Metodología de aprendizaje de la programación desde la música

Johnny Germán Cubides Castro

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica y Electrónica
Sede Bogotá, Colombia
2024

Metodología de aprendizaje de la programación desde la música

Johnny Germán Cubides Castro

Trabajo final presentado como requisito parcial para optar por el título de:
Magíster en Ingeniería Electrónica

Director(a):
Prof. Dr. Carlos Iván Camargo Bareño

Línea de investigación:
Sistemas Embebidos
Desarrollo de Habilidades del Siglo XXI

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica y Electrónica
2024

Agradecimientos

Quiero expresar mi sincero agradecimiento al profesor PhD. e Ing. Carlos Iván Camargo Bareño, mi profesor de siempre, quién me brindó la oportunidad de trabajar con diversas poblaciones, aprender el valor de enseñar mientras se aprende, quien ha sido mi mentor en mi proceso de crecimiento profesional y a quien admiro por su calidad humana.

Quiero agradecer también a mi tío Luis Alfredo Castro, mi madre Mary Luz Castro, mi abuelita Rita Antonia Camelo, mi hermana Carolina Cubides, Diana Jimenez, y a mis familiares y amigos, con quienes he transitado en esta experiencia de vida. De una manera u otra, todos me han dado un voto de confianza y una voz de aliento cuando lo he necesitado.

Listado de símbolos y abreviaturas

Abreviaturas	Descripción
ABF	Aprendizaje Basado en Fenómenos
ADC	Conversor Analógico a Digital
API	Interfaz de Programación de Aplicaiones
DAC	Conversir Digital a Analógico
DIP	Empaquetado de Doble Línea
DRC	Verificación de Reglas de Diseño
ERC	Verificación de Reglas Eléctricas
GPIO	Entradas/Salidas de Propósito General
GNU	GNU no es UNIX
HW	Hardware
IDE	Entorno de Desarrollo Integrado
IOT	Internet de las Cosas
LOLC	Laptop Orchestra “Live Coding”
MDF	Medium Density Fiberboard
MIDI	Interfaz Digital de Instrumentos Musicales
MQTT	Message Queuing Telemetry Transport
MVVM	Modelo-Vista-Vista-Modelo
M2M	Comunicación Máquina a Máquina
OSC	Open Sound Control
PC	Pensamiento Computacional
PCB	Print Circuit Board
PD	Public Domain License
QR	Quick Response Code
SMD	Dispositivo de Montaje Superficial
SoC	System on Chip
SW	Software
THT	Tecnología de agujeros pasantes
WiFi	Tecnología de Red Inalámbrica

Resumen

Metodología de aprendizaje de la programación desde la música

Este trabajo aborda la enseñanza del pensamiento computacional como una habilidad transversal del siglo XXI. Si bien, el pensamiento computacional se enseña a través de actividades de programación, es importante abordar la necesidad de mantener la motivación de los estudiantes por su aprendizaje. Tradicionalmente, los cursos de programación se enseñan a través de la solución de problemas, sin embargo, es posible enseñar el pensamiento computacional usando ejes motivacionales, como la robótica, el desarrollo de videojuegos o la música.

En este trabajo, se propone una metodología para enseñar la programación a través de la música. Para tal fin, se estudian los conceptos de la educación musical general, la apropiación social de la ciencia y la tecnología y los temas que son base del aprendizaje de la programación. En este proceso, se desarrolla un material didáctico concreto que incluye un kit de materiales basado en un sistema embebido electrónico, un entorno de desarrollo de programación llamado Catalejo Editor y un material de documentación para el desarrollo de actividades de autoaprendizaje. Esta metodología se centra en la experiencia práctica, lo cual permite a los estudiantes "pensar con las manos". El material es diseñado para un uso flexible al combinar la programación gráfica, la programación textual y la programación de artefactos electrónicos, permitiendo así, el desarrollo de habilidades técnicas de programación para aplicaciones y máquinas.

Finalmente, se desarrolla una experiencia de aprendizaje basada en esta metodología y material didáctico, con una población conformada por estudiantes de un colegio rural y un colegio urbano en Colombia. Esta experiencia enfrentó desafíos relacionados con falta de acceso a Internet y de equipos de programación, los cuales se abordaron mediante encuentros híbridos síncronos y asíncronos, el uso de dispositivos móviles y la réplica de las actividades mediadas por el profesor lideradas por estudiantes para enseñar a otros estudiantes sobre los temas propuestos.

Palabras clave: Material Didactico, Sistemas Embebidos, Desarrollo de Habilidades, Pensamiento Computacional

Abstract

Learning Methodology for Programming through Music

This work addresses the teaching of computational thinking as a 21st-century cross-cutting skill. While computational thinking is taught through programming activities, it is important to address the need to maintain students' motivation for learning. Traditionally, programming courses are taught through problem-solving; however, it is possible to teach computational thinking using motivational axes such as robotics, video game development, or music.

In this work, a methodology is proposed for teaching programming through music. To this end, the concepts of general music education, the social appropriation of science and technology, and the foundational themes of programming learning are studied. In this process, a specific didactic material is developed that includes a kit of materials based on an embedded electronic system, a programming development environment called Catalejo Editor, and documentation material for the development of self-learning activities. This methodology focuses on practical experience, allowing students to "think with their hands". The material is designed for flexible use by combining graphical programming, textual programming, and electronic artifact programming, thus enabling the development of technical programming skills for applications and machines.

Finally, a learning experience based on this methodology and didactic material is developed, with a population consisting of students from a rural school and an urban school in Colombia. This experience faced challenges related to lack of Internet access and programming equipment, which were addressed through synchronous and asynchronous hybrid meetings, the use of mobile devices, and the replication of teacher-mediated activities led by students to teach other students about the proposed topics.

Keywords: Didactic Material, Embedded Systems, Skill Development, Computational Thinking

Lista de figuras

Lista de tablas

Contenido

1 Introducción

La sociedad a nivel mundial demanda cambios rápidos, los cuales buscan mantener un constante avance en campos como la ciencia, la tecnología y la información. En la actualidad, la sociedad se encuentra inmersa en la cuarta revolución industrial o la era digital. Para esta (la sociedad), es crucial que las personas estén constantemente actualizando sus habilidades y competencias para adaptarse a las nuevas necesidades, las cuales constituyen la base de la economía del conocimiento. Sin embargo, no se trata exclusivamente de adquirir conocimientos, también es necesario fomentar el pensamiento crítico en el abordaje de situaciones del entorno y en la generación de alternativas de solución a problemas. El pensamiento computacional (PC) y la creatividad se han identificado como dos habilidades fundamentales para los ciudadanos de la era digital [7], [8].

A pesar de esto, la literatura relacionada se encuentra en una etapa inicial de desarrollo y está lejos de establecer una definición clara del pensamiento computacional (PC), así como la identificación de un método general del cómo los estudiantes aprenden a desarrollar el PC y del cómo evaluar esta habilidad. El PC tiene un gran valor no solo para programadores, sino también para profesionales de diversos campos [9], quienes, al dominarlo, pueden usarlo en soluciones integrales en diferentes áreas o disciplinas como la biología, el diseño, la medicina, el arte, entre otras. Además, en la actualidad, cualquier individuo digital, sin importar su edad, necesita tener habilidades básicas en informática relacionadas con los avances tecnológicos.

Al realizar un análisis sistemático de la literatura sobre PC [8][1] se tiene que la mayoría de ellas se concentran en *la resolución, comprensión y formulación de problemas*. Además, se observa que los artículos revisados carecen de teoría o presentan una base teórica que no es suficiente para la investigación. Sin embargo, se aprecia las metodologías de *El aprendizaje basado en juegos* (gamificación) y *el constructivismo* son base teórica para los artículos relacionados sobre el PC. Por tanto, para proponer una metodología y un enfoque que puedan apoyar el aprendizaje de habilidades del PC es necesario realizar las siguientes preguntas generadoras: ¿Cómo el estudiante aprende habilidades de PC?, ¿Cómo evaluar si los estudiantes han desarrollado habilidades de PC?, y ¿Cómo evaluar si los estudiantes pueden adoptar las habilidades de PC para enfrentar situaciones de la vida real?.

Para responder las preguntas generadoras se requiere continuar con revisiones teóricas; sin embargo, es importante explorar a nivel práctico las experiencias que desde otros enfoques están permitiendo que los estudiantes aprendan a programar. El PC recopila prácticas y herramientas mentales que son clave y se

[1] ...se realizó un análisis de contenido cualitativo inductivo en 125 artículos sobre PC, seleccionados de acuerdo con criterios predefinidos de seis bases de datos y bibliotecas digitales diferentes. ... Estas bases de datos y bibliotecas digitales son:

- Ebscohost
- ScienceDirect
- Web of Science
- Springer
- Biblioteca digital IEEE
- Biblioteca digital ACM

[8]

originan en la informática, las cuales, se dirigen a todas las áreas más allá de la informática. Un ejemplo de adaptación, es el caso del aprendizaje basado en juegos, el cual ha sido usado en la enseñanza de la programación; el estudiante a través de minijuegos e historias puede ir desarrollando habilidades de programación y medir su progreso, por ejemplo, Scratch[2]o de Codecombat[3], plataformas de aprendizaje de la programación de código abierto. Es así, que a través del juego los estudiantes aprenden programación estimulando la curiosidad por aprender y entender conceptos de programación como lo es la abstracción, el pensamiento algorítmico, la resolución de problemas, el reconocimiento de patrones y el pensamiento basado en el diseño [8]. Otro enfoque existente está asociado a la enseñanza de una disciplina o área de conocimiento, por ejemplo: hacer uso de la programación para aprender matemáticas. Una razón se asocia a la necesidad de requerir por parte del estudiante el replanteamiento de sus conocimientos en matemáticas para poder traducirlos en un lenguaje que la máquina entienda, adquiriendo de esa manera, estrategias para la resolución de problemas en contexto [10].

De manera tradicional, los estudiantes aprenden como resultado de un refuerzo, ya sea positivo o negativo; el cual es comportamiento enmarcado en la teoría del condicionamiento operante de la teoría conductual del aprendizaje. En este punto, la motivación del estudiante depende de la emoción frente a lo que le genera el conocimiento en sí mismo o de las dinámicas del espacio de aprendizaje. Es por esto, que el aprendizaje donde el estudiante interactúa con su entorno[4], a diferencia de las clases magistrales, permite que el 90% del material de aprendizaje sea recordado a largo plazo [11]. Es por esto que una estrategia oportuna es definir experiencias de aprendizaje nuevas y atractivas para toda aquella persona dispuesta a aprender.

Los proyectos robóticos facilitan la colaboración y el aprendizaje activo [12]; sin embargo, la población que tiene interés es por lo general personas que ya tienen un enfoque personal con el área. Como estrategia para que las personas se sientan atraídas y motivadas se propone la música como medio para el aprendizaje de la programación. Aprender a programar desde la música puede ser una experiencia divertida y a la vez una herramienta que pueda ser aplicada en el día a día [13]. La programación desde la música se convierte en una alternativa distinta a aprender con actividades informatizadas o desconectadas que buscan promover el PC en el currículo de la educación formal. Es en este punto en el que toma importancia el pensamiento musical computacional que es una relación entre la educación musical y la educación informática [14] para dar respuesta netamente a la estética personal y fomentar el autoaprendizaje.

De tal forma que con el presente trabajo se mostrará una forma de aprender programación desde la música como punto de partida motivacional, haciendo uso de una metodología de aprendizaje activo que permita una aproximación al desarrollo del pensamiento computacional y que además, pueda ser contextualizada en las condiciones y características de quien aprende a través de la apropiación social de la ciencia y la tecnología, lo cual puede permitir el desarrollar de un pensamiento crítico sobre la adquisición de las habilidades del PC aplicables a su propio entorno.

1.1 Objetivo General

Diseñar un entorno de desarrollo hardware-software acompañado de una metodología de aprendizaje que permita una aproximación al desarrollo de la habilidad del pensamiento computacional usando la música como foco motivacional.

[2]Sitio oficial de Scratch: `https://github.com/scratchfoundation`
[3]Sitio oficial de Codecombat: `https://github.com/codecombat/codecombat`
[4]También conocido como aprendizaje activo.

1.2 Objetivos Específicos

1. Diseñar una metodología de aprendizaje que permita una aproximación al desarrollo de la habilidad del pensamiento computacional.
2. Construir documentación de prácticas de programación desde la música a nivel básico como background para acompañar la metodología de aprendizaje.
3. Diseñar el hardware requerido a integrar en el entorno de desarrollo para la programación de sensores y actuadores.
4. Adaptar el software del entorno de desarrollo para que tenga la capacidad de programación de música y programación de hardware que controle sensores y actuadores.
5. Evaluar el comportamiento del entorno de desarrollo integrado.

2 Propuesta metodológica

Esta propuesta metodológica se basa en las metodologías de aprendizaje activo que a diferencia de las metodologías de aprendizaje tradicionales, estas se centran principalmente en el estudiante, modifican la relación existente entre el profesor y el estudiante en función del modo y la responsabilidad sobre la transferencia o la construcción de un conocimiento. Estas metodologías se basan en el constructivismo, el cual presenta la idea que el estudiante puede desarrollar su propio conocimiento a través de la interacción con el objeto de estudio, el entorno, las adecuaciones didácticas y actividades de aprendizaje [15]. El aprendizaje activo está basado en la experimentación, en la capacidad de generar relaciones entre experiencias previas y conocimientos nuevos, permitiendo promover habilidades y competencias como es el caso del pensamiento crítico, habilidades colaborativas, liderazgo, solución de problemas, entre otras. En ese marco referencial, se pretende ubicar al estudiante en el proceso de aprendizaje como eje principal del desarrollo del conocimiento a través de herramientas de participación y espacios de reflexión, procurando la puesta en práctica de los conocimientos en contexto que hayan sido adquiridos. Con respecto al rol del profesor, este se presenta como un facilitador, el cual debe crear espacios propicios para el desarrollo de experiencias de aprendizaje pertinentes, diseñar actividades de enseñanza-aprendizaje que fomenten la participación, es responsable de guiar a los estudiantes y genera procesos de retroalimentación constructiva, el profesor debe tener una postura que permita la adaptación de estrategias que permitan la reflexión y la aplicación práctica de conocimientos [15].

En la tabla **2-1** se presenta una comparativa a manera de síntesis de diferentes metodologías activas que se pueden usar para el diseño de esta propuesta. Al revisar las características, se selecciona la metodología de aprendizaje basado en fenómenos (ABF), ya que esta metodología permite hacer uso de un fenómeno como motivación intrínseca para que el estudiante pueda desarrollar conocimientos a través de la exploración y el abordaje del mismo, permite al profesor tener un rol activo en su propio desarrollo profesional. El ABF promueve el desarrollo de diferentes habilidades enmarcadas en las habilidades del siglo XXI y permite concebir el conocimiento como una construcción dada por la exploración y la experimentación en el estudio del fenómeno.

En la figura **2-1** se presenta la definición y las características de la metodología de aprendizaje basado en fenómenos, donde se representan los temas a tener en cuenta para el diseño de la propuesta metodológica del aprendizaje de la programación desde la música.

Tabla 2-1: Síntesis de metodologías de aprendizaje, adaptado de [4], [5] y [6].

Metodología	**Descripción**	**Roles**	**Enfoque**
Aprendizaje Basado en Proyectos	Esta metodología está orientada al aprendizaje del estudiante a través de la realización de proyectos o tareas concretas. Los estudiantes trabajan en grupos pequeños para abordar un problema o una situación compleja permitiendo el desarrollo de habilidades cognitivas sociales y emocionales	El profesor es un facilitador del aprendizaje, guía y supervisa ofreciendo retroalimentación constructiva, diseña proyectos a través de la colaboración con los estudiantes, evalúa el desempeño individual como grupal a través de criterios que hayas sido establecidos en el proyecto y finalmente es un motivador fomentando la curiosidad y el compromiso en la resolución de problemas complejos. El estudiante cumple un rol principal, en la participación activa de la planificación, ejecución canción de proyectos, teniendo eje un aprendizaje autónomo y autorregulado siendo protagonista de su propio proceso de aprendizaje.	Se enfoca en el desarrollo del pensamiento crítico, resolución de problemas, colaboración y trabajo en equipo, comunicación efectiva, autonomía y autorregulación del aprendizaje, creatividad y habilidades tecnológicas.
Aprendizaje Basado en Fenómenos	Esta metodología se centra en hacer uso de un fenómeno del mundo real como punto de partida para el aprendizaje, permitiendo al estudiante estudiar el fenómeno de manera integral desde su contexto y cruzar las fronteras entre las materias de manera natural, lo anterior permite generación de experiencias de aprendizaje enriquecedoras, autenticidad en el aprendizaje, enfoque en la transferencia y aplicación práctica de la información.	El profesor cumple el rol de ser el diseñador del proceso de la enseñanza y aprendizaje, ser el facilitador del aprendizaje y colaborar en su propio desarrollo profesional. El rol del estudiante es fundamental y es el centro del proceso educativo, ese creador del conocimiento, hace participación en el proceso de aprendizaje, es autónomo y hay motivación intrínseca y realiza aplicación práctica de la información.	Las habilidades que permite desarrollar esta metodología de aprendizaje son: habilidades de resolución de problemas, pensamiento crítico, colaboración y trabajo en equipo, autonomía y autorregulación del aprendizaje, aplicación práctica de conocimientos.
Aprendizaje Cooperativo	Estrategia pedagógica que fomenta la colaboración entre los estudiantes para lograr objetivos de aprendizaje comunes, los antes trabajan en grupos pequeños, dependientes y heterogéneos, miembro es responsable de su aprendizaje.	El rol del profesor es ser el facilitador, organiza los grupos, establece normas y expectativas, apoya y orienta, tiene una actitud de observador activo y promotor de la reflexión. El profesor contribuye al desarrollo de habilidades sociales, cognitivas finales de los estudiantes, así como el logro de los objetivos de de aprendizaje colaborativo. El rol del estudiante es ser el protagonista del aprendizaje, diseñador y gestor del aprendizaje, gestor de recursos y tiempo, autoevaluador, colaborador y comunicador efectivo, desarrollador de habilidades sociales y emocionales.	Se enfoca en el desarrollo de habilidades sociales, habilidades de Liderazgo compartido, unidades de pensamiento crítico y resolución de problemas, de comunicación, autoconocimiento y autorregulación y colaboración intercultural.
Simulación y Juegos	Esta metodología propone acondicionar un espacio donde el estudiante pueda aprender de manera interactiva a través de una experiencia vivida, situaciones a las que posiblemente no esté preparado el estudiante en la vida real. Da la oportunidad al estudiante de expresar sentimientos al respecto de sus aprendizajes y experimenta con nuevas ideas y procedimientos.	El profesor tiene la responsabilidad de manejar y dirigir situaciones, establecer la posible simulación o las dinámicas de juego a realizar e interroga sobre las situaciones a afrontar. El estudiante cumple un rol activo a través de la simulaciones propuesta, experimenta o juega con ellas, reacciona a las condiciones y variables emergentes.	Se enfoca en el desarrollo de habilidades específicas para enfrentar y resolver problemas de situaciones, en dar un valor aquello que se va descubriendo a través de la creación y utilización de sus propias experiencias y fomenta el desarrollo de capacidades interpersonales.

Figura 2-1: Definición y características de la metodología de Aprendizaje Basado en Fenómenos. Fuente [1]

2.1 El pensamiento computacional como fenómeno de estudio

Cuando una persona quiere abordar situaciones problemas ya sea en el contexto de la ciencia, tecnología o del día a día, requiere de ciertas herramientas que ha adquirido a lo largo de la vida, la manera en la que puede afrontar la situación problema o dar solución a esa situación va a depender de la optimización del método que haya aplicado.

En el contexto actual del desarrollo de diferentes áreas del conocimiento y de disciplinas se encuentra que el pensamiento computacional es transversal a las distintas herramientas que se usan para optimizar procesos y generar resultados. Es así, que el pensamiento computacional a través de las actividades de programación son la base de las tecnologías o de las herramientas del siglo XXI.

Para una persona, el poder desarrollar la habilidad del pensamiento computacional podría permitirle desarrollar en él las habilidades lógicas para así tener una herramienta que puede usar en el abordaje de las situaciones problemas en cualquiera de los contextos anteriormente mencionados, sin embargo, desarrollar esta habilidad requiere un esfuerzo y para poder cumplir con éxito este objetivo es necesario mantener la motivación. Para mantener la motivación se plantea el uso de la educación musical general como un tema atractivo, el cual va a requerir un esfuerzo para mantener la relación entre la música y las temáticas de estudio del pensamiento computacional.

Mientras que se desarrolle el estudio del fenómeno computacional se van a plantear diferentes situaciones que tratan de relacionar el Por qué y el para qué del pensamiento computacional, lo que puede significar el pensamiento computacional para la persona que está indagando sobre él, la relación que puede existir entre diferentes habilidades que ha trabajado a lo largo de la vida como el pensamiento crítico, matemáticas, creatividad, entre otras habilidades o competencias del siglo XXI.

Haciendo uso de la metodología de aprendizaje basado en fenómenos, los estudiantes, a través de la indagación, de la búsqueda de los vacíos de conocimiento, de la elaboración de un artefacto, del objetivo tangible o de su propósito y la realización de la documentación, el estudiante podrá hacer un seguimiento a su propio proceso de aprendizaje.

2.2 Ejes temáticos

Los ejes temáticos permiten definir el contenido de las actividades para alcanzar los objetivos propuestos. Para poder abordar el pensamiento computacional como un fenómeno de estudio se plantea la integración de temáticas asociadas al estudio del contexto, bases de programación y motivación. Para el estudio del contexto se plantea las temáticas asociadas a la apropiación social de la ciencia, tecnología e Innovación para el desarrollo humano, mientras que para el eje motivacional se aborda los temas asociados a la educación musical.

2.2.1 Temas en la enseñanza de la programación

La programación es una actividad que permite el desarrollo del pensamiento computacional abarcando distintas áreas de conocimiento y disciplinas. Se han desarrollado distintas técnicas de programación para responder a propósitos específicos según el contexto. Para el propósito de esta propuesta pedagógica se planteó abarcar los temas bases usados en la enseñanza de la programación en universidades [16] y [17]:

- **El camino hacia el programa**: Camino hacia el programa: Es la introducción sobre lo que es la programación, reconocer la diferencia entre los lenguajes naturales y formales, identificar los procesos de depuración de código y conocer los principios para aprender a programar.
- **Variables y operadores**: Se aborda la definición de los valores, los tipos de los valores y nombres de variables para almacenar datos, también se estudia las operaciones que se pueden realizar con los valores.
- **Control de flujo**: Se aborda las diferentes estructuras de control de flujo que permiten tomar decisión sobre la ejecución de un programa.
- **Repeticiones y bucles**: Se estudian los conceptos de iteración y bucles explorando las estructuras de repetición sobre bloques de código para el procesamiento de datos.
- **Funciones**: Se aborda el concepto de funciones para organizar y reutilizar código, se enfatiza en que estas hacen parte fundamental de la programación.

2.2.2 Temas en la enseñanza de la música

La música como expresión artística ha sido parte de la historia de la humanidad, desde el desarrollo del lenguaje y la creación de artefactos tecnológicos que ha permitido el desarrollo de la expresión cultural, la expresión emocional y la conservación de tradiciones.

La historia de la música abarca diferentes estilos, géneros y tradiciones, los cuales han evolucionado según transformaciones tecnológicas, culturales y de comportamiento humano, que por ejemplo, en cada periodo histórico han dejado huella en el desarrollo de los diferentes instrumentos musicales.

En el siglo XX la música manifestó una transformación de diversidad y de experimentación dando lugar a diferentes géneros que hoy en día con la influencia de la revolución electrónica y digital dan paso al desarrollo de formas musicales y expresiones artísticas abstractas o concretas, donde pueden intervenir máquinas, las cuales se involucran en la interpretación o creación de sonidos ya sea a través de la construcción de artefactos o de síntesis a través de tareas de hardware o software.

Educación musical

La educación presenta problemas complejos en los cuales la participación de la música puede desempeñar un rol importante, ya que esta enriquece al ser humano a través de experiencias relacionadas con el sonido, el ritmo y la armonía, con las cuales, la música impulsa la vida interior apelando a las facultades humanas como la voluntad, la creatividad y la inteligencia, lo cual, hace de la música un factor cultural indispensable. La educación musical une aspectos artísticos y científicos de la música armonizando el saber, la sensibilidad y la acción, por tanto, la cultura, la vida y la técnica deben complementarse para aportar en el desarrollo de una nueva humanidad que corresponda a las necesidades actuales [19]

La educación musical se puede poner en dos vías principales: **La educación musical general**, en la cual se centra la propuesta y la educación musical artística. Para diferenciar ambos tipos de educación, se tiene que, la educación musical general pretende en el estudiante el desarrollo de habilidades básicas, las cuales se asocian a la comprensión de los fundamentos musicales a nivel básico, estos incluyen la historia musical, la apreciación auditiva y la teoría musical los cuales pueden permitir el gozo y la participación por parte del estudiante en la música a distintos niveles de sensibilidad. Por otro lado, la educación musical artística se enfoca en la formación profesional, donde los estudiantes con talento e intereses musicales se enfocan en el desarrollo de habilidades avanzadas en áreas como la interpretación, la composición y la dirección. Con lo

anteriormente expuesto, se suscribe que el objetivo de esta propuesta pedagógica no es formar músicos, sino más bien fomentar el interés del público en general por desarrollar habilidades de programación, así como también temas relacionados con la sensibilidad musical.

A continuación, se presentan los diversos temas tratados en la educación musical que son aplicados en el ambiente general como en el artístico, los cuales son descritos en [18] y [19] y que son agrupados en subtemas para el propósito de esta propuesta.

El sonido, el silencio y el ruido El sonido es el elemento fundamental de la música, el cual es percibido por el oído produciendo una sensación dada por el movimiento vibratorio de los cuerpos en un medio elástico, entre los fenómenos del sonido que se aprecian en la música se encuentra el eco, la reverberación y la resonancia. Por otro lado, el silencio se puede definir como la ausencia del sonido, sin embargo, desde la música, se considera casi imposible lograr el silencio debido a que el cuerpo vivo está en constante vibración y la mente se mantiene en constante actividad, finalmente, el ruido se percibe como un sonido no deseado, que carece de armonía o que entorpece una entonación determinada[18].

Parámetros del sonido Los parámetros del sonido son variables físicas que se pueden percibir e interpretar dándole una finalidad al sonido. Entre los distintos parámetros del sonido se incluye la altura, la duración, la longitud temporal, la intensidad, el timbre, la calidad o color del sonido, entre otros.

Instrumentos musicales Los instrumentos musicales son dispositivos que al entrar en contacto con ellos pueden vibrar produciendo sonidos asociados a formas musicales. Estos se pueden clasificar en diferentes categorías, las cuales pueden ser instrumentos de cuerda, de viento, de percusión e inclusive electrónicos. Cada instrumento produce sonido de maneras diferentes asociados a las técnicas de construcción, materiales y mecanismos que le dan una particularidad en el timbre permitiendo su identificación.

El lenguaje musical La música es arte que se manifiesta por el aire propagándose hasta desaparecer a diferencia de otros artes. Esta, la música, se desarrolla a través del tiempo y luego de ser interpretada permanece únicamente en las memorias, el lenguaje musical es la expresión gráfica del sonido[18], el cual está formado por símbolos y reglas que se emplean para expresar y comunicar conceptos musicales. A continuación se listan aquellos que son elementos base de este lenguaje tomados de [18]:

- **Altura**: Se refiere a la percepción de los sonidos en términos de agudos o graves. La altura se asocia a la frecuencia de oscilación de la vibración de un sonido con respecto a una altura central, donde las frecuencias más altas a esta producirán sonidos más agudos y frecuencias más bajas producirán sonidos más graves. La representación de la altura se representa en el pentagrama a través de notas y claves musicales.
- **Duración**: La duración es una cualidad del sonido que identifica los sonidos largos de los sonidos cortos, se representa a través de figuras tanto para las notas como para los símbolos que de manera relativa indican la duración en tiempo de estas figuras.
- **El Ritmo**: Representa el movimiento, que sin este no sería posible la música. El ritmo se representa a través del pulso, el compás, entre otras, las cuales son la combinación de distintas duraciones.
- **La dinámica**: La dinámica hace referencia a la intensidad de sonido a través de unos términos denominados matices que indican el volumen del sonido, por ejemplo, piano para referirse a suave, forte para referirse a fuerte, crescendo para referirse a aumentar o decrescendo para decrementar.
- **La melodía**: Se trata de una sucesión lineal de notas que acompañadas de distintas alturas y ritmos configuran la idea musical de una pieza[18],

- **La armonía**: Hace referencia a la combinación de notas simultáneas (ejecuciones concurrentes) las cuales dan profundad al sonido musical, en el lenguaje musical se hace uso de la tonalidad, modalidad, los acordes, las disonancias, entre otras.
- **La textura**: Se refiere al tejido entre la melodía y la armonía de una composición, de los cuales existen los tipos de monódica, polifónica, homofónica y no melódica.
- **La forma musical**: Se refiere a la forma y estructuración del material sonoro, donde se mezclan, voces e instrumentos, texturas, cultas o populares, entre otras.

Relación entre los conceptos de la educación musical y la programación

Para poder hacer uso de los temas base de la educación musical en la enseñanza de la programación, se requiere identificar patrones que puedan ser relacionados con conceptos de la programación. En vista de que ambos comparten su representación en lenguajes textuales y gráficos y además, la música se manifiesta en un medio físico se pueden realizar las siguientes apreciaciones:

- **Sonido, silencio y ruido:** Para la representación de estas características a través de la programación se puede hacer uso de datos de audio o de síntesis por medio de algoritmos, los cuales son expresados al medio físico a través del DAC, por ejemplo, el silencio se puede manifestar a través de una instrucción de esperar un tiempo determinado que genere información de salida con intensidad igual a cero, con respecto al ruido, se puede generar datos de frecuencia e intensidad aleatoria.
- **Parámetros del sonido:** Con respecto a los parámetros del sonido, estos pueden ser representados a través de variables, por ejemplo, la altura, podría ser representado por espacio de memoria que contenga un número entero que represente la frecuencia, la intensidad podría ser otro campo de tipo flotante que represente el volumen, inclusive, hacer uso de protocolos MIDI que sirve como formato estándar para compartir datos a instrumentos musicales. El estado de los parámetros del sonido pueden tener cambios a través de la ejecución de instrucciones condicionales que se evalúan en la ejecución de una tarea.
- **Instrumentos musicales:** Los instrumentos musicales tienen características y capacidades propias, las cuales permiten su identificación y clasificación en grupos de instrumentos. Al realizar un ejercicio de abstracción, se puede traducir estas características a atributos y funciones en tareas que puede ejecutar una máquina, de tal manera que en un proceso de síntesis se puedan alterar los parámetros del sonido y obtener un resultado similares a aquellos producidos por los instrumentos musicales. Además, es posible la creación de instrumentos digitales que producen nuevos sonidos a partir de la electrónica.
- **El Lenguaje musical:** En vista de que el lenguaje de la música y el lenguaje de las computadoras contienen información sobre el cómo ejecutar o proceder frente a la producción de un resultado y que estos se pueden expresar de manera textual y gráfica, se puede extraer del lenguaje musical patrones de sintaxis y semántica que se pueden traducir a un lenguaje de programación, de tal manera que los parámetros de la música, el ritmo, la armonía, entre otros, pueden ser representados por datos y operaciones propias de los lenguajes de programación de las computadoras.
- **Otros elementos musicales:** Distintos elementos del lenguaje musical tienen paralelos con la estructura y la organización de los procesos ejecutados en una computadora, por ejemplo, el ritmo se puede asociar al flujo y la secuencia de instrucciones, la dinámica se puede relacionar a las variaciones en el tiempo o velocidad de ejecución de un programa, inclusive, la interpretación de más de un instrumento como pasa en las orquestas se puede representar a través de procesos concurrentes de instrumentos o sonidos sintetizados por tareas digitales.

Con este ejercicio de búsqueda de relaciones entre ambos lenguajes, el cual se puede extender, se señalan aspectos que se pueden aprovechar en la enseñanza de la programación. Estos elementos pueden ser el punto

de partida para el diseño de actividades didácticas junto a materiales que proporcionen un camino para el aprendizaje del pensamiento computacional a través de la programación.

2.2.3 Apropiación social de la ciencia y la tecnología

La apropiación social de la ciencia y la tecnología como eje temático presenta un esfuerzo adicional para quien hace parte de una experiencia de aprendizaje, moviliza a la necesidad de contextualizar los conocimientos construidos con su propia experiencia de vida y diversifica las posibilidades de darle sentido al ejercicio didáctico descubriendo relaciones que fortalecen el aprendizaje a lo largo de la vida. Para poder desarrollar una experiencia que involucre este eje temático es importante poder adaptar los materiales y las didácticas al escenario en el cual se desarrolla el proceso, estos escenarios pueden ser diversos según localización geográfica como también los recursos disponibles tecnológicos, formación de los facilitadores, interesados, entre otros. Para el caso de Colombia, existen brechas tecnológicas que presentan limitaciones en acceso a servicios como es el caso de Internet y de electricidad [20], lo cual, para esta propuesta metodológica acarrea dificultades en el desarrollo de conocimientos aplicables en contexto. Por tanto, además de ser este un eje temático para alcanzar los objetivos de adquirir conocimientos en programación, en sí misma se puede convertir en una meta, en vista de las dificultades que acarrea el desarrollo de actividades educativas tecnológicas y la necesidad de adaptar la experiencia en función de lograr conocimientos pertinentes.

La apropiación social del conocimiento es un proceso que implica a toda la sociedad, incluyendo diversos sectores y comunidades, las cuales participan activamente en la generación, difusión, comprensión y aplicación de conocimientos científicos y tecnológicos, modificando la idea tradicional donde este proceso es exclusivo de expertos, dando la oportunidad a la sociedad a través de mecanismos de participación la posibilidad de aportar a través de sus experiencias y saberes locales en el desarrollo de distintos tipos de conocimiento, involucrándose así en los procesos de investigación, innovación y toma de decisiones [21].

Algunos ejes temáticos de la apropiación social del conocimiento incluyen la participación ciudadana, el fomento de la comunicación bidireccional, la valoración de la pluralidad de saberes, la contextualización de la ciencia y la tecnología en su entorno socioeconómico y político, y la promoción de la innovación social. Estos aspectos son fundamentales para abordar las disparidades tecnológicas y promover un acceso equitativo al conocimiento en Colombia. A continuación se describen cada una de ellas (tomadas de [21]) para ser tenidas en cuenta en la contextualización de las diferentes acciones didácticas.

1. **Participación ciudadana:** Se centra en la importancia de la participación activa de los ciudadanos para la toma de decisiones y en la definición de las políticas científicas y tecnológicas.

2. **Comunicación bidireccional:** En esta se resalta la necesidad de establecer canales de comunicación efectivos entre los expertos y los ciudadanos, de modo que se pueda fomentar un diálogo constructivo y promoviendo la comprensión mutua.

3. **Pluralidad de saberes:** Aquí se reconoce la existencia de múltiples formas de conocimiento, que incluyen saberes ancestrales, indígenas y afrodescendientes, así como conocimientos populares y científicos. Se fomenta la valoración y el diálogo entre estos diferentes saberes.

4. **Contextualización:** Se enfatiza en la importancia de situar la ciencia y la tecnología en su contexto social, económico y político, se reconoce que los procesos de apropiación no son neutrales y dependen de los actores involucrados junto a los intereses relevantes.

5. **Innovación social:** Fomenta la innovación social, la cual, es entiende como la generación de soluciones sostenibles y creativas a los problemas sociales y ambientales, a través de la participación activa de los ciudadanos valorando sus saberes locales.

El enfoque de esta propuesta metodológica busca contextualizar la importancia desarrollar habilidades del siglo XXI, el desarrollo del pensamiento crítico a través de experiencias contextualizadas, y que en el caso de la programación, se promueva su integración como parte de un proceso continuo de aprendizaje en distintas áreas de conocimiento. Se pretende que el pensamiento computacional, en particular, capacite al estudiante para concebir la programación como una herramienta para resolver problemas situados o, al menos, para estructurar su pensamiento cuando desee emprender iniciativas de interés.

2.3 Diseño metodológico

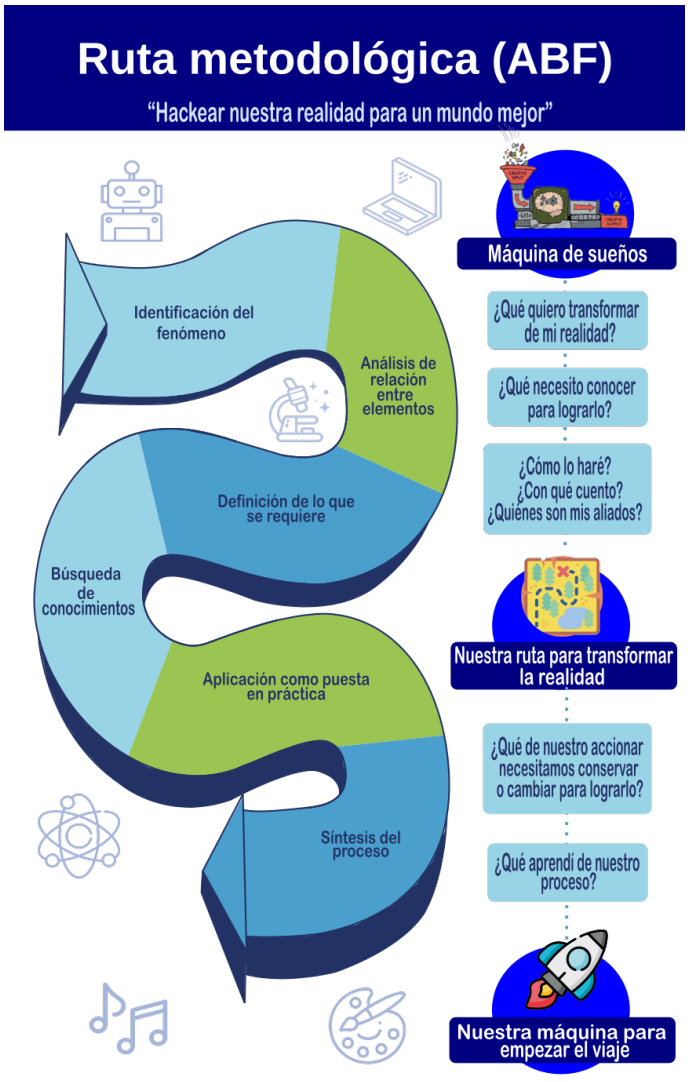


Figura 2-2: Ruta metodológica basada en ABF que propone diferentes artefactos y preguntas orientadoras, aplicable a diferentes experiencias de aprendizaje.

El diseño metodológico que se presenta a continuación, además de enmarcarse sobre una metodología de enseñanza, se sustenta en la experiencia adquirida en los diferentes programas o proyectos de aprendizaje para educación secundaria, desarrollados por la Universidad Nacional de Colombia a través de su función misional

de extensión en el departamento de ingeniería eléctrica y electrónica de la sede de Bogotá. Experiencias como aprendizaje basado en IOT (2017), desarrollo de habilidades para la cuarta revolución industrial (2018), diplomado de programación y música (2018), fortalecimiento de las competencias de los jóvenes de media para afrontar los retos del siglo XXI (2021), han permitido reflexionar sobre la pertinencia de las acciones didácticas, material didáctico entre otras.

Tabla 2-2: Matriz de diseño de propuesta metodológica, fases 1, 2, 3 y 4

Fases del ABF	**Objetivo**	**Objetivos específicos**	**Competencias del siglo XXI involucradas**	**Ejes temáticos**
1. Identificación de fenómenos . 2. Análisis de la relación entre elementos del fenómeno.	Reconocer a cada uno de nosotros como agentes de cambio que transforman y se transforman según la lectura de la realidad y la relación con la tecnología	1. Presentar el aula virtual/digital como un espacio seguro para expresar intereses, motivaciones, exploración de conocimientos y para propiciar aliados. 2. Plantear la realidad como un ambiente susceptible a cambios y transformaciones. 3. Reconocer la tecnología como un agente que se transforma y que transforma nuestra realidad (discutir sobre el impacto de la tecnología en las sociedad y su historia). 4. Explorar los elementos necesarios que se requieren para los procesos tecnológicos 5. Explorar la relación entre tecnología, intereses y nuestro contexto.	(Maneras de pensar) Pensamiento crítico, (Maneras de trabajar) comunicación, (Herramientas para trabajar) Alfabetización informacional, Alfabetización en tecnologías de la información y comunicación.	Apropiación social de la ciencia y la tecnología
3. Determinación de lo que se quiere hacer con el fenómeno (opciones de abordaje) y definición de los requerimientos (conocimientos - habilidades - recursos generales) para abordarlo. 4. Búsqueda de lo requerido	Apropiar la exploración del fenómeno identificado a través de una ruta de estudio propuesta desde la lectura de la realidad de las y los estudiantes, sus emociones e intereses	1. Fortalecer la confianza de los estudiantes con relación al aula como un lugar seguro y de respeto para compartir ideas y pensamientos diversos. 2. Incentivar al estudiante la exploración y apropiación del fenómeno implicando sus intereses, motivaciones, emociones y su realidad. 3. Crear una ruta de trabajo colaborativa sobre la exploración del fenómeno.	(Maneras de pensar) Creatividad e innovación, (Maneras de trabajar) colaboración y comunicación	Apropiación social de la ciencia y la tecnología. Educación musical. Pensamiento computacional

En la figura **2-2** se observa la ruta metodológica con sus diferentes fases y artefactos. Esta ruta se plantea de manera general, con la idea de ser aplicable a diferentes experiencias de aprendizaje, que para este caso, es el punto de partida para el diseño del banco de actividades y las acciones didácticas. De la ruta metodológica se indica los siguientes hitos como resultados a lograr:

1 En las fases de identificación de fenómenos y análisis de la relación entre elementos, los estudiantes son contextualizados sobre el fenómeno a estudiar, que en el caso particular, por tratarse de la programación se centra en el pensamiento computacional. En estas fases, el estudiante cuenta con herramientas que le permiten hacer una lectura de su realidad, vinculando con el mundo las emociones, experiencias locales y motivaciones. De este ejercicio se desarrolla el artefacto “explorando mi realidad”, el cual, puede dar cuenta de relaciones existentes entre el individuo, la sociedad, la ciencia, la tecnología y la música.

2 En las fases de determinación de lo que quiere hacer con el fenómeno y la búsqueda de conocimientos,

el estudiante requiere percibirse como parte activa de su realidad, de tal manera que pueda plantear transformaciones de esta vinculando el mundo, en esta parte, el estudiante identifica intereses, vacíos de conocimiento y manifiesta una perspectiva de esto a través de la "máquina de sueños", la cual es un artefacto visual con preguntas orientadoras sobre los temas señalados en esta fase. Con respecto al los temas asociados a la música y programación, las acciones didácticas se orientan en mostrar la relación e importancia de estas en el desarrollo humano.

3 En la fase de aplicación como puesta en práctica, las actividades se plantean en la lógica de "pensar con las manos", donde el eje principal es la exploración de la caja de herramientas, que para el caso particular hace referencia material concreto, con el cual se desarrollan las actividades de programación desde la música. Las acciones didácticas en este punto comprometen gran parte del tiempo de la experiencia de aprendizaje, ya que en esta fase, es donde se manifiestan las posibilidades de intervenir la realidad, que en caso de la música, se convierte en el desarrollo de sus propias interpretaciones y en otros casos, sus propias creaciones musicales.

4 En la etapa de síntesis, los estudiantes podrán establecer la manera de comunicar sus ideas y reflexiones, tanto de la realidad como de ellos mismos, en este punto se manifiesta el interés de la toma de decisiones basadas en la búsqueda de información y criterio. Los estudiantes socializan sus artefactos construidos los cuales son el vehículo para expresar su propia experiencia en la programación desde la música y sus proyecciones los cuales se manifiestan en un espacio llamado "Nuestra máquina para empezar el viaje".

Para terminar esta sección, se presenta la matriz de diseño en la tabla **2-2** y **2-3** junto a la rúbrica de evaluación en la tabla **2-4**, las cuales han sido usadas en experiencias asociadas al ABF.

