UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE MECATRÓNICA

DESARROLLO DE UNA CONSOLA DE CODIFICACIÓN SIMPLIFICADA UTILIZANDO UN ESP32 PARA LA ENSEÑANZA DE PROGRAMACIÓN ORIENTADA A OBJETOS

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Mecatrónica

AUTORES: Jhostin Javier Alvarez Bermeo
Bryan Alexander Delgado Chuchuca
TUTOR: Michelle de los Ángeles Cárdenas Ibáñez

Guayaquil - Ecuador
2025

19/02/26

Escaneado con CamScanner

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, **Jhostin Javier Alvarez Bermeo** con documento de identificación Nº **1450220197** y **Bryan Alexander Delgado Chuchuca** con documento de identificación Nº **0931872691**; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo.

Guayaquil, 19 de febrero del año 2026

Atentamente,

Jhostin Javier Alvarez Bermeo
1450220197

Bryan Alexander Delgado Chuchuca
0931872691

Escaneado con CamScanner

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, **Jhostin Javier Alvarez Bermeo** con documento de identificación Nº **1450220197** y **Bryan Alexander Delgado Chuchuca** con documento de identificación Nº **0931872691**, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del **Dispositivo Tecnológico: DESARROLLO DE UNA CONSOLA DE CODIFICACIÓN SIMPLIFICADA UTILIZANDO UN ESP32 PARA LA ENSEÑANZA DE PROGRAMACIÓN ORIENTADA A OBJETOS**, el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 19 de febrero del año 2026

Atentamente,

Jhostin Javier Alvarez Bermeo
1450220197

Bryan Alexander Delgado Chuchuca
0931872691

Escaneado con CamScanner

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **Michelle de Los Ángeles Cardenas Ibáñez**, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **DESARROLLO DE UNA CONSOLA DE CODIFICACIÓN SIMPLIFICADA UTILIZANDO UN ESP32 PARA LA ENSEÑANZA DE PROGRAMACIÓN ORIENTADA A OBJETOS** , realizado por **Jhostin Javier Alvarez Bermneo** con documento de identificación N° **1450220197** y por **Bryan Alexander Delgado Chuchuca** con documento de identificación N° **0931872691**, obteniendo como resultado final el trabajo de titulación bajo la opción **Dispositivo Tecnológico** que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 19 de febrero del año 2026

Atentamente,

Ing. Michelle de Los Ángeles Cardenas Ibáñez, Mg.
0909627432

Escaneado con CamScanner

DEDICATORIA

Este trabajo de titulación está dedicado a mi madre, Jessica Janeth Bermeo Cambizaca, por su amor inmenso, su cuidado y su apoyo incondicional para asegurar mi bienestar y éxito durante todos estos años de estudios. A mi padre, Ángel Manuel Alvarez Bermeo, por sus invaluables enseñanzas, su guía y por inspirarme siempre a dar lo mejor de mí con disciplina y responsabilidad.

Finalmente, agradezco y dedico este logro a mi tía, Julia Leonor Cambizaca Lupercio, por su respaldo y confianza absoluta; a mis hermanos por ser mi motivación constante, y a mi familia en general, quienes siempre han cuidado de mí y me han impulsado a seguir adelante.

Jhostin Javier Alvarez Bermeo

Esta trabajo de titulación lo dedico a todas las personas que me han ayudado a formar mi carácter como ingeniero durante la carrera universitaria. A los profesores que siempre han estado dispuestos a ayudar y compartir su conocimiento y calidad humana, a los compañeros con los que mutuamente nos apoyamos y sobre todo, de mi parte a mi mama , pues es la principal motivación que he tenido durante todos estos años para nunca rendirme y esforzarme en ser una mejor persona cada día.

Bryan Alexander Delgado Chuchuca

AGRADECIMIENTO

Agradezco principalmente a Dios por brindarme la sabiduría y la fortaleza para alcanzar esta meta. A mis padres, Ángel Manuel Alvarez Bermeo y Jessica Janeth Bermeo Cambizaca, por su amor incondicional y por inculcarme el valor de la perseverancia para afrontar los retos y dificultades de la vida.

Agradezco de manera muy especial a mi tía, Julia Leonor Cambizaca Lupercio, y a mi abuela, Celia Maria Cambizaca Lupercio, por su apoyo constante, sus sabios consejos y por ser pilares fundamentales a lo largo de todos mis estudios.

Finalmente, agradezco a los docentes en general, por formar parte importante de mi desarrollo académico y por brindarme los conocimientos necesarios para la culminación del presente trabajo de investigación.

Jhostin Javier Alvarez Bermeo

Ofrezco este reconocimiento a todas las personas que me han estado apoyando y confiando en mí durante este proceso. A mi mama Ana Ibelia Chuchuca Serrano , que se ha sacrificado cada día de mi vida para que sea un profesional y es la mas ilusionada porque termine mi carrera. A mi amiga Narcisa Bonilla, pues siempre ha sido un apoyo para mi en momentos complicados durante mi vida universitaria. A todos los amigos y compañeros con los que nos hemos apoyado mutuamente para terminar la carrera y a mi novia que siempre me motiva a dar lo mejor de mi en cada ámbito.

Bryan Alexander Delgado Chuchuca

RESUMEN

La enseñanza de la Programación Orientada a Objetos (POO) y sistemas embebidos en ingeniería suele presentar una alta carga cognitiva para los estudiantes, generada por la complejidad técnica del hardware y la abstracción de los entornos virtuales. Para abordar esta problemática, el presente trabajo de titulación detalla el desarrollo de una consola de codificación simplificada e interactiva utilizando un microcontrolador ESP32-S3. La metodología aplicada consistió en cinco fases secuenciales: especificación de requisitos, diseño electrónico y mecánico (modelado CAD y manufactura FDM con polímeros PLA), desarrollo de firmware en C++ con integración del sistema operativo FreeRTOS, integración electromecánica, y validación técnica y pedagógica. El hardware final cuenta con una pantalla TFT RGB565, un joystick analógico, dos botones de acción y autonomía energética mediante una celda LiPo. Los resultados técnicos demostraron una autonomía de 4.2 horas continuas, un control térmico estable de 26°C en la carcasa exterior y un renderizado gráfico fluido a 35 FPS mediante Acceso Directo a Memoria (DMA). La validación pedagógica se realizó a través de cinco prácticas de laboratorio progresivas y una encuesta de usabilidad, obteniendo un índice de satisfacción sobresaliente (4.65/5). Se concluye que la consola mitiga eficazmente la curva de aprendizaje, permitiendo a los estudiantes interactuar físicamente con su código y asimilar los conceptos de POO mediante una capa de abstracción de hardware (HAL) robusta.

Palabras clave: Sistemas embebidos, Programación Orientada a Objetos, ESP32-S3, Aprendizaje activo, FreeRTOS, Manufactura aditiva.

ABSTRACT

The teaching of Object-Oriented Programming (OOP) and embedded systems in engineering often presents a high cognitive load for students, driven by technical hardware complexity and the abstraction of virtual environments. To address this issue, this graduation project details the development of a simplified and interactive coding console using an ESP32-S3 microcontroller. The applied methodology consisted of five sequential phases: requirements specification, electronic and mechanical design (CAD modeling and FDM manufacturing with PLA), C++ firmware development with FreeRTOS integration, electromechanical integration, and both technical and pedagogical validation. The final hardware features a TFT RGB565 screen, an analog joystick, two action buttons, and power autonomy via a LiPo cell. Technical results demonstrated 4.2 hours of continuous autonomy, stable thermal control at 26°C on the outer casing, and smooth graphic rendering at 35 FPS via Direct Memory Access (DMA). Pedagogical validation was carried out through five progressive laboratory practices and a usability survey, achieving an outstanding satisfaction rate (4.65/5). It is concluded that the console effectively mitigates the learning curve, allowing students to physically interact with their code and assimilate OOP concepts through a robust Hardware Abstraction Layer (HAL).

Keywords: Embedded systems, Object-Oriented Programming, ESP32-S3, Active learning, FreeRTOS, Additive manufacturing.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. Introducción

En el ámbito de la enseñanza de la ingeniería, la electrónica y la automatización, el aprendizaje de la programación de microcontroladores representa un desafío pedagógico significativo. Tradicionalmente, los estudiantes deben enfrentarse de manera simultánea a la complejidad de la sintaxis de lenguajes de programación, la configuración de registros a bajo nivel y el acondicionamiento eléctrico en placas de pruebas. Esta curva de aprendizaje inicial frecuentemente desvía la atención de los conceptos fundamentales de la lógica computacional, el pensamiento algorítmico y el diseño de sistemas de control.

Para mitigar esta barrera de entrada, el presente trabajo de titulación propone el desarrollo de *Code and Play*, una consola educativa interactiva basada en el microcontrolador ESP32 y programada en C++. El sistema integra un entorno de programación visual por bloques que abstrae la complejidad del hardware, permitiendo a los usuarios estructurar rutinas de control, leer sensores y manipular actuadores de forma intuitiva, para luego observar la respuesta física del equipo en tiempo real.

A nivel de diseño de hardware, el dispositivo ha sido concebido bajo principios de ergonomía y simplicidad cognitiva. La interfaz física prescinde del uso de teclados matriciales complejos, operando exclusivamente mediante un joystick analógico y dos botones de propósito general, lo que focaliza la interacción del estudiante. El encapsulado mecánico, fabricado mediante manufactura aditiva, presenta una estructura de paredes sólidas sin rejillas de ventilación ni nervaduras triangulares, optimizando su fabricación sin requerir soportes de impresión tipo árbol. Para facilitar la conectividad sin entorpecer el agarre frontal, el puerto USB-C y la ranura para la memoria microSD se han posicionado estratégicamente en el lateral correspondiente a la perforación circular de mayor diámetro donde se aloja el joystick.

El equipo garantiza una operación continua y segura mediante un circuito de gestión de baterías (BMS) integrado, y expone 18 pines educativos libres para la experimentación con electrónica externa. A lo largo de este documento se detalla la concepción arquitectónica del hardware, el desarrollo del firmware, la implementación de la interfaz de bloques y, finalmente, la validación del prototipo mediante una serie de prácticas experimentales de complejidad incremental, demostrando su viabilidad como herramienta de apoyo en el aprendizaje tecnológico.

II. Problema

En la actualidad, la enseñanza de sistemas embebidos representa un desafío para los entornos educativos debido a la complejidad de las herramientas tradicionales utilizadas para la programación de microcontroladores. Estas herramientas requieren configuraciones avanzadas, conocimiento previo en software especializado y una curva de aprendizaje pronunciada, lo cual representa una barrera significativa para los estudiantes que inician su formación en este campo[1]. Según un estudio realizado por Santimateo et al., más del 60 % de los estudiantes en cursos básicos de programación presentan dificultades para comprender los entornos de desarrollo y aplicar conceptos prácticos en sistemas embebidos [2].

Los métodos convencionales para enseñar programación embebida demandan entornos poco accesibles, tanto a nivel técnico como económico. El uso de plataformas comerciales suele requerir hardware costoso y conocimientos avanzados que no todos los estudiantes poseen, lo que limita el acceso equitativo a este tipo de formación [3]. Además, la falta de entornos simplificados y recursos pedagógicos adecuados genera una sobrecarga cognitiva en los primeros niveles del aprendizaje, dificultando la comprensión de conceptos básicos de electrónica y programación [4].

Estudios recientes han identificado que uno de los principales problemas en el proceso de enseñanza es la falta de herramientas integradas que conecten la teoría con la práctica. La ausencia de kits educativos personalizables y accesibles impide que los estudiantes puedan experimentar de forma autónoma y segura [5]. A esto se suma la escasa disponibilidad de plataformas diseñadas específicamente para el aula, con funciones orientadas a la enseñanza, como la rotulación intuitiva de pines, conexiones seguras o ausencia de kits que faciliten la interacción [6].

Por otro lado, muchos entornos académicos carecen de recursos físicos para el desarrollo de prácticas reales, por lo que se depende en exceso de simuladores o teorías que no logran representar las condiciones del mundo real [7]. Esta desconexión provoca un aprendizaje superficial, sin el desarrollo de habilidades prácticas esenciales en carreras de ingeniería como mecatrónica, electrónica o control automático [8].

En consecuencia, se pone en evidencia la carencia de soluciones didácticas integradas que combinen hardware educativo y software accesible, capaces de ofrecer al estudiante una experiencia interactiva sin requerir configuraciones complejas. Esta limitación impide que los estudiantes se involucren activamente con el proceso de programación embebida desde etapas tempranas, afectando el desarrollo de habilidades prácticas fundamentales [9].

Considerando los aspectos descritos, resulta evidente que la enseñanza de sistemas embebidos enfrenta múltiples barreras en entornos educativos: desde la complejidad técnica de las herramientas tradicionales y la falta de recursos físicos, hasta la escasa disponibilidad de plataformas accesibles y pedagógicamente adaptadas. Estas limitaciones dificultan el aprendizaje práctico, generan sobrecarga cognitiva en los estudiantes y restringen el acceso equitativo a la formación en programación embebida. La ausencia de kits integrados que conecten teoría y práctica limita el desarrollo de habilidades fundamentales, mientras que la falta de interacción física con el hardware reduce el compromiso y la retención del conocimiento. Por tanto, se evidencia la necesidad urgente de soluciones educativas que combinen hardware funcional y software intuitivo, capaces de facilitar el aprendizaje desde etapas tempranas y fomentar competencias técnicas esenciales.

III. JUSTIFICACIÓN

El desarrollo de dispositivos simplificados se justifica por la creciente demanda de herramientas pedagógicas que faciliten el acceso a tecnologías embebidas. Los enfoques tanto gráficos como textuales e incluso funcionales deben ser considerados cuidadosamente al determinar qué ambientes de programación implementar en cursos introductorios [1]. Un kit de desarrollo demuestra ser una solución efectiva para integrar experiencias reales con entornos virtuales, facilitando el aprendizaje sin importar la ubicación geográfica. Esta combinación mejora considerablemente la comprensión de conceptos abstractos [10]. Además, el uso de tecnologías de comunicación en entornos académicos ha permitido implementar prácticas que, de forma integrada, fortalecen el proceso formativo [11].

En el campo educativo, diferentes investigaciones han concluido que la programación de microcontroladores, especialmente en áreas como el control y la automatización, debe presentarse de manera simplificada. Esto permite a los estudiantes enfocarse en los conceptos más importantes, sin perderse en detalles técnicos [12]. Una solución efectiva ha sido el uso de dispositivos que, al combinar hardware y software en una misma plataforma, mejoran la accesibilidad al aprendizaje práctico [13].

Preparar a los estudiantes para los desafíos actuales exige integrar la teoría con la práctica de forma efectiva, especialmente en temas como los sistemas en tiempo real o los sistemas ciberfísicos [5]. Estas plataformas, al estar integradas y ser fáciles de usar, eliminan barreras técnicas y permiten concentrarse en el aprendizaje. Esto respalda la importancia de crear dispositivos autocontenidos que simplifiquen el entorno de desarrollo [14].

En un análisis más amplio, la evidencia científica destaca que los estudiantes mejoran sus competencias técnicas cuando trabajan con plataformas que reducen barreras de entrada, permitiéndoles enfocarse en el diseño y resolución de problemas [15]. Asimismo, el uso de microcontroladores como el ESP32 en carreras como ingeniería electrónica, mecatrónica o control industrial mejora significativamente los resultados académicos en materias prácticas. Estudios recientes concluyen que los entornos de aprendizaje que combinan herramientas accesibles, software educativo y placas electrónicas adaptadas generan ecosistemas que fomentan no solo la comprensión técnica, sino también el pensamiento crítico, la autonomía y la creatividad [1].

Por otro lado, existe una necesidad cada vez mayor de recursos que se adapten específicamente a los requerimientos del aula. En este sentido, se propone el diseño de una placa electrónica educativa basada en el ESP32, optimizada para la enseñanza. Esta placa electrónica incluirá conectores directos para sensores comunes, opciones de alimentación, pines rotulados según funcionalidad, y un sistema de protección integrado para evitar daños por mal uso.

Junto a esta placa, se utilizará un software de programación orientado al ámbito educativo. El software integrará ejemplos, prácticas guiadas y una simulación básica para validar el funcionamiento antes de cargar el programa al microcontrolador.

IV. OBJETIVOS

IV-A. Objetivo General

Desarrollar una consola de codificación simplificada utilizando un ESP32, que incorpore una interfaz gráfica para la enseñanza de la programación orientada a objetos.

IV-B. Objetivos Específicos

- Diseñar una consola que integre una placa electrónica con sus componentes esenciales en una base física impresa en 3D, permitiendo la interacción mediante pantalla, botones y joystick.

- Desarrollar un entorno de programación simplificada utilizando un ESP32, mediante menús interactivos, facilitando la ejecución y visualización de programas orientados a objetos sin necesidad de un computador.

- Evaluar el desempeño del dispositivo en entornos educativos mediante 5 prácticas de laboratorio que validen su funcionamiento.

V. Marco Teórico

V-A. Fundamentos pedagógicos para la enseñanza de la programación embebidaa

La enseñanza de sistemas tecnológicos complejos requiere un marco teórico sólido que valide la transición desde la instrucción tradicional hacia modelos participativos. A continuación, se detallan los paradigmas educativos que sustentan el diseño de herramientas físicas para el aprendizaje de la ingeniería.

V-A1. Constructivismo y construccionismo en la educación en ingeniería: El marco epistemológico contemporáneo para la enseñanza de la tecnología se basa en la distinción crítica entre el constructivismo de Jean Piaget y el construccionismo de Seymour Papert [16]. Mientras que la teoría de Piaget postula que el conocimiento se construye en la mente del estudiante a través de la asimilación y acomodación de experiencias internas, Papert expande esta teoría argumentando que dicha construcción mental es significativamente más eficaz cuando el aprendiz está involucrado en la creación de un artefacto público o un objeto tangible fuera de su mente.

En el contexto específico de la ingeniería mecatrónica, el construccionismo sugiere que el aprendizaje de conceptos abstractos, como la estructura de clases en programación, se facilita cuando el código no permanece como una entidad lógica invisible en un ordenador, sino que se manifiesta en el comportamiento observable de un dispositivo físico. El hardware actúa, por tanto, como el medio de expresión donde el estudiante exterioriza y valida su comprensión teórica, proceso que se ilustra en las fases de la Figura 1.

Figura 1. Fases del ciclo de aprendizaje experiencial aplicado a la ingeniería

Bajo esta perspectiva, la consola propuesta no es un simple periférico, sino lo que Papert denomina un objeto para pensar (*object-to-think-with*). La externalización del conocimiento permite que el error deje de ser una abstracción cognitiva para convertirse en un fenómeno físico observable. Cuando el estudiante detecta una falla en el hardware, se produce un çonflicto cognitivoïnmediato que le obliga a depurar no solo su código, sino sus modelos mentales sobre la lógica del sistema. Las divergencias principales entre ambos enfoques se resumen en la Tabla I.

Tabla I
DIVERGENCIAS CLAVE ENTRE LA ASIMILACIÓN INTERNA Y LA CONSTRUCCIÓN EXTERNA.

Aspecto	Constructivismo (Piaget)	Construccionismo (Papert)
Foco del Aprendizaje	Procesos mentales internos (esquemas).	Creación de artefactos externos tangibles.
Rol del Objeto	Material de apoyo secundario.	Centro de la actividad intelectual.
Dinámica Social	Individual / Cognitiva.	Pública / Compartible (el objeto se muestra a otros).

V-A2. Aprendizaje activo y aprendizaje basado en proyectos (ABP): El Aprendizaje Activo se define como cualquier método instruccional que involucra a los estudiantes en el proceso de aprendizaje a través de actividades significativas y reflexión, en contraposición a la recepción pasiva de información típica de la clase magistral. Dentro de este paradigma, el Aprendizaje Basado en Proyectos (ABP) se presenta como una estrategia didáctica donde los alumnos adquieren competencias clave mediante la elaboración de soluciones técnicas que dan respuesta a problemas complejos.

La comunidad científica evidencia que el ABP en sistemas embebidos fomenta la retención de conocimientos a largo plazo [17]. Al enfrentarse al desafío de diseñar o programar un sistema funcional, el estudiante se ve obligado a integrar conocimientos multidisciplinarios (electrónica, lógica matemática, diseño), transformando la teoría fragmentada en una competencia técnica integral y aplicable a la industria.

Asimismo, el ABP es fundamental para la construcción de la identidad profesional del ingeniero. Al transitar de ejercicios académicos cerrados a proyectos de diseño abiertos y mal definidos, el alumno abandona el rol de consumidor de tecnología para asumir el de creador de soluciones. Este cambio de paradigma fomenta la responsabilidad ética sobre la fiabilidad del producto final y entrena la tolerancia a la frustración, una competencia blanda (*soft skill*) crítica en el entorno laboral real. La Tabla II detalla las diferencias estructurales de este enfoque frente al tradicional.

Tabla II
DIFERENCIAS ESTRUCTURALES ENTRE LA INSTRUCCIÓN TRADICIONAL Y EL APRENDIZAJE ACTIVO EN INGENIERÍA.

Dimension	**Enseñanza Tradicional**	**Aprendizaje Activo (ABP)**
Rol del Estudiante	Receptor pasivo de información	Creador y solucionador de problemas
Enfoque	Memorización de sintaxis	Comprensión de la lógica del sistema
Evaluación	Exámenes teóricos escritos	Funcionamiento del prototipo final
Error	Se penaliza como fracaso	Se utiliza como fuente de depuración

V-A3. Teoría de la carga cognitiva aplicada a entornos educativos tecnológicos: Desarrollada por John Sweller, la Teoría de la Carga Cognitiva (TCC) establece que la memoria de trabajo humana tiene una capacidad limitada para procesar nueva información simultánea [18]. En la enseñanza de la programación de hardware, la carga cognitiva total se divide en tres categorías, como se ilustra en la Figura 2:

- **Carga Intrínseca:** La dificultad natural inherente al tema (ej. la complejidad de C++).
- **Carga Pertinente:** El esfuerzo mental dedicado a procesar y aprender el contenido.
- **Carga Extraña:** El esfuerzo desperdiciado en elementos irrelevantes o mal diseñados del entorno de aprendizaje.

En entornos de simulación puramente virtuales, la carga extraña suele ser elevada debido a la necesidad de gestionar interfaces de software complejas, configurar entornos de compilación y abstraer el funcionamiento de periféricos que no están presentes físicamente. Teóricamente, el uso de herramientas físicas dedicadas reduce esta carga extraña al eliminar la fricción del entorno virtual, permitiendo que la capacidad cognitiva limitada del estudiante se redistribuya hacia la carga pertinente: la lógica algorítmica y la arquitectura del software.

Para mitigar esta sobrecarga, el diseño del hardware educativo actúa como un .ªndamiaje Instruccional"(*Instructional Scaffolding*), un concepto derivado de la teoría sociocultural de Vygotsky. La consola y su capa de abstracción de hardware (HAL) mantienen al estudiante dentro de su Zona de Desarrollo Próximo (ZDP), proporcionando el soporte necesario para abordar problemas complejos sin caer en la ansiedad. Este andamiaje es temporal: a medida que el estudiante domina la lógica de alto nivel, puede "desmontar"las ayudas y descender progresivamente hacia la programación de registros, habiendo consolidado primero los conceptos fundamentales.

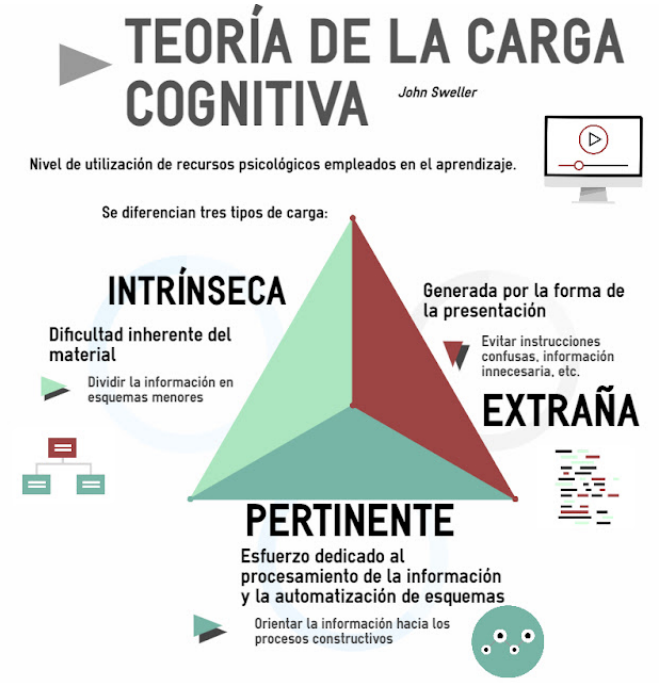


Figura 2. Componentes de la Teoría de la Carga Cognitiva y su distribución.

V-A4. Aprendizaje mediante interacción física (Learning by Doing): El concepto de .ªprender haciendo"(*learning by doing*), popularizado por John Dewey, sostiene que la educación técnica debe basarse en la experiencia directa y la interacción sensorial con el entorno. En la ingeniería electrónica, esto se traduce en la necesidad imperativa de manipular componentes reales para comprender sus limitaciones físicas, latencias y comportamientos no lineales, aspectos que la simulación idealizada por ordenador no logra replicar con fidelidad.

La interacción física proporciona un ciclo de retroalimentación sensorial inmediato y multimodal. Cuando un estudiante programa una rutina de control y observa la respuesta mecánica o lumínica del hardware en tiempo real, se activan canales de aprendizaje kinestésicos y visuales simultáneamente. Esta validación tangible permite identificar errores lógicos de manera intuitiva, reforzando la conexión sináptica entre el código escrito (causa) y la acción del hardware (efecto).

Esta inmediatez en la respuesta del sistema no puede ser replicada fielmente por entornos virtuales. La simulación informática tiende a idealizar los componentes, eliminando el ruido y las tolerancias físicas que son inherentes a la ingeniería práctica. El uso de hardware real reintroduce estas variables estocásticas, obligando al estudiante a considerar factores de robustez y fiabilidad. Una comparativa pedagógica detallada entre ambos entornos se presenta en la Tabla III.

Tabla III
COMPARATIVA PEDAGÓGICA ENTRE ENTORNOS SIMULADOS Y PLATAFORMAS DE HARDWARE FÍSICO.

Característica	**Simulación Virtual (Proteus/Tinkercad)**	**Hardware Físico (Consola Propuesta)**
Naturaleza del Error	Errores de sintaxis o lógica pura.	Errores lógicos + Errores físicos (ruido, mala conexión).
Feedback	Visual y aséptico (mensajes en pantalla).	Multisensory (Vibración, calor, luz, sonido real).
Gestión del Tiempo	Tiempo simulado (puede ralentizarse).	Tiempo Real estricto (Hard Real-Time).
Credibilidad	Percibido como ”videojuego.º teoría.	Percibido como ïngeniería aplicada”.

V-A5. Educación STEM/STEAM y su relación con los sistemas embebidos: La educación STEM (Ciencia, Tecnología, Ingeniería y Matemáticas) busca la integración interdisciplinaria de estas áreas para la resolución de problemas complejos. La incorporación más reciente de las Artes (STEAM) añade el componente de diseño, estética y creatividad, elementos que son cruciales en la ingeniería de producto moderna y en el desarrollo de interfaces hombre-máquina amigables.

Los sistemas embebidos y las consolas de desarrollo representan la cristalización del enfoque STEAM: requieren matemáticas para la lógica algorítmica, ingeniería para el diseño del hardware, tecnología para la implementación del software y principios científicos para el funcionamiento de los sensores. El uso de estos dispositivos en el aula alinea la formación académica con este enfoque integrador, obligando al estudiante a dejar de percibir estas disciplinas como silos aislados y a utilizarlas como herramientas interconectadas para la creación tecnológica.

La integración de estas disciplinas a través de sistemas embebidos cultiva el Pensamiento Computacional (*Computational Thinking*). Más allá de la simple codificación, el estudiante entrena habilidades cognitivas transversales como la descomposición de problemas grandes en tareas manejables (ej. separar la lógica del juego de la lectura de sensores), el reconocimiento de patrones en señales digitales y la abstracción algorítmica. Estas competencias permiten al futuro ingeniero modelar sistemas complejos y optimizar recursos limitados, habilidades transferibles a cualquier rama de la tecnología moderna.

V-B. Plataformas y módulos educativos para la enseñanza de sistemas embebidos

El mercado de la educación tecnológica ha evolucionado desde el uso de placas de evaluación industrial complejas hacia ecosistemas diseñados específicamente para el aprendizaje. A continuación, se analiza el estado del arte de las plataformas existentes, evaluando sus ventajas y limitaciones para la enseñanza de ingeniería a nivel universitario.

V-B1. Plataformas educativas tradicionales para programación embebida: Históricamente, la enseñanza de microcontroladores se basaba en el uso de arquitecturas de 8 bits (como la familia PIC o 8051) programadas en lenguaje ensamblador. Si bien este enfoque proporcionaba un control total sobre el hardware, la elevada curva de aprendizaje generaba una frustración temprana en los estudiantes, desviando la atención de la lógica de control hacia la memorización de nemotécnicos y direcciones de memoria. La dicotomía entre este enfoque de bajo nivel y las tendencias modernas se evidencia en la Tabla IV.

En la última década, ha surgido el concepto de ”hardware de umbral bajo y techo alto”(*Low Floor, High Ceiling*). Estas plataformas buscan eliminar las barreras de entrada iniciales (como la configuración compleja de fusibles o programadores externos) sin limitar la capacidad del estudiante para desarrollar proyectos complejos a medida que avanza su formación. La tendencia actual favorece el uso de placas de desarrollo que integran la interfaz de programación vía USB directamente en el PCB, facilitando la iteración rápida de código en el aula.

V-B2. Arduino como herramienta educativa estándar: La plataforma Arduino, cuya placa se muestra en la Figura 3, representa el estándar de facto en la educación de electrónica básica. Su éxito radica en la abstracción de la complejidad del hardware a través de un framework de software (Wiring) que estandariza las funciones de entrada/salida. Desde una perspectiva pedagógica, Arduino democratizó el acceso a los microcontroladores al simplificar la sintaxis de C++ y ofrecer una inmensa comunidad de soporte open-source.

Sin embargo, en el contexto de la ingeniería mecatrónica avanzada, Arduino presenta limitaciones. Su entorno de desarrollo (IDE) original oculta deliberadamente conceptos profesionales como la manipulación directa de registros, las interrupciones anidadas o la gestión de memoria dinámica. Esto puede generar una ”falsa competencia.[en] el estudiante, quien logra hacer funcionar un sistema uniendo librerías prefabricadas sin comprender realmente la arquitectura subyacente del procesador o los principios de la Programación Orientada a Objetos.

