Signatory of DORA
https://doi.org/10.56183/iberotecs.v4i1.643
Ibero-American Journal of Engineering & Technology Studies
OPEN ACCESS
BY-NC-ND
**Received**: 07/02/2024
**Accepted**: 29/03/2024
**Published**: 04/04/2024

# Creación de un IDE web de gestión de firmware multilenguaje para un dispositivo IOT ESP32

Creation of a multi-language firmware management web IDE for an ESP32 IOT device

**Miguel Ignacio Paladines Condoy**
mpaladines5@utmachala.edu.ec
https://orcid.org/0009-0007-8602-0668
Universidad Técnica de Machala, Ecuador

**Oscar Eduardo Vera Macias**
overa2@utmachala.edu.ec
https://orcid.org/0009-0008-6962-9102
Universidad Técnica de Machala, Ecuador

**Dixys Leonardo Hernández Rojas**
dhernandez@utmachala.edu.ec
https://orcid.org/0000-0002-2116-6531
Universidad Técnica de Machala, Ecuador

## RESUMEN

El Internet de las Cosas ha avanzado significativamente, generando soluciones mediante dispositivos mejorados o nuevos, equipados con microprocesadores y sensores. La programación de firmware es crucial en este contexto, pero la diversidad de sistemas y lenguajes ha fragmentado la comunidad de desarrolladores, restringiendo la flexibilidad en programación de microcontroladores. Para superar esto, se ha creado un sistema de programación de firmware multilenguaje (C++ y MicroPython), a través de un entorno de desarrollo integrado en la web en tiempo real, destacando una ventaja significativa al momento de la conexión multiusuarios. Esta innovación permite programar firmware de múltiples maneras, evitando la restricción a un solo lenguaje y fomentando el aprendizaje y entendimiento en este ámbito. Además, al admitir varios lenguajes, el sistema ofrece diferentes enfoques para resolver problemas, facilitando la colaboración y la experimentación en la programación de microcontroladores.

**Palabras claves:** iot; microcontrolador; ide; multiusuario.

## ABSTRACT

The Internet of Things has significantly progressed, offering solutions through enhanced or new devices equipped with microprocessors and sensors. Firmware programming is essential in this context, but the variety of systems and languages has fragmented the developer community, limiting flexibility in microcontroller programming. To overcome this, a multilanguage firmware programming system (C++ and MicroPython) has been developed, accessible through a real-time web-integrated development environment, highlighting a significant advantage in multiuser connection. This innovation allows for multiple ways of programming firmware, avoiding limitations to a single language and promoting learning and understanding in this field. Additionally, by supporting various languages, the system presents diverse approaches to problem-solving, enhancing collaboration and experimentation in microcontroller programming.

**Keywords:** iot; microcontroller; ide; multiuser

## INTRODUCCIÓN

El progreso en aplicaciones IoT se ha vuelto esencial para investigadores gracias a la omnipresencia de Internet. Para mejorar la facilidad de uso, las tecnologías web han cobrado relevancia en este campo innovador (Satyendra K Vishwakarma & Prashant Upadhyaya, 2019). Ante ello se busca la automatización de procesos, facilitando así la vida del ser humano, la comunicación simultanea entre usuarios y dispositivos agilizando la eficiencia al momento de resolver una interrogante o mejorar un proceso del diario a vivir (Al-Dabass & Institute of Electrical and Electronics Engineers, 2014).

Un objetivo específico de estas nuevas tecnologías se enfoca en conectar dispositivos y máquinas a través de Internet, permitiendo que se intercambien datos sobre su funcionamiento automáticamente, sin necesidad de intervención humana (Omar Otoniel Flores-Cortez & Verónica Idalia Rosa, 2018). Los microcontroladores cumplen un rol fundamental al momento de querer implementar las soluciones que mejorarían el aprendizaje de IoT. (Díaz Ronceros, 2020) Sostiene que el microcontrolador es un chip que contiene una estructura parecida a la de un ordenador, incluyendo elementos como memoria RAM, EEPROM, CPU y periféricos de entrada/salida, también conocidos como I/O.

En la actualidad existen una variedad de formas de usar un microcontrolador, ya sea en laboratorios remotos que posibilitan a los usuarios llevar a cabo experimentos prácticos en electrónica a través de Internet como lo indica (S. López & A. Carpeño, 2014). O al realizar pequeños proyectos, uno de los microcontroladores que mas resaltan son los ESP32 siendo de los más usados por su estructura de dos núcleos, una extensa variedad de periféricos, y su habilidad para realizar múltiples funciones, como la comunicación sin cables, el procesamiento de señales, y el control de sensores y actuadores (Marek & Pavel, 2019).

La programación de estos microcontroladores (ESP32) es esencial al momento de querer innovar en el mundo de la tecnología (Joni Welman Simatupang & Aida Mahdalena Lubis, 2022), por la múltiples funcionalidades que brindan y dentro

del ámbito de estos, dicho microcontrolador (ESP32) es el que más destaca o en su defecto siendo el preferido por muchos desarrolladores al momento de buscar componentes IoT para sus trabajos, proyectos y/o investigaciones (Husam Karim & Dmitry Dunaev, 2021).

En la forma de programar un ESP32 el lenguaje más común para hacerlo es el C++ como opina (Prem Prakash Murmu & Harshit Paul, 2019), en tal caso dentro del medio académico la mayoría de prácticas de laboratorios son realizadas con dicho principio por tener un bajo costo al momento de realizarlas (Cyprian N. Oton & M. Tariq Iqbal, 2021), pero sin explorar nuevas soluciones que en muchos casos pueden ser más eficientes o con un menor tiempo de respuesta a la información solicitada.

Siendo esta una de las principales razones por la cual se aborda el desarrollo de este Articulo, ya que dentro del ámbito de la "Programación de Microcontroladores" existen otros lenguajes para poder hacer lo mismo, tal es el caso de Micropython que se trata de un lenguaje de programación de alto nivel, interactivo e interpretado. Al ser una versión adaptada de Python 3, comparte numerosas características con el estándar del lenguaje Python como señala (Valeriu Manuel Ionescu & Florentina Magda Enescu, 2020).

Por ello se busca demostrar que lenguaje tiene un mejor tiempo de respuesta, mejor latencia y facilidad al momento de grabar el firmware a un ESP32, en palabras de (Fengpei Yuan & Xiaolei Wang, 2019) el firmware es una categoría especializada de software diseñada para gestionar las operaciones de bajo nivel del hardware específico en un dispositivo. Mientras que el software de aplicación entrega funciones directas a los usuarios, el firmware sirve como un enlace entre el hardware del dispositivo y los programas más complejos, dictando al hardware su modo de funcionamiento.