Tabla 2-3: Matriz de diseño de propuesta metodológica, fases 5 y 6

Fases del ABF	**Objetivo**	**Objetivos específicos**	**Competencias del siglo XXI involucradas**	**Ejes temáticos**
5. Aplicación como puesta en práctica	Acompañar a las y los estudiantes en la exploración y apropiación del fenómeno a través de escenarios propuestos, herramientas e información de interés que permitan problematizar situaciones y además sirvan de guía e insumos en la consecución de la ruta de estudio propuesta por los estudiantes.	1. Dinamizar los espacios de trabajo para propiciar en los estudiantes la exploración y apropiación del fenómeno. 2. Incentivar el desarrollo de habilidades en las y los estudiantes para que aprendan a hacer en contexto. 3. Conocer herramientas asociadas al fenómeno de estudio que puedan ser implicadas en la exploración realizada por las y los estudiantes	(Maneras de pensar) pensamiento crítico, creatividad e innovación, toma de decisiones, resolución de problemas, (Maneras de trabajar) colaboración y comunicación.	Enseñanza de la programación. Educación musical: Relación de concepto musicales y programación
6. Síntesis del proceso	Compartir sobre los aprendizajes identificados en el proceso de exploración y apropiación del fenómeno	1. Reconocer del proceso de manera grupal e individual las habilidades exploradas en contexto. 2. Realimentar a las y los estudiantes acerca del proceso vivido en las seis semanas. 3. Proponer una ruta a seguir después de la experiencia de aprendizaje realizada en las seis semanas	(Maneras de vivir en el mundo) Ciudadanía - local y global, Responsabilidad personal y social, (Maneras de trabajar) comunicación, (Maneras de pensar) Aprender a aprender / metacognición	Pensamiento Computacional. Apropiación social de la ciencia y la tecnología

Tabla 2-4: Rúbrica de evaluación de la experiencia de aprendizaje asociada a las fases del ABF

OBJETIVO GENERAL DEL CURSO:	Reconocer a cada uno de nosotros como agentes de transformación de nuestra realidad a través de la técnología				
Tipo de evaluación	Hetero-Auto-Co-evaluación				
Niveles de evaluación	A: Es un reto, B: Estoy superando el reto, C: Estoy construyendo una fortaleza, D: Es una fortaleza				
ARTEFACTO DE REFERENCIA EN EL PROCESO	Construcción individual denominada "Mi máquina de sueños" donde se implican las emociones, motivaciones del estudiante y se relacionan con la lectura de su realidad y el fenómeno que se está explorando				
Momento:	Final de la fase 2 del ABF				
Item a evaluar	Comunico de manera asertiva lo que me gusta, lo que me interesa y lo que quiero aprender	A	B	C	D
Item a evaluar	Vinculo mis emociones, motivaciones e intereses con acciones concretas para transformar o cambiar mi realidad.	A	B	C	D
Item a evaluar	Indago sobre los elementos de mi realidad (IDE, barrio, casa) que nos afectan o se relacionan	A	B	C	D
Item a evaluar	Reconozco necesidades de elementos y actores requeridos para el saber hacer en contexto	A	B	C	D
ARTEFACTO DE REFERENCIA EN EL PROCESO	Construcción grupal denominada "Nuestra ruta para transformar la realidad", el cual es un artefacto que están construyendo desde la fase 2, pero que sufre transformaciones dada por el diálogo de los estudiantes y la claridad dada por la exploración del fenómeno y su relación con su realidad				
Momento:	Final de la fase 4 del ABF				
Item a evaluar	Propongo una ruta para explorar los vacíos de conocimiento para el actuar/saber en contextos	A	B	C	D
Item a evaluar	Realizo acciones que aborden mi realidad y que pueden propiciar cambios en la manera de relacionarme con el mundo	A	B	C	D
Item a evaluar	Facilito estrategias que permitan articular me con mis compañeros para llegar a un mismo fin	A	B	C	D
Item a evaluar	Aporto al desarrollo de avances de lo que nos hemos propuesto realizar a través de mi rol en el equipo	A	B	C	D
ARTEFACTO DE REFERENCIA EN EL PROCESO	Se trata de un artefacto denominado "Nuestra máquina para empezar el viaje" creado en equipo a través de la ruta diseñada de exploración del fenómeno, el artefacto servirá para transmitir o comunicar el proceso vivido y sus aprendizajes en la explicación del fenómeno y su relación con la realidad.				
Momento:	Final de la fase 6 del ABF				
Item a evaluar	Reflexiono sobre la experiencia de aprendizaje vivida identificando aquello que puede servirme a lo largo de mi vida	A	B	C	D
Item a evaluar	Exploro y propongo herramientas que permitan la realización de aquello que nos hemos propuesto	A	B	C	D
Item a evaluar	Trabajo estrategias orientadas que faciliten el trabajo colaborativo para llegar a un mismo fin	A	B	C	D
Item a evaluar	Me identifico como un agente activo de mi realidad y reconozco que puedo tomar acciones desde mi ahora	A	B	C	D
Item a evaluar	Expreso mis aprendizajes e ideas de distintas maneras a través de artefactos que he creado con mis compañeros	A	B	C	D

2.4 Estrategias y herramientas didácticas para la experiencia de aprendizaje

2.4.1 Acciones didácticas

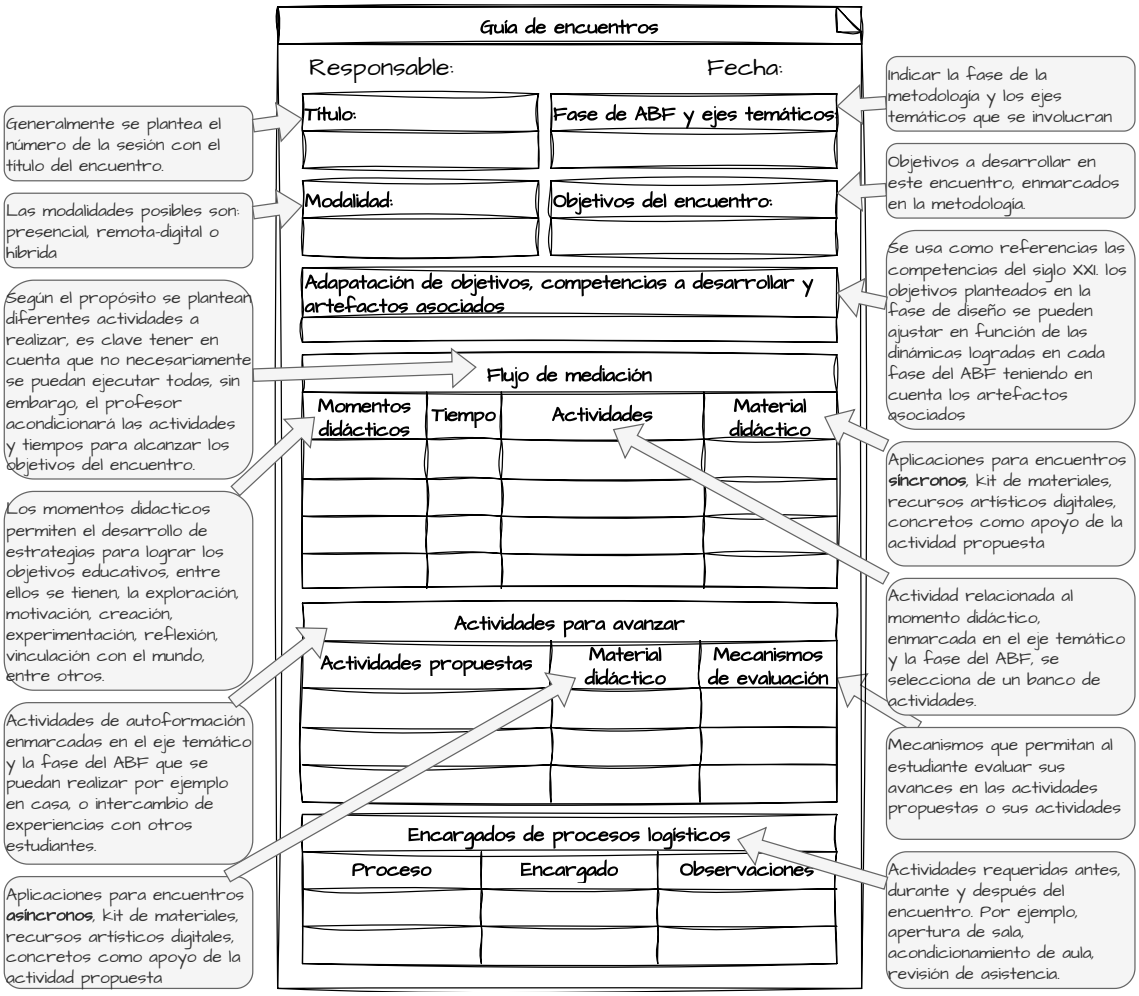


Figura 2-3: Guía de diseño de encuentros síncronos, asíncronos, y actividades para avanzar

Las acciones didácticas son las actividades o estrategias planificadas para auspiciar el aprendizaje significativo de los participantes. Las acciones didácticas propuestas son encuentros síncronos y asíncronos y actividades de formación. Con respecto a los encuentros síncronos se pueden dar de manera presencial o de manera virtual, si son presenciales se pueden diseñar ajustando los recursos de aula, en el caso de ser virtuales, se hace a través de plataformas virtuales, proponiendo que los espacios donde se recibe la formación digital, haciendo uso de las plataformas virtuales, se pueda considerar como un laboratorio para las experiencias de aprendizaje. Con referencia a los encuentros asíncronos se pueden realizar a través de plataformas virtuales, las cuales contengan las herramientas para compartir información sobre los procesos de aprendizaje, los cuales no requieran una respuesta inmediata por parte de los demás participantes de la experiencia. Las actividades de autoformación pueden ser guías de laboratorio o guías de encuentro que hayan sido adaptadas

por parte de quien dirige la experiencia de aprendizaje haciendo uso de un banco de actividades. En la figura **2-3** Se propone un formato para el proceso de diseño de la experiencia de aprendizaje que permita involucrar objetivos, competencias, herramientas, momentos didácticos y artefactos a crear por parte del estudiante. Se deben crear todas las guías de los encuentros a través de bancos de actividades ajustados según cada propósito o momento didáctico.

2.4.2 Material didáctico

El material didáctico consta de herramientas y de material concreto que junto a las acciones didácticas se usa para facilitar el aprendizaje de los estudiantes. El material didáctico que se propone para facilitar el aprendizaje de la programación es:

- **Guías para el desarrollo de actividades síncronas**: Estas guías están constituidas por un Banco de posibilidades actividades y de momentos didácticos que diseña el profesor.
- **Guías de actividades de autoaprendizaje asíncronas**: Estas guías se diseñan para que el estudiante pueda vivir experiencias que pueda modificar y a través de preguntas orientadoras se inicie la exploración de nuevos conocimientos.
- **Material concreto**:
  - **Kit de materiales**: Consiste en la caja de herramientas la cual sirve para materializar ideas, este material se concibe como un material incompleto para que el estudiante se involucre en completarlo.
  - **Entorno de desarrollo integrado**: El entorno se diseña para que haga de interfaz entre el material concreto y las maneras de abordar las iniciativas por parte del estudiante.

2.5 Producto

En la figura **2-4** se observa la estructura de la propuesta y el resultado final esperado. Desde el momento que el estudiante inicie su experiencia de aprendizaje con la metodología propuesta se enfrenta a una situación que requiere comprender para poderla afrontar. El estudiante recibe un acompañamiento y un material que le permite materializar sus ideas dándole una herramienta adicional en el proceso de aprendizaje donde se pueden reconocer nuevos aprendizajes como vacíos de conocimiento. El producto solicitado a los estudiantes consta de dos partes, el artefacto y la documentación. Por un lado, el artefacto el cual es la materialización de las iniciativas del estudiante, este elemento se puede convertir en el vehículo para el estudiante al manifestar aquellos aprendizajes. Con respecto a la documentación permite reflexionar sobre la experiencia vivida en donde se pueda reconocer nuevos aprendizajes, vínculos entre diferentes temas y relacionados y rutas de acción a futuro. Para el caso del producto de esta metodología de aprendizaje se plantea el uso de técnicas básicas de la educación musical como la tonalidad mayor, menor, pentatónica, entre otras para construir un artefacto tangible que haga alusión a afectos. Finalmente, se realiza la socialización de los resultados a los compañeros de experiencia e interesados.

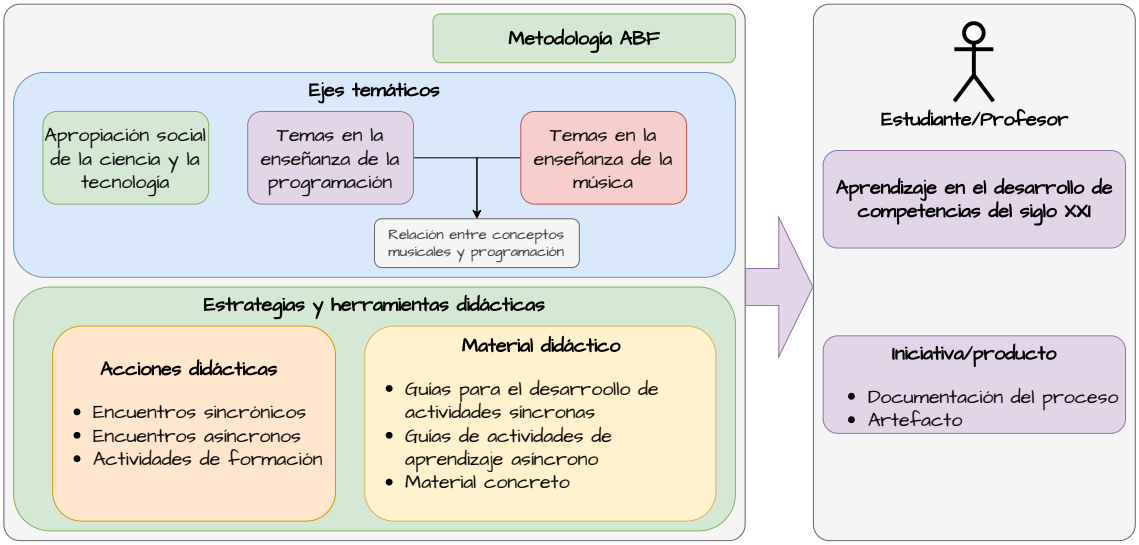


Figura 2-4: Estructura de la propuesta didáctica y producto final esperado

3 Diseño de material concreto

En el capítulo 2 se determinó la metodología a seguir para el aprendizaje del pensamiento computacional a través de actividades de programación haciendo uso de la música como eje motivacional y contextualizado en la apropiación social de la ciencia y tecnología. En este capítulo se abordará el **diseño y adecuación del material concreto**, el cual, es el vehículo que acompañado por diferentes actuaciones didácticas permitirá facilitar el proceso de enseñanza-aprendizaje. Para esta finalidad se hará uso de la metodología de desarrollo mostrada en la figura **3-1** la cual describe el diseño e implementación de un sistema embebido [2]. Esta metodología se adaptó al propósito del desarrollo del material concreto debido a que este, además del diseño de la tarjeta de desarrollo, requiere la creación de un IDE, un kit de materiales y documentación de uso y actividades. Las actividades de la metodología son las siguientes:

- Especificaciones del sistema: Allí sé planeta la definición de las funciones requeridas, así como las restricciones físicas, eléctricas y de índole económicas del sistema.
- Modelado del sistema: Requiere identificar las funciones necesarias del sistema a partir de las especificaciones establecidas.
- Elección de la arquitectura del sistema: Se propone la cantidad de componentes del sistema, el tipo de componente requerido y la manera en la que se interconectarán.
- Particionamiento de tareas y mapeo: Consiste en asignar funciones específicas a los diferentes componentes de HW y SW que hacen arquitectura.
- Planificación de tareas: Define el orden de ejecución y la prioridad de las tareas a realizar, como la manera en que se usa los recursos de hardware.
- Implementación del modelo: Se llevan a cabo las tareas de hardware utilizando dispositivos como FPGA, DSP o microcontroladores, mientras que las tareas de software se traducen en archivos binarios que serán ejecutados por procesadores. Se establece mecanismos de comunicación entre las tareas de hardware y software y comunicación entre subsistemas.
- Proceso de prototipado: Implica la creación del sistema físico teniendo presente los requerimientos funcionales como no funcionales.
- Pruebas del prototipo: Se evalúa el sistema físico creado para verificar si cumple con las especificaciones deseadas. En caso contrario, se deben realizar nuevas iteraciones en el proceso de diseño.

Para la implementación del prototipo se hará uso de herramientas de desarrollo tipo *copyleft* las cuales tienen la ventaja de permitir adaptaciones en hardware y software de otros proyectos y poder integrarlos de manera eficaz, disminuyendo tiempo de desarrollo, costos y fallos en el cumplimiento de los requerimientos funcionales, al resolver problemas por medio de framework orientados a propósitos específicos, en la figura **3-2** se puede ver el flujo de diseño de hardware que se puede alcanzar con estas herramientas.

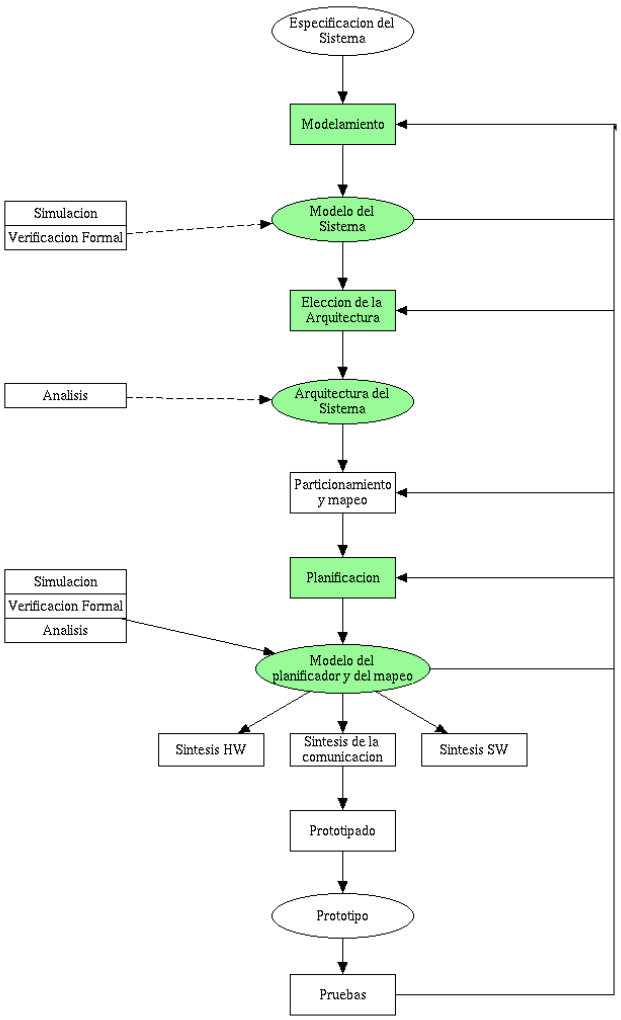


Figura 3-1: Flujo de diseño de un sistema digital. Fuente [2]

Requerimientos del sistema

El sistema a diseñar tiene los siguientes requerimientos de uso para el propósito pedagógico:

- Deberá facilitar el aprendizaje autónomo.
- El contenido documental se presentará en diferentes niveles de lectura para usuarios con diferentes modelos de comunicación o de conocimiento.
- Además de la programación de software, el sistema deberá permitir la programación de dispositivos provistos de sensores y actuadores para construir instrumentos musicales.
- El sistema debe contar con un IDE que facilite las diferentes herramientas requeridas en el proceso de creación de artefactos.
- El contenido documental deberá ser accesible de manera online como en modo offline.
- El IDE debe tener características de portabilidad para que sea posible acondicionarlo en otros sistemas.

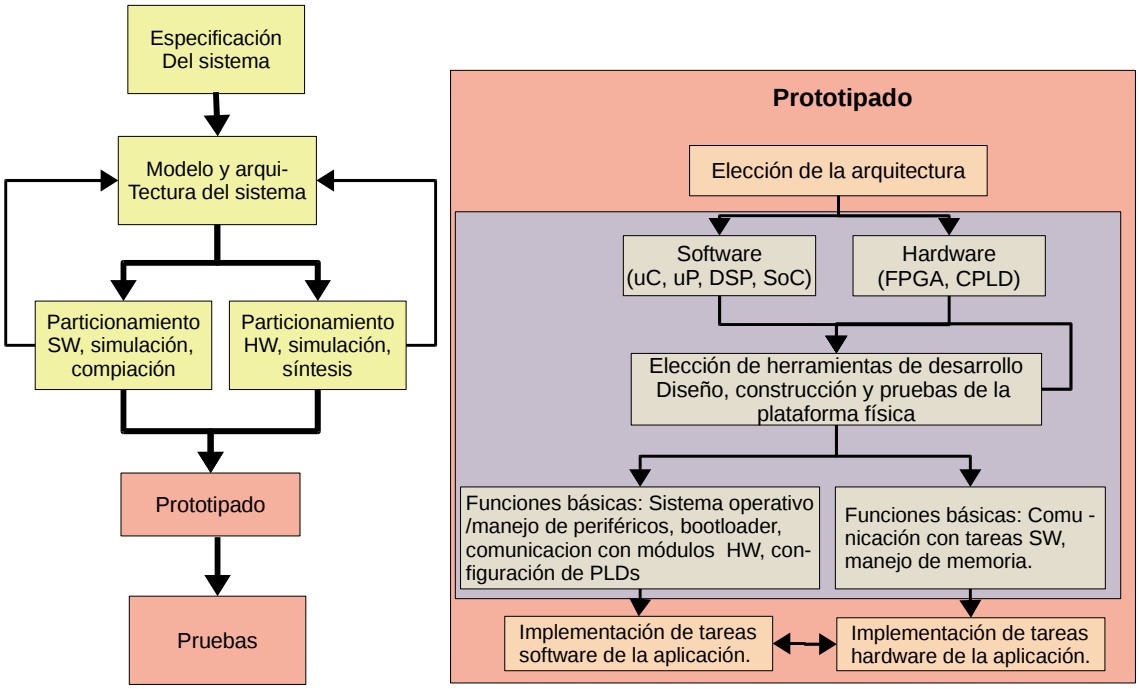


Figura 3-2: Flujo de diseño al utilizar hardware copyleft. Fuente [2]

- El material concreto deberá estar orientado a facilitar el *live coding music* el cual permite intervenir el producto final mientras este se ejecuta.
- En el mismo sentido de *live coding music* deberá permitir el desarrollo de orquestas LOLC en red donde pueda intervenir tanto los usuarios como las máquinas.

Los anteriores requerimientos listados permiten que las experiencias de aprendizaje tengan las siguientes posibilidades:

- Las experiencias se pueden diseñar en lugares en los cuales no se cuente con acceso a Internet.
- Se favorezca el desarrollo de instrumentos sonoros adaptando elementos del entorno.
- No solo desarrollar habilidades de programación de aplicaciones sino de cacharreo[1] con artefactos tangibles.
- El material permite el desarrollo de otras dinámicas de aprendizaje, por ejemplo, el estudiante puede compartir sus experiencias de aprendizaje autónomo a través de la exploración no guiada por un profesor.
- Este material concreto podrá convertir el lugar de aprendizaje en un laboratorio que permita al profesor y estudiante el desarrollo de actividades que antes no podía contemplarse.

En la figura **3-3** se muestra de manera gráfica las actividades que debe cumplir el sistema a diseñar, donde se aprecia la capacidad que debe tener para operar con diferentes dispositivos en red de manera síncrona para lograr el objetivo de *live coding music*.

[1] Consiste en la habilidad de interactuar con aparatos para resolver situaciones sin ser un experto en el área.

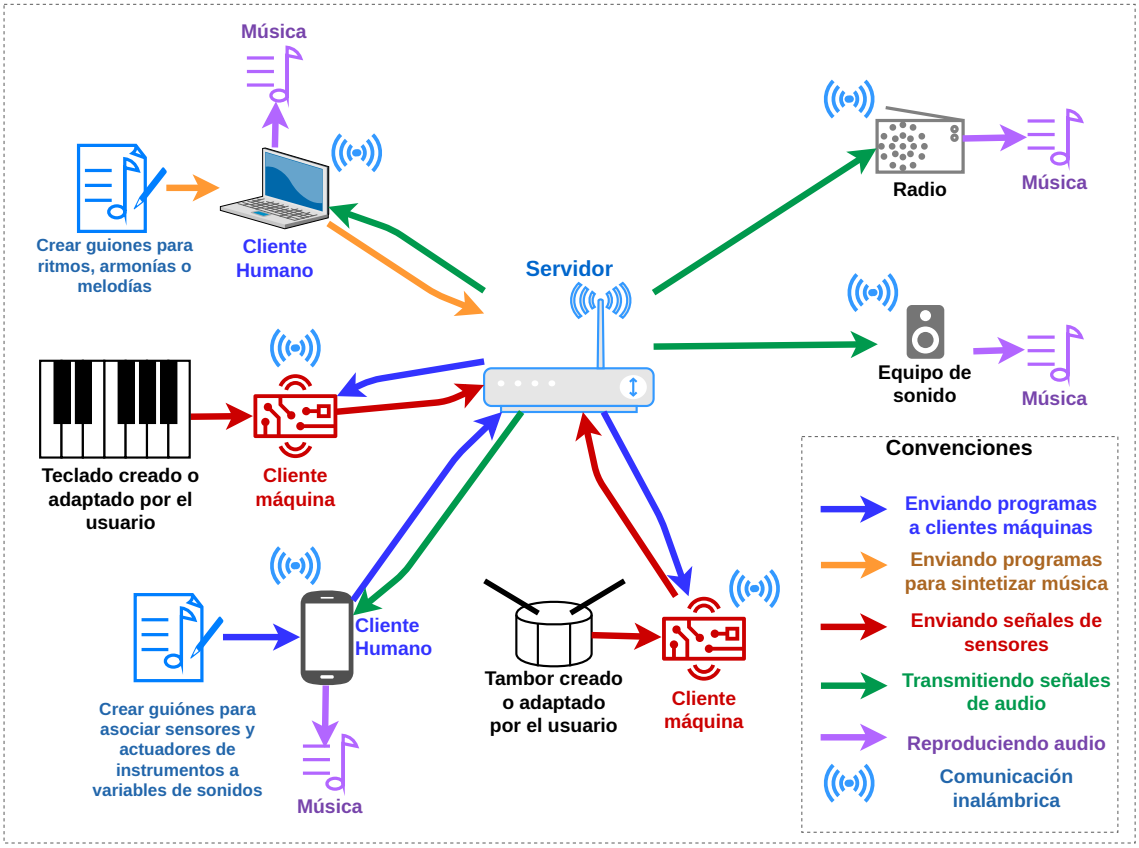


Figura 3-3: Representación del sistema general requerido

Modelamiento

Para poder cubrir los requerimientos planteados, se propone el desarrollo de un sistema que, de manera abstracta, contenga un módulo de interfaz de programación, un controlador de sensores y actuadores para el hardware y un sistema central de síntesis de sonido según se solicite ejecutar. En la figura **3-4** se aprecia que la interfaz de programación es accesible para distintos usuarios que permite comunicarse con la unidad central de síntesis enviando código, además, varios módulos de hardware pueden interactuar con el sistema central de síntesis de sonidos para monitorear y controlar variables, en conjunto, el sistema genera resultados sonoros que para el propósito se puede convertir en música.

Arquitectura

Para que la propuesta planteada en la figura **3-4** se pueda materializar, desde el punto de vista tangible, se propone tres módulos que se observan en la figura **3-5**; Se plantea el uso de un módulo servidor el cual gestionará las interfaces de comunicación como de audio, un módulo terminal que será la interfaz entre el usuario y el sistema y finalmente, una interfaz de instrumento. Tanto el servidor como la terminal pueden ser seleccionadas del mercado y dependiendo de los recursos con que se cuente en algún caso particular el servidor y la terminal pueden ser el mismo hardware, mientras que la interfaz del instrumento será creada como un sistema embebido digital que hará parte de un kit de materiales con sensores y actuadores.

Con respecto al software para implementar la propuesta, en la figura **3-6** se plantea que en el hardware se incorporen las tareas de software organizadas en tres módulos: Frontend, backend e instrumento. El frontend

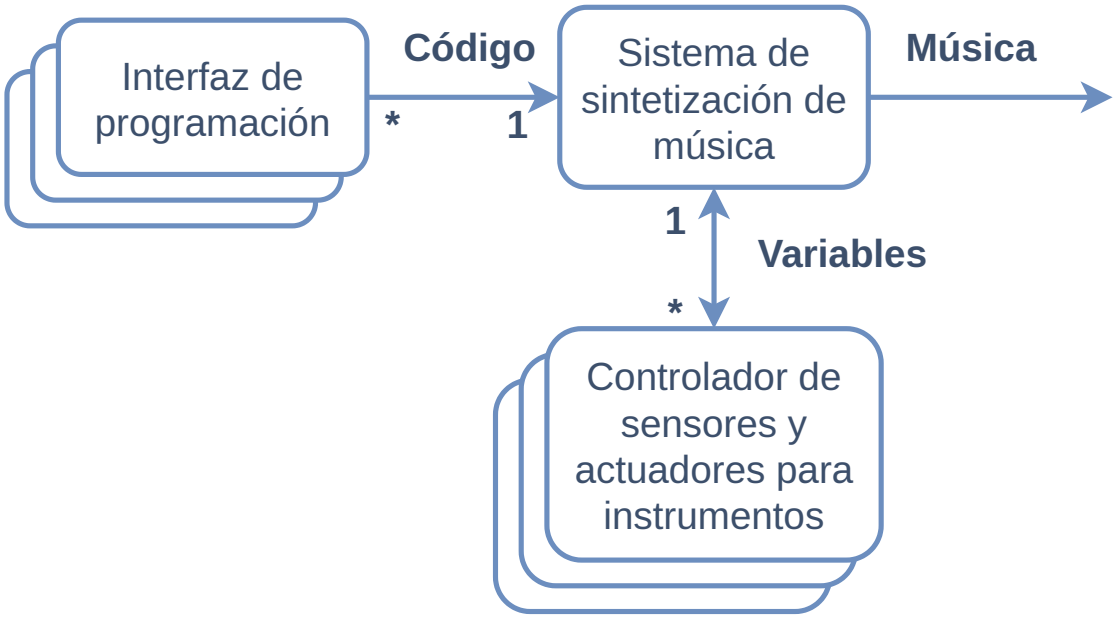


Figura 3-4: Representación abstracta del sistema de programación propuesto en el material concreto.

que deberá facilitar la experiencia de usuario, el backend contiene la lógica de los servicios de integración y un software embebido que permita el control del hardware de sensores y actuadores

Arquitectura del sistema y planificación

Con los requerimientos revisados, el modelo planteado y la arquitectura, se propuso la arquitectura del sistema que se muestra en la figura **3-7**, la cual, muestra los módulos de hardware con sus diferentes componentes en función del rol a cumplir, ya sea instrumento, el cual es planteado como un sistema embebido o módulo de servidor/terminal para el usuario final el cual puede ser otro sistema embebido, computador personal o dispositivo móvil. También se puede apreciar en esa imagen la necesidad de resolver criterios de selección de módulos ya sean de hardware o de software que se listan a continuación:

- Selección de un lenguaje de programación para la síntesis de sonidos.
- Selección de protocolos de comunicación para el envío de código para la programación y síntesis del sonido.
- Diseño de frontend
- Diseño de backend
- Selección de hardware para el servidor y la terminal
- Diseño de sistema embebido para el hardware de los instrumentos a crear.
- Diseño de la documentación del material didáctico.
- Presentación del kit enfocado a la experiencia de usuario.

El diseño de los módulos anteriormente listados serán desarrollados en los siguientes apartados.

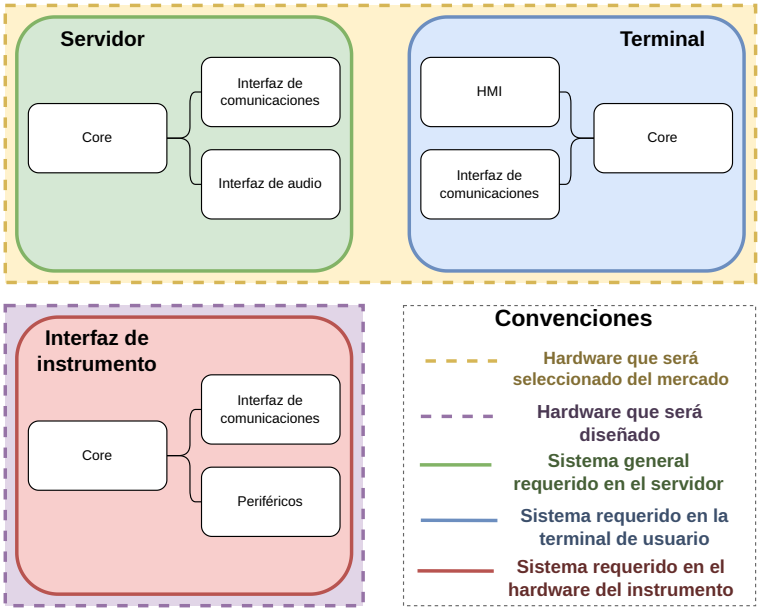


Figura 3-5: Módulos de hardware propuestos para el material concreto

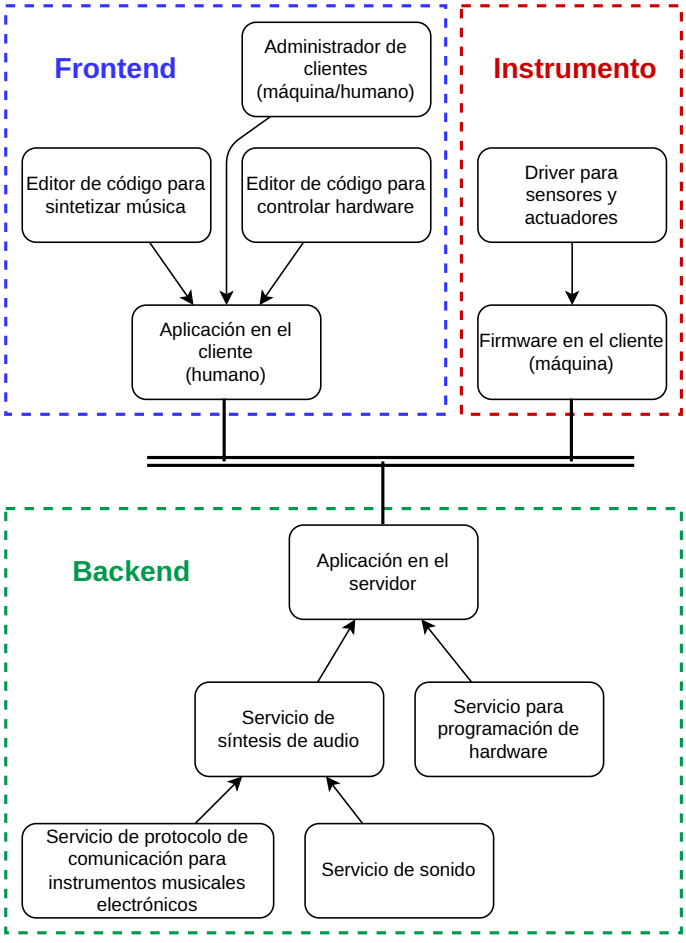


Figura 3-6: Módulos de software propuestos para el material concreto

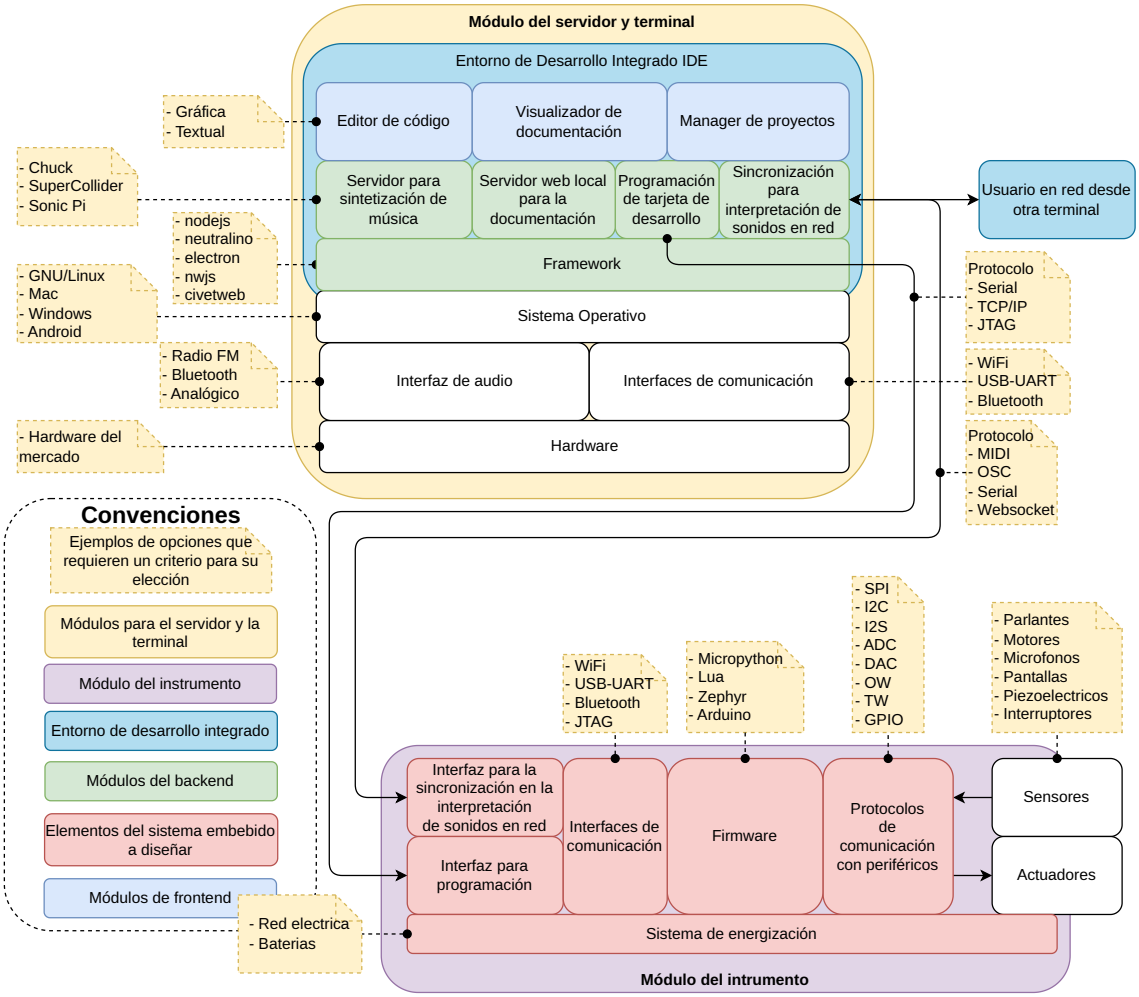


Figura 3-7: Sistema general a diseñar con los diferentes módulos los cuales requieren criterios para su selección

3.1 Selección del lenguaje de programación para síntesis de sonido

Con la aparición comercial del theremin en la década de los 30 del siglo XX, desarrollado por el científico ruso Lev Sergeivitch Termen, se sumaron al abanico de posibilidades la creación de nuevos sonidos, lo que generó nuevas propuestas en música y en cine, como fue el caso de la interpretación del sonido de un alienígena en la película *El día que la tierra se detuvo* del año 1951[2] o la interpretación del cisne del carnaval de los animales de Camille Saint-Saëns por Clara Rockmore[3] este instrumento tiene como base el control del volumen y la frecuencia de osciladores con sus formas de onda.

La tecnología del theremin está basada en la electrónica análoga que genera cambios en el valor de variables físicas de manera proporcional y continua a los eventos que lo estimulan. Con el desarrollo de la electrónica digital que tiene un fundamento discreto, surgieron otros dispositivos que generaban sonido y que se basan en osciladores diseñados por métodos digitales donde la salida de este proceso es conectado por cables aun conversor digital a analógico (DAC). Actualmente, este proceso está incorporado en el hardware de las computadoras de propósito general, como también en sistemas digitales de propósito específico, que con el

[2]Película de ciencia ficción dirigida por Robert Wise
[3]Clara Rockmore es conocida como la primera interprete profesional del Theremin

uso de herramientas de descripción de hardware o lenguajes de alto nivel para compilar programas se pueden generar sonidos como en un principio se hacía con el theremin.

A continuación se realiza una revisión de las herramientas disponibles para los generadores de sonido a través de osciladores con los lenguajes de programación y se realizará una selección del framework para incorporarlo en el backend del IDE a desarrollar.

3.1.1 Software para crear música mediante algoritmia

La síntesis de música requiere un generador de unidades para el procesamiento de señales de audio, tanto en los lenguajes de música por computadora como en los sintetizadores de hardware. Esta unidad constituye la base para generar osciladores que permitan modificar la frecuencia, el volumen y otras características del sonido. En cuanto a los lenguajes de programación utilizados, se pueden clasificar en dos categorías para proporcionar un contexto general: los lenguajes para música, que deben incluir declaraciones de modismos sincrónicos para manejar flujos temporales, y los lenguajes síncronos [22].

La Tabla **3-1** muestra una recopilación de algunos lenguajes de programación según esta clasificación.

Tabla 3-1: Lenguajes de programación preparados o adaptados para la creación de música.

Lenguajes para música	**Lenguajes síncronos**
Lenguajes textuales: Familia de lenguajes Music-N (I, II, III, IV, V), Csound, SAOL, Faust, Nyquist, SuperCollider, ChucK, Impromptu. **Lenguajes visuales**: Max/MSP, Pure Data, jMax, Open Sound World. **Sistemas de modelamiento físico**: Modalys, Chant, Genesis/Cordis-Anima. **Misceláneos**: Kyma (entorno gráfico de diseño de sonido), STK (kit de herramientas C++), CLAM, SndOb.	**Lenguajes textuales**: Esterel, Lustre, Signal, Concurrent ML, Larissa, Lucid Synchrone, Prelude, Quartz, Reactive ML, RMPL, SL, SOL, StreamIt, 8 1/2 **Lenguajes visuales**: Argos, Statecharts, SyncCharts, Argonaute, Polis, Polychrony, Scade, Simulink/Matlab. **Extensiones de lenguajes**: ECL (C), Jester (Java), Reactive-C (C), Real-time Concurrent C (C), RTC++ (C++), Scoop (Eiffel), SugarCubes (Java). **Lenguajes de descripción de hardware**: Lava, SystemC, Verilog, VHDL. **Modelos y formatos intermedios**: Averest, DC+, OC, SC, DC, CP, SDL, ULM, UML Marte.

Según [22], se evaluaron cinco lenguajes para música y cinco lenguajes síncronos para implementar un oscilador simple, obteniéndose conclusiones sobre las características generales de cada clasificación, como se muestra en la Tabla **3-2**.

Tabla 3-2: Resumen comparativo de conceptos de diseño.

Característica	**Lenguaje para música**	**Lenguaje síncrono**
Tamaño del Código	Mucho más corto	Extenso
Tiempo	Principalmente una noción de hardware	Principalmente una noción lógica
Señales	Mapeos simples de ticks	Relojes abstractos y complejos
Capas	De bajo nivel e interactivo de alto nivel	Capa síncrona y GALS

3.1.2 Frameworks para la programación de música

Existen varios entornos de programación diseñados para la síntesis de música, especialmente adaptados para el *live coding*. Entre ellos se destacan miniAudicle y Sonic Pi a nivel de programación textual, ambos equipados con protocolos de comunicación para conectar software con hardware musical. Por otro lado, está Scratch configurado para *live coding music*, que utiliza una programación por bloques (visual), reduciendo así los errores de sintaxis, lo que lo hace adecuado para principiantes. Este también puede adaptarse para conectar hardware musical, como se muestra en un caso específico.

En cuanto a hardware, el producto comercial *Speak & Spell* de Mattel es notable, ya que, mediante una API de código abierto, permite al usuario reconfigurar el hardware y convertirlo en un instrumento musical de sonidos alienígenas utilizando el protocolo OSC (Open Sound Control). Otra opción es el *SynthKit* con Little Bits, que permite crear un sintetizador musical interconectando módulos con componentes analógicos (programación tangible), aunque en este caso solo se realizan tareas de hardware.

En lo que respecta a librerías para control de hardware, se destaca *KappyBrackets*, una librería de código abierto diseñada para Raspberry Pi, que facilita la codificación creativa para configurar, programar y controlar múltiples dispositivos remotos.

MiniAudicle y Sonic Pi

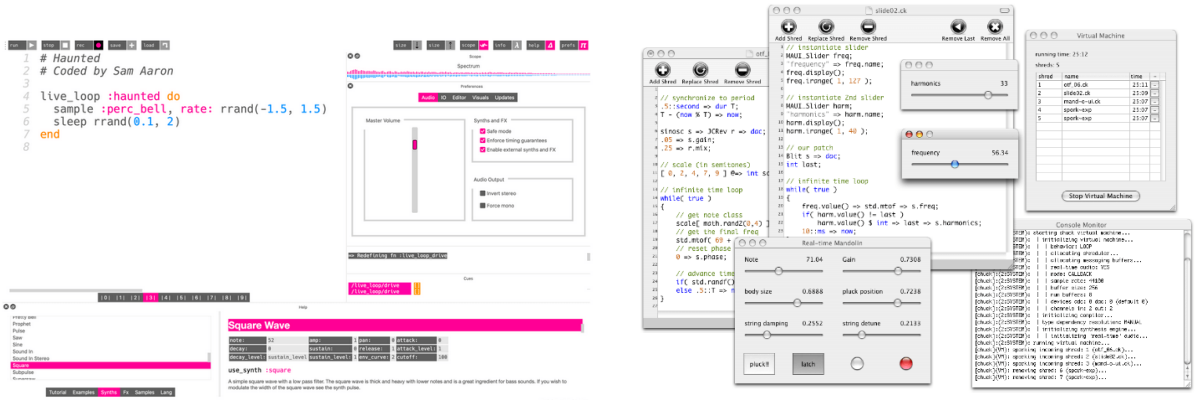
MiniAudicle es un entorno de desarrollo integrado de código abierto para *ChucK* [23], diseñado para sintetizar sonidos que se convierten en música (ver Figura **3-8**(b)).[4] El lenguaje de programación utilizado es *ChucK*, el cual actúa como servidor para la síntesis.

Sonic Pi, por otro lado, es un entorno de software de código abierto que ha ganado popularidad en los últimos años, diseñado inicialmente para Raspberry Pi y destinado a proporcionar una calidad adecuada en la síntesis de sonido (ver Figura **3-8**(a)).[5] Requiere un puerto de operación del servidor *SuperCollider* (SC) y utiliza una variante del lenguaje de programación *Ruby* en su sintaxis [26].

Tanto *Sonic Pi* como *ChucK* están equipados para conectarse con hardware musical a través de protocolos MIDI u OSC (Open Sound Control), permitiendo así una experiencia de "networked sound". En la Tabla **3-3** se presenta un resumen de algunas características para compararlos.

