# Robot Angel. Elaboración de un entorno integrado de desarrollo open source especializado en robótica

Alejandro Roa Aparicio

Universidad EAN

Proyecto de grado

PhD. Luisa Fernanda Carvajal Diaz

5 de agosto de 2025

**Resumen**

En este documento se elaborará Robot Angel, un entorno de desarrollo integrado IDE, especializado para el desarrollo de robótica con midlewares dedicados como micro-ROS o micro-Python, bajo una documentación y licencia open source. Este IDE surge de la necesidad de un entorno especializado en robótica para reducir la curva aprendizaje en entornos académicos y/o profesionales. Lo anterior mediante automatización de rutinas, carga de midlewares especializados en robótica con un solo clic, editor de código, visualización de componentes y que cualquier persona pueda ver, utilizar, modificar y distribuir libremente el proyecto. Logrando así una democratización de la robótica para la siguiente generación de desarrolladores.

**Robot Angel. Elaboración de un entorno integrado de desarrollo open source especializado en robótica**

**Índice**

## Índice de figuras

## Índice de cuadros

## Siglas

**DAP** *Debug Adapter Protocol* 17

**GPLv3** *GNU General Public License version 3* 11, 15, 20, 23, 24, 29

**IA** Inteligencia Artificial 9, 15

**IDE** *Integrated Development Environment* 2, 8, 9, 11, 13, 15–17, 19, 21–24, 27

**IoT** *Internet of Things* 20

**LED** *Light Emitting Diode* 20

**LSP** *Language Server Protocol* 17

**MRDS** *Microsoft Robotics Developer Studio* 16

**OCDE** Organización para la Cooperación y el Desarrollo Económicos 21

**PESTEL** Político, Económico, Social, Tecnológico, Ambiental y Legal 23

**RDS** *ROS Development Studio* 9, 12, 13

**ROS** *Robot Operating System* 9

**ROS2** *Robot Operating System 2* 16, 17

**STEM** *Science, Technology, Engineering and Mathematics* 3, 8, 12, 13, 18, 26

## Introducción

"Si he logrado ver más lejos ha sido porque he subido a hombros de gigantes."

-Newton, 1687-

La ciencia siempre avanza en conjunto, nos levantamos sobre las investigaciones de nuestros compañeros, para poder ver más allá, entender mejor el mundo y desplazar ese horizonte de conocimientos. Este proceso se ha podido ver en las iteraciones de todos los desarrollos tecnológicos (Palasí Lallana, 2004), de cómo se avanzó desde la máquina Enigma de Alan Turing hasta las nuevas portátiles, lo cual nos ha permitido moldear el mundo, ya que lo entendemos cada vez más, lo que nos da la capacidad de construir lo que hace décadas solo soñábamos como humanidad.

Gracias a este pensamiento y comportamiento de la comunidad STEM, surgió el movimiento del software libre en el cual lo más importante es la comunidad y el desarrollo conjunto (Mochi Alemán, 2002). Por eso, este proyecto elaborara una iniciativa open source, un modelo de desarrollo y distribución de software en el cual el código fuente está disponible públicamente para que cualquier persona pueda verlo, utilizarlo, modificarlo y distribuirlo libremente. Este proyecto se levanta sobre otros proyectos de software libre como el entorno Theia, un framework open source para construir entornos de desarrollo integrados basados en tecnologías web. Este entorno sera usado para construir los robots con los que hace décadas soñábamos e inspirar a otros a desarrollar más iniciativas libres que generen una comunidad y desarrollo tecnológico. Ya que el conocimiento como la ciencia se desarrolla y se construye en conjunto.

El desarrollo de la robótica actualmente se empieza desde la educación media, usualmente con la plataforma de Arduino, una iniciativa open source que surgió en 2003 (Munera et al., 2020). A pesar de que ha tenido actualizaciones, en el mundo especialmente volátil de la tecnología, ya es una plataforma de desarrollo con muchas limitaciones. En 2016 surgió la tarjeta de desarrollo ESP32, comúnmente utilizada con el mismo entorno de desarrollo (Arduino IDE) propuesto por Arduino, pero esta tarjeta de desarrollo ofrece la posibilidad de integrar tecnologías

nuevas, como IA y softwares dedicados a la robótica avanzada, como microROS o microPython.

