempo real con el Backend en el ESP32 la Fig. 47 muestra un fragmento del cliente Frontend consumiendo el servicio de WebSocket.

Figura 47

Función de servicio Socket (Frontend)

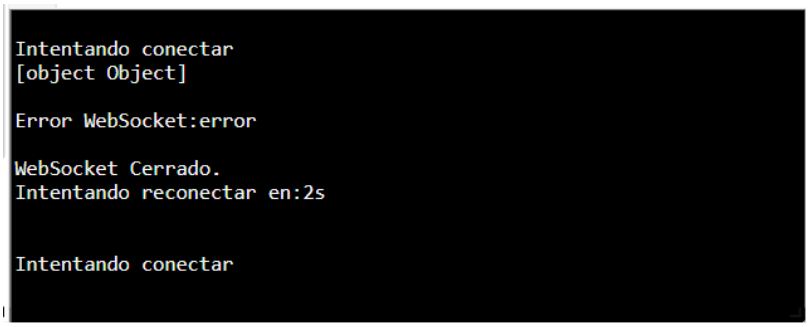
```
IniciarWs() {
  const ip = document.getElementById("IP") as HTMLInputElement;

  const reconnectInterval = 2000;  // Intervalo para reconectar

  this.term.ActualizarTexto("\nIntentando conectar");

  // Crear conexión WebSocket con la IP ingresada
  this.ws = new WebSocket('ws://' + ip.value + ':8080');

  // Cuando se establece la conexión
  this.ws.onopen = () => {
    this.term.ActualizarTexto("\nConectado al Web Socket\n");

    // Enviar mensaje de identificación al servidor
    if (this.ws) {
      this.ws.send('Frontend');
    }

    // Mantener la conexión activa enviando mensajes cada 10 segundos
    this.pingInterval = <number><any>setInterval(() => {
      console.log("Mantener Conexión");
      if (this.ws) {
        this.ws.send('Frontend');
      }
    }, 10000);
  };
```

Fuente: El Autor

Consola de Salidas. - El Frontend incluye una consola de salida que muestra los mensajes generados durante la ejecución del programa en el ESP32, los mensajes son enviados desde el ESP32 al Frontend a través de web sockets y se muestran en tiempo real en la consola, permitiendo al usuario monitorear el estado de su sistema IoT y depurar el código si es necesario, la Fig. 48 se observa la respuesta que se obtiene al conectarse al servicio web socket.

Figura 48

Consola web socket para cliente

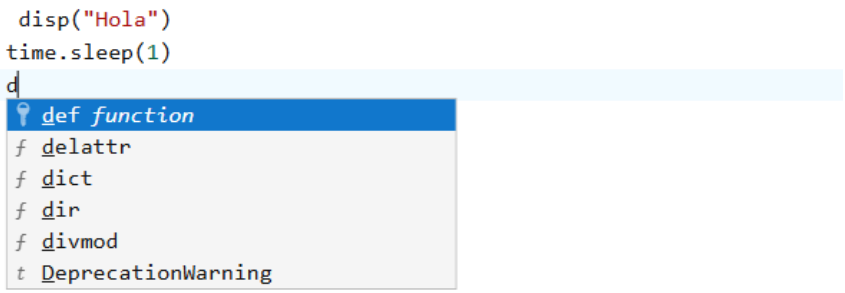


Fuente: El Autor

Editor de Código. - El Frontend también incluye un editor de código Fig. 49 en el cual el usuario puede ver y editar el código generado a partir de los bloques de Blockly. Este editor es interactivo y permite modificar directamente el código Micro Python si el usuario desea personalizarlo más allá de lo que permite el entorno visual.

Figura 49

Editor con autocompletado a texto plano



Fuente: El Autor

Envío de Código al Microcontrolador. - Una vez que el código visual es traducido a Micro Python o el usuario puede realizar modificaciones directamente en el editor de código, el Frontend envía este código al Backend del ESP32 mediante una solicitud HTTP Fig. 50.

Figura 50

Barra de Control Frontend

Fuente: El Autor

Solicitud HTTP: El código es enviado en el cuerpo de la solicitud HTTP al microcontrolador, donde es recibido por el Backend, que lo ejecuta de manera asincrónica utilizando las funciones definidas en setup y loop, la Fig. 51 muestra la función que se emplea para dicho propósito.

Figura 51

Solicitud HTTP para ejecución de código en Backend desde Frontend

```
Enviar() {
  // Obtener el código escrito en el editor
  const code = this.editorView?.state.doc.toString();

  if (code != '' && code != undefined) {
    // Separar el código en setup y loop
    this.separateCode(code);

    console.log('Código setup:', this.setupCode);
    console.log('Código loop:\n', this.loopCode);

    // Obtener la IP ingresada
    const ip = document.getElementById("IP") as HTMLInputElement;

    // Enviar el código de setup al ESP32 mediante HTTP POST
    this.http.post('http://' + ip.value + '/setup', this.setupCode, { responseType: 'text' })
      .subscribe(
        (response) => {
          // Mostrar respuesta del servidor y ejecutar loop
          console.log('ESP-BK:', response);
          this.term.ActualizarTexto(response);
          this.StartLoop(); // Iniciar envío del loop
        },
        (error) => {
          // Mostrar errores si falla la conexión
          console.error('ESP-BK:', error);
          this.term.ActualizarTexto(error.message);
        }
      );
  }
}
```

Fuente: El Autor

3.6.4. Funcionamiento y Control General

Alimentación y Carga. **-** La batería se carga a través de la entrada de 5V proporcionado por el módulo TP4056 esto asegura la carga correcta y a su vez se suministra energía el propio ESP-32. Esto permite que el sistema recargue una batería de 3.7 V y así lograr autonomía y permitir usar el dispositivo pese a que la fuente externa se desconecte.

Entradas. - Los botones permiten al usuario enviar señales digitales al ESP-32, en formato de uno o cero, mientras que los potenciómetros proporcionan entradas analógicas que pueden representar valores variables representados de manera continua.

Salidas. - Las salidas se demuestran mediante leds que se iluminan en función de las señales recibidas por el ESP-32 o programadas mediante el Frontend para el dispositivo principal se consideran dos diodos led Integrados, permitiendo mostrar visualmente las interacciones de la programación.

Control y Comunicación. - La ESP32 gestiona todas las operaciones del sistema, interpretando las entradas, determinando salidas o realizando comunicación hacia otros dispositivos empleando sockets

Interacción con el Usuario. - El usuario interactúa con el sistema a través de peticiones web dirigidas a diferentes endpoints, que pueden incluir solicitudes de datos o el envío de código. Si es la primera vez que se accede al sistema, el servidor proporciona los archivos necesarios para cargar la interfaz desarrollada en Angular.

Para controlar el dispositivo, el usuario utiliza la herramienta Blockly, donde realiza conexiones lógicas mediante bloques visuales. Una vez definida la lógica, el código generado se envía al backend mediante una solicitud web. El servidor procesa la solicitud y responde indicando si la ejecución fue exitosa o si ocurrió un error.

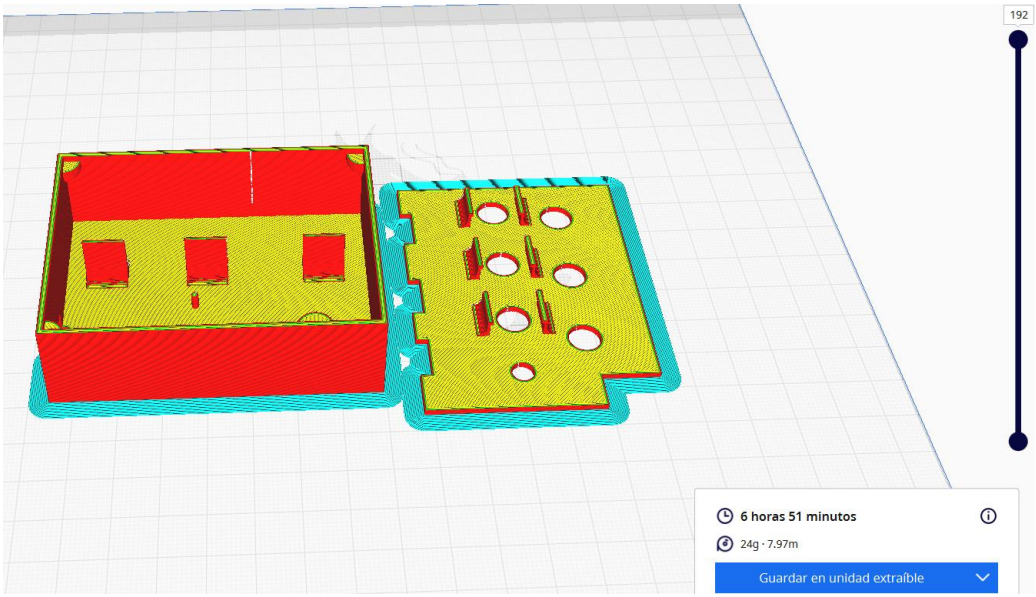
Para la comunicación en tiempo real, el usuario dispone de un botón que permite establecer una conexión mediante web socket, habilitando la visualización de datos en un formato similar a una consola serial física.

3.7. Transición

Para esta sección se emplea impresión 3D con filamento de tipo PLA el mismo que permite implementar los modelos a escala real, de esta manera el prototipo puede ser armado de acuerdo a la consideración de secciones previas, para esto el prototipo se compone de back cover y panel frontal, con la ayuda del software "Cura" se puede observar el proceso de impresión 3D de las partes que conforman la carcasa, el tiempo estimado de impresión así como la cantidad de material se muestran en la Fig. 52.

Figura 52

Previsualización de impresión software Cura



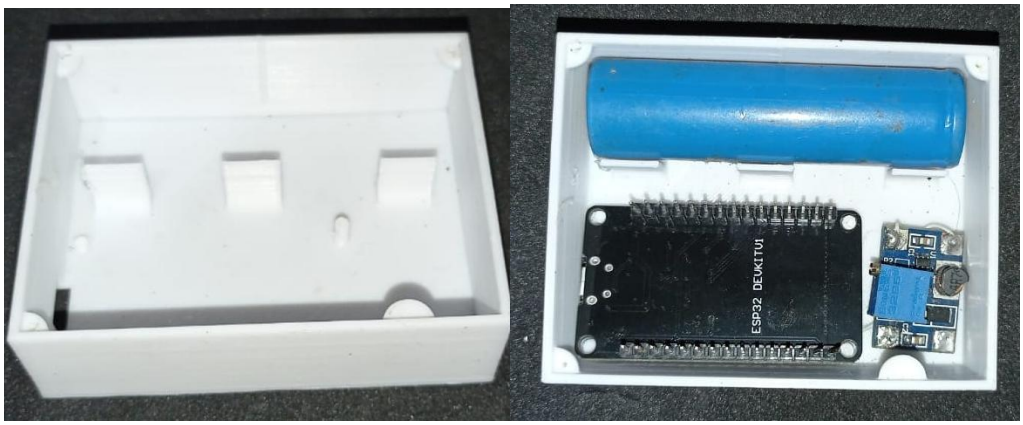
Fuente: El Autor

3.7.1. Back Cover

Una vez obtenido el back over se disponen los elementos a ser interconectados de esta manera se obtiene la gestión de carga la Fig. 53 muestra parte del proceso.

Figura 53

Back cover, microcontrolador, carga y batería



Fuente: El Autor

3.7.2. Panel Frontal

Para el panel fontal es necesario adecuar los switchs, potenciómetros y led realizando la interconexión de estos, de esta manera la Fig. 54 muestra el proceso realizado manualmente.

Figura 54

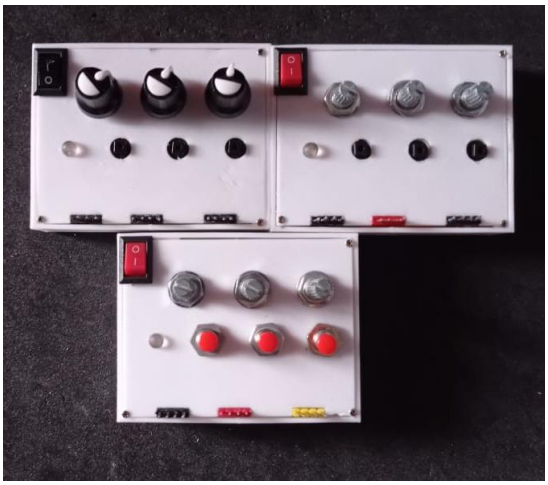
Panel frontal, potenciómetros, switchs, led y salidas de control

Fuente: El Autor

La Fig. 55 muestra los dispositivos finales, los cuales serán usados para realizar las pruebas correspondientes.

Figura 55

Dispositivos finales



Fuente: El Autor

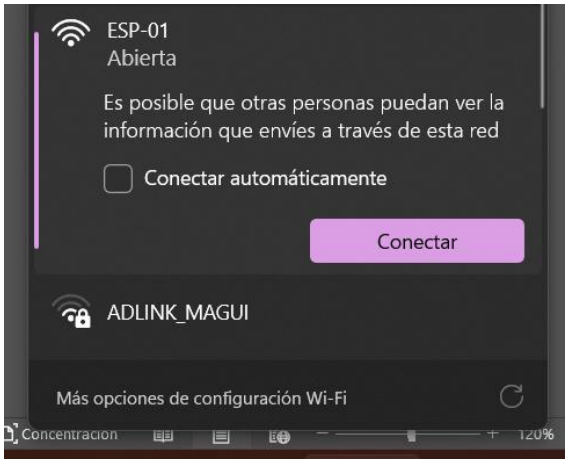
3.8. Producción

La fase de producción considera la primera prueba piloto en ESP-IDE que es interactuar con su consola y correspondiente con el típico “Hola mundo” de cualquier otro lenguaje por tal motivo se propone la siguiente prueba:

Ingreso al Frontend: Una vez la placa esta encendida el usuario tendrá que conectarse a la red existente o creada por el dispositivo por defecto la red es abierta inalámbrica es abierta Fig. 56.

Figura 56

Ingreso al sistema

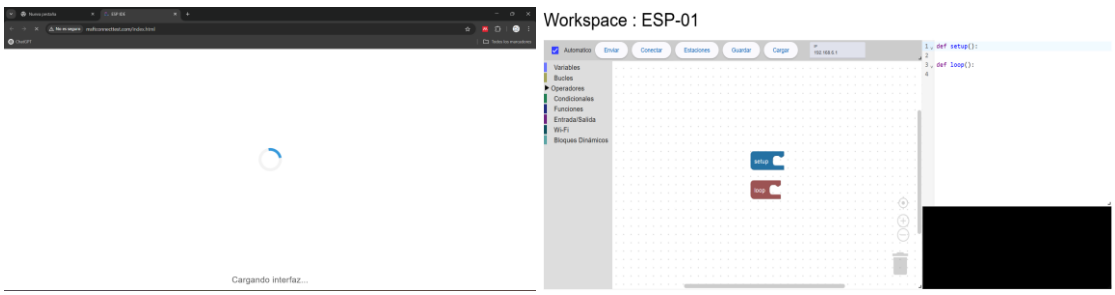


Fuente: El Autor

Una vez conectada el dispositivo Inalámbrico cargara todos los archivos necesarios que conforman parte de la Interfaz como lo muestra la Fig. 57.