[4]La imagen de *MiniAudicle* y la interfaz de *ChucK* fue obtenida de [24]. Cabe destacar que *MiniAudicle* también está disponible para iPad [25].

[5]La imagen del entorno de *Sonic Pi* fue obtenida del https://sonic-pi.net/tutorial.



(a) (b)

Figura 3-8: (a) Sonic Pi IDE, (b) MiniAudicle.

Tabla 3-3: Comparación entre MiniAudicle y Sonic-Pi.

Característica	**MiniAudicle**	**Sonic Pi**
Lenguaje utilizado	Chuck (GPL)	Ruby -> DSL
Servidor para síntesis musical	Chuck (GPL)	SuperCollider
Interfaz	Shell	IDE con visualización de señales
Plataforma	Multiplataforma	Multiplataforma
Programación	Textual	Textual
Desarrollo de Interfaz	C++ y Qt4	C++ y Qt5

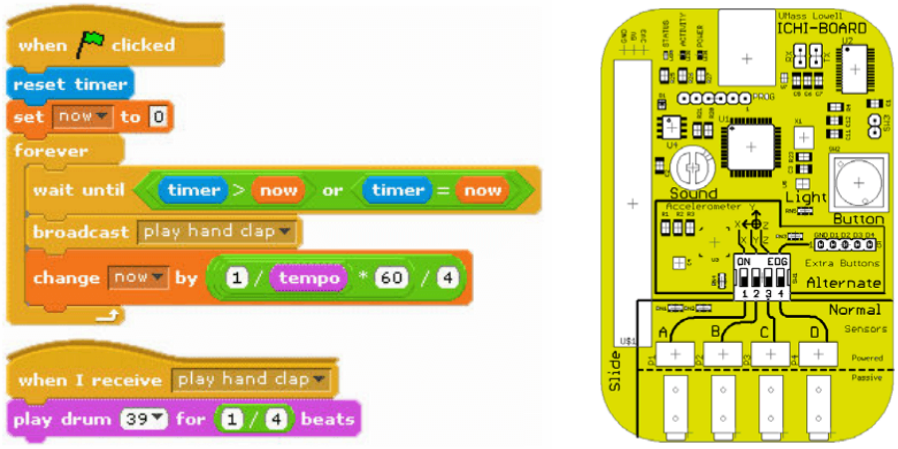
Scratch para live coding music

Scratch es un entorno de programación que emplea bloques para la construcción de scripts. El usuario ensambla estos bloques como si fueran piezas de rompecabezas para crear su guion. A través de una adaptación, se logró la conexión entre la placa *ichiBoard* y Scratch para posibilitar el Live Coding Music, utilizando hardware que permitiera la configuración de instrumentos electrónicos. La Figura **3-9** muestra un ejemplo de código para reproducir una batería (parte (a)) y el esquema de la placa *ichiBoard* (parte(b)), que incorpora hardware para música y sensores para controlar el "instrumento" [27].

Speak & Spell y SynthKit

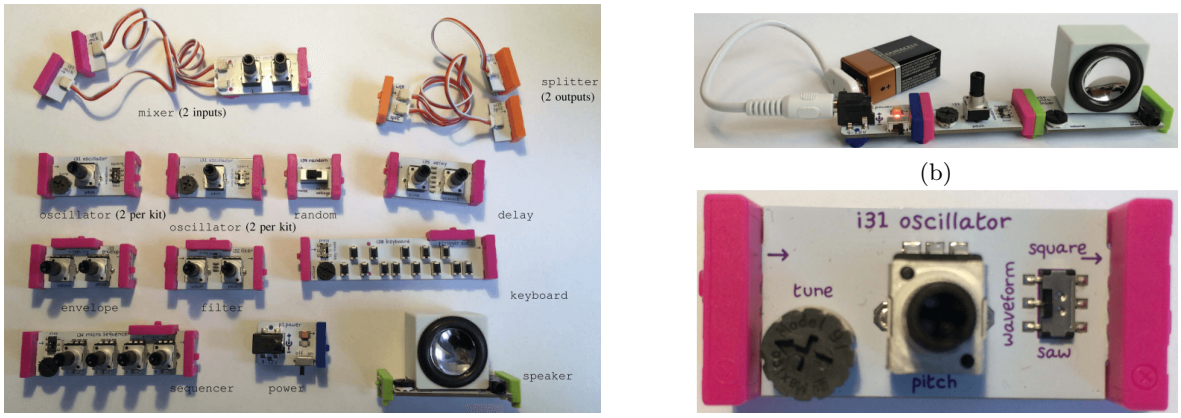
En el ámbito del hardware, existen plataformas diseñadas para la síntesis de audio, como por ejemplo *SynthKit* (ver Figura **3-10**). Este es un conjunto de componentes analógicos de código abierto preensamblados, que se pueden armar, desmontar y reutilizar con facilidad [28]. Con *SynthKit*, los usuarios pueden crear una amplia variedad de circuitos y dispositivos sin necesidad de programación, cableado ni soldadura, e incluso pueden conectarse a un módulo en la nube para acceso a Internet. Los módulos están codificados por colores para indicar su función: azul para alimentación, amarillo para conectores, rosado para entradas y verde para salidas. El flujo de ensamblaje se realiza de izquierda a derecha. En este contexto, los ajustes en la síntesis pueden realizarse en tiempo real manipulando los componentes de los osciladores, filtros y otros mecanismos analógicos de los módulos.

Otro ejemplo de hardware es el ***Speak*** & ***Spell*** de Mattel (ver Figura **3-11**), que con la adición de algunos



(a) (b)

Figura 3-9: (a) Ejemplo de código para "Live Coding Music" en *Scratch*. (b) Placa *ichiBoard* utilizada en "Live Coding Music" con *Scratch*.



(a) (b) (c)

Figura 3-10: Synth Kit con Little Bits. (a) Los diferentes módulos de SynthKit. (b) Módulo energizado del Kit con Little Bits (fuente, oscilador y parlante). (c) Modulo Little Bits oscilador.

botones y potenciómetros puede transformarse en un instrumento musical de sonidos alienígenas. En este caso, se deben exponer los puntos de contacto, ya que la API utilizada es de código abierto, y mediante un protocolo de comunicación como OSC (Open Sound Control), se puede experimentar con las asignaciones para obtener diferentes salidas para el instrumento musical.

(a)

(b)

Figura 3-11: (a) Kit Speak & Spell. (b) Speak & Spell ejemplo de uso.

Biblioteca HappyBrackets para Raspberry Pi

HappyBrackets es una biblioteca diseñada para gestionar hardware equipado con una variedad de sensores especialmente destinados para la música, conectados a través del IOT. Este proyecto se ha creado con el propósito de facilitar la codificación creativa en múltiples dispositivos remotos, incluyendo Raspberry Pi [29]. No obstante, al ser una biblioteca, su utilización puede estar limitada a personas con experiencia previa en desarrollo y programación.

> HappyBrackets proporciona una práctica biblioteca de sensores que automáticamente detecta varios dispositivos como acelerómetros, giroscopios, sensores de temperatura y magnetómetros. Además, los usuarios tienen la capacidad de escribir sus propios controladores de sensores o dispositivos de salida mediante el acceso al GPIO, sin la necesidad de desarrollar módulos específicos del kernel o modificar archivos subyacentes de Java de HappyBrackets. Uno de los autores actualmente utiliza este enfoque para interactuar con codificadores rotativos de efecto Hall en una instalación que emplea patines de ruedas musicales robóticos. [29]

3.1.3 Laptop Orchestra "Live Coding" (LOLC)

Los principales retos para esta puesta en escena son la sincronización y el uso de material compartido entre los miembros del ensamble ("orchestra"). Para este tipo de ensambles los entornos de interpretación textual tienen una ventaja en eficiencia y flexibilidad, aunque se limiten en la cantidad de músicos. Colaboración basada en conjuntos [30]

Aunque está inspirado en los sistemas de codificación en vivo, LOLC no es en sí mismo un lenguaje de programación: no es Turing completo ni permite a sus usuarios definir nuevos procesos computacionales.

En cambio, su diseño, en el que los músicos crean patrones rítmicos basados en archivos de sonido, comparten y transforman esos patrones a través de una red local, facilitando un paradigma de interacción inspirado en otras formas de improvisación de conjuntos, particularmente en el jazz y la música de vanguardia.

3.1.4 Selección del lenguaje de programación

De la revisión anterior, el lenguaje seleccionado para la integración en el IDE es ChucK: [6], debido a que ChucK está escrito en C, está diseñado para ser usado a modo de librería o aplicación, su desarrollo está separado de su interfaz gráfica y tiene implementado los servicios para LOLC a través de OSC. Aunque Sonic Pi tiene esos mismos servicios, en principio fue diseñado para la placa de desarrollo Raspberry Pi, el footprint de este es del orden de los cientos de megas (ChucK a penas una decena de megas) y aunque la interfaz gráfica de Sonic Pi está diseñada para favorecer la experiencia de usuario, esta no será necesaria, pues la interfaz de programación estará embebida en el IDE a diseñar, así que será una característica que el usuario final no podrá aprovechar.

En resumen ChucK tiene las siguientes características aprovechables para el diseño del IDE.

- ChucK es portable a otros sistemas operativos debido a que hace uso de las herramientas de compilación de GNU
- El footprint de la aplicación en el backend está al rededor de 3MB.
- Cuenta con los servicios OSC que favorecen el LOLC como la comunicación M2M.
- Cuenta con una sintaxis similar a la de C para la creación de sonido.

3.2 Diseño del kit de materiales

Teniendo presente los objetivos a alcanzar en esta metodología descrita en el capítulo 2, además, de tener una experiencia en programación de computadores, se busca que el estudiante descubra su entorno como una fuente de recursos e inspiración para los posibles artefactos a realizar, para lograr este objetivo, se debe pensar en un material concreto que sirva de interfaz entre lo que desea programar el estudiante y los diferentes recursos. En la figura **3-12** se muestra el proceso de creación de artefactos que pueden funcionar como instrumentos musicales; el estudiante cumple un rol de compositor de guiones, con ayuda de los materiales del kit adapta un artefacto y con el guion que ha construido al ser ejecutado por el material concreto, el artefacto se comporta como si se tratara de un tambor, un piano, una flauta o cualquier otro instrumento que no necesariamente esté estandarizado.

Para desarrollar el prototipo del material concreto, se propone los siguientes criterios:

- El material concreto se debe concebir como una **caja de materiales incompleta** de tal manera que el estudiante se vea involucrado en su adaptación, seleccionando los demás materiales que deben hacer parte de las herramientas que a su criterio, debe contener.
- Los elementos a integrar como material concreto deben ser seleccionados con el criterio de ser aquellos que no sean fáciles de encontrar en el entorno por parte del estudiante.
- Los materiales deben ser sustituibles en el caso de pérdida o daño, económicos y preferiblemente de la línea educativa.
- Cada elemento a agregar debe facilitar la creación de sistemas de entradas y salidas de información.
- No debe representar un peligro para el usuario final.

[6] Sitio oficial de ChucK: `https://chuck.stanford.edu/`, código fuente: `https://github.com/ccrma/chuck`

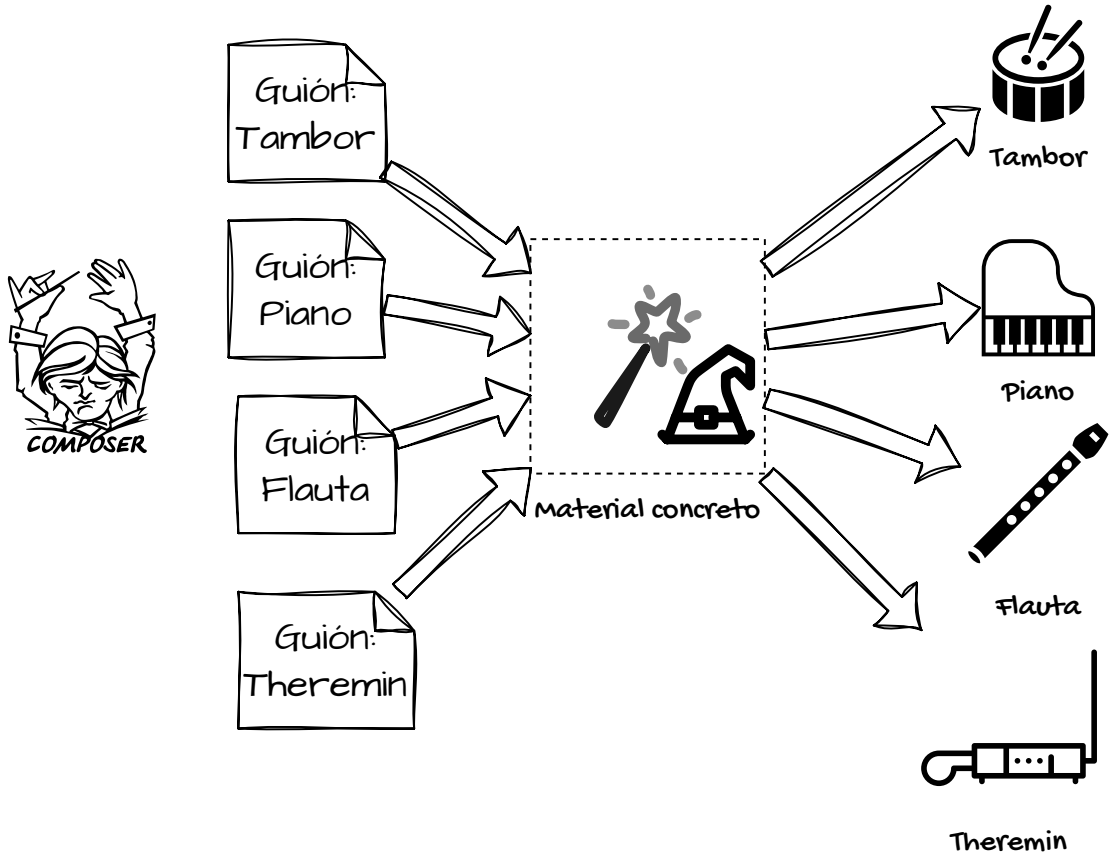


Figura 3-12: Representación abstracta del kit de materiales adaptado por parte del estudiante para que se comporte como uno o varios instrumentos musicales dentro de una orquesta.[7]

Teniendo en cuenta los criterios y además el propósito a cumplir, se plantea que para la construcción del material concreto se debe comprender cómo se espera que interactúe el material con el entorno en vista de la información, variables relevantes, recursos y energía.

Desde el punto de vista de los sistemas, los cuales requieren para su conformación de recursos, energía e información, podemos ver en la figura **3-13** cómo interactúa los diferentes módulos que son responsables de los propósitos de monitoreo y control del estado de variables. En esa misma figura se observa que desde el entorno hay diferentes actores de los cuales se puede tener información, como es el caso de máquinas, animales, variables ambientales, de humanos e inclusive de artefactos que se encuentran fuera de la tierra. Para obtener la información y transportarla, se requiere de transductores, que en el caso de la figura, traduce la información en señales eléctricas con información de tipo analógica o digital; este transductor se conoce como sensor. Esta información en el flujo mostrado en la figura, es entregada al controlador el cual puede estar basada en tecnología analógica o digital para que pueda responder o tomar una decisión según unos parámetros, enviado señales eléctricas digitales o analógicas a una máquina transductora la cual convertirá las señales eléctricas en otras manifestaciones físicas, como es el caso de sonido, calor, movimiento, entre otras. A esta máquina transductora es lo que se le denomina actuador.

Además de lo mencionado, estos sistemas pueden contar con interfaces máquina-humano y máquina-máquina (M2M) que permiten compartir información en protocolos establecidos, con la finalidad de cambiar criterios o verificar el correcto funcionamiento, esta comunicación puede ser guiada o no guiada. Finalmente para que

[7]Las imágenes creadas para esta figura son obtenidas del sitio `https://www.svgrepo.com/page/licensing` las cuales son de licencia copyleft.

esto sea posible se requiere definir el método de alimentación de energía que permita realizar estos procesos.

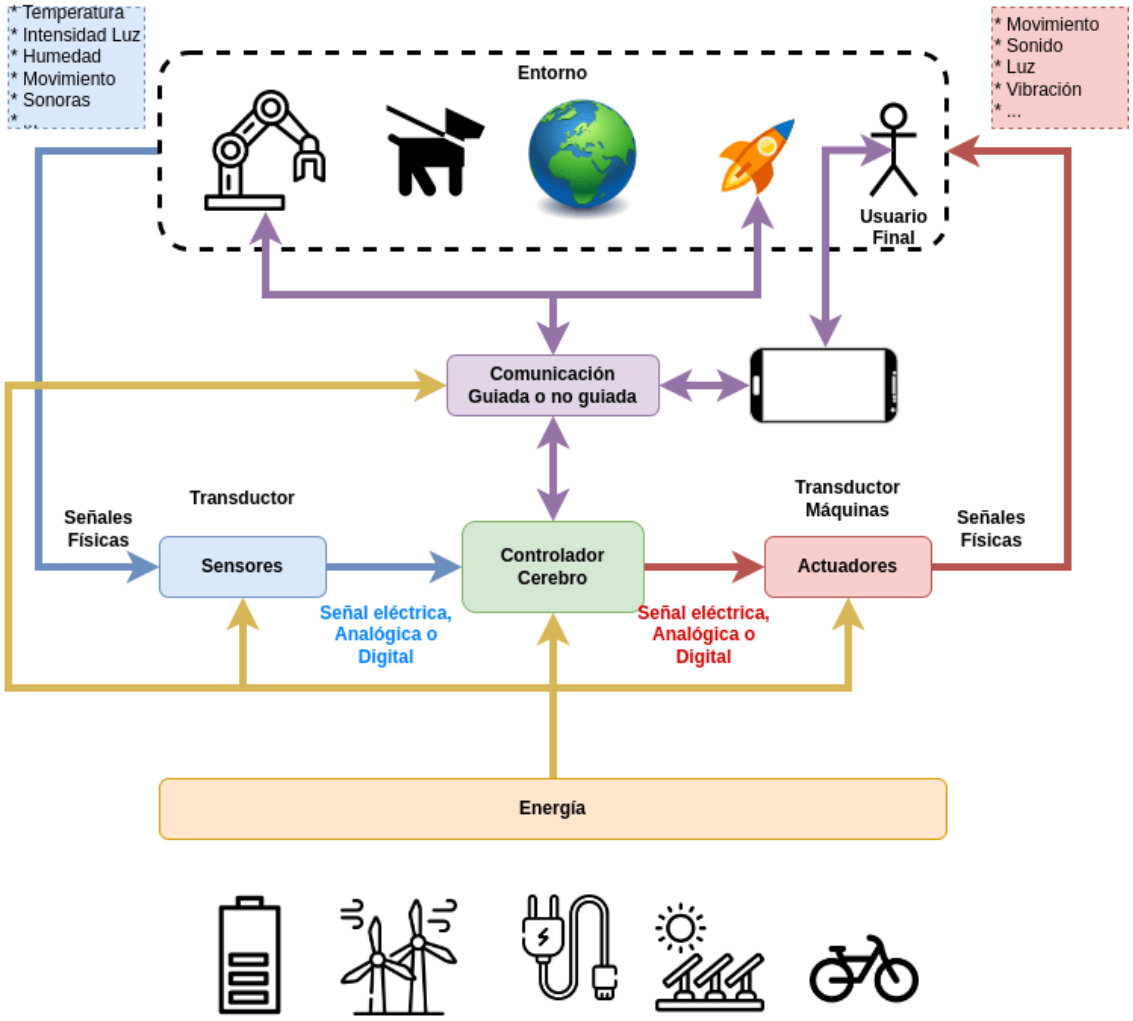


Figura 3-13: Caja negra y relación con el monitoreo y control de variables del entorno

Para el desarrollo del material didáctico es importante realizar la analogía de la figura **3-14**, debido al enfoque motivacional desde la música que tiene la propuesta. En esta figura, se manifiesta que los sensores permiten sentir del entorno, mientras que los actuadores permiten expresar al entorno aquello que es de interés para el usuario final haciendo del material concreto una extensión de su cuerpo.

En resumen, para el desarrollo del material concreto se requiere definir lo siguiente:

- Sensores
- Actuadores
- Controlador
- Conectores
- Energía
- Interfaz de comunicación con otros sistemas

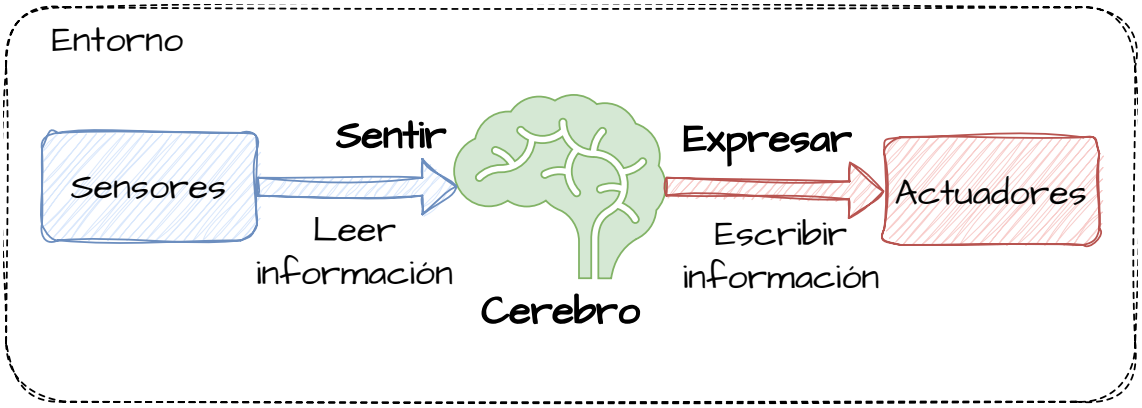


Figura 3-14: Representación de las funcionalidades que tiene los sensores y actuadores desde el enfoque motivacional de la música en esta propuesta.

Con respecto al controlador e interfaz de comunicación, el diseño es abordado en el apartado de diseño de tarjeta de desarrollo 3.3.

Selección de conectores y suministro de energía

Para el tipo de artefactos a construir con el material concreto, se requiere que haya autonomía eléctrica, principalmente para comunicaciones inalámbricas y/o para embeber estos componentes en instrumentos creados o adaptados por el usuario, además, los niveles de tensión a usar deben ser seguros. Debido a que en el mercado se encuentran dispositivos electrónicos de prototipado rápido de la línea educativa entre 3.3V y 5V el material didáctico a diseñar, trabajará al nivel de tensión de 5V, el cual es seguro y es compatible con el nivel de tensión USB. El kit de materiales contará con una batería recargable que le ofrecerá la autonomía requerida según sea el propósito. Los anteriores criterios serán integrados en la tarjeta de desarrollo a diseñar.

Selección de sensores y actuadores

Este item dentro del material concreto tiene una importante relevancia, pues desde esta selección, el usuario podrá crear diferentes artefactos con opciones de uso. El criterio de selección ha sido diversificar las posibilidades en variables medibles y controlables sin importar un objetivo en concreto, acompañando a este material de indicaciones de uso para que desde un ejercicio creativo, el usuario pueda imaginar cómo podrá integrar el dispositivo a su propia iniciativa. En el **anexo E** podrá conocer la lista de sensores y actuadores seleccionados con las indicaciones del propósito. El método de uso es anexado en la documentación del material concreto.

3.3 Diseño de tarjeta de desarrollo

La tarjeta de desarrollo en la figura **3-7** cumple el rol de módulo de instrumento, el cual, debe tener la capacidad de controlar sensores, actuadores, comunicarse con el módulo de servidor-terminal y permitir la reprogramación de tareas en él para la adaptación del instrumento. Este módulo será diseñado a modo de *Sistema Embebido*, el cual se diseña con un propósito específico, donde el procesador es encapsulado por el

sistema que él controla [2]. En la figura **3-15** se observa la arquitectura de un sistema embebido, el cual es un sistema heterogéneo donde interactúan módulos de hardware y de software para realizar las distintas tareas requeridas.

Para la implementación del sistema embebido se cuenta con tres posibles diseños heterogéneos: Una implementación basada en SoC y dispositivos lógicos programables, basada exclusivamente en FPGA y una basada en un SoC exclusivamente [2]. Para la implementación de esta tarjeta de desarrollo se decide la implementación en SoC, en vista que no hay restricciones temporales importantes, el usuario final no es un experto y no se requiere la reconfiguración de tareas de hardware, ya que para el propósito del material concreto se podrían emular a través del software.

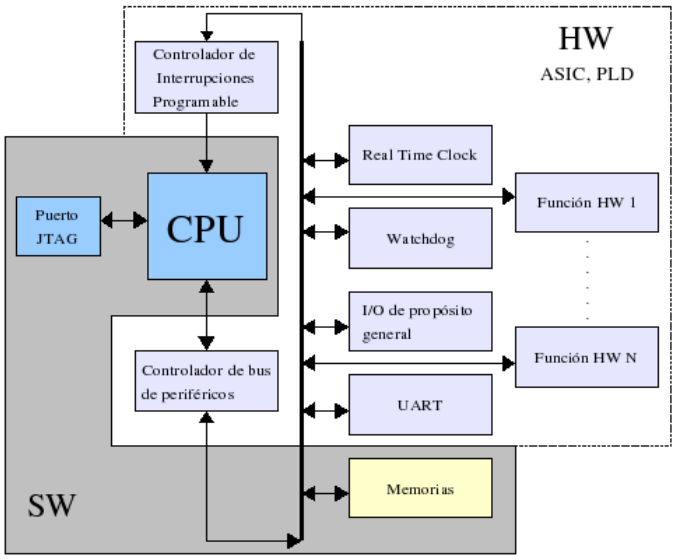


Figura 3-15: Arquitectura de un Sistema Embebido. Fuente [2]

Para la selección del SoC se realiza la tabla **3-4** teniendo en cuenta el costo del SoC, módulos de HW, comunicaciones, memoria principal, arquitectura del procesador, lenguajes de programación y frameworks diseñados con herramientas *copyleft*.

Tabla 3-4: Comparativa de SoC candidatos para el diseño de la tarjeta de desarrollo

Característica	**ATmega328P**	**RP2040**	**ESP8266**	**ESP32**
Memoria RAM	2 KB	264 KB	160 KB	520 KB
Periféricos	UART, SPI, I2C, ADC	UART, SPI, I2C, ADC	UART, SPI, I2C, ADC, PWM, WiFi	UART, SPI, I2C, ADC, PWM, Bluetooth, WiFi
Framework	Arduino	Arduino, MicroPython	Arduino, MicroPython, NodeMCU	Arduino, MicroPython, ESP-IDF
Costo	Bajo	Moderado	Bajo-Moderado	Moderado-Alto
Arquitectura	AVR	ARM Cortex-M0+	Tensilica Xtensa LX106	Tensilica Xtensa LX6
Lenguajes de Programación	C/C++, Arduino	C/C++, Arduino, MicroPython	C/C++, Arduino, Lua, MicroPython	C/C++, Arduino, MicroPython

Realizando una revisión del costo-beneficio de la tabla **3-4** se ha seleccionado el esp8266, debido a su bajo costo, integración de antena para comunicación inalámbrica, capacidad de procesamiento y empaquetado esp8266-e12 que facilita la integración y reemplazo en la PCB. En la tabla **3-5** y la figura **3-16** se observa los módulos de hardware y las características del SoC las cuales fueron propuestas en el diagrama general de la figura **3-7**.

Seleccionado el SoC teniendo en cuenta la metodología de diseño de sistemas embebidos se describe a continuación el diseño del hardware y del software embebido.

Categories	Items	Parameters
Wi-Fi	Standard	FCC/CE/TELEC/SRRC
	Protocols	802.11 b/g/n/e/i
	Frequency Range	2.4 G ~ 2.5 G (2400 M ~ 2483.5 M)
	Tx power	802.11 b: + 20 dBm
		802.11 g: + 17 dBm
		802.11 n: + 14 dBm
	Rx Sensitivity	802.11 b: -91 dBm (11 Mbps)
		802.11 g: -75 dBm (54 Mbps)
		802.11 n: -72 dBm (MCS7)
	Antenna	on-board, external, IPEX connector, ceramic chip
Hardware	Peripheral interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/PWM
	Operating voltage	2.5 V ~ 3.6 V
	Operating current	Average: 80 mA
	Operating temperature range	-40 ℃ ~ 125 ℃
	Storage temperature range	-40 ℃ ~ 125 ℃
Categories	Items	Parameters
	Package size	QFN32-pin (5 mm x 5 mm)
	External interface	N/A
Software	Wi-Fi mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware upgrade	UART Download/OTA (via network)
	Software development	SDK for customized development/cloud server development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User configuration	AT Instruction Set, Cloud Server, Android/ iOS app

Tabla 3-5: Especificaciones del SoC esp8266. Fuente [3]

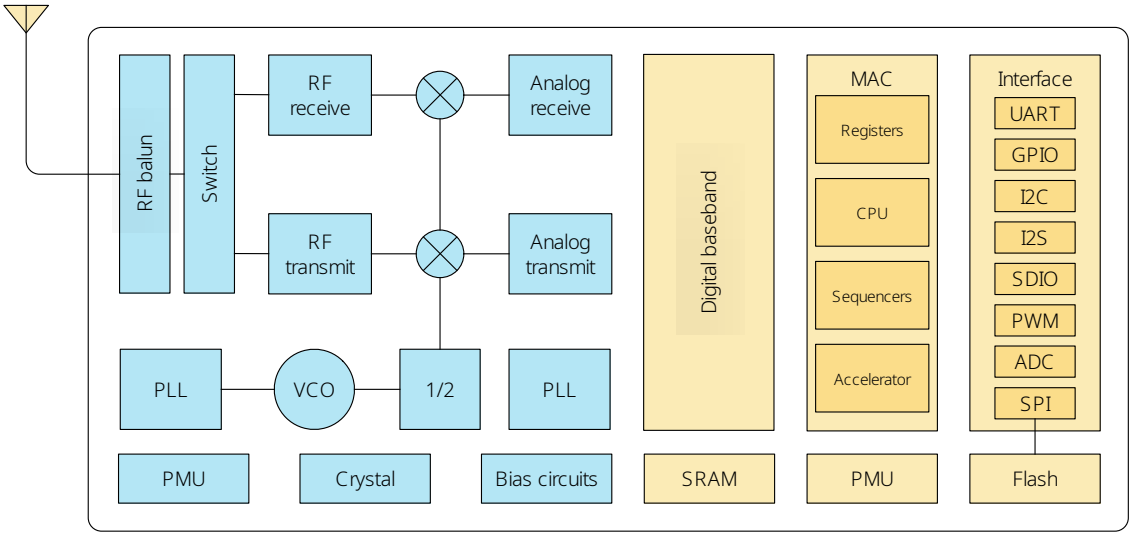


Figura 3-16: Diagrama de bloques del esp8266x. Fuente [3]

3.3.1 Diseño de hardware embebido

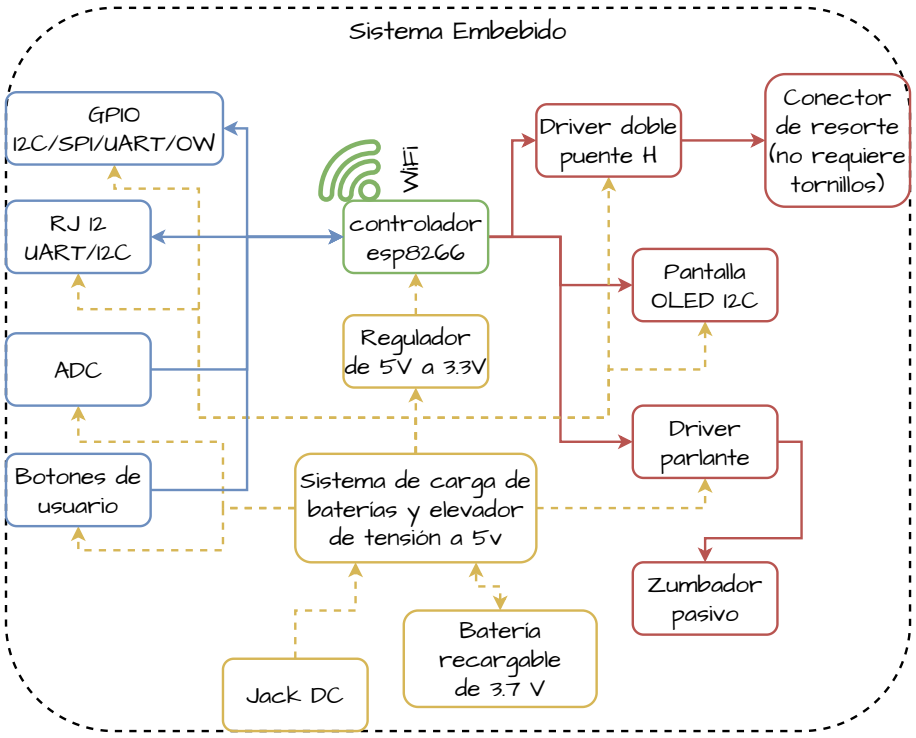


Figura 3-17: Diagrama de bloques de los módulos que serán requeridos en el diseño del sistema embebido

El proceso de diseño y fabricación de PCB tiene un nivel de complejidad importante, requiere una revisión detallada de los requerimientos según el propósito, la tecnología disponible para el proceso de fabricación, los materiales disponibles, entre otros. Entre las necesidades a resolver para la creación de la PCB se encuentran:

- Criterios para la ubicación espacial de los componentes seleccionados en la PCB.
- Técnicas para protección y seguridad según propósito.
- Tecnología de los componentes electrónicos.
- Vida útil del sistema.
- Tipo de conectores para la interfaz de la PCB.
- Capacidad y tipo de sistema de energía de la placa electrónica.

Para el diseño del hardware embebido además de atender los requerimientos funcionales, se debe tener presente los objetivos pedagógicos, donde el usuario final no es un experto, en resumen, se requiere que el hardware esté enfocado a facilitar la experiencia y exploración del material concreto y que a su vez, no eleve los costos de producirlo. Las características y enfoques del hardware diseñado serán mostrados al final de este apartado.

Los bloques funcionales que requiere la placa de desarrollo a diseñar se pueden observa en la figura **3-17**, en ella, de color azul se encuentran los módulos que son entradas y salidas de información, los cuales están conectados al SoC esp8266, los módulos actuadores están enmarcados en color rojo y finalmente en amarillo, se puede apreciar el modelo de alimentación de energía, el cual, permitirá la autonomía para proyectos inalámbricos.

Para la creación del sistema embebido, se hace uso de la metodología de diseño de PCB sugerida que se observa en la figura **3-18**, la cual, permite la evaluación e iteración de prototipos, además, se requiere la revisión y aplicación de normas IPC de diseño de PCB [31].

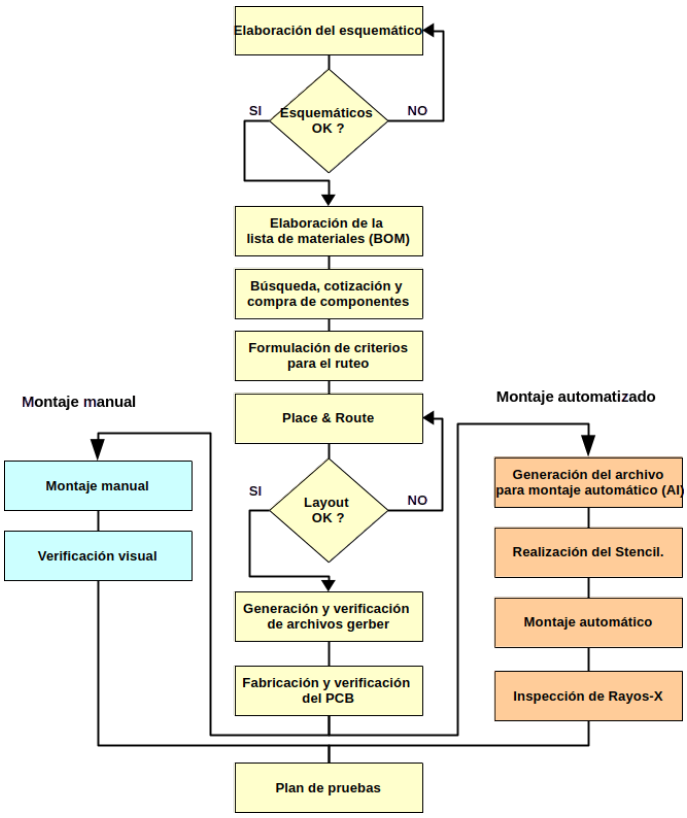
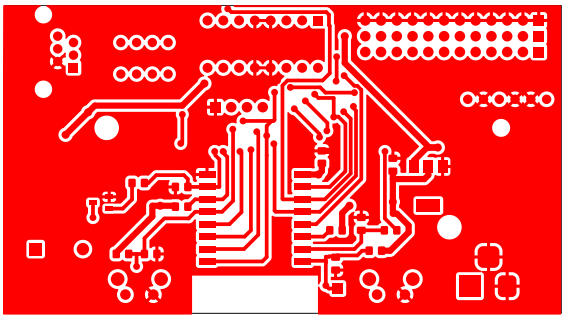


Figura 3-18: Proceso de fabricación de PCB. Fuente [2]

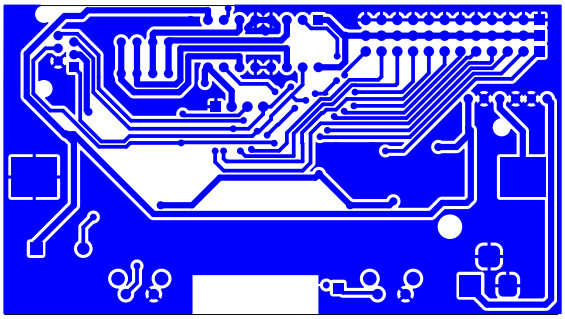
Diseño de esquemático, ruteo y generación de archivos Gerber

Para el proceso de creación del esquemático, *place and route* y exportación de archivos gerber se hace uso del software de diseño asistido por computadora (CAD) llamado KiCAD[8], el cual tiene licencia GPLv3, compatible con copyleft el cual permite reutilizar diseños, footprints y módulos para editarlos según necesidades.

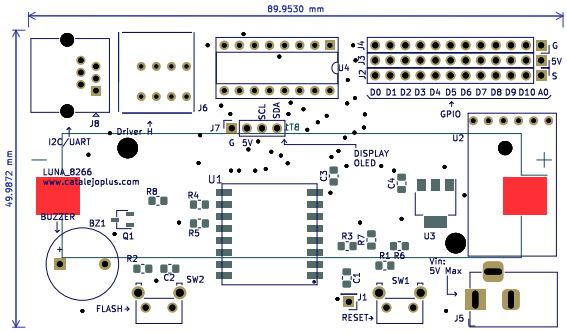
El diseño del esquemático eléctrico de la tarjeta de desarrollo se realiza de manera jerárquica y con comprobación de las reglas eléctricas (ERC) con las herramientas proporcionadas por KiCAD, el resultado es mostrado en el **anexo A**. Realizado el esquemático, se asocia los footprints de las tecnologías a usar y se procede a realizar el proceso *place and route*, este es diseñado para una PCB de dos caras (ver figura **3-19**) teniendo presente las características de implementación que se tiene en Colombia, que en este caso, puedan ser de bajo costo. Con respecto a la tecnología de empaquetado de los componentes son tipo THT y SMD; la selección de los componentes fue teniendo presenta la posibilidad de dar mantenimiento a la tarjeta de desarrollo con componentes que se pueden conseguir en Colombia, como fue el caso del driver puente H de referencia L293D, con empaquetado DIP, que al ser necesaria una sustitución, solo será quitarlo de la base para su cambio, sin requerir soldadura.



(a) Ruteo de cara superior.



(b) Ruteo de cara inferior

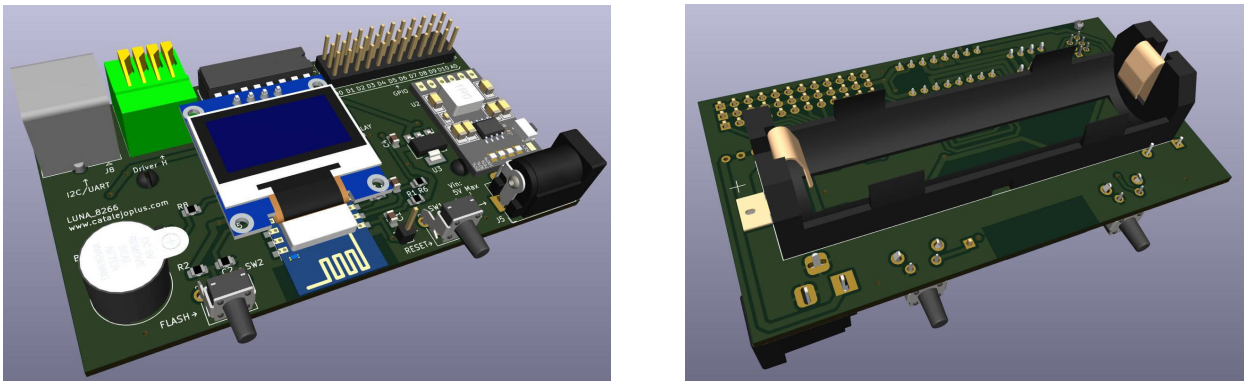


(c) Disposición espacial de los componentes

Figura 3-19: Diseño de la placa de desarrollo creado con KiCAD

Para tener una perspectiva sobre cómo se verá la PCB con los componentes soldados, se presenta en la figura **3-20** la vista 3D por ambas caras, la cual es generada con las herramientas provistas por KiCAD. Lo anterior permite realizar ajustes en el ruteo antes de implementar la PCB, reduciendo los errores de diseño que tienen un costo menor con respeto a la implementación.

[8]Sitio ofical de KiCAD `https://www.kicad.org/`



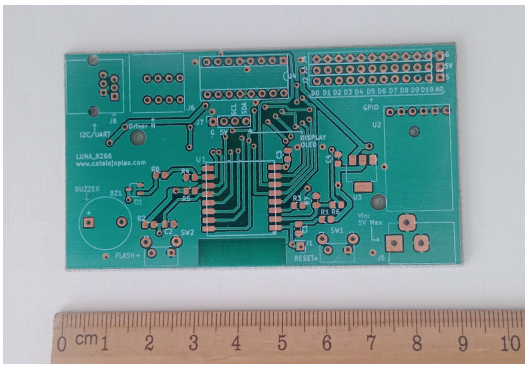
(a) Vista cara frontal (b) Vista cara inferior

Figura 3-20: Vista en tres dimensiones de la tarjeta de desarrollo

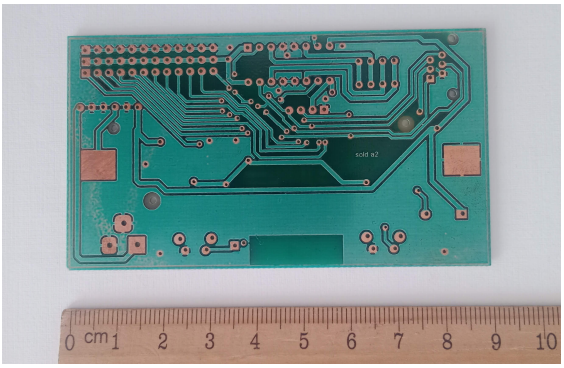
Fabricación y verificación de PCB

Realizado el diseño y comprobadas las reglas DRC, se exporta el resultado a archivos *Gerber*[9], los cuales, tienen instrucciones para la implementación de la PCB. En la figura **3-21** se puede ver el resultado de los circuitos impresos que fueron creados en Colombia, el montaje de los componentes se realizó manualmente junto a la verificación visual del resultado.

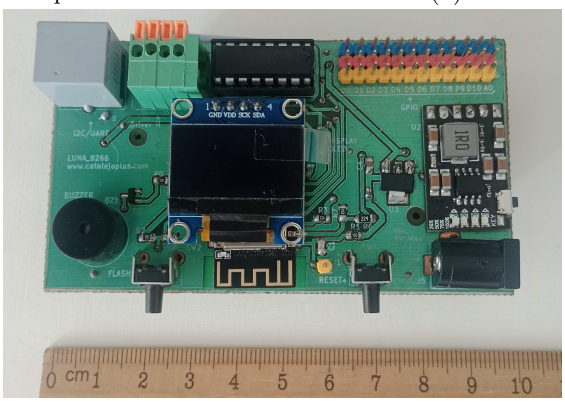
[9]Archivos que contienen la información sobre la disposición y las características de los diseños para su implementación en una PCB



(a) PCB cara superior



(b) PCB cara inferior



(c) PCB con componentes soldados

Figura 3-21: Implementación de tarjeta de desarrollo con componentes soldados

Diseño de carcasa para la tarjeta de desarrollo

Para proteger la tarjeta de desarrollo se diseña una carcasa. El proceso de diseño parte del modelo creado con KiCAD, este modelo contiene las medidas exactas que se deben tener en cuenta para el diseño de la carcasa. El modelo 3D es manipulado con la herramienta FreeCAD[10] la cual también funciona en la lógica de copyleft, desde esta aplicación se requiere instalar KiCadStepUp[11] la cual es opensource y permite generar un modelo 2D a partir del modelo 3D parametrizado con KiCAD, en este proceso se selecciona la cara superior del modelo para la exportación. El modelo generado es un modelo vectorial el cual tendrá dos propósitos; en primer lugar servirá para crear el diseño 2D de la carcasa, en segundo lugar, es la base para la creación de la documentación de uso del material concreto, con este modelo se podrá presentar los diagramas pictográficos, los cuales son usados en el proceso de imitación de montajes a realizar.