Además de la comparación de los dos lenguajes existe la pregunta, ¿Cómo se podrá grabar dos diferentes lenguajes de programación a un ESP32?, por ende se implementó el desarrollo de un IDE que permita la grabación de firmware a través de una red local mediante el protocolo Over-the-Air (OTA) ya que ha aportado muchos beneficios en lo que a nuevas tecnologías se refiere (WenXuan Wang & Guihe Qin, 2022).

Otra pregunta para abordar es, ¿Cómo se podrán conectar múltiples usuarios al IDE de manera simultánea?, para dar una pequeña idea previa se usara un repositorio, que es GitHub Page que ofrece la característica de solicitud de extracción, la cual habilita a los desarrolladores para duplicar un proyecto, aplicar modificaciones al código y pedir a los dueños del proyecto que examinen e incorporen estas alteraciones del código en la línea principal del proyecto como opina (El Mezouar, 2019).

Dentro de este IDE diseñado en Visual Studio Code, hay una extensión importante para el desarrollo de la grabación hacia el ESP32 pero con lenguaje C++, la extensión en este caso es PlatformIO que permite unir las propuestas de tecnología física de los productores en un entorno creativo y eficiente para generar mayor innovación y rendimiento, para lenguaje C++ basado en Arduino (Miladinović, 2020).

Para el desarrollo de lenguaje MicroPython se usó otra extensión de Visual Studio Code, en este caso es Pymark que es un software creado para analizar el rendimiento y comparar cómo diferentes versiones de Python funcionan en diferentes sistemas. Su propósito es asistir a los desarrolladores para comprender y contrastar la eficacia del código escrito en Python bajo variadas condiciones y entornos(Scott, 2023) .

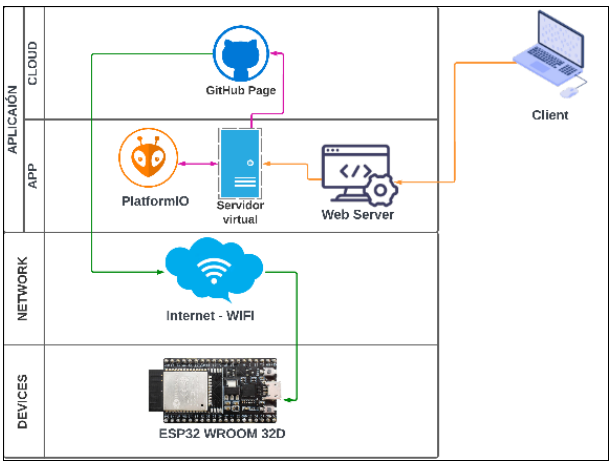
Por último punto al tratar el plan de evaluación se realizara en base a la norma ISO-25010, que en palabras de (Rocha, 2020) es un estándar global que establece y clasifica los aspectos de calidad de los sistemas y productos de software. Forma parte de la serie ISO/IEC 25000, conocida también como SQuaRE (Requisitos y Evaluación de la Calidad del Producto de Software). Dicho estándar identifica ocho atributos clave de calidad en el software, que incluyen funcionalidad, eficiencia de rendimiento, compatibilidad, facilidad de uso, fiabilidad, seguridad, facilidad de mantenimiento y capacidad de ser trasladado a otros sistemas.

## MÉTODOS

El estudio realizado al estar enfocado en el desarrollo y la mejora de firmware, se basó en un estudio mixto combinado análisis cuantitativo (rendimiento, eficiencia) y cualitativo (usabilidad, experiencia del usuario). Y con una investigación exploratoria porque se aborda aspectos innovadores o poco estudiados del firmware y sus usos.

Para respaldar nuestra investigación y facilitar el desarrollo del sistema, adoptamos la arquitectura de tres capas: Capa de dispositivo, Capa de Red y Capa de Aplicación (Chuquimarca & Maita, 2022). Esta arquitectura fue la base para el desarrollo, tal y como se representa a continuación en la Figura 1.

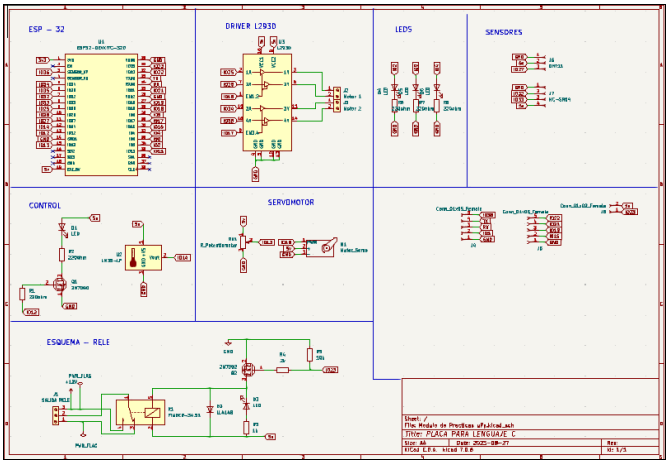
**Figura 1:** Representación de Arquitectura de 3 Capas.




### Capa de Dispositivo

El dispositivo fundamental que se uso es el ESP32 acompañado de otros dispositivos IOT, todo esto formando un circuito completo como se muestra en la Figura 2, donde se detalla cada uno de ellos. Los prototipos son conectados con el ESP32, que cumplen determinadas funciones, ya sea desde encender led's hasta mover un motor de 5v, a excepción de configuraciones de comunicación Inalámbrica, almacenamiento de datos y configuraciones que comprometan a la configuración principal del microprocesador.

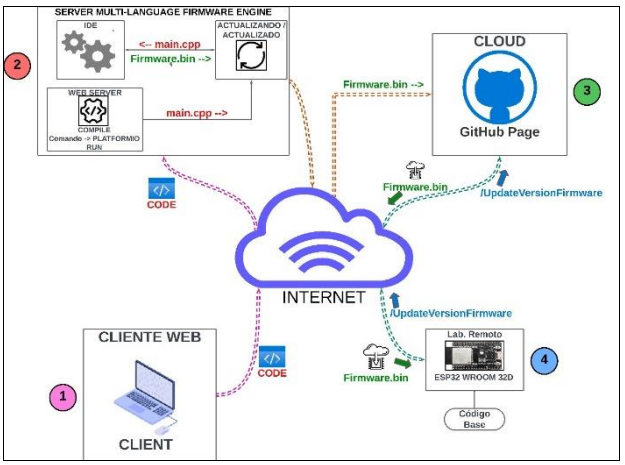
**Figura 2:** Circuito ESP32.




### Capa de Red

La capa de red es la encargada de establecer y gestionar la conectividad entre los dispositivos IoT y los servidores asegurando que sus datos lleguen a sus destinos sin ningún problema tal y como se muestra los puntos 2 y 3 de la Figura 3.