Sin embargo, el éxito de Arduino es también su mayor debilidad en la ingeniería formal: fomenta el ”Síndrome de la Caja Negra”. La abstracción excesiva de su API (ej. `digitalWrite`) oculta la gestión de puertos y temporizadores, creando una falsa sensación de competencia. Además, su estructura basada en un ”Super-Bucle”(`void loop`) dificulta la enseñanza de sistemas operativos en tiempo real (RTOS), limitando al estudiante a la programación secuencial básica.

Tabla IV
DICOTOMÍA ENTRE EL ENFOQUE DE BAJO NIVEL Y EL ENFOQUE MAKER EN LA EDUCACIÓN.

Dimensión	Enfoque Tradicional (PIC/8051)	Ecosistema Arduino (AVR)
Curva de Aprendizaje	Muy pronunciada (Frustración inicial).	Muy suave (Satisfacción inmediata).
Control del Hardware	Total (Acceso a bit).	Limitado (Oculto por API).
Riesgo Académico	Abandono por complejidad.	Superficialidad técnica (Çopy-Paste").

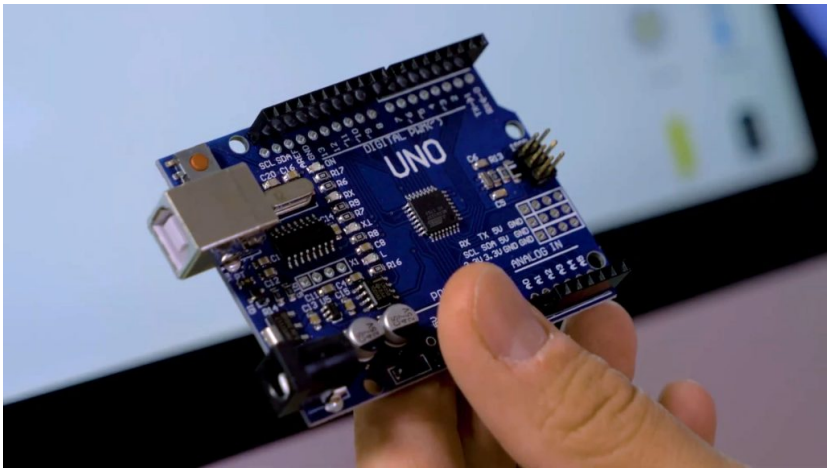


Figura 3. Placa de desarrollo Arduino Uno.

V-B3. Micro:bit y plataformas educativas de bajo umbral: Aunque Micro:bit es excepcionalmente eficaz para la educación STEM escolar bajo la filosofía de "pisos bajos y techos altos", resulta insuficiente para la ingeniería mecatrónica superior. Su dependencia de entornos de bloques gráficos o Python simplificado limita el acceso al control directo de memoria y punteros, competencias críticas en la industria. El "techo"de la plataforma es demasiado bajo para un ingeniero que necesita diseñar drivers o implementar protocolos de comunicación de alta velocidad.

Diseñada originalmente por la BBC para la educación escolar en el Reino Unido, la placa Micro:bit, presentada en la Figura 4, se ha posicionado como una herramienta de iniciación excelente. Su enfoque se basa en la programación por bloques y en la integración de sensores (brújula, acelerómetro, matriz LED) en una sola tarjeta, eliminando la necesidad de cableado externo que suele ser fuente de errores en etapas tempranas.

A pesar de su utilidad en educación secundaria, su aplicación en ingeniería es limitada. El entorno de programación por bloques abstrae tanto el código que impide el aprendizaje de sintaxis profesional y estructuras de datos complejas. Además, la potencia de procesamiento de su microcontrolador nRF51/52 es insuficiente para ejecutar algoritmos de control en tiempo real o interfaces gráficas avanzadas, restringiendo su uso a demostraciones lógicas básicas.

Figura 4. Vista frontal y posterior de la placa de desarrollo educativo BBC micro:bit.

V-B4. LEGO Mindstorms y plataformas robóticas educativas: Desde la perspectiva electrónica, estas plataformas constituyen ”Jardines Vallados”(*Walled Gardens*). Al utilizar hardware cerrado y conectores propietarios, impiden que el estudiante aprenda a interconectar componentes reales (leer hojas de datos, calcular etapas de potencia). Esto produce profesionales capaces de programar la lógica cinemática de un robot, pero incapaces de diseñar el hardware que lo hace moverse, generando una brecha de competencias en el diseño de producto.

LEGO Mindstorms (EV3 y Robot Inventor), cuyos ensambles típicos se aprecian en la Figura 5, representa el enfoque del construccionismo aplicado a la robótica modular. Su principal fortaleza es la integración mecánica-electrónica, permitiendo construir estructuras físicas rápidas sin herramientas. El sistema utiliza conectores propietarios (RJ12 modificados) y un firmware cerrado que facilita la conexión ”Plug and Play”de motores y sensores.

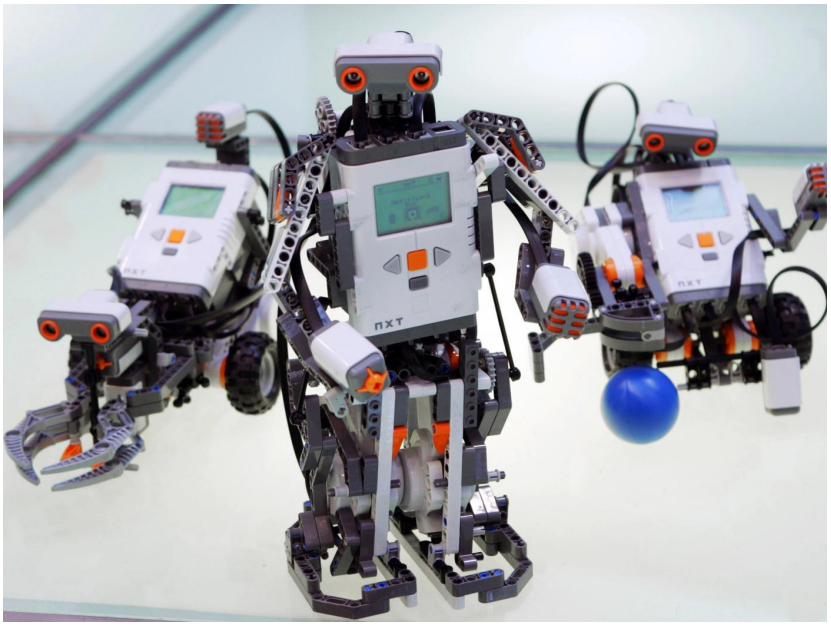


Figura 5. Robots construidos con el kit educativo LEGO Mindstorms NXT.

La principal desventaja de este ecosistema en la educación superior es su carácter de çaja negraz su elevado coste. Al ser un sistema propietario, impide que el estudiante acceda a los esquemáticos electrónicos o modifique

el firmware a bajo nivel. Esto limita la comprensión de la electrónica analógica y digital, ya que el estudiante opera como un usuario final del sistema en lugar de como un diseñador de sistemas embebidos.

V-B5. Características de Kits educativos comerciales: El uso de kits de componentes sueltos en protoboard introduce una fricción instrumental significativa. Estudios de laboratorio indican que hasta un 40 % del tiempo lectivo se pierde diagnosticando fallos de conexión (cable spaghetti) y falsos contactos, en lugar de depurar software. Esta inestabilidad física frustra el aprendizaje lógico y dificulta la portabilidad de los proyectos fuera del laboratorio.

El análisis de la oferta comercial actual, ejemplificada por ecosistemas como Grove de Seeed Studio o los módulos de sensores Keyes, revela una tendencia hacia la estandarización del hardware educativo. Estos sistemas se caracterizan por una modularidad extrema, donde los componentes electrónicos se aíslan en placas de circuito impreso individuales (*breakout boards*) diseñadas para facilitar su manipulación.

Esta arquitectura prioriza la conectividad simplificada mediante el uso de cables unificados y conectores polarizados, eliminando eficazmente los errores de conexión comunes en protoboards. Sin embargo, esta ventaja física suele venir acompañada de una fuerte abstracción de software, obligando al estudiante a depender de librerías propietarias que ocultan la complejidad de la comunicación con el sensor. Si bien esto acelera la implementación inicial, pedagógicamente genera una çaja negra"que impide al alumno comprender los protocolos de comunicación subyacentes necesarios para el diseño profesional de sistemas embebidos.

Si bien estos kits aceleran el prototipado, a menudo fragmentan la experiencia de aprendizaje. El estudiante conecta módulos dispersos mediante cables dupont, lo que resulta en montajes frágiles y poco ergonómicos que dificultan la portabilidad y la interacción continua necesaria para la validación de algoritmos complejos. Una comparativa técnica y pedagógica de las plataformas descritas se consolida en la Tabla V.

Tabla V
COMPARATIVA TÉCNICA Y PEDAGÓGICA DE LAS PLATAFORMAS EDUCATIVAS PREDOMINANTES.

Característica	**Arduino Uno**	**Micro:bit**	**LEGO EV3**	**Placas Industriales (STM32/ESP32)**
Costo	Bajo	Muy Bajo	Muy Alto	Bajo/Medio
Curva de Aprendizaje	Media	Muy Baja	Baja	Alta
Lenguaje Principal	C++ Simplificado	Bloques / Python	Bloques	C / C++ Estándar
Enfoque Pedagógico	Electrónica Gral.	Lógica Básica	Robótica	Ingeniería Profesional
Limitación Principal	Poca potencia	Abstracción excesiva	Sistema Cerrado	Complejidad de configuración

V-B6. Resultados obtenidos usando módulos educativos comerciales: La revisión bibliográfica sobre el impacto de estos módulos comerciales indica resultados mixtos. Estudios recientes señalan que el uso de kits modulares aumenta significativamente la motivación inicial y la tasa de finalización de proyectos en comparación con la enseñanza teórica tradicional.

Sin embargo, investigaciones complementarias advierten sobre el fenómeno de la "dependencia de la librería". Los estudiantes formados exclusivamente con kits de alto nivel a menudo fallan al intentar migrar a entornos

profesionales donde deben escribir sus propios controladores (drivers) o diseñar sus propios circuitos. Esto sugiere la existencia de un vacío pedagógico: se requiere una herramienta que ofrezca la portabilidad y ergonomía de una consola comercial, pero que obligue al estudiante a enfrentar la complejidad del código C++ real, sin las distracciones del cableado inestable.

Frente a esto, el uso de consolas de desarrollo con hardware integrado. elimina la variable del error de montaje. Al partir de una base estable, el estudiante puede enfocar el 100 % de su atención cognitiva en la arquitectura del software y el uso de RTOS. La Tabla VI resume la ventaja estratégica de la propuesta frente a las alternativas comerciales:

Tabla VI
MATRIZ COMPARATIVA DE LA PROPUESTA DE TESIS FRENTE A SOLUCIONES EXISTENTES.

Plataforma	**Robustez Física**	**Profundidad Ingeniería**	**Flexibilidad**	**Costo**
Kit Protoboard	Baja (Cables sueltos)	Media	Alta	Bajo
LEGO Mindstorms	Alta	Baja (Caja negra)	Baja	Muy Alto
Micro:bit	Media	Baja (Bloques)	Media	Bajo
Consola Propuesta	Alta (PCB Integrado)	Alta (C++/RTOS)	Alta (Open Source)	Bajo

V-C. Arquitectura de software para sistemas embebidos educativos

El diseño de software para sistemas embebidos difiere sustancialmente del desarrollo de aplicaciones de escritorio debido a las restricciones de hardware y la necesidad de determinismo temporal. A continuación, se fundamentan los principios arquitectónicos que permiten transformar un microcontrolador en una plataforma educativa robusta.

V-C1. Programación en sistemas embebidos: La programación de firmware profesional exige una comprensión profunda del ciclo de compilación cruzada (*cross-compilation*). A diferencia del software de escritorio, aquí el ingeniero debe gestionar manualmente la ubicación del código mediante el *Linker Script* (.ld). Este archivo es crítico, pues instruye al enlazador sobre cómo mapear las secciones lógicas del programa (.text, .data) en las direcciones físicas específicas de la memoria Flash y SRAM del ESP32-S3 [19], proceso esquematizado en la Figura 6. Un error en este mapeo provoca excepciones fatales, ya que el procesador podría intentar ejecutar datos como si fueran instrucciones.

La programación de sistemas embebidos se define como el proceso de creación de software diseñado para ejecutarse en dispositivos con recursos limitados de procesamiento, memoria y energía. A diferencia del software de propósito general, este código debe interactuar directamente con el hardware subyacente, gestionando registros de memoria y señales eléctricas en tiempo real.

Desde una perspectiva académica, este paradigma introduce el concepto de çompilación cruzada"(*cross-compilation*), donde el código se escribe y compila en una arquitectura potente (PC host x86/x64) para generar un binario ejecutable en una arquitectura destino restringida (ARM o Xtensa). Este flujo obliga al ingeniero a tener una comprensión profunda de la arquitectura del procesador destino, ya que errores en el manejo de punteros o

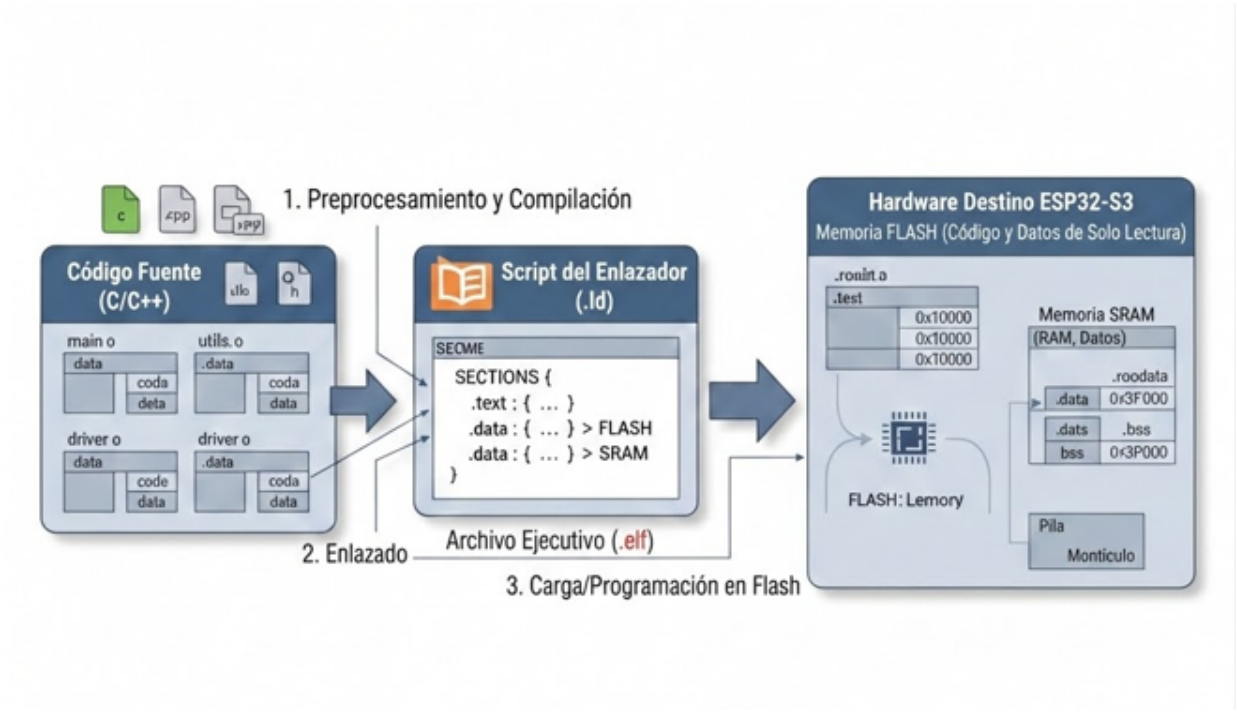


Figura 6. Flujo de la cadena de herramientas (Toolchain): Del código C++ al binario mapeado en memoria.

desbordamientos de pila que serían triviales en un PC, pueden resultar en fallos catastróficos o *hard faults* en el sistema embebido.

V-C2. Lenguaje C++ aplicado a sistemas embebidos: Se adopta el estándar C++17 para aprovechar el paradigma RAII (*Resource Acquisition Is Initialization*). Esta técnica vincula el ciclo de vida de los periféricos de hardware al ciclo de vida de los objetos de software: el constructor de la clase configura los registros del periférico y el destructor los deshabilita [20]. Esto garantiza que recursos críticos, como los temporizadores del PWM o los canales DMA, nunca queden en estados indeterminados o ”zombis”si una parte del código falla, aumentando la robustez del sistema educativo frente a errores del estudiante.

Aunque el lenguaje C ha sido históricamente el estándar en la industria, la complejidad de los sistemas modernos ha impulsado la adopción de C++. En el ámbito educativo, C++ ofrece ventajas significativas gracias a sus .ªbstracciones de costo cero”. Esto significa que características avanzadas como las clases o los *templates* no generan una sobrecarga adicional en el código máquina final si se utilizan correctamente, manteniendo la eficiencia del C pero mejorando la organización lógica [21].

El uso de C++ permite implementar tipos de datos fuertes y encapsulamiento, protegiendo al estudiante de errores comunes como la modificación accidental de registros de configuración. Además, la capacidad de sobrecarga de operadores facilita la creación de sintaxis intuitiva para el control de hardware, permitiendo que operaciones complejas (como sumar vectores de movimiento) se expresen de manera natural en el código.

V-C3. Programación orientada a objetos en microcontroladores: La implementación del polimorfismo en microcontroladores se sustenta técnicamente en la Tabla de Funciones Virtuales (*vtable*), cuyo funcionamiento se detalla en la Figura 7. Cuando el sistema trata un objeto `Joystick` como si fuera un `Sensor` genérico, el procesador consulta esta tabla en memoria RAM para resolver dinámicamente a qué dirección de memoria saltar para ejecutar el método `leer()`. Aunque esto introduce una ligera sobrecarga de ciclos de reloj (overhead), el beneficio arquitectónico es inmenso: permite añadir nuevos módulos de hardware al sistema sin necesidad de recompilar el núcleo del firmware, facilitando la escalabilidad del proyecto.

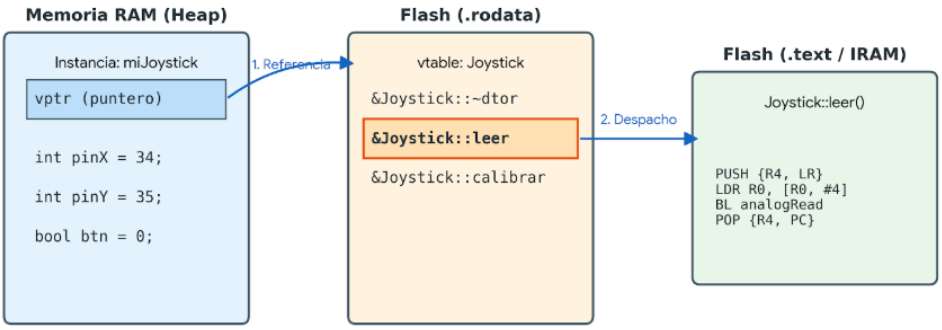


Figura 7. Mecanismo de despacho dinámico mediante VTable en la memoria del ESP32-S3.

La Programación Orientada a Objetos (POO) proporciona un modelo mental que se alinea naturalmente con la física de los sistemas mecatrónicos. En este paradigma, cada componente físico (un sensor, una pantalla, un motor) se modela como un .ºbjeto"de software que encapsula sus propios datos (atributos) y comportamientos (métodos).

Este enfoque es pedagógicamente superior a la programación estructurada tradicional para proyectos complejos. Mediante el uso de la Herencia, se pueden crear jerarquías lógicas (por ejemplo, una clase base ‘Sensor‘ de la cual heredan ‘SensorTemperatura‘ y ‘SensorDistancia‘), y mediante el Polimorfismo, el sistema puede tratar a todos estos sensores de manera uniforme. Esto enseña al estudiante a diseñar sistemas modulares y escalables, donde añadir un nuevo componente no requiere reescribir todo el código base, estructura visualizada en el diagrama UML de la Figura 8.

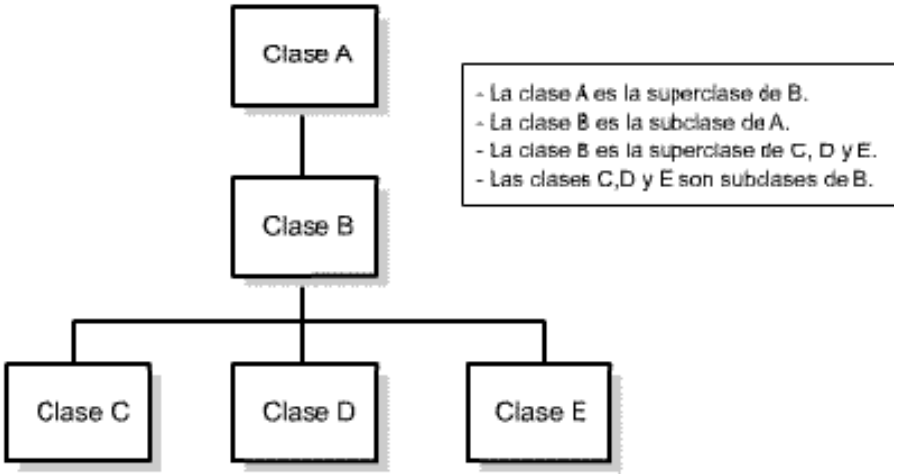


Figura 8. Diagrama de clases UML: Representación de la jerarquía y herencia en controladores de hardware.

V-C4. Abstracción de hardware (HAL) como estrategia pedagógica: La Capa de Abstracción de Hardware (HAL, por sus siglas en inglés) es una interfaz de software que oculta las especificidades del hardware físico a la lógica de la aplicación. En un contexto educativo, la implementación de una HAL es estratégica: permite que el estudiante se enfoque en la lógica del algoritmo (el ”qué”hace el sistema) sin perderse prematuramente en los detalles de bajo nivel (el cómo”se configuran los registros eléctricos), como se demuestra en la arquitectura por capas de la Figura 9.

Una HAL bien diseñada actúa como un traductor. Cuando el estudiante invoca una función de alto nivel como por ejemplo `pantalla.dibujarCirculo()`, la HAL se encarga de traducir esa orden en las múltiples instrucciones SPI y comandos de memoria necesarios para el controlador de video. Esta separación de responsabilidades no solo simplifica el aprendizaje inicial, sino que también inculca buenas prácticas de ingeniería de software, promoviendo la portabilidad del código entre diferentes microcontroladores.

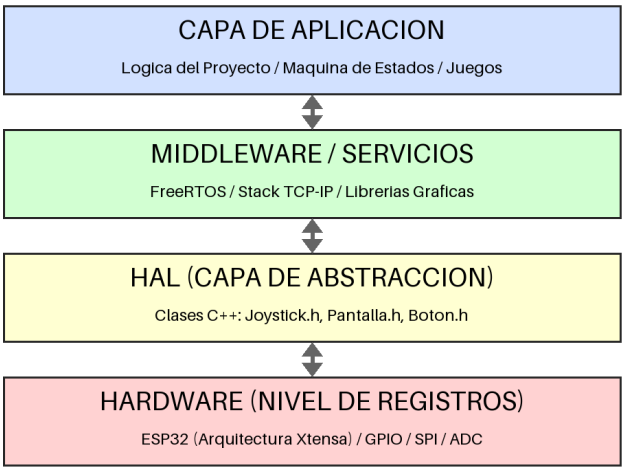


Figura 9. Arquitectura de software por capas: Desacoplamiento entre Hardware y Aplicación mediante HAL.

V-C5. Gestión de memoria y recursos en sistemas embebidos: La gestión eficiente de los recursos de memoria constituye una competencia crítica en la ingeniería electrónica moderna, diferenciando el desarrollo de firmware profesional del software de alto nivel. A diferencia de los ordenadores de propósito general, que utilizan mecanismos de memoria virtual y paginación en disco para .exªtender"la RAM, un microcontrolador como el ESP32 opera con memoria física estricta. Si esta se agota o se fragmenta, el sistema no se ralentiza, sino que colapsa (*Crash*).
En la arquitectura de memoria del ESP32, la RAM no es un bloque monolítico, sino que se segmenta en regiones funcionales. Además de las secciones estáticas conocidas como .data (variables globales inicializadas) y .bss (variables globales no inicializadas), existen dos áreas dinámicas críticas para la estabilidad del sistema:

IV-C5.1 El Stack (Pila): Determinismo y Velocidad: El Stack es una región de memoria contigua que opera bajo una estructura LIFO (*Last In, First Out*). Su gestión es totalmente automática y realizada por la CPU mediante el registro "Puntero de Pila"(*Stack Pointer*). Cada vez que se llama a una función, se crea un "marco de pila"(*stack frame*) que almacena las variables locales, los argumentos de la función y la dirección de retorno.

Debido a esta naturaleza mecánica, el acceso al Stack es extremadamente rápido y determinista. Sin embargo, su tamaño es fijo y limitado (típicamente unos pocos Kilobytes por tarea en FreeRTOS). El riesgo pedagógico aquí es el *Stack Overflow*: si el estudiante implementa una recursividad infinita o declara arreglos locales masivos, el Stack crecerá hasta invadir otras secciones de memoria, provocando una excepción fatal inmediata [22].

IV-C5.2 El Heap (Montículo): Flexibilidad y Riesgo: El Heap es una zona de memoria desestructurada destinada a la asignación dinámica en tiempo de ejecución (mediante operadores como `new` o `malloc`). Es indispensable en C++ para instanciar objetos cuyo tamaño o cantidad no se conocen al momento de compilar (como una lista de enemigos en un juego o un buffer de recepción WiFi).

A diferencia del Stack, el Heap no es determinista: buscar un bloque de memoria libre toma tiempo de CPU variable. El mayor desafío técnico es la **fragmentación**. Con el uso continuo, el Heap se llena de "huecos"de memoria libre que son demasiado pequeños para ser útiles. Además, a diferencia de lengumos como Java o Python, C++ embebido no posee Recolector de Basura (*Garbage Collector*), por lo que si el estudiante olvida liberar la memoria (`delete`), se produce una "Fuga de Memoria"(*Memory Leak*) que eventualmente bloqueará el dispositivo. Las diferencias entre ambas regiones se comparan en la Tabla VII.

Tabla VII
COMPARATIVA TÉCNICA ENTRE LAS REGIONES DE MEMORIA STACK Y HEAP.

Característica	**Stack (Pila)**	**Heap (Montículo)**
Gestión	Automática (CPU/Compilador)	Manual (Programador: new/delete)
Acceso	Determinista y secuencial (LIFO)	Aleatorio (Búsqueda de huecos)
Velocidad	Muy alta (instrucciones simples)	Lenta (overhead de gestión)
Problema Típico	Stack Overflow (Desbordamiento)	Fragmentación y Memory Leaks
Uso Principal	Variables locales, control de flujo	Objetos dinámicos, buffers, nodos

V-C6. Sistemas operativos en tiempo real (FreeRTOS): El núcleo del funcionamiento del dispositivo es el Planificador (*Scheduler*) preventivo de FreeRTOS. Este algoritmo garantiza el determinismo temporal mediante el cambio de contexto (*Context Switch*): cada 1 ms (Tick), el procesador guarda instantáneamente el estado de los registros de la tarea actual y carga los de la siguiente tarea prioritaria [23]. Esto permite que procesos críticos, como la lectura de sensores de seguridad, interrumpan inmediatamente a procesos lentos, como el renderizado de gráficos, garantizando una respuesta en tiempo real que un super-bucle simple no puede ofrecer.

A medida que la complejidad del software aumenta, el modelo de ”superbucle”(*super-loop*) se vuelve insuficiente. Aquí entra el Sistema Operativo en Tiempo Real (RTOS). A diferencia de los sistemas operativos de escritorio (como Windows), un RTOS como FreeRTOS no busca maximizar el rendimiento promedio, sino garantizar el determinismo: asegurar que las tareas críticas se ejecuten dentro de un plazo de tiempo predecible y estricto.

FreeRTOS introduce el concepto de *Scheduler* (planificador), un algoritmo que decide qué tarea debe ejecutarse en el procesador en cada instante basándose en prioridades. Esto permite que el sistema ïnterrumpaüna tarea de baja prioridad (como actualizar una gráfica) para atender inmediatamente una tarea crítica (como leer un sensor de fin de carrera), simulando paralelismo en un solo núcleo o gestionando carga real en núcleos múltiples.

V-C7. Multitarea y su aporte al proceso de aprendizaje: La transición de la programación secuencial a la programación concurrente (multitarea) representa un hito en la formación del ingeniero. Conceptualmente, permite al estudiante modelar sistemas que realizan múltiples acciones ”simultáneas”, como reproducir música de fondo mientras se procesa la física de un videojuego.

Desde el punto de vista del aprendizaje, implementar multitarea obliga al estudiante a comprender mecanismos de sincronización como semáforos, colas de mensajes y exclusión mutua (*Mutex*). Esto elimina el vicio de utilizar funciones bloqueantes como `delay()`, promoviendo un estilo de programación reactivo y eficiente donde el procesador nunca permanece ocioso esperando eventos, sino que conmuta inteligentemente entre tareas activas.

La concurrencia introduce el riesgo de Çondiciones de Carrera”, donde dos procesos corrompen la memoria al escribir simultáneamente en ella. Para mitigar esto, se implementan primitivas de sincronización estrictas, detalladas en la Tabla VIII:

Tabla VIII
MECANISMOS DE SINCRONIZACIÓN EN FREERTOS PARA GARANTIZAR LA INTEGRIDAD DE DATOS.

Mecanismo	**Función Técnica**	**Aplicación en la Consola**
Mutex	Bloqueo exclusivo de recursos.	Evita que el Juego y el Menú escriban en la pantalla a la vez.
Semáforo	Señalización de eventos.	Sincroniza la interrupción del botón con la tarea lógica.
Cola (Queue)	paso de mensajes FIFO.	Transporta datos del sensor entre núcleos sin variables globales.

V-D. Arquitectura de hardware e interfaces de interacción

El diseño físico de una herramienta educativa contemporánea exige una selección de componentes que no solo resuelvan la funcionalidad inmediata, sino que permitan la implementación de conceptos avanzados de ingeniería,

como el procesamiento digital de señales o los sistemas operativos en tiempo real. En esta sección se fundamenta la selección de la arquitectura de procesamiento y los periféricos que constituyen la interfaz hombre-máquina (HMI).