El archivo vectorial exportado desde FreeCAD es importado en otra aplicación copyleft llamada LibreCAD[12], la cual es la capa base para el modelo a diseñar. Diseñado el modelo como es mostrado en la figura **3-22** se procede a exportar en formato PDF para ser cargado en una cortadora láser que genera el resultado final. El armado de la carcasa consta de encajar cada una de las caras del modelo y atornillar en conjunto.

[10]Sitio oficial de FreeCAD: `https://www.freecad.org/`
[11]Wiki del pluging para FreeCAD: `https://wiki.freecad.org/KicadStepUp_Workbench`
[12]Sitio oficial de LibreCAD: `https://librecad.org/`

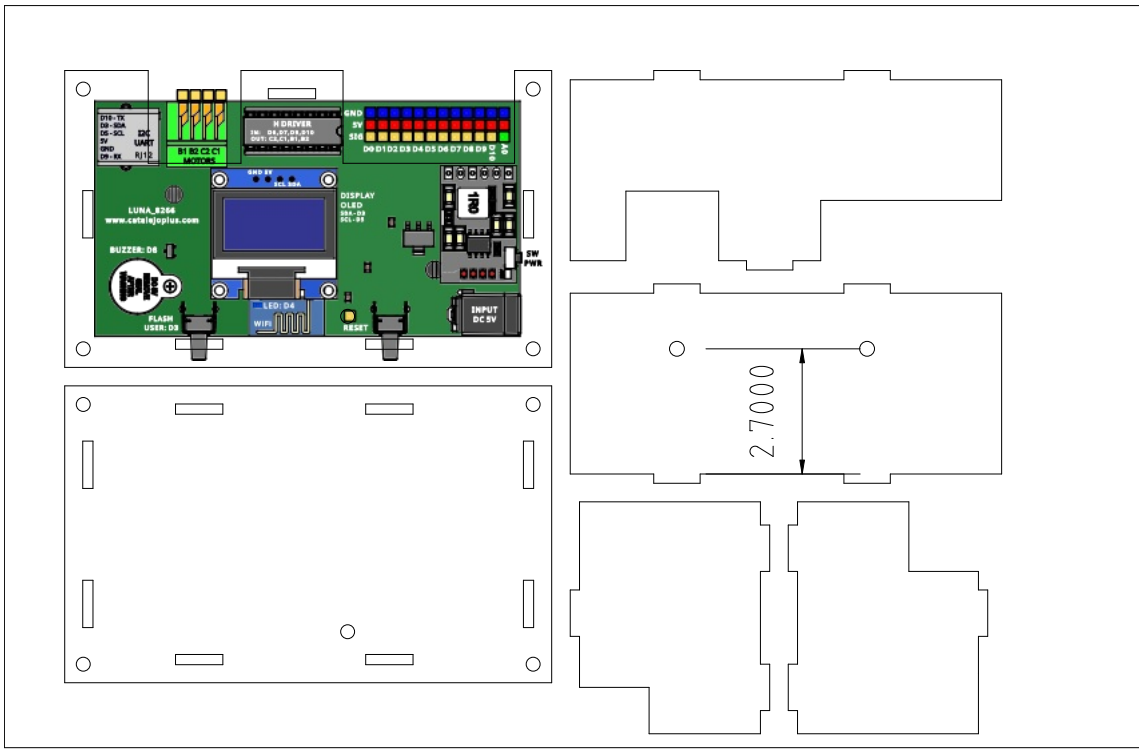


Figura 3-22: Diseño de carcasa para tarjeta de desarrollo

Resultado final de la tarjeta de desarrollo y características

El costo por unidad de la tarjeta de desarrollo se muestra en la tabla **3-6**, siendo aproximadamente de 106.200COP. Este valor incluye el proceso de montaje de la tarjeta, los componentes, la batería recargable y la carcasa.

Tabla 3-6: Costos aproximados de los componentes y montaje de la tarjeta de desarrollo

CONCEPTO	COSTO
Implementación PCB	25000 COP
Componentes	44000 COP
Montaje manual	10000 COP
Subtotal montaje	**79000 COP**
Batería recargable	20000 COP
Carcasa en acrílico	6000 COP
Tornillos	1200 COP
Total montaje	**106200 COP**

En la figura **3-23** se presenta la tarjeta de desarrollo después de haber completado exitosamente las pruebas visuales, eléctricas, de programación y el ensamblaje de la carcasa.

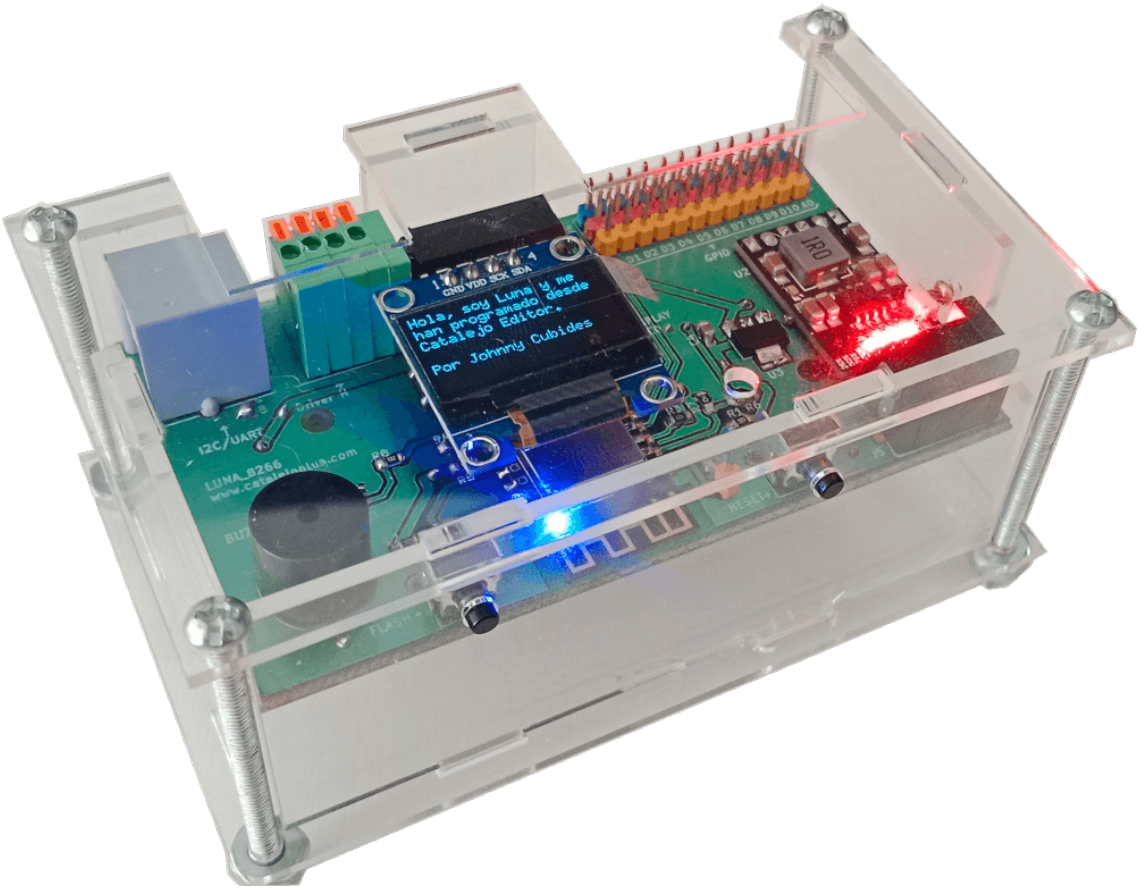


Figura 3-23: Tarjeta de desarrollo terminada cubierta por carcasa creada en acrílico y corte láser

Finalmente, en la figura **3-24** se presentan los módulos incorporados en la tarjeta de desarrollo que, en conjunto, confieren a la tarjeta las siguientes características que pueden ser aprovechadas por el usuario final:

- Comunicación inalámbrica: Permite la comunicación con otros sistemas a través de wifi para la programación de la tarjeta de desarrollo y el intercambio de datos entre máquinas, soportando protocolos TCP, UDP, OSC, entre otros.
- Protocolos de comunicación por Hardware: Soporta los protocolos estándar de comunicación con periféricos, como UART, I2C, I2S, SDIO, SPI, lo cual permite a la tarjeta conectarse con sensores, actuadores e incluso con otros controladores.
- Pantalla gráfica: Cuenta con una pantalla OLED I2C removible en la cual se pueden imprimir textos y crear gráficas. El usuario puede imprimir información de variables obtenidas por el hardware o desde la red local.
- Zumbador pasivo: El usuario puede generar sonidos a diferentes frecuencias desde la placa de desarrollo sin necesidad de incorporar más hardware.
- Conector RJ12: Fue incorporado en la tarjeta para facilitar la comunicación con otros sistemas a través de los protocolos UART e I2C. Por ejemplo, facilita la reprogramación del firmware de manera cableada

y permite el uso de dispositivos esclavos a través de I2C para extender funciones, como agregar más GPIO requeridos en proyectos. Esto es posible debido a que facilita la conexión y desconexión rápida de cables, requiriendo únicamente un conector macho RJ12 crimpado.

- Controlador Puente H: La funcionalidad de este dispositivo es proveer a la tarjeta señales de salida que puedan hacer funcionar hasta 1 amperio (2 motores en total) u otros dispositivos que requieran energía. Este módulo fue incorporado en la tarjeta de manera removible en caso de daño por razones de uso.

- Bloque conector tipo resorte: Este elemento está conectado a la salida del puente H y está incorporado para permitir al usuario conectar los cables de un motor u otro dispositivo sin requerir una herramienta adicional, simplemente oprimiendo las palancas que controlan unos resortes incorporados que sujetan los cables.

- Puertos de conexión: Diseñados con una intención didáctica, están distribuidos en 3 filas y con colores para distinguir su funcionalidad. Los pines azules representan el terminal negativo de energía, los pines rojos son el terminal positivo a 5V de tensión y finalmente, los pines amarillos representan los pines de información, los GPIO para conectar sensores y activarlos. Este puerto está implementado con pines macho de $2.54mm$, que en la línea educativa electrónica es la misma distancia que existe en los agujeros de un tablero de prototipos (protoboard) así como en los demás módulos de electrónica de la línea mencionada.

- Batería recargable: La batería seleccionada para esta tarjeta es la de referencia 18650. Para su uso, se incorporó un porta-baterías con la finalidad de sustituir la batería cuando sea necesario.

- Cargador de batería: Se incorpora con el módulo CD42-MH para cargar la batería 18650. Este módulo, a través de 4 LEDS incorporados, permite observar el estado de carga de la batería, lo que es ideal para que el usuario pueda saber cuándo se requiere recargar la batería. Además de esto, eleva la tensión de la batería de 3.7V a 5V para que los demás sensores y actuadores funcionen adecuadamente. Finalmente, este dispositivo cuenta con un interruptor de encendido que controla las funciones de la tarjeta de desarrollo.

- Pulsadores de reset: Ubicados lateralmente en la tarjeta para facilitar el acceso al usuario, están diseñados para dejar la tarjeta de desarrollo en valores de configuración de fábrica.

- Conector de recarga: Se trata de un conector Power DC a 5V. La tarjeta se entrega con un cable Power DC a USB para que el usuario pueda cargar la tarjeta mediante un cargador de celular.

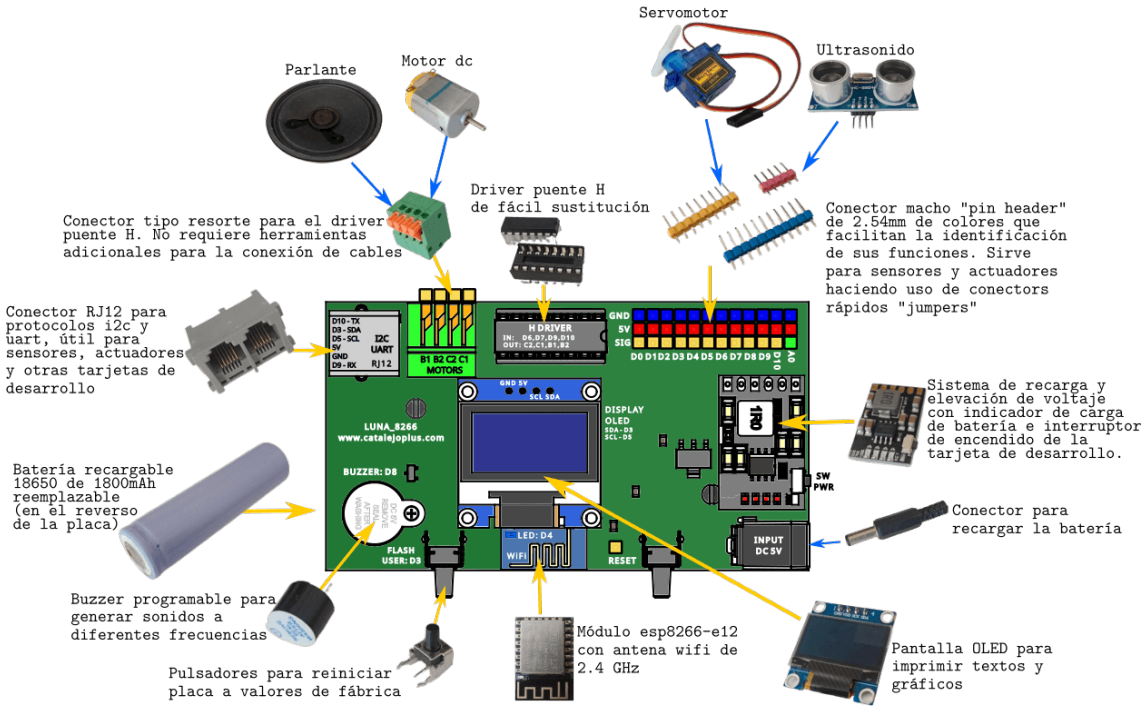


Figura 3-24: Tarjeta de desarrollo con los módulos incorporados y características principales

3.3.2 Diseño de software embebido

Según el SoC a usar, el fabricante proporciona las herramientas de desarrollo necesarias para crear la aplicación que será ejecutada en el SoC, para el caso del esp8266 fabricado por *Espressif System* se cuenta con un framework con licencia *copyleft*, copyleft permite la libre distribución, edición y adaptación de aquellas herramientas y productos, lo cual permite reducir costos y tiempo en el diseño del software.

Las herramientas copyleft basadas en el compilador de GNU cumplen el flujo de desarrollo que se muestra en la figura **3-25** el cual da como resultado un programa que debe ser almacenado en la memoria persistente disponible para el SoC (memoria interna o externa), el cual será cargado en la memoria principal disponible en el SoC para su ejecución.

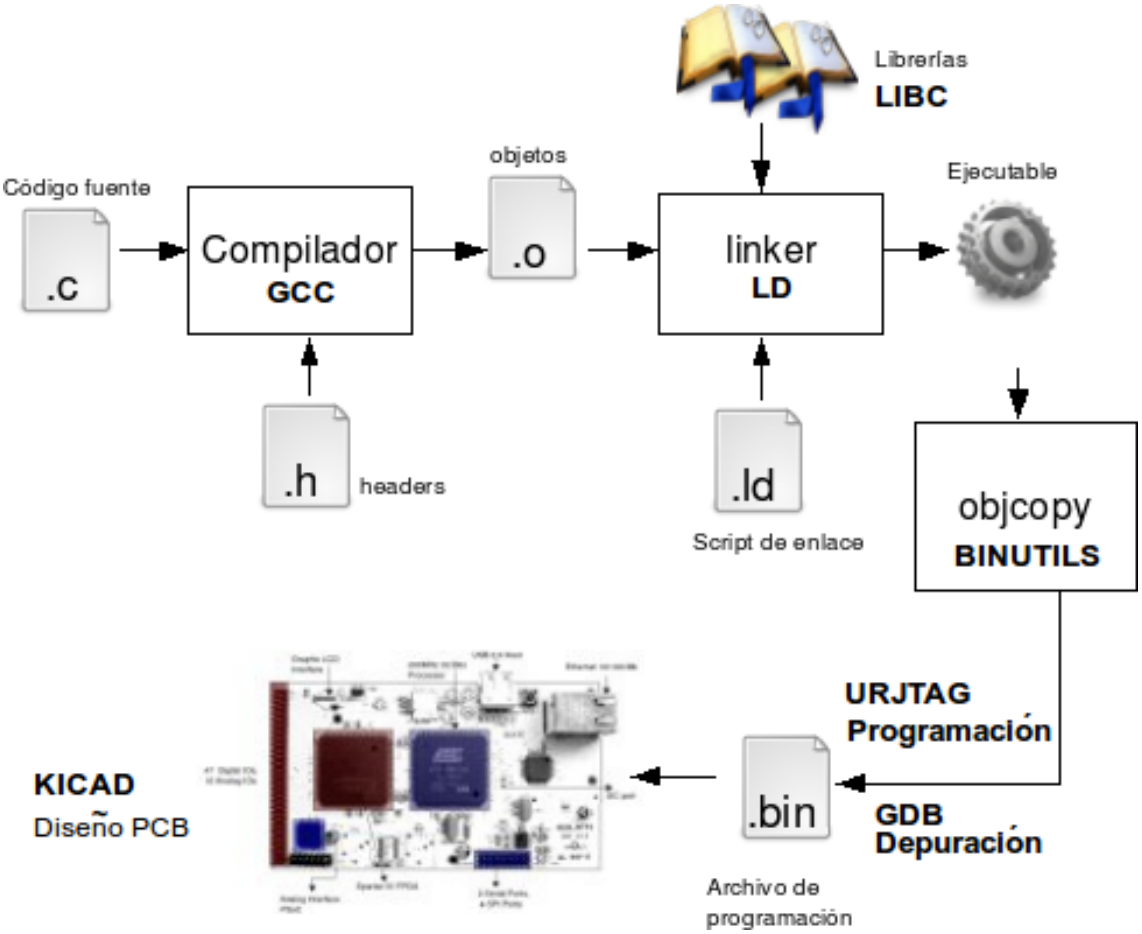


Figura 3-25: Flujo de diseño para la creación de aplicaciones. Fuente [2]

El SoC seleccionado actualmente cuenta con diferentes frameworks y ports de otros software adaptados, los cuales tienen diferentes propósitos, como es el caso de prototipado rápido, enseñanza de la programación o aplicación en campos industriales como la robótica o el Internet de las cosas. En la tabla **3-7** se muestra distintos firmwares creados con el toolchain opensource provisto por el fabricante del SoC donde se puede apreciar el lenguaje con el que es desarrollado, los objetivos con los que fueron diseñados, periféricos soportados, footprint entre otros. La tabla se diseña observando los criterios requeridos para el propósito de este kit.

Tabla 3-7: Comparativa de distintos firmwares para el SoC esp8266

Framework	**Footprint del firmware (kB)**	**Memoria RAM (kB)**	**Lenguaje de desarrollo**	**Lenguaje para Usuario Final**	**Periféricos soportados**	**Enfoque**	**Curva de aprendizaje**
Arduino	Alrededor de 300-400 kB	Alrededor de 80 kB	C/C++	C/C++	GPIO, I2C, SPI, UART, ADC, PWM, WiFi	Flexible y fácil de usar	Moderada
ESP8266 SDK	Alrededor de 100-200 kB	Alrededor de 50 kB	C	C	GPIO, I2C, SPI, UART, ADC, PWM, WiFi	Bajo nivel y potencialmente más eficiente	Moderada
MicroPython	Alrededor de 300-400 kB	Alrededor de 100 kB	C	Python	GPIO, I2C, SPI, UART, ADC, PWM, WiFi	Interactivo y fácil de aprender	Baja
NodeMCU	Alrededor de 300-400 kB	Alrededor de 100 kB	C	Lua	GPIO, I2C, SPI, UART, ADC, PWM, WiFi	Rápido desarrollo de prototipos	Baja
Zephyr	Variable	Variable	C/C++	C/C++	GPIO, I2C, SPI, UART, ADC, PWM, WiFi, entre otros	RTOS	Alta
RIOT	Variable	Variable	C	C	GPIO, I2C, SPI, UART, ADC, PWM, WiFi, entre otros	RTOS	Alta

Para seleccionar alguno de estos desarrollos y modificar las fuentes que sean necesarias para el usuario final se hace indispensable conocer el proceso de programación que deberá realizar éste mismo al interactuar en el proceso de desarrollo de su programa; en la figura **3-26** se puede observar que en **1** desde la perspectiva del usuario final, construirá un programa haciendo uso de un lenguaje de programación, el cual, deberá entregar a un traductor para que este convierta el programa escrito en un lenguaje de alto o medio nivel a un programa de bajo nivel llamado lenguaje máquina. Para el propósito anteriormente mencionado, el usuario final tendrá dos flujos de desarrollo diferentes que dependerán de las técnicas a usar; por un lado como se muestra en la figura anteriormente citada, en **2**, se puede realizar un proceso de programación compilada el cual requiere de un compilador cruzado para el proceso de traducción, el cual será necesario cada vez que se crea un nuevo programa, este es el caso que se da cuando se hace uso de por ejemplo Arduino IDE, Zephyr OS, RIOT OS vistos en la tabla **3-7** . Si se toma el camino mostrado en la figura **3-26** marcado con **3** el usuario tendrá que escribir un programa en un lenguaje que generalmente es de alto nivel y enviarlo al intérprete que estará en el SoC, el cual, lee el script, evalúa las reglas, ejecuta y se queda en un bucle esperando más instrucciones, sin embargo, para que sea posible el proceso de interpretación por parte del SoC, se debe realizar un trabajo previo el cual es mostrado en la figura **3-26** con **4** el cual se realiza desde la perspectiva del desarrollador, el cual, debe preparar los recursos de intérprete y programar el SoC.

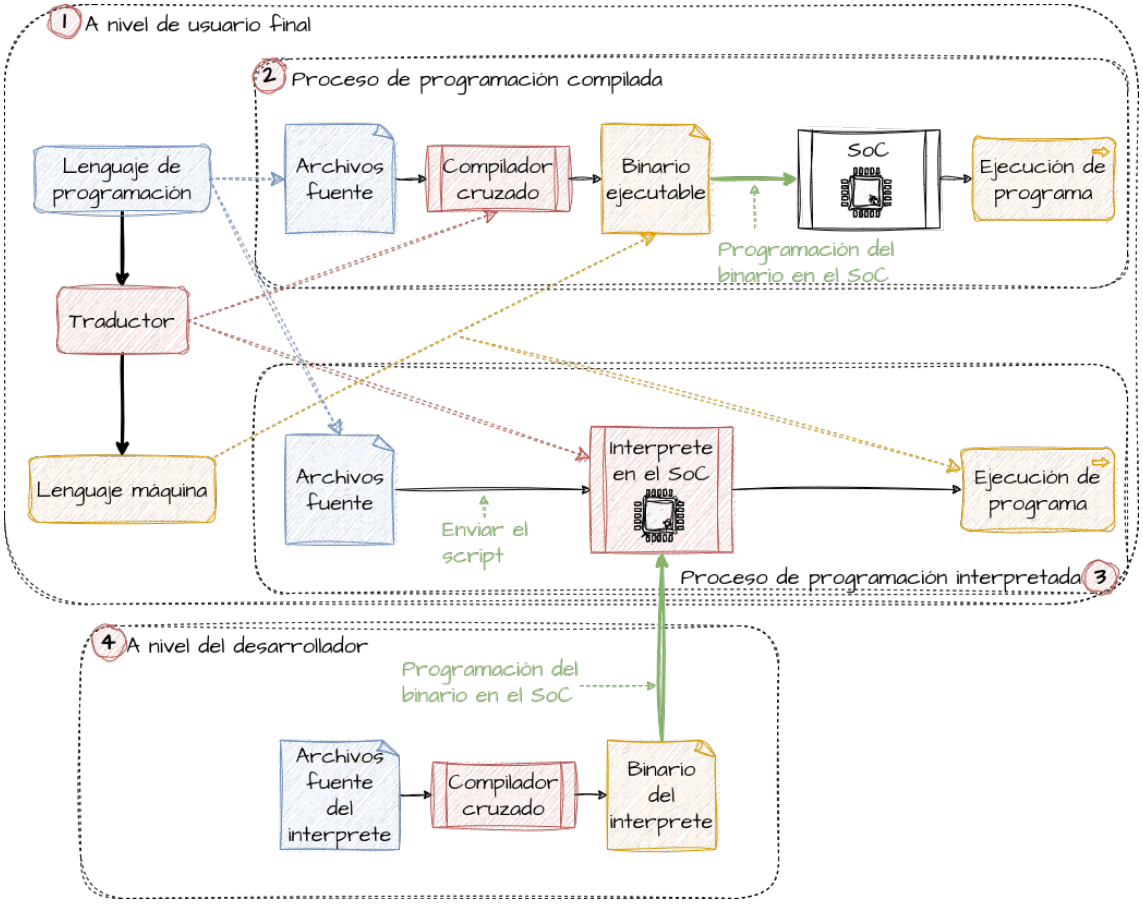


Figura 3-26: Comparación entre el proceso de programación compilada e interpretada para un SoC

Para la creación de este prototipo la técnica seleccionada es la programación interpretada, debido a que desde la perspectiva del usuario final tendrá las siguientes ventajas:

- **No se requiere compilador cruzado**: generalmente el compilador cruzado es embebido en la aplicación que se encuentra en el HOST (Dispositivo diferente del SoC, el SoC es el objetivo), requerirá de un equipo con un sistema operativo para el compilador, difícilmente la aplicación donde el usuario crea su programa podrá ser portada a otros sistemas, además el usuario podrá quedar limitado a métodos de programación del SoC donde deberá hacer un esfuerzo mayor para solucionar problemas antes de guardar el programa realizado a la memoria persistente del SoC
- **Lenguaje de programación flexible y limpio**: Al seleccionar un lenguaje como puede ser Python o Lua, permite que el usuario se enfoque en el algoritmo a diseñar en lugar de tener presente las diferentes reglas que tienen los lenguajes compilados como es el caso de C para optimizar el programa, aunque la optimización que se ha puesto como ejemplo es relevante para los SoC por sus restricciones de procesamiento, temporales y de memoria, el SoC seleccionado, cuenta con las características necesarias que permitieron los ports en lenguajes *Lua* y *Python* de la tabla **3-7**
- **Programación inalámbrica del SoC**: Desde el punto de vista del usuario final, la programación del SoC es un proceso que puede ser engorroso, por ejemplo, la instalación de drivers en el HOST para controlar la comunicación con el SoC puede fallar debido a las actualizaciones del sistema operativo del HOST que dejan inutilizado este procedimiento; debido a que el SoC cuenta con un módulo de comunicación no guiada, se puede ajustar el intérprete para que inicie un servicio que permita

almacenar los scripts en la memoria persistente convirtiendo este proceso en una tarea transparente para el usuario, permitiendo la programación del SoC desde diferentes dispositivos como puede ser, tablets, celulares o computadores personales.

Teniendo presente las características que se quieren tener para favorecer el proceso de programación que realiza el usuario final de la tabla **3-7** se ha seleccionado NodeMCU por las siguientes razones:

- NodeMCU inicialmente está orientado al SoC esp8266.
- Este framework hace uso del lenguaje de programación Lua, el cual es un lenguaje de scripting extensible, simple, eficiente y portable[32] , es usado en el desarrollo de videojuegos como es el caso de *love2d*[13], Roblox[14] y Grand Theft Auto V[15] , además, Lua que está escrito en C fue diseñado para poder ser embebido principalmente en C lo que permite el desarrollo de proyectos de scripting haciendo uso de Zephyr OS[16] , RIOT OS[17] , Elua[18], entre otros, que a futuro facilitarán el desarrollo de aplicaciones para el SoC.
- NodeMCU tiene soportado los periféricos requeridos para este kit en lenguaje C y en Lua.
- Debido a que está basado en Lua, se permite la rápida adaptación de librerías en C para agregar funcionalidades al SoC.
- El framework cuenta con un script de tamaño reducido llamado **luacross** que permite reducir el tamaño de los scripts de Lua creados, traduciendo a bytecodes que a su vez, aumenta la velocidad de ejecución y la eficiencia en el uso de memoria principal en el SoC.

Haciendo uso de las herramientas provistas por los desarrolladores de NodeMCU en el repositorio opensource `https://github.com/nodemcu` se construyen los diferentes módulos requeridos. Se destaca en el codigo 3.1 la incorporación del código del Open Sound Control (OSC) como demostración de las reglas a seguir para agregar módulos escritos en C al intérprete. OSC permite al SoC hacer parte de una red de instrumentos musicales que reciben y envían mensajes para funcionar de manera síncrona como lo hace una orquesta. El protocolo OSC portado del repositorio `https://github.com/mhroth/tinyosc` permite que el dispositivo se una a una red de dispositivos musicales a través de un identificador del tipo árbol como se hace en GNU/Linux o en MQTT.

```
1  // Nombre del archivo: osc.c
2  // Autor: Johnny Cubides
3  // Licencia: GPL
4
5  #include "lua.h"
6  #include "module.h"
7  #include "lauxlib.h"
8  #include "platform.h"
9  #include "osc/osc.h"
10
11 #include <stdint.h>
12 #include <string.h>
13 #include "user_interface.h"
14
```

[13]Sitio oficial del framework: `https://love2d.org/`
[14]Sitio de Roblox relacionado a Lua scripting: `https://create.roblox.com/docs/es-es/studio/script-editor`
[15]Sitio de *fivem* para scripting en lenguaje Lua: `https://docs.fivem.net/docs/scripting-manual/introduction/creating-your-first-script/`
[16]Sitio oficial de Zephyr OS con Lua embebido: `https://docs.zephyrproject.org/latest/releases/release-notes-3.1.html`
[17]Lua portado en RIOT: `https://api.riot-os.org/group__pkg__lua.html`
[18]Sitio oficial de eLua: `https://eluaproject.net/`

```
15 static int osc_set_ifs( lua_State* L)
16 {
17   const char *address = luaL_checkstring(L, 1); // address
18   unsigned arg1 = luaL_checkinteger( L, 2 ); // arg1 integer
19   float arg2 = (float)luaL_checknumber( L, 3 ); // arg2 float
20   const char *arg3 = luaL_checkstring(L, 4);  // arg3 string
21
22   char buffer[128];
23   char *pBuffer = buffer;
24
25   int size = tosc_writeMessage(
26       buffer,
27       sizeof(buffer),
28       address,
29       "ifs",
30       arg1,
31       arg2,
32       arg3
33       );
34   lua_pushlstring(L, pBuffer, size);
35   lua_pushinteger(L, size);
36   return 2;
37 }
38
39 static int osc_set_iiff( lua_State* L)
40 {
41   const char *address = luaL_checkstring(L, 1); // address
42   unsigned arg1 = luaL_checkinteger( L, 2 ); // arg1 integer
43   unsigned arg2 = luaL_checkinteger( L, 3 ); // arg2 integer
44   float arg3 = (float)luaL_checknumber( L, 4 ); // arg3 float
45   float arg4 = (float)luaL_checknumber( L, 5 ); // arg4 float
46
47   char buffer[128];
48   char *pBuffer = buffer;
49
50   int size = tosc_writeMessage(
51       buffer,
52       sizeof(buffer),
53       address,
54       "iiff",
55       arg1,
56       arg2,
57       arg3,
58       arg4
59       );
60   lua_pushlstring(L, pBuffer, size);
61   lua_pushinteger(L, size);
62   return 2;
63 }
64
65 static int osc_get_iifs( lua_State* L)
66 {
67   uint8_t size = luaL_checkinteger(L, 1);
68   const char *pBuffer = luaL_checkstring(L, 2); // buffer
69
70   int len = strlen(pBuffer); // the number of bytes read from the socket
71   tosc_message osc; // declare the TinyOSC structure
72
73   char buffer[size +1];
```

```
74   /* strcpy(buffer, pBuffer); */
75   memcpy(buffer, pBuffer, size+1);
76
77   char address[24];
78   char *pAddress = address;
79   /* char *pAddress; */
80   int argInt[] = {0,0};
81   float argFloat = 0.0;
82
83   if (!tosc_parseMessage(&osc, buffer,size))
84   {
85     pAddress = tosc_getAddress(&osc);
86     argInt[0] = tosc_getNextInt32(&osc);
87     argInt[1] = tosc_getNextInt32(&osc);
88     argFloat = tosc_getNextFloat(&osc);
89     const char *text = tosc_getNextString(&osc);
90     lua_pushlstring(L, pAddress, strlen(pAddress));
91     lua_pushinteger(L, argInt[0]);
92     lua_pushinteger(L, argInt[1]);
93     lua_pushnumber(L, argFloat);
94     lua_pushlstring(L, text, strlen(text));
95     return 5;
96   }
97   lua_pushlstring(L, pAddress, strlen(pAddress));
98   lua_pushinteger(L, argInt[0]);
99   lua_pushinteger(L, argInt[1]);
100  lua_pushnumber(L, argFloat);
101  lua_pushlstring(L, "e", 1);
102  return 5;
103 }
104
105 LROT_BEGIN(osc, NULL, 0)
106   LROT_FUNCENTRY( set_ifs, osc_set_ifs )
107   LROT_FUNCENTRY( set_iiff, osc_set_iiff )
108   LROT_FUNCENTRY( get_iifs, osc_get_iifs )
109 LROT_END(osc, NULL, 0)
110
111
112 NODEMCU_MODULE(OSC, "osc", osc, NULL);
```

Listing 3.1: Módulo en lenguaje C incorporado al interprete de NodeMCU para soportar el protocolo OSC

Resultado final del software embebido

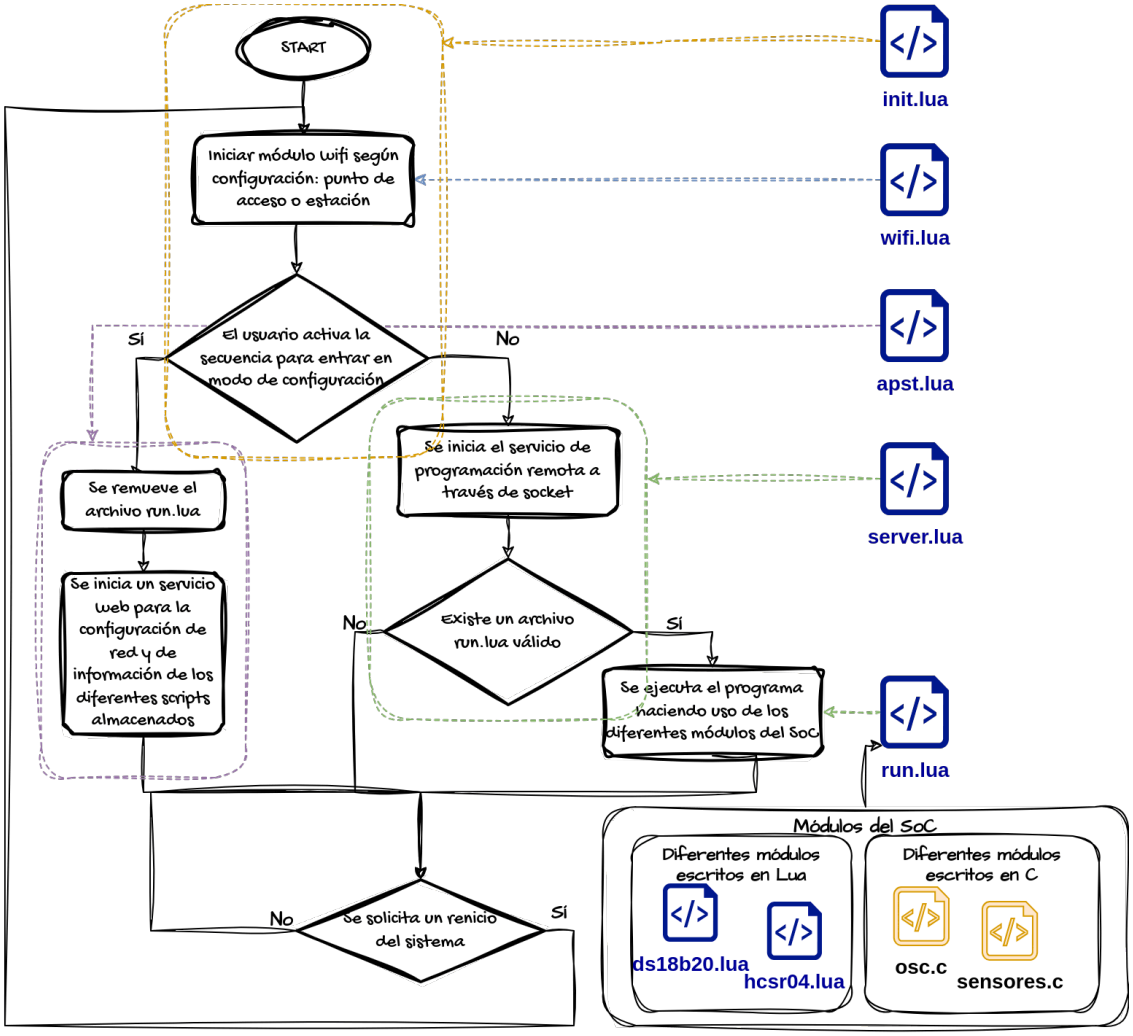


Figura 3-27: Configuración del software embebido para el SoC basado en NodeMCU

Para finalizar este apartado se muestra la configuración final del software embebido, en la figura **3-27** se describe el flujo que sigue en ejecución el SoC con las siguientes características:

- Cuando se inicia el software NodeMCU en el SoC después de levantar el intérprete, este busca un archivo *init.lua* válido, este script está configurado para dar inicio a los servicios de comunicación inalámbrica descritos en el script *wifi.lua*.
- El script *wifi.lua* verifica la última configuración realizada al módulo de red y establece los modos de punto de acceso, estación o un modo híbrido entre estos dos.
- Otra responsabilidad del archivo *init.lua* es atender las solicitudes de configuración de valores iniciales por parte del usuario final, las cuales son dadas por una secuencia con los botones de reset y de usuario; si se hace esta solicitud cargará el script *apst.lua*, en caso de no ser activada la secuencia se lanzará el script *server.lua*, el cual corresponde a la secuencia normal de programación del SoC.

- Cuando se activa el script *apst.lua* la primera acción a realizar es remover el archivo *run.lua*; este archivo contiene el programa que ha sido escrito por el usuario, es una acción útil para los casos donde el script *run.lua* contiene instrucciones con errores semánticos que afectan los procesos de reprogramación.

- El script *apst.lua* también inicia un servicio web que permite la configuración del modo estación wifi, esto permite que el dispositivo sea agregado a una red local con acceso opcional a Internet. Este servicio web también permite obtener Información sobre los scripts almacenados en el intérprete, en la figura **3-28** puede observar un ejemplo de la información reportada por el intérprete.

- El script *server.lua* al ser cargado, lanza un servicio TCP que es responsable de recibir el guion creado por el usuario y almacenarlo en un archivo *run.lua*; este servicio cuenta con unos filtros sobre las instrucciones recibidas por el sockect que le permite realizar diferentes operaciones como borrar el script, almacenar el script, reiniciar el intérprete, reportar las IPs a las que tenga acceso entre otras opciones.

- El script *server.lua* ejecuta un archivo *run.lua* válido el cual es programado por el usuario final.

Finalmente la información técnica sobre la implementación de estos scripts se encuentra en el **anexo B**.

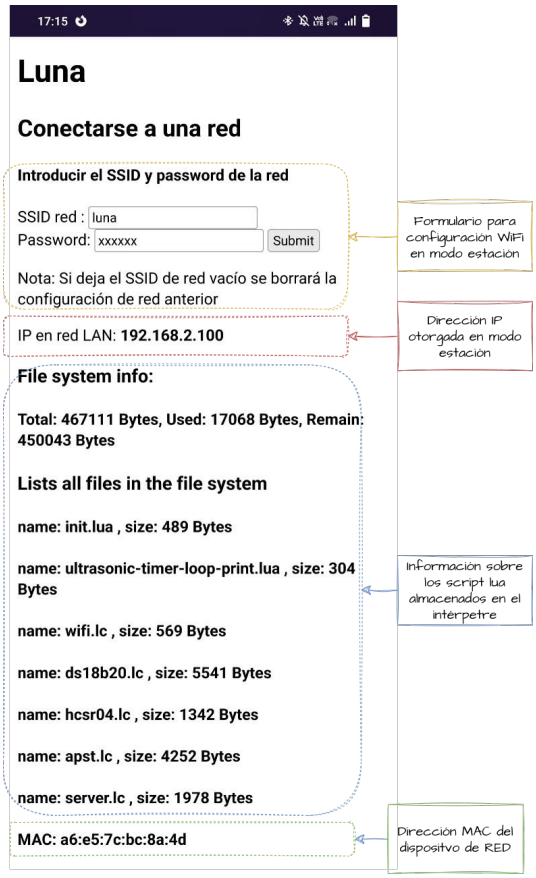


Figura 3-28: Servicio web presentado por el SoC cuando este entra en modo de configuración

3.4 Diseño de entorno de desarrollo integrado IDE

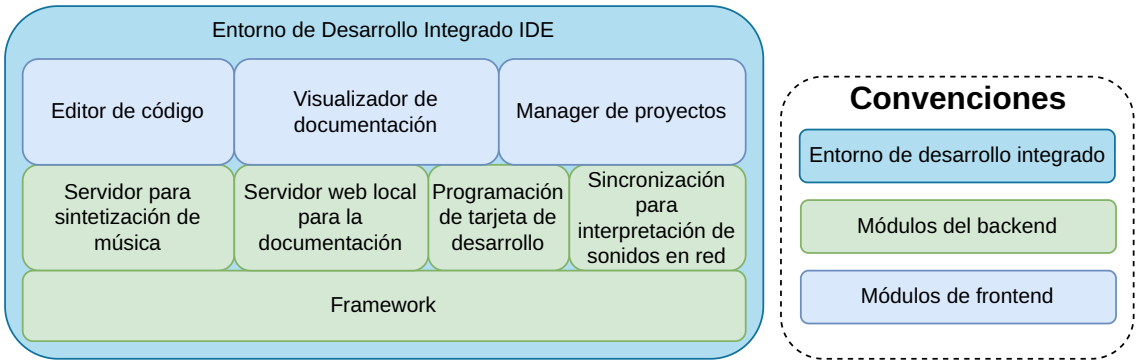


Figura 3-29: Módulos requeridos en el IDE a diseñar

En este apartado se describe el diseño de la aplicación con la cual el usuario interactúa para hacer uso del material concreto que fue definido. En la Figura **3-29** se muestran los diferentes módulos que debe tener la aplicación en el diseño conceptual. Para definir la tecnología de implementación, es necesario conocer las características deseadas en la experiencia de usuario, las cuales se listan a continuación:

- Portabilidad: la aplicación a diseñar deberá ser ejecutada en ambientes basados en Windows, MacOS o Linux, de tal manera que el usuario pueda hacer uso de los recursos disponibles.
- Servicio Web: los recursos documentales de acceso en línea deben ser integrados para experiencias donde no se cuente con internet.
- Interoperabilidad con otras Apps: se requiere acceder y controlar otros servicios que brinden mayores posibilidades al momento de crear artefactos con el material concreto.
- Software productivo: las diferentes tareas a integrar, desde la perspectiva copyleft, podrían ser resueltas por medio de bibliotecas de código abierto, las cuales son usadas en entornos productivos, favoreciendo la experiencia de usuario para no sufrir fallos de ejecución debido al proceso de desarrollo.

Según las características planteadas, en este caso se va a requerir el uso de un framework que facilite el proceso de desarrollo de un prototipo funcional para su evaluación.

3.4.1 Selección del Framework

Existen diferentes tecnologías que se pueden usar para la creación de aplicaciones. Para el desarrollo de este prototipo se hace uso de un Framework basado en tecnología web. En la Tabla **3-8** se muestran 6 frameworks populares para el desarrollo de aplicaciones con tecnología web, mostrando sus principales características.

Tabla 3-8: Comparación de frameworks para el desarrollo de aplicaciones de escritorio con tecnología web

Framework	**Lenguaje de programación**	**APIs para control de procesos**	**Flexibilidad**	**Tamaño de la hermienta de desarrollo**	**Lenguaje del framework**
Electron.js	JavaScript, HTML, CSS	Acceso al sistema de archivos, sockets, procesos hijos	Moderada	Moderado	JavaScript, C++
Flutter	Dart	Acceso al sistema de archivos, sockets, conexiones TCP/IP	Alta	Grande	Dart
NW.js	JavaScript, HTML, CSS	Acceso al sistema de archivos, sockets, procesos hijos	Moderada	Moderado	JavaScript, C++
Node.js	JavaScript	Acceso al sistema de archivos, sockets, procesos hijos	Alta	Pequeño	JavaScript, C++
Tauri	Rust, JavaScript, HTML, CSS	Acceso al sistema de archivos, sockets, conexiones TCP/IP	Alta	Pequeño	Rust, JavaScript
NeutralinoJS	JavaScript, HTML, CSS	Acceso al sistema de archivos, sockets, procesos hijos	Alta	Pequeño	JavaScript, C++

El framework seleccionado para el desarrollo del prototipo IDE ha sido NWJS[19] por las siguientes razones:

- El portal oficial de NWJS contiene el framework compilado para los sistemas operativos más populares, lo que acelera el proceso de desarrollo.
- Los proyectos empaquetados con NWJS tienden a ser más pequeños en comparación con Electron y Flutter.
- Las bibliotecas distribuidas en npmjs son compatibles con NWJS.
- NWJS proporciona diferentes APIs de acceso a funciones del sistema operativo que facilitan el desarrollo de aplicaciones.

3.4.2 Diseño de backend

El backend tiene la responsabilidad de gestionar los diferentes servicios que serán requeridos por el usuario. En la Figura **3-29** se tienen los servicios de síntesis de música, servidor web para la documentación, servicio de programación de la tarjeta de desarrollo, sincronización para la interpretación de sonidos en red.