Figura 57

Interfaz de Usuario



Fuente: El Autor

Establecer Conexión con socket. - El siguiente paso es establecer una conexión con el socket por tal motivo se presiona el botón conectar y se observa la Salida en la consola tal como muestra la Fig. 58.

Figura 58

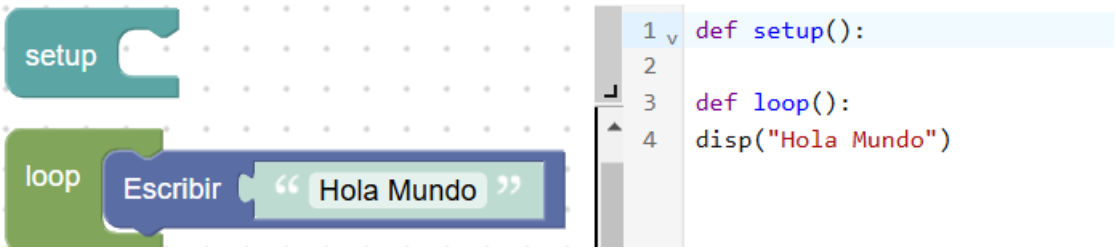
Conexión Web socket

Fuente: El Autor

Una vez conectado al socket se debe realizar la programación por tal motivo se selecciona Funciones dentro del toolbox de blocky y se arrastra el bloque "Escribir" dentro del Bloque **"Loop",** esta acción hará que el mensaje que se escriba y se realice repetidamente, el código dinámico se genera conforme se conectan los bloques como se muestra la Fig. 59.

Figura 59

Primer Programa

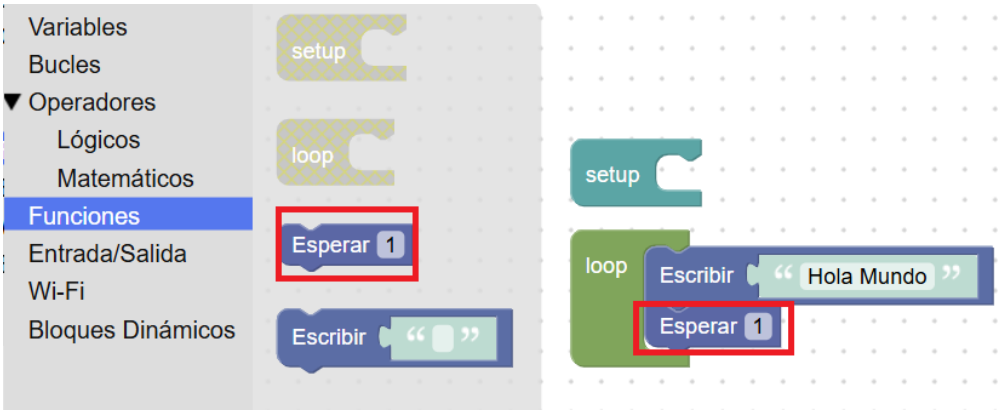


Fuente: El Autor

El siguiente paso es añadir el bloque de espera ya que, sin él, código se ejecuta sin un control sobre el mismo, el efecto de retardo de 1 segundo puede añadirse con el bloque "Esperar**",** como se muestra en la Fig. 60.

Figura 60

Ejecución controlada

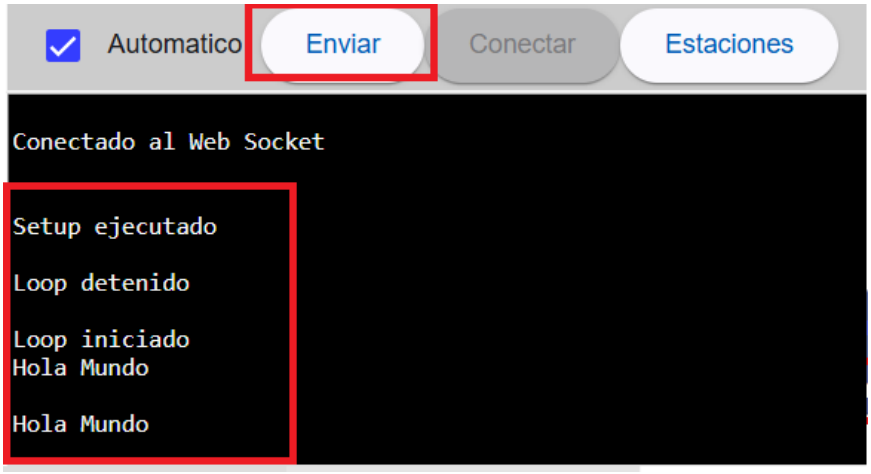


Fuente: El Autor

Completado el paso se realiza la comprobación mediante la consola y el botón Enviar, por tal motivo el código será enviado al microcontrolador y se ejecutará al instante como se muestra en la Fig. 61.

Figura 61

Evaluación de Código



Fuente: El Autor

CAPITULO IV. RESULTADOS

4.1. Pruebas Realizadas

Para garantizar una evaluación estructurada y rigurosa, se tomó como base ISO25010 (2011), la cual define un modelo de calidad para productos de software mediante un conjunto de características y sub características que permiten medir su comportamiento funcional y no funcional. Esta norma proporciona un marco adecuado para valorar aspectos como la usabilidad, la funcionalidad, la fiabilidad, entre otros, facilitando así un análisis del sistema la Fig. 62 resume los aspectos de misma.

Figura 62

Calidad de Producto según ISO 25010

CALIDAD DEL PRODUCTO SOFTWARE								
ADECUACIÓN FUNCIONAL	EFICIENCIA DE DESEMPEÑO	COMPATIBILIDAD	CAPACIDAD DE INTERACCIÓN	FIABILIDAD	SEGURIDAD	MANTENIBILIDAD	FLEXIBILIDAD	PROTECCIÓN
COMPLETITUD FUNCIONAL CORRECCIÓN FUNCIONAL PERTINENCIA FUNCIONAL iso25000.com	COMPORTAMIENTO TEMPORAL UTILIZACIÓN DE RECURSOS CAPACIDAD	COEXISTENCIA INTEROPERABILIDAD	RECONOCIBILIDAD DE ADECUACIÓN APRENDIZABILIDAD OPERABILIDAD PROTECCIÓN FRENTE A ERRORES DE USUARIO INVOLUCRACIÓN DEL USUARIO INCLUSIVIDAD ASISTENCIA AL USUARIO AUTO-DESCRIPTIVIDAD	AUSENCIA DE FALLOS DISPONIBILIDAD TOLERANCIA A FALLOS RECUPERABILIDAD	CONFIDENCIALIDAD INTEGRIDAD NO-REPUDIO RESPONSABILIDAD AUTENTICIDAD RESISTENCIA	MODULARIDAD REUSABILIDAD ANALIZABILIDAD CAPACIDAD DE SER MODIFICADO CAPACIDAD DE SER PROBADO	ADAPTABILIDAD ESCALABILIDAD INSTALABILIDAD REEMPLAZABILIDAD	RESTRICCIÓN OPERATIVA IDENTIFICACIÓN DE RIESGOS PROTECCIÓN ANTE FALLOS ADVERTENCIA DE PELIGRO INTEGRACIÓN SEGURA

Fuente: https://www.iso25000.com/images/figures/iso_25010.png

A partir de este modelo de calidad propuesto por la ISO 25010, se seleccionaron los atributos más relevantes para el contexto del kit educativo IoT, como la adecuación funcional, la usabilidad y la experiencia de usuario basada en la ISO 9241.

Funcionalidad. **-** Pressman (2010) menciona que la funcionalidad se refiere a la capacidad de un sistema o producto para proporcionar funciones que satisfagan las necesidades explícitas e implícitas de los usuarios, esto asegura que el software cumpla con los requisitos especificados y realice las tareas para las cuales fue diseñado.

Usabilidad. **–** El artículo publicado por Baquero & Baquero (2024) mención que es la medida en que un producto o sistema puede ser utilizado por usuarios específicos para

lograr objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso determinado.

Experiencia de Usuario. **–** La Norma ISO9241 (2010) define la experiencia de usuario (UX) como la experiencia de usuario que abarca las percepciones y reacciones de un individuo derivadas del uso o de un producto, sistema o servicio. Incluye aspectos como emociones, creencias, preferencias, percepciones, respuestas físicas y fisiológicas, comportamientos y logros de los usuarios, los cuales se presentan antes, durante y después de la interacción con el producto.

Por esta razón las pruebas de próximas secciones serán empleadas de acuerdo con los conceptos abordados la Tabla 13 muestra las abreviaturas correspondientes.

Tabla 13
Definición de abreviaturas para pruebas

Abreviatura	**Descripción**
Test0x_F	Test de Funcionalidad
Test0x_E	Test de Usabilidad
Test0x_UX	Test de Experiencia de Usuario

El **ANEXO D: Plan de Pruebas** muestra el plan de pruebas presentado en la institución, mientras la Tabla 14 muestra la síntesis para la evaluación.

Tabla 14
Plan de Pruebas y Resultados

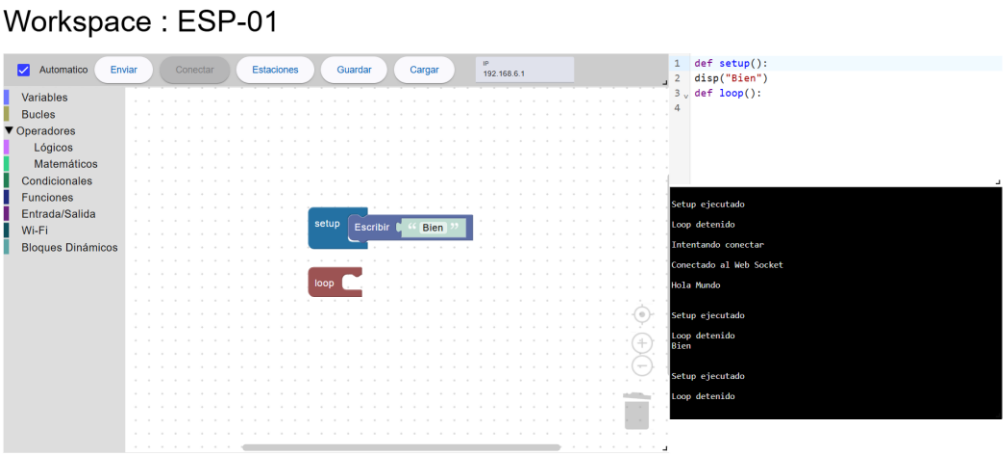
Prueba	**Test**	**Descripción**	**Objetivo**	**Resultado esperado**
Introducción y Configuración Inicial	Test01_F	Preguntas sobre programación y electrónica básica, y ejecución del dispositivo.	Introducción e interacción al prototipo	El dispositivo puede mantener la alimentación y la interfaz inalámbrica activa en la sesión
Familiarización con el prototipo	Test01_U	Manejo de variables digitales mediante botones y Led RGB	Verificar interacción básica	El dispositivo permite el control de variables digitales entradas y salidas.
Introducción Conceptos IoT y programación	Test02_F	Obtención de variables analógicas y visualización por comunicación inalámbrica	Obtener y visualizar variables analógicas	El dispositivo permite gestionar entradas analógicas y representarlas por Websocket
Clase Práctica	Test02_U	Control de datos con pulsadores y potenciómetros y led RGB	Control de variables	El dispositivo permite reconocer y manejar diferentes tipos de variables.

Trabajo en equipo y personalización	Test03_F	Personalización de lógica para generar nuevos programas	Personalización e integración	El usuario puede integrar lógica y personalizar código desde Frontend.
Generación de Proyecto	Test03_U	Proyecto colectivo que integran otros nodos	Aplicación de control	El dispositivo puede comunicarse y controlar otros nodos
Exposición de Proyecto	Test04_UX	Encuestas para obtener retroalimentación del sistema	Presentación y cierre	Se obtiene el porcentaje de aceptación del dispositivo

4.1.1. Pruebas Funcionales del Kit

Introducción y Configuración Inicial. - Para poder realizar esta prueba se dispone del piloto ESP-IDE en el cual se menciona como comprobar la transmisión de datos para esto se comprueba, accediendo al sistema y empleando la definición de "Setup" para poder enviar un dato cambiante, luego se conecta el socket y se revisan con los mensajes de prueba como muestra la Fig. 63 esto permite considerar que la interfaz inalámbrica y alimentación son estables.

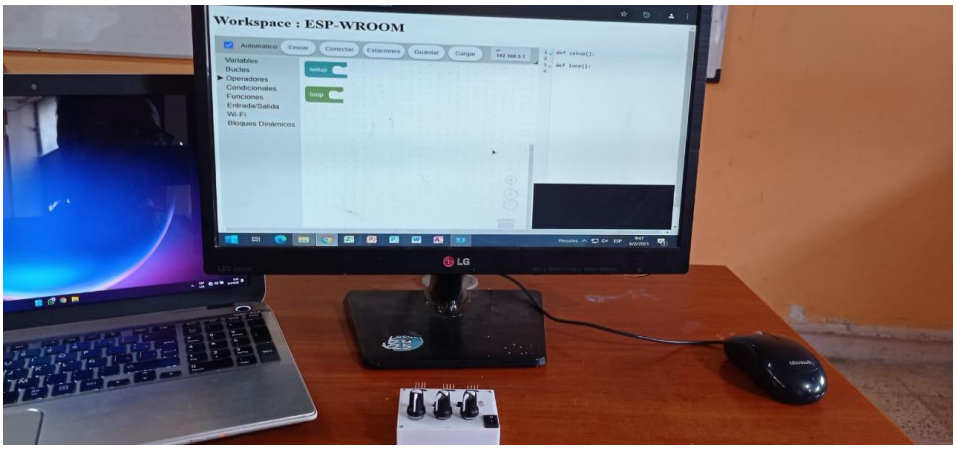
Figura 63
Programa base de introducción



Fuente: El Autor

Luego se comprueba la accesibilidad del dispositivo en las maquinas disponibles en la institución como muestra la Fig. 64.