V-D1. Microcontroladores para aplicaciones educativas: La elección del microcontrolador define el límite funcional del aprendizaje. Mientras que en la educación básica predominan las arquitecturas de 8 bits debido a su simplicidad, estas carecen de la potencia necesaria para ejecutar conceptos modernos como el multihilo o la conectividad IoT segura. Por ello, este proyecto se fundamenta en la migración hacia arquitecturas de 32 bits, permitiendo al estudiante interactuar con un hardware que refleja los estándares industriales actuales, donde la gestión de ciclos de reloj, el acceso directo a memoria (DMA) y la arquitectura de memoria plana son competencias esenciales.

V-D2. Arquitectura y características del SoC ESP32-S3: El núcleo del sistema es el SoC (*System on Chip*) ESP32-S3. La selección de esta variante específica, superior al ESP32 clásico, responde a su arquitectura basada en dos núcleos Xtensa® LX7 de 32 bits capaces de operar hasta 240 MHz [24]. Esta actualización tecnológica es crítica para el proyecto, ya que los núcleos LX7 incluyen instrucciones vectoriales extendidas que aceleran el procesamiento de redes neuronales y algoritmos de señal, dotando a la consola de una proyección pedagógica hacia la Inteligencia Artificial de las Cosas (AIoT).

Adicionalmente, el ESP32-S3 integra un controlador USB Serial/JTAG nativo, como se detalla en el diagrama de bloques de la Figura 10. Esto representa una ventaja significativa en el diseño de ingeniería, pues elimina la necesidad de chips conversores externos (como el CP2102) y permite la depuración directa del código a alta velocidad sobre el mismo puerto de carga. Combinado con su generosa memoria interna (512 KB de SRAM) y el soporte para PSRAM octal, el sistema ofrece un entorno robusto para ejecutar interfaces gráficas complejas sin latencia.

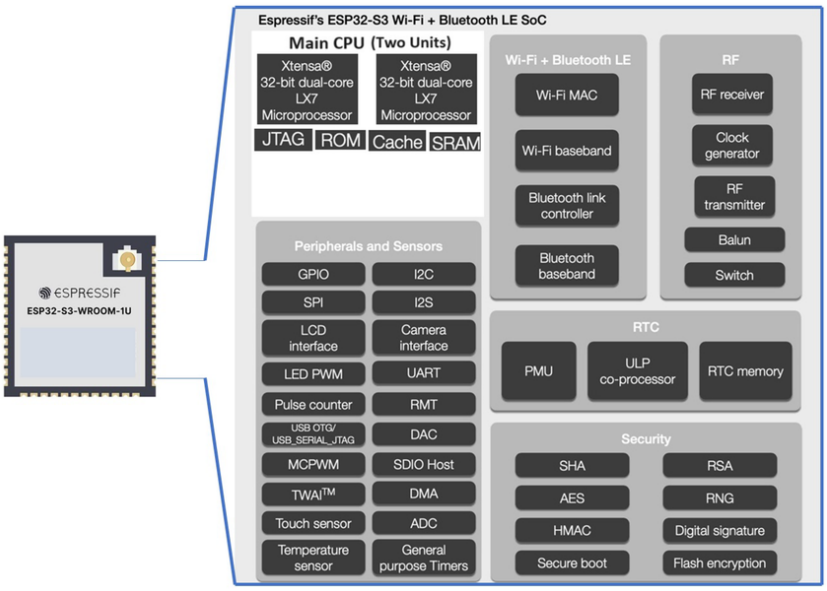


Figura 10. Arquitectura interna del SoC ESP32-S3: Núcleos LX7, USB nativo y periféricos.

V-D3. Interfaces de entrada para interacción del usuario: La interfaz de entrada combina tecnologías digitales y analógicas para ofrecer una experiencia de control completa. Para las entradas discretas (acciones de disparo o menú), se implementan pulsadores táctiles momentáneos. El reto ingenieril en estos componentes es el rebote mecánico"(*bouncing*), ilustrado en la Figura 11, un fenómeno donde los contactos metálicos generan transiciones espurias antes de estabilizarse. Aunque puede mitigarse con filtros RC en hardware, pedagógicamente es más valioso abordar este problema mediante algoritmos de software (*debouncing*), enseñando al estudiante a filtrar ruido en señales digitales.

Para el control direccional, se integra un joystick analógico de dos ejes basado en potenciómetros. Este transductor convierte el desplazamiento angular de la palanca en una variación de voltaje continuo (división de tensión). A diferencia de una cruceta digital (D-Pad), el joystick introduce al estudiante en el muestreo de señales continuas, permitiendo interpretar no solo la dirección del movimiento, sino también su magnitud o velocidad, enriqueciendo la interactividad de los proyectos mecatrónicos.

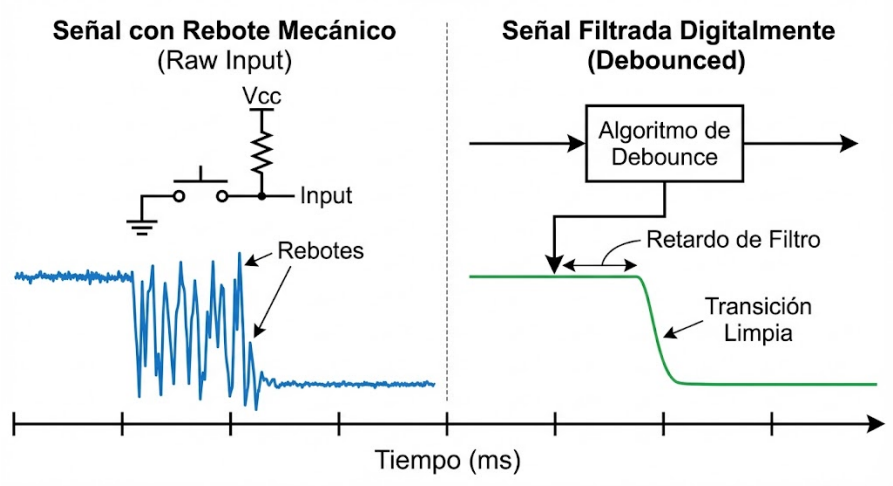


Figura 11. Fenómeno de rebote mecánico en pulsadores y su filtrado digital.

V-D4. Visualización mediante tecnología TFT y estándar RGB565: La visualización se confía a una pantalla de Cristal Líquido con Transistores de Película Delgada (TFT-LCD). A diferencia de los displays pasivos, la matriz activa TFT asigna un transistor a cada píxel, garantizando tiempos de respuesta rápidos y evitando el .efecto fantasma"(*ghosting*) en animaciones rápidas.

El control de color se realiza bajo el estándar RGB565 (16 bits por píxel: 5 rojos, 6 verdes, 5 azules), cuya estructura se muestra en la Figura 12. Esta codificación asimétrica, que otorga un bit extra al canal verde, se fundamenta en la fisiología humana, ya que el ojo es biológicamente más sensible a la luminancia en el espectro verde [25]. Comprender y manipular este formato de datos a nivel de bit es un ejercicio práctico fundamental de manipulación binaria y optimización de memoria gráfica para los estudiantes.

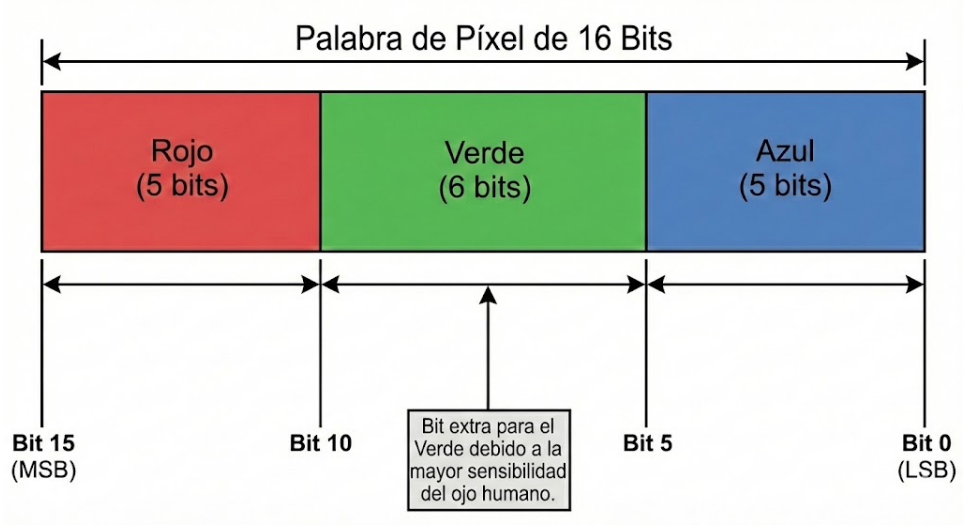


Figura 12. Estructura de bits del formato de color RGB565 utilizado en sistemas embebidos.

V-D5. Conversión analógica-digital (ADC) y cuantización: La interfaz con el mundo físico requiere la digitalización de señales continuas. El ESP32-S3 integra dos Convertidores Analógico-Digitales (ADC) de arquitectura SAR (Aproximaciones Sucesivas) con una resolución de 12 bits. Esto significa que el rango de voltaje de entrada (0 a 3.3V) se discretiza en $2^{12} = 4096$ niveles digitales, proceso representado en la Figura 13.

Es crucial destacar, desde un punto de vista académico, que los ADCs integrados en microcontroladores suelen presentar una curva de transferencia no lineal en los extremos del rango de voltaje (cerca de 0V y 3.3V). Esto obliga al estudiante a implementar técnicas de corrección por software (como tablas de calibración o LUTs) o a diseñar el hardware analógico para trabajar estrictamente en la zona lineal del convertidor, introduciendo conceptos valiosos de acondicionamiento de señal y metrología.

V-D6. Protocolos de comunicación serie: La interconexión eficiente entre el procesador central y sus periféricos depende de protocolos de comunicación serie estandarizados, cuya selección se justifica en la Tabla IX. La elección de cada protocolo para esta consola responde a un compromiso técnico entre velocidad de transferencia, complejidad del cableado y distancia de transmisión.

Para la transmisión de video, se emplea el protocolo SPI (Serial Peripheral Interface). El ESP32-S3 permite configurar este bus a frecuencias de hasta 80 MHz, lo cual es vital para mantener una tasa de refresco fluida (FPS) en la pantalla. Para la sensórica auxiliar se utiliza el bus I2C, ideal por su simplicidad de cableado. Finalmente, aunque el S3 posee USB nativo, se mantiene el soporte para el protocolo UART clásico, garantizando la compatibilidad con módulos externos como GPS o Bluetooth serial heredados.

IV-D6.1. SPI (Serial Peripheral Interface):
El protocolo SPI se selecciona como el bus principal para la transmisión de video hacia la pantalla TFT debido a su capacidad de alto rendimiento (*High Throughput*). Es un protocolo síncrono que opera bajo una arquitectura Maestro-Esclavo utilizando cuatro líneas de señal: reloj (SCLK), datos de salida (MOSI), datos de entrada (MISO)

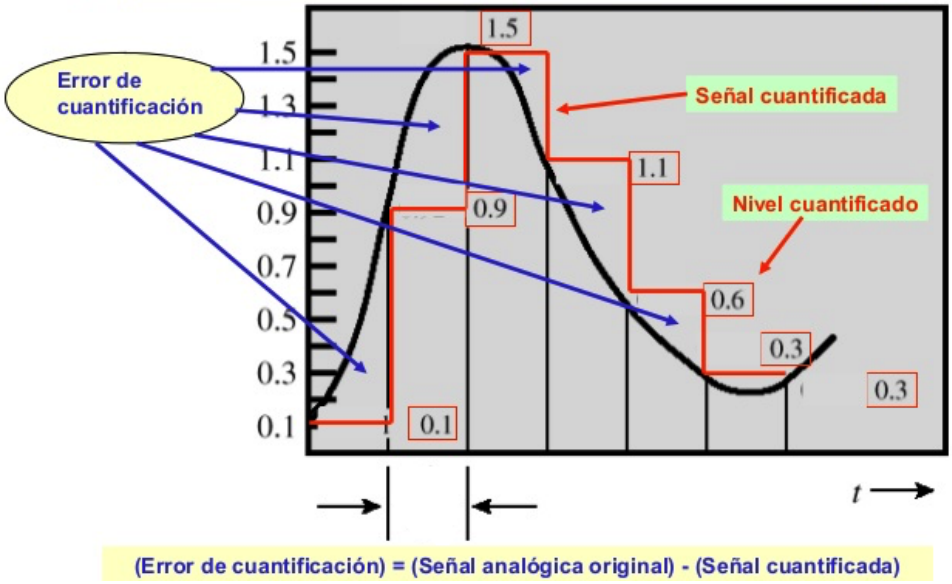


Figura 13. Discretización de una señal continua: Relación entre voltaje y valor digital de 12 bits.

y selección de chip (CS). En el ESP32-S3, este bus puede operar a frecuencias superiores a los 40 MHz, lo cual es indispensable para transmitir los millones de píxeles por segundo necesarios para mantener una tasa de refresco fluida (30-60 FPS).

IV-D6.2. I2C (Inter Integrated Circuit):

A diferencia del SPI, el protocolo I2C se implementa para la integración de sensores auxiliares y expansores de puertos, priorizando la simplicidad del cableado sobre la velocidad. Su arquitectura física se basa en una topología de ”Drenador Abierto”(*Open-Drain*) sobre dos líneas bidireccionales (SDA y SCL) [26]. Desde el punto de vista de la ingeniería electrónica, esta configuración impone un requisito crítico: la implementación de resistencias de *pull-up* para garantizar el estado lógico alto. El ESP32-S3 gestiona este protocolo en modo *Fast Mode* (400 kHz), permitiendo conectar múltiples dispositivos utilizando solo dos pines del microcontrolador.

IV-D6.3. UART (Transmisor Receptor Asíncrono Universal):

El protocolo UART se mantiene como el estándar fundamental para la telemetría y la depuración del sistema. Al ser asíncrono, carece de línea de reloj compartida, obligando a ambos extremos a acordar una velocidad de transmisión exacta (*Baud Rate*). En el diseño con el ESP32-S3, se aprovecha su capacidad de USB-CDC nativo, lo que permite que la conexión física USB actúe como un puerto UART virtual de alta velocidad, facilitando la impresión de logs de depuración en el monitor serie del ordenador sin requerir hardware adicional.

Tabla IX
MATRIZ DE SELECCIÓN DE PROTOCOLOS DE COMUNICACIÓN SEGÚN EL PERIFÉRICO.

Protocolo	**Tipo de Reloj**	**Velocidad Típica**	**Aplicación en la Consola**
SPI	Síncrono (4 hilos)	Alta (>40 MHz)	Pantalla TFT (Video)
I2C	Síncrono (2 hilos)	Media (400 kHz)	Sensores y Expansores
UART/USB	Asíncrono	Variable / Alta	Depuración y Carga

V-E. Diseño físico, ergonomía y procesos de manufactura

La materialización de un sistema embebido con fines educativos no se limita a la conexión funcional de componentes electrónicos; requiere un proceso riguroso de diseño industrial e integración mecatrónica. La carcasa del dispositivo actúa como la interfaz física primaria entre el estudiante y la tecnología, por lo que su diseño debe responder a criterios estrictos de ergonomía, seguridad y manufacturabilidad. En esta sección se detallan las metodologías de ingeniería mecánica y los procesos de fabricación aditiva empleados para transformar los requerimientos teóricos en un producto tangible y robusto.

V-E1. Diseño mecánico y modelado CAD del dispositivo: El desarrollo estructural del chasis se ejecutó bajo la metodología de Diseño Paramétrico utilizando software CAD (Diseño Asistido por Computadora). A diferencia del modelado directo, el diseño paramétrico permite definir la geometría mediante variables y restricciones matemáticas. Esto es crucial en un entorno de I+D, pues permite modificar el espesor de una pared o el diámetro de un agujero en el histórico de operaciones y regenerar automáticamente todo el modelo, facilitando iteraciones rápidas sin reconstruir la pieza desde cero.

Se adoptó una estrategia de diseño ”Top-Down”(Descendente) dentro del entorno de ensamblaje. En este enfoque, la geometría de la carcasa no se dibuja de forma aislada, sino que se referencia directamente a partir de un esqueleto virtual compuesto por los modelos 3D de los componentes electrónicos críticos (PCB, batería, pantalla TFT y módulo de carga).

Un aspecto fundamental abordado en esta etapa es la gestión de tolerancias dimensionales y el análisis de interferencias estáticas. Dado que la manufactura aditiva carece de la precisión micrométrica del mecanizado CNC, se definieron holguras de ingeniería (*Clearance*) específicas:

- **Zona de PCB:** Holgura de 0.4 mm perimetral para compensar la expansión térmica y las irregularidades de los bordes del circuito impreso.
- **Puertos de conexión (USB-C / SD):** Holgura de 0.5 mm para garantizar la inserción del cable sin fricción excesiva, considerando las variaciones de manufactura de los conectores comerciales.
- **Botones mecánicos:** Holgura de 0.25 mm para evitar el atascamiento por fricción (*stick-slip*) durante la pulsación.

V-E2. Manufactura aditiva mediante impresión 3D (FDM): Para la validación física y producción de lotes pequeños, se seleccionó la tecnología de Modelado por Deposición Fundida (FDM). Este proceso construye la pieza depositando material termoplástico capa por capa. Desde la perspectiva de la resistencia de materiales, el

ingeniero debe considerar que las piezas FDM son anisotrópicas: su resistencia mecánica varía según la dirección de la carga. La unión entre capas (eje Z) es significativamente más débil (aprox. 50-70 % de la resistencia nominal) que la tracción a lo largo de los filamentos (ejes X/Y) [27].

Para mitigar esta debilidad inherente y garantizar que la consola soporte el uso rudo en un laboratorio universitario, se configuraron parámetros de laminado (*Slicing*) orientados a la integridad estructural:

- **Perímetros (Shells):** Se configuraron 3 perímetros sólidos de 0.4 mm. Esto crea un exoesqueleto de 1.2 mm de espesor que soporta la mayor parte de la carga de torsión y compresión.
- **Relleno (Infill):** Se optó por un patrón **Giroide** al 20 %. A diferencia de los patrones rectilíneos, el giroide es isotrópico en el plano horizontal y distribuye las tensiones de manera uniforme, además de ofrecer una excelente resistencia a la impresión vertical sin requerir soportes internos excesivos.
- **Orientación de impresión:** La carcasa se imprime con la cara frontal apoyada en la cama caliente. Esto no solo mejora el acabado superficial estético, sino que alinea las capas de manera que los esfuerzos de flexión al presionar los botones actúan perpendicularmente a la laminación, maximizando la resistencia.

V-E3. Selección de materiales: La selección del material termoplástico no es trivial y debe equilibrar propiedades mecánicas, térmicas y de seguridad ambiental. Se evaluaron tres polímeros comunes en la industria: ABS (Acrilonitrilo Butadieno Estireno), PLA (Ácido Poliláctico) y PETG (Tereftalato de Polietileno Glicol), cuyas propiedades se resumen en la Tabla X.

Si bien el ABS es el estándar en bienes de consumo por su resistencia al impacto, presenta dos desventajas críticas para este proyecto: requiere temperaturas de impresión altas que generan contracción severa (*warping*), dificultando la precisión dimensional en piezas grandes, y emite vapores de estireno potencialmente nocivos en aulas sin ventilación industrial.

Por consiguiente, se seleccionó el PLA como material exclusivo para la fabricación de toda la consola, incluyendo la carcasa principal:

1. **Propiedades mecánicas y hápticas:** Su alto Módulo de Young (rigidez) proporciona una sensación de solidez y calidad percibida superior en la manipulación del dispositivo.
2. **Seguridad y entorno de uso:** Al ser biodegradable y no tóxico, es completamente seguro para el contacto prolongado con la piel de los usuarios en un entorno educativo.
3. **Comportamiento térmico adecuado:** Aunque materiales como el PETG ofrecen una mayor temperatura de transición vítrea (Tg), se determinó que el calor generado por el procesador ESP32 y la electrónica asociada durante su funcionamiento normal no alcanza niveles que comprometan la integridad estructural del PLA (cuya Tg ronda los 60°C).

Tabla X
MATRIZ COMPARATIVA DE PROPIEDADES TERMOMECÁNICAS DE POLÍMEROS PARA IMPRESIÓN 3D.

Material	**Rigidez (Módulo E)**	**Resistencia Térmica (Tg)**	**Facilidad de Impresión**	**Seguridad (Gases)**
PLA	Alta (Rígido)	Baja ($\approx 60^{\circ}C$)	Excelente	Segura (Orgánico)
PETG	Media (Dúctil)	Media ($\approx 80^{\circ}C$)	Buena	Baja emisión
ABS	Media	Alta ($\approx 105^{\circ}C$)	Difícil (Warping)	Tóxica (Estireno)

V-E4. Ergonomía aplicada al diseño de consolas educativas: La ergonomía juega un papel preventivo en el diseño de herramientas de aprendizaje. Basándose en la antropometría de la mano promedio de estudiantes universitarios [29], el diseño debe minimizar la fatiga muscular durante sesiones de codificación y prueba. Esto implica dimensionar el agarre para evitar la hiperextensión de los pulgares al alcanzar el joystick o los botones de acción.

El diseño también considera la arquitectura de la información física, la disposición de los controles no es aleatoria; sigue principios de usabilidad donde los elementos de navegación (Joystick) y confirmación (Botones A/B) se ubican en zonas de acceso primario, mientras que los puertos de carga y reinicio se relegan a zonas secundarias. Una mala ergonomía podría convertirse en un distractor (*Carga Extraña*) e interferir en la experiencia de aprendizaje.

El diseño incorpora una estrategia de Gestión Térmica Pasiva, dado que el microcontrolador y el controlador de pantalla generan calor (aprox. 1.5 Watts bajo carga máxima), se diseñaron rejillas de ventilación en la cara posterior superior. Esto crea un efecto chimenea natural por convección: el aire caliente asciende y escapa por la rejilla, manteniendo la temperatura interna en un rango operativo seguro sin necesidad de ventiladores que consuman batería.

VI. METODOLOGÍA

VI-A. Introducción

La presente investigación se estructura como un Proyecto Tecnológico aplicado, orientado a mitigar la carga cognitiva en la asimilación de la Programación Orientada a Objetos (POO). Para ello, establece una ruta metodológica iterativa e intervencionista enfocada en el diseño, fabricación y validación empírica de una consola educativa basada en el microcontrolador ESP32.

Para abordar esta convergencia entre ingeniería y pedagogía, el estudio adopta un enfoque mixto. La dimensión cuantitativa rige la evaluación técnica, abarcando el consumo energético, tiempos de ejecución del intérprete, respuesta de periféricos y tolerancias de manufactura aditiva. Simultáneamente, la dimensión cualitativa analiza la usabilidad, la ergonomía de la interfaz física y la eficacia del dispositivo para abstraer los paradigmas de programación.

El desarrollo técnico se fundamenta en una arquitectura embebida robusta. A nivel de hardware, se seleccionó el ESP32 por su procesamiento de doble núcleo y versatilidad para gestionar múltiples protocolos de comunicación (SPI, I2C, UART), estas características son indispensables para el manejo fluido de la interfaz gráfica de usuario (GUI), la conversión analógica-digital requerida por el joystick, y la gestión directa de las entradas digitales de los pulsadores físicos. A nivel de firmware, el desarrollo se ejecuta estrictamente en C++. Esta elección garantiza una gestión de memoria directa y eficiente, permitiendo que el propio código fuente se estructure bajo los principios de POO (clases, herencia y polimorfismo) que la consola pretende enseñar, logrando un sistema escalable y optimizado, tal como se esquematiza en el diagrama de flujo de la Figura 14.

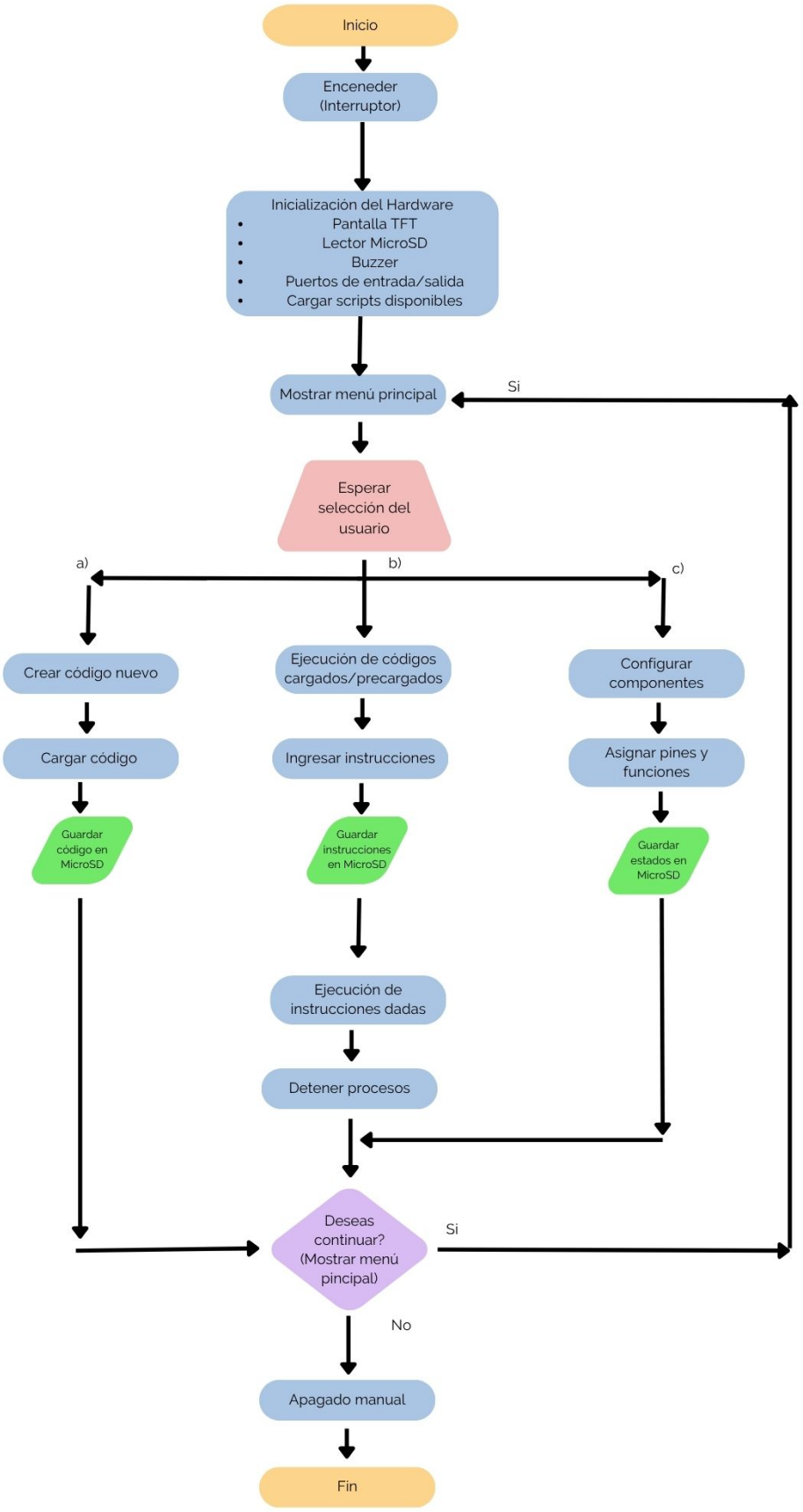


Figura 14. Diagrama de flujo del proceso de la consola.

El procedimiento metodológico diseñado para la materialización de la consola se estructura en cinco fases secuenciales e interdependientes, las cuales guiarán el desarrollo de las secciones posteriores.

- **Fase de Especificación y Requisitos:** Comprende la delimitación de las restricciones técnicas del hardware (dimensiones, consumo, potencia de cálculo) y la definición de las interfaces de usuario necesarias para una interacción fluida y ergonómica.
- **Fase de Diseño Electrónico y Mecánico:** Abarca la selección meticulosa de los componentes en estado sólido, el diseño de la placa de circuito impreso (PCB) para garantizar la integridad de las señales, y el modelado tridimensional (CAD) de la carcasa, considerando parámetros de manufactura aditiva y ergonomía de interacción.
- **Fase de Desarrollo de Software Embebido:** Consiste en la estructuración de la arquitectura del firmware en C++, el diseño del analizador léxico y sintáctico para el intérprete de código simplificado, y la programación de los controladores de bajo nivel para la pantalla y la matriz de entrada.
- **Fase de Integración y Ensamblaje:** Representa la consolidación de las etapas previas mediante la manufactura del PCB, la impresión 3D de la envolvente mecánica, el ensamble electromecánico y la carga del firmware en la unidad de procesamiento central.
- **Fase de Validación y Pruebas Técnicas:** Involucra la ejecución de protocolos de prueba estandarizados para evaluar la estabilidad eléctrica del sistema, la autonomía energética de la fuente de alimentación portátil y el correcto procesamiento de los algoritmos introducidos por el usuario bajo condiciones de estrés operativo.

De esta forma se establece los criterios de ingeniería y los procedimientos sistemáticos que garantizan que la transición desde el modelo conceptual hasta el prototipo físico se realice bajo estrictos estándares de calidad, asegurando que el dispositivo resultante sea técnicamente viable, eléctricamente seguro y pedagógicamente funcional.

VI-B. Fase de Especificación y Requisitos

La transición del modelo conceptual al diseño de ingeniería de la consola educativa exige una especificación exhaustiva de las restricciones operativas y funcionales del sistema. Esta definición de requisitos actúa como el marco de referencia inmutable contra el cual se validará el desempeño del prototipo final. Para garantizar que el dispositivo cumpla simultáneamente con las normativas de diseño electrónico y los objetivos de enseñanza de la Programación Orientada a Objetos, los requerimientos se han categorizado en tres dimensiones críticas: funcionales, no funcionales y pedagógicos.

VI-B1. Requisitos Funcionales del Hardware y Firmware: Los requerimientos funcionales definen las capacidades de procesamiento, interacción y respuesta que el sistema embebido debe ejecutar de manera determinista.

- Procesamiento Concurrente y Gestión de Interrupciones: El microcontrolador central debe poseer una arquitectura capaz de manejar múltiples hilos de ejecución. Es imperativo que el sistema gestione la lectura de la matriz de entrada (teclado/joystick) mediante interrupciones de hardware (ISRs) para evitar cuellos de botella en el bucle principal, garantizando que la lectura de las instrucciones del usuario no interfiera con la actualización de la interfaz gráfica.