A continuación se muestra la implementación de cada uno de estos servicios haciendo uso de los recursos previstos por NWJS y otras aplicaciones que se integran en el backend.

[19]Sitio oficial de NWJS: `https://nwjs.io/`

Servicio de síntesis de música

Este servicio es implementado con ChucK, el cual es compilado con las herramientas GNU para los diferentes sistemas operativos. Este servicio es controlado por el backend a través de un proceso hijo que controla los procesos de ejecución, tiene una API diseñada para las distintas tareas de usuario. En el Código 3.2 se observan las funciones de la API accesibles al usuario, las cuales hacen uso de procesos hijos a través de SPAWN, el cual es una API otorgada por NWJS.

```
1  // Snippet
2  //  Autor: Johnny Cubides
3  //  Licencia: GPL
4  // -- START CHUCK OPTIONS --
5  let optionsChuck: Array<string>;
6  optionsChuck = [];
7  export function setOptionChuck(newOptions: Array<string>) {
8    optionsChuck = newOptions;
9    return optionsChuck;
10 }
11 // -- END CHUCK OPTIONS --
12
13 // -- START COMANDOS CHUCK --
14 export async function killChuck(): Promise<string> {
15   const ret = await cmd(chuckExec, ["--kill"]);
16   chuckOn = ChuckStatus.DISCONNECT;
17   return ret;
18 }
19
20 export async function removeTheLastScripChuck(): Promise<string> {
21   const ret = await cmd(chuckExec, ["--"]);
22   return ret;
23 }
24
25 export async function removeScriptChuck(id: string): Promise<string> {
26   const ret = await cmd(chuckExec, ["-", id]);
27   return ret;
28 }
29
30 export async function removeAllScriptChuck(): Promise<string> {
31   const ret = await cmd(chuckExec, ["--remove.all"]);
32   return ret;
33 }
34
35 export async function addScriptChuck(file: string): Promise<string> {
36   const ret = await cmd(chuckExec, ["+", file]);
37   return ret;
38 }
39
40 export function cmd(program: string, argument: Array<string>): Promise<string> {
41   const child = spawn(program, [...optionsChuck, ...argument]);
42   return new Promise((resolveFunc) => {
43     child.stdout.on("data", async (data: ArrayBuffer) => {
44       log(await ab2str(data));
45     });
46     child.stderr.on("data", async (data: ArrayBuffer) => {
47       log(await ab2str(data));
48     });
49     child.on("close", (data: string) => {
50       log("close: " + data);
```

```
51      resolveFunc(data);
52    });
53  });
54 }
55 // -- END COMANDOS CHUCK --
```

Listing 3.2: Código en JavaScript de la API para el uso del servicio de ChucK

Servicio web local para la documentación

La documentación a alojar es principalmente estática, NWJS permite la carga de estos servicios, sin embargo, para este prototipo se planteó el montaje de un servidor web dedicado como es el caso de Apache, Nginx, Lighttpd. Para este caso, se decidió el uso del servidor web Civetweb[20], el cual es portable y está preconfigurado para presentar contenido desde un directorio particular, en esta lógica de diseño, para agregar contenido documental, basta con anexar la documentación en una carpeta y lanzar el servidor web. En el código 3.3 se observa cómo se hace uso de este servicio nuevamente a través de la API de SPAWN.

```
1
2 /**
3  * @function startCivetweb
4  * Se inicia civetweb de manera local y se guarda una referencia de esté
5  * proceso hijo en children.
6  */
7 export async function startCivetweb(): Promise<string | undefined> {
8   const child = await spawn(civetWebExec, ["--document_root", "../dist"]);
9   children.push(child);
10  debug("init civetweb");
11  civetwebState = WebserviceStatus.START;
12  return new Promise((resolveFunc) => {
13    child.stdout.on("data", async (data: ArrayBuffer) => {
14      log(await ab2str(data));
15    });
16    child.stderr.on("data", async (data: ArrayBuffer) => {
17      const message = await ab2str(data);
18      log("civetweb.stderr: " + message);
19      handleChatLog(message);
20    });
21    child.on("close", (data: string) => {
22      log("civetweb.close: " + data);
23      civetwebState = WebserviceStatus.STOP;
24      resolveFunc(data);
25    });
26  });
27 }
```

Listing 3.3: Código en JavaScript de la API para el uso del servicio web CivetWeb

Programación de la tarjeta de desarrollo

Para realizar la programación de la tarjeta existen dos métodos: programación por puerto serial y programación a través de socket TCP/IP. NWJS cuenta con el acceso a ambos métodos. Sin importar el método de envío, el código Lua deberá ser modificado con el formato de la función *saveLuna* indicado en el Apéndice

[20] Sitio oficial de Civetweb: `https://github.com/civetweb/civetweb`

B para que sea almacenado en el sistema embebido. En el código 3.4 se muestra la función responsable de la codificación.

```
1
2  class CodeToSend {
3    code: string;
4    splitCode: Array<string>;
5    lenSplitCode: number;
6    constructor() {
7      this.code = "";
8      this.splitCode = [];
9      this.lenSplitCode = 0;
10   }
11   setCode(code: string) {
12     this.code = code;
13     this.splitCode = code.split("\n");
14     this.lenSplitCode = this.splitCode.length;
15   }
16   getLenSplitCode() {
17     return this.lenSplitCode;
18   }
19   getCodeFromArray(index: number) {
20     if (index < this.lenSplitCode) {
21       return this.splitCode[index] + "\n";
22     }
23     return "undefined";
24   }
25   setCodeToSaveInLuna8266() {
26     let tempCode = "";
27     for (const indexCode in this.splitCode) {
28       tempCode += "save___data" + this.splitCode[indexCode] + "\n";
29     }
30     tempCode = "save___begin\n" + tempCode + "save___end\n";
31     this.splitCode = tempCode.split("\n");
32     this.lenSplitCode = this.splitCode.length;
33   }
34 }
```

Listing 3.4: Código en JavaScript sobre la generación del formato para el almacenamiento de scripts en la tarjeta de desarrollo

Servicio de sincronización para la interpretación de sonidos en red

Este servicio es una característica LOLC la cual está incorporada en ChucK. Los sonidos en red pueden ser generados a través de dos métodos: conexión a un servicio de ChucK de manera remota y uso del protocolo OSC para la modificación de estado de variables de un programa. La comunicación remota a un servicio de ChucK permite que en una red exista un único servicio de ChucK que interpreta todos los scripts de una orquesta en el modelo de *Live Orchestra*. Este servicio es posible realizando una modificación a la variable *optionChuck* a través de la función *checkChuckRemoteConnection* que se muestra en el Código 3.5.

```
1 /**
2  * @function checkChuckRemoteConection
3  * Esta función comprueba que en la IP especificada haya un
4  * servicio chuck remoto y si es así realiza una configuración
5  * en las opciones del comando cmd() responsable de las tareas
6  * de chuck
7  */
```

```
export async function checkChuckRemoteConection(ip: string) {
  const ret = await cmd(chuckExec, [
    "--remote:" + ip,
    "--port:2215",
    "--status",
  ]);
  if (ret == "0") {
    setOptionChuck(["--remote:" + ip, "--port:2215"]);
    return true;
  }
  return false;
}
```

Listing 3.5: Código en JavaScript para configurar el envío de scripts a un servicio de ChucK remoto

Para el uso del protocolo OSC, se debe escribir un guion en el lenguaje ChucK y este debe ser ejecutado por el servicio. En los Códigos 3.6 y 3.7 se puede observar un ejemplo donde la tarjeta de desarrollo envía datos a un servicio de ChucK, el servicio de ChucK al detectar que la dirección desde donde provienen los datos es */chuck/piano* modifica la nota que debe ejecutar el piano. En los Códigos 3.8 y 3.9, el servicio de ChucK envía datos a la tarjeta de desarrollo, cuando esta detecta que provienen de la dirección */ping* imprime en la terminal los datos recibidos.

```
// Programa para recibir datos enviados desde el protocolo OSC
OscIn oin;
6449 => oin.port;
OscMsg msg;

// El filtro de los mensajes debe corresponder
// al address "/ping"
oin.addAddress( "/chuck/piano" );
Rhodey piano => dac;

while (true)
{
    // Se activa los eventos de escucha para este protocolo
    oin => now;

    // Del mensaje recibido, se toma los siguientes datos
    // para alterar el instrumento piano
    while (oin.recv(msg) != 0) {
        msg.getInt(0) => int note;
        Std.mtof(note) => piano.freq;
        <<< note >>>;
    }
}
```

Listing 3.6: Ejemplo de código en lenguaje Chuck sobre la recepción de información en el servicio de ChucK por medio del protocolo OSC

```
wifi.setmode(wifi.STATION)
cfg = {}
cfg.ssid="nombre_de_red"
cfg.pwd="password"
wifi.sta.config(cfg)
wifi.sta.autoconnect(1)

sk = net.createUDPSocket()
-- Función de envío de notas desde el prompt de lua
-- ejemplo: sc(60) -- el 60 corresponde al do central
```

```
-- en midi
function sc(i)
  s, len = osc.set_iiff("/chuck/piano", i, 0, 0, 0 )
  print(len)
  print(s)
  sk:send(6449, "192.168.1.5", s)
end
```

Listing 3.7: Ejemplo de código en Lua del envío de información desde NodeMCU a través del protocolo OSC

```
// Mensaje de salida OSC
OscOut xmit;

// Parámetros del servicio donde se requiere enviar el mensaje OSC
6449 => int port;
xmit.dest ( "192.168.222.212", port );

while( true )
{
    xmit.start("/ping")
        .add(1) // Entero
        .add(256) // Entero
        .add(0.7777777) // Float
        .add("off") // String
    .send();

    1 :: second => now;
    <<<"test">>>;
}
```

Listing 3.8: Ejemplo en lenguaje ChucK del envío de información desde el servicio ChucK a través del protocolo OSC

```
-- Ejemplo de recepción de datos con el protocolo OSC
--
wifi.setmode(wifi.STATION)
cfg = {}
cfg.ssid="nombre_de_red"
cfg.pwd="pasword"
wifi.sta.config(cfg)
wifi.sta.autoconnect(1)

-- Set up UDP client socket
local sk = net.createUDPSocket()

local osc_udp = {}

local function receive(_, data)
  print(string.len(data))
  -- La función get_iifs recive [adress, entero, entero, float, string]
  osc_udp.a, osc_udp.i1, osc_udp.i2, osc_udp.f, osc_udp.s = osc.get_iifs(string.len(data), data)
  if osc_udp.a == "/ping" then
    print(osc_udp)
  end
end

sk:listen(6449)

-- Se inicia un servicio para recibir mensajes a traves del protocolo OSC
```

```
27 sk:on("receive", receive)
```

Listing 3.9: Ejemplo en lenguaje Lua de la recepción de información desde NodeMCU a través del protocolo OSC

API Controladora

Los anteriores servicios son accesibles al usuario a través de una capa llamada *controller*, la cual tiene la responsabilidad de facilitar el proceso de portación de código. Es decir, en el caso de no hacer uso de NWJS y usar otro framework, bastaría con agregar el código específico para que el backend pueda realizar las mismas operaciones deseadas sin afectar la experiencia de usuario en gran medida.

3.4.3 Diseño del frontend

En este apartado se describe la implementación de los módulos del frontend, haciendo referencia a la figura **3-29**. Los módulos incluyen el editor de código, el visualizador de documentación y el gestor de proyectos. La capa de frontend es la parte de la aplicación responsable de la experiencia de usuario y, además de definir los elementos visuales, hace uso de los recursos del backend.

A continuación, se describe cada uno de ellos.

Editor de código

Una de las principales actividades en el IDE es la modificación de código. Para el propósito educativo de la propuesta, es relevante facilitar este proceso para un usuario inexperto. Normalmente, los editores de código están asociados con lenguajes textuales, es decir, son principalmente editores de texto. Actualmente, existen interfaces de programación gráfica basadas en objetos que se interconectan para generar código, como es el caso de Scratch y Blockly[21].

El editor de código diseñado para este IDE deberá permitir tanto la programación gráfica como la programación textual, para que el usuario pueda aprender en diferentes niveles de complejidad que a futuro pueden darle flexibilidad en el desarrollo de programas. Para el editor de código se utilizará el framework CodeMirror[22]. Este framework permite el uso de diferentes lenguajes de programación, como Lua, JavaScript, C, entre otros. Para la programación gráfica, se utilizará el framework denominado Blockly. Para el diseño de este prototipo de IDE, se requiere el soporte inicial de dos lenguajes de programación, ChucK y Lua. En el caso de CodeMirror, se puede hacer una adaptación fácil para el lenguaje de programación ChucK, ya que es similar a JavaScript y a C en su sintaxis.

En el caso del framework Blockly, se requerirá una modificación adicional. Blockly genera código para otros lenguajes, que es su propósito. Por defecto, Blockly genera código para Lua, Dart, JavaScript, Python y PHP. Desde el punto de vista de la tarjeta de desarrollo, no se requiere adaptación, ya que la tarjeta utiliza el lenguaje de programación Lua. Sin embargo, en el caso de ChucK, que es un lenguaje de nicho, un lenguaje de propósito específico, no existe una traducción directa de bloques a lenguaje textual.

En la figura **3-30** se puede observar la arquitectura de los módulos correspondientes al código Open Source del framework Blockly. Entre los principales que deben intervenirse se encuentran los directorios *generators*

[21] Sitio oficial de Blockly: `https://github.com/google/blockly`
[22] Sitio oficial de Codemirror: `https://codemirror.net/`

y *blocks*. El directorio *generators* es responsable de las traducciones a los diferentes lenguajes. En este caso, se trata de agregar las traducciones de los bloques al lenguaje ChucK. En el código 3.10 se pueden observar los módulos que fueron traducidos desde los bloques al lenguaje ChucK, como es el caso de procesos, listas, funciones matemáticas, variables, operaciones lógicas, texto, entre otras.

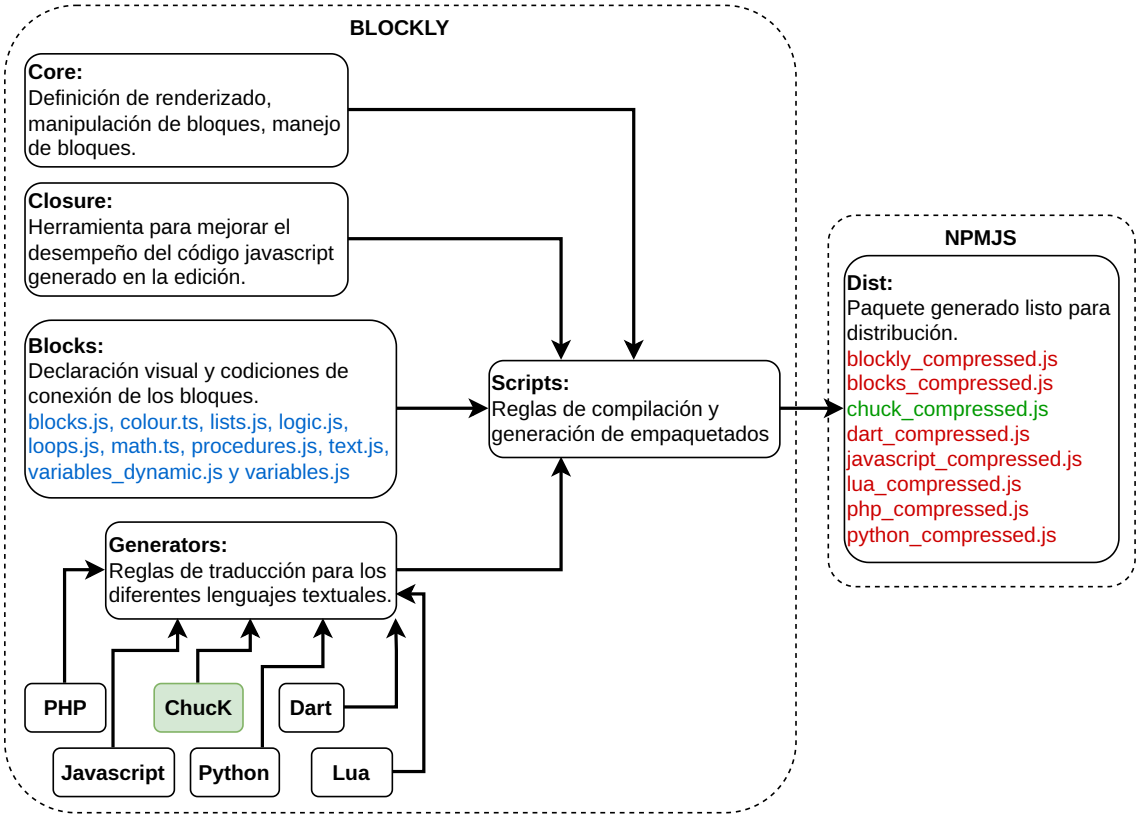


Figura 3-30: Módulos principales de *Blockly* que intervienen en el desarrollo del framwork.

```
/**
 * @license
 * Copyright 2021 Google LLC
 * SPDX-License-Identifier: Apache-2.0
 */

/**
 * @fileoverview Complete helper functions for generating Chuck for
 *     blocks.  This is the entrypoint for dart_compressed.js.
 * @suppress {extraRequire}
 */
'use strict';

goog.module('Blockly.Chuck.all');

const moduleExports = goog.require('Blockly.Chuck');
// goog.require('Blockly.Chuck.colour');
goog.require('Blockly.Chuck.lists');
goog.require('Blockly.Chuck.logic');
goog.require('Blockly.Chuck.loops');
goog.require('Blockly.Chuck.math');
goog.require('Blockly.Chuck.procedures');
```

```
goog.require('Blockly.Chuck.texts');
goog.require('Blockly.Chuck.variables');
goog.require('Blockly.Chuck.variablesDynamic');

exports = moduleExports;
```

Listing 3.10: Elementos agregados en el "generators" de blockly para la traducción al lenaguaje textual ChucK

El directorio *blocks*, que es responsable de la visualización y de las reglas de conexión entre bloques, también es editado. En este caso, debido a que ChucK es un lenguaje tipado, se requiere agregar a los bloques una instrucción que indique los diferentes tipos con los que pueden operar. En el código 3.11, se observan los lugares que deben ser editados para agregar las opciones de tipado que pueden tener los diferentes bloques. Finalmente, con los archivos fuente editados, se hace uso de las reglas de compilación que trae el framework de Blockly, con el cual, se genera un archivo comprimido que representa el framework para usar en el IDE. Para facilitar el flujo de desarrollo, se hace uso de un repositorio de compilaciones denominado NPM[23], el cual permite agregar en un proyecto la compilación de Blockly con los respectivos ajustes.

```
// Block for 'for_chuck' loop.
{
'type': 'controls_for_chuck',
'message0': '%{BKY_CONTROLS_FOR_TITLE}',
'args0': [
  {
    'type': 'input_value',
    'name': 'VAR',
    'check': ['Number', 'int'],
  },
  {
    'type': 'input_value',
    'name': 'FROM',
    'check': ['Number', 'int'],
    'align': 'RIGHT',
  },
  {
    'type': 'input_value',
    'name': 'TO',
    'check': ['Number', 'int'],
    'align': 'RIGHT',
  },
  {
    'type': 'input_value',
    'name': 'BY',
    'check': ['Number', 'int'],
    'align': 'RIGHT',
  },
],
'message1': '%{BKY_CONTROLS_REPEAT_INPUT_DO} %1',
'args1': [{
  'type': 'input_statement',
  'name': 'DO',
}],
'inputsInline': true,
'previousStatement': null,
'nextStatement': null,
'style': 'loop_blocks',
'helpUrl': '%{BKY_CONTROLS_FOR_HELPURL}',
```

[23]Sitio oficial de NPM: `https://www.npmjs.com/`

```
'extensions': [
  'contextMenu_newGetVariableBlock',
  'controls_for_tooltip',
],
},
```

Listing 3.11: Ejemplo en JavaScript sobre la modificación de bloques de *Blockly* para agregar la funcionalidad de revisión de tipo “int”

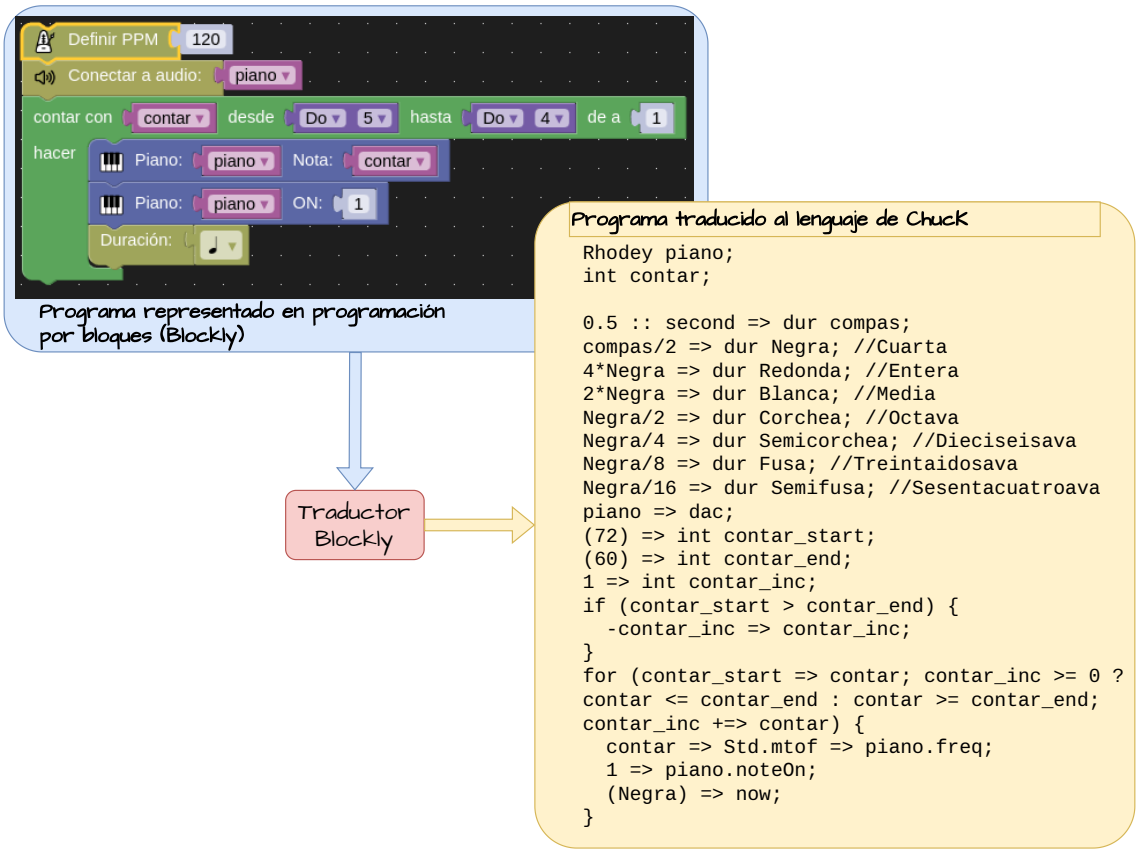


Figura 3-31: Representación del proceso de traducción de un programa descrito en Blockly al lenguaje textual de ChucK

En el proceso de modificación de Blockly mencionado anteriormente, se realizó la configuración o adaptación del lenguaje ChucK para los propósitos generales de traducción. Como ChucK es un lenguaje orientado a la síntesis de sonidos y la generación de música, se requiere crear bloques adicionales con propósitos específicos. Para tal propósito, se hace uso de la herramienta Blockly llamada *Blockly Factory*, la cual permite construir bloques a la medida de las necesidades y genera código ya sea en JavaScript o JSON para determinar la traducción al lenguaje específico que debe generarse.

Una vez generados los archivos de traducción de bloques a ChucK, se hace uso de una herramienta denominada *Closure Compiler*[24] para generar un compilado en JavaScript de los diferentes bloques de traducción. El resultado final de este compilado es agregado al IDE para que pueda hacer uso de los bloques construidos para el propósito específico. En la figura **3-31** se muestra el proceso de traducción con los bloques agregados a Blockly, donde se han agregado diferentes bloques y son traducidos a su correspondiente código textual de ChucK.

[24]Sitio oficial de Closure: `https://github.com/google/closure-compiler`

Visualizador de documentación y gestor de proyectos

Tanto el visualizador de documentación como el gestor de proyectos utilizan las APIs proporcionadas por NWJS a través del backend diseñado. En el caso del visualizador de documentación, se aprovecha el servicio de Civetweb mediante la apertura de una ventana emergente que redirige a una URL del directorio que contiene la documentación deseada. Por otro lado, en el caso del gestor de proyectos, se accede a la API responsable del manejo de archivos en el sistema operativo, y se genera un archivo JSON que contiene las instrucciones o el código que representa los diferentes buffers del editor, ya sea para Blockly o para Codemirror.

Diseño de la interfaz de usuario

El requerimiento principal para la interfaz de usuario es la capacidad de realizar operaciones sencillas y guiadas, en vista del tipo de usuario inexperto al que está dirigido.

Una interfaz de usuario bien diseñada, orientada a la experiencia del usuario, sus necesidades y los recursos gráficos visuales adecuados, facilita la interacción y contribuye al éxito de la aplicación. Por el contrario, una interfaz deficiente puede resultar en la frustración del usuario e incluso llevarlo a buscar alternativas más adecuadas a sus necesidades. Algunas características importantes en el desarrollo de una interfaz de usuario incluyen la usabilidad, la claridad en la navegación, el feedback visual, la consistencia, el atractivo visual, la estabilidad y el rendimiento.

Para iniciar con el proceso de diseño de la interfaz de usuario, se requiere realizar un bosquejo (MockUp) de las diferentes pantallas con las que interactúa el usuario durante el uso de la aplicación. Este bosquejo proporciona una guía visual para el diseño y puede consultarse en el anexo **C**.

Una vez definido el comportamiento y las pantallas de la interfaz de usuario, se procede a seleccionar una tecnología para su implementación. En la Tabla **3-9** se presenta una comparativa entre diferentes tecnologías basadas en diseño web, considerando criterios como la curva de aprendizaje, la arquitectura, el rendimiento, el tamaño y el lenguaje de programación principal.

Tabla 3-9: Comparativa de tecnologías web para el diseño de interfaces gráficas

Framework	**Lenguaje de programación**	**Curva de aprendizaje**	**Flexibilidad**	**Arquitectura**	**Rendimiento**	**Tamaño**
Reat.js	JavaScript	Moderada	Alta	Componentes	Bueno	Mediano
Vue 3	JavaScript/-TypeScript	Baja	Alta	Componentes	Excelente	Pequeño
Angular JS	JavaScript/-TypeScript	Alta	Baja	MVVM	Bueno	Grande
Vanilla	JavaScript	Baja	Alta	N/A	Excelente	Muy Pequeño

Para el diseño de la interfaz de usuario en este proyecto, se ha seleccionado la tecnología *Vue 3*[25]. Esta elección se basa en la baja curva de aprendizaje requerida con respecto a las otras tecnologías, la orientación a componentes que facilita la reutilización del código y su tamaño reducido. En la figura **3-32** se muestra una representación abstracta de los componentes diseñados para la interfaz gráfica de usuario, los cuales están directamente relacionados con el bosquejo presentado en el anexo **C**.

[25]Sitio oficial de Vue: `https://vuejs.org/`

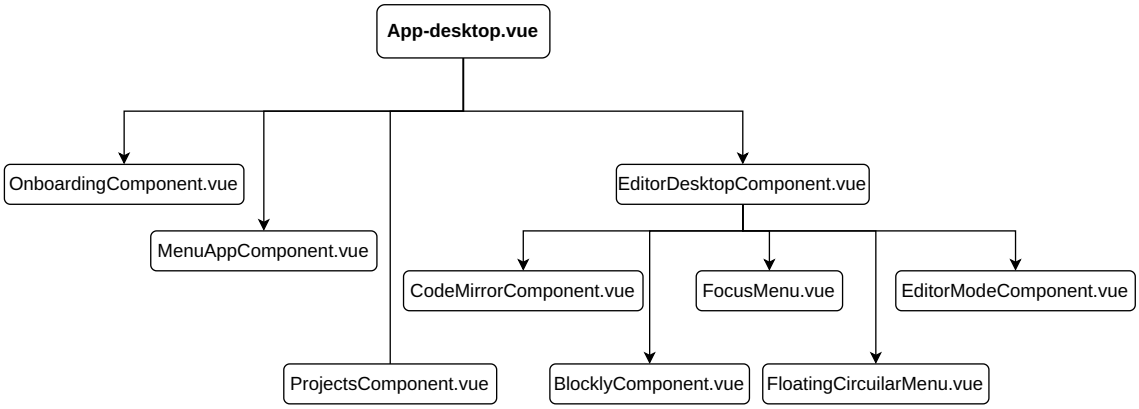


Figura 3-32: Estructura de los componentes diseñados en la tecnología *Vue 3* para el desarrollo de la interfaz de usuario

3.4.4 Implementación del IDE Catalejo Editor

Después de realizar el diseño y la selección de las diferentes tecnologías para cada uno de los módulos que conforman el entorno de desarrollo integrado, se procede a mostrar la implementación del entorno de desarrollo integrado (IDE) anexo **C**, el nombre propuesto para este programa, *Catalejo Editor*, es adoptado de aplicaciones que se han desarrollado para anteriores experiencias de aprendizaje, donde el nombre se relaciona con la posibilidad de usar una herramienta que permita ver/conocer/explorar un poco más allá y determinar rutas de aprendizaje pertinentes.

En la Figura **3-33** se presenta la pantalla de bienvenida de la aplicación *Catalejo Editor*, donde se puede apreciar un botón de aceptar junto con un mensaje de bienvenida. Posteriormente, en la Figura **3-34** se muestra la pantalla que aparece después de presionar el botón de aceptar en la pantalla de bienvenida. En esta pantalla se presentan tres opciones: la primera opción está relacionada con las posibilidades de edición de proyectos con *Catalejo Editor*, mientras que las otras dos opciones están relacionadas con la revisión de la documentación tanto de ChucK como para NodeMCU.

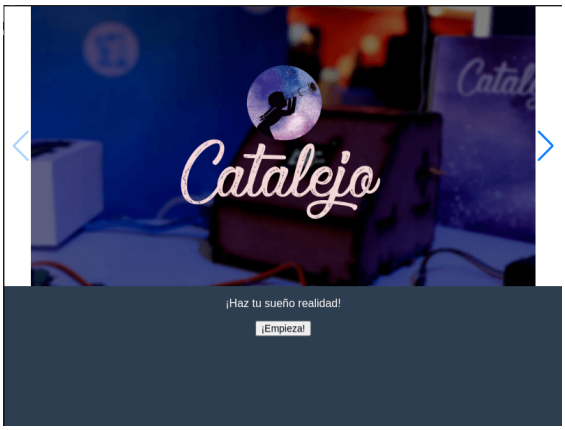


Figura 3-33: Pantalla de presentación de la aplicación Catalejo Editor

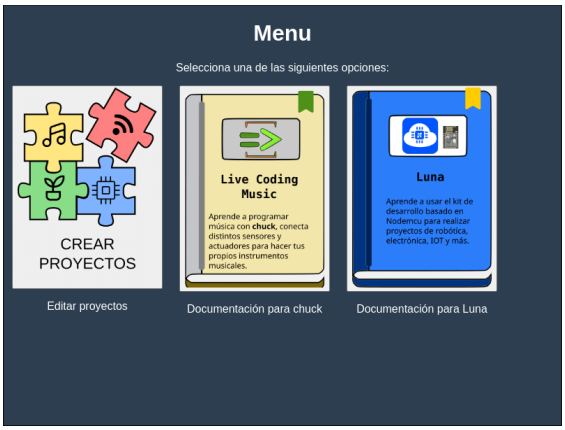


Figura 3-34: Pantalla de menú de los servicios ofrecidos por Catalejo Editor

La pantalla representada en la Figura **3-35** muestra las distintas opciones disponibles para la creación o

edición de proyectos. Estas incluyen la posibilidad de crear un proyecto para NodeMCU, uno para ChucK, y también la opción de abrir un proyecto previamente creado con Catalejo Editor, con la finalidad de continuar con el proceso de elaboración. En la Figura **3-36** se ilustra que al seleccionar alguna de las opciones de documentación en el menú de la pantalla, se despliega una ventana emergente que enlaza una URL con la documentación correspondiente que el usuario desea consultar en ese momento.

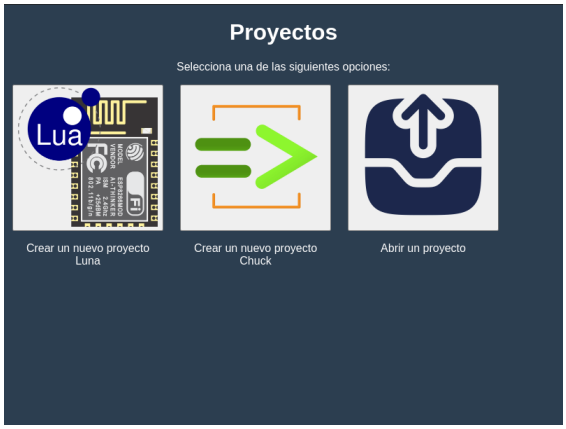


Figura 3-35: Pantalla de proyecto de Catalejo Editor

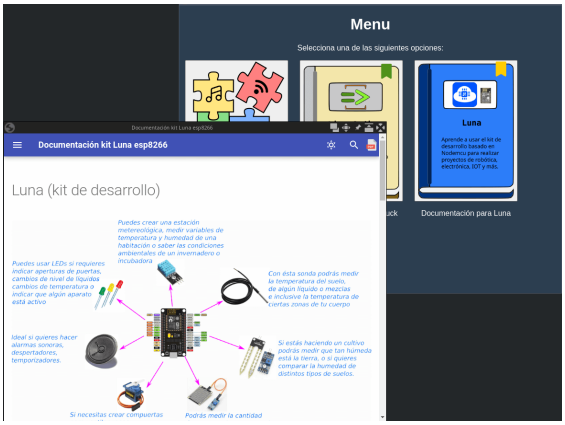


Figura 3-36: Lanzamiento de la documentación en tecnología web de Catalejo Editor

La pantalla representada en la Figura **3-37** presenta las distintas opciones disponibles para el usuario abrir un proyecto previamente creado con Catalejo Editor. Una vez que se carga el archivo correspondiente al formato JSON de almacenamiento, la aplicación carga el editor con las herramientas necesarias según el entorno de programación seleccionado, ya sea para ChucK o para NodeMCU. En la Figura **3-38** se muestra la pantalla del editor de texto, la cual ha sido cargada con un programa previamente escrito utilizando Catalejo Editor. Aquí se pueden visualizar los bloques que han sido configurados para un proyecto anterior y que aún pueden ser editados en este punto.

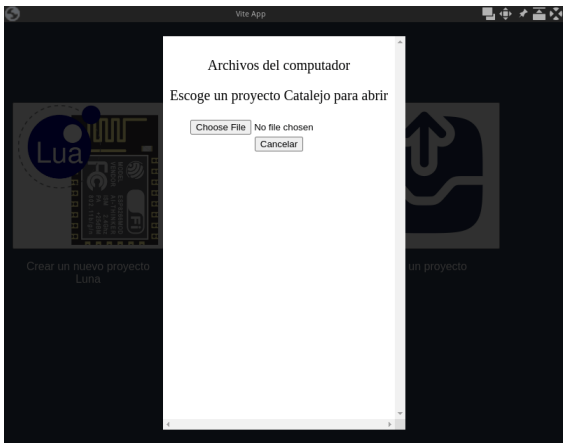


Figura 3-37: Pantalla de selección de proyectos a editar con Catalejo Editor

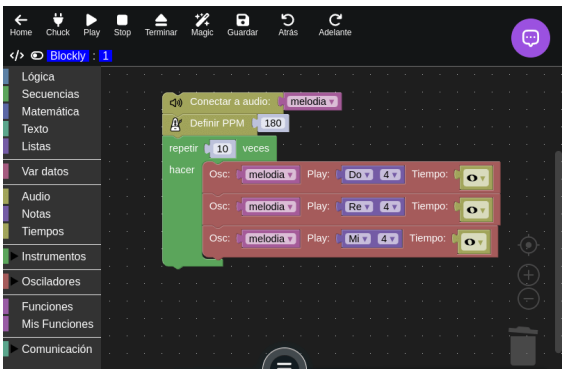


Figura 3-38: Pantalla de presentación del editor de código por medio de *Blockly* de Catalejo Editor

En la pantalla que se muestra en la Figura **3-39**, se puede apreciar una opción disponible en *Catalejo Editor*, la cual permite interactuar con el sistema de registro de operaciones realizadas por el editor. Esto resulta útil, por ejemplo, cuando se ejecutan programas y es necesario verificar que el intérprete está llevando a cabo los procesos de manera correcta. En la Figura **3-40**, se presenta la caja de herramientas asociada a Blockly

dentro del entorno de ChucK. Este mismo principio se aplica a NodeMCU, donde la caja de herramientas se adapta al entorno específico.

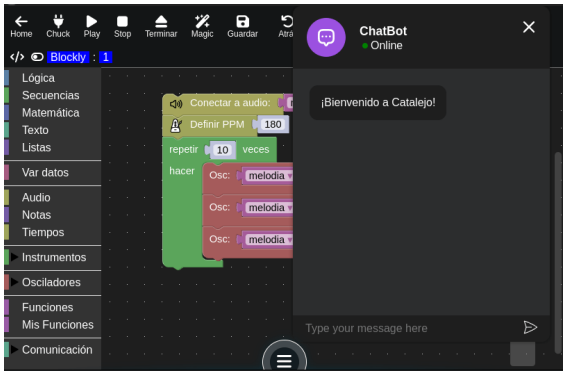


Figura 3-39: Chat de mensajes para registro de eventos de los procesos de Catalejo Editor

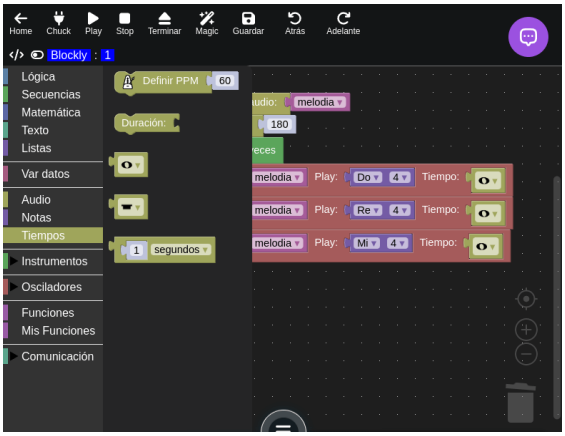


Figura 3-40: Herramientas de *Blockly* disponibles para la traducción textual de Catalejo Editor

En la Figura **3-41**, se exhibe la pantalla correspondiente al menú de acceso rápido que proporciona diferentes funciones del editor. Por otro lado, en la Figura **3-42**, se presenta el editor de texto basado en *Codemirror*. En esta imagen, se observa la creación de código mediante el traductor de Blockly, el cual se ha integrado en el editor de texto mediante el uso del botón *Magic*.

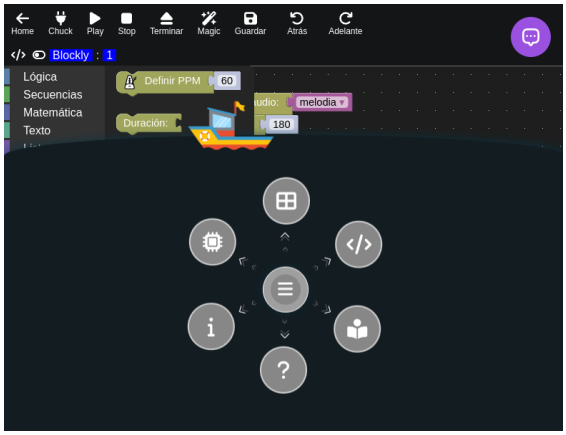


Figura 3-41: Menú flotante para acceso directo a otros servicios de Catalejo Editor

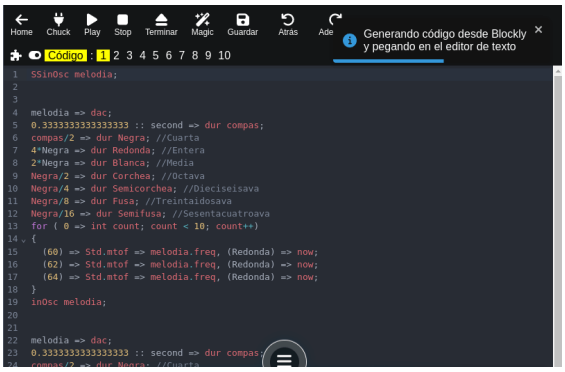


Figura 3-42: Editor de texto basado en *CodeMirror* de Catalejo Editor

Finalmente, la Figura **3-43** muestra la pantalla vinculada a la configuración de la conexión de los servicios de *ChucK*, tanto localmente, es decir, en el mismo equipo, como de forma remota, hacia un servicio de *ChucK* alojado en otro equipo.

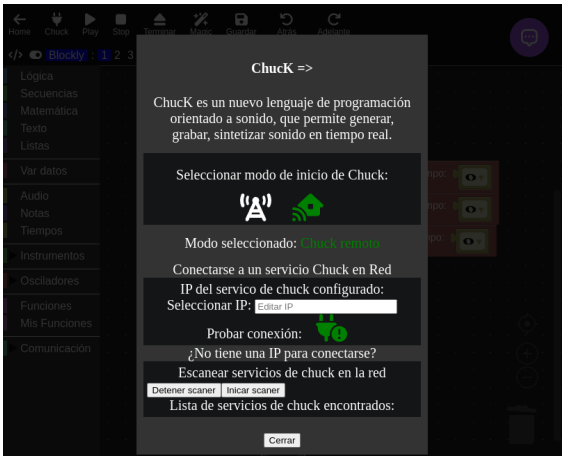


Figura 3-43: Inicio de servicios de conexión de *ChucK* remoto de Catalejo Editor

3.5 Diseño de la documentación para el material concreto

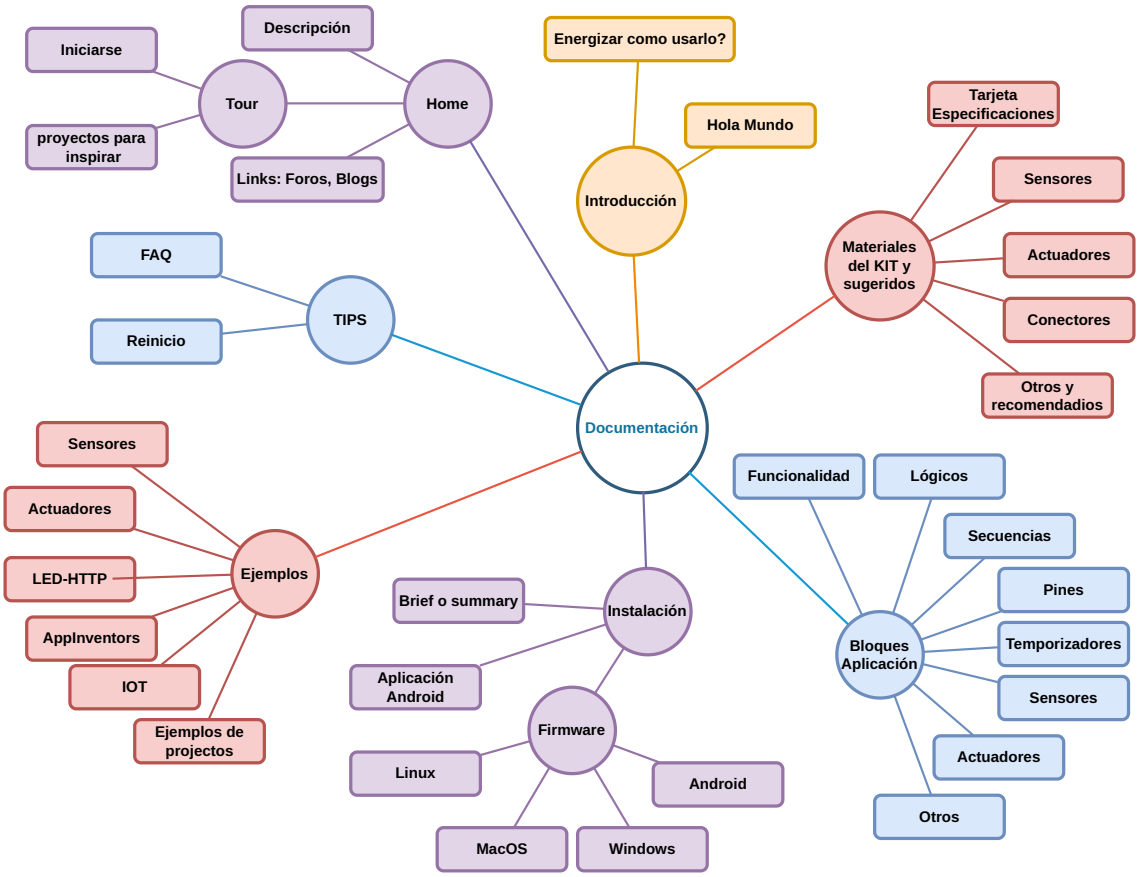


Figura 3-44: Grafo que representa los temas del material concreto

Para el diseño de la documentación del material concreto, es esencial considerar tanto el contenido como la presentación. El contenido debe ser accesible para diversos niveles de comprensión, ya que los usuarios pueden tener diferentes niveles de familiaridad con estas herramientas, incluyendo principiantes, intermedios y avanzados. Además, la documentación debe incluir una referencia técnica para proporcionar orientación sobre aspectos no cubiertos directamente.