Figura 64
Prueba base de soporte



Fuente: El Autor

Finalmente, la Tabla 15, muestra las observaciones recibidas por usuarios en la sesión de prueba del dispositivo:

Tabla 15
Prueba de Introducción y Configuración Inicial

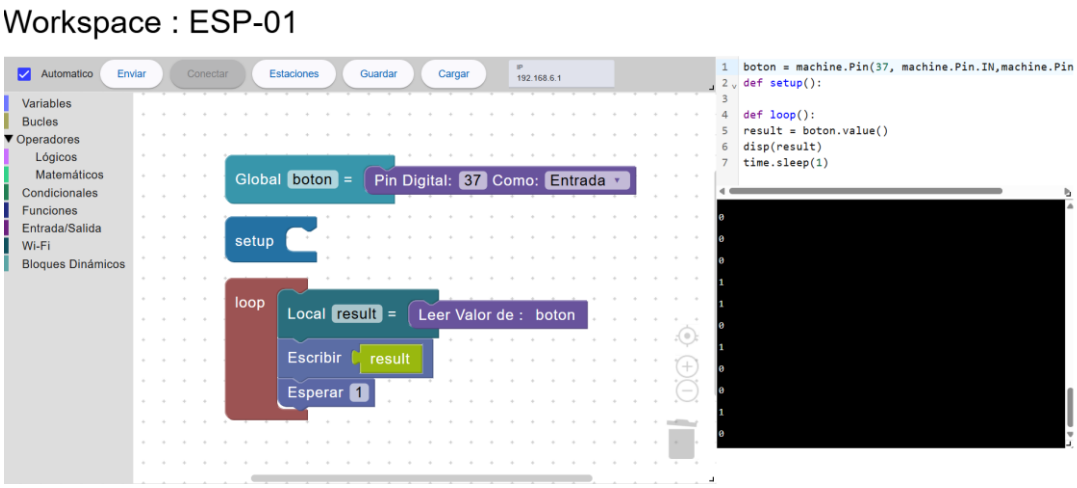
Grupo	Estudiantes	Observaciones	Resultado
Grupo 1	5	El dispositivo se mantuvo correctamente en toda sesión	3
Grupo 2	5	El dispositivo se mantuvo correctamente en toda sesión.	3
Grupo 3	5	El dispositivo tuvo problemas de conexión, solucionado tras desconectar cable Ethernet.	2
Grupo 4	6	Nombre de interfaz inalámbrica duplicada produce interferencia.	1
Grupo 5	6	El dispositivo presenta fallos por alimentación.	1

Nota: *El resultado tiene valores cuantitativos los cuales representan:*
1. El dispositivo no cumple
2. El dispositivo presenta problemas
3. El dispositivo cumple

Introducción Conceptos IoT y programación. **–** Para esta prueba se propone cargar un nuevo programa que me permita obtener el estado de una variable digital “boton” o analógica “potenciómetro” la Fig. 65 muestra el resultado de la interacción de la variable física hacia la consola de salida, mostrando el estado de la variable.

Figura 65

Programa base de manejo de variables



Fuente: El Autor

Teniendo en consideración los comentarios recibidos por los grupos de estudiantes se detalla las observaciones correspondientes en el manejo de variables que sirven de introducción a conceptos básicos de IoT y programación como muestra la Tabla 16**.**

Tabla 16

Introducción Conceptos IoT y programación

Grupo	Estudiantes	Observaciones	Resultado
Grupo 1	5	El dispositivo no presenta problemas	3
Grupo 2	5	El dispositivo no responde a ciertos botones	2
Grupo 3	5	El dispositivo no responde con potenciómetros	1
Grupo 4	6	El dispositivo no presenta problemas	3
Grupo 5	6	El dispositivo no responde a ciertos botones	2

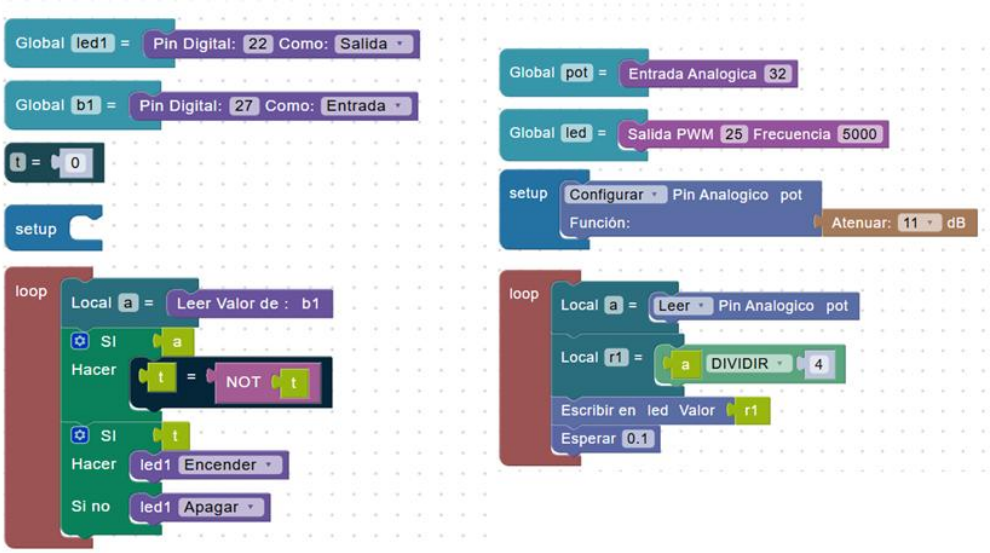
Nota: *El resultado tiene valores cuantitativos los cuales representan:*
1. El dispositivo no cumple
2. El dispositivo presenta problemas
3. El dispositivo cumple

Trabajo en equipo y personalización. **–** En esta prueba el sistema debe promover la integración de los estudiantes con nuevas ideas, de esta manera se propone el uso de botones o potenciómetros para controlar el estado del led RGB y alterar el estado

guardando en memoria, deben aplicar conceptos obtenidos y colaborar en el desarrollo de la lógica, el programa propuesto esta mostrado en la Fig. 66.

Figura 66

Programa base de botones-potenciómetros y leds.



Fuente: El Autor

La (**Tabla 17**) muestra la Tabulación de Observaciones para la prueba gracias a programas anteriores se corrigen errores en el proceso de conexión de botones, potenciómetros y leds para mostrar un correcto funcionamiento.

Tabla 17

Trabajo en equipo y personalización

Grupo	Estudiantes	Observaciones	Resultado
Grupo 1	5	El dispositivo no presenta problemas	3
Grupo 2	5	El dispositivo no presenta problemas	3
Grupo 3	5	El dispositivo no presenta problemas	3
Grupo 4	6	El dispositivo no presenta problemas	3
Grupo 5	6	El dispositivo no presenta problemas	3

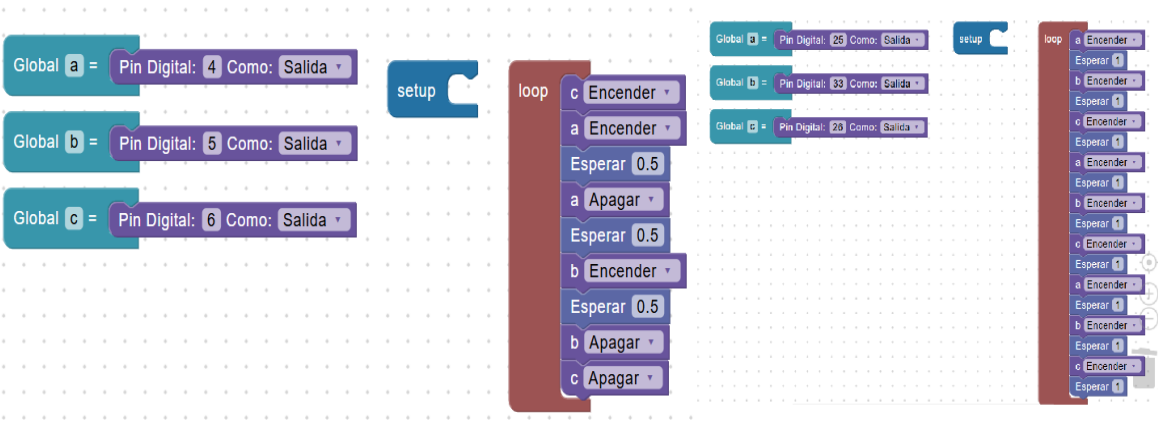
Nota: *El resultado tiene valores cuantitativos los cuales representan:*
1. El dispositivo no cumple
2. El dispositivo presenta problemas
3. El dispositivo cumple

4.1.2. Pruebas de Usabilidad del Kit

Familiarización con el prototipo. – En el desarrollo de esta prueba los estudiantes pueden experimentar el software de control mediante la interfaz de usuario modificando código de manera visual, lo que hace que el prototipo sea interactivo, mediante una clase se obtiene los archivos desarrollados por los estudiantes, mostrando diferente tipo de lógica para un mismo problema.

Las Programas realizados por estudiantes son mostrados en el **ANEXO F** y la Fig. 67 muestra las variaciones en la lógica para el control, se evidencia de igual manera variaciones en el nombre de variables y como son trabajadas dentro de la función "loop" ofertando diversos criterios lógicos en el desarrollo de una solución para un problema planteado.

Figura 67
Comparación de programas entre grupos



Fuente: El Autor

Teniendo en cuenta los comentarios anteriores la Fig. 68 muestra el uso del prototipo sobre un equipo perteneciente a la unidad educativa, es mismo se observa la interacción del usuario y la Interfaz.

Figura 68

Desarrollo de programa sobre equipo físico

Fuente: El Autor

Stewart (2025) menciona que la cuantificación de la información permite transformar las percepciones y comentarios de los usuarios en datos numéricos. Esto facilita el análisis estadístico, permitiendo identificar patrones que orienten la mejora del diseño del sistema evaluado por esta razón la Tabla 17 muestra la cuantificación entre valores de 1 a 3.

Tabla 18

Cuantificación de prueba de Familiarización y Configuración Inicial

Grupo	Estudiantes	Observaciones	Resultado
Grupo 1	5	El grupo completo la práctica correctamente	3
Grupo 2	5	El grupo pudo completar la práctica con algunas modificaciones de código	2
Grupo 3	5	El grupo completo la práctica correctamente	3
Grupo 4	6	Problemas de acceso en la interfaz, solucionado tras actualización de software	2
Grupo 5	6	El grupo completo la práctica correctamente	3

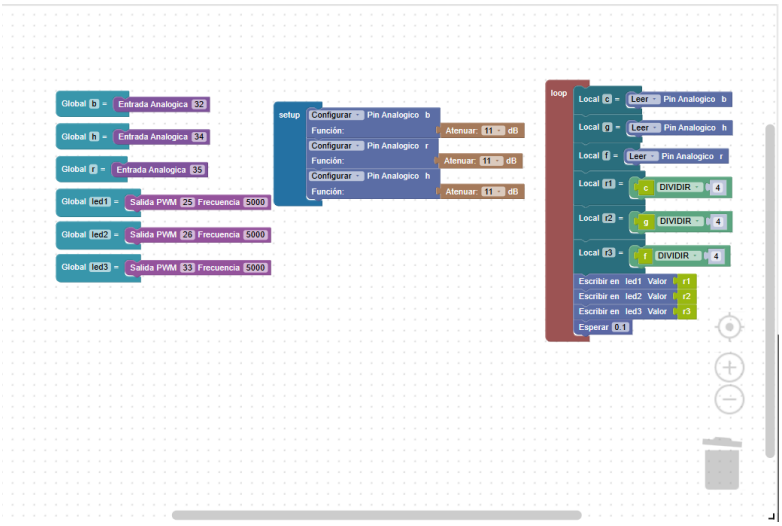
Nota: *El resultado tiene valores cuantitativos los cuales representan:*
1. Desarrollo correcto de práctica
2. Desarrollo Parcial de práctica
3. No pudo desarrollar la práctica

Clase Práctica. **–** En esta prueba los estudiantes deben programar combinaciones básicas que permitan controlar el Led RGB mediante, las señales digitales o analógicas

estas promueven la disposición para reconocer tipos de variables y como estas cambian a lo largo de tiempo, la Fig. 69 muestra el programa propuesto realizando una lectura de variables analógicas potenciómetros y adecuan el valor para las salidas RGB.

Figura 69

Programa propuesto para lectura potenciómetros y ejecución con led RGB.

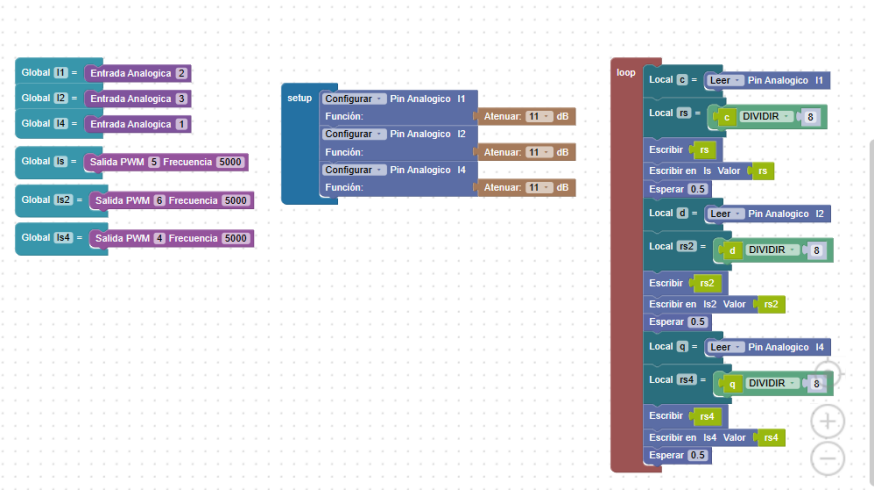


Fuente: El Autor

El **ANEXO G** muestra los diversos programas realizados en clase por estudiantes los mismos tienen definiciones lógicas de variables similares y en la Fig.70 se observa un ejemplo de la lógica empleada en dicho proceso.

Figura 70

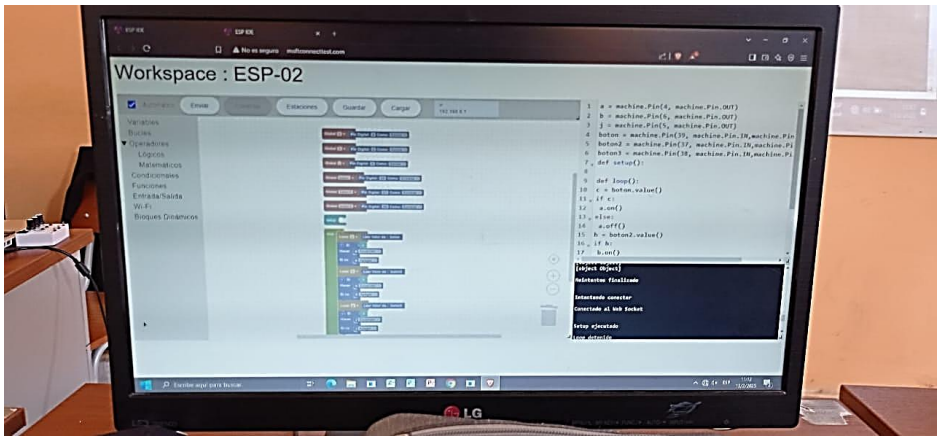
Programa en clase desarrollado por estudiantes.



Fuente: El Autor

Así, mismo la Fig.71 muestra la generación e implementación del código sobre el dispositivo de esta manera se considera que el dispositivo es capaz de reconocer diversos tipos de variables y mostrar sus cambios de estado o valor.