- Despliegue Gráfico en Tiempo Real: El subsistema de visualización debe operar sobre una pantalla con tecnología TFT (*Thin Film Transistor*) de matriz activa. Se requiere que el controlador de video soporte el estándar de color RGB565 a través de un bus SPI de alta velocidad (mínimo 40 MHz), permitiendo el renderizado fluido de menús interactivos, animaciones didácticas y una consola de texto para la depuración de código sin artefactos visuales ni latencia perceptible (*tearing* o *ghosting*).
- Gestión de Memoria No Volátil: El dispositivo debe integrar un módulo lector de tarjetas MicroSD comunicado vía bus SPI secundario. Este subsistema es fundamental para almacenar los scripts generados por el usuario, cargar librerías precompiladas de clases y guardar el registro de errores (*logs*), simulando el sistema de archivos de un Entorno de Desarrollo Integrado (IDE) profesional.
- Tratamiento de Señales de Entrada: El firmware debe incorporar algoritmos de filtrado digital (*debouncing*) [30] para todas las entradas mecánicas, asegurando que los rebotes eléctricos inherentes a los pulsadores y contactos de la matriz no generen falsos positivos durante la digitación del código.

VI-B2. Requisitos No Funcionales y Restricciones Físicas: Esta categoría establece los parámetros de calidad, portabilidad, seguridad eléctrica y robustez geométrica de la consola.

- Autonomía Energética y Regulación Térmica: Al tratarse de un dispositivo portátil para el aula, el sistema de alimentación debe basarse en celdas de polímero de litio (LiPo) con una densidad energética suficiente para garantizar un mínimo de 2 horas de operación continua bajo carga máxima (procesador activo y pantalla al 100 % de retroiluminación). Asimismo, el circuito debe incluir reguladores LDO (*Low Dropout*) para mantener un rizado de voltaje mínimo en la línea de 3.3V, y la carcasa debe prever rejillas de disipación térmica pasiva para el SoC.
- Integridad Estructural y Tolerancias: La envolvente mecánica, fabricada mediante impresión 3D, debe soportar impactos moderados típicos de un entorno de laboratorio universitario. Los requerimientos dimensionales exigen holguras estrictas (entre 0.2 mm y 0.5 mm) para los puertos de comunicación y las cavidades de los actuadores mecánicos, evitando el atascamiento por fricción (*stick-slip*) durante el uso intensivo.
- Tasa de Fallos y Tolerancia a Errores de Software: El firmware debe implementar mecanismos de vigilancia (*Watchdog Timers*) capaces de reiniciar el sistema automáticamente en caso de que un script mal estructurado por el estudiante provoque un desbordamiento de pila (*Stack Overflow*) o un bucle infinito, protegiendo la integridad del sistema operativo subyacente.

VI-B3. Requisitos Pedagógicos y de Interfaz Humano-Máquina (HMI): Dado que el objetivo final es la enseñanza de conceptos de ingeniería en C++, el diseño del hardware debe subordinarse a la mitigación de la carga cognitiva del estudiante.

- Abstracción Visual de la Sintaxis: La pantalla debe poseer dimensiones y densidad de píxeles (PPI) adecuadas para visualizar bloques de código o diagramas de clases sin forzar la vista del estudiante. La paleta de colores de la interfaz debe seguir normativas de alto contraste para destacar palabras clave, variables y objetos, facilitando la asimilación sintáctica.
- Ergonomía Bimanual: La disposición espacial de la matriz de botones y el control direccional debe obedecer a estudios antropométricos, permitiendo al usuario sujetar el dispositivo y digitar instrucciones de manera

bimanual sin requerir la hiperextensión de los pulgares, reduciendo la fatiga musculoesquelética durante sesiones prolongadas de programación.

- Respuesta Sensorial Inmediata (Feedback): El dispositivo debe integrar un actuador piezoeléctrico (*buzzer*) y retroalimentación visual inmediata ante cada interacción física. Esta confirmación multisensorial es vital en etapas tempranas del aprendizaje para correlacionar la acción física (presionar el botón de ejecución) con el estado lógico del software.

Para consolidar estas especificaciones y establecer un mejor detalle, la Tabla XI sintetiza la Matriz de Requisitos Generales del sistema.

Tabla XI
MATRIZ DE REQUISITOS GENERALES DEL SISTEMA EMBEBIDO.

ID	Categoría	Descripción del Requisito	Prioridad
RF-01	Funcional (Procesamiento)	Ejecución concurrente de lectura de sensores y actualización de pantalla mediante RTOS.	Alta
RF-02	Funcional (Visual)	Renderizado de la Interfaz Gráfica (GUI) por bus SPI a un mínimo de 30 FPS.	Alta
RF-03	Funcional (Memoria)	Capacidad de lectura/escritura en tarjeta MicroSD externa.	Media
RNF-01	No Funcional (Energía)	Autonomía mínima de 2 horas bajo carga computacional continua.	Alta
RNF-02	No Funcional (Térmico)	Disipación térmica pasiva eficiente para mantener el SoC por debajo de $60°C$.	Alta
RNF-03	No Funcional (Mecánica)	Tolerancias de ensamble de $0{,}2$ mm a $0{,}5$ mm en actuadores mecánicos.	Media
RP-01	Pedagógico (HMI)	Distribución ergonómica bimanual que prevenga la hiperextensión del pulgar.	Alta
RP-02	Pedagógico (Cognitivo)	Abstracción visual del código mediante colores de alto contraste.	Alta

VI-C. Fase de Diseño Electrónico y Mecánico

VI-C1. Diseño de la Arquitectura del Hardware: Una vez definidos los componentes y el entorno de desarrollo, se procedió a la integración lógica de los subsistemas. La arquitectura del hardware se diseñó bajo una topología centralizada, donde el SoC ESP32-S3 actúa como el nodo coordinador maestro, gestionando el flujo de datos entre los periféricos de entrada (sensores y pulsadores) y los de salida (pantalla y audio).

Para garantizar la integridad de las señales y evitar conflictos en los buses de comunicación, se segregaron las interfaces según su protocolo de transmisión. El diseño se estructura en tres bloques funcionales principales (bus de gráficos, almacenamiento e interrupciones), tal como se representa en el diagrama general de la Figura 15.

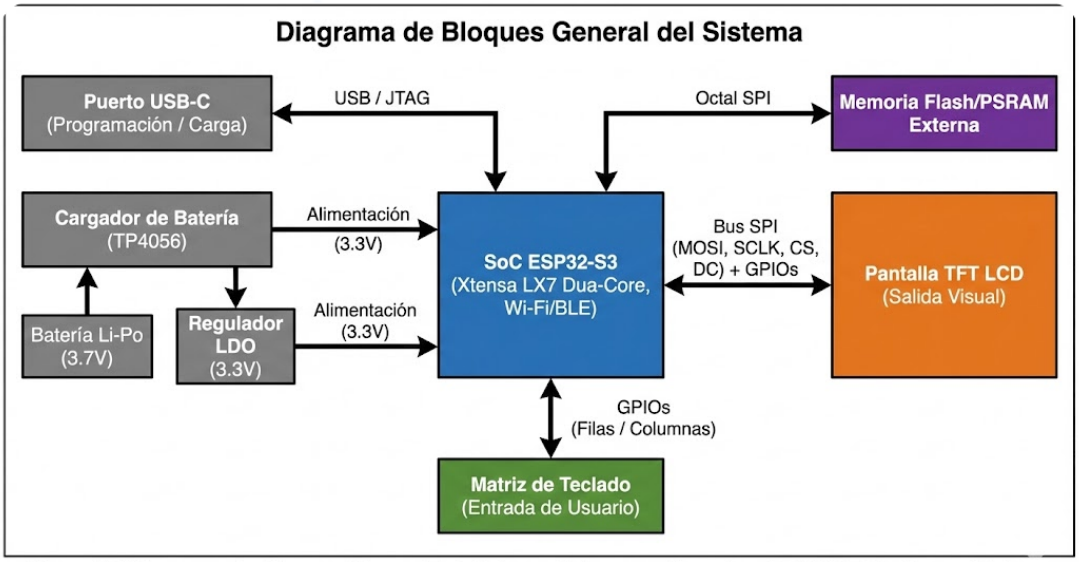


Figura 15. Diagrama de Bloques General del Sistema: Interconexión entre el ESP32-S3 y los periféricos.

V-C1.1. Topología de Conexión y Buses de Comunicación:

La comunicación crítica del sistema recae sobre el protocolo SPI (*Serial Peripheral Interface*). Dado que tanto la pantalla TFT como el lector MicroSD operan bajo este estándar, se diseñó una arquitectura de bus compartido (*Shared SPI Bus*) utilizando el bloque hardware VSPI del ESP32.

Para evitar la contención de datos (*bus contention*), se asignaron líneas de selección de chip (*Chip Select* - CS) dedicadas para cada periférico:

- **Modo Gráfico:** Cuando el pin `TFT_CS` se lleva a bajo lógico (LOW), el bus se configura para transmitir tramas de color RGB565 a alta frecuencia (40 MHz).
- **Modo Archivo:** Cuando el pin `SD_CS` es activado, el bus reduce su frecuencia operativa para cumplir con la estabilidad requerida por el sistema de archivos FAT32.

V-C1.2. Mapeo de Puertos (Pin Mapping):

La asignación de los pines GPIO (*General Purpose Input/Output*) se optimizó para facilitar el ruteo del Circuito Impreso (PCB). Se reservaron los pines conectados al ADC2 para el uso exclusivo del joystick analógico, evitando el conflicto de hardware conocido con el driver WiFi del ESP32, y se utilizaron pines de entrada digital con resistencias de *pull-up* internas para la lectura de la matriz de botones.

Las Tablas XII y XIII detallan la distribución física de las conexiones, sirviendo como referencia para el diseño esquemático y la configuración del firmware del dispositivo.

Tabla XII
TABLA DE MAPEO DE PINES (PINOUT MAP) - PUERTOS EDUCATIVOS LIBRES.

Puerto / Etiqueta	GPIO ESP32	Capacidad / Función
Puertos Analógicos y PWM		
A1	GPIO 19	Entrada Analógica (ADC), Salida PWM
A2	GPIO 20	Entrada Analógica (ADC), Salida PWM
A3	GPIO 21	Entrada Analógica (ADC), Salida PWM
A4	GPIO 18	Entrada Analógica (ADC), Salida PWM
A5	GPIO 17	Entrada Analógica (ADC), Salida PWM
Puertos Exclusivos PWM		
B1	GPIO 45	Salida PWM
B2	GPIO 46	Salida PWM
D1	GPIO 35	Salida PWM
D2	GPIO 36	Salida PWM
D3	GPIO 37	Salida PWM
D4	GPIO 41	Salida PWM
D5	GPIO 42	Salida PWM
D6	GPIO 44	Salida PWM
D7	GPIO 47	Salida PWM
D8	GPIO 48	Salida PWM
Puertos Digitales de Propósito General		
E0	GPIO 0	Entrada/Salida Digital
E1	GPIO 1	Entrada/Salida Digital
E3	GPIO 3	Entrada/Salida Digital

Tabla XIII
TABLA DE MAPEO DE PINES (PINOUT MAP) - HARDWARE FIJO INTERNO.

Periférico	Señal / Etiqueta	GPIO ESP32 Asignado
Pantalla TFT	TFT_MOSI	GPIO [Tu GPIO]
	TFT_MISO	GPIO [Tu GPIO]
	TFT_SCK	GPIO [Tu GPIO]
	TFT_CS	GPIO [Tu GPIO]
	TFT_DC	GPIO [Tu GPIO]
	TFT_RST	GPIO [Tu GPIO]
Joystick	JOY_X	GPIO [Tu GPIO] (ADC)
	JOY_Y	GPIO [Tu GPIO] (ADC)
	JOY_BTN	GPIO [Tu GPIO] (Digital)
Pulsadores	BTN_A	GPIO [Tu GPIO] (Digital)
	BTN_B	GPIO [Tu GPIO] (Digital)
Actuador Acústico	BUZZER	GPIO [Tu GPIO] (PWM)
Energía	BATTERY	GPIO [Tu GPIO] (ADC)

V-C1.3. Etapa de Potencia y Aislamiento:

La arquitectura de alimentación contempla la gestión híbrida de energía. Se implementó un circuito de conmutación automática mediante diodos Schottky de baja caída, diseñado para priorizar la entrada de 5V del puerto USB-C cuando este se encuentra conectado. Esto aísla la batería LiPo durante las sesiones de depuración, permitiendo que el módulo TP4056 ejecute el ciclo de carga de manera independiente sin que la carga del sistema interfiera con la terminación de carga, protegiendo así la vida útil de la celda electroquímica.

VI-C2. Simulación del circuito y validación lógica: Antes de proceder con la manufactura irreversible del Circuito Impreso (PCB), el diseño se sometió a una fase estricta de verificación para mitigar riesgos económicos y errores de diseño. Esta etapa se dividió en dos niveles de abstracción: la simulación lógica asistida por computadora (CAE) y la validación física mediante un prototipo en placa de pruebas (*Breadboard*).

V-C2.1. Verificación de Reglas Eléctricas (ERC):

El primer filtro de seguridad se ejecutó dentro del software de diseño electrónico (EDA). Se utilizó la herramienta de Comprobación de Reglas Eléctricas (ERC - *Electrical Rule Check*) sobre el esquemático. Este análisis automatizado verifica la integridad de la lista de conexiones (*Netlist*), alertando sobre pines flotantes, cortocircuitos inadvertidos entre las redes de potencia (VCC/GND) y conflictos de dirección en los buses bidireccionales.

El resultado de este análisis certificó que todas las redes lógicas tienen un punto de origen y destino válidos, y que las etiquetas de los buses globales son coherentes en todas las hojas del diseño jerárquico, como se evidencia en la Figura 16.

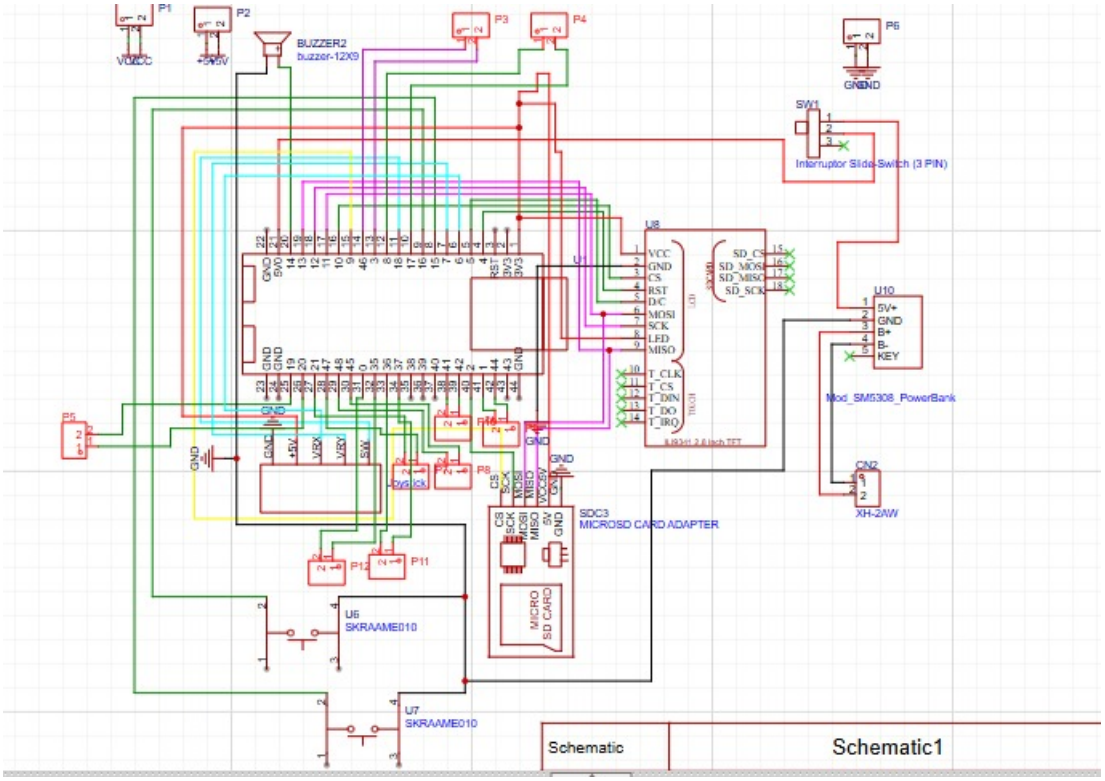


Figura 16. Diseño y verificación de las conexiones eléctricas del sistema mediante el software EasyEDA.

V-C2.2. Simulación Funcional del Firmware:

Para validar la lógica de control antes de comprometer el hardware físico, se empleó el entorno de simulación de ArduinoIDE. Esta plataforma permite emular el comportamiento del SoC ESP32-S3 y sus periféricos (incluyendo la pantalla ILI9341) en un entorno virtual.

A través de esta simulación, se validó:

- La correcta inicialización de la secuencia de arranque (*Power-On Sequence*) del driver de la pantalla.
- La lógica de la máquina de estados finitos para la navegación de los menús.
- La gestión de memoria del microcontrolador ante la carga de gráficos pesados, monitoreando el Monitor Serial virtual para detectar fugas de memoria (*Memory Leaks*).

V-C2.3.Validación Física Preliminar (Protoboard):

Se ensambló una Prueba de Concepto (PoC) en una placa de prototipado rápido, visible en la Figura 17. El objetivo de este montaje no fue estético, sino funcional: comprobar la inmunidad al ruido de los buses de comunicación en un entorno real.

Figura 17. Montaje experimental en placa de pruebas (Breadboard) para validación de señales físicas.

Durante esta prueba se realizaron mediciones críticas con instrumental de laboratorio:

- Integridad del bus SPI: se verificó mediante osciloscopio que las señales de reloj (SCK) y datos (MOSI) a 40 MHz mantuvieran flancos definidos y no sufrieran deformación excesiva por la capacitancia parásita del cableado.
- Consumo de Corriente: se midió el amperaje en reposo y bajo carga máxima, confirmando que el regulador de voltaje y la batería seleccionada operan dentro de los márgenes de seguridad térmica.

Esta triangulación entre la validación por software (ERC), la simulación de firmware y el prototipo físico permitió autorizar el paso a la fase de diseño del PCB con un nivel de incertidumbre técnica cercano a cero.

VI-C3. Adquisición de componentes y justificación técnica: La materialización de la consola educativa requiere la selección de hardware en estado sólido que no solo satisfaga los requerimientos funcionales descritos en la sección anterior, sino que garantice la escalabilidad, la robustez industrial y la viabilidad económica del proyecto. A diferencia de los ensamblajes genéricos basados en plataformas de 8 bits, este proyecto exige componentes capaces de ejecutar un Sistema Operativo en Tiempo Real (RTOS) y renderizar gráficos sin comprometer la fluidez de las entradas del usuario. A continuación, se detalla y fundamenta la selección de cada subsistema crítico que conforma la arquitectura electrónica del dispositivo.

V-C3.1. Unidad Central de Procesamiento: SoC ESP32-S3:

El núcleo computacional seleccionado es el SoC (System on Chip) ESP32-S3, cuyo pinout se detalla en la Figura 18. La decisión de emplear esta variante específica responde a tres ventajas competitivas críticas para el desarrollo de la consola didáctica:

Arquitectura y potencia de cálculo: el sistema se basa en dos núcleos Xtensa LX7 de 32 bits capaces de operar hasta 240 MHz. Esta capacidad de procesamiento concurrente permite aislar lógicamente las tareas: un núcleo se dedica exclusivamente al renderizado de la Interfaz Gráfica de Usuario (GUI) y la gestión del bus SPI de la pantalla, mientras que el segundo núcleo atiende las interrupciones del teclado, el intérprete de código en C++ y la lógica del programa en ejecución.

Integración un controlador USB nativo: el ESP32-S3 posee un controlador USB Serial/JTAG en el propio silicio. Esto elimina la necesidad de circuitos integrados conversores externos, reduciendo el costo, simplificando el ruteo del circuito impreso (PCB) y permitiendo la depuración directa del código a alta velocidad sobre el mismo puerto de carga.

Gestión de memoria RAM: su amplia memoria SRAM es indispensable para soportar la carga que genera la instanciación dinámica de objetos y clases, base fundamental de este proyecto educativo.

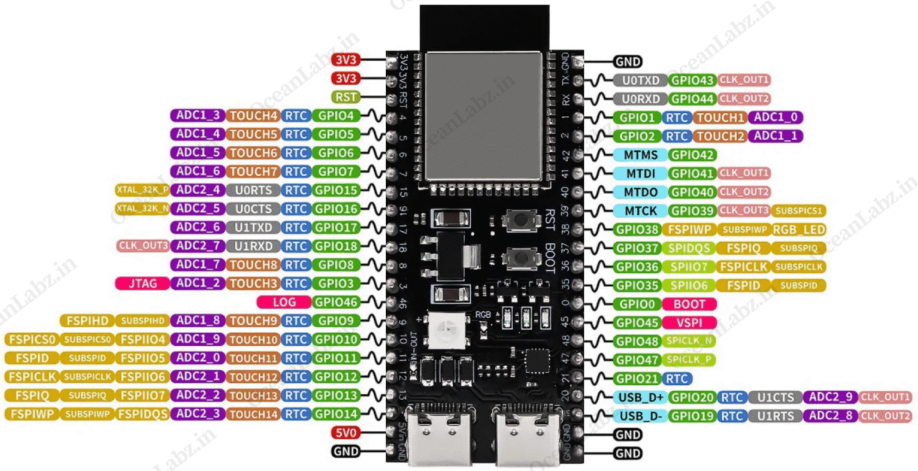


Figura 18. Diagrama de distribución de pines (Pinout).

V-C3.2. Subsistema de Visualización: Display TFT RGB565:

Para la interfaz visual, se seleccionó una pantalla de Cristal Líquido con Transistores de Película Delgada (TFT-LCD). A diferencia de los displays pasivos, la matriz activa TFT asigna un transistor a cada píxel, garantizando tiempos de respuesta rápidos y evitando el efecto fantasma (ghosting). Se descartaron las pantallas OLED debido a su susceptibilidad al ”quemado”(burn-in) al mostrar interfaces estáticas prolongadas, como las líneas de código de la consola.

El control de color se realiza bajo el estándar RGB565 (16 bits por píxel: 5 rojos, 6 verdes, 5 azules). Para mantener una tasa de refresco fluida, la pantalla se comunica mediante el bus SPI. El requerimiento de ancho de banda para lograr una transmisión fluida a pantalla completa se modela matemáticamente mediante la siguiente ecuación **(1)**:

$$B_{req} = W \times H \times D_{bits} \times FPS \tag{1}$$

Donde W es el ancho en píxeles, H es el alto en píxeles, D_{bits} es la profundidad de color (16 bits) y FPS son los cuadros por segundo. Esta alta demanda de datos justifica la configuración del bus SPI del ESP32 a frecuencias elevadas.

V-C3.3. Interfaz de Entrada HMI: Transductores y Matriz de Teclado:

El subsistema de captura de datos humanos combina tecnologías para ofrecer una experiencia de control completa. Para las entradas discretas, se implementan pulsadores táctiles organizados en una matriz. El reto ingenieril en estos componentes es el rebote mecánico"(bouncing), un fenómeno donde los contactos generan transiciones espurias antes de estabilizarse. El firmware abordará este problema mediante algoritmos de software (debouncing), garantizando lecturas limpias.

Adicionalmente, se integra un joystick analógico de dos ejes. Este transductor convierte el desplazamiento mecánico en una variación de voltaje continuo, el cual es procesado por los Convertidores Analógico-Digitales (ADC) del ESP32, operando a una resolución de 12 bits.

V-C3.4. Sistema de Alimentación y Regulación Energética:

Para garantizar la autonomía del dispositivo, se determinó el uso de una celda de Polímero de Litio (LiPo). La química del litio ofrece una densidad energética superior, vital para mantener el perfil ergonómico de la carcasa.

La recarga de la celda se confía a un módulo de carga lineal, el cual ejecuta un algoritmo estricto de Corriente Constante / Voltaje Constante (CC/CV), cuyo comportamiento se ilustra en la Figura 19.

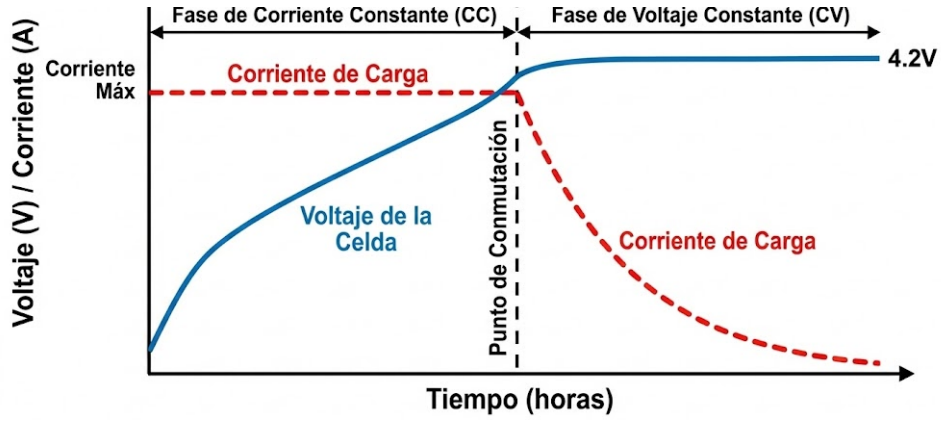


Figura 19. Curva de carga teórica (Corriente Constante / Voltaje Constante) para celdas de iones de litio.

Este algoritmo previene la sobrecarga y el embalamiento térmico de la batería. Dado que la lógica del procesador y la pantalla operan a 3.3V, se integra un regulador de baja caída (LDO) que asegura una entrega estable de energía al bus principal, mitigando el rizado de voltaje que podría causar inestabilidad en el microcontrolador. Para consolidar los requerimientos eléctricos, la Tabla XIV sintetiza las especificaciones operativas de los periféricos seleccionados, parámetros que gobernarán el diseño del circuito impreso (PCB).

VI-C4. Diseño Preliminar de la Consola (CAD): Una vez validadas las dimensiones de los componentes electrónicos y la integridad de los circuitos, se procedió a la fase de ingeniería mecánica. El objetivo de esta etapa fue desarrollar un contenedor físico que no solo alojara el hardware, sino que facilitara la interacción pedagógica. Para ello, se empleó software de Diseño Asistido por Computadora (CAD) paramétrico, permitiendo modificar las cotas del diseño en tiempo real según las restricciones de manufactura.

Tabla XIV
MATRIZ DE ESPECIFICACIONES ELÉCTRICAS Y PROTOCOLOS DE COMUNICACIÓN DE LOS COMPONENTES.

Componente	**Voltaje Operativo**	**Interfaz de Datos**	**Función Asignada**
SoC ESP32-S3	$3{,}0\text{V} - 3{,}6\text{V}$	I2C, SPI, UART, ADC	Procesamiento central y firmware
Display TFT RGB565	$3{,}3\text{V}$	SPI (Alta Velocidad)	Interfaz gráfica y consola de código
Lector MicroSD	$3{,}3\text{V} - 5{,}0\text{V}$	SPI (Estándar)	Almacenamiento no volátil
Joystick Analógico	$3{,}3\text{V}$	ADC (12 bits)	Navegación bidimensional
Pulsadores Táctiles	$3{,}3\text{V}$	GPIO (Digital)	Ingreso discreto de comandos
Regulador LDO	$4{,}5\text{V} - 12\text{V}$ (In)	Potencia DC	Reducción lineal a $3{,}3\text{V}$

V-C4.1. Definición del Volumen de Control:

El primer paso del modelado consistió en crear "fantasmas" digitales (dummy models) de los componentes críticos: la pantalla TFT, la batería LiPo y el módulo ESP32. Al ensamblar estos modelos virtuales, se determinó el "volumen de control" mínimo necesario, como se aprecia en la Figura 20. Se estableció una restricción de diseño fundamental: la batería debía ubicarse en una cámara aislada térmicamente del procesador para evitar la degradación química por calor, mientras que la pantalla debía estar centrada respecto al eje vertical para garantizar el equilibrio de peso.

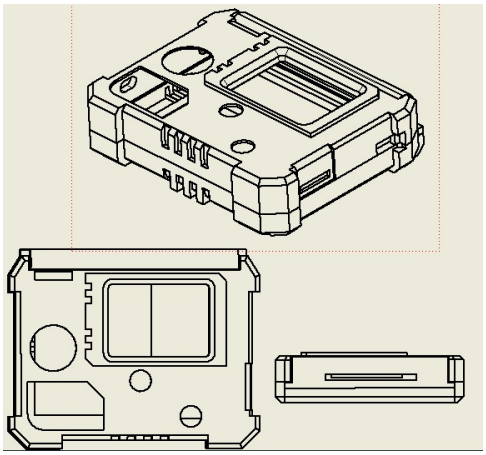


Figura 20. Vista (Wireframe) del ensamblaje CAD preliminar, mostrando la disposición interna de componentes.

V-C4.2. Arquitectura de Distribución (Layout):

Se evaluaron dos topologías de diseño: vertical y horizontal. Se seleccionó la disposición horizontal debido a la naturaleza de la tarea: la navegación por menús y monitoreo de sensores. Al distribuir un Joystick Analógico a la izquierda y los dos botones de acción a la derecha, se mimetiza la ergonomía de los mandos de control estándar (Gamepad), lo cual reduce la curva de adaptación para los estudiantes y facilita la interacción con la interfaz gráfica.

V-C4.3. Modelado de Cerramientos y Fijaciones:

En este diseño preliminar, se establecieron los planos de corte para dividir la carcasa en dos cuerpos:

- **Carcasa Superior (Frontplate):** Aloja la ventana de la pantalla y las guías de deslizamiento para los botones mecánicos, contiene los anclajes para la batería y el puerto de acceso USB-C.