La documentación no solo debe ofrecer información básica, sino también servir como un punto de partida para acceder a recursos adicionales que enriquezcan el aprendizaje con el material. En términos de presentación, es fundamental utilizar una variedad de recursos para facilitar la comprensión de conceptos, actividades e implementaciones, promoviendo así el desarrollo de habilidades a través de métodos de autoaprendizaje.

En cuanto al contenido esencial, la figura **3-44** ilustra las áreas temáticas que deben cubrirse en la documentación del material concreto:

1. **Instalación:** Se documenta el proceso de instalación de la aplicación Catalejo Editor en diferentes sistemas operativos, así como la actualización del firmware de la tarjeta de desarrollo.

2. **Tips de uso:** Se proporcionan referencias sobre el reinicio de problemas asociados a la tarjeta de desarrollo y se abordan preguntas frecuentes relacionadas con su uso.

3. **Inicio:** Se describe el material concreto y se proporcionan enlaces a comunidades o documentación adicional, como foros, blogs y tutoriales de inicio, además de proyectos inspiradores.

4. **Introducción:** Se detalla el proceso de programación y creación de artefactos utilizando el kit, con ejemplos prácticos como ”Hola Mundo” desde la programación y la electrónica, junto con instrucciones sobre cómo energizar el dispositivo.

5. **Ejemplos:** Se presentan diversas formas de utilizar el material concreto con ejemplos simples que abarcan sensores, actuadores, tecnologías de comunicación y proyectos estimulantes que los usuarios pueden imitar y modificar para ampliar sus habilidades.

6. **Uso del IDE:** Se enfatizan los recursos proporcionados por la interfaz de desarrollo integrada para la programación y traducción de lenguajes, incluyendo temas como funciones, operaciones lógicas, manejo de pines, temporizadores y otros procesos relacionados con la programación.

Al seguir esta estructura y enfoque, se puede crear una documentación completa y efectiva que satisfaga las necesidades de los usuarios en diferentes niveles de experiencia y conocimiento.

3.5.1 Herramientas de documentación

Para atender estas necesidades relacionadas con el contenido y la presentación, es esencial emplear tecnologías que faciliten el mantenimiento y la actualización de la documentación. Este proceso es crucial para corregir errores y mantener la calidad del material. En la Tabla **3-10** se presenta una revisión de tecnologías web de código abierto populares que están diseñadas para la creación sencilla de contenido mediante lenguajes de marcado. Estas tecnologías permiten exportar el contenido a diversos formatos y plataformas, como recursos en línea, documentación portátil en PDF y otros formatos integrables en aplicaciones.

Tabla 3-10: Comparación de tecnologías de documentación opensource según el formato de entrada y salida

Tecnología de documentación	Principal formato de entrada	Principales formatos de salida
Material para MKDocs	Markdown	HTML, PDF (con plugins adicionales)
reStructuredText	Markdown	HTML, LaTeX, PDF, ePub, OpenDocument, Jira, MediaWiki
Pandoc	Varios (incluye Markdown)	HTML, LaTeX, PDF, ePub, OpenDocument, MediaWiki, Markdown
Hugo	Markdown	HTML, JSON, RSS, XML, AMP, CSV, ePub, OpenDocument
MDBook	Markdown	HTML, ePub, PDF

Selección de herramienta de documentación

Entre las tecnologías evaluadas, se ha seleccionado *Material for MkDocs*[26], que, mediante plugins, posibilita la exportación a múltiples formatos de salida, como HTML y PDF. *Material for MkDocs* aprovecha Markdown, un lenguaje de marcado ligero, lo que simplifica el proceso de escritura al ofrecer reglas simples para construir diversos elementos que, en otros lenguajes, requerirían una sintaxis más compleja. Además, *Material for MkDocs* facilita la interacción con la documentación gracias a que en su proceso de compilación, genera como resultado una interfaz de usuario basada en HTML, incluye un buscador de palabras y coincidencias, así como una tabla de contenido accesible en todo momento.

Finalmente, en la Figura **3-45** se presentan las características implementadas en la documentación del material concreto. En ella se pueden observar los diferentes elementos abordados en este apartado, el formato de salida en PDF el cual está referenciado en el anexo **D**, fundamentales para el proceso de documentación del kit de materiales tanto online como offline.

Figura 3-45: Vista de la documentación implementada en *Material for MkDocs*

[26]Sitio oficial de Material of MkDocs: `https://squidfunk.github.io/mkdocs-material/`

Documentación de montajes eléctricos

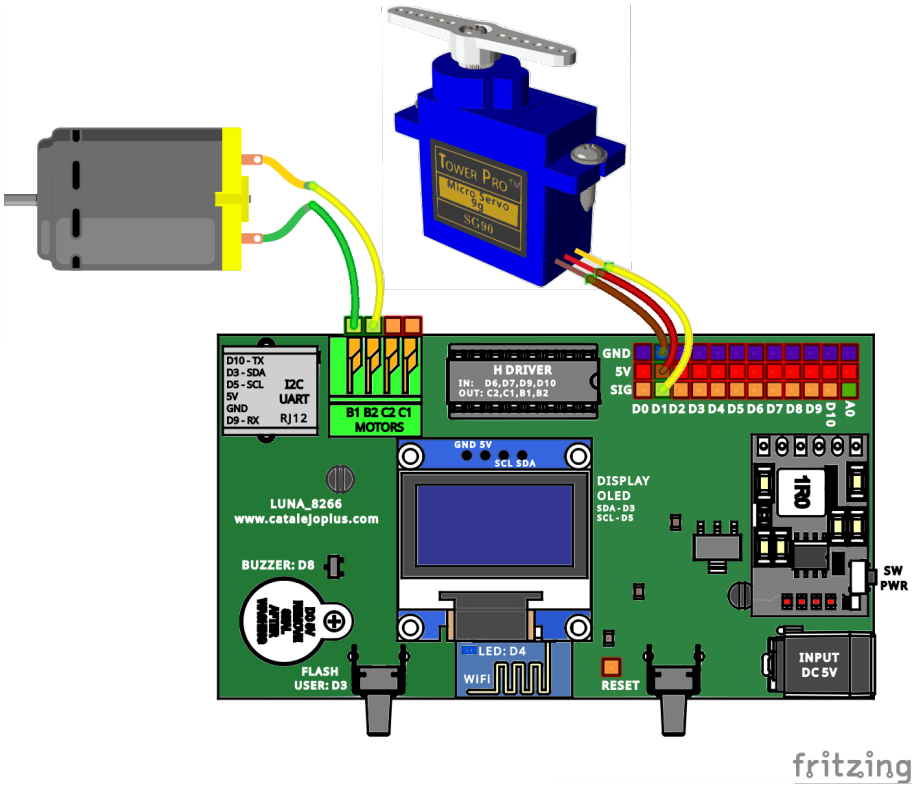


Figura 3-46: Diagrama pictográfico de la tarjeta de desarrollo diseñado en *Fritzing*

Para la realización de los montajes propuestos, los cuales los estudiantes en principio imitarán, se requiere hacer uso de diagramas pictográficos, los cuales son aproximaciones de esquemas de conexión que no requieren un nivel de abstracción mayor que los requeridos en esquemas con símbolos electrónicos. Para este caso, la tarjeta de desarrollo fue exportada a *Fritzing*[27], lo cual, facilita la conexión con diferentes dispositivos electrónicos de la línea educativa. En la figura **3-46** se aprecia un ejemplo de documentación de un montaje entre la tarjeta de desarrollo, un servomotor y un motor DC.

3.6 Presentación de material concreto

Se ha desarrollado un material didáctico que combina tanto elementos de hardware como de software. En este capítulo, se detalló el diseño de cada uno de los módulos y la implementación utilizando tecnologías compatibles con la filosofía *copyleft*. La presentación de este material al usuario final es de suma importancia, por esta razón, el material concreto se ha organizado en una caja de materiales, más específicamente, en un kit que incluye una variedad de elementos como sensores, actuadores y una tarjeta de desarrollo.

Este kit se presenta en dos versiones: una fabricada mediante corte láser (ver Figura **3-47**) y la otra elaborada con una caja plástica (ver Figura **3-48**). Ambas versiones están acompañadas por un logo que incorpora información en forma de código QR. Este código permite acceder a la aplicación Catalejo Editor diseñada

[27] Sitio oficial de Fritzing `https://fritzing.org/`

para controlar el kit, así como a documentación en línea que proporciona al usuario todas las herramientas necesarias para utilizar el kit de manera efectiva.

Es importante destacar que la función principal de este kit es facilitar experiencias de aprendizaje en programación, con un enfoque motivador centrado en la música.

(a) Vista de la caja diseñada en corte láser con código QR

(b) Vista de los módulos internos con los componentes del kit

Figura 3-47: Presentación del kit de materiales en caja diseñada en 2D implementada en MDF con corte láser

(a) Vista superior con código QR para acceso a la documentación

(b) Vista desde la plataforma superior del kit

(c) Vista interna inferior

Figura 3-48: Presentación del kit de materiales en caja plástica

4 Experiencia de aprendizaje

Figura 4-1: Estudiantes, interesados y aliados de la implementación del curso de programación desde la música

Para la implementación de la metodología de aprendizaje, se realizó una búsqueda de posibles colaboradores y aliados que pudiesen facilitar los recursos requeridos para la adquisición de los kits de materiales, entre otros materiales didácticos necesarios. A través de personas que hacen parte de la Universidad Nacional de Colombia se realizó un proceso de divulgación que dio como resultado, que personas del municipio de Prado, ubicado en el departamento del Tolima se comprometieran con la donación de los recursos necesarios para la implementación de la experiencia de aprendizaje en el municipio.

La experiencia de aprendizaje se diseñó teniendo presente las características y condiciones localizadas de la población, lo que planteó la elaboración de sesiones híbridas, combinando encuentros síncronos y asíncronos aprovechando los medios virtuales disponibles como también realizar una visita a uno de los colegios del municipio. Para el apoyo asíncrono se consideró el uso de una plataforma de mensajería instantánea, la cual permite el acompañamiento cercano y continuo a los estudiantes a lo largo del proceso.

En la actividad presencial se hizo entrega de los kits de materiales a los participantes, con ello, se tuvo la oportunidad de explorar el contenido y participar de las actividades de programación de electrónica y de música. Con respecto a las actividades síncronas remotas se hizo uso principalmente del IDE Catalejo Editor, la aplicación desempeñó un papel central en el desarrollo de las actividades, con la cual, se desarrollaron actividades prácticas sobre los conceptos y habilidades relacionadas a la programación.

En la figura **4-1** se puede ver los diferentes actores de esta experiencia de aprendizaje en los cuales están los

aliados, los estudiantes, los profesores del colegio y el mediador de la experiencia de aprendizaje.

En resumen la implementación de la metodología fue posible gracias a la colaboración de los diferentes actores, quiénes facilitaron los recursos. La combinación de las de las actividades presenciales y virtuales junto al uso de la herramienta tecnológica contribuyó a una experiencia de aprendizaje integral y motivadora para los participantes.

4.1 Población

Figura 4-2: Ubicación de los colegios Luis Felipe Pinto, sector urbano y Colegio José Celestino Mutis, sector rural

La población que hizo parte de la experiencia de aprendizaje está constituida por estudiantes que son procedentes de dos instituciones educativas, la primera de ellas es la institución educativa Luis Felipe Pinto, la cual se encuentra en la zona urbana, esta es una institución oficial con matrícula mixta, en esta experiencia de aprendizaje participaron alrededor de 20 estudiantes que fueron sugeridos por los profesores del plantel. La segunda institución que participó fue el colegio José Celestino Mutis, la cual está ubicada en la vereda de Montoso sector rural, también es de matrícula mixta y de allí participaron 15 estudiantes en la experiencia de aprendizaje. En ambos colegios los estudiantes participantes estuvieron distribuidos en los grados de sexto a once.

Ambos colegios están situados en el municipio de Prado, en el departamento del Tolima (ver figura **4-2**). El colegio urbano dispone de acceso a Internet y de herramientas para la realización de las actividades virtuales, algunos estudiantes cuentan con acceso desde sus hogares. Por otro lado, el colegio rural aunque tiene Internet presenta interferencias e interrupciones en el acceso del mismo, algunos estudiantes para participar de las actividades de programación adquirieron paquetes de datos de manera individual.

Es importante mencionar que el colegio rural se encuentra a una distancia considerable del casco urbano, 3 horas de viaje en motocicleta. Esta situación requirió de un esfuerzo adicional por parte de los estudiantes, ya que la actividad presencial fue desarrollada en el colegio urbano en horas de la mañana.

4.2 Material didáctico usado en la experiencia de aprendizaje

Figura 4-3: Kits de material concreto armados para la experiencia de aprendizaje

El material didáctico utilizado en la experiencia de aprendizaje se compone de varios elementos esenciales: Está el material concreto, el entorno de desarrollo integrado (Catalejo Editor), y las guías de actividades tanto síncronas como asíncronas. Considerando las posibilidades y los recursos que fueron conseguidos a través de los aliados se hizo la adquisición del material necesario para la creación de 20 kits completos (ver Figura **4-3**). Estos kits se distribuyeron entre estudiantes y profesores participantes de la experiencia. Para fomentar el trabajo en equipo y colaborativo a los estudiantes se les asignó el material por parejas, teniendo la oportunidad de llevar el kit a sus hogares donde podrían realizar las actividades diseñadas para promover el aprendizaje práctico y el autoaprendizaje. Posterior a esto, los estudiantes podían compartir con sus compañeros los resultados de las experiencias con el kit lo cual enriquece el proceso de aprendizaje a través del intercambio de ideas y perspectivas.

4.3 Evidencia de las actividades realizadas

En el marco de las actividades de programación en primera instancia se hizo uso de sesiones síncronas, en la figura **4-4** se representa las sesiones donde los estudiantes se encuentran en el aula tanto aquellos pertenecientes al colegio rural como del colegio urbano, donde los estudiantes están acompañados por sus respectivos profesores. La interacción entre los estudiantes y el mediador es a través de una videollamada, el mediador proporciona orientaciones sobre las actividades a realizar, el rol del mediador no es garantizar el seguimiento adecuado en las tareas, sino que facilitar el proceso de retroalimentación de las actividades y así auspiciar un aprendizaje enriquecido y colaborativo.

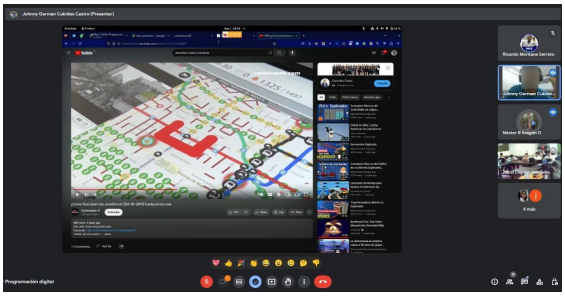
Sabiendo que la experiencia de aprendizaje tiene un enfoque híbrido con actividades tanto síncronas como asíncronas de forma remota donde la lógica de “pensar con las manos” y el autoaprendizaje son fundamentales para el desarrollo de la experiencia, el hecho de no tener el material concreto mientras se consiguen los recursos y se hace el armado de los kits necesarios para las actividades de cacharreo, se convierte en una

dificultad para el proceso de aprendizaje, sin embargo, el diseño de actividades de contingencia a través de Catalejo Editor, permitió que los estudiantes fueran partícipes de las actividades de aprendizaje, inclusive en aquellas donde el material concreto es clave.

(a) Primera participación de estudiantes del colegio rural

(b) Primera participación de estudiantes del colegio urbano

(c) Actividad de presentación del curso



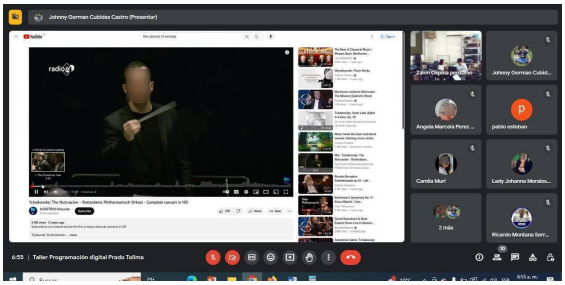
(d) Actividad de programación, identificando algoritmos del entorno

Figura 4-4: Sesiones síncronas con participación de estudiantes en el aula

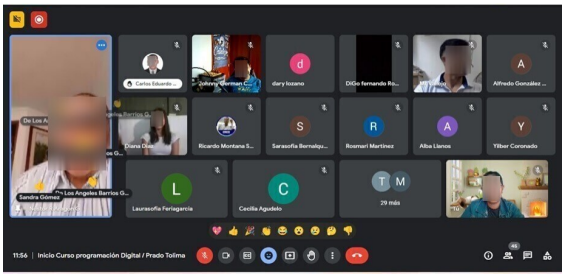
En la figura **4-5** se puede apreciar las invitaciones a los encuentros síncronos, así como diversos ejercicios relacionados con la programación y música. Se destaca también la participación de los aliados que han contribuido significativamente en el enfoque contextualizado de las actividades, aunque el tema central es el desarrollo del pensamiento computacional es importante compartir a los estudiantes las perspectivas y experiencias sobre las decisiones qué influyen en el desarrollo profesional y personal para su presente y futuro.

Como fue mencionado, en el marco de la experiencia de aprendizaje se planteó el desarrollo de un encuentro presencial, el cual es crucial para el desarrollo del proceso educativo. En estas sesiones estudiantes se reunieron con el mediador en las instalaciones del colegio urbano, donde también participaron los estudiantes de colegio rural, ver figura **4-6**. Allí se realizó una orientación detallada sobre cómo usar el kit, se abordó también la documentación asociada y se presentaron ideas de proyectos que se pueden usar con el kit (“cacharros que inspiran”), además, se ofreció a los estudiantes una perspectiva sobre lo que son los paradigmas y un acercamiento a la concepción del conocimiento. Un aspecto fundamental en esta actividad fue el énfasis en el desarrollo de la habilidad de autoaprendizaje, debido a que esta fue la única sesión presencial y que posteriormente el aprendizaje dependería mayormente del trabajo autónomo de los estudiantes, se les motivó a asumir un rol activo en su proceso de aprendizaje, se alentó a los estudiantes en la exploración de las maneras de intervenir el kit e integrarlo con el entorno, la búsqueda de formas de adquirir el conocimiento, fomentando una actitud proactiva hacia el aprendizaje continuo y autodirigido.

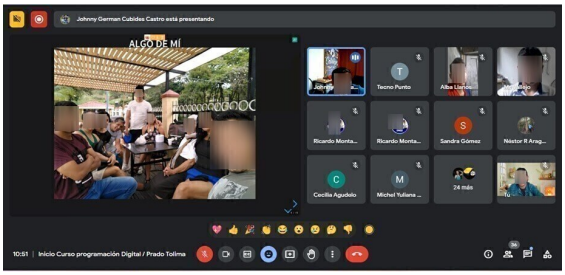
(a) Volantes usados para invitar a los estudiantes a las actividades virtuales



(b) Sesión de actividades relacionadas a la música



(c) Presentación de interesados y aliados de la experiencia de aprendizaje



(d) Sesión de reconocimiento de aula y planes de contingencia

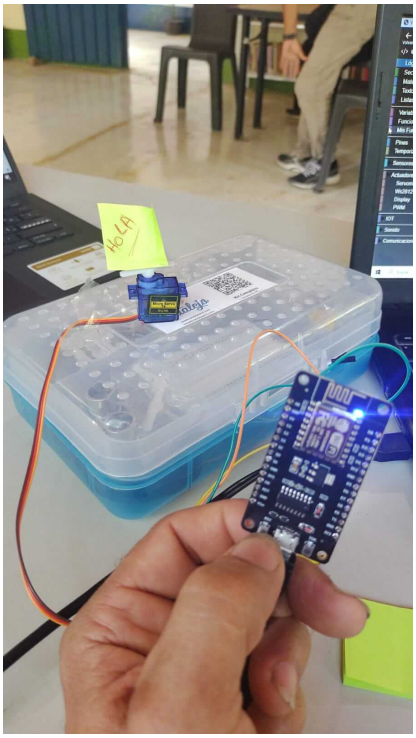
Figura 4-5: Sesiones síncronas virtuales con presencialidad desde casa y colegios

(a) Estudiantes del colegio José Celestino Mutis, sector rural

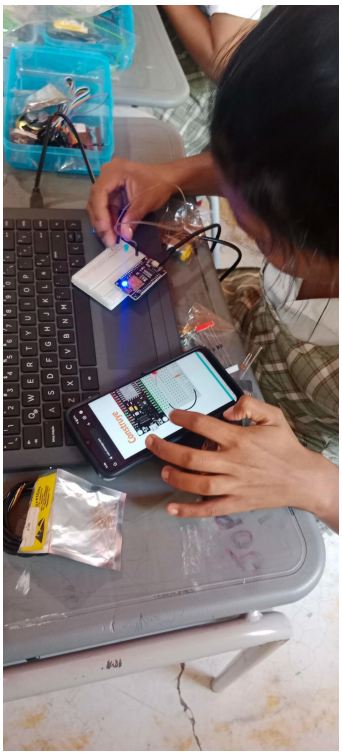
(b) Estudiantes del colegio Luis Felipe Pinto, sector urbano

Figura 4-6: Actividades presenciales en el colegio urbano

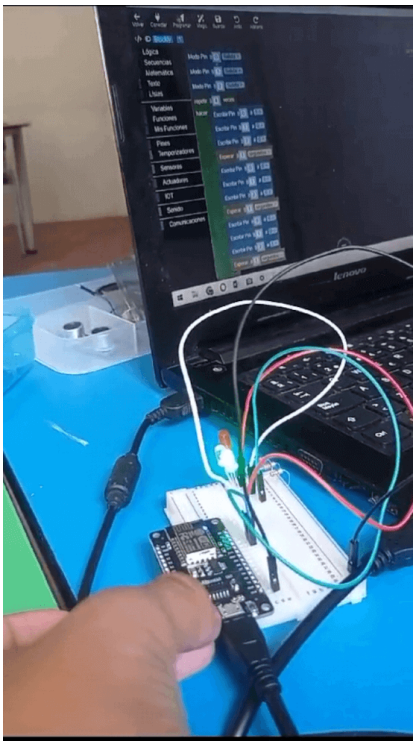
4.4 Resultados de la experiencia de aprendizaje



(a) Prueba de funcionamiento de un servomotor



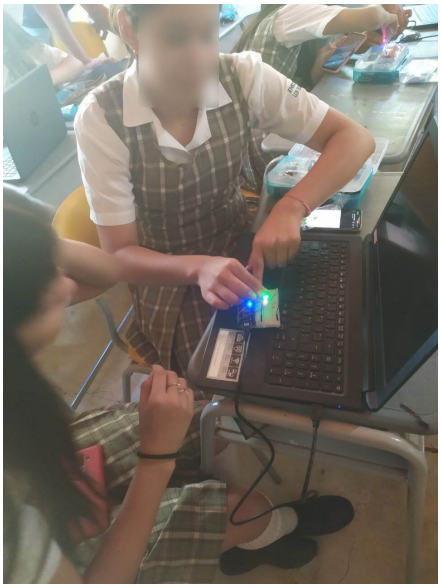
(b) Estudiante revisando las instrucciones del montaje



(c) Estudiante realizando pruebas al circuito después de modificar el código

(d) Estudiantes energizando su circuito

(e) Profesores participando junto a sus estudiantes



(f) Niñas adaptando el circuito al agregar más componentes

Figura 4-7: Estudiantes realizando diferentes montajes con el material concreto.

Durante el transcurso de la experiencia de aprendizaje, los estudiantes se sumergieron en la manipulación de diversos artefactos, utilizando los materiales proporcionados. Siguiendo las pautas y sugerencias detalladas de la documentación y el acompañamiento del mentor, exploraron las posibilidades de construcción y experimentaron con las modificaciones de los artefactos propuestos. Este proceso les permitió comprender la facilidad con la que pueden concebir y materializar sus iniciativas. Al adentrarse en la exploración del kit, los estudiantes tuvieron aproximaciones sobre conceptos de sensores, actuadores, controladores, como también, sobre el rol que cumplen ellos en el desarrollo de proyectos.

En la figura **4-7**, se puede observar algunos artefactos que crearon los estudiantes durante el proceso, como el encendido de LEDs, las aplicaciones a los códigos propuestos y el manejo de servo mecanismos. También se destaca la realización de montajes de circuitos eléctricos a través de diagramas pictográficos lo cual hace parte del proceso de abstracción. Además, en esta figura, se evidencia la participación de los estudiantes en equipos de trabajo, lo cual cultiva las habilidades de comunicación y trabajo en equipo, fundamentales para el éxito de un proyecto o iniciativa problematizada en la realidad.

Los estudiantes expresaron su entusiasmo y motivación compartiendo en redes sociales las actividades y artefactos que desarrollaron, ver Figura **4-8**. En los grupos creados para las actividades asíncronas, continuaron mostrando sus creaciones expresando interés por aprender más sobre la programación, esto refleja su interés genuino por el deseo de aprender.

(a) Estudiantes compartiendo sus apreciaciones de la experiencia

(b) Estudiantes compartiendo su experiencia en redes sociales

(c) Estudiantes del colegio rural de José Celestino Mutis

Figura 4-8: Reacciones de los participantes de la experiencia de aprendizaje

5 Conclusiones y recomendaciones

5.1 Conclusiones

Las conclusiones extraídas del desarrollo de este trabajo de grado generan los siguientes hallazgos:

- El aprendizaje del pensamiento computacional a través de las actividades de programación requiere más que propender por adquirir habilidades técnicas. Es esencial mantener la motivación de los estudiantes mediante la relación que puede existir entre la programación con los aspectos relevantes de la vida real. Por ejemplo, al utilizar la educación musical general como un vehículo motivacional, se puede generar interés sobre el estudio de los temas asociados a la programación, por tanto, la vinculación de la programación en diferentes áreas de interés y ramas del conocimiento, resulta más atractiva en comparación a un enfoque de aprendizaje de la programación meramente en sus aspectos técnicos.

- El uso del material didáctico concreto y la realización de experiencias de aprendizaje prácticas, en el concepto de ”pensar con las manos”, fomenta el desarrollo de habilidades creativas y la resolución de problemas. Estas experiencias permiten a los estudiantes generar conocimiento desde una postura activa, mientras que el mediador cumple un rol de facilitador de procesos de comprensión y reflexión.

- El diseño de herramientas de programación que cuenta con el enfoque planteado para de Catalejo Editor, ofrece diferentes caminos para desarrollar el pensamiento computacional. Este IDE, que permite la ejecución de tareas como la traducción de bloques de programación hacia un lenguaje textual, la programación de tarjetas de desarrollo y la capacidad embeber documentación sobre el uso de estas herramientas, proporciona elementos y funcionalidades diversas para comprender los conceptos de traducción de los lenguajes de alto nivel a bajo nivel, superar los problemas asociados a la curva de aprendizaje de un lenguaje, por ejemplo, su semántica y sintaxis, e imitar programas a través de la documentación embebida para luego ser susceptibles de modificaciones a través de la experimentación. Por tanto, hace que la herramienta sea adaptable a diferentes propósitos y a poblaciones con habilidades diversas y de distintos niveles de comprensión aprovechables para el desarrollo del pensamiento computacional.

- Se hace necesario que las experiencias de aprendizaje aborden aspectos contextuales, como es el caso de la relación entre la ciencia, tecnología y sociedad. Esto permite que los estudiantes comprendan su rol tanto en el proceso educativo como en la sociedad en general; al reconocer cómo sus situaciones y problemáticas locales se relacionan con objetivos globales, los estudiantes pueden emprender una ruta educativa y profesional más consciente e intencionada.

5.2 Recomendaciones

Las siguientes recomendaciones están orientadas a la ejecución de la metodología diseñada y a diferentes modificaciones a realizar al material didáctico:

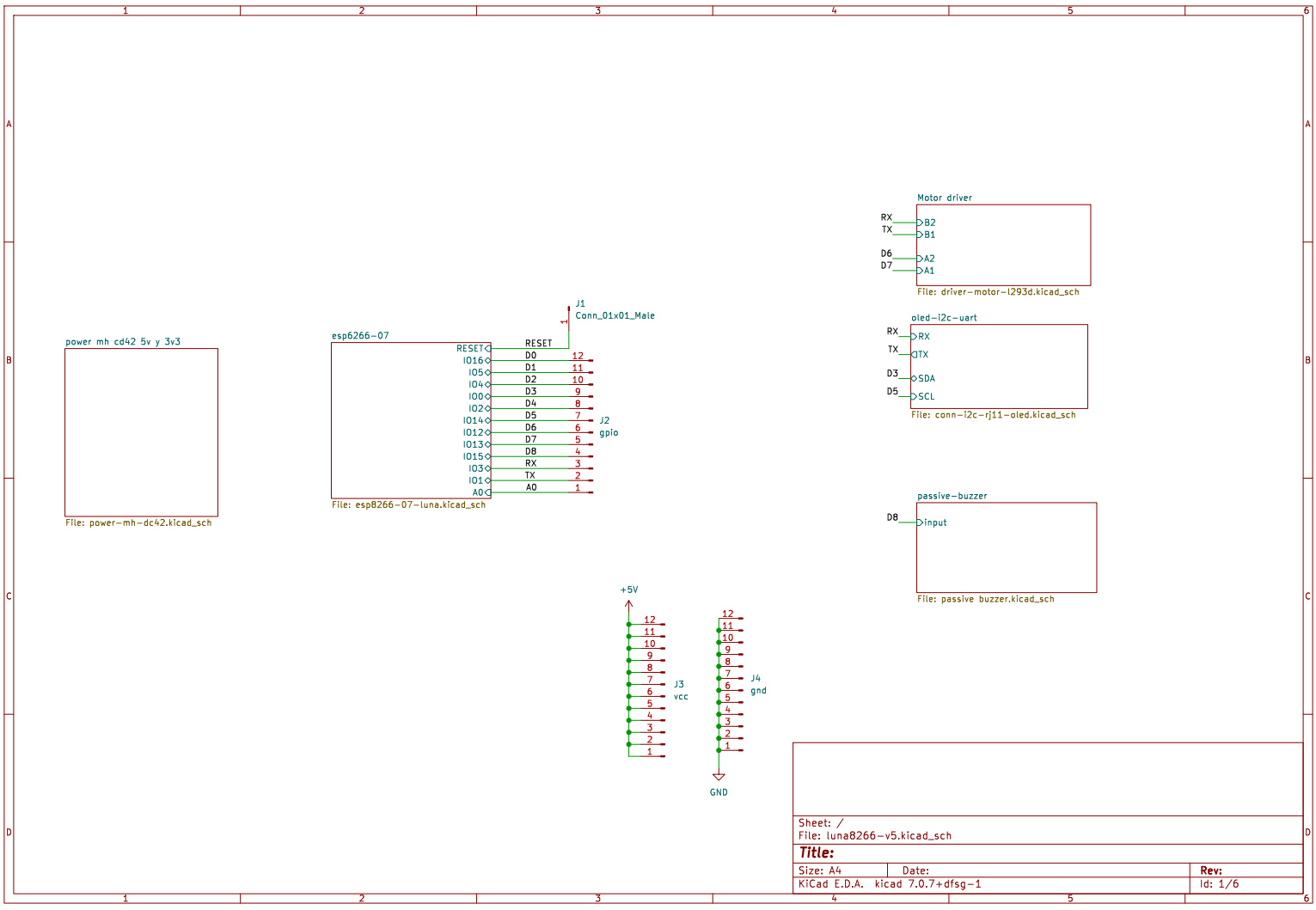
- El material didáctico está diseñado para permitir experiencias de aprendizaje autorregulables. Se recomienda que el banco de actividades se desarrolle teniendo en cuenta las características de la población que va a realizar la experiencia de aprendizaje. El profesor, que desempeña el rol de facilitador o mediador, deberá involucrar los intereses y formas de comprender el mundo de los estudiantes en la planificación de las actividades.
- Para la implementación de una experiencia de aprendizaje en contextos rurales donde haya dificultades para el acceso a Internet y acceso a computadoras, se puede hacer uso de un sistema embebido que pueda contener el material didáctico y el entorno de desarrollo en una lógica de cliente-servidor; el desarrollo presentado en este documento se ha construido bajo tecnologías de código abierto y web que permiten la fácil portabilidad de la herramienta en el sistema propuesto, donde lo único que haría falta es el acceso a la interfaz de usuario a través de dispositivos portátiles que tengan acceso a un navegador web, como puede ser el caso de tabletas y teléfonos celulares.
- Como el material concreto es concebido como una caja de herramientas incompleta donde el estudiante puede hacer las adecuaciones que crea pertinentes, agregando otros componentes, también es posible hacer modificaciones a la tarjeta de desarrollo propuesta. Recordando que este desarrollo es opensource, se pueden realizar modificaciones a la tarjeta de desarrollo haciendo uso de otro SoC que sea capaz de soportar un firmware en el nivel de intérprete (Python y Lua) y que pueda cumplir con otros requisitos para implementaciones futuras.
- El entorno de desarrollo integrado Catalejo Editor, está diseñado para agregarle otras funcionalidades, ya sea en otros lenguajes, otras tarjetas de desarrollo e incluso otros contextos de programación. Se invita a integrar otras herramientas, como por ejemplo hacer uso de Love2D para añadir a la experiencia la posibilidad de desarrollo de videojuegos o de expresiones artísticas musicales que requieran efectos visuales.

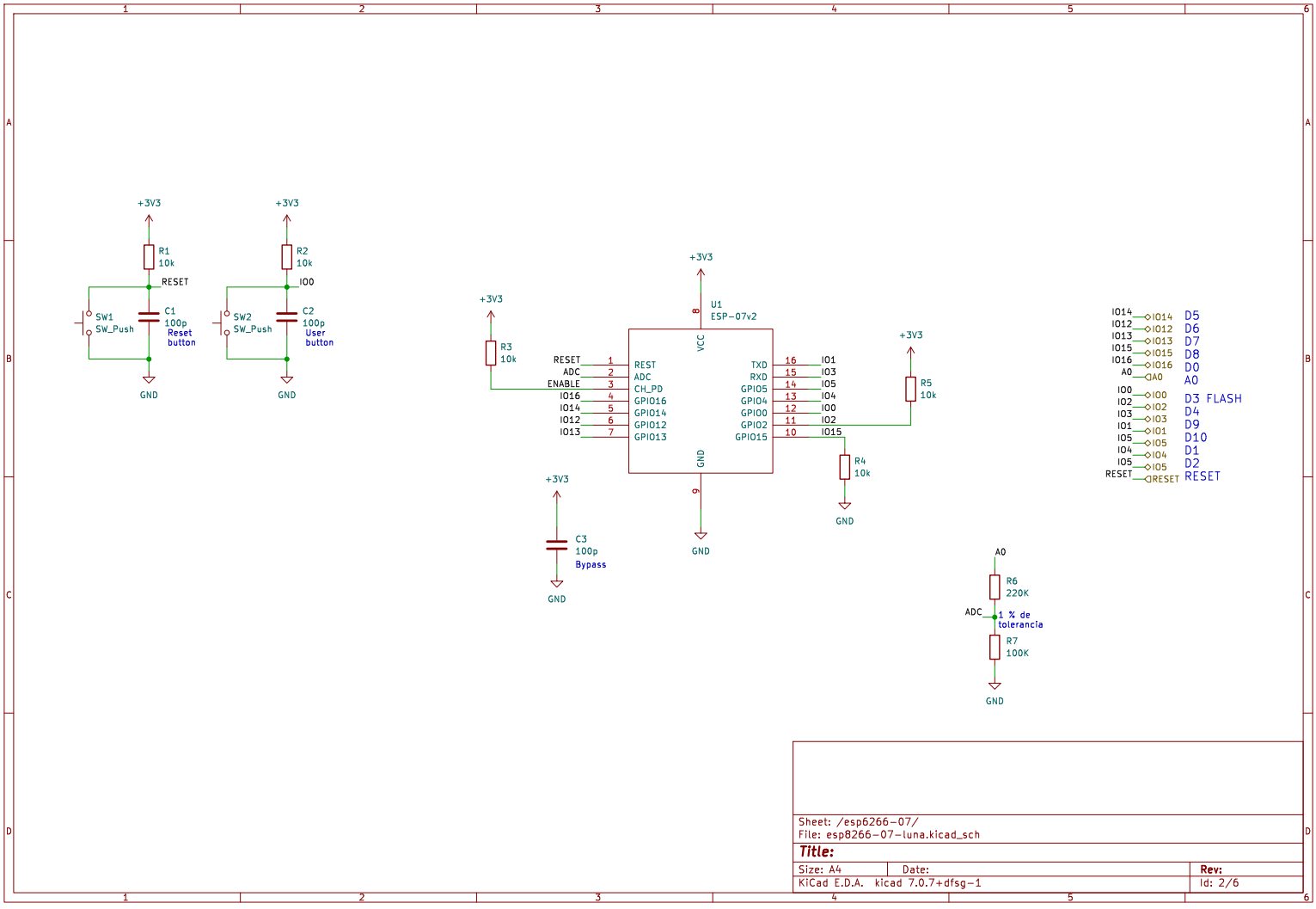
A Anexo: Circuito esquemático de la placa de desarrollo Luna

En este anexo se presenta el esquemático del circuito eléctrico de la tarjeta de desarrollo para el kit. Este esquema fue desarrollado con el software de diseño asistido por computadora denominado KiCad, el cual es opensource y es usada en el diseño de circuitos impresos (PCB).

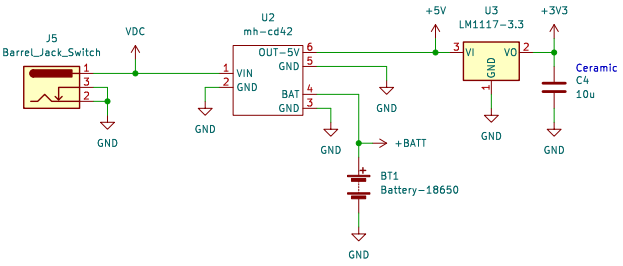
El diseño sigue el enfoque jerárquico y está constituido por los siguientes módulos:

1. El primer módulo hace referencia al controlador de la tarjeta de desarrollo, el cual consiste en un SoC con el circuito de polarización y control de señales para arranque del SoC como de la adecuación del conversor ADC.
2. El segundo módulo hace referencia a la entrada de energía, regulación de tensión de los periféricos de entrada y salida de la tarjeta y carga de batería para funcionamiento autónomo.
3. El tercer módulo está dedicado a los periféricos de entrada y salida, los cuales son adecuados a los niveles de tensión de la tarjeta de desarrollo.
4. Finalmente, el cuarto módulo está dedicado a los conectores para la comunicación entre otros periféricos del exterior.

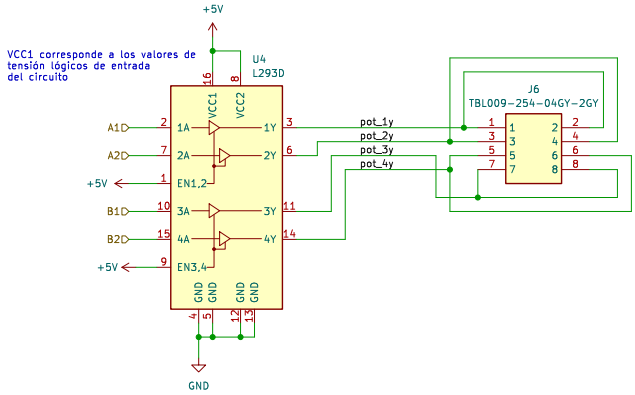

power mh cd42 5v y 3v3
File: power-mh-dc42.kicad_sch
esp6266-07
RESET
IO16
IO5
IO4
IO0
IO2
IO14
IO12
IO13
IO15
IO3
IO1
A0
D0
D1
D2
D3
D4
D5
D6
D7
D8
RX
TX
A0
12
11
10
9
8
7
6
5
4
3
2
1
J2
gpio
File: esp8266-07-luna.kicad_sch
J1
Conn_01x01_Male
1
Motor driver
RX
TX
D6
D7
B2
B1
A2
A1
File: driver-motor-l293d.kicad_sch
oled-i2c-uart
RX
TX
D3
D5
RX
TX
SDA
SCL
File: conn-i2c-rj11-oled.kicad_sch
passive-buzzer
D8
input
File: passive buzzer.kicad_sch
+5V
12
11
10
9
8
7
6
5
4
3
2
1
J3
vcc
12
11
10
9
8
7
6
5
4
3
2
1
J4
gnd
GND
Sheet: /
File: luna8266-v5.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 7.0.7+dfsg-1
Id: 1/6



Sheet: /esp6266-07/		
File: esp8266-07-luna.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 7.0.7+dfsg-1		Id: 2/6



Sheet: /power mh cd42 5v y 3v3/		
File: power-mh-dc42.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 7.0.7+dfsg-1		Id: 3/6

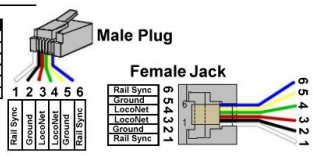


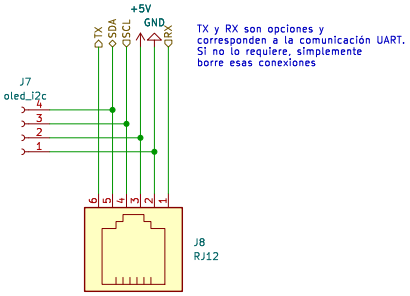
Sheet: /Motor driver/		
File: driver-motor-l293d.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 7.0.7+dfsg-1		Id: 4/6

Digitrax Wiring Standards

Pin No	Colour		Function	Voltage
1	White		Rail Sync-B	7 vdc
2	Black		Ground	—
3	Red		LocoNet	14.5 vdc
4	Green		LocoNet	14.5 vdc
5	Yellow		Ground	—
6	Blue		Rail Sync-A	7 vdc

All components are RJ12 6-wire
Do not use RJ11 4-wire.