**Figura 3:** Arquitectura General detallada




### Capa de Aplicación

La capa de aplicación en la arquitectura de tres capas de IoT se divide en dos partes; la Cloud (GitHubPage) y la App.

Continuando con el proceso de desarrollo, se detalla a continuación en la Tabla 1.

| **Tabla 1** | |
|---|---|
| **Back-End** | Node.js permite la ejecución asíncrona de JavaScript, facilitando aplicaciones de red escalable (Rappl, 2023). A diferencia de modelos con hilos, gestiona múltiples conexiones sin bloqueo, y admite uso eficiente de múltiples núcleos. En la implementación de un sistema embebido, especialmente aquellos basados en el microprocesador como ESP32, una parte relevante del proceso es la compilación y actualización del firmware. |
| **Front-End** | El FrontEnd fue diseñado en React ya que una de las bibliotecas más ampliamente utilizadas en el mundo de JavaScript para crear aplicaciones tanto móviles como web (Dabit, 2019). En conjunto con el lenguaje TypeScript debido a que proporciona respuestas a una variedad de desafíos en el ámbito de JavaScript y se ha diseñado específicamente para la creación de aplicaciones sólidas (Da Costa, 2021). Introduce funcionalidades en el lenguaje que facilitan la creación de herramientas más avanzadas para el desarrollo de aplicaciones. Con el uso en conjunto de lo mencionado se realizó el FrontEnd. |
| **Base de datos y Creación del login** | Para la base de datos se usó la "Firestore DataBase" que es de FireBase, y por esta misma razón para la creación del Login se usó FireBase RealTime Database (Joni Welman Simatupang & Aida Mahdalena Lubis, 2022), ya que al momento en el que el usuario se logea en el sistema, de manera automática se le genera un userId con su propio token, esto le permite al usuario que cada vez que ingrese, tenga el mismo usuario que le designo de manera automática cuando se logeo con su correo por primera vez. |
| **GitHub** | En el GitHub se creará el número de carpetas que se requiera para cada usuario, estas carpetas se enlazaran a FireBase, para cuando se registre un nuevo usuario se le designe una determinada carpeta donde solo el usuario pueda interactuar. |
| **PlatformIO** | Como se muestra en la Figura 5, es la forma de cómo se implementó PlatformIO para poder programar el microcontrolador ESP32, y en la Figura 6 se describe la forma en cómo se realizó la intercomunicación entre el IDE y el microcontrolador, facilitando el proceso de grabar el firmware. |
| **Pymark** | Para la funcionalidad del Pymark para hacer que funcione el lenguaje MicroPython se dividió de la siguiente manera como se muestra en la Figura 4, boot.py: realiza la conexión a internet de la red local, main.py: es la plantilla editable donde se insertara el código para poder programar el ESP32 y ota.py: compara las versiones para que se cargue al ESP32 la más reciente. |

**Figura 4:** Desarrollo del Entorno Pymark.

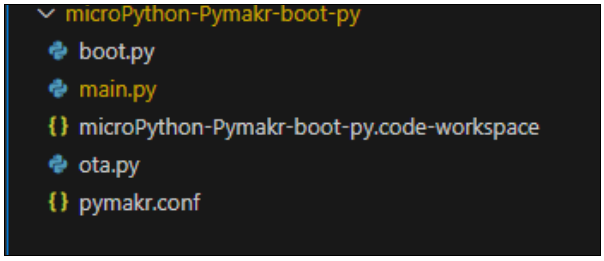



**Figura 5:** Código implementado de PlatformIO.

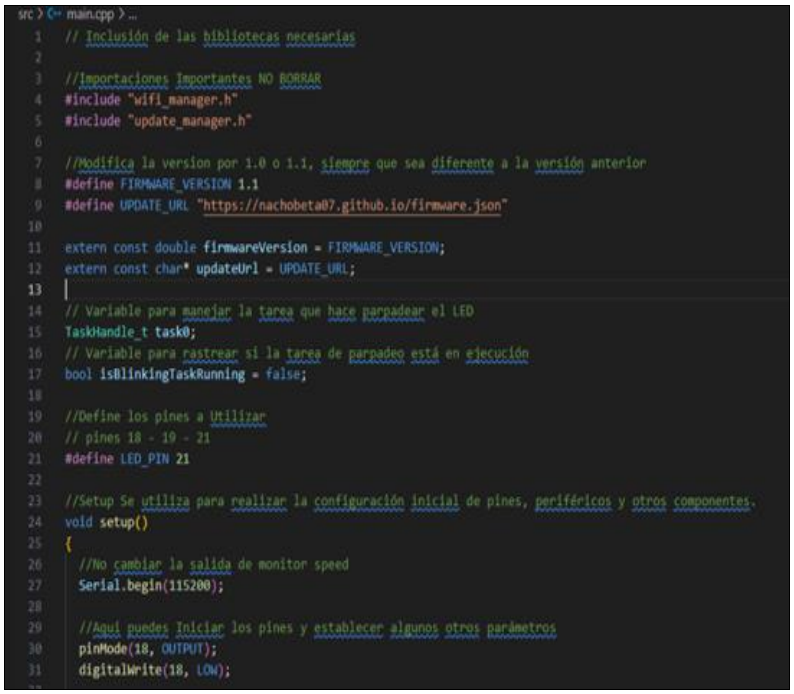
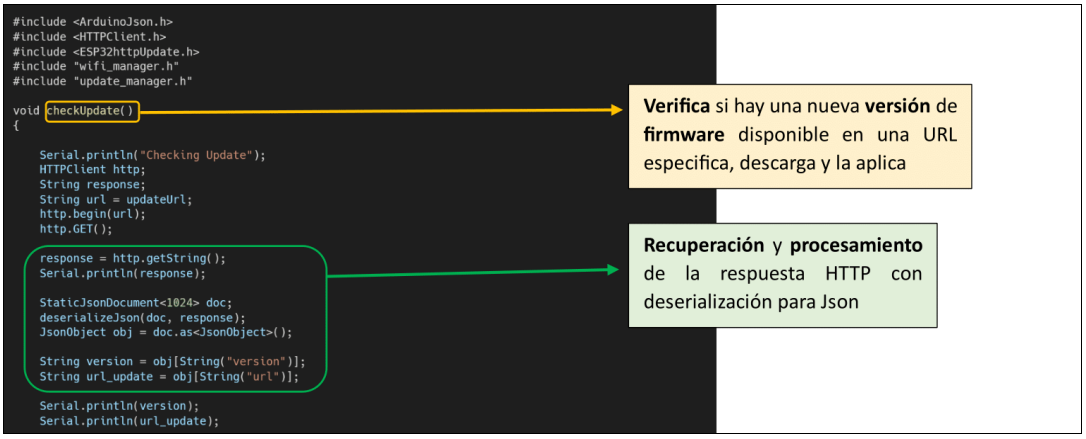