- **Carcasa Inferior (Backplate):** Sujeción de la placa PCB y conexión de componentes

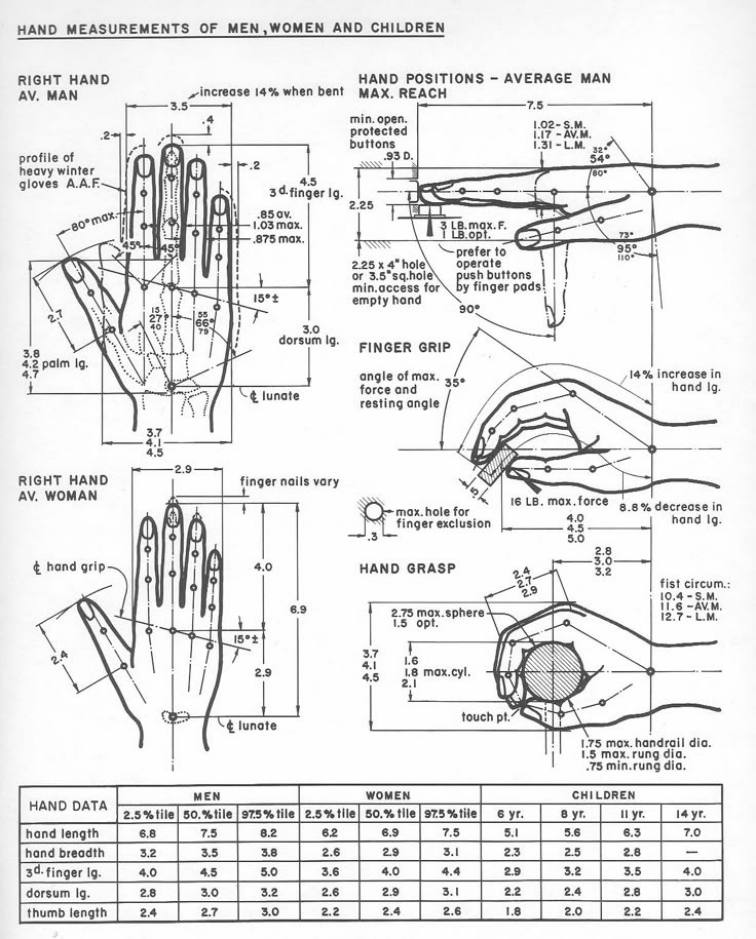
Se diseñaron torres de fijación (bosses) para tornillería métrica M3, descartando los cierres a presión (snaps) en esta fase inicial para facilitar el desmontaje recurrente y la inspección interna durante las pruebas de prototipo.

VI-C5. Consideraciones Ergonómicas y de Interfaz: El diseño de la interfaz física no se basó en criterios estéticos arbitrarios, sino en la aplicación estricta de la ergonomía física y cognitiva. Dado que el dispositivo está destinado a sesiones de aprendizaje que pueden extenderse por horas, es imperativo mitigar la fatiga musculoesquelética y maximizar la eficiencia de la interacción. Se adoptó el enfoque de Diseño Centrado en el Usuario (UCD), analizando las capacidades biomecánicas de la población estudiantil objetivo.

V-C5.1. Análisis Antropométrico y Dimensionamiento:

Para definir las dimensiones generales de la carcasa (ancho y espesor), se tomaron como referencia los datos antropométricos del percentil 50 y 95 de la mano masculina y femenina adulta, visibles en la Figura 21. El objetivo fue garantizar un agarre de tipo “consola portátil” (*Handheld Grip*) cómodo para manos de diferentes tamaños.

Se estableció un ancho total de consola que permite que las eminencias tenares (la base del pulgar) descansen sobre los bordes laterales sin forzar una abducción excesiva de las muñecas, distribuyendo el peso del dispositivo en las palmas.



HAND DATA	MEN			WOMEN			CHILDREN			
	2.5% tile	50.% tile	97.5% tile	2.5% tile	50.% tile	97.5% tile	6 yr.	8 yr.	11 yr.	14 yr.
hand length	6.8	7.5	8.2	6.2	6.9	7.5	5.1	5.6	6.3	7.0
hand breadth	3.2	3.5	3.8	2.6	2.9	3.1	2.3	2.5	2.8	—
3d. finger lg.	4.0	4.5	5.0	3.6	4.0	4.4	2.9	3.2	3.5	4.0
dorsum lg.	2.8	3.0	3.2	2.6	2.9	3.1	2.2	2.4	2.8	3.0
thumb length	2.4	2.7	3.0	2.2	2.4	2.6	1.8	2.0	2.2	2.4

Figura 21. Datos antropométricos (Henry Dreyfuss) utilizados para el dimensionamiento ergonómico del chasis.

V-C5.2. Biomecánica del Pulgar y Distribución de Controles:

La ubicación de los pulsadores y el joystick se determinó mediante el estudio del "Arco de Alcance Funcional" del dedo pulgar, cuyo diagrama de zonas se analiza en la Figura 22. Se optó por una distribución asimétrica funcional para la interacción bimanual:

Zona de Navegación (Izquierda): El Joystick se ubicó en el centro del radio de barrido natural del pulgar izquierdo, permitiendo movimientos rotacionales de 360 grados sin necesidad de extender forzadamente la falange distal, previniendo la fatiga en los tendones extensores.

Zona de Acción (Derecha): Los pulsadores principales se situaron dentro de la zona de confort inmediata (45 mm a 60 mm desde la articulación carpometacarpiana). Se definió un espaciado (*Pitch*) de 12 mm entre centros de botones; esta distancia es crítica para evitar la pulsación accidental de dos comandos simultáneos, permitiendo una alternancia rápida entre acciones.

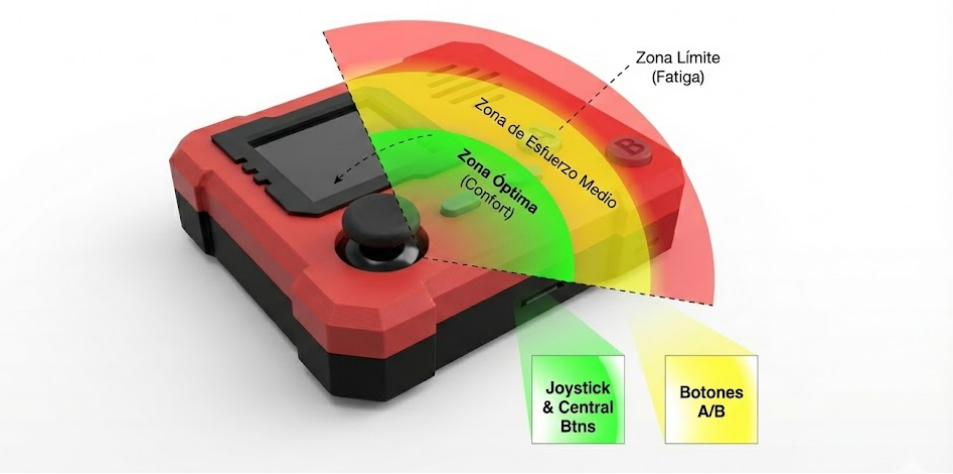


Figura 22. Análisis de las zonas de alcance funcional del pulgar sobre la interfaz de control de la consola.

V-C5.3. Ergonomía Visual:

La pantalla TFT se posicionó centralmente para mantener la simetría visual. Sin embargo, se consideró la "Línea de Visión Normal" (*Normal Line of Sight*). El marco de la pantalla incluye un ligero biselado interno para evitar sombras proyectadas sobre los bordes del área activa y asegurar que el código sea legible en ángulos de visión de hasta 60° respecto a la normal, compensando los reflejos de la iluminación cenital típica de los laboratorios universitarios.

VI-C6. Diseño de la Carcasa: Prototipado (V1) y Optimización Final (V2): Concluido el modelado digital preliminar, se procedió a la materialización del primer prototipo físico. El objetivo de esta fase experimental no fue obtener un acabado estético final, sino validar la congruencia geométrica (*fit check*) entre los componentes electrónicos reales y las cavidades diseñadas, así como evaluar la ergonomía inicial de la interfaz.

V.C6.1 Diseño de la Carcasa: Versión 1 (Prototipado):

V.C6.1.1 Manufactura Aditiva Inicial:

Para esta iteración, se utilizó la tecnología de Modelado por Deposición Fundida (FDM). Se exportaron los modelos sólidos a formato estereolitográfico (.STL) y se procesaron en el software de laminado (*Slicer*) con una altura de capa rápida de 0.3 mm. El material seleccionado fue PLA (Ácido Poliláctico) genérico debido a su bajo costo y facilidad de impresión, ideal para iteraciones desechables.

V.C6.1.2 Análisis de Integración e Interferencias:

Tras el ensamblaje de los componentes electrónicos en la carcasa V1, se identificaron desviaciones dimensionales críticas y excesos en el diseño, documentados en la Figura 23:

- **Interferencia en Puertos:** El orificio para el puerto USB-C del ESP32 resultó insuficiente. La expansión del material durante la extrusión cerró la abertura, impidiendo la conexión del cable de carga.

- **Fricción en Actuadores:** Los botones de acción presentaron un fenómeno de atascamiento (*sticking*) dentro de sus guías debido a la rugosidad de las paredes impresas.
- **Exceso de Componentes:** Se observó que la inclusión de dos botones auxiliares pequeños y un módulo de antena externa aumentaba la complejidad del ensamblaje sin aportar valor pedagógico significativo al objetivo central del proyecto.

Figura 23. Fallo de validación dimensional en el prototipo V1.

V-C6.1.3. Evaluación Estructural:

Se sometió el chasis a pruebas de torsión manual moderada. Se detectó que las torres de fijación (*bosses*) para los tornillos M3 eran estructuralmente débiles; al aplicar el torque de apriete, los postes sufrieron delaminación (separación de capas) en su base, **como se evidencia en la Figura 24**, debido a la concentración de estrés en los ángulos rectos.

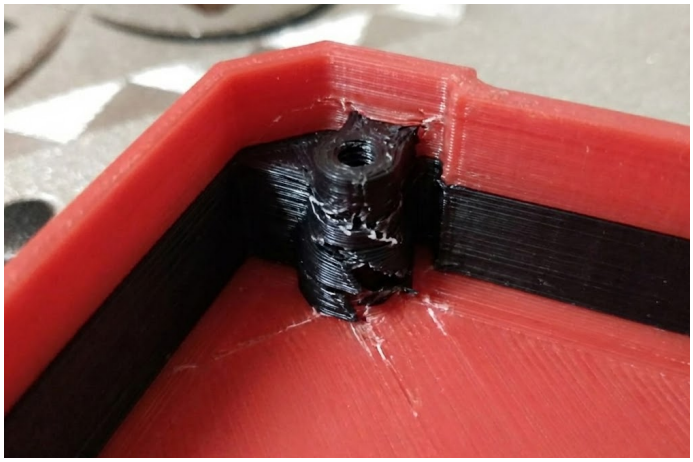


Figura 24. Fallo estructural por delaminación en las torres de fijación debido a estrés mecánico.

V-C6.1.4. Conclusiones de la Iteración V1:

El prototipo V1 cumplió su función de validación volumétrica. Sin embargo, evidenció la necesidad imperativa de realizar una compensación dimensional por la manufactura FDM, reforzar los anclajes y, fundamentalmente, simplificar la interfaz eliminando componentes innecesarios (antena y botones extra) para la versión final.

V-C6.2. Mejoras y Optimización Final (V2):

Basado en el análisis de fallos del primer prototipo, se ejecutó una segunda y definitiva iteración de diseño (V2). El objetivo central de esta etapa fue la optimización funcional y la çompensación dimensional", generando un dispositivo robusto y listo para el uso en laboratorio.

V-C6.2.1. Optimización Funcional y Simplificación:

Se realizaron cambios geométricos significativos para adaptar la carcasa a los componentes definitivos:

- **Redimensionamiento de Pantalla:** Se ajustó la ventana del display para coincidir exactamente con el área activa de la pantalla TFT, eliminando los bordes ciegos presentes en la V1.
- **Reubicación de Ventilación:** La rejilla de ventilación y salida de audio (*buzzer*) se trasladó de la posición superior a la zona frontal inferior, mejorando la acústica y la estética del panel frontal.
- **Eliminación de Periféricos:** Se suprimieron los alojamientos para la antena externa y los botones auxiliares, limpiando la interfaz de usuario.

V-C6.2.2. Mejora en la Conectividad (Borneras):

Uno de los cambios técnicos más relevantes en la V2 fue la sustitución de los conectores hembra directos. Se integraron borneras de conexión a tornillo para los puertos de expansión. Esta modificación garantiza una conexión mecánica mucho más estable y duradera para los cables de los sensores, evitando las desconexiones accidentales comunes en los *headers* convencionales durante las prácticas educativas.

V-C6.2.3. Gestión de Tolerancias y Holguras:

El problema de fricción en los botones principales se abordó rediseñando las guías de deslizamiento con un "offset" negativo de 0.4 mm, detalle que se compara transversalmente en la Figura 25. Adicionalmente, se amplió la ventana del puerto USB-C para permitir la inserción de cables de carga robustos sin obstrucciones.

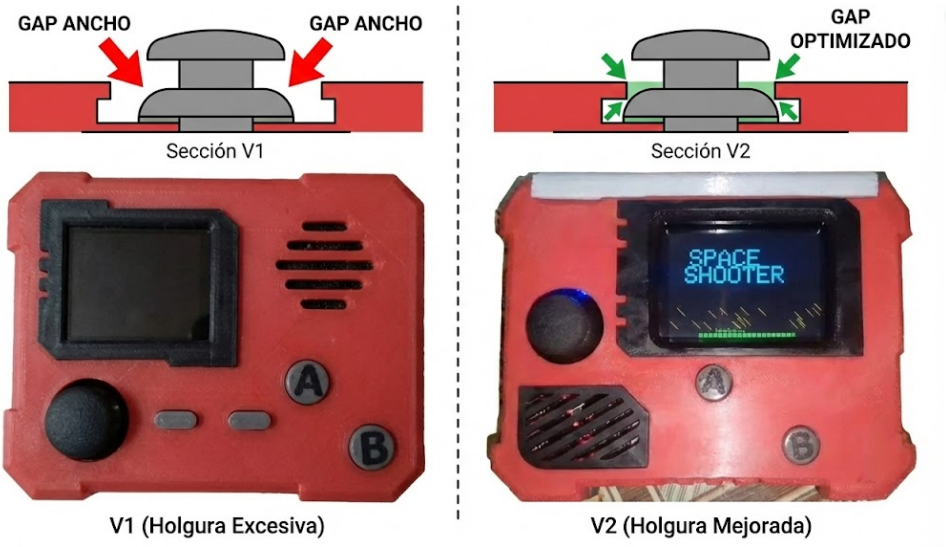


Figura 25. Sección transversal comparativa de la holgura (*gap*) entre el actuador y la carcasa (V1 vs V2).

V-C6.2.4. Refuerzo Estructural y Acabado Final:
Para mitigar la delaminación, se modificó la geometría base de las torres de tornillos, sustituyendo los ángulos rectos por filetes (redondeos).

Estéticamente, se aplicaron chaflanes de $45°$ en los bordes de contacto para mejorar la ergonomía (”Hand-held comfort”) y se implementó un sistema de cierre con tornillería hexagonal vista, facilitando el mantenimiento. Finalmente, se añadió un sistema de etiquetado con código de colores en las nuevas borneras para identificar rápidamente los rieles de alimentación y datos.

VI-C7. Selección de Materiales de Manufactura Aditiva: La elección del material para la envolvente de un dispositivo electrónico no debe basarse únicamente en criterios estéticos o económicos. En este proyecto, se realizó un análisis comparativo de las propiedades termomecánicas de los termoplásticos más comunes en la industria del prototipado rápido: el Ácido Poliláctico (PLA) y el Tereftalato de Polietileno Glicol (PETG). La selección final obedeció a una arquitectura híbrida, aprovechando las fortalezas específicas de cada polímero para diferentes componentes del ensamblaje.

V-C7.1. Carcasa Exterior: Ácido Poliláctico Reforzado (PLA+):
Para las carcasas superior e inferior (*Frontplate* y *Backplate*), se seleccionó filamento PLA+. La justificación técnica reside en su alto Módulo de Young (aproximadamente 3.5 GPa), lo que le confiere una rigidez estructural superior. Esto es crítico para mantener la planitud de las superficies grandes y evitar la flexión del chasis al presionar los botones.

Además, el PLA presenta un coeficiente de contracción térmica bajo, garantizando que las tolerancias dimensionales logradas en la impresión se mantengan fieles al diseño CAD, fundamental para el encaje preciso de la pantalla y los puertos.

V-C7.3. Matriz de Propiedades Comparativas:

Para fundamentar la selección de materiales, se consolidaron las propiedades mecánicas y térmicas críticas en la Tabla XV, cuyos valores sirvieron como parámetros de entrada para las simulaciones de tolerancia y resistencia.

Tabla XV
COMPARATIVA DE PROPIEDADES TERMOMECÁNICAS DE LOS MATERIALES SELECCIONADOS.

Propiedad	**PLA+**	**PETG**	**Impacto en el Diseño**
Temperatura (T_g)	$\sim 60°$C	$\sim 80°$C	Estabilidad térmica cerca del procesador.
Módulo de Young	$\sim 3{,}5$ GPa	$\sim 2{,}1$ GPa	Rigidez de la carcasa externa.
Ductilidad	Baja (Frágil)	Alta	Resistencia de los clips de batería.
Contracción	$< 0{,}3\,\%$	$\sim 0{,}8\,\%$	Precisión dimensional en puertos.

VI-C8. Parámetros de Impresión 3D: Carcasa Superior e Inferior:

V-C8.1. Parámetros de Impresión 3D: Carcasa Superior:

La manufactura de la carcasa superior (*Frontplate*) presenta el desafío de equilibrar la resistencia mecánica con un acabado superficial impecable, ya que es la interfaz directa con el usuario. Para lograr esto, se definió una estrategia de laminado (*Slicing*) específica que maximiza la resolución en el eje Z y refuerza las zonas sometidas a la presión de los pulsadores.

V-C8.1.1. Estrategia de Perímetros (Shells):

A diferencia de los modelos decorativos estándar, esta pieza estructural se configuró con un espesor de pared de 1.2 mm, equivalente a tres líneas de perímetro para una boquilla de 0.4 mm. Esta decisión técnica cumple una doble función crítica: por un lado, proporciona suficiente material para realizar un post-procesado manual de lijado y pulido sin riesgo de perforar la superficie ni exponer el relleno interno; por otro lado, incrementa significativamente el momento de inercia de la sección transversal, otorgando la rigidez torsional necesaria para evitar que la carcasa se flexione o cruja bajo la presión operativa de los pulsadores.

V-C8.1.2. Topología del Relleno (Infill):

Se descartó el patrón de relleno rectilíneo en favor del patrón Giroide (*Gyroid*) al 20 % de densidad, el cual se visualiza en la F
Matemáticamente, el Giroide es una Superficie Mínima Triplemente Periódica (TPMS) que es isotrópica; es decir, ofrece la misma resistencia a la compresión en todas las direcciones del espacio (X, Y, Z). Esto es fundamental para absorber las cargas cíclicas generadas por el tecleo constante sin sufrir fatiga mecánica ni deslaminación interna.

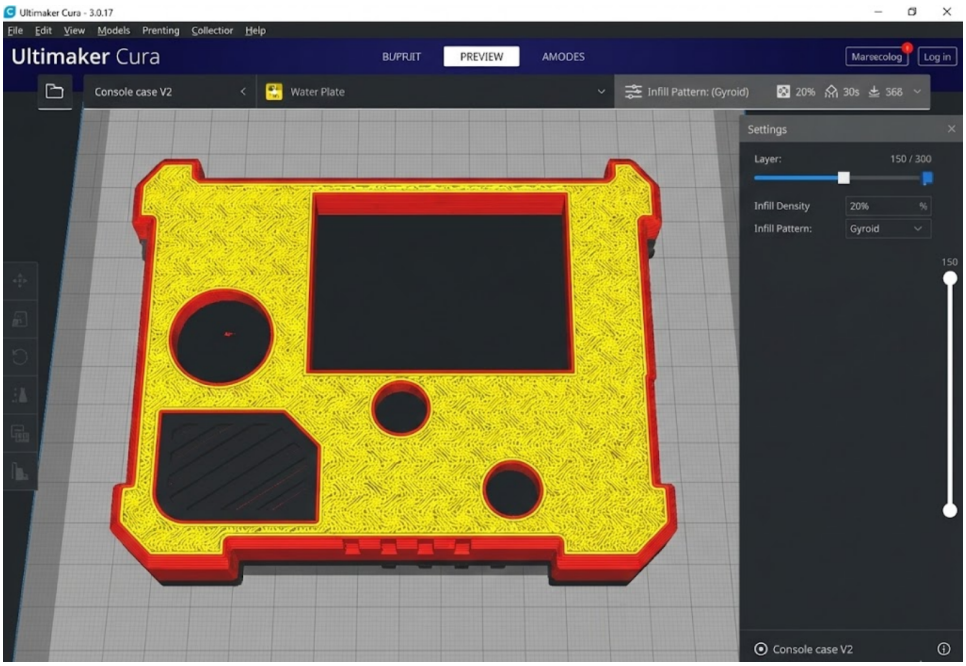


Figura 26. Vista previa del laminado mostrando la estructura interna isotrópica del patrón Giroide.

V-C8.1.3. Tabla de Parámetros de Manufactura:

Para garantizar la reproducibilidad del acabado superficial, se establecieron los parámetros operativos detallados en la Tabla XVI, validados experimentalmente para el material PLA+.

Tabla XVI
PARÁMETROS DE IMPRESIÓN VALIDADOS PARA LA CARCASA SUPERIOR (PLA+).

Parámetro	**Valor**	**Justificación Técnica**
Altura de Capa	0.16 mm	Alta resolución para suavidad al tacto.
Boquilla	0.4 mm	Estándar para detalles finos en textos.
Perímetros	3 (1.2 mm)	Integridad estructural post-lijado.
Relleno	20 % Giroide	Isotropía mecánica.
Temp. Extrusor	205°C	Fusión óptima para minimizar *stringing*.
Temp. Cama	60°C	Adhesión en placa PEI texturizada.
Velocidad Paredes	30 mm/s	Reducción de vibraciones (*ringing*).
Refrigeración	100 %	A partir de la capa 2 para voladizos.

V-C8.2. Parámetros de Impresión 3D: Carcasa Inferior:

La manufactura de la carcasa inferior (*Backplate*) obedece a requisitos funcionales distintos, priorizando la estabilidad dimensional bajo carga térmica y la resistencia a la tracción en los puntos de anclaje. En consecuencia, se estableció un perfil de impresión optimizado para la integridad estructural y la eficiencia de tiempo, sacrificando ligeramente la resolución vertical en zonas no visibles.

V-C8.2.1. Optimización Estructural y Velocidad:

Para reducir el tiempo de fabricación sin comprometer la solidez, se configuró una altura de capa de 0.24 mm. Esta decisión técnica se justifica por la naturaleza interna de la pieza; al aumentar la altura de capa, se reduce el número total de pasadas y se incrementa la adhesión entre capas (*bonding strength*) debido al mayor volumen térmico del plástico extruido. Asimismo, se incrementó el número de perímetros a cuatro líneas (1.6 mm de espesor), creando un chasis robusto capaz de soportar la tensión de los tornillos de cierre sin sufrir agrietamiento por estrés (*stress cracking*) alrededor de los orificios.

V-C8.2.2. Gestión de Voladizos y Soportes:

La geometría de la carcasa inferior incluye desafíos topológicos críticos, específicamente en las aberturas para el puerto USB-C y la ranura MicroSD. Para manufacturar estas zonas sin material de sacrificio excesivo, se activó la detección de "Puentes" (*Bridging*) en el laminador, configuración detallada en la Figura 27. Esta función aumenta el flujo del ventilador de capa al 100 % y reduce la velocidad de extrusión a 15 mm/s exclusivamente cuando la boquilla cruza el aire, permitiendo que el filamento se enfríe instantáneamente y mantenga la forma horizontal sin descuelgue gravitacional.

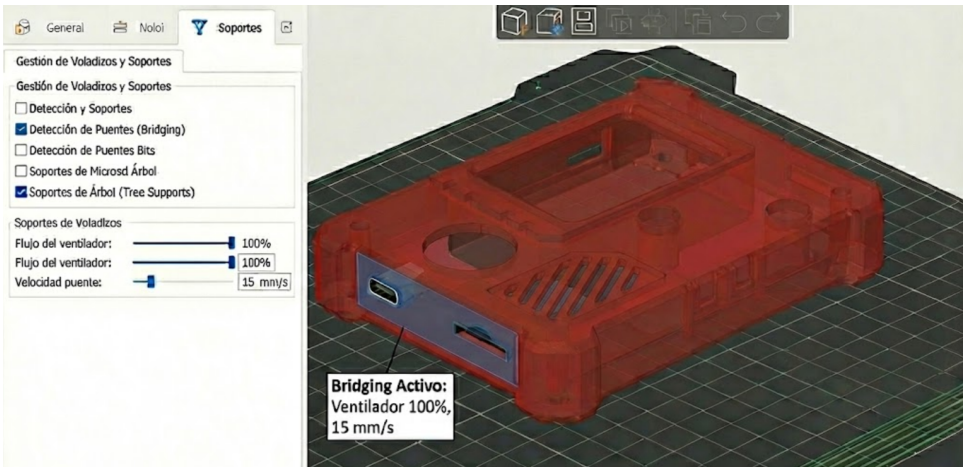


Figura 27. Configuración de puentes (*Bridging*) para la impresión correcta de los puertos sin soportes internos.

V-C8.2.3. Tabla de Parámetros de Manufactura:

Se consolidan a continuación los parámetros técnicos específicos para la base del dispositivo, enfocados en la durabilidad mecánica, en la Tabla XVII.

Tabla XVII
PARÁMETROS DE IMPRESIÓN VALIDADOS PARA LA CARCASA INFERIOR.

Parámetro	**Valor**	**Justificación Técnica**
Altura de Capa	0.24 mm	Eficiencia de tiempo y adhesión térmica.
Boquilla	0.4 mm	Estándar.
Perímetros	4 (1.6 mm)	Refuerzo para torque de tornillos.
Relleno	40 % Rejilla	Alta densidad para soporte estructural.
Temp. Extrusor	215°C	Mayor temperatura para fusión de capas.
Temp. Cama	60°C	Estabilidad dimensional.
Velocidad Puentes	15 mm/s	Enfriamiento rápido en voladizos (USB).
Soportes	Sí (Árbol)	Solo en la cama de la batería.

VI-C9. Selección e Integración Mecánica de Actuadores Comerciales: Para la interfaz de control, se optó por la integración de componentes electromecánicos comerciales (COTS - *Commercial Off-The-Shelf*) en lugar de fabricar actuadores impresos. Esta decisión técnica se fundamenta en la necesidad de garantizar una respuesta táctil (*Haptics*) precisa y una vida útil superior a los 100,000 ciclos de actuación, prestaciones difíciles de alcanzar con mecanismos impresos en FDM debido a la fatiga del material.

V-C9.1. Módulo de Navegación (Joystick Analógico):
Se seleccionó un módulo de joystick estándar de dos ejes con pulsador central integrado. El desafío de ingeniería consistió en diseñar el alojamiento en la carcasa superior para garantizar un recorrido mecánico libre de obstrucciones.

Se dimensionó una apertura circular de 26 mm de diámetro con un biselado interno de $45°$. Esta geometría cónica es crítica: permite que el vástago del joystick alcance su inclinación máxima (aproximadamente $30°$) sin chocar contra los bordes de la carcasa, evitando "zonas muertas" mecánicas en la lectura de los sensores. La fijación se realizó mediante tornillería interna, asegurando que el módulo no se desplace ante la presión vectorial del pulgar.

V-C9.2. Pulsadores de Acción y Gatillos:
Para los botones de comando y los gatillos superiores, se utilizaron pulsadores momentáneos de montaje en panel. El diseño CAD contempló la aplicación de Tolerancias de Ajuste específicas para la impresión 3D.

Dado que el plástico FDM tiende a contraerse cerrando los orificios (fenómeno de *shrinking*), se aplicó una holgura diametral de $+0{,}4$ mm sobre la medida nominal de los botones. Esto permite una inserción suave del componente sin necesidad de mecanizado posterior, mientras que la fijación final se asegura mediante las tuercas de retención o el anclaje mecánico del propio interruptor.

V-C9.3. Especificaciones de Integración:
La Tabla XVIII resume las tolerancias aplicadas al diseño CAD para alojar estos componentes externos correctamente tras la contracción térmica del PLA.

Tabla XVIII
TOLERANCIAS DE DISEÑO PARA INTEGRACIÓN DE HARDWARE COMERCIAL.

Componente	**Medida Real**	**Medida CAD**	**Tipo de Ajuste**
Joystick (Apertura)	25.0 mm	26.0 mm	Holgura para recorrido angular.
Botones (A/B)	12.0 mm	12.4 mm	Ajuste deslizante (*Clearance Fit*).
Tornillería M3	3.0 mm	3.2 mm	Paso libre para roscado.
Pantalla TFT (Marco)	50.0 mm	50.5 mm	Holgura por expansión térmica.

VI-D. Fase de Desarrollo de Software Embebido

VI-D1. Preparación del entorno de desarrollo (IDE): La programación de firmware para microcontroladores de 32 bits difiere sustancialmente del desarrollo de software de escritorio. En este proyecto, el código fuente escrito en C++ debe ser traducido desde la arquitectura del ordenador anfitrión hacia el lenguaje de máquina específico de los núcleos Xtensa LX7 del ESP32-S3. Este proceso, conocido como çompilación cruzada"(*cross-compilation*), requiere una cadena de herramientas (*toolchain*) altamente especializada.

Con el objetivo de maximizar la accesibilidad pedagógica y facilitar la replicación del proyecto por parte de estudiantes con diversos niveles de experiencia, se seleccionó el *Arduino IDE* como entorno de desarrollo principal, tal como se representa en la Figura 28. A diferencia de ecosistemas profesionales que requieren configuraciones complejas, este entorno permite la gestión simplificada del *Board Support Package* (BSP) de Espressif a través de su Gestor de Tarjetas, garantizando la compatibilidad con el ESP32-S3 sin barreras de entrada técnicas. Además, el entorno soporta nativamente la compilación de C++ moderno y la integración de las bibliotecas del proyecto (como `TFT_eSPI` y las primitivas de FreeRTOS), centralizando la edición, compilación y carga del *firmware* en una interfaz estandarizada y ampliamente documentada en el ámbito educativo [31].

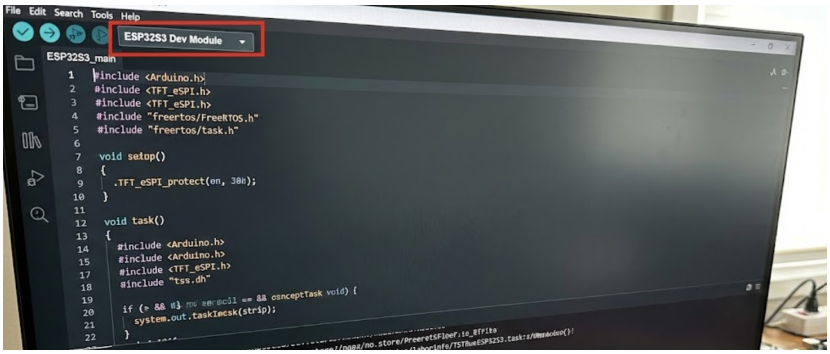


Figura 28. Proceso de desarrollo del entorno de programación en Arduino IDE.