Aunque el ESP32 y otras placas de desarrollo como la Raspberry Pi Pico ofrecen integrar estas tecnologías (Mamani-Saico & Yanyachi, 2023) y la posibilidad de mejorar la calidad de proyectos de robótica e introducirse en temas más actuales, se hace mediante un proceso tardío e ineficiente en el que es necesario cargar paquetes por terminal y programar directamente en editores de texto de línea de comandos simple o terminales Linux. Así como el desarrollo de software, el pilar del mundo tecnologico actual, avanzo para que actualmente no sea necesario saber programar en lenguaje maquina e integrando muchos proyectos libres como IDEs que facilitan el desarrollo de software, la comunidad tiene que empezar a construir plataformas que suban el nivel mínimo del desarrollo de robótica, facilitando los procesos de educación y permitiendo a los desarrolladores y futuros ingenieros en robótica aprender de tecnologías actuales para el desarrollo de robótica, pudiendo cargar micro-ROS o MicroPython con solo oprimir un botón, y tener un entorno donde se pueda desarrollar controlando variables, códigos, nodos, algoritmos de IA y visión artificial, sin complicaciones, para un desarrollo rápido de prototipos y abrirle la puerta a todas las personas que se han cautivado al ver una imagen o un video de robótica.

Aunque existen herramientas como RDS que permiten programar y simular robots usando ROS desde la nube, y entornos como PlatformIO, que ofrecen compilación, depuración y carga eficiente de código en microcontroladores como el ESP32, ambas presentan limitaciones importantes en contextos educativos y abiertos. RDS, si bien es una solución poderosa, requiere una suscripción para acceder a sus funcionalidades avanzadas, lo que lo hace poco accesible para escuelas públicas y estudiantes con recursos limitados (Bone et al., 2022). Por otro lado, PlatformIO está más orientado a usuarios intermedios o avanzados, carece de integración nativa con entornos de simulación robótica y no ofrece una interfaz pedagógica para aprender tecnologías como microROS o visión artificial. Frente a este panorama, Robot Angel surge como una propuesta open source y modular, orientada a facilitar el desarrollo robótico desde etapas tempranas, permitiendo cargar herramientas como microROS o MicroPython de forma sencilla,

todo desde un entorno accesible. De esta manera, se busca democratizar la robótica y formar nuevas generaciones de desarrolladores que comprendan y construyan las tecnologías que en este momento solo podemos imaginar.

# Objetivos

## Objetivo General

Desarrollar un IDE open source especializado en el desarrollo robótica

## Objetivos Específicos

- Integrar el entorno Theia como núcleo del compilador y editor de código fuente dentro del IDE
- Establecer una función que permita la carga automática de middlewares como micro-ROS y MicroPython en placas de desarrollo
- Publicar el proyecto con una documentación completa y open source bajo una licencia GPLv3

## Problema

En la actualidad, la robótica se ha convertido en una herramienta clave para el desarrollo de habilidades STEM desde edades tempranas y uno de los campos de mayor atracción en ingenieros y jóvenes (Pittí Patiño et al., 2012) pero a menudo las personas interesadas se ven forzadas a abandonar estas áreas de interés por el costo de desarrollo tanto en hardware como software.

En segunda instancia las personas que superan la primera limitación al tener los recursos, se encuentran con una curva de aprendizaje pronunciada por la falta de comunidad en el ámbito. Personas que al igual de como sucede en otros nichos comparables como el desarrollo de software, de manera open source (Patiño-Toro et al., 2022), comparten y construyen plataformas que faciliten el aprendizaje y automatizan rutinas tardías, que extienden el tiempo de preparación previo al desarrollo y prototipado.

Actualmente no encontramos IDEs especializadas para robótica completamente open source, ya que existen alternativas como MatLab o RDS las cuales son cerradas o parcialmente cerradas ademas de suscripción paga o limitadas en la versión gratuita. Este panorama restringe la democratización de la robótica.