**Figura 6:** Código de actualización de versiones.




## RESULTADOS Y DISCUSIÓN

Para la obtención de datos empíricos necesarios para este estudio, se implementó una metodología de recolección de información a través de una encuesta estructurada (Díaz-Muñoz, 2020). Esta encuesta fue administrada a una muestra representativa de estudiantes de la Universidad Técnica de Machala, específicamente a aquellos matriculados en los semestres séptimo, octavo, noveno y décimo de la carrera de tecnologías de la Información. Se opto por una técnica de muestreo por convivencia, la cual, aunque no se adhiere a los principios del muestreo probabilístico o aleatorio, se fundamenta en la selección de sujetos basada en su accesibilidad y disponibilidad en un momento dado. En este caso, se seleccionaron estudiantes por cada nivel semestral mencionado, conformando así una muestra total de treinta individuos. Esta técnica se eligió debido a su eficiencia practica y operativa en el contexto del estudio en cuestión.

Se usará la Tabla 2, para poder evaluar el sistema de acuerdo a la Norma ISO-25010.

**Tabla 2:** Tabla de porcentajes de evaluación

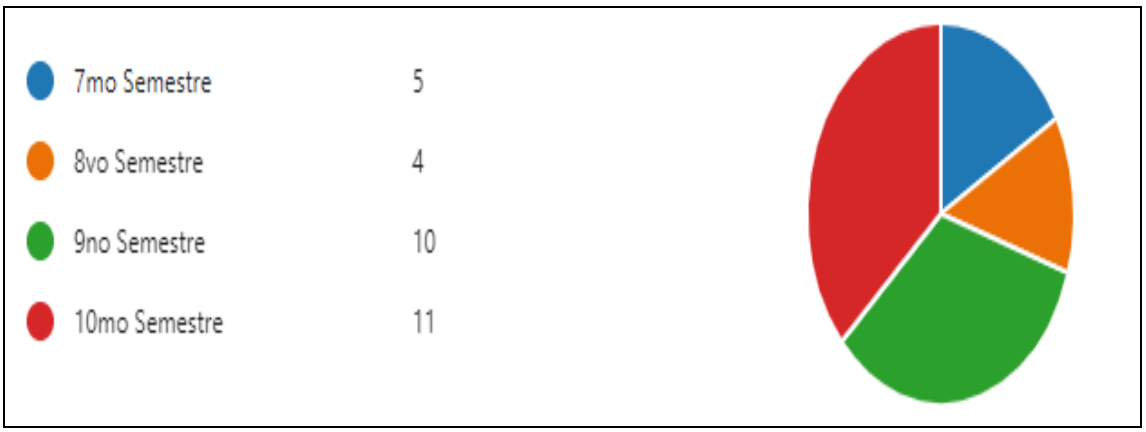
| Muy aceptable | Aceptable | Poco aceptable | Nada aceptable |
|---|---|---|---|
| **100% - 75%** | 74% - 51% | 26% - 49% | 0% - 25% |

A continuación, se detallará las preguntas realizadas a los encuestados y sus respuestas:

**1.**

**2. ¿En qué semestre te encuentras cursando de tu carrera de Ingeniería de tecnología de la Información?**

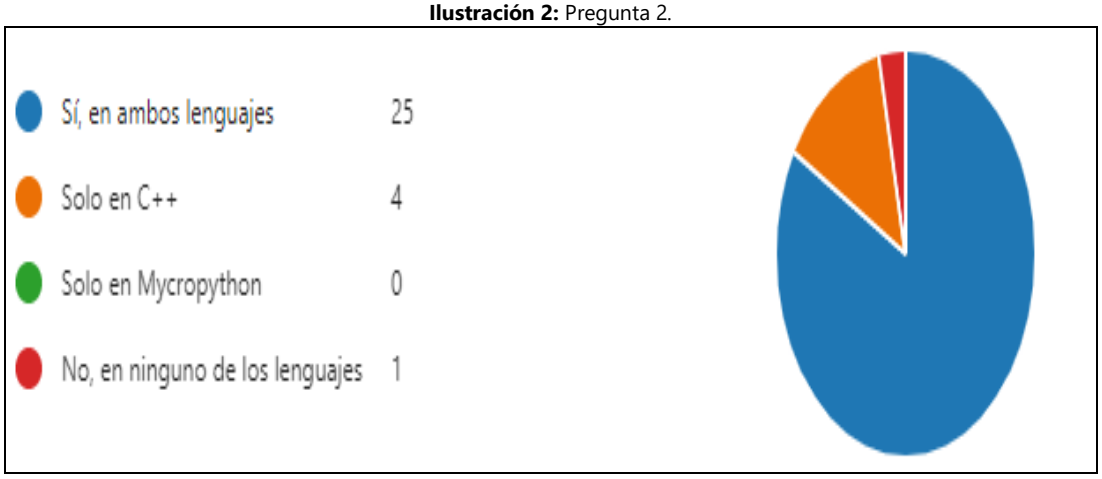
**Ilustración 1:** Pregunta 1




**-Funcionalidad**

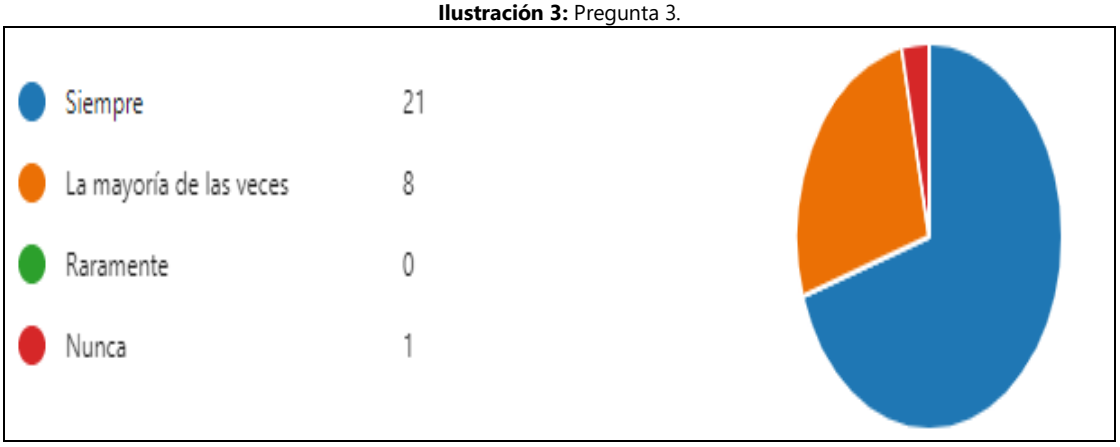
**3. ¿El sistema permite grabar firmware en los lenguajes C++ y Micropython de manera efectiva?**

**Ilustración 2:** Pregunta 2.




**Conclusión:** Un 83% de las personas respondió "Sí, en ambos lenguajes" y 4% de las personas respondieron No, en ninguno de los lenguajes.