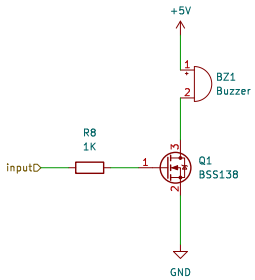




Sheet: /oled-i2c-uart/
File: conn-i2c-rj11-oled.kicad_sch

Title:

Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 7.0.7+dfsg-1		Id: 5/6



Sheet: /passive-buzzer/		
File: passive buzzer.kicad_sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 7.0.7+dfsg-1		Id: 6/6

B Anexo: Código Lua para tarjeta de desarrollo

En este anexo se expone la implementación en código necesaria para cumplir con los objetivos y requisitos de programación para la integración de la tarjeta de desarrollo en el material concreto. Dado que el firmware utilizado en la tarjeta de desarrollo ESP8266 es NodeMCU, se requiere la redacción de los scripts en el lenguaje de programación Lua. Cada uno de los scripts presentados a continuación lleva a cabo diversas funciones que, en conjunto, realizan las siguientes actividades:

- Inicialización de servicios de instrucciones remotas.
- Puesta en marcha de servicios de red, ya sea como estación o punto de acceso Wi-Fi.
- Configuración del menú de la tarjeta de red y supervisión del estado de los archivos almacenados en la tarjeta de desarrollo.
- Control de sensores o actuadores mediante el uso del lenguaje de programación Lua integrado en la memoria persistente de la tarjeta de desarrollo.

```
1  -- Nombre de archivo: init.lua
2  -- Autor: Johnny Cubides
3  -- Licencia: GPL
4
5  tmr.delay(1000000)
6  gpio.mode(4, gpio.OUTPUT)
7
8  function start()
9    dofile("wifi.lc")
10   gpio.mode(9,gpio.INPUT,gpio.PULLUP)
11   gpio.mode(3,gpio.INPUT,gpio.PULLUP)
12   gpio.write(4,gpio.LOW)
13   print(wifi.ap.getip())
14   if gpio.read(9) == 0 or gpio.read(3) == 0 then
15     uart.setup(0, 115200,8,0,1,1)
16     print("config ap-st")
17     collectgarbage()
18     dofile('apst.lc')
19   else
20     uart.setup(0, 115200,8,0,1,1)
21     print("server.lc")
22     collectgarbage()
23     dofile('server.lc')
24   end
```

```
end

start()
```

Listing B.1: Codigo de inicio en lenguaje Lua

```
-- Nombre de archivo: server.lua
-- Autor: Johnny Cubides
-- Licencia: GPL

gpio.mode(4, gpio.OUTPUT)
counter = 0
function saveLua(data2save)
  if string.match(data2save,'^.-data')=="save___data" then
    local data_save = string.sub(data2save,11)
    file.write(data_save)
    gpio.write(4,gpio.HIGH)
  elseif string.match(data2save,'^.-begin')=="save___begin" then
    file.remove("run.lua")
    file.open("run.lua","w")
    file.write([[""]])
    file.close()
    file.open("run.lua","w+")
    gpio.write(4,gpio.HIGH)
  elseif string.match(data2save,'^.-end')=="save___end" then
    file.close()
    gpio.write(4,gpio.HIGH)
  elseif string.match(data2save,'^.-rm')=="save___rm" then
    file.remove("run.lua")
    gpio.write(4,gpio.HIGH)
  elseif string.match(data2save,'^.-rs')=="save___rs" then
    node.restart()
  else
    file.close()
  end
end

sv1522 = net.createServer(net.TCP, 30)

function receiver1522(sck, data)
  gpio.mode(4, gpio.OUTPUT)
  gpio.write(4,gpio.LOW)
  saveLua(data)
  counter= counter + 1
  sck:send(counter.."\r\n")
  --sck:close()
end
if sv1522 then
  sv1522:listen(1522, function(conn)
    conn:on("receive", receiver1522)
    -- conn:send("ok\r\n")
    conn:on("connection", function(sck)
      counter = 0
      sck:send("luabot,"..(wifi.sta.getip() and wifi.sta.getip() or "no_sta").." \r\n")
      -- sck:send("luabot\r\n")
    end)
  end)
end

```

```
54 if file.exists('run.lua') then
55   print("run")
56   collectgarbage()
57   dofile('run.lua')
58 end
```

Listing B.2: Código del servicio en lenguaje Lua

```
1  -- Nombre de archivo: apst.lua
2  -- Autor: Johnny Cubides
3  -- Licencia: GPL
4
5  file.remove('run.lua')
6  gpio.write(4,gpio.LOW)
7  local str=wifi.ap.getmac();
8  local ssidTemp=string.format("%s%s",string.sub(str,13,14),string.sub(str,16,17));
9  cfg={}
10 cfg.ssid="catalejo-"..ssidTemp;
11 cfg.pwd="87654321"
12 cfg.save=true
13 wifi.ap.config(cfg)
14 cfg={}
15 cfg.ip="192.168.4.1";
16 cfg.netmask="255.255.255.0";
17 cfg.gateway="192.168.4.1";
18 wifi.ap.setip(cfg);
19 wifi.setmode(wifi.STATIONAP, true)
20 node.heap()
21 wifi.ap.getmac()
22 str=nil;
23 ssidTemp=nil;
24 collectgarbage();
25 gpio.write(4,gpio.HIGH)
26
27 ap_mac = wifi.ap.getmac();
28
29 ledStatusSTA = tmr.create()
30 ledStatusSTA:register(1000, tmr.ALARM_AUTO, function()
31   if wifi.sta.status() == wifi.STA_GOTIP then
32     gpio.write(4,gpio.LOW)
33     ledStatusSTA:stop()
34   end
35 end)
36
37 function connect2AP(ssid,pass)
38   -- wifi.setmode(wifi.STATION)
39   ledStatusSTA:stop()
40   station_cfg={}
41   station_cfg.ssid=ssid
42   station_cfg.pwd=pass
43   station_cfg.save=true
44   station_cfg.auto=true
45   wifi.sta.config(station_cfg)
46   -- wifi.sta.connect()
47   ledStatusSTA:start()
48 end
49
50 srv=net.createServer(net.TCP,60);
51 srv:listen(80, function(conn)
```

```
52  conn:on("receive", function(conn,payload)
53    ssid_start,ssid_end=string.find(payload,"SSID=");
54    if ssid_start and ssid_end then
55      amper1_start, amper1_end =string.find(payload,"&", ssid_end+1);
56      if amper1_start and amper1_end then
57        http_start, http_end =string.find(payload,"HTTP/1.1", ssid_end+1);
58        if http_start and http_end then
59          ssid=string.gsub(string.sub(payload,ssid_end+1, amper1_start-1),'+',' ');
60          password=string.gsub(string.sub(payload,amper1_end+10, http_start-2),'+',' ');
61          if ssid == "" then
62            wifi.sta.clearconfig()
63            print("STA config deleted")
64          elseif ssid and password then
65            connect2AP(ssid,password)
66          end
67        end
68      end
69    end
70  end)
71  local remaining, used, total=file.fsinfo()
72  local l = file.list();
73  local fileAndSize=""
74  for k,v in pairs(l) do
75      fileAndSize = fileAndSize .. "<h4>name: "..k.."\t, size: "..v.." Bytes</h4>\r\n"
76  end
77  local body= "<!DOCTYPE html>\r\n"
78  .."<html>\r\n"
79  .."<head>\r\n"
80  .."<meta name=\"viewport\" charset=\"utf-8\" content=\"width=device-width, initial-scale=1\">\r\n"
81  .."</head>\r\n"
82  .."<body>\r\n"
83  .."<h1>Luna </h1>\r\n"
84  .."<h2>Conectarse a una red</h2>\r\n"
85  .."<h4>Introducir el SSID y password de la red</h4>\r\n"
86  .."<form action='' method='get'>\r\n"
87  .."<label for='SSID'>SSID red  : </label>"
88  .."<input type='text' name='SSID' value='' maxlength='100'/>\r\n"
89  .."<br/>"
90  .."<label for='Password'>Password: </label>"
91  .."<input type='text' name='Password' value='' maxlength='100'/>\r\n"
92  .."<input type='submit' value='Submit' />\r\n"
93  .."</form>\r\n"
94  .."<p>Nota: Si deja el SSID de red vacío se borrará la configuración de red anterior</p>\r\n"
95  .."<p>IP en red LAN: "..(wifi.sta.getip() and "<b>"..wifi.sta.getip().."</b>" or "no conectado a una red.").."
        </p>\r\n"
96  .."<h3>File system info:</h3>\r\n"
97  .."<h4>Total: "..total.." Bytes, Used: "..used.." Bytes, Remain: "..remaining.." Bytes\r\n</h4>"
98  .."<h3>Lists all files in the file system</h3>\r\n"
99  ..fileAndSize
100 .."<h4>MAC: "..ap_mac.."</h4>\r\n"
101 .."<h2>www.catalejoplus.com</h2>"
102 .."</body>\r\n"
103 .."</html>\r\n"
104 local pageWifi = "HTTP/1.1 200 OK\r\n"
105 .."Connection: Keep-Alive\r\n"
106 .."Content-Length: "..string.len(body).."\r\n"
107 .."Content-Type: text/html; \r\n\r\n"
108 ..body
109 conn:send(pageWifi)
```

```
110 end)
```

Listing B.3: Código de configuración de conexión y configuración del sistema embebido en lenguaje Lua

```
1  -- Nombre de archivo: hcsr04.lua
2  -- Autor: Johnny Cubides
3  -- Licencia: GPL
4
5  HCSR04 = {}
6
7  HCSR04.new = function (trig, echo)
8    local self = {}
9    -- 0 = succesful, 1 = en proceso, -1 = error, -2 = activate, 2 = desactivado
10   local t_rising, t_falling
11   local status = 2 -- desactivado
12
13   local function cb_echo(level, when)
14     if level == gpio.HIGH then
15       t_rising = when
16     elseif level == gpio.LOW then
17       t_falling = when
18       if t_falling > t_rising then
19         status = 0 -- succesful
20       else
21         status = -1 -- error
22       end
23     end
24   end
25
26   function self.activate()
27     gpio.mode(trig, gpio.OUTPUT)
28     gpio.write(trig, gpio.LOW)
29     gpio.mode(echo, gpio.INT)
30     gpio.trig(echo, "both", cb_echo)
31     status = -2 -- activate
32   end
33
34   function self.deactivate()
35     gpio.trig(echo, "none")
36     status = 2 -- desactivado
37   end
38
39   function self.status()
40     return status
41   end
42
43   function self.trigger()
44     t_rising, t_falling, status = 0, 0, 1 -- en proceso
45     gpio.write(trig, gpio.HIGH)
46     tmr.delay(10)
47     gpio.write(trig, gpio.LOW)
48   end
49
50   function self.measure()
51     -- 340/2 = 170
52     return (t_falling - t_rising) * 170 / 1000 -- centimetros
53   end
54
55   self.activate()
```

```
56
57   return self
58 end
```

Listing B.4: Código diseñado para el uso del sensor ultrasonido HCSR04 en lenguaje Lua

C Anexo: Bosquejo del diseño del IDE Catalejo Editor

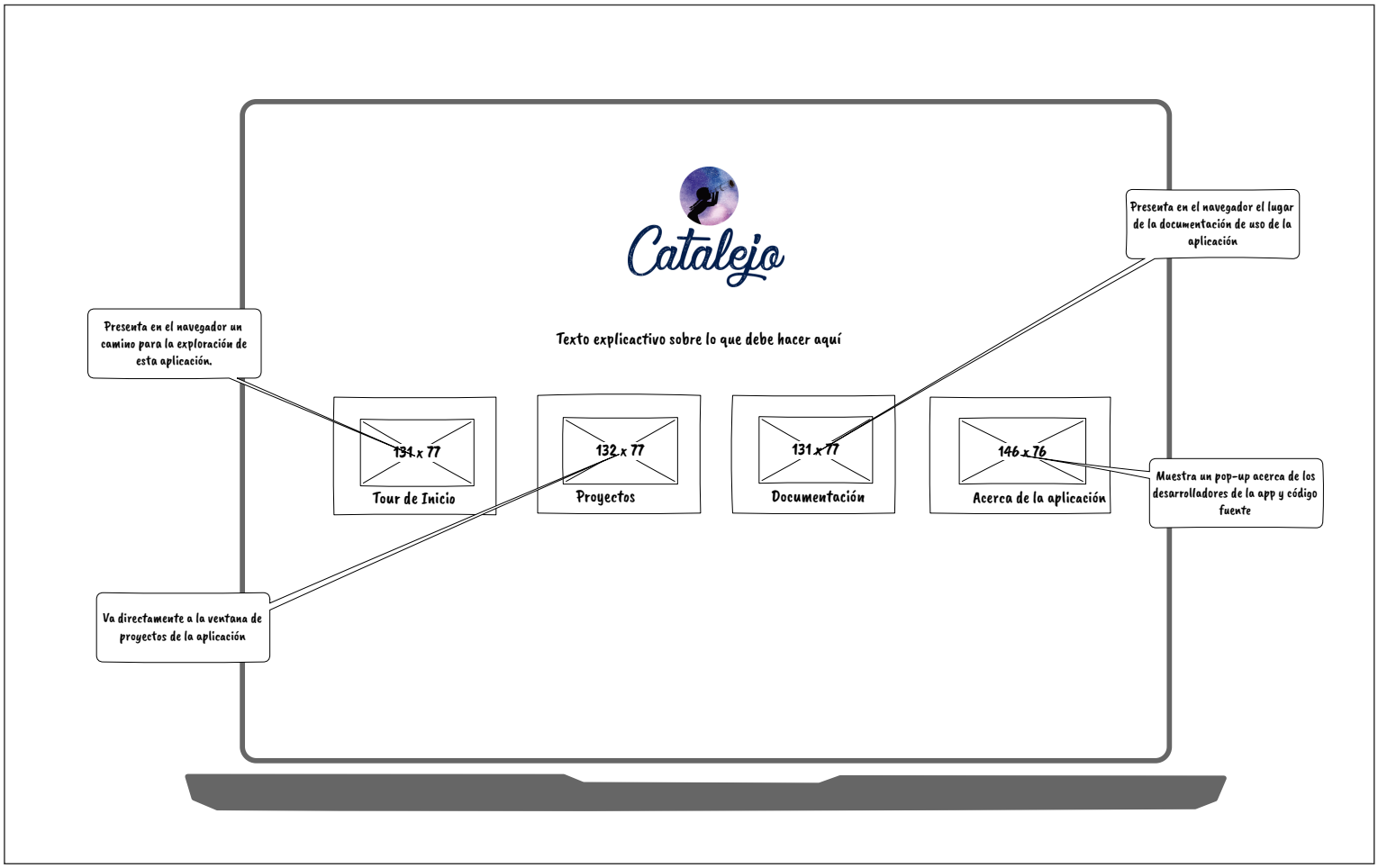
En esta sección se presenta el diseño elaborado para la construcción de la interfaz visual del entorno de desarrollo Catalejo Editor. El criterio de diseño de este esquema tiene como objetivo satisfacer las necesidades de la experiencia del usuario. En él se muestran las diversas pantallas que utilizará el usuario, como el proceso de incorporación (onboarding), el menú principal, el gestor de proyectos, el explorador de archivos, la edición de código, la edición de múltiples códigos, así como los distintos menús, como el menú flotante, y los editores de código específicos para guardar los proyectos de Luna y ChucK. Estos bosquejos fueron elaborados utilizando la herramienta Pencil [1], la cual es de código abierto (open source).

[1] Sitio oficila de pencil: `https://pencil.evolus.vn/`

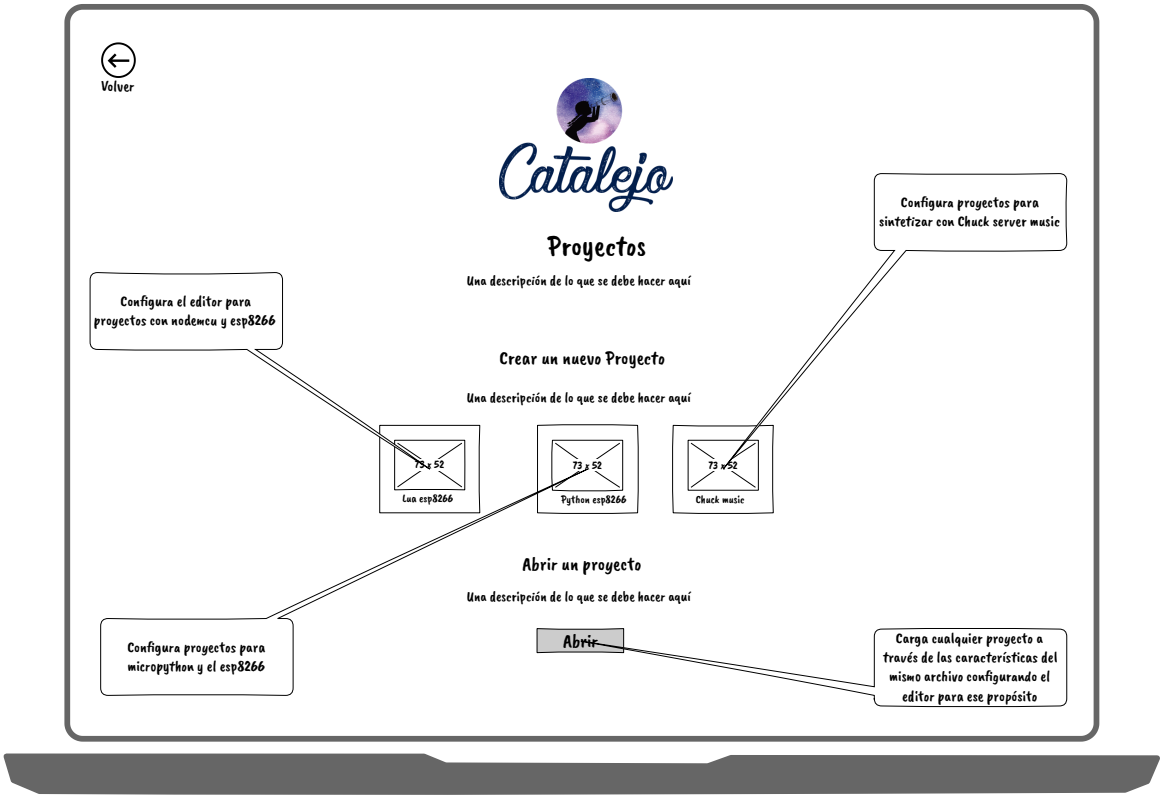
Onboarding


Catalejo
Distintos slides que representan las principales características de la aplicación
Haz tus sueños realidad
¡Empieza!

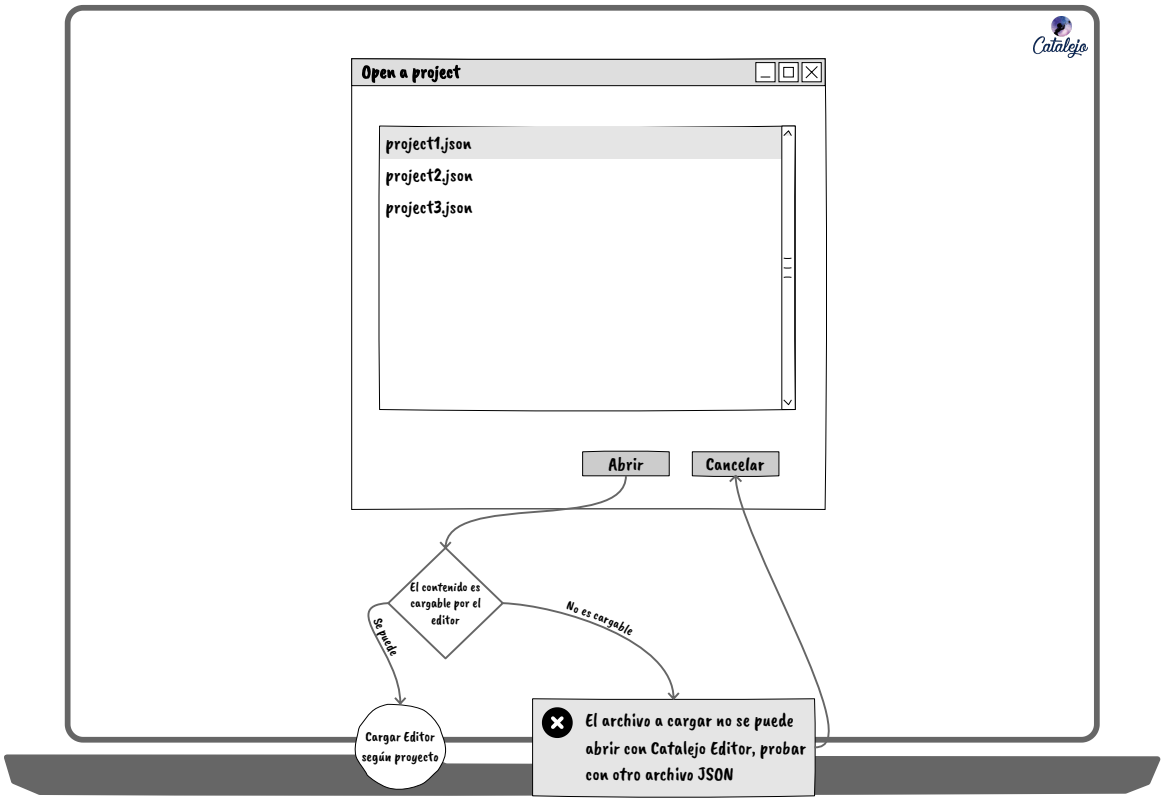
Menu


Catalejo
Texto explicactivo sobre lo que debe hacer aquí
Presenta en el navegador un camino para la exploración de esta aplicación.
Presenta en el navegador el lugar de la documentación de uso de la aplicación
Muestra un pop-up acerca de los desarrolladores de la app y código fuente
Va directamente a la ventana de proyectos de la aplicación
134 x 77
Tour de Inicio
132 x 77
Proyectos
131 x 77
Documentación
146 x 76
Acerca de la aplicación

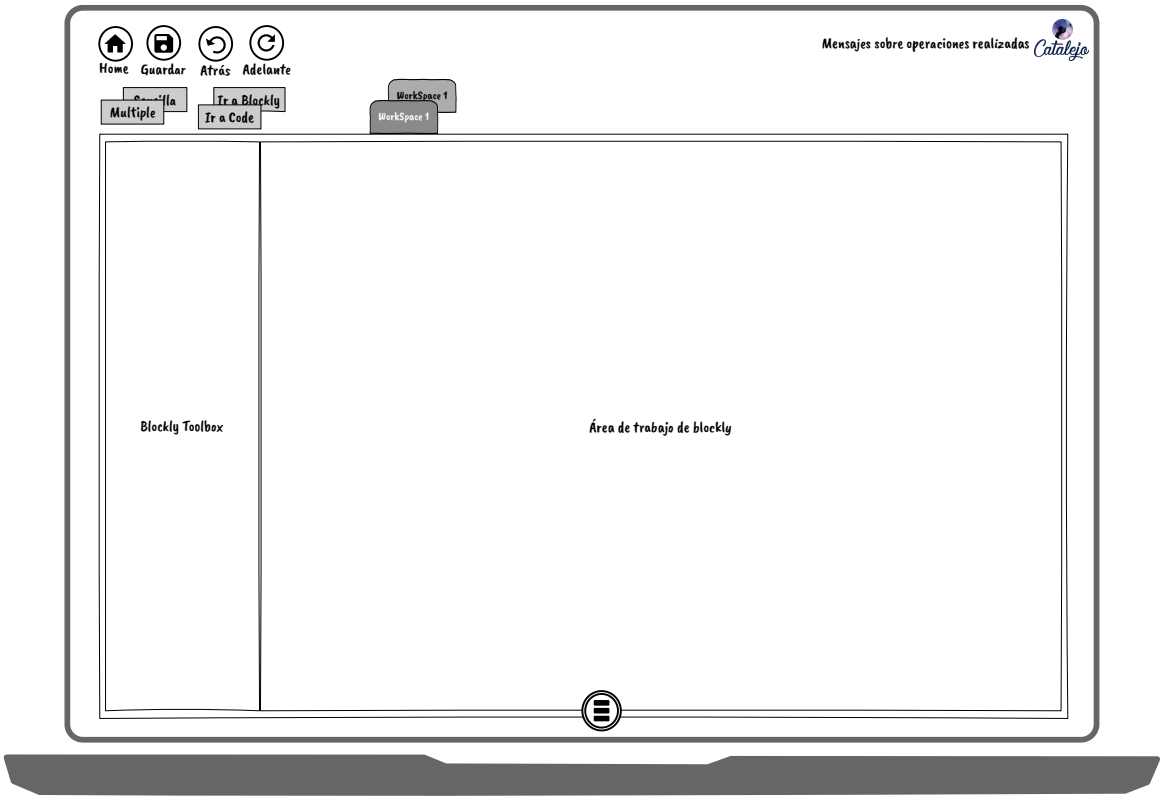
Projects


Volver
Catalejo
Proyectos
Una descripción de lo que se debe hacer aquí
Configura el editor para proyectos con nodemcu y esp8266
Configura proyectos para sintetizar con Chuck server music
Crear un nuevo Proyecto
Una descripción de lo que se debe hacer aquí
73 x 52
Lua esp8266
73 x 52
Python esp8266
73 x 52
Chuck music
Abrir un proyecto
Una descripción de lo que se debe hacer aquí
Configura proyectos para micropython y el esp8266
Abrir
Carga cualquier proyecto a través de las características del mismo archivo configurando el editor para ese propósito

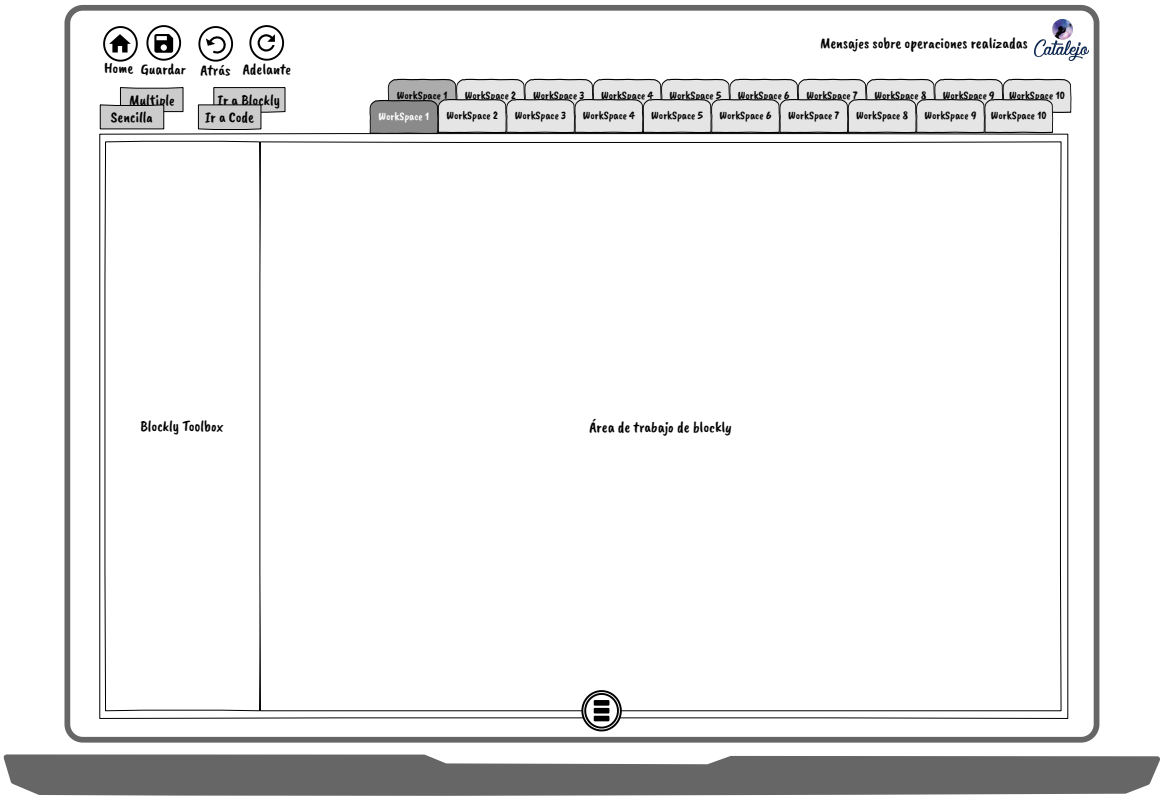
File-Explorer



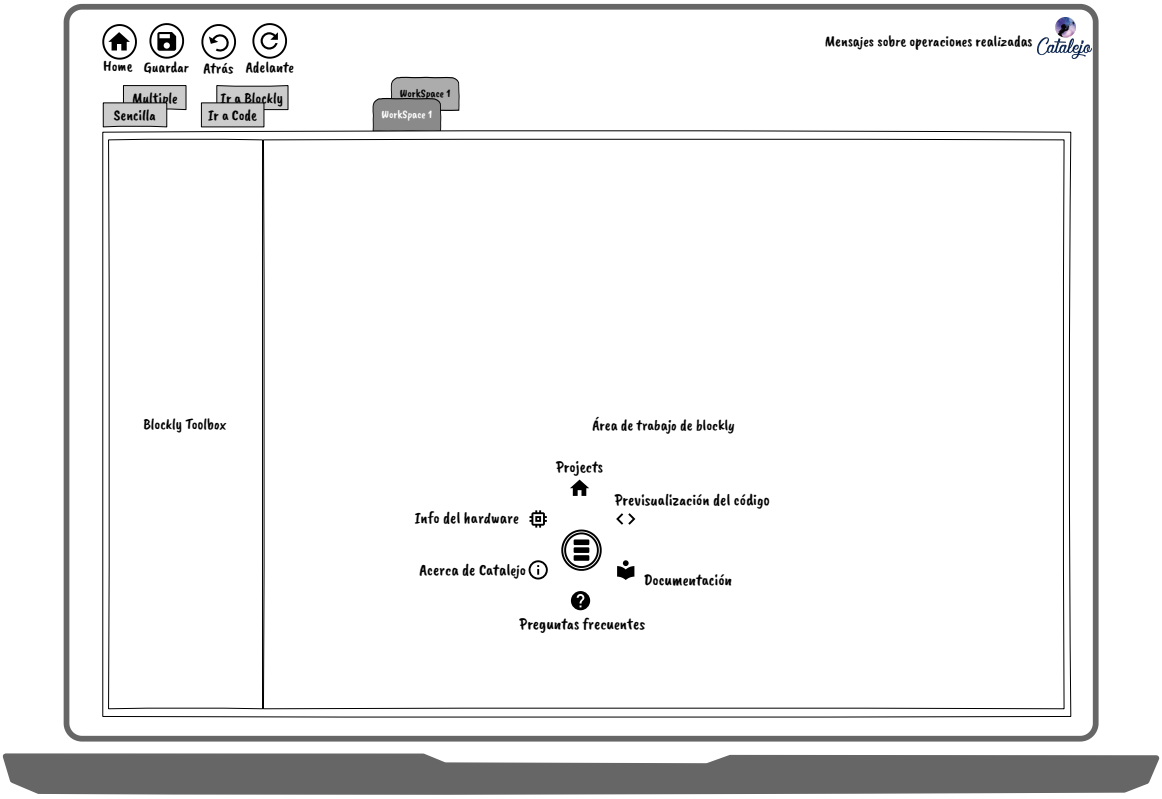
Editor simple



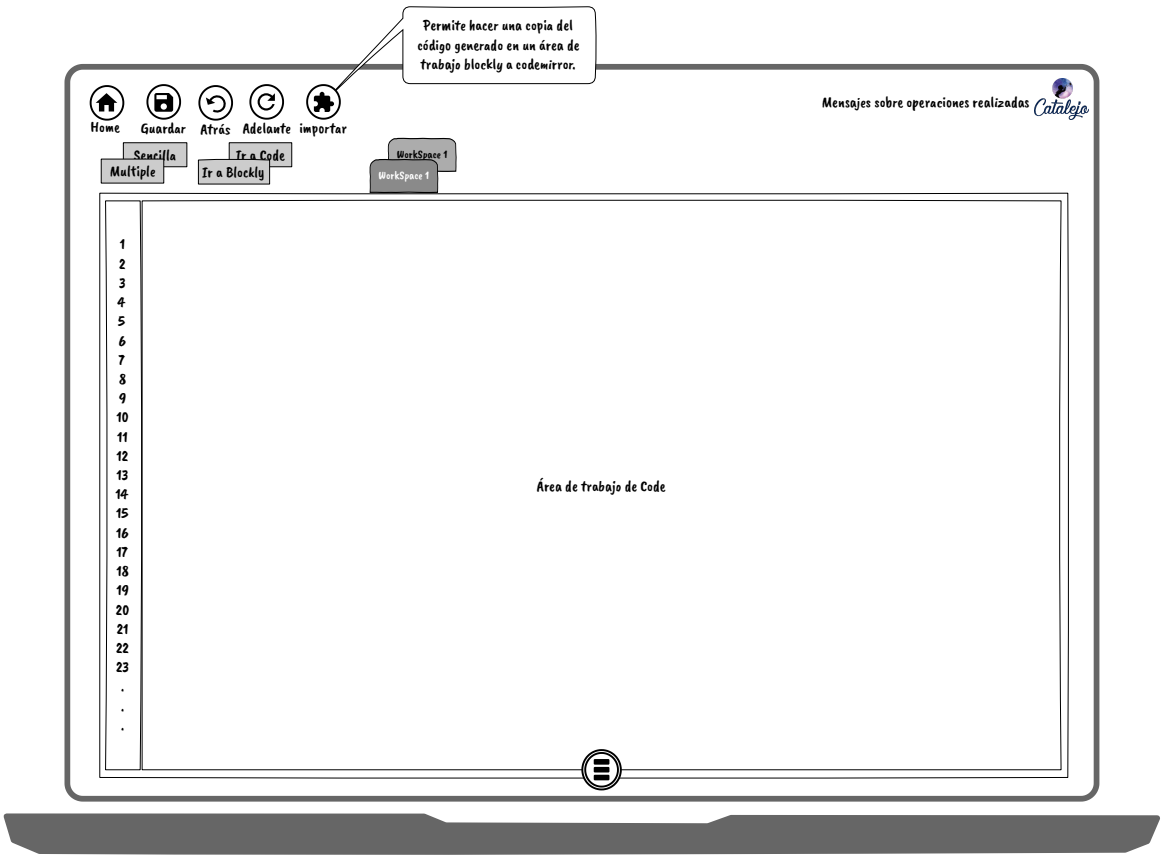
Editor muiltiple



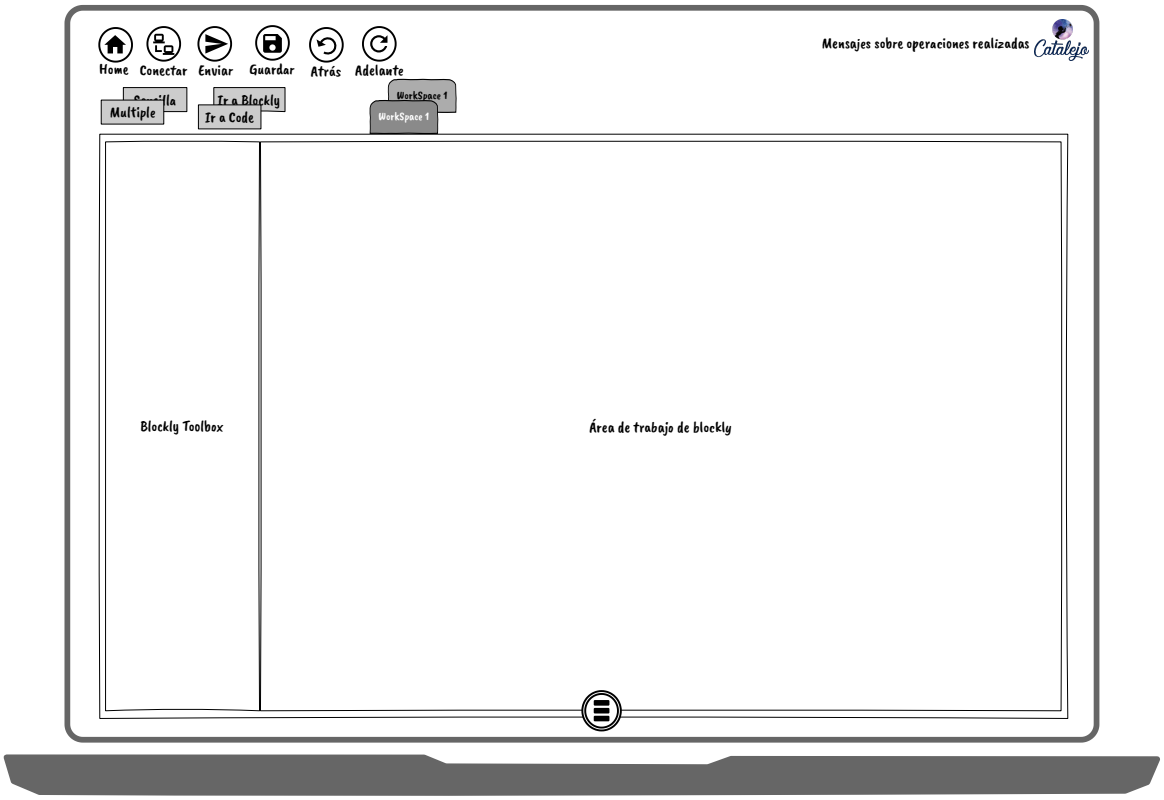
Float menu


Home
Guardar
Atrás
Adelante
Mensajes sobre operaciones realizadas
Catalejo
Multiple
Sencilla
Ir a Blockly
Ir a Code
WorkSpace 1
WorkSpace 1
Blockly Toolbox
Área de trabajo de blockly
Projects
Previsualización del código
Info del hardware
Acerca de Catalejo
Documentación
Preguntas frecuentes

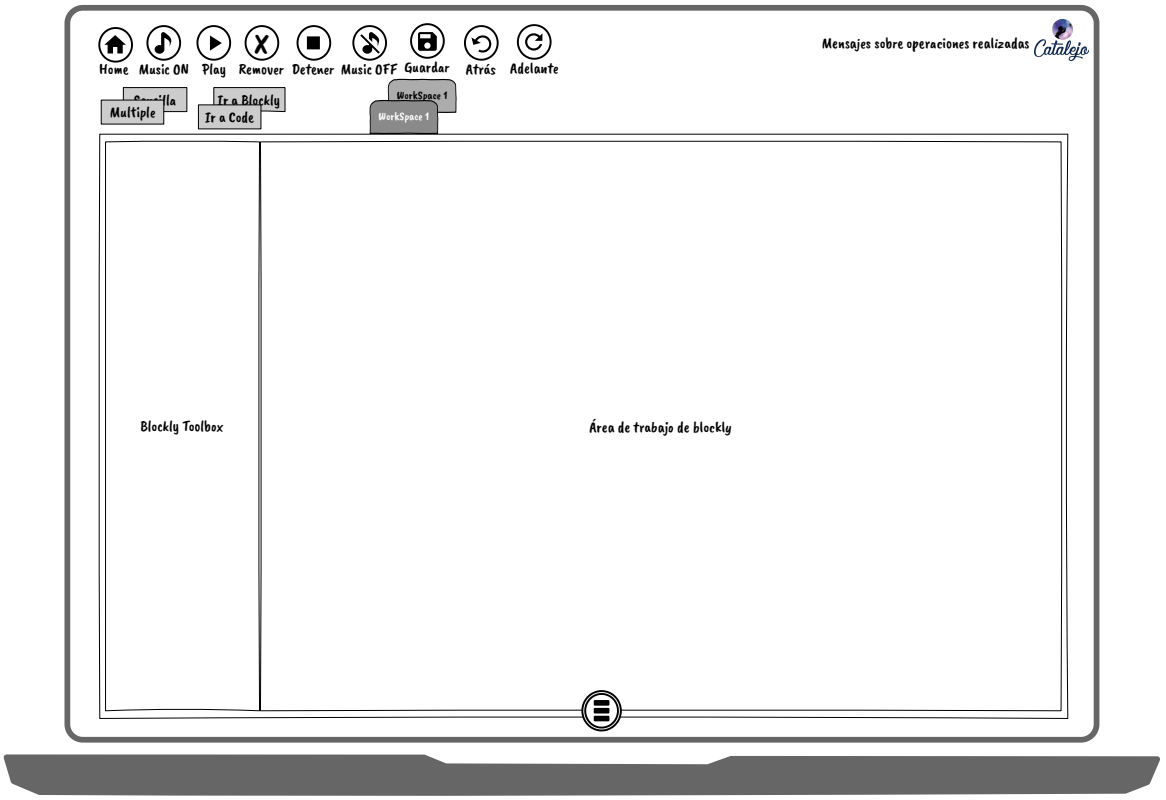
Editor code


Permite hacer una copia del código generado en un área de trabajo blockly a codemirror.
Home
Guardar
Atrás
Adelante
importar
Mensajes sobre operaciones realizadas
Catalejo
Sencilla
Multiple
Ir a Code
Ir a Blockly
WorkSpace 1
WorkSpace 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 . . .
Área de trabajo de Code

editor-lua-esp8266



editor-chuck



D Anexo: Material didáctico para Catalejo Editor con Chuck

Este anexo hace referencia a la documentación del material didáctico, el cual, está compuesto por el material concreto, es decir, el material tangible y el software de programación llamado Catalejo Editor. El propósito principal de este material es incentivar al estudiante el desarrollo de su aprendizaje a través de la autoformación. Inicialmente, se busca motivar al estudiante a desarrollar actividades donde a través de la imitación, replique los experimentos y revise los resultados. En el siguiente nivel de aprendizaje, además de replicar las experiencias, se incentiva a realizar modificaciones de estos experimentos para explorar las diversas posibilidades que ofrece el Kit. Además, este material ofrece diferentes niveles de lectura para que el estudiante pueda acceder a este de acuerdo a su nivel de compresión: material gráfico, enlaces a vídeos, código de prueba, tanto en código textual como también programación en bloques.

La documentación se estructura en tres partes fundamentales: una ruta de aprendizaje que orienta al estudiante a través de actividades progresivas para alcanzar objetivos específicos, ejercicios técnicos que abordan temas particulares de programación, como procesos, funciones, operadores y rutinas, y finalmente, una guía de instalación de herramientas para facilitar el proceso de implementación del material didáctico.

Live Coding Music con Catalejo-ChucK

None

Johnny Cubides

None

Table of contents

1. Documentación de chuck para catalejo editor

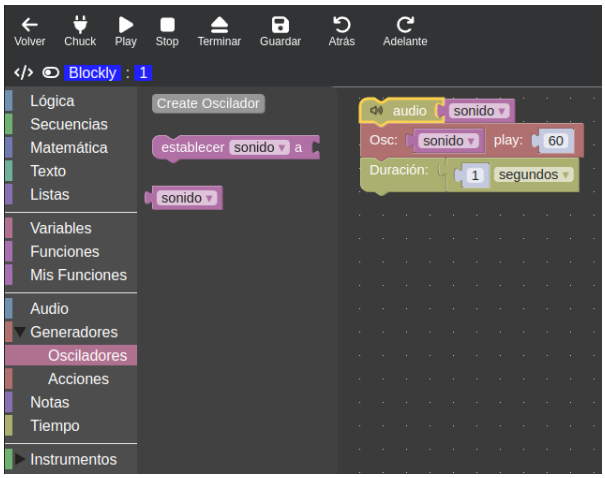
Sitio Web

2. Mis primeros pasos

2.1 Mi primera melodía

2.1.1 Mi primer tono

En catalejo editor en modo Blockly (programación por bloques) construye este código, activa ChucK y dale clic en el botón de **Play**.

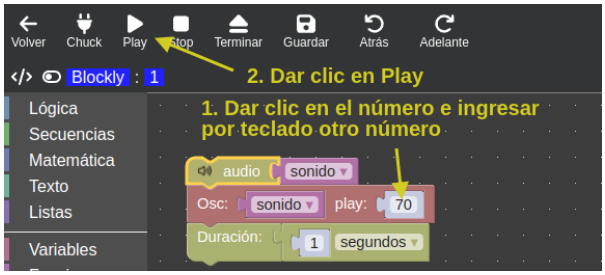


Para poder escuchar un sonido se requiere de dos conjuntos de bloques:

- Se requiere un tono que es representado en este caso por el número 60
- Se requiere una duración que es representada por el bloque de 1 segundo.

Si no se asigna la duración del tono la computadora tomará el tiempo más rápido posible por ella de ejecutar

2.1.2 Cambiando el tono

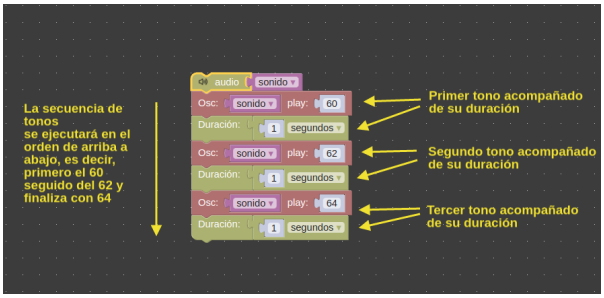


Prueba cambiando el número 60 por otros números y responde las siguientes preguntas:

- ¿Cómo se percibe el sonido con números menores a 60?
- ¿Cómo se percibe el sonido con números mayores a 60?

2.1.3 Tocando una melodía

Para tocar una melodía se requiere ordenar otros tonos de manera descendente (de arriba a bajo), cada tono nuevamente debe estar acompañada de su duración, ejemplo:



Experimente creando su propia melodía

1. Reproduzca la melodía de ejemplo
2. Cambie el tono de los bloques y escuche como suena

2.1.4 Alterando el ritmo

En este caso basta con cambiar el valor del tiempo por un otro número, ejemplo:

Experimente cambiando el ritmo

1. Reproduzca la melodía de ejemplo
2. Cambie el tiempo de los bloques y escuche como suena

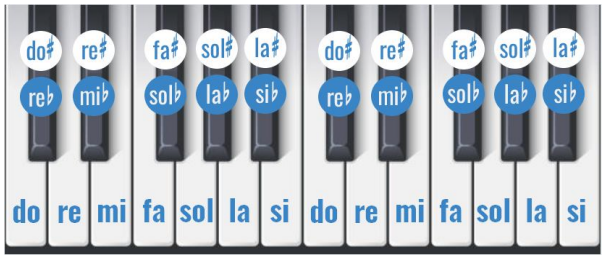
2.1.5 Creando su propia melodía

Construya una melodía en la cual tenga en cuenta lo siguiente:

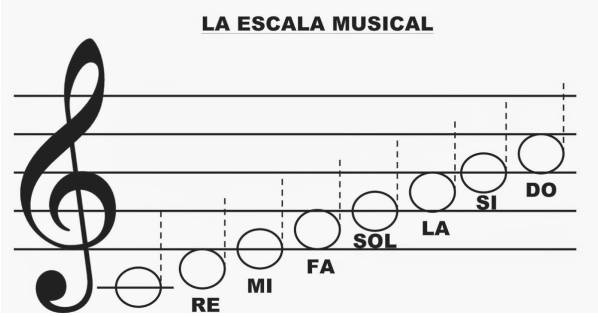
1. Tenga más de tres tonos
2. Altere el ritmo
3. Enséñela a una persona y pregúntele qué opina sobre cómo suena.

2.2 Mi segunda melodía

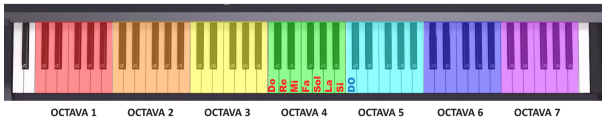
2.2.1 Notas en el piano



2.2.2 Escala musical de SOL



2.2.3 Octavas del piano



2.2.4 Significado del número 60 en el tono

- Notación midi

	Notas											
Octava	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71

	Notas											
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

	Notas											
Octava	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

Do en cuarta octava

- Toca una melodía: Agrega más bloques hacia abajo ...
- Altera el ritmo: Cambia el número de segundos para escuchar qué sucede, ejemplo: 0.5, 0.1, 2.
- Cambiando el tiempo:

2.2.5 Duración figuras musicales

Metrónomo

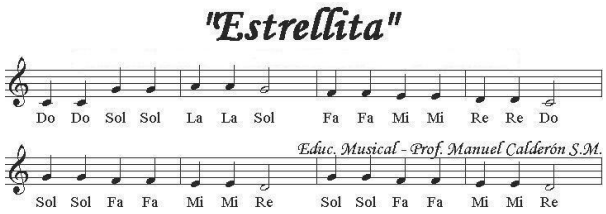
A continuación podrás reconocer la duración de las figuras musicales:

Duración de las Figuras Musicales

Nombre	Figura	Duración	Silencio
Redonda		4 tiempos	
Blanca		2 tiempos	
Negra		1 tiempo	
Corchea		1/2 tiempo	
Semicorchea		1/4 tiempo	

Pruba cambiando los PPM, qué pasa con valores de 400 o 80.

- Melodía a interpretar:



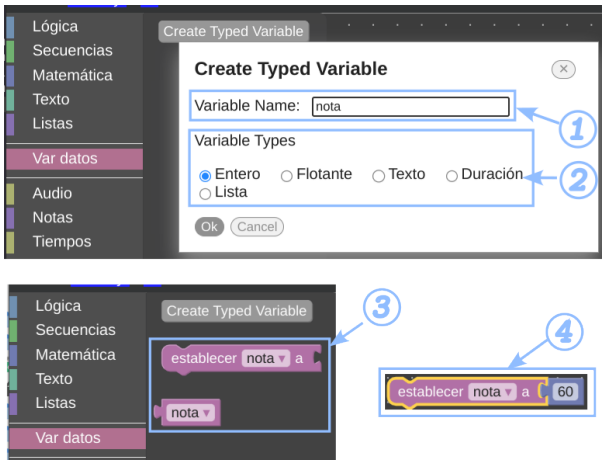
3. Introducción a la programación

3.1 Variables y Operadores

3.1.1 Variables

Las variables son nombres o identificadores de valores; los valores son cosas que corresponden a un tipo de dato como es el caso de números, letras, e inclusive instrumentos.