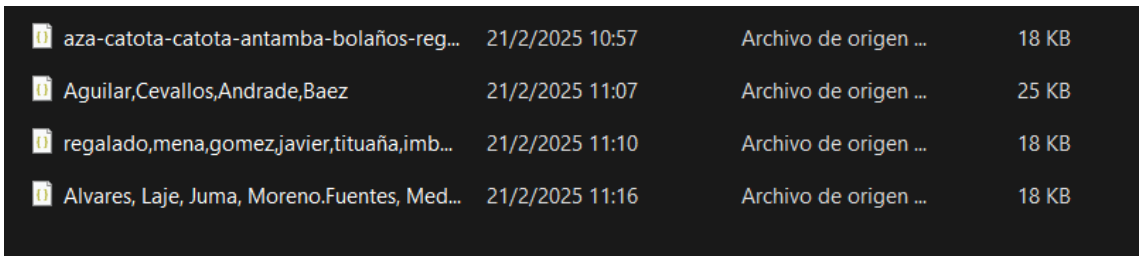
Figura 71
Implementación de Programa en Clase



Fuente: El Autor

Para evaluar el tiempo requerido por los grupos en completar el desarrollo del programa, se utilizó como métrica la hora de entrega de los archivos generados. El punto de partida fue establecido a las 10:30 horas del 21 de febrero de 2025, y se analizaron las marcas de tiempo registradas al momento de guardar cada archivo. La Fig.72 muestra el listado de entregas, lo cual permitió estimar el tiempo real que cada equipo necesitó para finalizar su trabajo.

Figura 72
Fecha de entrega para el trabajo



Fuente: El Autor

Considerando lo anterior la Tabla 19 muestra los tiempos que cada grupo tuvo que utilizar para completar la práctica.

Tabla 19

Estimación de tiempo en completar la practica

Grupo	Estudiantes	Observaciones	Tiempo	Resultado
Grupo 1	5	El grupo completo la práctica correctamente	27 min	1
Grupo 2	5	El grupo completo la práctica correctamente	40 min	1
Grupo 3	5	El grupo completo la práctica correctamente	37 min	1
Grupo 4	6	El grupo tiene problemas con estructuras de control	46 min	1
Grupo 5	6	El Grupo no pudo completar la practica	50 min	0

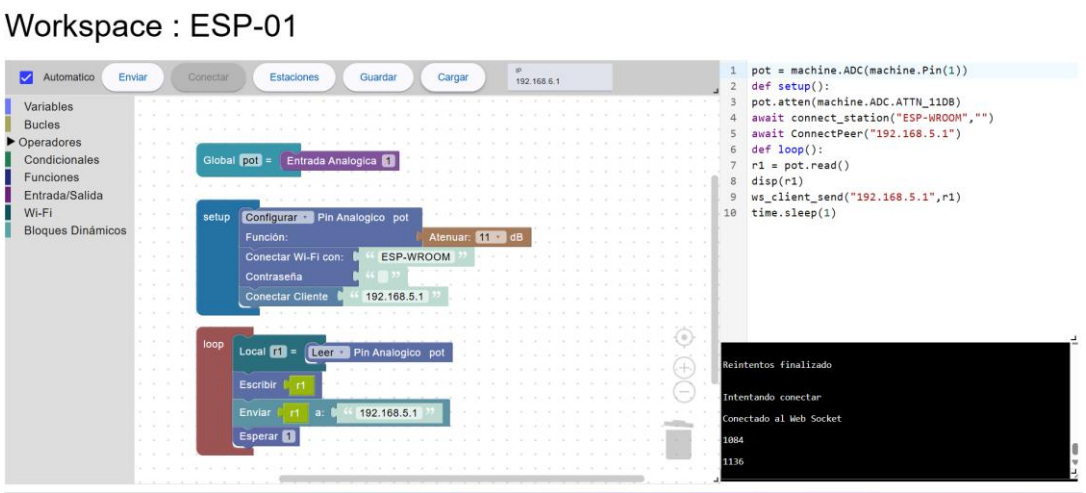
Nota: *El resultado tiene valores cuantitativos los cuales representan:*
1. Desarrollo correcto de práctica
0. Fallo de Desarrollo de práctica

Generación de Proyecto. – Para esta prueba se usa la comunicación Inalámbrica de los nodos de esta manera se busca la colaboración y se genera la interacción entre dos dispositivos los cuales permiten evaluar la estabilidad de la comunicación y la transmisión de datos en tiempo real usando web sockets.

Para este propósito se designa un dispositivo el cual implementa el programa cliente web socket este dispositivo, se conecta a la red inalámbrica de su dispositivo servidor y envía los datos correspondientes como muestra la Fig. 73

Figura 73

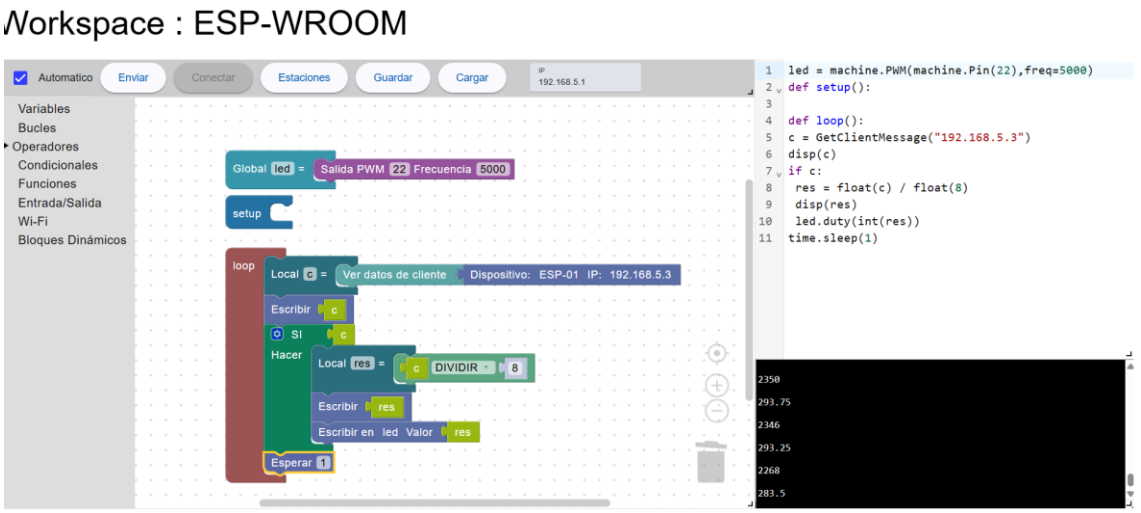
Programa para enviar lectura de potenciómetro (Cliente)



Fuente: El Autor

De igual forma se debe crear la lógica para procesar la información recibida desde el dispositivo receptor para poder ejecutarla en el led RGB destino, la Fig. 74 muestra los valores recibidos por el web socket.

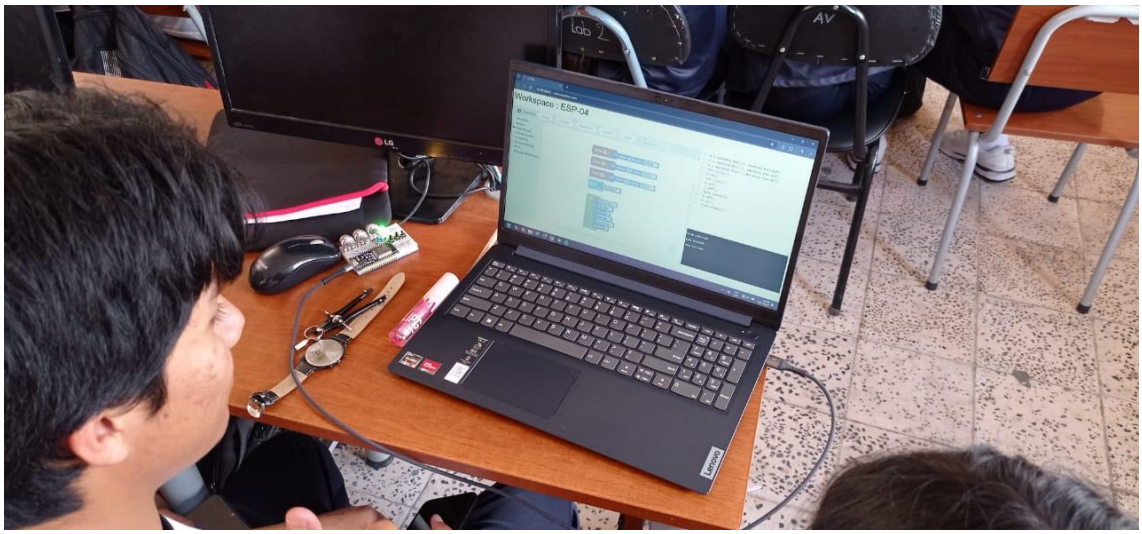
Figura 74
Programa para recepción y ejecución de en led RGB



Fuente: El Autor

Finalmente, el proyecto fue comprobado mediante los dispositivos y los estudiantes pudieron entender el proceso de comunicación de datos entre dispositivos en tiempo real Fig. 75.

Figura 75
Ejecución e implementación de receptor



Fuente: El Autor

Realizando el proceso de cuantificación se pudo observar lo siguiente:

Tabla 20

Cuantificación de comunicación inalámbrica entre nodos

Grupo	Estudiantes	Observaciones	Resultado
Grupo 1	5	Nodo en modo cliente ejecutado exitosamente	1
Grupo 2	5	Nodo en modo servidor ejecutado exitosamente	1
Grupo 3	5	Falta de conceptos direcciones IP y Sockets	0
Grupo 4	6		0
Grupo 5	6	Nodo en modo cliente ejecutado exitosamente	1

Nota: *El resultado tiene valores cuantitativos los cuales representan:*
1. Desarrollo correcto de práctica
0. Fallo de Desarrollo de práctica

4.1.3. Evaluación de la Experiencia de Usuario

Para poder evaluar la experiencia de usuario al final del proceso de pruebas se realizó una encuesta a los estudiantes, de esta manera se comparte la retroalimentación de los resultados, el **ANEXO C**, muestra las respuestas de cinco preguntas realizadas mediante la herramienta Google Forms, obteniendo 29 respuestas por parte de los estudiantes y de esta manera la Tabla 21 muestra la selección de las tres preguntas cuantitativas.

Tabla 21

Preguntas cuantitativas para evaluación de experiencia de usuario

Pregunta	Objetivo	Resultado
1	Facilidad de Uso	El 77.33% de los estudiantes respondieron que el uso de software fue fácil o muy fácil el uso del Software
3	Dificultades y Errores	El 75.00% de los estudiantes seleccionaron que no hubo ninguna dificultad
4	Herramienta de Apoyo	El 96.50% de los estudiantes respondieron que la herramienta ayudo en el proceso de enseñanza en electrónica y programación

4.2. Resultados de las Pruebas

Tras la ejecución de las pruebas, se recopilaron datos y observaciones que permitieron analizar el desempeño del kit en distintos escenarios, la Tabla 22 muestra el puntaje total para cada prueba según los grupos de control en cada dispositivo de esta manera los puntos son de 1 a 3 obteniendo un máximo de 15 para cada prueba.

Tabla 22

Puntaje para pruebas de funcionalidad

Test	Grupo1	Grupo2	Grupo3	Grupo4	Grupo 5	Total
Test01_F	3	3	2	1	1	10
Test02_F	3	2	1	3	2	11
Test03_F	3	3	3	3	3	15

Para las pruebas de usabilidad algunos parámetros deben ser cuantificados como se mencionó previamente por esta razón la Tabla tiene valores para Test01_U de hasta 15 puntos y Test02_U, Test03_U con la valoración de 5 puntos cada una.

Tabla 23

Puntaje para pruebas de usabilidad

Test	Grupo1	Grupo2	Grupo3	Grupo4	Grupo 5	Total
Test01_U	3	2	3	2	3	13
Test02_U	1	1	1	1	0	4
Test03_U	1	1	0	0	1	3

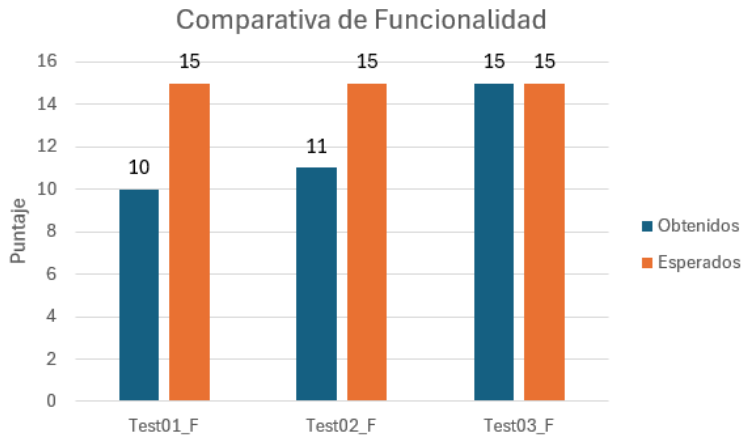
Nota: *El resultado del primer test tiene un puntaje de 15 puntos mientras que los dos restantes 5.*

4.2.1. Indicadores de funcionalidad y usabilidad

Para medir la efectividad del kit, se analizaron varios indicadores clave de eficiencia y usabilidad durante las pruebas, por tal motivo se presenta la tabulación de los datos obtenidos en el periodo de pruebas para un total de 5 Grupos de control estudiantes participantes en el proyecto.

Porcentaje de funcionalidad. – en este análisis se comprueba la evolución del sistema al gestionar los recursos disponibles por esta razón se comprueba mediante La Fig. 76 muestra una comparativa de los puntos esperados en la prueba y los compara con los puntos obtenidos por la misma mostrando la evolución en cada iteración obteniendo un 100% de funcionalidad en la tercera iteración.

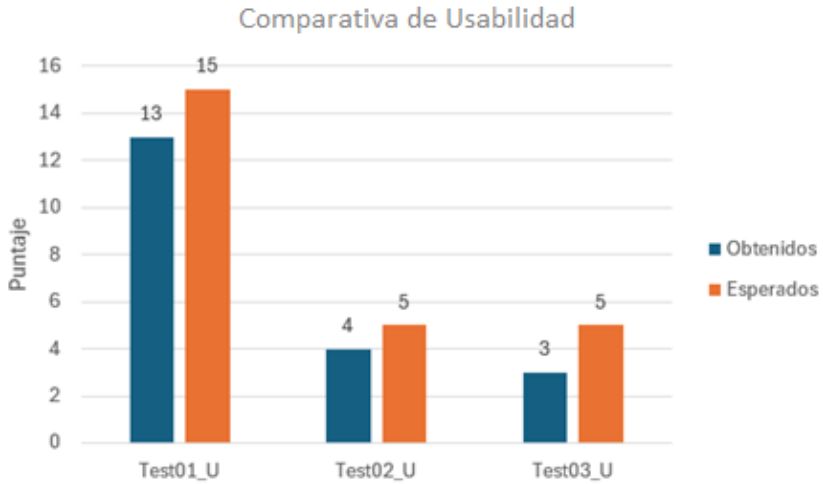
Figura 76
Comparación de Pruebas Funcionales



Fuente: El Autor

Porcentaje de Usabilidad. – para poder obtener un indicador sobre las pruebas se puede considerar sumar todos los puntos obtenidos en las pruebas de usabilidad y compararlos con los puntos totales de la prueba, estos valores son mostrados en la Fig. 77.