**4. ¿La programación inalámbrica (OTA) funciona correctamente?**
**5.**
**Ilustración 3:** Pregunta 3.

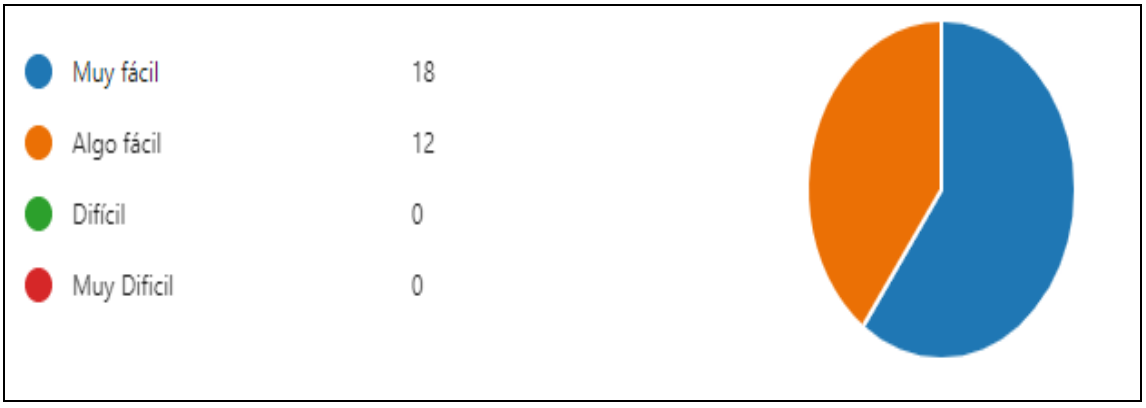



**Conclusión:** Un 68% de las personas respondió "Siempre" y 4% de las personas respondieron Nunca para esta pregunta.

**Tabla 3:** Resultado Obtenidos de la Funcionalidad del Sistema

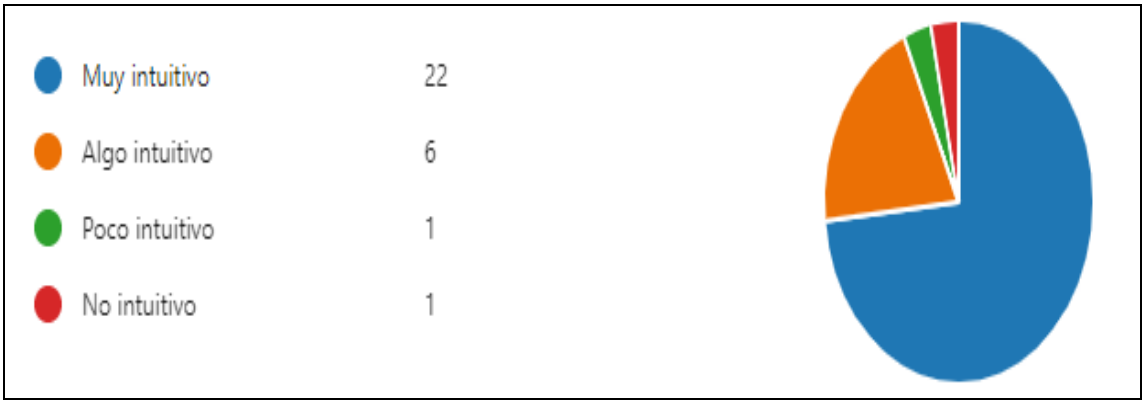
| Funcionalidad del Sistema | Porcentaje | Porcentaje total | Conclusión |
|---|---|---|---|
| **Pregunta 2** | 83% | 75,50% | En base a la recolección de información se ha obtenido así un resultado "Muy aceptable" de acuerdo a la Tabla 2. |
| **Pregunta 3** | 68% | | |

**-Usabilidad**
**1. ¿Cómo calificarías la facilidad de uso de la interfaz web?**
**Ilustración 4:** Pregunta 4.




**Conclusión:** Un 43% de las personas respondió "Algo fácil" y 58% de las personas respondieron Muy fácil para esta pregunta.

**2. ¿El proceso para seleccionar lenguajes, escribir código y compilar/cargar es intuitivo?**
**Ilustración 5:** Pregunta 5



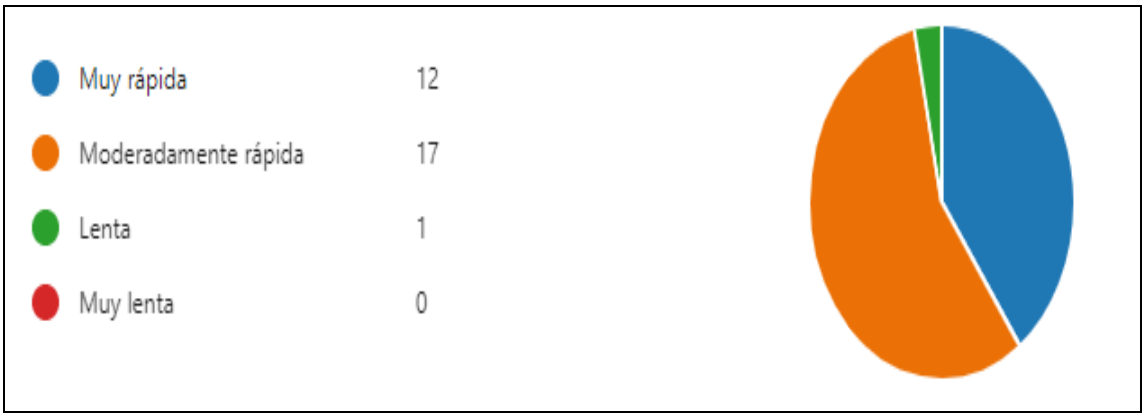

**Conclusión:** Un 4% de las personas respondió "Poco intuitivo" y 72% de las personas respondieron Muy intuitivo para esta pregunta.