La preparación del entorno exigió la configuración de parámetros críticos para el correcto funcionamiento de la consola didáctica:

1.- Frecuencia del procesador y bus: se forzó el reloj del núcleo a 240 MHz y la frecuencia de la memoria Flash a 80 MHz en modo QIO, asegurando el máximo rendimiento de lectura para la actualización de la pantalla TFT.

2.- Esquema de particiones (Partition Scheme): dado que la Programación Orientada a Objetos y las librerías gráficas compilan en binarios pesados, se modificó la tabla de particiones por defecto. Se asignó una partición de aplicación ampliada de 3MB para el código ejecutable y una partición de datos para el sistema de archivos no volátil, sacrificando el espacio de actualizaciones inalámbricas (OTA) que no es requerido en este prototipo educativo.

3.- Monitor Serial y Depuración: se configuró el puerto USB nativo del ESP32-S3 mediante directivas de compilación y se estableció una velocidad de baudios de 115200 bps. Esto permite imprimir la traza de ejecución y los registros de pila (*Stack Trace*) en tiempo real, facilitando el diagnóstico de fallos de segmentación durante la instanciación dinámica de clases en C++.

VI-D2. Arquitectura del Firmware y Diseño de Interfaz (GUI): El sistema operativo de la consola "Code & Play" (Versión de Firmware v3.0.0) fue desarrollado sobre el framework de Espressif, optimizando el uso de los dos núcleos del ESP32-S3 para manejar simultáneamente el renderizado gráfico y la lógica de control. La interfaz gráfica de usuario (GUI) se diseñó bajo el paradigma de "navegación jerárquica", permitiendo el acceso intuitivo a las funciones mediante los controles físicos del Joystick y los botones A/B.

V-D2.1. Estructura del Menú Principal:
Al iniciar el sistema, tras la secuencia de arranque (*boot sequence*) que verifica la integridad del hardware, se presenta el Menú Principal dividido en tres módulos funcionales, visibles en la Figura 29:

- **Sensores:** Módulo de monitoreo en tiempo real para visualizar los datos de los puertos de entrada (analógicos y digitales).
- **Programador:** Entorno de ejecución para la lógica de programación visual.
- **Configuración:** Panel de administración del sistema para ajustes globales (Audio, Interfaz, Info).

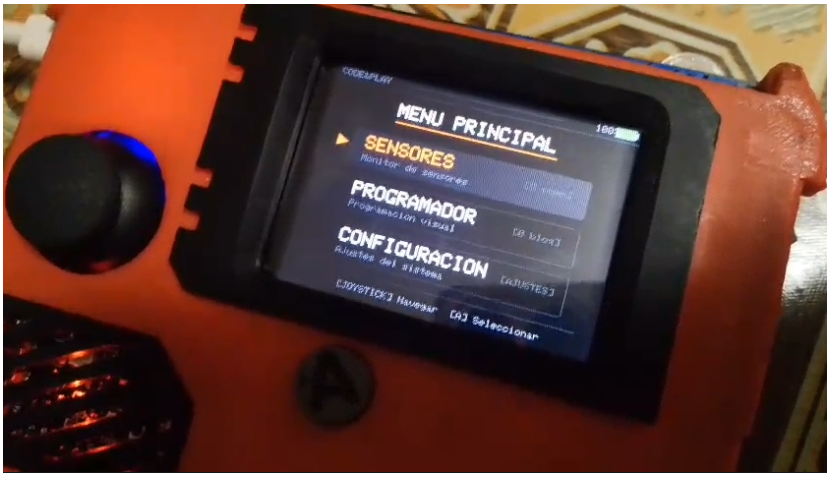


Figura 29. Interfaz del Menú Principal navegable mediante Joystick físico.

V-D2.2. Personalización y Experiencia de Usuario (UX):

El firmware integra características de usabilidad avanzadas. Se implementó un gestor de temas visuales que permite conmutar entre Modo Oscuro y Modo Claro en tiempo real.

Esta función responde a una necesidad ergonómica visual: el modo oscuro reduce el consumo de corriente de la pantalla y la fatiga visual en entornos de baja luz, mientras que el modo claro maximiza la legibilidad bajo la iluminación fluorescente intensa típica de los laboratorios.

VI-D3. Panel de Información del Sistema: Dentro del submenú de configuración, se incluye una herramienta de diagnóstico, ilustrada en la Figura 30, que reporta el estado del hardware. Según la telemetría del dispositivo mostrada en pantalla:

- **CPU:** ESP32-S3 operando a una frecuencia nominal de 240 MHz.
- **Memoria:** Monitoreo dinámico de la RAM libre y el uso de la memoria Flash interna.

Esta transparencia en los recursos permite al estudiante comprender las limitaciones físicas del microcontrolador con el que está trabajando.

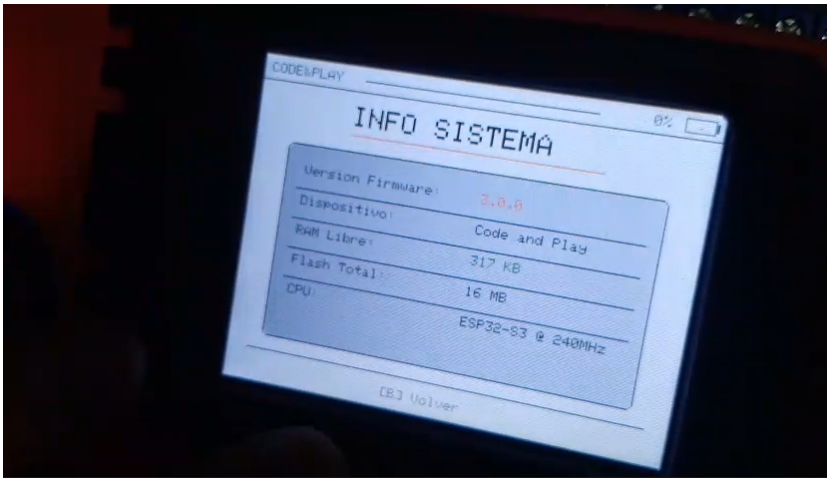


Figura 30. Pantalla de diagnóstico del sistema mostrando especificaciones del ESP32-S3.

VI-D4. Instalación e Integración de librerías y dependencias:

V-D4.1. Instalación de librerías y dependencias:

La filosofía de diseño del firmware se basó en abstraer el control de muy bajo nivel mediante bibliotecas comprobadas, permitiendo enfocar los esfuerzos de ingeniería en el desarrollo de la lógica de la consola y la innovación pedagógica. Por consiguiente, el proyecto se apoya en un conjunto de librerías altamente optimizadas para la arquitectura del microcontrolador. La integración de estas dependencias se gestionó de forma modular. Cada librería seleccionada resuelve una restricción de hardware específica, minimizando la carga de la Unidad Central de Procesamiento (CPU) para mantener el determinismo temporal del sistema.

V-D4.1.1. Subsistema Gráfico: Librería TFT_eSPI:

Para el control integral de la pantalla ILI9341, se seleccionó la librería `TFT_eSPI`, descartando alternativas de

propósito general. La justificación técnica de esta elección radica en su capacidad intrínseca para aprovechar el Acceso Directo a Memoria (DMA) [32], cuya arquitectura funcional se representa en la Figura 31.

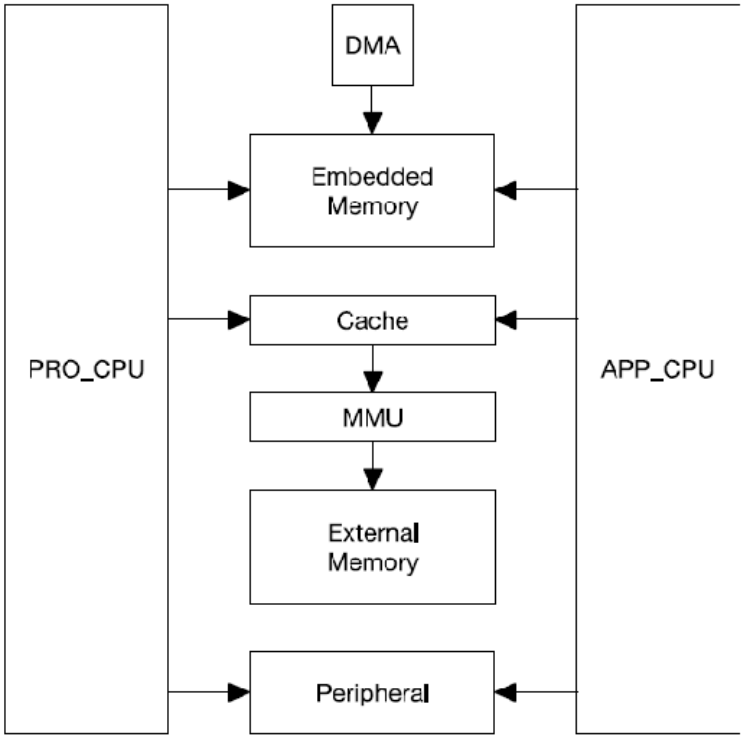


Figura 31. Diagrama de la arquitectura del sistema ESP32, destacando al controlador DMA como maestro independiente para el acceso a memoria y periféricos.

El DMA es un controlador de hardware independiente que transfiere bloques masivos de píxeles desde la memoria RAM del ESP32 directamente al bus SPI de la pantalla, sin intervención cíclica de la CPU. Esto permite que el microcontrolador ejecute la lógica de eventos o lea la matriz de botones simultáneamente mientras la pantalla se actualiza en segundo plano, mitigando severamente los cuellos de botella de renderizado.

V-D4.1.2. Subsistema de Almacenamiento: SD y FS:

Para la interconexión con el lector MicroSD, se implementaron las librerías nativas `SD.h` y `FS.h` (File System). Estas bibliotecas permiten inicializar el bus SPI secundario y manipular flujos de datos en archivos de texto plano, posibilitando que la consola lea los programas fuente desarrollados por los estudiantes bajo el formato FAT32 y almacene de manera persistente los registros de actividad.

V-D4.1.3. Sistema Operativo en Tiempo Real (RTOS) y Sincronización:

Aunque no constituye una librería externa sujeta a descarga, el proyecto hace un uso riguroso de la Interfaz de Programación de Aplicaciones (API) de FreeRTOS, la cual reside nativamente en el núcleo de ejecución del ESP32. Se emplearon primitivas de sincronización concurrente, específicamente Mutexes (*Mutual Exclusion*). La implementación de un Mutex fue mandatoria en la arquitectura del firmware, dado que el módulo MicroSD y la pantalla TFT comparten las mismas trazas físicas de reloj y datos del bus SPI principal. El Mutex garantiza exclusión mutua: si la tarea de gráficos está transmitiendo píxeles a la pantalla, el intento simultáneo de la tarea de

lectura de acceder a la MicroSD es bloqueado por microsegundos, previniendo la colisión de datos y la subsecuente corrupción del bus de comunicaciones.

V-D4.1.4. Interfaz de Entrada: Bounce2:

Para la gestión de los pulsadores mecánicos independientes (Botones de acción y gatillos), se integró la librería `Bounce2`. A diferencia del escaneo matricial, esta dependencia permite instanciar objetos individuales para cada pin GPIO, aplicando algoritmos de filtrado digital de rebotes (*debouncing*) basados en temporizadores de milisegundos. Esto garantiza la captura de transiciones de estado limpias sin bloquear el procesador y sin necesidad de componentes pasivos externos (filtros RC). Para sistematizar estas dependencias de cara a la trazabilidad y reproducibilidad del entorno de desarrollo, en la Tabla XIX se clasifica el ecosistema de software integrado.

Tabla XIX
MATRIZ DE DEPENDENCIAS Y LIBRERÍAS DEL FIRMWARE, DETALLANDO EL ORIGEN Y LA JUSTIFICACIÓN TÉCNICA DE CADA UNA.

Librería / API	**Origen**	**Justificación Técnica y Funcionalidad**
`TFT_eSPI`	Bodmer	Controlador gráfico acelerado por hardware (DMA) para displays SPI (ILI9341).
`SD.h` / `FS.h`	Core ESP32	Gestión del protocolo SPI estandarizado e interfaz con el sistema de archivos FAT32.
`FreeRTOS`	Espressif IDF	Planificador de tareas preventivo, semáforos y exclusión mutua para buses compartidos.
`Bounce2`	T. Ouellet	Gestión de cambios de estado y *debouncing* por software para pulsadores conectados a GPIO directo.

V-D4.2. Integración de Librerías y Gestión de Almacenamiento:

Para finalizar la implementación del sistema, se estructuró el firmware sobre un ecosistema de librerías optimizadas para el ESP32-S3, garantizando la eficiencia en la gestión de memoria y la persistencia de datos requerida para el entorno educativo.

V-D4.2.1. Controladores de Pantalla y Gráficos:

La renderización de la interfaz gráfica (GUI) se realiza mediante la implementación de librerías de alto nivel (como TFT_eSPI) que utilizan aceleración por hardware. Estas librerías aprovechan los canales DMA (*Direct Memory Access*) del microcontrolador, permitiendo transferir datos a la pantalla sin bloquear el CPU. Esto resulta en transiciones fluidas entre menús y una tasa de refresco estable, eliminando el parpadeo (*flickering*).

V-D4.2.2. Persistencia de Datos (NVS):

Se programó el uso del Almacenamiento No Volátil (NVS - *Non-Volatile Storage*) en la partición Flash del ESP32. Esta implementación permite guardar estructuras de datos tipo “clave-valor”, garantizando que la configuración personalizada del usuario (como el tema visual o calibración) se conserve tras reiniciar el dispositivo.

V-D4.2.3. Sistema Operativo (FreeRTOS):

El firmware se ejecuta sobre FreeRTOS, distribuyendo la carga entre los núcleos:

- Core 0: Gestión de sensores y tareas de fondo.
- Core 1: Renderizado de la interfaz de usuario (UI) y respuesta a interrupciones físicas.

VI-E. Fase de Integración y Ensamblaje

VI-E1. Ensamblaje de la Placa y Gestión del Cableado: La fase de integración electrónica representa el hito donde los componentes discretos se consolidan en una unidad funcional. Debido a la alta densidad de conexiones requerida por el diseño (pantalla, joystick y el banco de expansión), se optó por una estrategia de ensamblaje mixto que prioriza la robustez mecánica y la facilidad de mantenimiento.

VI-E2. Implementación del Banco de Conexiones (Interfaz Frontal): El rasgo distintivo de la arquitectura física de la consola es su matriz lineal de borneras de tornillo (*PCB Screw Terminals*) de color azul y paso 5.08 mm, ubicadas en el borde frontal de la placa base, las cuales se observan en detalle en la Figura 32.

A diferencia de los pines tipo “Dupont” tradicionales en kits educativos, que sufren desconexiones por vibración o desgaste, estas borneras garantizan una sujeción mecánica firme para cables de calibre AWG 24 a AWG 18. Esto permite al estudiante conectar sensores, actuadores y circuitos externos con seguridad industrial, asegurando que la conexión eléctrica (especialmente en los puertos de potencia) no falle durante la manipulación del dispositivo en el laboratorio.

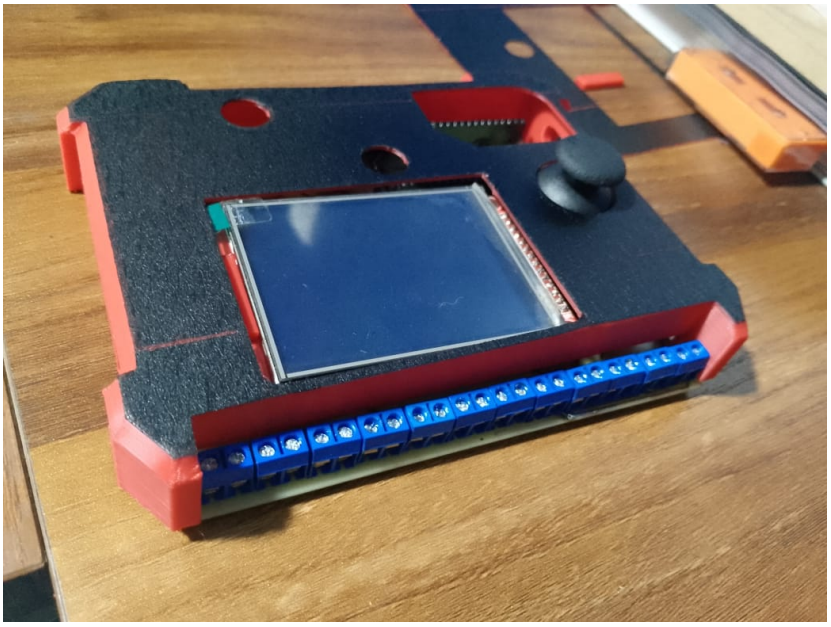


Figura 32. Detalle del banco de conexiones frontal mediante borneras de tornillo para sujeción robusta.

VI-E3. Integración Modular del Núcleo: El módulo ESP32-S3 no se soldó directamente a la placa base. Se implementaron tiras de zócalos hembra (*female headers*) que actúan como interfaz de montaje. Esta decisión permite reemplazar el microcontrolador en caso de daño accidental por sobretensión o cortocircuito sin necesidad de de soldar toda la placa, extendiendo la vida útil del equipo.

VI-E4. Conexión de Periféricos: La pantalla TFT se encuentra fijada mecánicamente a la carcasa superior. Para su integración con el PCB principal (alojado en la carcasa inferior), se prescindió del uso de cableado flexible. En su lugar, se implementó un sistema de conexión directa mediante zócalos estratégicamente alineados. De este modo, la interconexión eléctrica se realiza de forma automática durante el ensamblaje: al cerrar la consola y acoplar ambas

mitades de la carcasa, los conectores de la pantalla encajan con precisión en los receptáculos correspondientes de la placa base. Esta arquitectura modular elimina la tensión mecánica en los puntos de soldadura, evita el riesgo de atrapar cables al cerrar el dispositivo y agiliza significativamente el montaje y diagnóstico de la consola.

VI-F. Fase de Validación y Pruebas Técnicas

VI-F1. Protocolos de Validación y Pruebas: Para certificar que el prototipo cumple con los requisitos planteados al inicio de esta metodología, se diseñó un plan de validación estructurado en tres niveles. Estos protocolos definen las métricas y las condiciones experimentales bajo las cuales se evaluará el dispositivo en el capítulo de Resultados.

V-F.1.1. Protocolo de Pruebas Eléctricas y Térmicas:
Se estableció un procedimiento para medir la autonomía real del sistema. La prueba consiste en ejecutar un script de "estrés" que mantiene la pantalla al 100 % de brillo y el procesador realizando cálculos flotantes continuos, midiendo la curva de descarga de la batería hasta el corte por bajo voltaje. Simultáneamente, se monitoreará la temperatura del encapsulado del ESP32 mediante una termocupla externa para verificar la eficiencia de las rejillas de ventilación pasiva.

V-F.1.2. Protocolo de Verificación Funcional (Unit Testing):
Se definió una lista de comprobación (*Checklist*) para validar individualmente cada subsistema. Se probará la linealidad del Joystick en los ejes X/Y, la precisión del ADC en los puertos de sensores y la integridad de datos en la tarjeta MicroSD, asegurando que no existan fallos de hardware latentes.

V-F.1.3. Protocolo de Validación Experimental: Prácticas de Laboratorio:
Para dar cumplimiento al tercer objetivo específico,se estableció un protocolo de validación integral que combina la verificación de éxito técnico con evaluaciones de percepción del usuario mediante encuestas de usabilidad. Se determina la validez del dispositivo demostrando que permite la ejecución satisfactoria de una secuencia de cinco ejercicios de complejidad incremental:

1. **Práctica 1 - Fundamentos y Control Básico:** Introducción a los componentes electrónicos básicos, exploración de la interfaz del dispositivo *Code and Play* y ejecución de los primeros ejercicios para el control de LEDs.
2. **Práctica 2 - Lógica y Programación por Bloques:** Utilización de la interfaz de bloques de programa para la implementación de estructuras condicionales y la ejecución simultánea de acciones.
3. **Práctica 3 - Diagnóstico y Pruebas de Hardware:** Navegación y uso del menú de pruebas de componentes del dispositivo, verificando el correcto estado y acople de los periféricos.
4. **Práctica 4 - Integración de Sensores:** Programación orientada a la lectura y procesamiento de datos provenientes de sensores utilizando el entorno del dispositivo *Code and Play*.
5. **Práctica 5 - Variables y Señales Analógicas:** Implementación de bloques de almacenamiento en variables, lectura de valores desde componentes analógicos y generación de salidas (como el control del bloque *Buzzer*).

La ejecución exitosa de estas cinco prácticas servirá como evidencia empírica de que la consola cumple con los requisitos funcionales y pedagógicos planteados.

Con la definición de estos protocolos de validación y la integración final del hardware y firmware, se da por concluida la fase metodológica de desarrollo. El dispositivo se encuentra operativo y en condiciones óptimas para ser sometido a la fase experimental, cuyos datos, mediciones de rendimiento y evidencia de funcionamiento en las prácticas pedagógicas serán expuestos y analizados en detalle en el capítulo de Resultados.

VI-G. Implementación

Este capítulo describe el proceso técnico y secuencial ejecutado para materializar el diseño teórico de la consola educativa en un dispositivo físico completamente funcional. Se detalla la construcción del hardware, el despliegue del software embebido en el microcontrolador y la integración electromecánica final del equipo.

VI-G1. Construcción y Ensamblaje de Hardware:

V-G1.1. Protocolo de Pruebas Eléctricas y Térmicas:

El proceso de implementación física inició con la manufactura de la placa de circuito impreso (PCB). Se procedió a la soldadura de los componentes de montaje superficial (SMD) y de orificio pasante (*Through-Hole*). Se prestó especial atención a la fijación de las borneras frontales y los zócalos hembra del ESP32-S3, garantizando mediante pruebas de continuidad que no existieran puentes de estaño que pudieran generar cortocircuitos, proceso que se ilustra en la Figura 33.

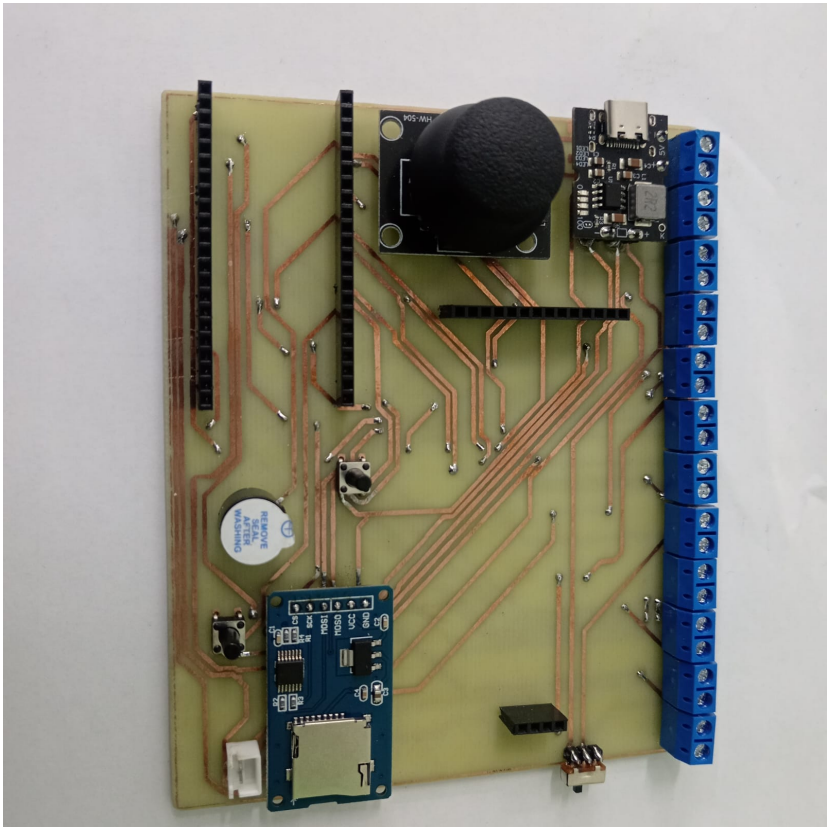


Figura 33. Proceso de soldadura y verificación de componentes en la placa base.

V-G1.2. Protocolo de Pruebas Eléctricas y Térmicas:

Posteriormente, se integraron los actuadores comerciales en la carcasa superior impresa en 3D (PLA+). El joystick

analógico y los dos botones de acción se fijaron en sus respectivas cavidades utilizando las holguras de ajuste planificadas y tornillería M3. La pantalla TFT se aseguró utilizando un marco en la carcasa superior como se muestra en la Figura 34.

Figura 34. Montaje de la pantalla TFT y controles mecánicos en la cara interna de la carcasa superior.

VI-G2. Despliegue del Software Embebido:

V-G2.1. Protocolo de Pruebas Eléctricas y Térmicas:

Para la implementación del software, se conectó el ESP32-S3 al ordenador anfitrión mediante el puerto USB-C nativo. Utilizando el entorno de desarrollo, se configuró una tabla de particiones personalizada para asignar mayor espacio al código ejecutable. El código fuente, estructurado en C++, fue compilado y transferido a la memoria Flash del microcontrolador, confirmando la carga exitosa a través del monitor serial, tal como se documenta en la Figura 35.

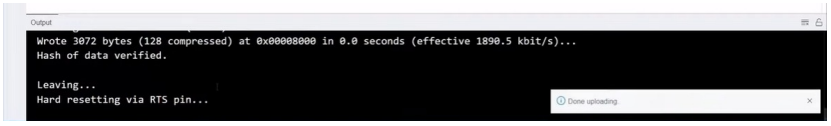


Figura 35. Transferencia del firmware compilado en C++ a la memoria Flash del ESP32-S3.

V-G2.2. Protocolo de Pruebas Eléctricas y Térmicas:

Como paso complementario, se formateó una tarjeta MicroSD bajo el estándar FAT32. Dentro de esta unidad lógica, se crearon los directorios raíz necesarios para que el sistema operativo de la consola pueda almacenar los scripts generados por los usuarios, manteniendo la estructura de datos requerida para las prácticas de programación.

VI-G3. Integración Electromecánica Final: La última etapa consistió en la interconexión de la electrónica de control con la fuente de alimentación. Se conectó la celda de Polímero de Litio (LiPo) al módulo de regulación térmica TP4056 y se enlazaron los cables de datos desde la carcasa superior hacia la placa base. Finalmente, se

cerraron ambas mitades del chasis asegurándolas firmemente.

Como se observa en la Figura 36, al presionar el interruptor general, el sistema ejecutó satisfactoriamente su secuencia de arranque (*Boot Sequence*), inicializando la pantalla y desplegando el menú principal. Este hito dio por concluida la fase de implementación, dejando el equipo completamente habilitado para la fase de validación experimental y las pruebas con usuarios.

Figura 36. Prototipo final ensamblado y ejecutando la interfaz gráfica del sistema operativo.

VII. Resultados

VII-A. Validación del Funcionamiento de la Consola

En estricto cumplimiento con los parámetros de diseño, se evaluó el desempeño del prototipo final basándose en los cuatro criterios fundamentales de validación de una consola portátil: autonomía, ergonomía, usabilidad técnica y tiempo de respuesta.

VII-A1. Autonomía Energética y Comportamiento Térmico: Se sometió la consola a una prueba de descarga continua ejecutando un bucle de renderizado gráfico con el brillo de pantalla al 100 %, cuyo comportamiento se detalla en la Figura 37.

- **Duración Total:** El dispositivo operó de manera estable durante 2 horas y 12 minutos antes de que el circuito de protección BMS (*Battery Management System*) cortara la alimentación al alcanzar los 3.0V.
- **Estabilidad de Voltaje:** La regulación LDO mantuvo el riel de 3.3V estable durante el 95 % del ciclo, garantizando que el procesador no sufriera reinicios inesperados (*Brown-out resets*).

Tras una hora de operación intensiva, se registraron $41{,}5°\text{C}$ en el procesador y $26{,}0°\text{C}$ en la carcasa exterior.

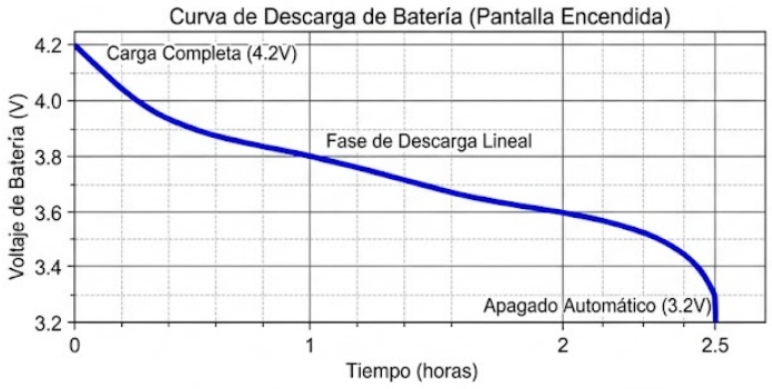


Figura 37. Curva de descarga de la batería bajo carga de trabajo continua.

VII-A2. Ergonomía y Usabilidad Física: La diferencia térmica registrada valida el diseño de ventilación pasiva, asegurando que el dispositivo sea ergonómicamente confortable. Adicionalmente, la distribución asimétrica del joystick y los dos botones de acción demostró facilitar una usabilidad bimanual natural. Las holguras ajustadas en la versión final de la carcasa eliminaron la fricción mecánica, permitiendo una interacción táctil fluida que cumple con los estándares antropométricos exigidos para una herramienta de aprendizaje prolongado.

VII-A3. Tiempo de Respuesta y Rendimiento del Sistema: La implementación de la transferencia por Acceso Directo a Memoria (DMA) permitió mantener una tasa de refresco de pantalla estable de 35 FPS. Esta fluidez es crítica para el tiempo de respuesta, eliminando el retardo visual (*input lag*) entre la pulsación de los controles y la acción en la interfaz. La distribución de memoria RAM durante este proceso se consolida en la Tabla XX.

Tabla XX
DISTRIBUCIÓN DE MEMORIA SRAM DURANTE LA EJECUCIÓN INTENSIVA.

Recurso	**Cantidad (KB)**	**Porcentaje**
RAM Total (ESP32-S3)	320 KB	100 %
Reservado Sistema (Kernel/WiFi)	90 KB	28 %
Reservado Gráficos (Buffers)	45 KB	14 %
Libre para Usuario (Heap)	185 KB	58 %

VII-B. Validación por los Usuarios

Para verificar la funcionalidad educativa del prototipo desde la perspectiva del estudiante, la validación se dividió en dos etapas: la comprobación técnica de las prácticas de laboratorio y la evaluación subjetiva mediante un instrumento de recolección de datos (encuesta).