Figura 77
Comparación de Pruebas de usabilidad



Fuente: El Autor

$$\text{Promedio de usabilidad} = \frac{17}{20} * 100\ \% = 85\ \%$$

El valor mostrado corresponde al margen de uso y disponibilidad del dispositivo en todas las actividades previamente realizadas.

Tiempo Promedio de resolución. - La prueba de usabilidad Test02_U asociado a **Tabla 19** indaga sobre el tiempo que se empleó para resolver un problema complejo partiendo de una base, lo que permite añadir lógica y funcionalidad; de esta manera se dispuso de 50 min para poder completar el problema, ya que en la prueba 4 grupos de control entregaron la práctica se tiene que el tiempo promedio es:

$$\text{Tiempo Promedio} = \frac{150\ \text{min}}{4} = 37.5\ [\text{min}]$$

Porcentaje de aceptación. – Para obtener la evaluación de experiencia de usuario se consideran promediar los porcentajes obtenidos al final del proceso de pruebas referente a la evaluación de Experiencia de usuario (**Tabla 21**) esta manera:

$$\text{Porcentaje de aceptación} = \frac{0.7733 + 0.75 + 0.965}{3} * 100\ \% = 82.94\ \%$$

Tabla 24

Resumen de indicadores funcionalidad y usabilidad

Indicador	**Valor Obtenido**
Porcentaje de Funcionalidad	100 %
Porcentaje de Usabilidad	85.00%
Porcentaje de Aceptación	82.94%
Tiempo de Resolución Problema	37.5 [min]

4.2.2. Análisis de los Resultados Obtenidos

Luego de la realización de las pruebas funcionales, de usabilidad y de experiencia de usuario, se identificaron diversas tendencias en el uso del kit. Se observó que el sistema presentaba una curva de aprendizaje progresiva, donde los estudiantes inicialmente mostraban dificultades por la adaptación visual, pero lograban un dominio más fluido tras realizar pequeños programas guiados. En términos de funcionalidad, se detectaron pequeños inconvenientes en la conectividad del web socket y en la alimentación, los cuales fueron corregidos mediante ajustes en el código base y mejoras en el circuito de

carga. La percepción general de los usuarios reflejó una alta aceptación del kit, destacando su interactividad y aplicabilidad en entornos educativos.

Los indicadores cuantitativos presentados en la Tabla 24 muestra que:

- El valor de 100 % en funcionalidad confirma que el sistema logró cumplir satisfactoriamente con los requerimientos establecidos, ejecutando correctamente las tareas definidas.

- La usabilidad, con un resultado de 85 %, sugiere que el kit fue bien recibido en términos de facilidad de uso, navegación e interacción, más sin embargo existen aspectos mejorables como la claridad de ciertos elementos visuales o la simplificación de pasos en la interfaz.

- En cuanto a la aceptación del prototipo, se obtuvo un 82.94 %, lo que, valida su aplicabilidad y utilidad en el contexto educativo, especialmente considerando el entusiasmo expresado por los estudiantes al interactuar con el sistema.

Por otro lado, el tiempo promedio de resolución de un problema complejo fue de 37.5 [min] en donde este valor refleja una duración moderada para tareas prácticas, considerando que los usuarios disponían de 50 [min] para finalizar dicha tarea.

No obstante, este indicador también ofrece una oportunidad para optimizar el proceso de enseñanza y adaptación a la herramienta.

4.2.3. Retroalimentación Recibida de los Usuarios

Durante la fase final del proceso, los estudiantes compartieron sus observaciones sobre la herramienta realizando una encuesta en Google Forms, El **ANEXO C**. destacó la importancia de contar con tutoriales más detallados sobre el uso de los bloques de programación y se sugirió incluir más ejemplos prácticos dentro de la plataforma. Asimismo, algunos estudiantes mencionaron que la integración con dispositivos móviles podría ser una mejora valiosa para aumentar la accesibilidad del kit. Esta retroalimentación permitió establecer recomendaciones para futuras iteraciones del prototipo.

La retroalimentación de los usuarios fue en su mayoría positiva. Los estudiantes valoraron la claridad y la facilidad de uso del prototipo, aunque algunos sugirieron mejoras

- **Documentación:** La mayoría de los grupos indicaron que la documentación podría mejorarse, especialmente en lo que respecta a ejemplos prácticos de código y situaciones complejas.

4.3. Limitaciones Identificadas en el Proyecto

Las limitaciones identificadas durante las pruebas del kit IoT no disminuyen su efectividad como herramienta educativa, pero sí señalan áreas importantes de mejora para su evolución. Las propuestas de mejora detalladas en esta sección proporcionan un camino claro para optimizar el prototipo en futuras iteraciones, garantizando una mejor experiencia de aprendizaje para los estudiantes y una mayor escalabilidad y robustez en el sistema.

Tabla 25

Síntesis de limitaciones encontradas para el proyecto

Limitación Identificada	Descripción	Impacto
Curva de Aprendizaje Inicial	La falta de experiencia previa en dispositivos electrónicos y tecnología IoT generó una brecha en el proceso de enseñanza.	El tiempo necesario para que los estudiantes comprendieran el funcionamiento del sistema y su programación se incrementó significativamente.
Problemas de Conectividad (Web Socket)	La saturación de redes Wi-Fi en la frecuencia de 2.4 GHz causó inestabilidad durante las pruebas, afectando la comunicación por Web Socket.	La intermitencia en la conexión generó dificultades en la sincronización de datos y una mala experiencia de usuario, lo que comprometió la eficiencia del sistema.
Limitaciones en Tecnología 2.4 GHz	La ESP32 no es capaz de manejar protocolos de comunicación más avanzados, lo que restringe su rendimiento en redes de mayor capacidad.	Esta limitación afectó la escalabilidad del sistema, restringiendo las opciones para integrar tecnologías de comunicación más modernas.

Limitaciones en la Memoria del Microcontrolador	A pesar de contar con una capacidad de memoria superior a otros microcontroladores, la ESP32 sigue teniendo limitaciones que afectan el manejo de procesos complejos o la ejecución de múltiples tareas simultáneas.	Esto restringió la ejecución de aplicaciones más complejas, afectando la capacidad del sistema para gestionar múltiples conexiones o procesos de manera simultánea.
Limitaciones de las Librerías de Micro Python	Las librerías de Micro Python disponibles para la ESP32 no están completamente optimizadas y presentan limitaciones en cuanto a su funcionalidad o soporte.	Algunas funciones o módulos clave requerían programación adicional y no estaban bien documentados, lo que dificultó su integración en el proyecto y aumentó el tiempo de desarrollo.

4.4. Comparación con Otros Estudios o Herramientas Similares

El diseño del kit de aprendizaje IoT propuesto en esta investigación se orienta a fortalecer el pensamiento lógico en estudiantes. En esta sección se presenta una comparación con otras herramientas educativas tecnológicas similares, así como estudios previos que abordan objetivos pedagógicos afines.

Kit Micro: bit. - BBC micro:bit es una plataforma ampliamente usada en contextos educativos, que permite la programación mediante bloques (MakeCode) y también en Python. Su enfoque está en promover el pensamiento computacional y la creatividad en estudiantes de educación básica y secundaria. No obstante, su capacidad de expansión hacia entornos de Internet de las Cosas (IoT) requiere módulos adicionales, lo cual puede dificultar su integración con sensores específicos sin conocimientos técnicos previos Micro:bit (2025)

Kit Arduino Uno Educativo. - Arduino Starter Kit ha sido empleado durante años en instituciones técnicas para la enseñanza de programación y electrónica básica. Aunque ofrece una experiencia práctica directa, la curva de aprendizaje del lenguaje C/C++ puede representar una barrera para principiantes. Además, la integración con interfaces visuales es limitada sin el uso de extensiones o herramientas de terceros Banzi et al. (2015)

Tinkercad Circuits. -Tinkercad Circuits es una herramienta virtual que permite simular circuitos electrónicos con microcontroladores como Arduino. Aunque es útil en

entornos sin acceso a hardware físico, la ausencia de interacción sensorial o física con componentes limita el desarrollo del pensamiento lógico vinculado a la manipulación real, la resolución de errores físicos o el análisis en contextos IoT Autodesk (2023)

LEGO Education SPIKE Prime. – Este kit ofrece una experiencia educativa basada en la construcción con bloques y programación visual. Su reciente integración con módulos IoT permite a los estudiantes abordar problemas reales mediante la conectividad y la automatización. A pesar de su atractivo diseño y valor educativo, su elevado costo y dependencia de componentes propietarios limitan su adopción en escuelas públicas con bajo presupuesto LEGO (2025).

Tabla 26

Resumen de Herramientas Similares

Criterio	Kit IoT Propuesto (ESP-IDE)	Arduino Educativo	micro:bit	Tinkercad Circuits	LEGO SPIKE Prime IoT
Programación Visual	(Blockly + MicroPython)	Opcional (S4A, Ardublock)	Sí (MakeCode)	Sí (bloques)	Sí (Scratch-like)
Interacción Física	Alta (sensores IoT reales)	Alta	Alta	Nula	Alta
Conectividad IoT	Sí (WiFi nativo con ESP32)	Sí (requiere módulos extra)	Limitada (con accesorios)	No	Sí (requiere configuración)
Nivel Educativo	Secundaria / Técnica	Secundaria / Universitaria	Primaria / Secundaria	Secundaria	Primaria / Secundaria
Costo aproximado	Bajo / Medio	Bajo	Medio	Gratuito (simulación)	Alto
Accesibilidad	Alta (hardware abierto)	Alta (abierto)	Media (requiere importación)	Muy alta (solo internet)	Media (requiere licencias)
Enfoque en Pensamiento Lógico	Alto	Alto	Alto	Medio	Alto

El kit de aprendizaje IoT desarrollado en esta investigación ofrece una solución balanceada entre funcionalidad, accesibilidad económica y enfoque pedagógico. A diferencia de herramientas como Tinkercad, que son limitadas a la simulación, o micro:bit, que requiere módulos adicionales para conectividad avanzada, el uso del microcontrolador ESP32 permite una experiencia completa de programación.

Y gracias a la combinación de programación visual con traducción automática a código estructurado en Micro Python permite que los estudiantes obtengan una base para

la programación textual, fortaleciendo así el pensamiento lógico y la comprensión algorítmica.

4.5. Proyecciones Futuras

Considerando las limitaciones presentadas en la sección anterior se plantean posibles mejoras y ampliaciones del kit de aprendizaje IoT. La Tabla 27 explora ajustes en el diseño, optimización de funcionalidades y nuevas aplicaciones que podrían incorporarse en futuras versiones, las cuales por medio de actualizaciones de hardware y software puedan albergar el mismo sistema de comunicación presentado en el presente trabajo.

Tabla 27
Tabla de futuras Aplicaciones o Extensiones

Proyección Futura	Descripción	Impacto Esperado
Ajustes en el Backend	Refactorización y optimización del backend para soportar múltiples protocolos de comunicación modernos y mejorar su rendimiento general.	Mayor flexibilidad para integrar dispositivos diversos, mejor rendimiento en la gestión de datos y mayor capacidad de respuesta en tiempo real.
Migración a Placa Más Potente	Consideración de migrar a una placa más potente, como una Raspberry Pi o ESP32 con más recursos para mejorar el rendimiento.	Mejora en la capacidad de procesamiento y la gestión de múltiples tareas y conexiones simultáneas, permitiendo aplicaciones más complejas.
Integración con Nuevas Tecnologías IoT	Incorporación de tecnologías como LoRaWAN, NB-IoT o 5G para mejorar la conectividad en entornos de largo alcance.	Expansión del rango de cobertura y mayor robustez en la comunicación, especialmente en entornos con alta interferencia o distancias largas.
Soporte de Seguridad Inalámbrica Mejorado	Implementación de protocolos de seguridad avanzados como TLS y OAuth para garantizar la integridad de los datos y la seguridad en la comunicación.	Aumento de la confiabilidad y la protección de la información en redes inalámbricas, garantizando un entorno seguro para el intercambio de datos.
Optimización del Frontend	Mejora de la interfaz de usuario (UI) para mejorar la experiencia en dispositivos con menos capacidad de procesamiento.	Mejora de la accesibilidad, rapidez de respuesta y adaptabilidad a diferentes dispositivos, optimizando la experiencia del usuario final.
Actualizaciones por OTA	Implementación de un sistema de actualización por Over-the-Air (OTA) para el firmware y el software del dispositivo, permitiendo actualizar el sistema sin necesidad de conexión física.	Reducción de los tiempos de inactividad y facilidad para mantener el sistema actualizado sin requerir intervención manual, mejorando la gestión de dispositivos a gran escala.

4.5.1. Nuevas Aplicaciones o Extensiones del Sistema

El sistema podría extenderse a nuevas aplicaciones, como:

- Conexión con Dispositivos Móviles: Implementar aplicaciones móviles para monitorear y controlar el prototipo de manera remota.
- Expansión a Otros Sensores IoT: Incluir más tipos de sensores para ampliar las capacidades del kit, como sensores de gases o sensores de calidad del aire.
- Integración con Plataformas de Cloud Computing: Permitir la integración con plataformas como Google Cloud o AWS para almacenar y analizar datos de manera remota.

Conclusiones y recomendaciones

Conclusiones

A partir del desarrollo y la implementación del "Kit de Aprendizaje IoT", se ha logrado cumplir de manera efectiva con los objetivos establecidos en esta investigación, lo que ha permitido aportar significativamente en el proceso de enseñanza de programación a estudiantes y por tanto se detallan las principales conclusiones:

El análisis de las necesidades y habilidades de aprendizaje de los estudiantes permitió identificar que, a pesar de contar con conocimientos básicos en programación, se requerían herramientas que facilitaran la conexión entre teoría y práctica, ya que la mayoría de los estudiantes desconocían los microcontroladores. Los resultados de las encuestas y las observaciones del docente a cargo confirmaron que el kit promovió una mayor comprensión conceptual y motivación, al permitir aplicar los conocimientos en un entorno tangible.

El enfoque de prototipado evolutivo permitió realizar ajustes y correcciones a lo largo del proceso, lo que condujo a la obtención de un nivel de estabilidad del 100% en el funcionamiento del kit. Esta estabilidad aseguró un desempeño óptimo durante las prácticas realizadas, garantizando que el kit cumpliera de manera efectiva su propósito educativo al integrar hardware y software de manera fluida y sin errores significativos.

El kit combinó programación visual basada en Blockly con componentes electrónicos físicos como botones, potenciómetros y LEDs, lo que facilitó la comprensión de los conceptos de electrónica y lógica por parte de los estudiantes. Los resultados reflejaron que el 77.33% consideró el software como fácil o muy fácil de utilizar, mientras que el 75% no reportó dificultades significativas, evidenciando una curva de aprendizaje favorable y una interfaz amigable.

Las pruebas piloto permitieron validar la funcionalidad, usabilidad y aceptación del prototipo. Se alcanzó un 85% de usabilidad y un 82.94% de aceptación, indicadores que demuestran la efectividad del kit como recurso pedagógico. Además, se registró un tiempo promedio de 38 minutos para la resolución de problemas, lo cual se considera adecuado para el nivel educativo de los participantes.