**Tabla 4:** Resultado Obtenidos de la Usabilidad del Sistema

| Funcionalidad del Sistema | Porcentaje | Porcentaje total | Conclusión |
|---|---|---|---|
| **Pregunta 4** | 58% | 65,00% | En base a la recolección de información se ha obtenido un resultado "Aceptable" de acuerdo a la Tabla 2. |
| **Pregunta 5** | 72% | | |

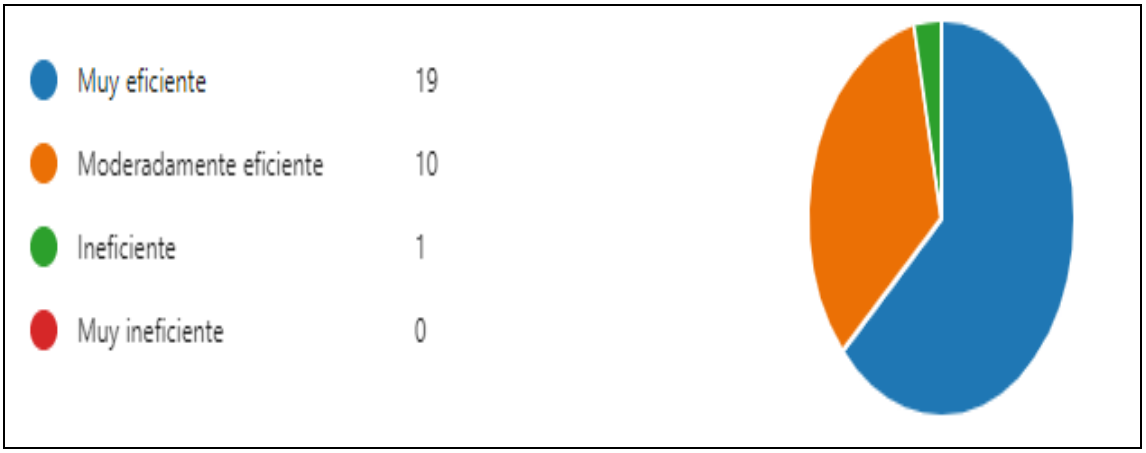
**- Eficiencia**

**3. ¿Cuál es la velocidad de compilación y carga del firmware?**
**Ilustración 6:** Pregunta 6.




**Conclusión:** Un 36% de las personas respondió "Muy rápida" y un 51% respondió que es "Moderadamente rápida" a esta pregunta.

**4. ¿Cómo evalúas la gestión de recursos del sistema durante operaciones intensivas?**
**Ilustración 7:** Pregunta 7.




**Conclusión:** Un 61% de las personas respondió "Muy eficiente" y un 32% respondió que es "Moderadamente eficiente" a esta pregunta.

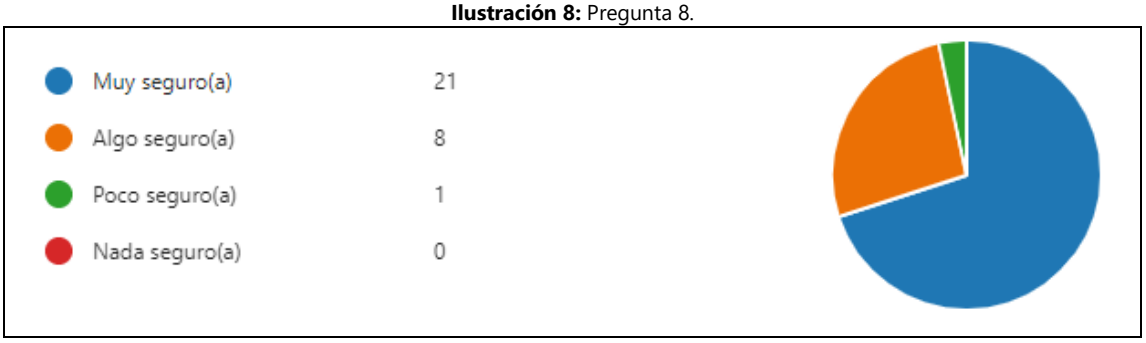
**Tabla 5:** Resultado Obtenidos de la Eficiencia del Sistema

| Funcionalidad del Sistema | Porcentaje | Porcentaje total | Conclusión |
|---|---|---|---|
| **Pregunta 6** | 51% | 56,00% | En base a la recolección de información se ha obtenido un resultado "Aceptable" de acuerdo a la Tabla 2. |
| **Pregunta 7** | 61% | | |

**- Seguridad**

**5. ¿Te sientes seguro(a) al usar el sistema durante el login y la manipulación de firmware?**

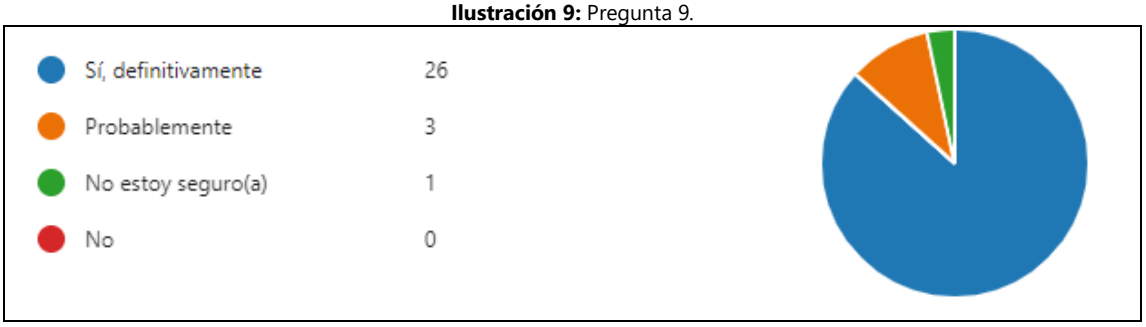
**Ilustración 8:** Pregunta 8.




**Conclusión:** Un 72% de las personas respondió "Muy seguro(a)" y 4% de las personas respondieron Poco seguro(a) para esta pregunta.