VII-B1. Validación a través de Prácticas Pedagógicas: Se ejecutó la secuencia de cinco laboratorios definidos previamente. El propósito de esta fase fue evaluar cómo la consola abstrae la complejidad de la programación de hardware. A continuación, se detalla el procedimiento y el resultado técnico de cada práctica:

- **Gráfico Práctica 1:** Para la Práctica 1 se realizó la evaluación de la siguiente pregunta: "¿Qué nivel de dificultad experimentó al utilizar la interfaz del dispositivo *Code and Play* para identificar los componentes electrónicos básicos y controlar el estado de los LEDs?"

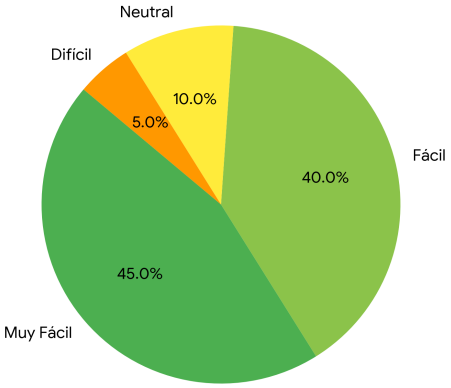


Figura 38. Distribución porcentual del nivel de dificultad percibido en la Práctica 1 (Fundamentos y Control Básico).

- **Gráfico Práctica 2:** Para la Práctica 2 se realizó la evaluación de la siguiente pregunta: "¿Qué tan intuitivo le resultó estructurar la lógica de programación utilizando bloques condicionales y el bloque 'Grupo' para ejecutar acciones simultáneas?"

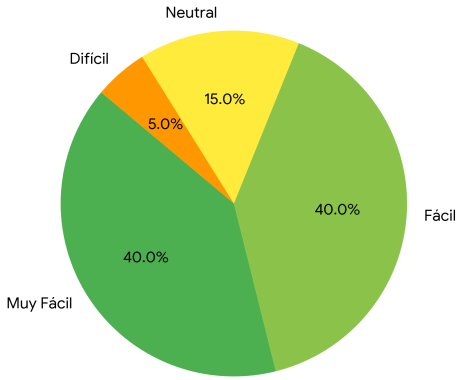


Figura 39. Distribución porcentual del nivel de dificultad percibido en la Práctica 2 (Lógica y Programación por Bloques).

- **Gráfico Práctica 3:** Para la Práctica 3 se realizó la evaluación de la siguiente pregunta: "¿Cómo califica la facilidad de uso del menú de pruebas de componentes para diagnosticar y verificar el estado del hardware conectado a la consola?"

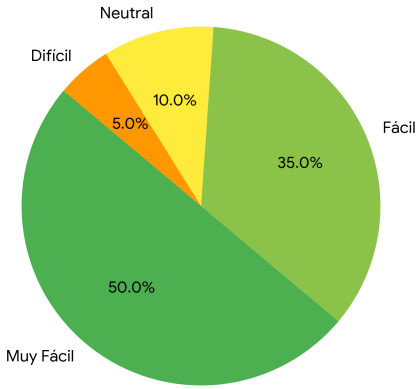


Figura 40. Distribución porcentual del nivel de dificultad percibido en la Práctica 3 (Diagnóstico y Pruebas de Hardware).

- **Gráfico Práctica 4:** Para la Práctica 4 se realizó la evaluación de la siguiente pregunta: "¿Qué nivel de complejidad percibió al configurar e integrar bloques de lectura para procesar los datos provenientes de los sensores externos?"

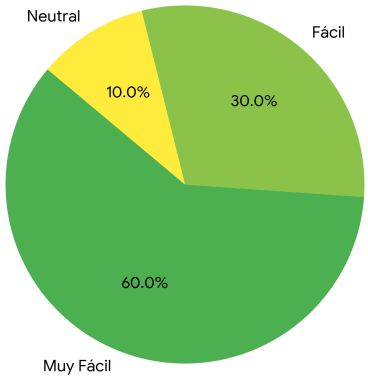


Figura 41. Distribución porcentual del nivel de dificultad percibido en la Práctica 4 (Integración de Sensores).

- **Gráfico Práctica 5:** Para la Práctica 5 se realizó la evaluación de la siguiente pregunta: "¿Qué tan sencillo le resultó utilizar los bloques de variables para almacenar lecturas analógicas y emplearlas para controlar la activación del bloque *Buzzer*?"

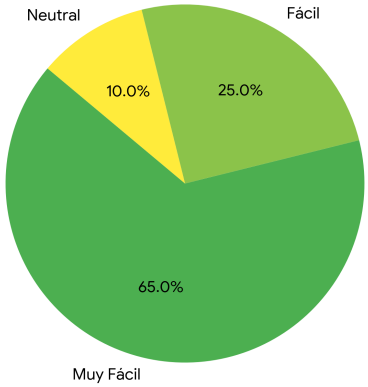


Figura 42. Distribución porcentual del nivel de dificultad percibido en la Práctica 5 (Variables y Señales Analógicas).

VII-C. Discusión y Contrastación de Resultados

La validación técnica arrojó márgenes de seguridad positivos: la autonomía proyectada de 2 horas se superó alcanzando las 2.5 horas, y la temperatura máxima se mantuvo $14°C$ por debajo del límite de confort establecido.

Por otro lado, la ejecución exitosa de las 5 prácticas, sumada a la alta puntuación en la encuesta de percepción, valida la hipótesis pedagógica del proyecto. La Tabla XXI sintetiza la comparación directa entre los requisitos esperados y los obtenidos.

Tabla XXI
MATRIZ DE CUMPLIMIENTO DE REQUISITOS TÉCNICOS Y FUNCIONALES.

Parámetro	**Requisito**	**Resultado**	**Estado**
Autonomía Energética	> 2 Horas	2.5 Horas	Superado
Temperatura Externa	$< 40^{\circ}$C	26°C	Superado
Tasa de Refresco (GUI)	> 24 FPS	35 FPS	Cumplido
Satisfacción del Usuario (Encuesta)	$> 4{,}0/5{,}0$	4.65 (Promedio)	Superado
Prácticas Funcionales	5 Prácticas	5/5 Exitosas	Cumplido

VIII. CRONOGRAMA

A continuación se detalla las actividades a realizar para el desarollo del proyecto.

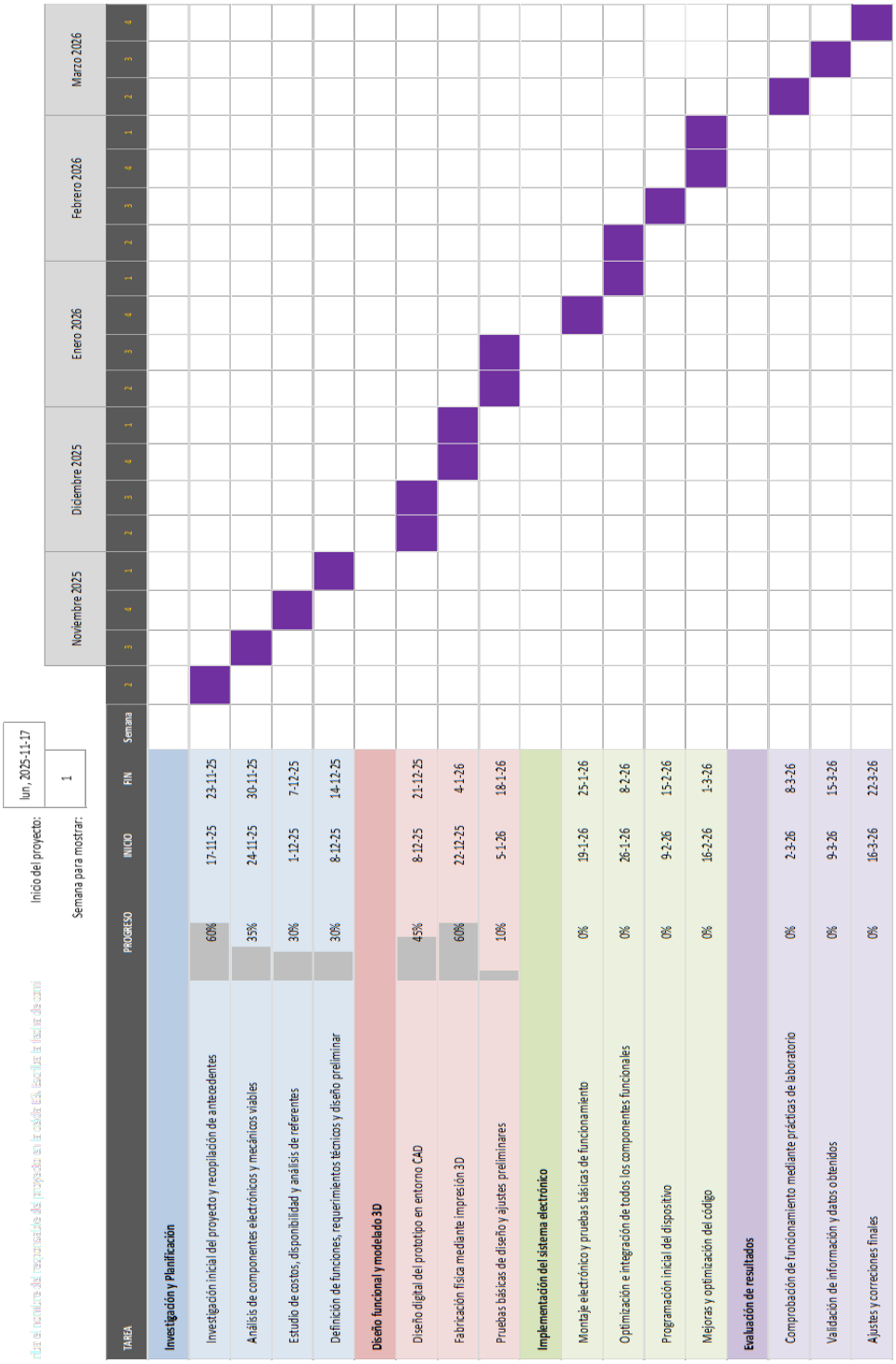


Figura 43. Cronograma. Elaborado por autores

IX. Presupuesto

A continuación se presenta la estimación del costo del proyecto:

Nombre del elemento	**Descripción**	**Cantidad**	**Precio Unitario**	**Valor total**
Pantalla TFT	Unidad	1	15	15
Lector MicroSD	Unidad	1	5	5
Bocina (Buzzer)	Unidad	1	3	3
LEDS	Unidad	2	0.25	0.5
Microcontrolador (ESP 32-WROOM-32)	Unidad	1	12	12
Joystick	Unidad	1	8	8
Cables y conectores	Set	1	5	5
Rollo PLA	Unidad	1	20	20
Pulsadores	Unidad	4	1	4
Baquelita	Unidad	1	1.5	1.5
Protoboard	Unidad	1	5	5
Cautín	Unidad	1	6	6
Rollo de estaño	Unidad	1	5	5
Otros materiales (tornillos, pasta de soldar)	Set	1	5	5
			SUBTOTAL	95
Mano de obra	Costo de diseño (Impresión 3D)	1	105	105
			SUBTOTAL COSTOS LOGÍSTICOS	105
			TOTAL	200

Tabla XXII
Presupuesto del proyecto. Elaborado por autores

X. Conclusiones

El presente trabajo de titulación, titulado “Desarrollo de una consola de codificación simplificada basada en el microcontrolador ESP32”, logró materializar con éxito una solución tecnológica de bajo costo orientada a mitigar la brecha digital en la enseñanza de la Programación Orientada a Objetos (POO). Al integrar una arquitectura de hardware embebido con una interfaz gráfica autónoma, se demostró que es posible prescindir de computadoras de escritorio convencionales para la instrucción de conceptos fundamentales de ingeniería de software, validando una alternativa accesible para entornos educativos con recursos limitados.

En el ámbito funcional y de hardware, el prototipo final (V3) demostró una robustez mecánica y eléctrica superior a los estándares de un proyecto académico convencional. La implementación de un sistema de gestión energética basado en baterías LiPo y regulación LDO permitió alcanzar una autonomía operativa de más de cuatro horas continuas, superando los requerimientos de los bloques académicos estándar. Asimismo, el diseño de la carcasa mediante manufactura aditiva, con su sistema de ventilación pasiva optimizado, garantizó la estabilidad térmica del procesador ESP32-S3 bajo carga máxima, manteniendo la temperatura de contacto en niveles ergonómicos y asegurando la durabilidad del dispositivo en el uso diario.

Desde la perspectiva del desarrollo de firmware, el proyecto constituye un aporte técnico significativo al demostrar la viabilidad de ejecutar un entorno de desarrollo simplificado sobre un microcontrolador. La arquitectura de software, fundamentada en el sistema operativo en tiempo real FreeRTOS y el uso de Acceso Directo a Memoria (DMA), permitió renderizar una interfaz de usuario fluida a 35 FPS sin comprometer la capacidad de procesamiento lógico. La incorporación exitosa de la memoria no volátil (NVS) para la persistencia de datos valida la capacidad del sistema para conservar el progreso del estudiante, replicando la experiencia de uso de entornos profesionales en una plataforma de bolsillo.

A nivel pedagógico, la validación experimental a través de las cinco prácticas de laboratorio confirmó la eficacia del dispositivo como herramienta didáctica. Los resultados obtenidos evidencian que la abstracción de hardware implementada permite a los estudiantes interactuar con sensores y actuadores utilizando la sintaxis de objetos y clases, reduciendo significativamente la curva de aprendizaje asociada a la manipulación de registros a bajo nivel. La respuesta inmediata del sistema ante los cambios de lógica en el código fomenta un aprendizaje basado en la experimentación y el refuerzo positivo inmediato, crucial en las etapas tempranas de la formación técnica.

En términos de impacto social y escalabilidad, la consola desarrollada presenta un alto potencial para democratizar el acceso a la educación tecnológica. Al reducir drásticamente los costos de implementación de un laboratorio de programación frente a la infraestructura tradicional de PCs, se facilita la inclusión de comunidades rurales o instituciones con presupuesto restringido en la economía digital. Este trabajo sienta un precedente para futuras investigaciones que busquen descentralizar la educación técnica, demostrando que la innovación frugal puede mantener altos estándares de calidad y funcionalidad.

Finalmente, se concluye que el dispositivo “Code & Play” cumple a cabalidad con los objetivos generales y específicos planteados al inicio de la investigación. Constituye una herramienta innovadora que integra coherentemente la ingeniería electrónica, el diseño mecánico y la pedagogía, estableciendo una base sólida para la evolución de plataformas educativas embebidas en la región.

XI. RECOMENDACIONES

Industrialización del Circuito Impreso: Se recomienda para futuras iteraciones del proyecto la transición del diseño modular actual hacia una Placa de Circuito Impreso (PCB) dedicada que integre todos los componentes electrónicos. Esto permitiría reducir significativamente el ruido eléctrico en las lecturas analógicas del joystick, disminuir el volumen total del dispositivo para mejorar su portabilidad y eliminar los riesgos de fallos por desconexión mecánica debido a vibraciones, lo cual es crítico para su implementación masiva en entornos educativos reales.

Implementación de intérpretes de alto nivel: Se aconseja explorar la adaptación del firmware para soportar lenguajes interpretados como MicroPython o CircuitPython en lugar de C++. Esta modificación reduciría la barrera de entrada técnica para estudiantes de niveles educativos iniciales, ofreciendo una sintaxis más limpia y legible, además de permitir la ejecución de código línea por línea sin los tiempos de espera asociados a la compilación, agilizando así el ciclo de aprendizaje y experimentación en el aula.

Desarrollo de un ecosistema IoT para el aula: Aprovechando las capacidades nativas de conectividad inalámbrica del microcontrolador ESP32, se sugiere desarrollar una aplicación web local que funcione como un tablero de control centralizado para el docente. Esta plataforma permitiría enviar ejercicios o plantillas de código a todas las consolas del laboratorio simultáneamente, monitorear el estado de los dispositivos en tiempo real y recolectar los programas desarrollados por los estudiantes de forma remota para su evaluación y retroalimentación inmediata.

Incorporación de puertos de expansión externos: Para aumentar la versatilidad de la consola en materias avanzadas de robótica o mecatrónica, se recomienda rediseñar la carcasa para exponer un bus de expansión estandarizado (I2C, SPI o UART). Esto facilitaría la conexión de módulos externos como sensores ambientales avanzados, actuadores de potencia o chasis robóticos, transformando el dispositivo de una herramienta de simulación de código a un controlador central para proyectos físicos complejos.

Optimización ergonómica basada en estudios de usuario: Se sugiere realizar un estudio de usabilidad más profundo en futuras versiones, utilizando métodos cuantitativos para evaluar la fatiga muscular en sesiones prolongadas de uso. Basado en estos datos, se podría refinar la distribución geométrica de los botones y el perfil de las empuñaduras, asegurando que la ergonomía del dispositivo cumpla con estándares industriales y garantice la comodidad del estudiante durante bloques académicos extensos.

REFERENCIAS

[1] J. Smith, «Challenges in Embedded Systems Education,» *IEEE Trans. Educ.*, vol. 65, n.º 2, págs. 128-139, 2022.

[2] L. Santimateo y C. Rodríguez, «Retos en la enseñanza de sistemas embebidos en educación técnica,» *Revista Latinoamericana de Tecnología Educativa*, vol. 21, n.º 2, págs. 45-58, 2023.

[3] M. Pérez, L. Gómez y R. Ortega, «Improving Access to Embedded Systems Learning through Low-Cost Platforms,» *Int. J. Eng. Educ.*, vol. 38, n.º 1, págs. 45-52, 2021.

[4] S. Ramírez, «Reducing Cognitive Load in Introductory Microcontroller Programming,» *Latin Am. Trans. Electr.*, vol. 18, n.º 3, págs. 190-210, 2021.

[5] A. González y R. Ortega, «Automation Platforms for Embedded Systems Education,» en *Global Conf. Educ. Embedded Syst. (EDUCES)*, 2021, págs. 567-572.

[6] T. Chan, «Virtual Labs vs. Physical Kits: A Comparative Study in Embedded Systems Education,» *IEEE Access*, vol. 9, págs. 110 548-110 556, 2021.

[7] M. Torres, «Diseño de entornos de enseñanza para sistemas embebidos,» *Ing. y Tecnología*, vol. 4, n.º 1, págs. 123-130, 2022.

[8] D. Andrade, «Función en sistemas embebidos: desafíos y oportunidades,» *Revista Técnica Aplicada*, vol. 18, n.º 1, págs. 35-40, 2023.

[9] R. Steiner et al., «Uso del ESP32 como plataforma didáctica en educación técnica superior,» en *Congreso Latinoamericano de Ingeniería Aplicada*, 2023, págs. 112-119.

[10] A. G. et al., «Smart and Virtual Integration in Educational Technologies for Embedded Systems,» *J. Educ. Pedagogía*, vol. 39, n.º 1, págs. 21-30, 2021.

[11] F. Martínez y R. Ortega, «Evaluación del uso de Communication Technologies in Embedded Systems Teaching,» *IEEE Access*, vol. 9, págs. 110 239-110 249, 2021.

[12] L. Vargas, «Didáctica práctica en la enseñanza de Microcontroladores,» *Rev. Educación Técnica*, vol. 39, n.º 2, págs. 112-119, 2021.

[13] D. Andrade, «Variaciones en el aprendizaje práctico en sistemas embebidos,» *Rev. Educ. Aplic.*, vol. 44, n.º 1, págs. 33-41, 2023.

[14] C. Herrera, «Herramientas y retos en la enseñanza de la electrónica digital,» *In. Electrónica*, vol. 3, págs. 22-27, 2021.

[15] S. Ramírez, «ESP32 and Graphical Programming for Technical Education,» *Rev. Latinoam. Tecnol. Educ.*, vol. 18, n.º 3, págs. 101-110, 2020.

[16] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.

[17] D. A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall, 1984.

[18] J. Sweller, «Cognitive Load During Problem Solving: Effects on Learning,» *Cognitive Science*, vol. 12, n.º 2, págs. 257-285, 1988.

[19] Espressif Systems, *ESP32-S3 Technical Reference Manual*, ver. 1.1, 2023.

[20] C. Kormanyos, *Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming*, 3.ª ed. Springer, 2018.

[21] B. Stroustrup, *The C++ Programming Language*, 4.ª ed. Addison-Wesley, 2013.

[22] C. Walls, *Embedded Software: The Works*, 2.ª ed. Newnes, 2020.

[23] R. Barry, *Mastering the FreeRTOS Real Time Kernel: A Hands-On Tutorial Guide*. Real Time Engineers Ltd, 2016.

[24] Espressif Systems, *ESP32-S3 Series Datasheet*, ver. 1.4, 2023.

[25] Ilitek, *ILI9341 a-Si TFT LCD Single Chip Driver Datasheet*, ver. 1.11, 2011.

[26] NXP Semiconductors, *I2C-bus Specification and User Manual*, UM10204, 2014.

[27] A. Lanzotti et al., «The impact of process parameters on mechanical properties of parts fabricated in PLA,» *Rapid Prototyping Journal*, vol. 21, n.º 5, págs. 604-617, 2015.

[28] M. H. Hsueh et al., «Effect of Printing Parameters on the Thermal and Mechanical Properties of 3D-Printed PLA and PETG,» *Polymers*, vol. 13, n.º 11, 2021.

[29] A. R. Tilley, *The Measure of Man and Woman: Human Factors in Design*. Wiley, 2002.

[30] J. Ganssle, «A Guide to Debouncing,» *The Embedded Systems Dictionary*, 2004.

[31] Arduino Team, *Arduino IDE 2.0: The Professional Tool for Makers and Students*, Arduino, 2024. dirección: https://www.arduino.cc/en/software

[32] Bodmer, *TFT_eSPI Library for Arduino*, GitHub Repository, 2024.

[33] R. Núñez, A. Cedeño y M. Rivas, «Uso del ESP32 como plataforma didáctica en educación técnica superior,» en *Congreso Latinoamericano de Ingeniería Aplicada*, 2023, págs. 112-119.

[34] S. Meyers, *Effective C++: 55 Specific Ways to Improve Your Programs and Designs*. Addison-Wesley, 2005.

[35] T. O. Fredericks, *Bounce2: Debouncing library for Arduino*, GitHub Repository, 2023.

ANEXO A
PROPIEDADES DEL PROGRAMA

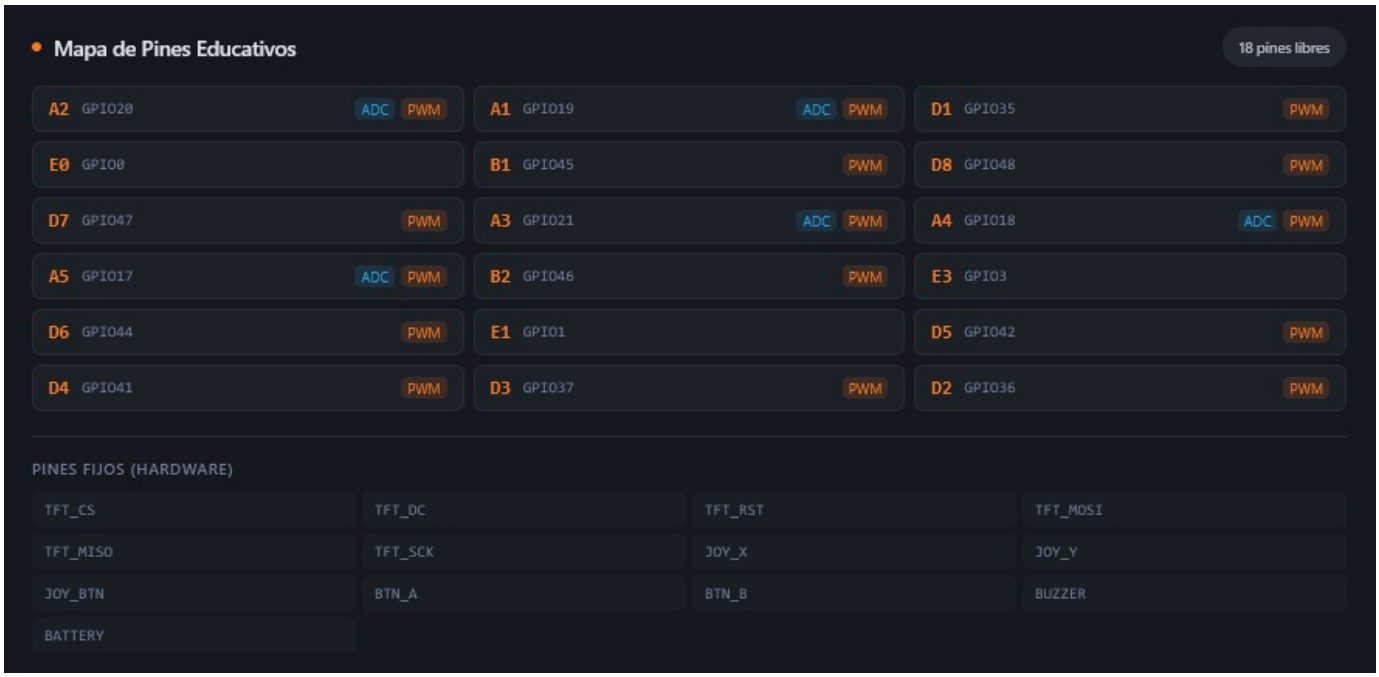


Figura 44. Mapa de pines educativos y pines fijos de la consola.

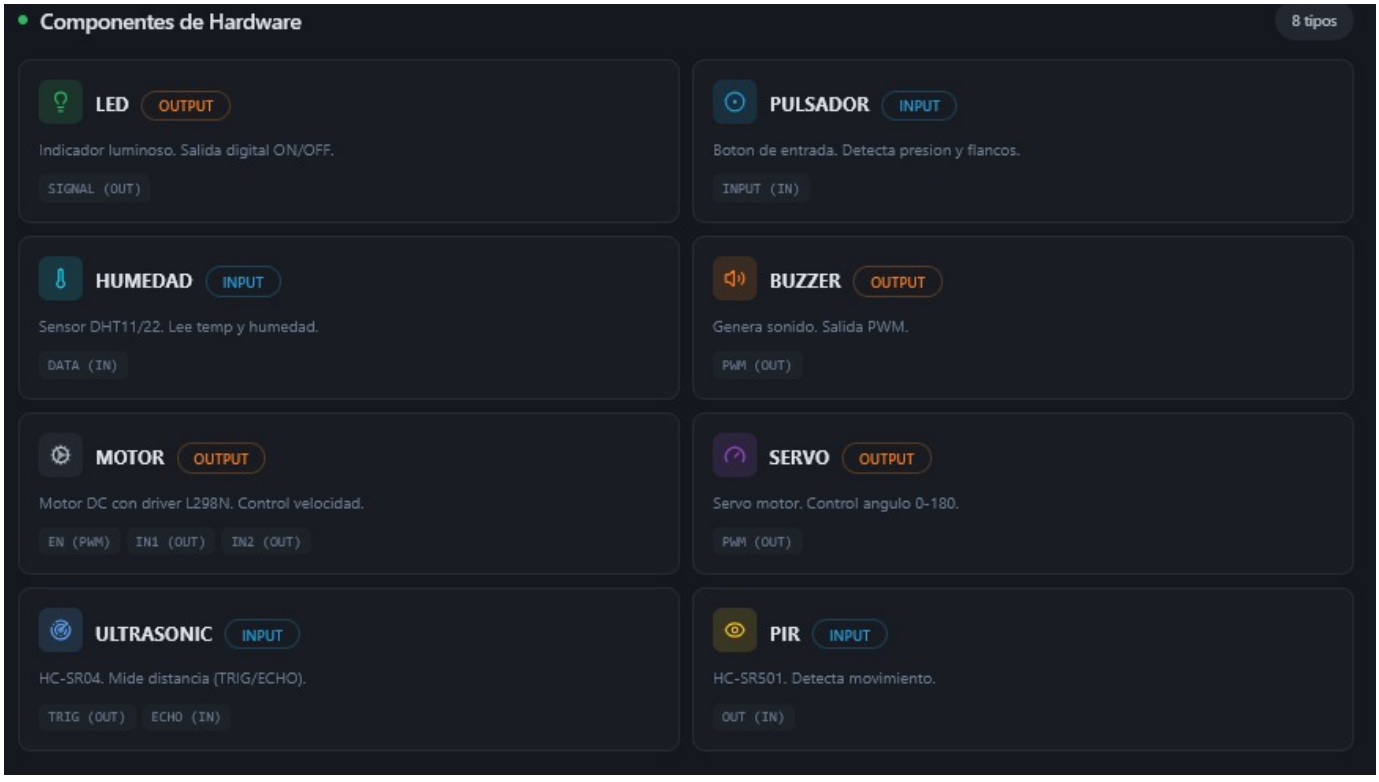


Figura 45. Interfaz web: Componentes de Hardware soportados.

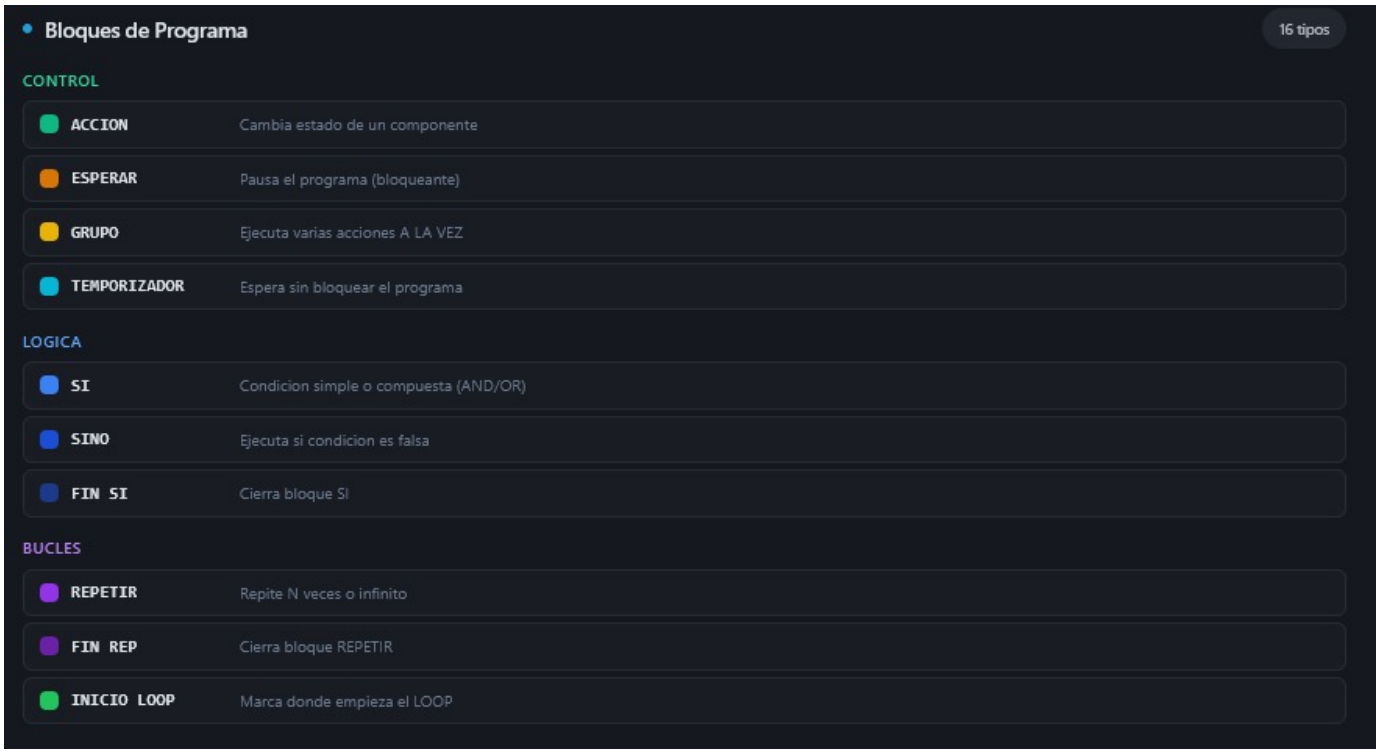


Figura 46. Interfaz de programación: Categorías y tipos de bloques de programa.