Recomendaciones

A continuación, se proponen algunas recomendaciones que pueden mejorar la efectividad del "Kit de Aprendizaje IoT" y ampliar su alcance en futuros desarrollos:

Se recomienda proporcionar guías más detalladas y ejemplos prácticos para los estudiantes y docentes. Incluir tutoriales interactivos y documentación visual podría mejorar aún más la experiencia de aprendizaje y facilitar la integración del kit en diferentes contextos educativos.

Si bien el kit fue efectivo en la enseñanza de fundamentos de programación y IoT, se recomienda practicar con más módulos y sensores para alcanzar una mayor gama de aplicaciones. Esto permitiría a los estudiantes desarrollar proyectos más complejos y aplicados a problemas reales, ampliando su comprensión del IoT.

Aunque los estudiantes respondieron bien al kit, la capacitación es necesaria ya que de esta manera se adquiere conocimiento de cómo integrar mejor las herramientas del kit, esto también promueve al proceso de investigación y desarrollo de más funciones integrables a las funciones existentes presentadas.

Se sugiere implementar un sistema de retroalimentación constante, donde los estudiantes y docentes puedan ofrecer sus opiniones sobre el funcionamiento del kit. Esto permitiría identificar áreas de mejora y hacer ajustes periódicos, asegurando que el kit se mantenga actualizado y útil para los objetivos educativos.

Dado que muchos estudiantes tienen menos experiencia con lenguajes de programación estructurados, sería beneficioso integrar plataformas de programación visual, como Blockly o Scratch, que permitan a los estudiantes experimentar con conceptos de programación sin la barrera del código complejo.

REFERENCIAS BIBLIOGRÁFICAS

Aníbal, P. B. J. (2007). *Módulo para verificar el funcionamiento de los programas grabados en el PIC16F84A*. http://bibdigital.epn.edu.ec/handle/15000/1307

Antonelli, A. F., & Gayoso, C. A. (2021). FlowHDL, lenguaje de programación visual para el diseño digital de lógica programable. *Elektron*, *5*(1). https://doi.org/10.37537/rev.elektron.5.1.113.2021

Autodesk. (2023). *Circuitos - Tinkercad*. https://www.tinkercad.com/circuits

Banzi, M., Shiloh, M., & Jepson, B. (2015). *Getting Started with Arduino*. http://www.safaribook-sonline.com

Baquero, L., & Baquero, L. (2024). *¿Está desapareciendo el concepto de Usabilidad? - AICS® - Asociación Internacional de Calidad de Software*. https://aicsvirtual.org/esta-desapareciendo-el-concepto-de-usabilidad/

Castro, M. (2023). *Lógica de programación a partir del pensamiento lógico con software DFD, PSEINT y LPP* [Universidad Santo Tomás]. https://repository.usta.edu.co/handle/11634/51328

Dale, N., & Weems, C. (2007). *Programación y resolución de problemas con C++* (Pablo E & Vázquez Roig, Eds.; 4a ed.). McGRAW-HILL.

David, K., Sacta, C., Calle, I. E., Msc, O., & Ortiz, I. M. (2011). *Desarrollo de un lenguaje de programación gráfico para microcontroladores*. http://dspace.ups.edu.ec/handle/123456789/1430

Davila, D., Barbe, C., Penaherrera, K., Espinel, C., & Meza, M. C. (2021). The weaknesses in the didactic material of the Ecuadorian educational system, allow the integration of design, innovation and creativity. *Revista Minerva: Multidisciplinaria de Investigación Científica, ISSN-e 2697-3650, Vol. 2, Nº. 6, 2021 (Ejemplar dedicado a: Minerva Journal of Scientific Research), págs. 58-69*, *2*(6), 58–69. https://doi.org/10.47460/minerva.v2i6.43

Domínguez, A. (2016). *Diseño e implementación de una arquitectura IoT basada en tecnologías Open Source.* Universidad de Sevilla.

Dominguez, D., & Pucha, J. (2022). Explorando el Uso de la Tecnología Educativa en la Educación Básica. *PODIUM*, *41*(41), 91–104. https://doi.org/10.31095/podium.2022.41.6

Gallardo, A. R., Herrera Morales, J. R., Sandoval Carrillo, S., Andrade Aréchiga, M., & Ramos Michel, E. M. (2023). *Internet de las cosas*. Universidad de Colima. https://doi.org/10.53897/LI.2023.0001.UCOL

Galván Cardoso, A. P., & Siado Ramos, E. (2021). Educación Tradicional: Un modelo de enseñanza centrado en el estudiante. *CIENCIAMATRIA, ISSN-e 2610-802X, ISSN 2542-3029, Vol. 7, Nº. 12, 2021, págs. 962-975*, *7*(12), 962–975. https://doi.org/10.35381/cm.v7i12.457

García, M. (2022). La innovación educativa como elemento transformador para la enseñanza en la unidad educativa "Augusto Solórzano Hoyos. *Revista EDUCARE - UPEL-IPB - Segunda Nueva Etapa 2.0*, *26*(2), 310–330. https://doi.org/10.46498/reduipb.v26i2.1775

González García, A. J., López, J., Xavi, V., & Guillen, V. (2017). *IoT: Dispositivos, tecnologías de transporte y aplicaciones*. https://openaccess.uoc.edu/handle/10609/64286

IBM. (2024). *¿Qué es la Topología de red? | IBM*. https://www.ibm.com/mx-es/topics/network-topology

ISO9241. (2010). *ISO 9241-210:2010 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. https://www.iso.org/standard/77520.html

ISO25010. (2011). *ISO 25010*. https://iso25000.com/en/iso-25000-standards/iso-25010

Lara Bermúdez, M. J. (2018). *Lenguaje visual de programación por bloques para la construcción de ciudades inteligentes*. https://riuma.uma.es/xmlui/handle/10630/15409

LEGO. (2025). *LEGO Education SPIKE*. https://spike.legoeducation.com/

Mamani Vilca, E., Cabrera, M., & Ramos, O. (2019, febrero). *Propuesta de un enfoque para el desarrollo de software educativo intercultural*.

Micro:bit. (2025, mayo 3). *¿Qué es el micro:bit? | micro:bit*. https://microbit.org/es-es/get-started/what-is-the-microbit/

Ministerio de Educación. (2017). *SUBSECRETARÍA DE FUNDAMENTOS EDUCATIVOS*. 1–31. https://educacion.gob.ec/wp-content/uploads/downloads/2020/04/EGC_Informatica.pdf

Ministerio de Educación. (2021). Guía de apoyo para los docentes en la implementación de metodología STEM-STEAM. *Ministerio de Eduación*, 1–34. www.educacion.gob.ec

Naranjo, J., Lilian, M., Peña, P., & Luis, A. (2016). *El pensamiento lógico-abstracto como sustento para potenciar los procesos cognitivos en la educación*. Sophia, Colección de Filosofía de la Educación. https://www.redalyc.org/articulo.oa?id=441849209001

Orzuza, E. L. (2022). *Recopilación y análisis de los protocolos de aplicación utilizados en IoT*. http://ria.utn.edu.ar/xmlui/handle/20.500.12272/6787

Picie-Alcaraz, I., Olivares-Zepahua, B. A., López-Martínez, I., Romero-Torres, C., Reyes-Hernández, L. Á., Picie-Alcaraz, I., Olivares-Zepahua, B. A., López-Martínez, I., Romero-Torres, C., & Reyes-Hernández, L. Á. (2021). Herramienta para la Enseñanza de la Programación usando Elementos Gráficos. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, *2021*(41), 50–62. https://doi.org/10.17013/RISTI.41.50-62

Pressman, R. S. (2010a). *INGENIERIA DE SOFTWARE*.

Pressman, R. S. (2010b). *Ingeniería del software U N E N F O Q U E P R Á C T I C O SÉPTIMA EDICIÓN*.

Quilo, M., & Javier, A. (2023). *Estrategias motivacionales para la enseñanza aprendizaje de circuitos eléctricos en los estudiantes de segundo año de bachillerato de la Unidad Educativa Priorato año lectivo 2021-2022*. http://repositorio.utn.edu.ec/handle/123456789/13743

Ramírez, J. (2020). PROTOTIPO ROBÓTICO APLICANDO METODOLOGÍA STEAM Y ROBÓTICA EDUCATIVA. JOSÉ JULIÁN PEÑA RAMÍREZ UNIVERSIDAD COOPERATIVA DE COLOMBIA FACULTAD DE

INGENIERIAS PROGRAMA INGENIERÍA DE SISTEMAS NEIVA 2020. *NEIVA*. https://repository.ucc.edu.co/server/api/core/bitstreams/ea28afb4-a186-4d10-8a29-6d8905fe74dd/content

Raposo, M., García, O., & Martínez, M.-E. (2022). La robótica educativa desde las áreas STEAM en educación infantil: Una revisión sistemática de la literatura (2005-2021). *Prisma Social: revista de investigación social, ISSN-e 1989-3469, Nº. 38, 2022 (Ejemplar dedicado a: Critical Thinking, Creativity and Computational Thinking in the Digital Society), págs. 94-113*, *38*, 94–113. https://dialnet.unirioja.es/servlet/articulo?codigo=8532275&info=resumen&idioma=ENG

Senescyt. (2018, octubre 30). *STEM Ecuador incentiva el estudio de las ciencias en la niñez – Senescyt – Secretaría de Educación Superior, Ciencia, Tecnología e Innovación*. Boletin de Prensa. https://www.educacionsuperior.gob.ec/stem-ecuador-incentiva-el-estudio-de-las-ciencias-en-la-ninez/

Solano, A. (2020). *Prototipo de juguete electrónico didáctico, como elemento de apoyo para la enseñanza del idioma kichwa en niños de 5 años en la unidad educativa Benito Juárez comunidad de pucará de San Roque cantón Antonio Ante*. http://repositorio.utn.edu.ec/handle/123456789/10102

Stewart, L. (2025). *Cuantificación de datos cualitativos: Guía paso a paso*. https://atlasti.com/es/research-hub/cuantificacion-de-datos-cualitativos

Tonche Garcia, R. J. (2010). *Diseño y desarrollo de un compilador visual para la enseñanza de la robótica básica.*

UIT-T. (2012). *Términos y definiciones para la Internet de las cosas*.

Vásconez Acuña, D. F. (2014). *Red Inalámbrica tipo malla (WNM) estándar 802.11 de transmisión y la optimización de cobertura en los Colegios de la Provincia de Tungurahua*. https://repositorio.uta.edu.ec:8443/jspui/handle/123456789/6989

ANEXOS

ANEXO A: Formato de Entrevista para Docente

Entrevista

Tema: Diseño de un kit de aprendizaje IoT físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de programación.

Objetivo: Esta encuesta tiene como objetivo explorar las experiencia y perspectiva del docente en la enseñanza de programación. Se busca conocer los lenguajes de programación que se imparten en clases, las herramientas de desarrollo utilizadas, así como la percepción sobre el uso de plataformas de programación visual incluyendo la integración de conocimientos electrónicos.

La información recopilada ayudará a mejorar las estrategias educativas y a diseñar herramientas más efectivas para el aprendizaje de programación. Su participación y honestidad en las respuestas serán fundamentales para el éxito de este estudio.

1. **¿Qué lenguajes de programación se imparte en clase y qué herramientas de desarrollo son utilizadas para ello?**

   _____

   _____

   _____

2. **¿Cuáles son los principales desafíos que encuentra al enseñar programación?**

   _____

   _____

   _____

3. **¿Ha considerado alguna plataforma de programación visual? En caso afirmativo, ¿Cuál fue su experiencia al utilizarla?**

   _____

   _____

   _____

4. **¿Qué tipos de recursos considera más útiles para enseñar programación a los estudiantes? Por ejemplo: tutoriales, guías, proyectos, participación colectiva o trabajo en clase.**

_____

_____

_____

5. **¿Considera importante integrar la programación con conocimientos de electrónica? ¿Por qué?**

_____

_____

_____

6. **¿Qué características considera más importantes en el desarrollo del kit de aprendizaje inalámbrico?**

_____

_____

_____

Revisado por:	Realizado por:
Director: Msc. Suárez Zambrano Luis Edilberto	Estudiante: Jaime Alexander Olmedo Moreno

ENCUESTA RESPONDIDA

La encuesta fue realizada mediante la herramienta de Google Forms la respuesta obtenida por el Profesor Héctor Carrera es la siguiente:

23/3/25, 21:39 Diseño de un kit de aprendizaje IoT físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de program...

Diseño de un kit de aprendizaje IoT físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de programación.

Esta encuesta tiene como objetivo explorar las experiencia y perspectiva del docente en la enseñanza de programación. Se busca conocer los lenguajes de programación que se imparten en clases, las herramientas de desarrollo utilizadas, así como la percepción sobre el uso de plataformas de programación visual incluyendo la integración de conocimientos electrónicos.

La información recopilada ayudará a mejorar las estrategias educativas y a diseñar herramientas más efectivas para el aprendizaje de programación. Su participación y honestidad en las respuestas serán fundamentales para el éxito de este estudio.

Nombres y Apellidos *

Hector Carrera

¿Qué lenguajes de programación se imparte en clase y qué herramientas de desarrollo son utilizadas para ello? *

En lenguaje principal está basado en programación orientada a objetos en el cual los estudiantes desarrollan la solución para problemas matemáticos, se imparten las sintaxis de control básicas y se integran con C Sharp # el cual propone el vinculo necesario para control e interacción usuario-maquina.

¿Cuáles son los principales desafíos que encuentra al enseñar programación? *

La complejidad de instalación de software, algunos estudiantes tienen problemas con librerías, falta de interés, el uso de herramientas externas como Chat GPT en resolución de problemas.

¿Ha considerado alguna plataforma de programación visual? En caso afirmativo, ¿Cuál fue su experiencia al utilizarla? *

En el proceso no se ha considerado una herramienta de programación visual

¿Qué tipos de recursos considera más útiles para enseñar programación a los estudiantes? Por ejemplo: tutoriales, guías, proyectos, participación colectiva o trabajo en clase. *

Considero parte importantes las Guías, proyectos y la participación colectiva entre estudiantes ya que esto supone una forma de investigación y motivación para generar conocimiento a futuro.

¿Considera importante integrar la programación con conocimientos de electrónica? ¿Por qué? *

Sí, ya que muchos proyectos hoy en día son generados con lenguajes que integran o vinculan el control con interfaz de usuario, y generalmente es necesario conocer tanto de Hardware como Software.

¿Qué características considera mas importantes en el desarrollo del kit de aprendizaje inalámbrico? *

- [ ] Conectividad inalámbrica (Wi-Fi, Bluetooth).
- [ ] Compatibilidad con sensores y actuadores.
- [x] Interfaz de programación visual.
- [x] Componentes modulares para proyectos personalizables.
- [x] Documentación y recursos educativos interactivos (tutoriales, guías, proyectos predefinidos)

ANEXO B: Formato de Encuesta para Estudiantes

Encuesta

Tema: Encuesta de requerimientos de usuario del prototipo IoT para enseñanza de lógica de programación

Objetivo: Esta encuesta está dirigida a los estudiantes de la Unidad Educativa Alberto Enríquez para poder considerar requerimientos previos de usuario, en ella se abordan temas de programación, electrónica y software de desarrollo así mismo la disponibilidad y motivación en el desarrollo de lógica de programación.