Para crear una variable se debe seleccionar en la caja de herramientas el tipo de variable a crear, poner un nombre y asignar un valor, ejemplo:



En la imagen de arriba los números representan lo siguiente:

1. para crear una variable del tipo entero se debe ir a la catagoría *Var datos* y en *Variable Name* se ha puesto el nombre de la variable como **Nota**.
2. En *Variable Types* seleccionar *Entero* y dar clic en el botón de Ok.
3. En la categoría *Var datos* se han creado dos bloques que representan la variable creada.
4. Como *nota* es una variable del tipo *Entero* desde la categoría Matemáticas se le ha asignado un valor entero de 60.

En el lenguaje de Chuck esta variable se representa como sigue:

```
int nota;

60 => nota;
```

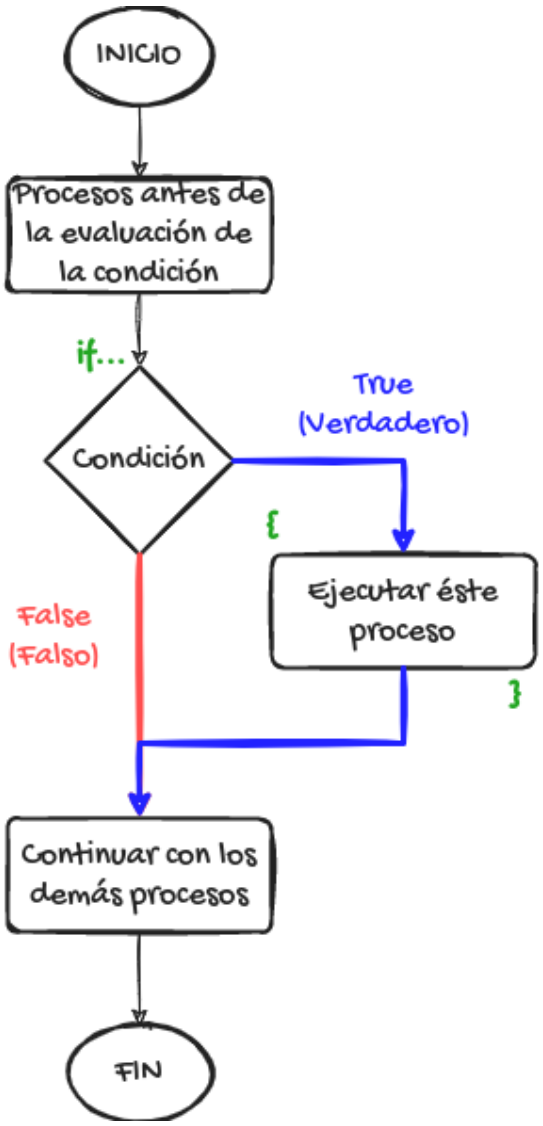
> **Otros tipos de variables**
>
> En chuck-blockly hay distintos tipos de variables que se pueden ubicar en las categorías de Var datos, Var instrumento, Var oscilador y Var comunicación.

3.1.2 Operadores

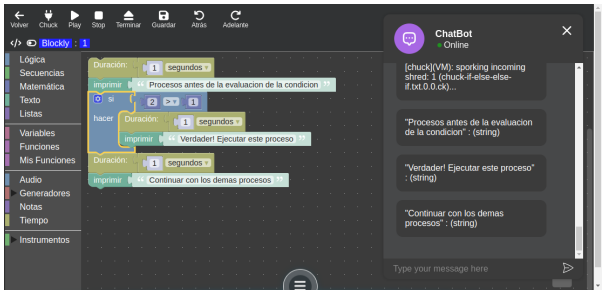

```
int nota1;
int nota2;
int nota3;

nota1 == nota2;

nota1 + nota2;

nota1 > nota2 || nota1 > nota3;
```

3.2 Control de Flujo

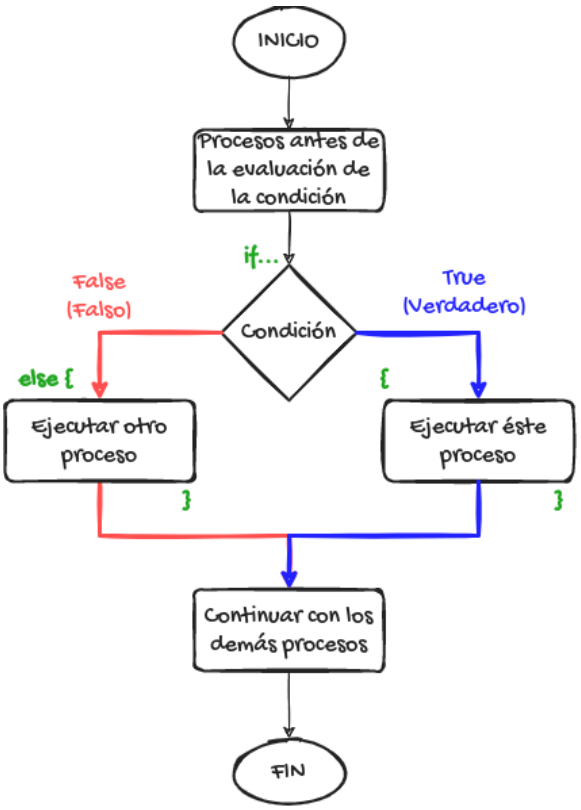
3.2.1 Estructura IF



Ejemplo IF en Blockly

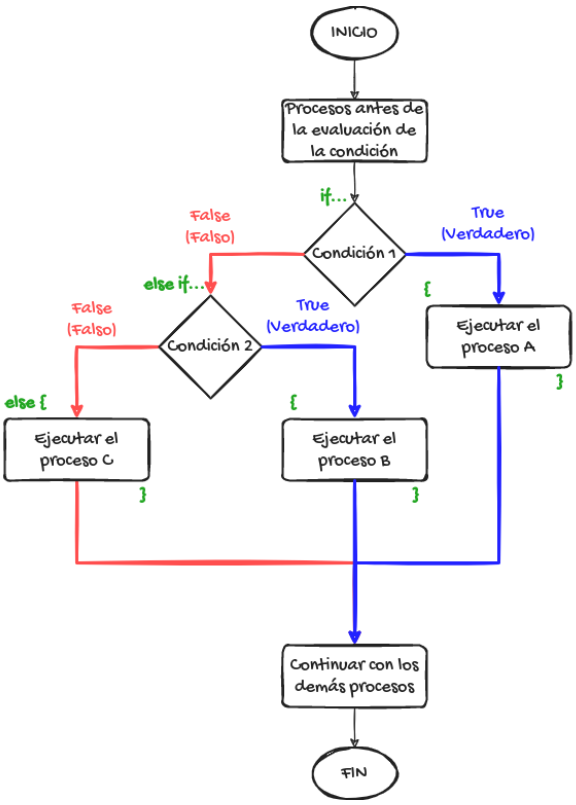


3.2.2 Estrutura ELSE



Ejemplo de ELSE en Blockly

3.2.3 Estrutura ELSE IF



Ejemplo de ELSE IF en Blockly

3.3 Notas midi

	Notas											
Octava	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

4. Instalación de herramientas

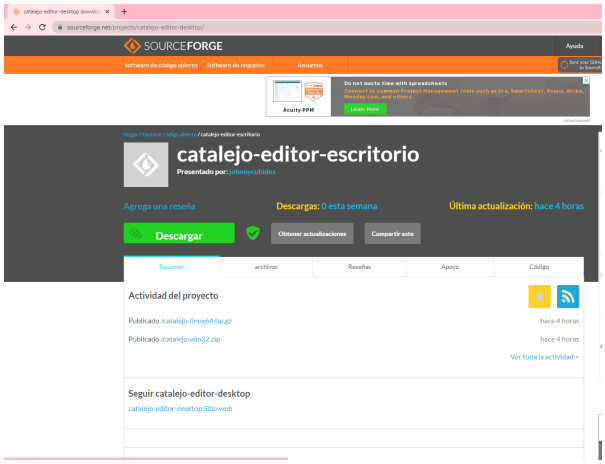
4.1 Catalejo editor

4.1.1 Catalejo Editor

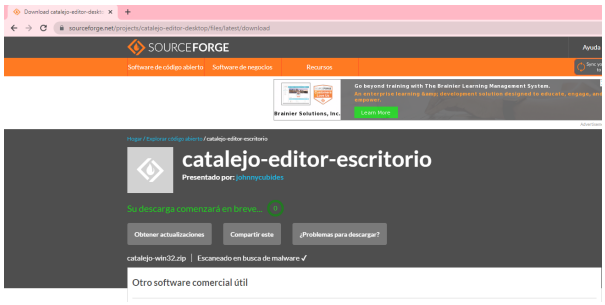
Instalar Catalejo Editor en Windows

DESCARGAR LA APLICACIÓN COMPRIMIDA CATALEJO-WIN32.ZIP

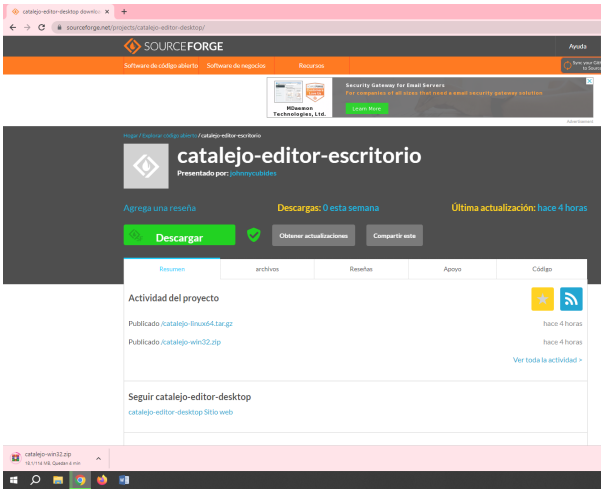
- En su navegador de preferencia diríjase al enlace https://sourceforge.net/projects/catalejo-editor-desktop/ e inicie la descarga de *Catalejo Editor* desde el botón de descarga (botón verde).



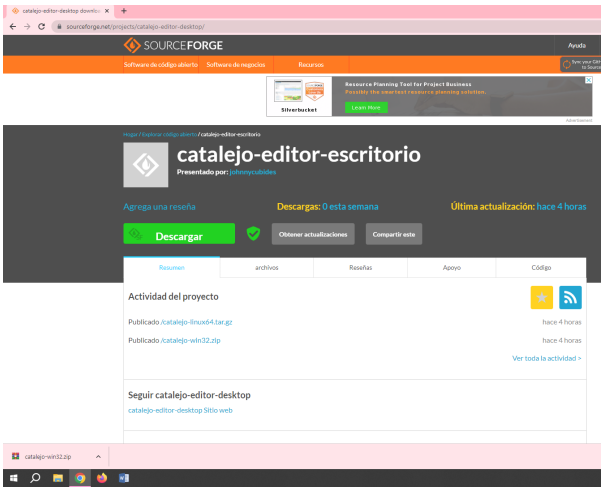
- El sistema reconocerá automáticamente su sistema y realizará la descarga del la aplicación para su sistema.



- El proceso de descarga durará algunos minutos.

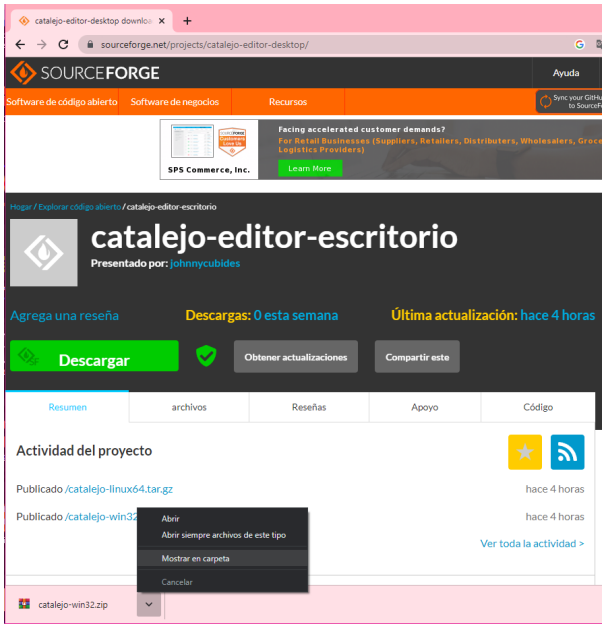


- Proceso de descarga terminado

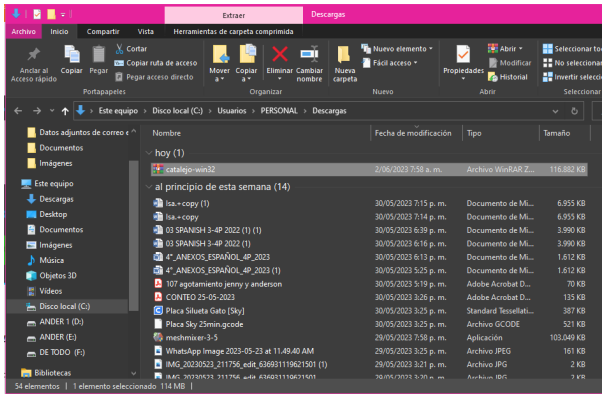


MOVER EL CATALEJO-WIN32.ZIP

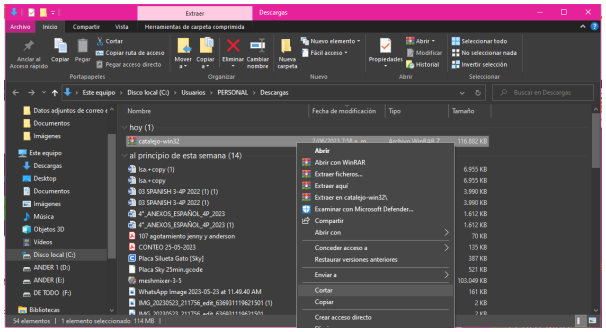
- Desde el navegador puede mostrar la carpeta donde fue descargada la aplicación



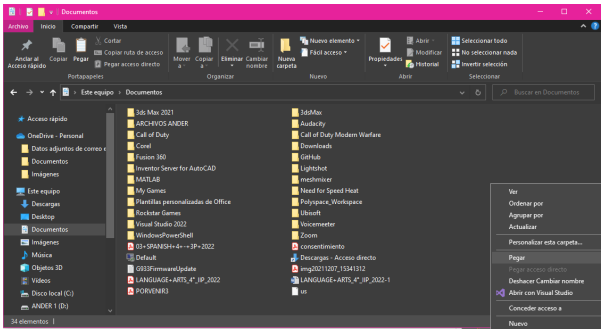
- Desde el explorador de archivos de Windows podrá ubicar catalejo-win32.zip

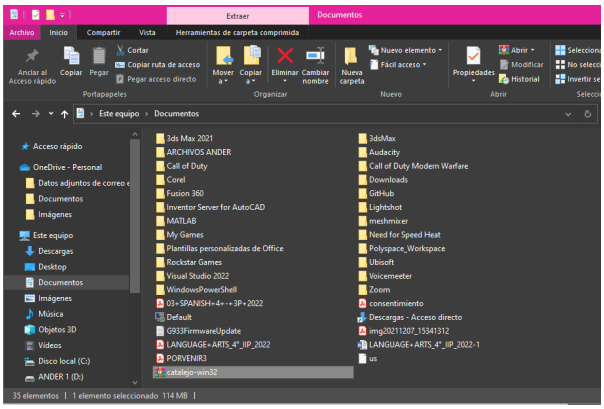


- Traslade el archivo.zip a otro lugar para descomprimir, como ejemplo, será trasladado a al directorio *Documentos*



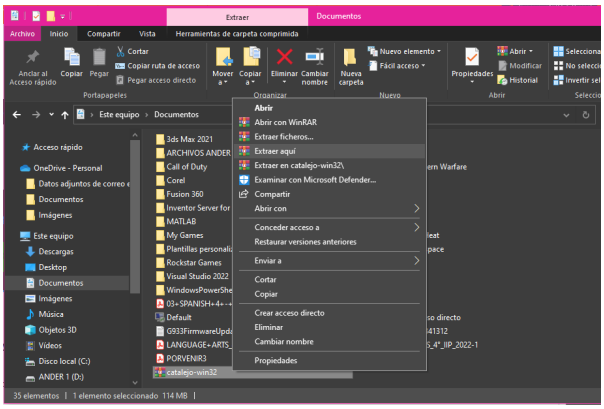
- Con clic derecho en el gestor de archivos pegue el archivo en el destino seleccionado



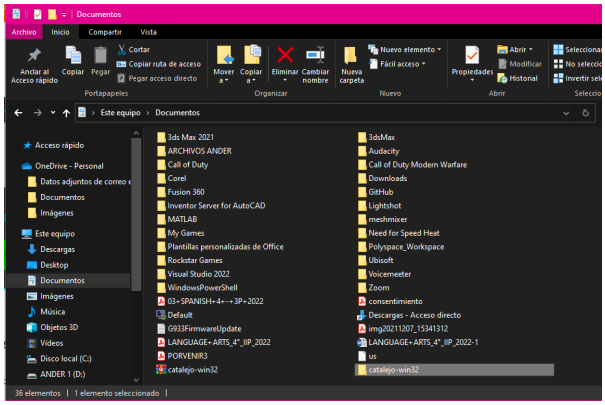


DESCOMPRIMIR EL CATALEJO-WIN32.ZIP

- A continuación extraiga el contenido del archivo.zip

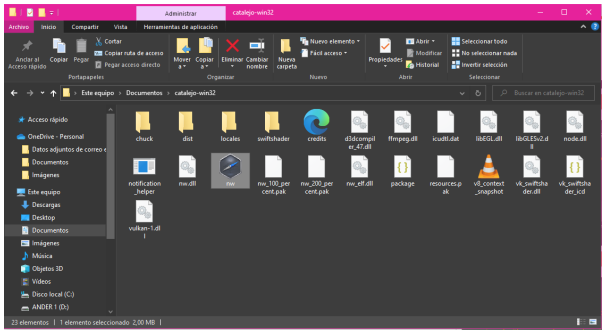


- Al extraer el contenido, se genera un nuevo directorio llamado `catalejo-win32`

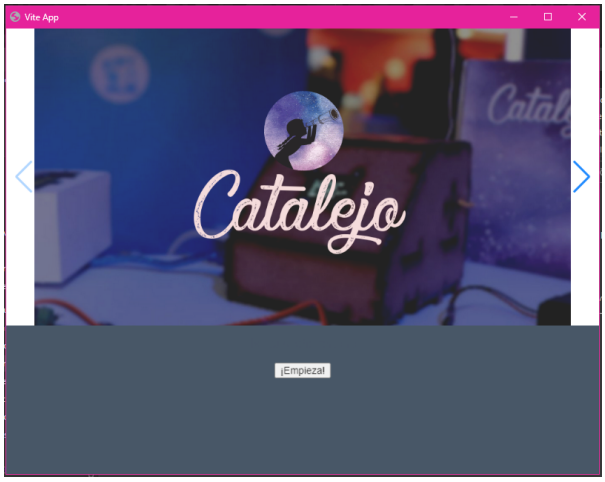


EJECUTAR LA APLICACIÓN CATALEJO EDITOR

- Al entrar al directorio descomprimido busque el ícono que tiene por nombre `nw` y dar en él doble clic

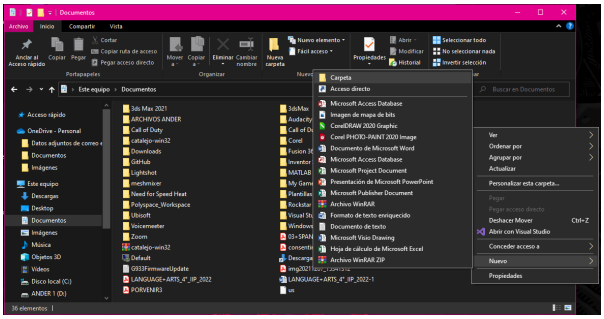


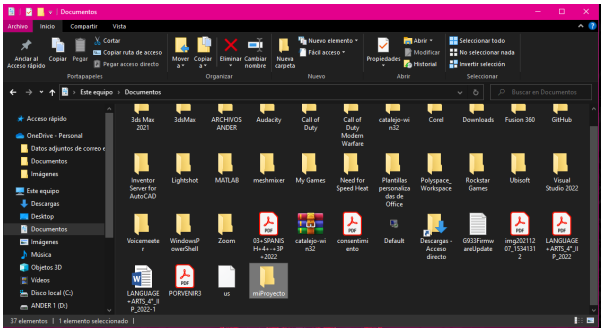
- Ya podrá interactuar con la aplicación



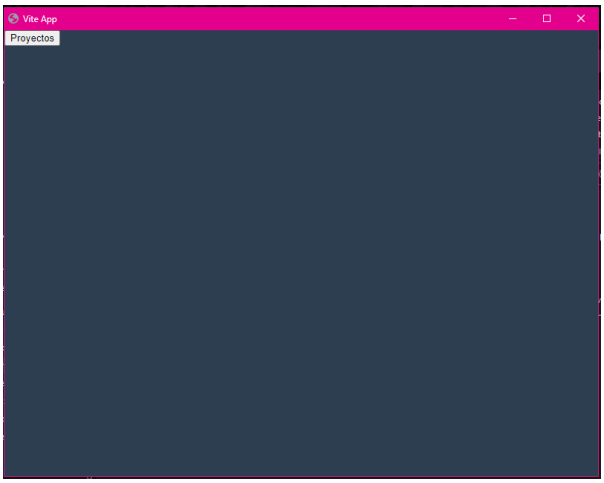
Mi primer Tono con Catalejo Editor y Chuck

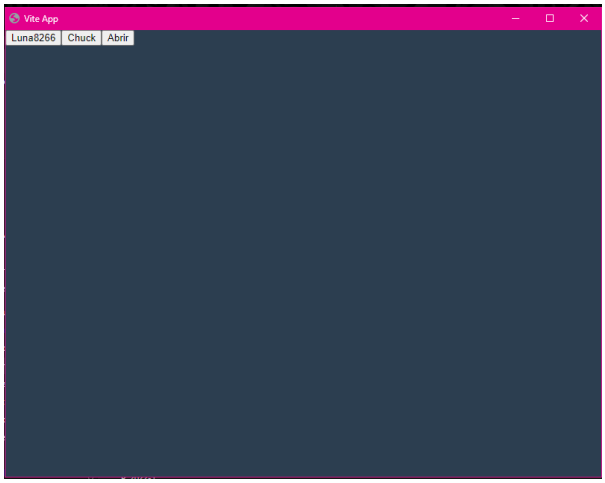
- Para iniciar un proyecto en *Catalejo Editor* es importante crear un directorio donde se pueda alojar cada proyecto, es decir, un directorio por proyecto, por ejemplo, se creará un directorio en `Documentos` llamado `miProyecto`

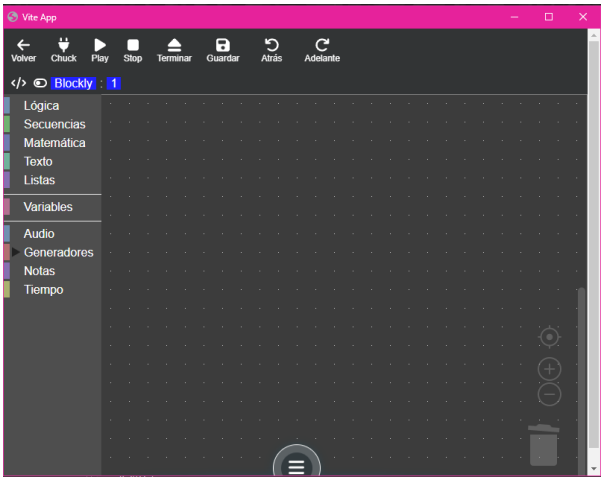


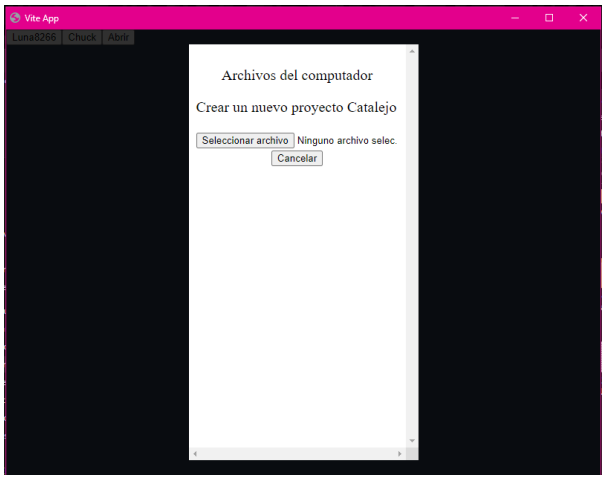


- Inicie la aplicación Catalejo Editor y en el botón al botón `¡Empieza!` dar clic, esto avanzará al botón `Proyectos`, dar clic, en proyectos seleccionar la opción `Chuck` y en la ventana emergente dar clic en el botón `Seleccionar archivo`

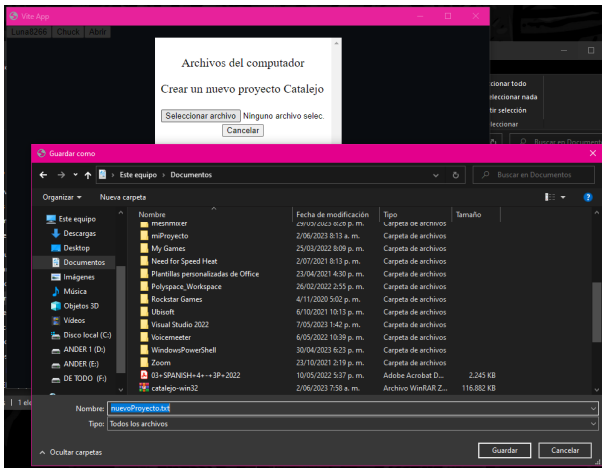






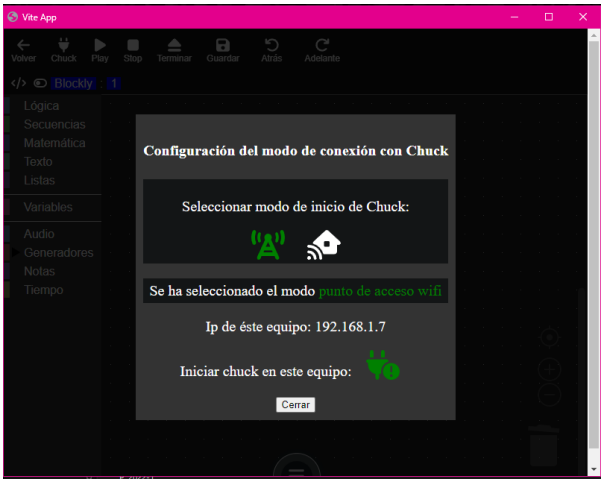


- Crear un nuevo archivo para guardar el código a realizar, en el ejemplo, `nuevoProyecto.txt`

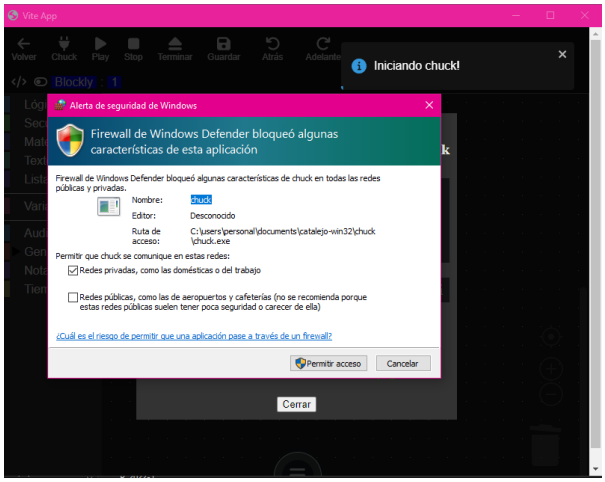


- Lo anterior abrirá un área de trabajo en Catalejo Editor

- Antes de editar el primer tono, se debe iniciar el servicio de chuck, dar clic en el ícono relacionado a chuck y en la ventana emergente, dar clic en el ícono asociado a `Iniciar chuck en este equipo`



- Por ser la primera vez que inicia chuck en éste equipo Windows pedirá permisos de ejecución, dar clic en permitir, y deberá cerrar la aplicación y volverla abrir realizando el mismo procedimiento de esta sección, este permiso es solicitado únicamente la primera vez que abre Catalejo Editor, así que puede luego continuar con el siguiente paso.


Vite App
Volver
Chuck
Play
Stop
Terminar
Guardar
Atrás
Adelante
Iniciando chuck!
Blockly
Alerta de seguridad de Windows
Firewall de Windows Defender bloqueó algunas características de esta aplicación
Firewall de Windows Defender bloqueó algunas características de chuck en todas las redes públicas y privadas.
Nombre: chuck
Editor: Desconocido
Ruta de acceso: C:\users\personal\documents\catalejo-win32\chuck\chuck.exe
Permitir que chuck se comunique en estas redes:
Redes privadas, como las domésticas o del trabajo
Redes públicas, como las de aeropuertos y cafeterías (no se recomienda porque estas redes públicas suelen tener poca seguridad o carecer de ella)
¿Cuál es el riesgo de permitir que una aplicación pase a través de un firewall?
Permitir acceso
Cancelar
Cerrar

E Anexo: Contenido del kit de materiales

En este apartado se presenta el material concreto, es decir, el kit de materiales. Esta documentación tiene como objetivo ser presentada junto al kit de materiales para que los estudiantes puedan identificar los diferentes dispositivos que lo componen. Se pretende que sea una guía visual que permita a los estudiantes reconocer cada uno de los componentes, acompañada de una descripción de sus funciones. Además, se proporciona un enlace que redirige a una sección documental en línea sobre proyectos inspiradores asociados al kit. Estos proyectos, conocidos como "cacharros que inspiran", son un conjunto de proyectos que se pueden realizar utilizando este kit, junto con la documentación sobre la implementación del software en la aplicación Catalejo Editor.

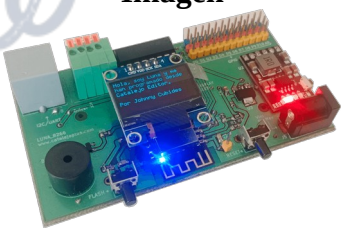
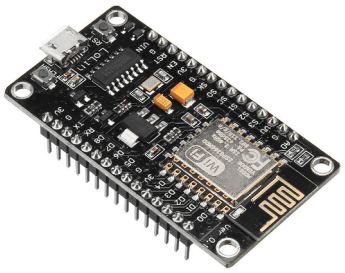
El Kit de materiales

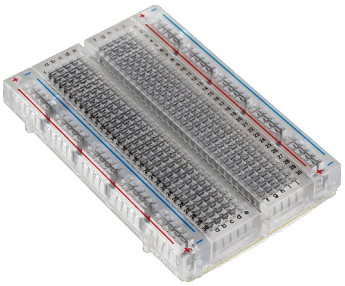
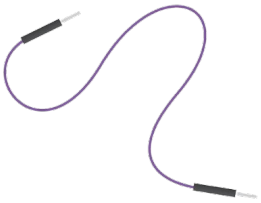
Índice

Resumen

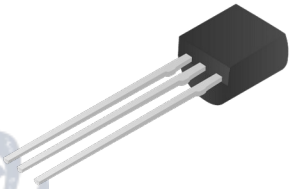
Se presenta a continuación el Kit de materiales, el cual, es una herramienta didáctica que permite la materialización de las diferentes iniciativas que pueda tener un estudiante. Se concibe como una caja de herramientas incompleta permitiendo al estudiante agregar aquellas cosas que considere relevantes. Esta herramienta acompañada por el software Catalejo Editor permite el aprendizaje de habilidades de pensamiento computacional, habilidades manuales y de creación entre otras.

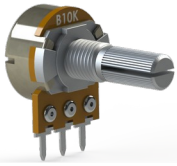
Componentes del kit de materiales

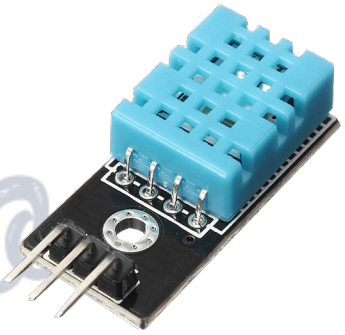
Cantidad	Componente	Característica	Imagen
1	Tarjeta de desarrollo Luna o NodeMCU	Es el cerebro de cada proyecto ya que permite procesar la información que recibe de cada sensor y controlar componentes como controladores, tiene disponible conexión WIFI para conexión con otros dispositivos como celulares, PC, etc.	 

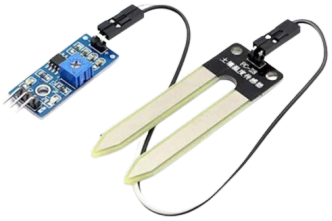
1	Cable de energía o USB	El cable micro USB es necesario para transferir datos entre la PC o dar energía a la tarjeta de desarrollo. Se puede conectar en puertos USB disponibles en Cargadores, PowerBank y PC.	
1	Protoboard hall de 400 puntos (con marca roja y azul)	La protoboard es una placa con varios puntos de conexión que nos permite unir varios componentes en un solo lugar.	
10	Jumper H/H x 20cm	Este tipo de conector macho-macho son de conexión rápida permiten conectar sensores o actuadores a la tarjeta de desarrollo.	
10	Jumper M/M x 20cm	Este tipo de conector macho-macho son de conexión rápida permite realizar conexiones directamente en la protoboard entre diferentes componentes como actuadores, resistencias, etc.	

10	Jumper H/M x 20cm	Este tipo de conector macho-hembra son de conexión rápida permiten conectar sensores o actuadores en la protoboard.	
10	LED 5mm	Leds de diferentes colores difusos (4 rojos, 3 verdes, 3 amarillos). El LED es un componente que nos permite indicar por medio de la luz cambios de estado presentes en nuestros proyectos. Este tipo de led es difuso ya que evita que su luz se propague al exterior.	
2	LED 5mm chorro	Led de chorro de luz blanca. El LED es un componente que nos permite indicar por medio de la luz cambios de estado presentes en nuestros proyectos. Este tipo de led es de chorro ya que permite que su luz se propague al exterior.	
60	Resitencias de 1/4 vatio surtidas	La resistencia es un elemento que se opone al paso de corriente, este tipo de resistencia ya tienen un valor establecido por defecto.	

4	Caiman-caimán	Este tipo de cable tiene en los extremos unas pinzas similares a la mandíbula de un caimán, permiten realizar puentes haciendo retención firme en sus conexiones.	
4	Pulsadores	Interruptor pulsador de 2 pines para protoboard. El pulsador permite realizar un puente entre dos puntos de conexión mientras se mantenga pulsado.	
4	2n2222A	El transistor es un componente que nos permite aumentar la energía requerida para componentes más grandes como altavoces y motores.	
4	Diodo 1N4004	El diodo es un componente electrónico que solo deja circular corriente en un único sentido.	

2	Fotoresistor	Es una resistencia eléctrica la cual varía su valor en función de la cantidad de luz que incide sobre su superficie.	
1	Potenciometro de 100 Kohms	El potenciómetro es una resistencia variable por movimiento mecánico. Es decir, se puede variar su valor si aplicamos giros sobre él.	
1	Motor DC de 3 a 6 V	El motor nos permite convertir la energía eléctrica en movimiento.	
1	Buzzer Activo	El buzzer es un componente que nos permite convertir la electricidad en sonido.	
1	Codensador	Condensandor electrolitico 330 uF a 16-25 V	

1	Condesandor 104	Condensandor cerámico 104	
1	LED RGB	Led que emite luz en rojo, verde, azul de 5mm	
1	Sensor DHT11	Este sensor captura datos de humedad y temperatura. Es decir se puede aplicar en actividades como invernaderos, agricultura, monitoreo de temperatura, etc.	
1	Sensor DS18B20	Este sensor permite tomar datos de temperatura hasta en sustancias líquidas, lo que permite determinar en que estado se puede encontrar dicha sustancia.	

1	Sensor ultrasonido HC-SR04	El sensor de ultrasonido tiene por objetivo brindar información sobre la distancia a su alrededor, de esta manera los murciélagos se mantienen informados de los obstáculos en su entorno para así poder desplazarse de un lugar a otro.	
1	Sensor sonda de humedad YL-100	El sensor de humedad de suelo puede ser enterrado en el suelo y nos permite determinar que tan húmedo se encuentra este.	
1	Sensor de nivel de agua YL-83	Sensor de nivel de agua o lluvia tipo sonda que funciona a través del método de medición de conducción. Este módulo permite indicar cuando en su superficie hay agua. Es decir, varia su resistencia en presencia de agua.	
1	Parlante bocina	Actuador, Elemento sonoro de 8 Omhs o 4 Omhs, medio vatio o similares. El parlante es un componente que nos permite convertir la electricidad en sonido.	

1	servomotor	Actuador de posicionamiento de brazo en un ángulo de 0 a 180 grados, torque máximo un kilogramo, material. El servomotor tiene la función de mover su mecanismo a un angulo determinado, su movimiento se asemeja al de un brazo humano. Con él se pueden mover cosas.	
1	Caja pástica	Esta caja nos sirve para almacenar todos los componentes que contiene el kit y poderlos transportar a cualquier lugar.	

Proyectos que se pueden realizar con el kit de materiales

En la siguiente imagen se observa la relación que existe entre los diferentes componentes del kit que pueden dar origen a soluciones de problema a través de artefactos.

Para conocer más proyectos que se pueden realizar con este kit le invitamos a ver el enlace relacionado a *cacharros que inspiran*.

Elaborado por Johnny Cubides

jgcubidesc@catalejoplus.com

Referencias Bibliográficas

[1] Diseño y Arquitectura Pedagógica. Aprendizaje activo 4.0 aprendizaje basado en fenómenos (abf). https://hdl.handle.net/11285/643915, 2022.

[2] Carlos Iván Camargo Bareño. Transferencia tecnológica y de conocimientos en el diseño de sistemas embebidos. page 135, 2011. URL http://www.bdigital.unal.edu.co/5696/.

[3] Espressif Systems. Hardware Design Guidelines. *Esp32*, page 38, 2022. URL https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[4] María Angélica Arán and María Luisa Ortega. Enfoques de aprendizaje y hábitos de estudio en estudiantes universitarios de primer año de tres carreras de la Universidad Mayor Temuco, Chile 2011. *Hekademos: revista educativa digital*, 11:37–46, 2012. URL http://dialnet.unirioja.es/descarga/articulo/4059756.pdf%5Cnhttp://dialnet.unirioja.es/servlet/extart?codigo=4059756.

[5] The Bolonia. Metodologías activas para la formación de competencias. pages 35–56, 2006.

[6] Pasi Mattila and Pasi Silander. *How to Create the School of the Future– Revolutionary thinking and design from Finland*. 2015. ISBN 9789526208183. URL http://nebula.wsimg.com/57b76261c219f5e7083e9978cd2cd66d?AccessKeyId=3209BE92A5393B603C75&disposition=0&alloworigin=1.

[7] Michael D Deschryver and Aman Yadav. Creative and Computational Thinking in the Context of New Literacies : Working with Teachers to Scaffold Complex Technology-Mediated Approaches to Teaching and Learning. *Journal of Technology and Teacher Education*, 23(3):411–431, 2015.

[8] Filiz Kalelioglu, Yasemin Gulbahar, and Volkan Kukul. A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing*, 4(3):583, 2016. ISSN 2255-8942.

[9] Susanne Hambrusch, Christoph Hoffmann, John T. Korb, Mark Haugan, and Antony L. Hosking. A multidisciplinary approach towards computational thinking for science majors. *SIGCSE Bulletin Inroads*, 41(1):183–187, 2009. ISSN 10963936. doi: 10.1145/1539024.1508931.

[10] J. A.M. Howe, P. M. Ross, K. R. Johnson, F. Plane, and R. Inglis. Teaching mathematics through programming in the classroom. *Computers and Education*, 6(1):85–91, 1982. ISSN 03601315. doi: 10.1016/0360-1315(82)90016-1.

[11] Vernon A Magnesen. A Review of Findings from Learning and Memory Retention Studies. *Innovation Abstracts*, 5(25), 1983. URL http://search.ebscohost.com/login.aspx?direct=true{&}db=eric{&}AN=ED234878{&}site=ehost-live.

[12] Stephen Paul Linder, Brian Edward Nestrick, Symen Mulders, and Catherine Leah Lavelle. Facilitating active learning with inexpensive mobile robots. *Journal of Computing Sciences in Colleges*, 16(4):21–33, 2001. ISSN 1098-6596.

[13] Adriano Baratè, Andrea Formica, Luca A. Ludovico, and Dario Malchiodi. Fostering computational thinking in secondary school through music: An educational experience based on Google Blockly. *CSEDU 2017 - Proceedings of the 9th International Conference on Computer Supported Education*, 2 (Csedu):117–124, 2017. doi: 10.5220/0006313001170124.

[14] Alexander Repenning, Jürg Zurmühle, Anna Lamprou, and Daniel Hug. Computational music thinking patterns: Connecting music education with computer science education through the design of interactive notations. *CSEDU 2020 - Proceedings of the 12th International Conference on Computer Supported Education*, 1(Csedu):641–652, 2020. doi: 10.5220/0009817506410652.

[15] Cambridge Assessment-International Education. ¿Cuál es el significado de Aprendizaje Activo? *Ucles*, pages 1–5, 2019.

[16] Allen Downey and Chris Meyers. *Introducción a la programación con Python*. 2009. ISBN 9789588347226. URL `https://books.google.com.co/books?id=0r_MjgEACAAJ`.

[17] Allen B Downey. *Pensando la computación como un científico (con Java)*. 2012. ISBN 9789876301176.

[18] Pilar Pascual Mejia. *Didáctica de la Música para Educación Infantil*, volume 2006. 2006. ISBN 9788483223031.

[19] Edgar Willems. *Las bases psicológicas de la educación musical*. 2011. ISBN 9788449326080.

[20] Víctor Práxedes, Saavedra Rionda, Daniel Ospina, Celis Juan, Carlos Upegui, and Diana C León Torres. *DOCUMENTOS 71 Desigualdades digitales*. 2021. ISBN 9789585597860.

[21] M Franco-Avellaneda and T Arboleda. Apropiación social de la ciencia, tecnología e innovación para el desarrollo humano. *Bogotá DC: Escuela Virtual PNUD-Colciencias/Diplomado ASCTI*, 2014.

[22] Karim Barkati and Pierre Jouvelot. Synchronous programming in audio processing: A lookup table oscillator case study. *ACM Computing Surveys*, 46(2), 2013. ISSN 03600300. doi: 10.1145/2543581.2543591.

[23] Ge Wang, Ajay Kapur, Perry Cook, and Spencer Salazar. *Programming for Musicians and Digital Artists*. 2014. ISBN 9781617291708.

[24] Spencer Salazar, Ge Wang, and Perry Cook. MiniAudicle and chuck shell: New interfaces for chuck development and performance. *International Computer Music Conference, ICMC 2006*, pages 63–66, 2006.

[25] Spencer Salazar and Ge Wang. MiniAudicle for iPad touchscreen-based music software programming. *Proceedings - 40th International Computer Music Conference, ICMC 2014 and 11th Sound and Music Computing Conference, SMC 2014 - Music Technology Meets Philosophy: From Digital Echos to Virtual Ethos*, pages 686–691, 2014.

[26] Samuel Aaron, Alan F. Blackwell, and Pamela Burnard. The development of Sonic Pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music, Technology and Education*, 9(1):75–94, may 2016. ISSN 17527074. doi: 10.1386/jmte.9.1.75_1.

[27] Alex Ruthmann, Jesse M. Heines, Gena R. Greher, Paul Laidler, and Charles Saulters. Teaching computational thinking through musical live coding in Scratch. *SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, (April 2014):351–355, 2010. doi: 10.1145/1734263.1734384.

[28] James Noble. Livecoding the SynthKit: Little bits as an embodied programming language. In *Proceedings - 2nd IEEE Working Conference on Software Visualization, VISSOFT 2014*, pages 40–44. Institute of Electrical and Electronics Engineers Inc., dec 2014. ISBN 9780769553054. doi: 10.1109/VISSOFT.2014.16.

[29] Angelo Fraietta, Oliver Bown, Sam Ferguson, Sam Gillespie, and Liam Bray. Rapid composition for networked devices: Happybrackets. *Computer Music Journal*, 43(2-3):89–108, 2020. ISSN 15315169. doi: 10.1162/COMJ_a_00520.

[30] Pedro Rebelo and Alain Renaud. The frequencyliator - distributing structures for networked laptop improvisation. pages 53–56, 01 2006.

[31] Gary Ferrary, Kate Mayer, Cheroe Liston, and Michael Creeden. *IPC CID Study Guide*. 2015.

[32] Roberto Ierusalimschy. Programming in Lua , Fourth Edition Roberto Ierusalimschy. 2017(build 21), 2017. doi: 10.1093/nar/gkp710.

[33] Ge Wang and Perry Cook. ChucK: A Programming Language for On-the-fly, Real-time Audio Synthesis and Multimedia. Number January, page 812, 2004. ISBN 1581138938. doi: 10.1145/1027527.1027716.