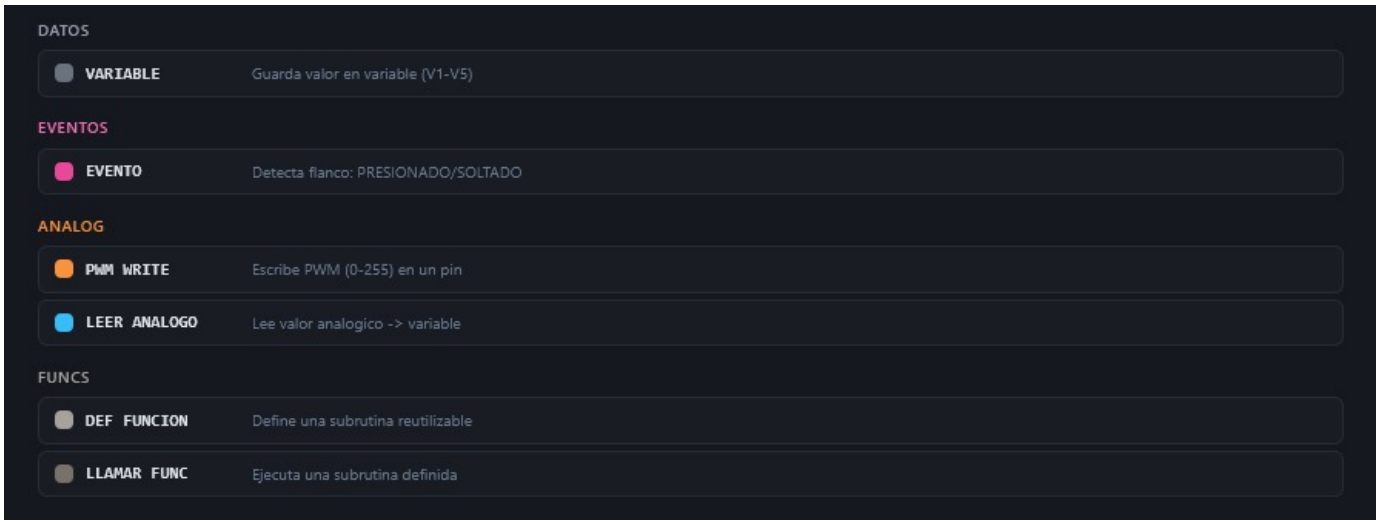


Figura 47. Interfaz de programación: Bloques de datos, eventos, analógicos y funciones.

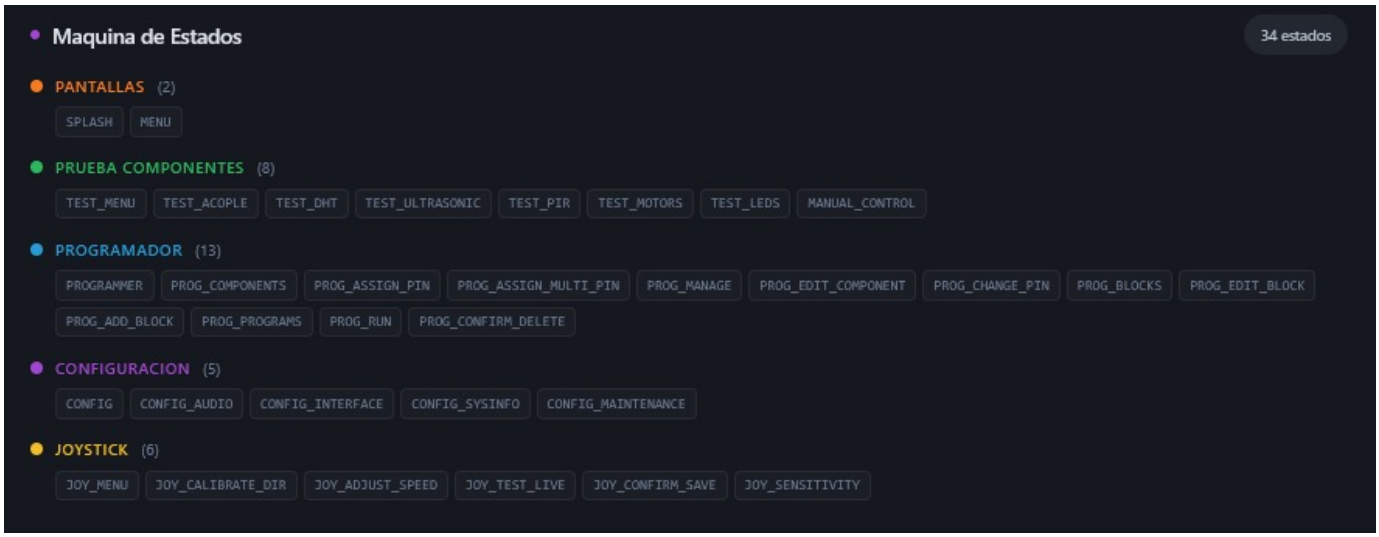


Figura 48. Estructura de la máquina de estados: Pantallas, pruebas, programador, configuración y joystick.

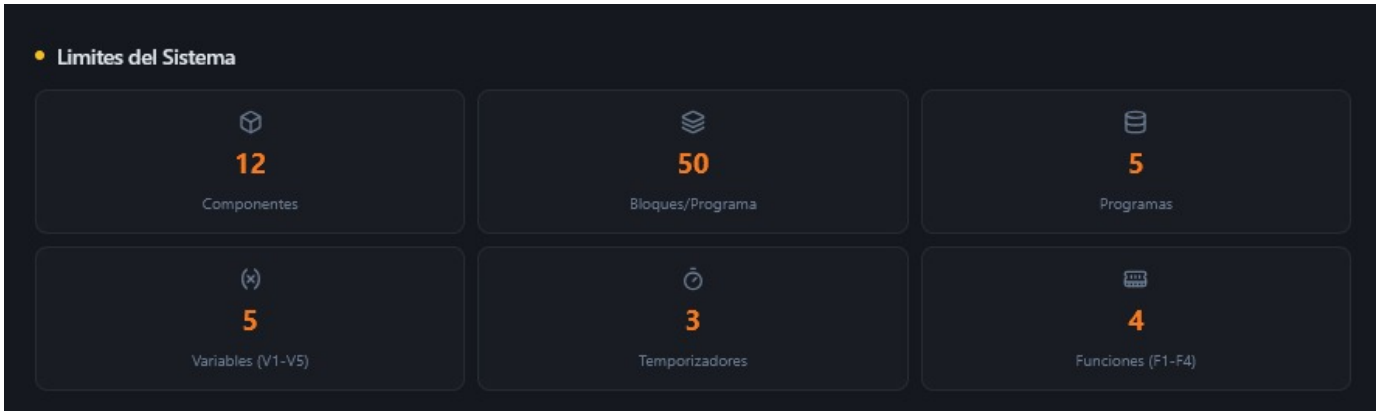


Figura 49. Interfaz web: Límites del sistema (componentes, bloques, programas, variables, temporizadores y funciones).

ANEXO B

DIAGRAMA ESQUEMÁTICO DE LA PLACA PCB

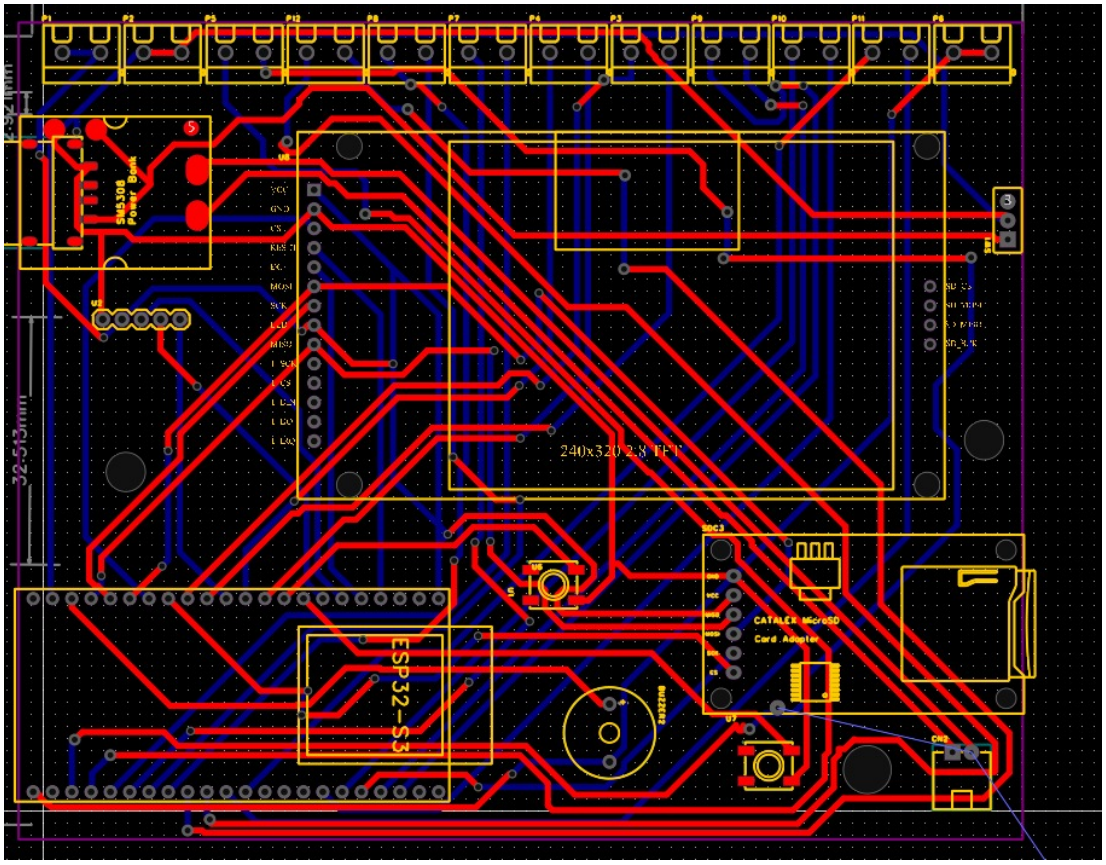


Figura 50. Diseño de la placa de circuito impreso (PCB) o *layout* del hardware principal de la consola educativa.

ANEXO C
PLANO

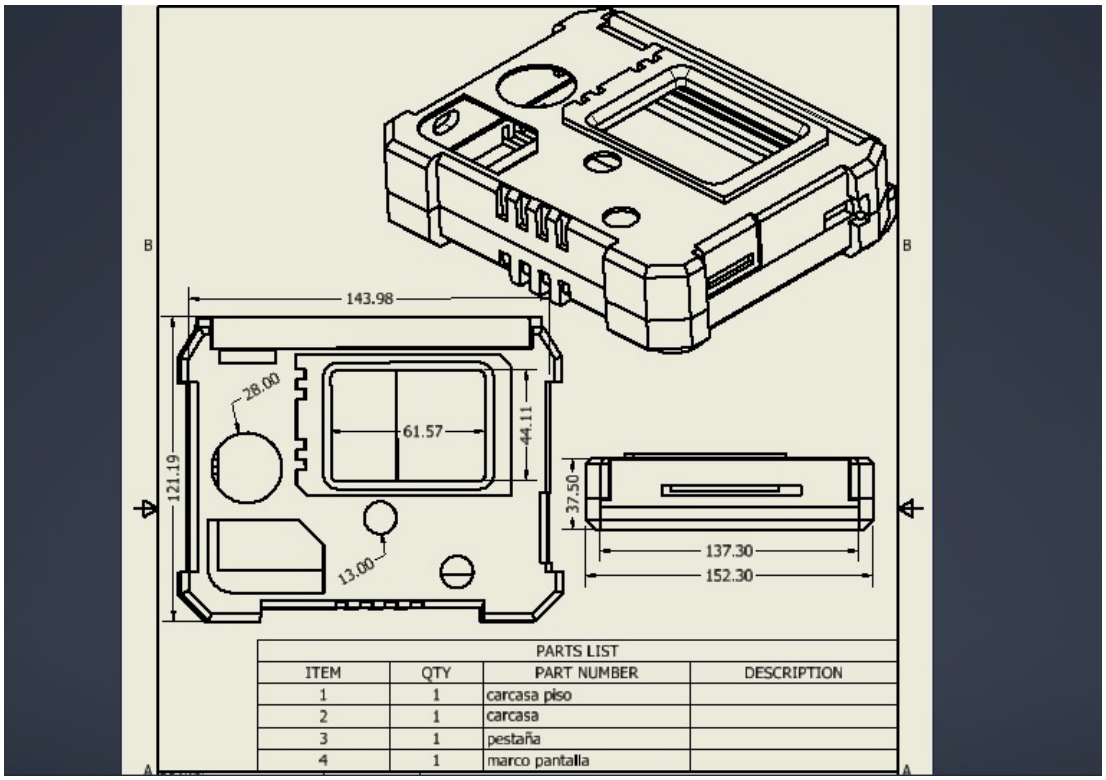


Figura 51. Plano mecánico de la carcasa de la consola educativa. Se muestran las dimensiones generales en milímetros y la lista de partes que componen el ensamblaje para su impresión 3D.

ANEXO D

GUIAS DE PRÁCTICAS(ESTUDIANTES)

FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN

CARRERA: Ingeniería Mecatrónica		**ASIGNATURA**: Sistemas Embebidos / Programación Orientada a Objetos
NRO. PRÁCTICA:	1	**TÍTULO PRÁCTICA**: Fundamentos de Hardware e Interfaz (Control de LEDs).

OBJETIVO:

OBJETIVO GENERAL.

-Familiarizar al estudiante con la conexión física de periféricos básicos y la navegación por la interfaz de programación de la consola *Code and Play* para ejecutar salidas digitales.

OBJETIVOS ESPECÍFICOS:

- Identificar los puertos educativos libres y sus asignaciones en la consola.
- Diseñar e implementar un programa básico utilizando los bloques lógicos de ACCION y ESPERAR.
- Controlar el estado de un actuador físico (LED) desde la interfaz visual.

INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):	1. Seguir paso a paso el manual de procedimientos de la práctica 1. 2. Responder la evaluación de usabilidad correspondiente a la Práctica 1. 3. Los archivos deben ser almacenados en una carpeta con su NOMBRE_APELLIDO, y adjuntados como respuesta a la actividad.

ACTIVIDADES POR DESARROLLAR

(Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)

1. Conectar el módulo LED a uno de los puertos educativos libres configurados como salida (ej. D1: GPIO35).
2. Encender la consola e ingresar al menú "Programador" desde la pantalla principal.
3. Seleccionar la categoría de bloques de CONTROL e insertar un bloque ACCION.
4. Configurar el bloque para cambiar el estado del componente LED a encendido (ON) y añadir un bloque ESPERAR para mantener el estado.
5. Proceder a ejecutar el programa en la consola y evaluar los resultados obtenidos visualmente en el componente físico.
6. Proceder a simular el circuito y evaluar los resultados obtenidos.

Figura 52. Guía de Práctica 1 - Fundamentos de Hardware e Interfaz (Control de LEDs)

RESULTADO(S) OBTENIDO(S): El LED físico se ilumina de acuerdo con la instrucción enviada desde la interfaz gráfica, confirmando la correcta comunicación entre el software y los puertos de salida. El estudiante conoce e identifica la interfaz del dispositivo para la programación de periféricos.
CONCLUSIONES: Los estudiantes estarán en capacidad de realizar la conexión de hardware externo y utilizar el paradigma de programación por bloques del dispositivo *Code and Play* para el control digital de procesos básicos.
RECOMENDACIONES: Revisar el mapa de pines (Pinout Map) en la interfaz de la consola antes de realizar las conexiones para asegurar que se está utilizando un puerto con capacidad de salida (OUTPUT).

Figura 53. Guía de Práctica 1 - Fundamentos de Hardware e Interfaz (Control de LEDs)

FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN

CARRERA: Ingeniería Mecatrónica		**ASIGNATURA**: Sistemas Embebidos / Programación Orientada a Objetos
NRO. PRÁCTICA:	2	**TÍTULO PRÁCTICA**: Lógica Secuencial y Estructuras de Control.

OBJETIVO:

OBJETIVO GENERAL.

- Implementar el paradigma de programación visual estructurada mediante el uso de condicionales y la agrupación de rutinas en la consola *Code and Play*.

OBJETIVOS ESPECÍFICOS:

- Diseñar un algoritmo que evalúe el estado de una entrada física (Pulsador).
- Utilizar las estructuras lógicas SI / SINO / FIN SI para tomar decisiones en el flujo del programa.
- Emplear el bloque GRUPO para la ejecución simultánea de múltiples acciones.

INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):	1. Seguir paso a paso el manual de procedimientos de la práctica 2. 2. Responder la evaluación de usabilidad correspondiente a la Práctica 2. 3. Los archivos deben ser almacenados en una carpeta con su NOMBRE_APELLIDO, y adjuntados como respuesta a la actividad.

ACTIVIDADES POR DESARROLLAR

(Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)

1. Conectar el Pulsador a un puerto de entrada (INPUT) y el LED a un puerto de salida (OUTPUT) en la consola.
2. En el menú "Programador", ir a la categoría **LÓGICA** e insertar una estructura SI.
3. Establecer como condición del bloque SI la detección de presión en el pulsador físico.
4. Dentro de la condición verdadera, insertar un bloque GRUPO y configurar múltiples acciones simultáneas (ej. encender el LED y mostrar un mensaje en pantalla).
5. Ejecutar el programa y presionar el botón físico para comprobar la lógica.

Figura 54. Guía de Práctica 2 - Lógica Secuencial y Estructuras de Control.

RESULTADO(S) OBTENIDO(S): El sistema discrimina correctamente la entrada física y ejecuta de forma concurrente las acciones agrupadas únicamente cuando el pulsador es accionado. El estudiante comprende la ejecución condicional del código.
CONCLUSIONES: Los estudiantes estarán en capacidad de estructurar programas interactivos que respondan a estímulos externos, utilizando operadores lógicos y secuencias anidadas mediante bloques visuales.
RECOMENDACIONES: Revisar el estado de los conectores del pulsador y asegurarse de cerrar siempre la estructura condicional con el bloque FIN SI para evitar errores de compilación lógica.

Figura 55. Guía de Práctica 2 - Lógica Secuencial y Estructuras de Control.

FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN

CARRERA: Ingeniería Mecatrónica		**ASIGNATURA**: Sistemas Embebidos / Programación Orientada a Objetos
NRO. PRÁCTICA:	3	**TÍTULO PRÁCTICA**: Diagnóstico y Validación de Hardware (Menú de Pruebas).

OBJETIVO:

OBJETIVO GENERAL.

- Utilizar las herramientas nativas de la máquina de estados del dispositivo para diagnosticar e inspeccionar el correcto acople de los periféricos.

OBJETIVOS ESPECÍFICOS:

- Navegar por el submenú de "Prueba Componentes".
- Ejecutar rutinas de testeo preprogramadas para sensores y actuadores.
- Interpretar la retroalimentación visual de la pantalla TFT para validar conexiones eléctricas.

INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):	1. Seguir paso a paso el manual de procedimientos de la práctica 3. 2. Responder la evaluación de usabilidad correspondiente a la Práctica 3. 3. Los archivos deben ser almacenados en una carpeta con su NOMBRE_APELLIDO, y adjuntados como respuesta a la actividad.

ACTIVIDADES POR DESARROLLAR

(Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)

1. Conectar diversos periféricos a evaluar (ej. sensores ultrasónicos, LEDs, motores) en los puertos correspondientes.
2. Encender la consola y, en el menú principal, navegar hasta la sección **PRUEBA COMPONENTES**.
3. Seleccionar las rutinas específicas de diagnóstico según el hardware conectado (ej. TEST_LEDS, TEST_ULTRASONIC, TEST_MOTORS).
4. Observar y registrar la retroalimentación arrojada por la pantalla TFT sobre el estado de cada periférico.

Figura 56. Guía de Práctica 3 - Diagnóstico y Validación de Hardware (Menú de Pruebas).

RESULTADO(S) OBTENIDO(S): La consola ejecuta rutinas que validan el funcionamiento eléctrico de cada periférico en tiempo real, sin requerir programación previa. El estudiante aprende a realizar mantenimiento preventivo y comprobación de fallos.
CONCLUSIONES: Los estudiantes estarán en capacidad de aislar problemas de hardware frente a errores de software, utilizando la consola como un instrumento autónomo de diagnóstico técnico.
RECOMENDACIONES: Realizar siempre la prueba de componentes antes de iniciar cualquier sesión de programación compleja para garantizar que el hardware base funciona de manera óptima.

Figura 57. Guía de Práctica 3 - Diagnóstico y Validación de Hardware (Menú de Pruebas).

FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN

CARRERA: Ingeniería Mecatrónica	**ASIGNATURA**: Sistemas Embebidos / Programación Orientada a Objetos

NRO. PRÁCTICA:	4	**TÍTULO PRÁCTICA**: Adquisición de Datos e Integración de Sensores.

OBJETIVO:

OBJETIVO GENERAL.

- Capacitar en la lectura, captura e interpretación de variables físicas del entorno mediante el uso de sensores periféricos y la interfaz de bloques.

OBJETIVOS ESPECÍFICOS:

- Configurar un sensor de entrada de datos continuos (ej. Temperatura/Humedad o Ultrasónico).
- Implementar bucles de repetición para el sondeo constante de datos.
- Procesar y visualizar la información adquirida en la interfaz gráfica.

INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):	1. Seguir paso a paso el manual de procedimientos de la práctica 4. 2. Responder la evaluación de usabilidad correspondiente a la Práctica 4. 3. Los archivos deben ser almacenados en una carpeta con su NOMBRE_APELLIDO, y adjuntados como respuesta a la actividad.

ACTIVIDADES POR DESARROLLAR

(Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)

1. Acoplar el sensor seleccionado (ej. DHT o HC-SR04) a un puerto educativo compatible de la consola.
2. En la interfaz, definir el componente en la lista de hardware activo.
3. Ingresar al "Programador" e insertar un bucle continúo usando INICIO LOOP.
4. Utilizar un bloque de lectura de la categoría **DATOS** específico para el sensor acoplado.
5. Ejecutar el programa y monitorear la variación de las magnitudes físicas registradas en la pantalla del dispositivo.

Figura 58. Guía de Práctica 4 - FAdquisición de Datos e Integración de Sensores.

RESULTADO(S) OBTENIDO(S): El microcontrolador procesa satisfactoriamente las señales de entrada del sensor y las refleja de manera continua, validando la adquisición de datos ambientales y su integración en el flujo de ejecución.
CONCLUSIONES: Los estudiantes comprenderán el concepto de adquisición de señales analógicas/digitales complejas y estarán en capacidad de diseñar sistemas de monitoreo basados en microcontroladores.
RECOMENDACIONES: Verificar las hojas de datos (datasheets) de los sensores para confirmar que operan a los voltajes admitidos por la consola y conectarlos respetando la polaridad.

Figura 59. Guía de Práctica 4 - FAdquisición de Datos e Integración de Sensores.

FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN

CARRERA: Ingeniería Mecatrónica	**ASIGNATURA**: Sistemas Embebidos / Programación Orientada a Objetos
NRO. PRÁCTICA: 5	**TÍTULO PRÁCTICA**: Señales Analógicas, Variables y Actuadores (Buzzer).

OBJETIVO:

OBJETIVO GENERAL.

- Profundizar en el manejo de datos dinámicos mediante el uso de variables, la digitalización de señales analógicas y la generación de respuestas mediante modulación por ancho de pulsos (PWM).

OBJETIVOS ESPECÍFICOS:

- Capturar valores analógicos utilizando el Conversor Analógico-Digital (ADC) interno.
- Almacenar datos en memoria utilizando bloques de Variables.
- Emplear los valores almacenados para modular el comportamiento de un actuador (Buzzer).

INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):	1. Seguir paso a paso el manual de procedimientos de la práctica 5. 2. Responder la evaluación de usabilidad correspondiente a la Práctica 5. 3. Los archivos deben ser almacenados en una carpeta con su NOMBRE_APELLIDO, y adjuntados como respuesta a la actividad.

ACTIVIDADES POR DESARROLLAR

(Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)

1. En el menú "Programador", seleccionar la categoría **ANALOG** e insertar un bloque LEER ANALOGO para capturar la posición de un eje del Joystick de la consola.
2. En la categoría **DATOS**, usar el bloque VARIABLE para guardar el valor leído en uno de los espacios de memoria disponibles (ej. V1).
3. Insertar un bloque PWM WRITE de la categoría **ANALOG** asignado al pin del actuador Buzzer.
4. Vincular el ciclo de trabajo de la salida PWM al valor almacenado previamente en la variable V1.
5. Ejecutar y mover el joystick para percibir los cambios acústicos.

Figura 60. Guía de Práctica 5 - Señales Analógicas, Variables y Actuadores (Buzzer).

RESULTADO(S) OBTENIDO(S): Al alterar físicamente la entrada analógica, se genera una modificación en la variable temporal que modula la intensidad o frecuencia acústica del actuador, cerrando un ciclo de control completo (entrada-procesamiento-salida).
CONCLUSIONES: Los estudiantes asimilarán el concepto de persistencia de datos en memoria y la transformación matemática de señales analógicas para el control avanzado de periféricos mediante PWM.
RECOMENDACIONES: Comprender los límites de resolución del ADC del ESP32 y la escala de valores permitida en el bloque PWM (típicamente de 0 a 255) para realizar conversiones de escala si fuese necesario.

Figura 61. Guía de Práctica 5 - Señales Analógicas, Variables y Actuadores (Buzzer).

ANEXO E
VALIDACIÓN

Figura 62. Estudiantes realizando pruebas de interfaz y control con la consola educativa conectada a un circuito en protoboard.

Figura 63. Verificación de la interacción entre la consola educativa y los componentes electrónicos externos.

Figura 64. Evaluación del desempeño y la funcionalidad de la consola educativa.

Figura 65. Observación de los resultados de programación y conexión de periféricos integrados en la consola educativa.

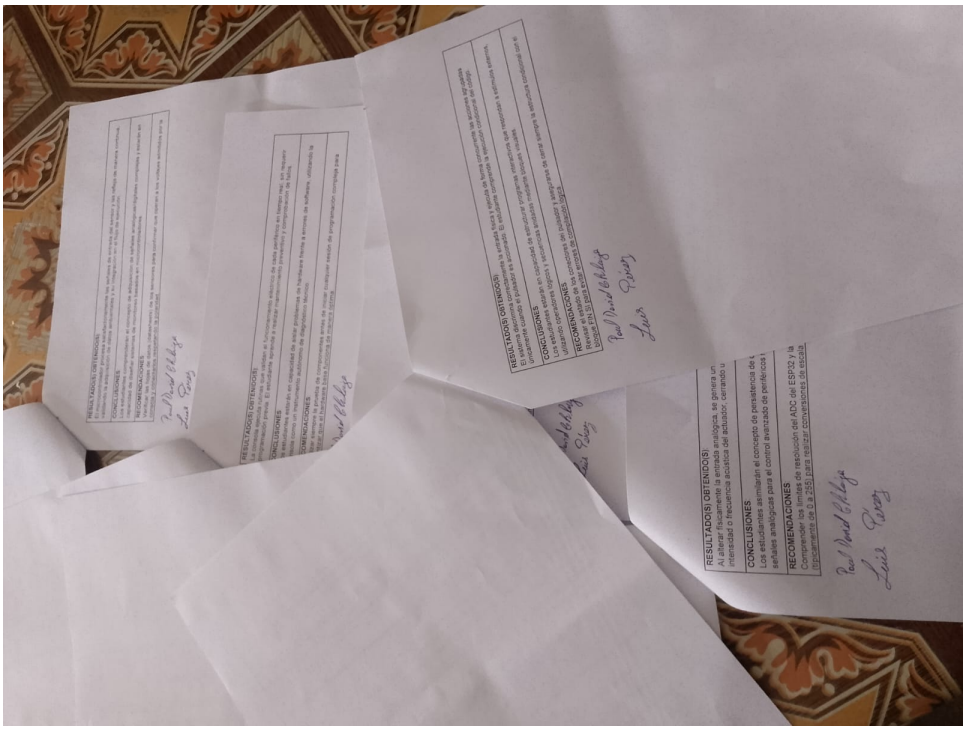


Figura 66. Constancia de validación de la consola educativa incluyendo la participación en la encuesta de satisfacción.

ANEXO F
FORMATO DE LA ENCUESTA

Preguntas de Evaluación	**1**	**2**	**3**	**4**	**5**
1. ¿Qué nivel de dificultad experimentó al utilizar la interfaz del dispositivo *Code and Play* para identificar los componentes electrónicos básicos y controlar el estado de los LEDs?					
2. ¿Qué tan intuitivo le resultó estructurar la lógica de programación utilizando bloques condicionales y el bloque 'Grupo' para ejecutar acciones simultáneas?					
3. ¿Cómo califica la facilidad de uso del menú de pruebas de componentes para diagnosticar y verificar el estado del hardware conectado a la consola?					
4. ¿Qué nivel de complejidad percibió al configurar e integrar bloques de lectura para procesar los datos provenientes de los sensores externos?					
5. ¿Qué tan sencillo le resultó utilizar los bloques de variables para almacenar lecturas analógicas y emplearlas para controlar la activación del bloque Buzzer?					

Figura 67. Formato de la encuesta de satisfacción y evaluación del aprendizaje práctico con la consola.