1. **¿Qué lenguajes de programación conoces? (Selecciona todos los que apliquen)**

C	C#	Python	Java	JavaScript	Scratch	Otros

2. **¿Tienes experiencia previa en programación con microcontroladores (como Arduino, ESP32, etc.)?**

Si	No	Un Poco

3. **¿Qué herramientas de desarrollo has utilizado anteriormente? (Selecciona todas las que apliquen)**

Arduino IDE	Visual Studio	Thonny	Plataformio	Otros

4. **¿Qué componentes electrónicos conoces o has utilizado? (Selecciona todos los que apliquen)**

Led	Resistor	Motor	Sensor de Luz	Transistor	Sensor de Proximidad	Otros

5. **¿Cuál es tu método preferido para aprender programación y electrónica? (Selecciona todos los que apliquen)**

Clases Teóricas	Clases Prácticas (Laboratorio)	Video Tutoriales	Guías Paso a Paso	Programación Visual	Otros

6. **¿Qué características te gustaría que tuviera un kit de aprendizaje IoT? (Selecciona todos los que apliquen)**

componentes físicos (sensores, cables, etc.)	Software de programación visual	Manuales o guías paso a paso	Proyecto o desafío final para aplicar lo aprendido	Conexión con servicios en la nube (como Firebase)	Otros

7. **¿Qué tipo de proyectos te gustaría realizar con el kit de aprendizaje IoT? (Selecciona todos los que apliquen)**

Medir temperatura	Controlar luces	Crear sistemas de monitoreo	Automatizar hogar	Crear Robots o vehículos controlados	Otros

8. **¿Te gustaría utilizar un software de programación visual para programar?**

Si	No	Tal Vez

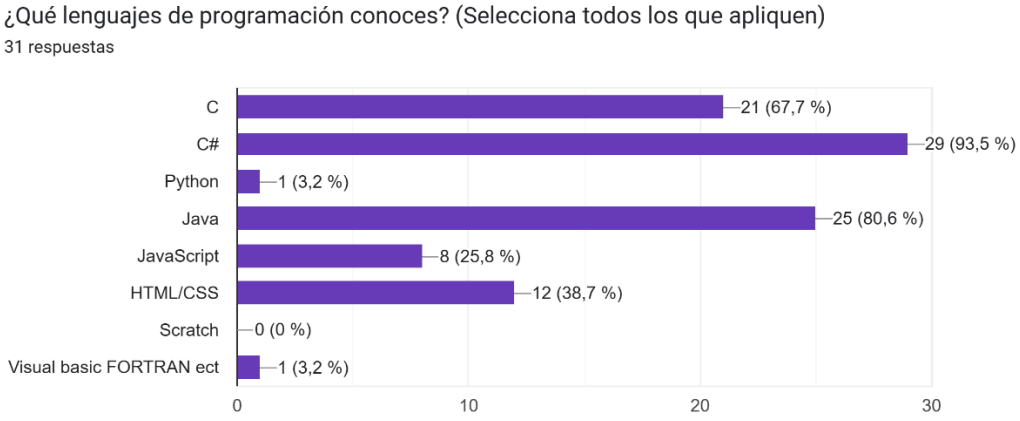
9. **¿Estarías dispuesto a aprender programación con un kit que tenga un enfoque práctico, utilizando hardware físico y software?**

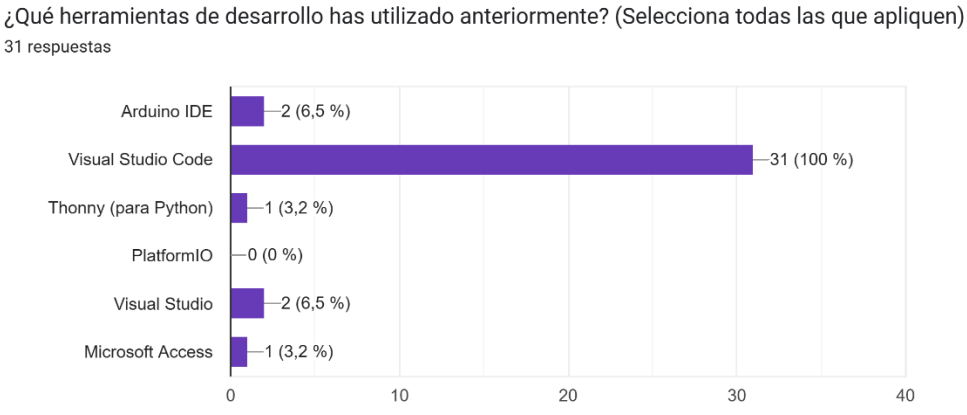
Si	No	Tal Vez

Revisado por: Director: Msc. Suárez Zambrano Luis Edilberto	Realizado por: Estudiante: Jaime Alexander Olmedo Moreno

ENCUESTA RESPONDIDA

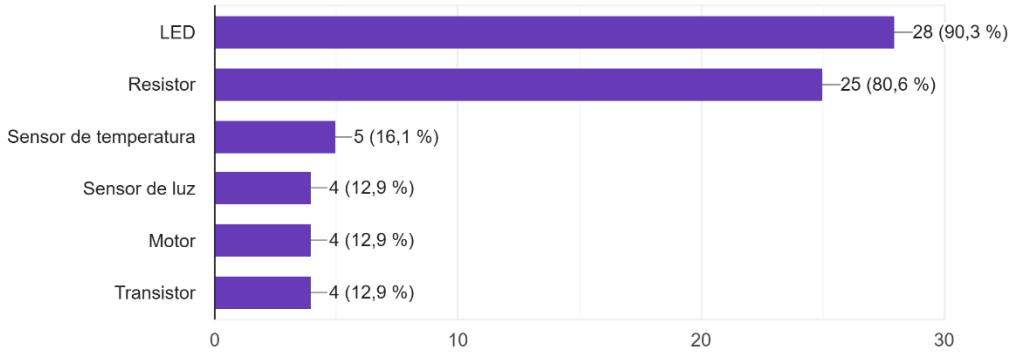
Gracias a la infraestructura disponible se utilizó Google Forms en donde el resumen de treinta y un participantes fue:





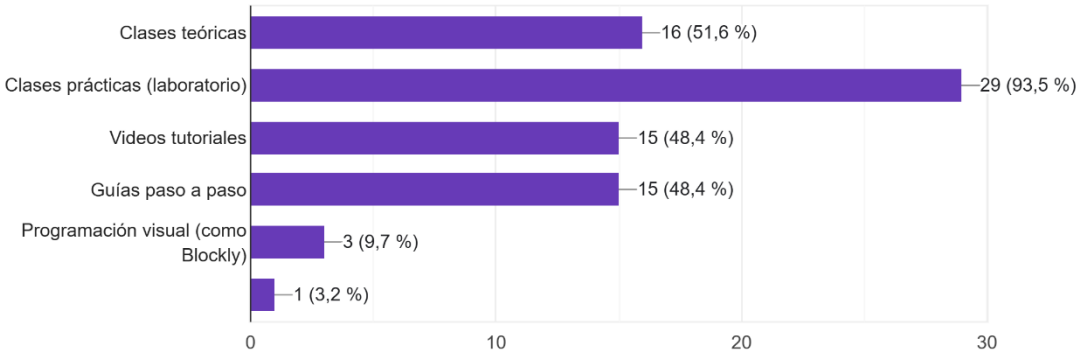
¿Qué componentes electrónicos conoces o has utilizado? (Selecciona todos los que apliquen)

31 respuestas



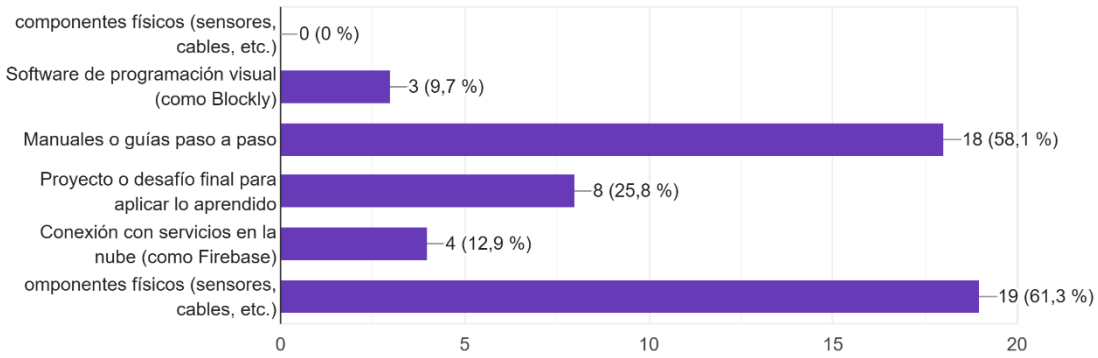
¿Cuál es tu método preferido para aprender programación y electrónica? (Selecciona todos los que apliquen)

31 respuestas



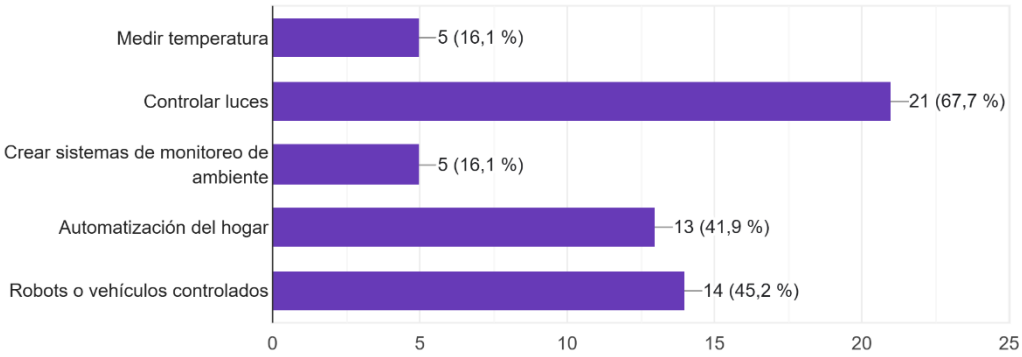
¿Qué características te gustaría que tuviera un kit de aprendizaje IoT? (Selecciona todos los que apliquen)

31 respuestas



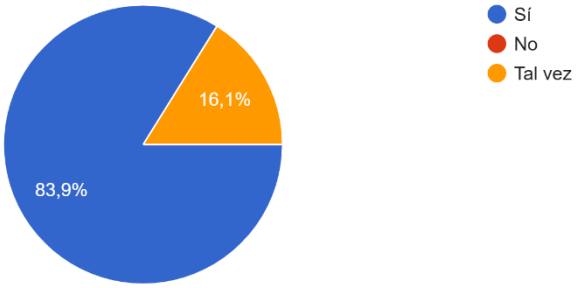
¿Qué tipo de proyectos te gustaría realizar con el kit de aprendizaje IoT? (Selecciona todos los que apliquen)

31 respuestas



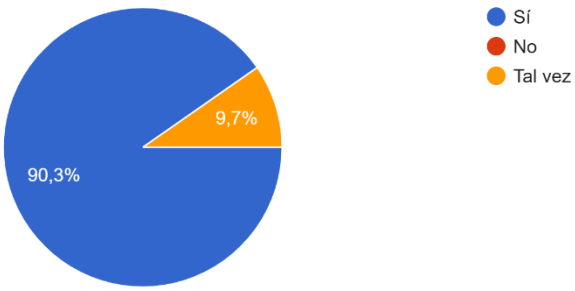
¿Te gustaría utilizar un software de programación visual para programar ?

31 respuestas



¿Estarías dispuesto a aprender programación con un kit que tenga un enfoque práctico, utilizando hardware físico y software?

31 respuestas



ANEXO C: Evaluación de Experiencia de Usuario

Encuesta de Experiencia de Usuario

Tema: Diseño de un kit de aprendizaje IoT físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de programación.

Objetivo: Recolectar información sobre la experiencia de los usuarios al interactuar con el kit de aprendizaje IoT físico, con el fin de evaluar su efectividad en el impulso del pensamiento lógico y la comprensión de los fundamentos de programación, así como identificar oportunidades de mejora en su diseño y funcionalidad.

1. **¿Cómo calificarías la facilidad de uso del kit de aprendizaje IoT?**

Muy Fácil de Usar	Fácil de usar	Neutral	Difícil de usar	Muy difícil de usar

2. **¿Qué actividad o proyecto con el kit te resultó más interesante? (Selecciona todas las que apliquen)**

Teoría sobre electrónica y programación	Control de luces	Control de botones	Control de Potenciómetros	Comunicación Inalámbrica	Interfaz Visual	Otro

3. **¿Qué dificultades encontraste al usar el kit de aprendizaje IoT? (Selecciona todas las que apliquen)**

Dificultad para entender la programación	Problemas con el hardware (componentes, conexiones, etc.)	No entendí cómo usar el software de programación	Problemas de conectividad o red	Ninguna dificultad	Otra

4. **¿Consideras que el kit de aprendizaje IoT te ayudó a comprender mejor los conceptos de programación y electrónica?**

Sí, mucho	Sí, un poco	No mucho	No, nada

5. **¿Qué mejoras sugerirías para el kit de aprendizaje IoT? (Selecciona todas las que apliquen)**

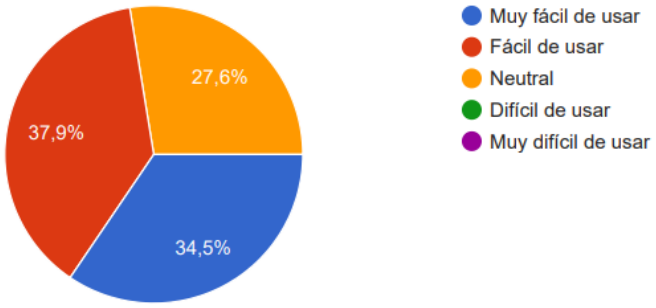
Incluir más proyectos prácticos	Mejorar la explicación de los conceptos en los manuales	Mejorar la calidad de los componentes hardware	Hacer el software más fácil de usar	Incluir videos tutoriales adicionales	Otro

Revisado por: Director Msc. Suárez Zambrano Luis Edilberto	Realizado por: Estudiante: Jaime Alexander Olmedo Moreno