**6. ¿El sistema parece implementar prácticas seguras en la gestión de archivos y comunicación?**

**Ilustración 9:** Pregunta 9.




**Conclusión:** Un 86% respondió "Si, definitivamente" a esta pregunta.

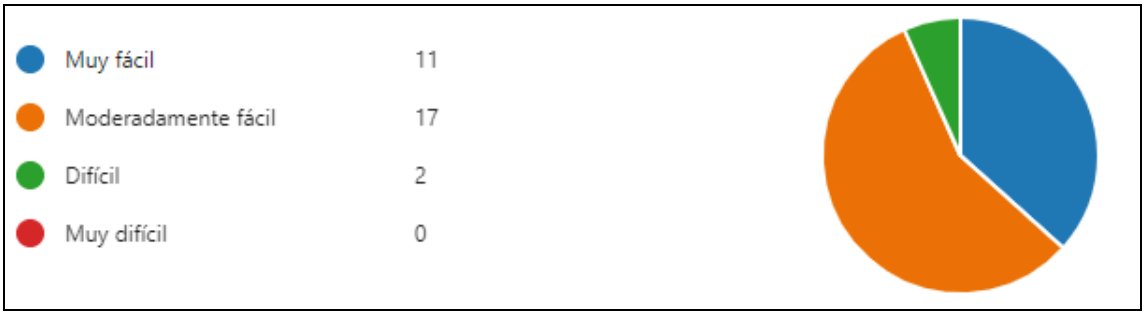
**Tabla 6:** Resultado Obtenidos de la Usabilidad del Sistema.

| Funcionalidad del Sistema | Porcentaje | Porcentaje total | Conclusión |
|---|---|---|---|
| **Pregunta 8** | 72% | 79,00% | En base a la recolección de información se ha obtenido un resultado "Muy aceptable" de acuerdo a la Tabla 2. |
| **Pregunta 9** | 86% | | |

**-Mantenibilidad.**

**7. ¿Consideras que sería fácil realizar cambios o actualizaciones en el sistema?**

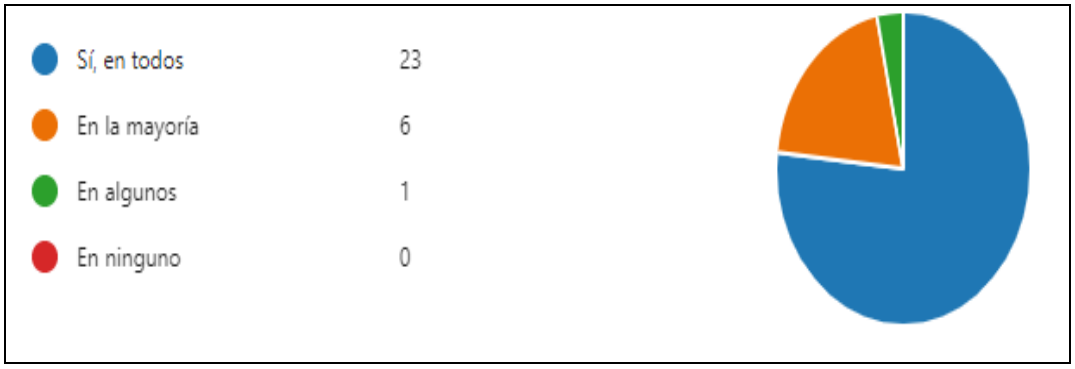
**Ilustración 10:** Pregunta 10.




**Conclusión:** Un 36% de las personas respondió "Muy fácil", un 55% respondieron "Moderadamente fácil" y 8% de las personas respondieron Difícil para esta pregunta.

**8. ¿El código te parece bien documentado y estructurado?**

**Ilustración 11:** Pregunta 11.




**Conclusión:** Un 54% de las personas respondió "Sí, totalmente" y 8% de las personas respondieron Poco para esta pregunta.

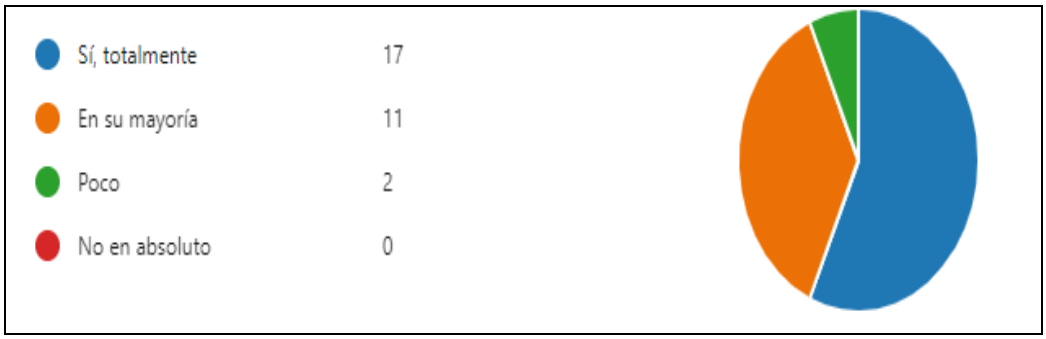
**Tabla 7:** Resultado Obtenidos de la Mantenibilidad del Sistema.

| Funcionalidad del Sistema | Porcentaje | Porcentaje total | Conclusión |
|---|---|---|---|
| **Pregunta 10** | 55% | 54,50% | En base a la recolección de información se ha obtenido un resultado "Aceptable" de acuerdo a la Tabla 2. |
| **Pregunta 11** | 54% | | |

**- Portabilidad**

**9. ¿El sistema funciona bien en diferentes navegadores y dispositivos?**

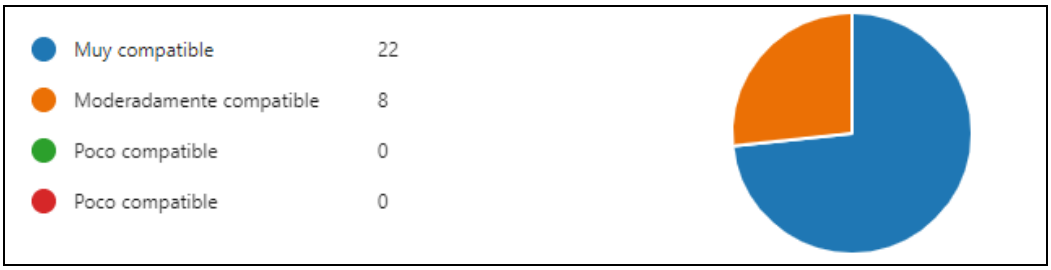
**Ilustración 12:** Pregunta 12.




**Conclusión:** Un 75% de las personas respondió "Sí, en todos" y 4% de las personas respondieron En algunos para esta pregunta y la mayoría respondió "En su mayoría" a la pregunta.

**10. ¿Cómo calificarías la compatibilidad del sistema con distintas versiones de dispositivos IoT (Microprocesadores - ESP32)?**

**Ilustración 13:** Pregunta 13.




**Conclusión:** Un 72% de las personas respondió "Muy compatible" y un 28% "Moderadamente compatible" a esta pregunta.

**Tabla 8:** Resultado Obtenidos de la Portabilidad del Sistema.

| Funcionalidad del Sistema | Porcentaje | Porcentaje total | Conclusión |
|---|---|---|---|
| **Pregunta 12** | 75% | 73,50% | En base a la recolección de información se ha obtenido un resultado "Aceptable" de acuerdo a la Tabla 2. |
| **Pregunta 13** | 72% | | |

## CONCLUSIONES

El estudio llevado a cabo ha propiciado la creación de un enfoque tecnológico innovador para la carga local de firmware en un dispositivo ESP32, utilizando dos lenguajes de programación distintos. Los análisis realizados a partir de la recopilación de datos obtenidos de encuestas a estudiantes nos han proporcionado una evaluación según la Norma ISO-25010:

En términos de funcionalidad, la calificación del 75.5% señala una robusta capacidad del entorno de desarrollo integrado web para la gestión de firmware en múltiples lenguajes. Esto indica una implementación efectiva que permite abordar la heterogeneidad de sistemas presentes en el entorno del Internet de las Cosas.