ENCUESTA RESPONDIDA

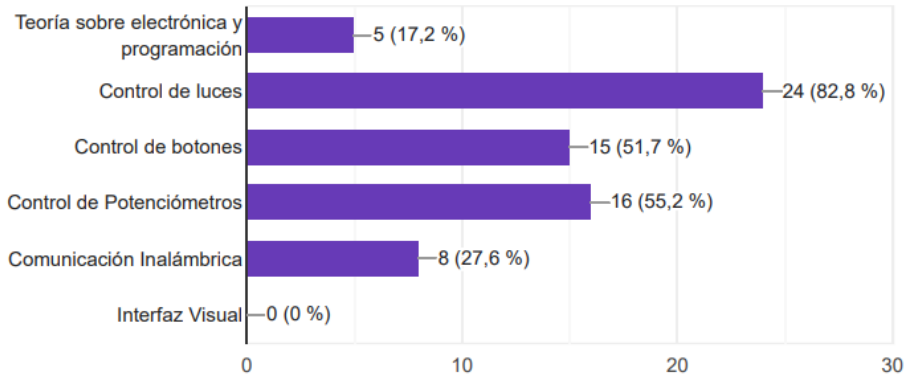
¿Cómo calificarías la facilidad de uso del kit de aprendizaje IoT? Copiar

29 respuestas



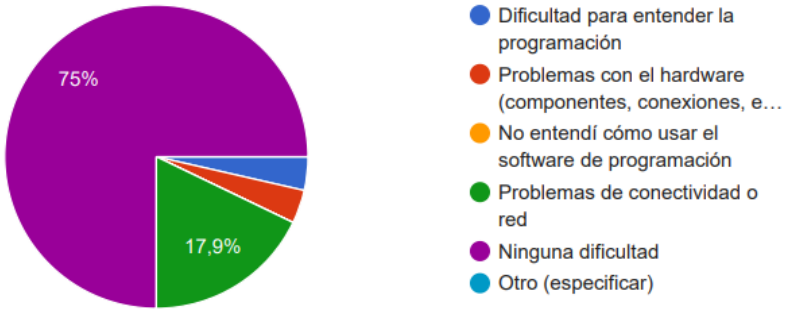
¿Qué actividad o proyecto con el kit te resultó más interesante? (Selecciona todas las que apliquen) Copiar

29 respuestas



¿Qué dificultades encontraste al usar el kit de aprendizaje IoT? (Selecciona todas las que apliquen) Copiar

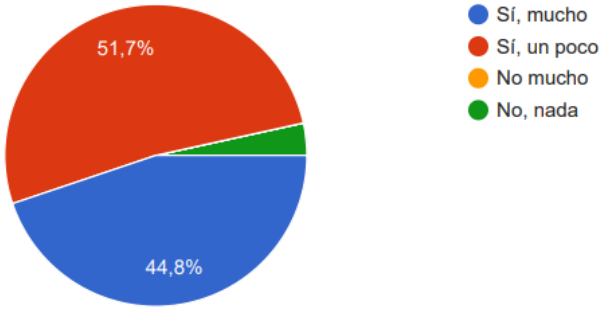
28 respuestas



¿Consideras que el kit de aprendizaje IoT te ayudó a comprender mejor los conceptos de programación y electrónica?

Copiar

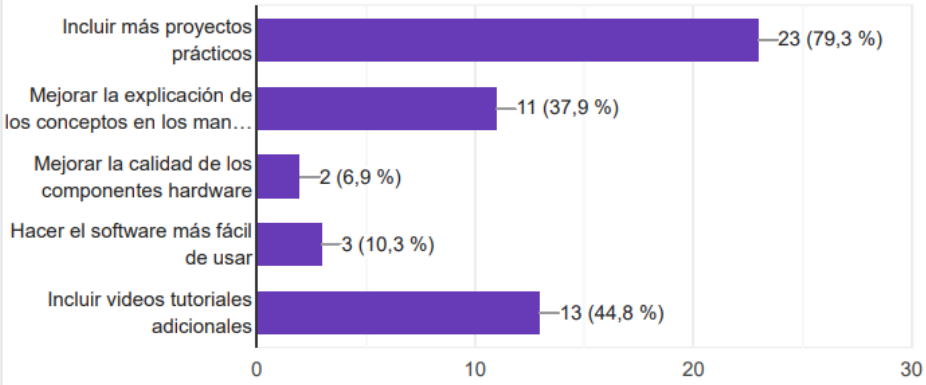
29 respuestas



¿Qué mejoras sugerirías para el kit de aprendizaje IoT? (Selecciona todas las que apliquen)

Copiar

29 respuestas



ANEXO D: Plan de Pruebas

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE TELECOMUNICACIONES RPC-SO-31-No 573-2016

Matriz y Cronograma de Actividades

Unidad/Empresa/Beneficiario:
Estudiante: Olmedo Moreno Jaime Alexander
Tutor: Msc. Luis Suarez Director (x) Asesor ( )

Nombre Trabajo de Titulación
Diseño de un 'Kit de Aprendizaje IoT' físico que permita impulsar el desarrollo del pensamiento lógico en fundamentos de programación

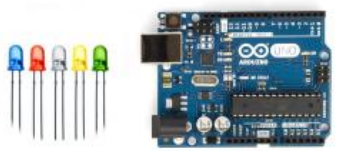
Objetivo General	Objetivos Específicos	Actividades	Metodología	Resultados por Objetivo	Medio de Verificación	Días 1	2	3	4	5	6	7	8	9	10	% de avance Programado	Ponderación % del global	Observaciones
Diseñar un "Kit de Aprendizaje IoT" físico el cual disponga de sensores y actuadores con el propósito de impulsar el desarrollo del pensamiento lógico y crítico en fundamentos de programación	Introducción	Introducción y configuración inicial	Metodología en cascada: Fase Introduccion	Presentar el prototipo y sus respectivas funciones a estudiantes	Introduccion y Conceptos base											15	7,5	
		Familiarización con el prototipo															7,5	
	Fundamentos y Primeros Pasos	Introducción a conceptos de IoT y programación	Metodología en cascada: Fase Marco Teórico	Comprensión de métodos logicos y fundamentos bases de programación aplicados con prototipo	Estructura Hardware Y software mediante divulgación de Turoria											15	7,5	
		Clase práctica															7,5	
	Personalización e Integración	Trabajo en equipo y personalización	Metodología en cascada: Fase Implementación	Trabajo en equipos para la compresión de comunicación entre nodos	Prototipado con estudiantes											40	20	
		Generación de proyecto															20	
	Presentación y Cierre	Exposición de los proyectos desarrollados.	Metodología en cascada: Fase Discusión y Evaluación	Documentación de pruebas Realizadas	Videos de pruebas piloto											30	10	
		Discusión sobre Prototipo y manejo de errores															10	
		Recopilación de observaciones y retroalimentación.															10	
Registro de Aprobacion				Observaciones												100	100	

Firma Tutor o Asesor

Firma Estudiante

ANEXO E: Diapositivas Clase Introducción

Microcontroladores



PROCESAN

CONTROL

EJECUTAN

Telecomunicaciones

Física

Temperatura

Tiempo

Peso

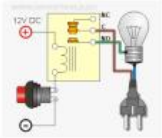
Cantidad de luz

Sensores

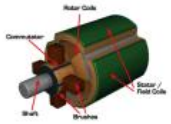
Telecomunicaciones

Física

Relé



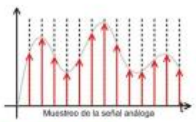
Motor

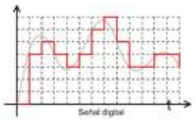


Servo Motor

Telecomunicaciones

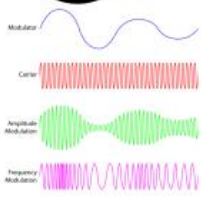
Señales digitales y analógicas

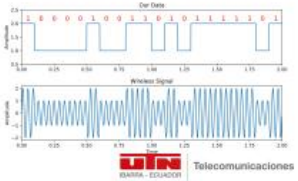




Telecomunicaciones

Señales Inalámbricas





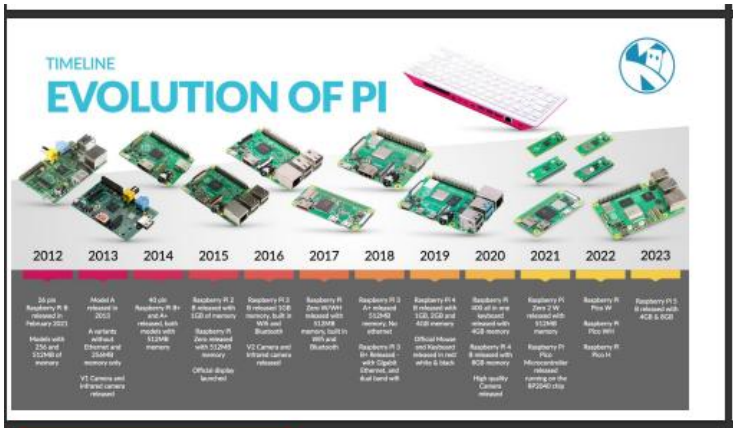
Telecomunicaciones

Nuevo Dispositivos

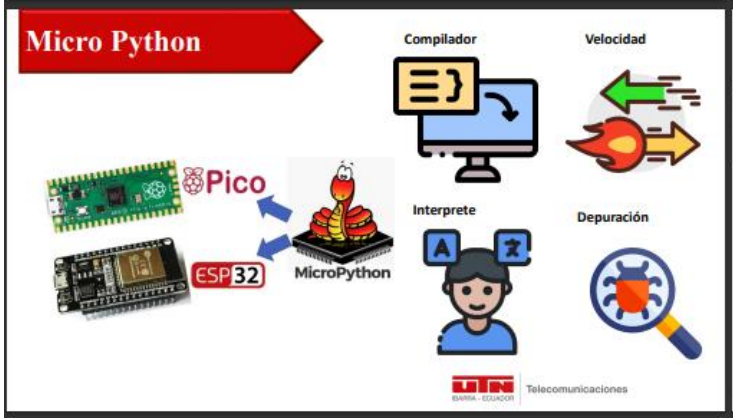
Raspberry

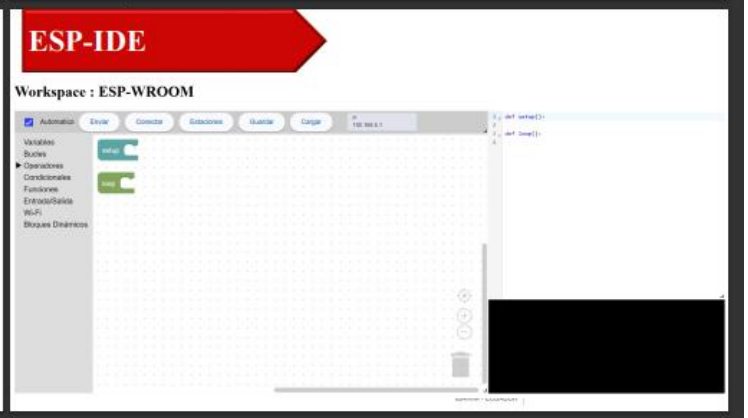
MODULO ESP

Telecomunicaciones


TIMELINE
EVOLUTION OF PI
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023

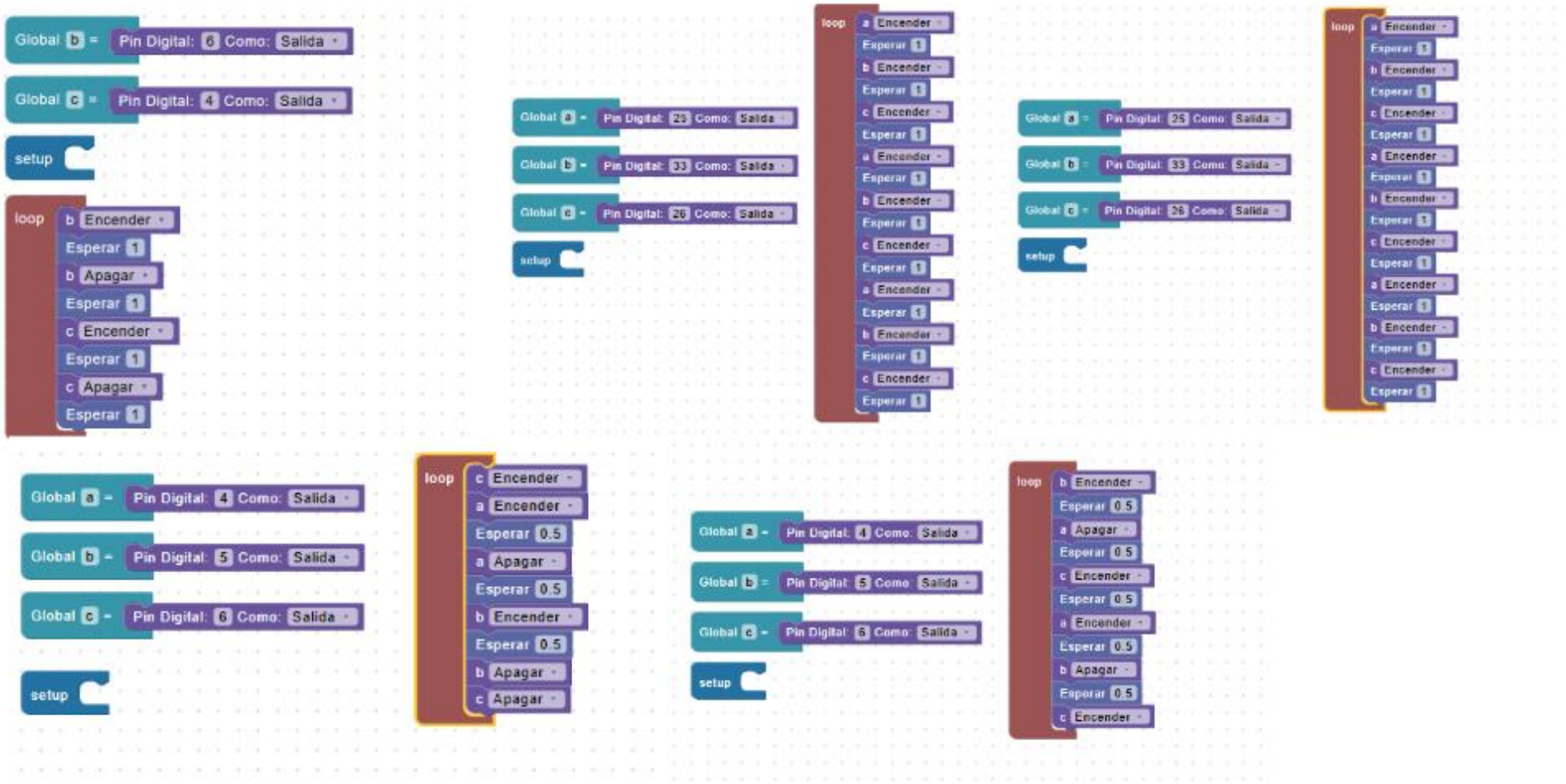

PASCAL
node
JS
php
Microsoft
.NET
C
A
JS
C++
RAILS
TS
C#
Java
HTML
Lua
X
free pascal


Micro Python
Compilador
Velocidad
Pico
ESP32
MicroPython
Interprete
Depuración
Telecomunicaciones


ESP-IDE
Workspace : ESP-WROOM


GRACIAS
AÑOS
36
UTN
Innovar
para asegurar
un buen futuro

ANEXO F: Prueba Usabilidad (Familiarización)



ANEXO G: Prueba Usabilidad (Clase Práctica)