En cuanto a la usabilidad, a pesar de alcanzar un puntaje del 65%, se identifican áreas de mejora. Aspectos como interfaces más intuitivas y un flujo de trabajo más eficiente son esenciales, especialmente para usuarios con menos experiencia en la programación de firmware.

La eficiencia, reflejada en una calificación del 56%, indica la necesidad de optimizar el sistema. Estrategias para la gestión óptima de recursos, reducción de tiempos de carga y mejora del rendimiento global serían fundamentales para una experiencia de usuario más fluida.

En términos de mantenibilidad, la calificación de 54.50 sugiere la necesidad de esfuerzos adicionales. Se requieren estrategias efectivas para la gestión del código, documentación detallada y herramientas para facilitar actualizaciones y correcciones futuras.

La portabilidad, con una puntuación de 73.50, destaca la capacidad del sistema para adaptarse a diversos entornos, permitiendo a los usuarios trabajar en diferentes dispositivos y plataformas, lo que amplía su versatilidad.

En resumen, el desarrollo de un entorno de desarrollo web para gestionar firmware en múltiples lenguajes para dispositivos IoT ESP32 representa una iniciativa altamente prometedora. A pesar de las áreas de mejora identificadas en usabilidad, eficiencia y mantenibilidad, las evaluaciones positivas en funcionalidad y portabilidad resaltan su valor en el ámbito del desarrollo de firmware. La investigación basada en encuestas proporcionó una base sólida para evaluar la aplicabilidad práctica de esta solución entre estudiantes de tecnologías de la información. El enfoque en la flexibilidad de la programación, el estímulo al aprendizaje y la promoción de la colaboración entre diferentes enfoques para resolver problemas respaldan su potencial innovador en el contexto del Internet de las Cosas.

### Futuros desarrollos

Se considera adaptar el IDE desarrollado para que pueda ser implementado en un servidor web, lo que posibilitará un acceso remoto y una mayor flexibilidad en su utilización en laboratorios a distancia. Esta evolución no solo ampliará su cobertura, sino que también brindará una solución más adaptable y escalable, simplificando su inserción en entornos educativos e investigativos. Así mismo, se investigará la viabilidad de agregar características adicionales, como la gestión centralizada de varios dispositivos y la integración con sistemas de supervisión y control, con el fin de ofrecer una herramienta aún más completa y adecuada a las necesidades en constante evolución de la comunidad académica y tecnológica.

## REFERENCIAS

Al-Dabass, D., & Institute of Electrical and Electronics Engineers (Eds.). (2014). *Parameter-Based Mechanism for Unifying User Interaction, Applications and Communication Protocols*. IEEE.

Chuquimarca, C. E. T., & Maita, S. S. (2022). Análisis comparativo entre arquitecturas de sistemas IoT. *Revista de Investigación en Tecnologías de la Información: RITI*, *10*(21), 55-70.

Cyprian N. Oton & M. Tariq Iqbal. (2021). *Low-Cost Open Source IoT-Based SCADA System for a BTS Site Using ESP32 and Arduino IoT Cloud*. The e Institute of Electrical and Electronics Engineers, Inc.

Da Costa, L. (2021). *Testing JavaScript applications*. Manning Publications Co.

Dabit, N. (2019). *React Native in action: Developing iOS and Android apps with JavaScript*. Manning Publications.

Díaz Ronceros, E. (2020). Relevancia de la ejecución experimental de proyectos con microcontroladores en el aprendizaje de la ingeniería electrónica. *Educación*, *29*(56), 48-72.

Díaz-Muñoz, G. (2020). Metodología del estudio piloto. *Revista chilena de radiología*, *26*(3), 100-104.

El Mezouar, M. (2019). An empirical study on the teams structures in social coding using GitHub projects. *Empirical Software Engineering*, *24*, 3790-3823.

Fengpei Yuan & Xiaolei Wang. (2019). *The Method of Embedded Device Firmware Update Under Multi-layer Heterogeneous Network*. Conference Publishing Services, IEEE Computer Society.

Husam Karim & Dmitry Dunaev. (2021). *The Working Principles of ESP32 and Analytical Comparison of using Low-Cost Microcontroller Modules in Embedded Systems Design*. IEEE.

Joni Welman Simatupang & Aida Mahdalena Lubis. (2022). *IoT-Based Smart Parking Management System Using ESP32 Microcontroller*. IEEE.

Marek, B., & Pavel, S. (Eds.). (2019). *Using the ESP32 Microcontroller for Data Processing*. IEEE.

Miladinović, V. (2020). *Razvojna platforma PlatformIO: zaključno delo* [PhD Thesis]. Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko.

Omar Otoniel Flores-Cortez & Verónica Idalia Rosa. (2018). *Monitoreo remoto usando internet de las cosas*. IEEE.

Prem Prakash Murmu & Harshit Paul. (2019). *A Novel modernistic techniques in women security system using ESP32 and Arduino Uno*. IEEE.

Rappl, F. (2023). *Modern frontend development with Node.js the compendium for web development within the Node.js ecosystem*. Packt Publishing.

Rocha, Á. (2020). *Analysis of the Quality in Use and Greenability with the ISO/IEC 25010 Standard*. IEEE.

S. López, & A. Carpeño (Eds.). (2014). *Laboratorio remoto eLab3D: Un mundo virtual inmersivo para el aprendizaje de la electrónica*. IEEE.

Satyendra K Vishwakarma & Prashant Upadhyaya. (2019). *Smart Innovation and Usages (IoT-SIU)*. IEEE.

Scott, M. (2023). Apples to Oranges: Using Python and the pymarc library to match bookstore ISBNs to locally held eBook ISBNs. *Code4Lib Journal*, *56*.

Valeriu Manuel Ionescu & Florentina Magda Enescu. (2020). *Investigating the performance of MicroPython and C on ESP32 and STM32 microcontrollers*. IEEE.

WenXuan Wang & Guihe Qin. (2022). Un protocolo orientado a OTA para protección de seguridad. *2023 3.ª Conferencia Internacional sobre las Fronteras de la Electrónica, las Tecnologías de la Información y la Computación (ICFEICT)*.