Cuando los interesados en este rubro más persistentes logran traspasar las primeras barreras, se encuentran con rutinas complejas de cargas de paquetes y librerías mediante terminales o editores de texto planos linea por linea, para poder configurar un entorno de desarrollo o acceder a midlewares especializados en robótica como micro-ROS o herramientas útiles como micro-python (Bell, 2017), proceso en el cual a menudo surgen errores de compilación o cargas incompletas.

Debido a todo lo anterior surge una problemática que plantea una curva de aprendizaje lenta para la robótica(Balich et al., 2024), desde antes de configurar el entorno de desarrollo para empezar a explorar tecnologías y temas robóticos, a parte de las opciones open source reducidas e incompletas.

## Justificación

En respuesta a este problema surge Robot Angel un IDE especializado en robótica de código abierto, disponible para que la comunidad pueda ver, modificar, mejorar y distribuir, de manera libre y gratuita para lograr la democratización de la robótica.

Robot Angel se crea bajo la iniciativa de reducir la curva de aprendizaje en el sector de la robótica, reduciendo tiempos de preparación, automatizando rutinas, ofreciendo un entorno claro y completo para desarrollar robots.

De esta manera Robot Angel se convierte en una alternativa a softwares como MatLab o RDS, bajo un concepto de código abierto y sin costo lo cual lo hace viable para escuelas publicas y jóvenes aficionados que no cuentan con los recursos para costear este tipo de softwares (Vega-Moreno et al., 2016).

El papel de la robótica en el desarrollo de la comunidad STEM se ha vuelto uno de los pilares y mejores estrategias para fomentar el desarrollo del pensamiento lógico y habilidades del siglo XXI (Tinizaray & Juarros, 2025). Robot Angel se proyecta a ser adoptado desde edades tempranas, reconociendo el valor del papel de la robótica en la educación (Zorrilla-Puerto et al., 2023), actualizando herramientas obsoletas que aún se emplean en robótica educativa y acercando a los estudiantes a tecnologías vigentes y aplicadas en la industria actual.

En este contexto, Robot Angel no solo presenta una solución técnica viable, sino también se desarrolla en la comunidad de software libre, optando por la innovación y construcción conjunta. Su desarrollo responde a la necesidad urgente de herramientas accesibles, actualizadas y libres, reduciendo la curva de aprendizaje de la robótica acompañando a las nuevas generaciones en su formación en ciencia, tecnología e ingeniería logrando así la democratización de la robótica.

# Marco Teórico

## El movimiento open source y el software libre

Las expresiones “software libre” y “código abierto” suelen confundirse, pero ambas hacen referencia a programas cuya licencia garantiza libertades de uso, estudio y modificación, pero mientras que el termino open source solo se enfoca en el apartado técnico, es decir que el código fuente este publicado y disponible en algún repositorio, el termino software libre se centra en la ética y la comunidad, en la idea de darle libertad a los usuarios para decidir que utilizan, conociendo todo lo que hay detrás y brindándoles la oportunidad de mejorarlo. Esto ocasiona que muchos proyectos sean open source y de software libre pero también existe la posibilidad de ser open source pero sin ser software libre, incluso si acuñen las mismas libertades(Bacon & Dillon, 2006). En contraposición, el software de código cerrado niega la posibilidad de acceder al código fuente y limita su modificación o redistribución.

La filosofía en la comunidad del software libre se ha construido en torno a la idea de un software mejor para hacer un mundo mejor, mediante la construcción colaborativa del software, y la libertad de escoger todo, sin secretos o prohibiciones. Centrarse en la libertad de los usuarios y nutriese de la misma ya que cuando se logra generar identidad, causa que las personas quieran aportar al grupo al que pertenecen(Bacon & Dillon, 2006).

Cuando Linus Torvalds una de las personas mas influyentes en el mundo del software y uno de los grandes lideres de la comunidad del software libre, estaba creando Linux, se encontró con un gran problema, el control de versiones, necesitaba una herramienta que le permitiera modificar el software, guardarlo y probar funciones de manera segura para el desarrollo, de ahí surgió Git, la herramienta mas popular en el mundo para controlar versiones de software y años después Tom Preston-Werner, Chris Wanstrath, PJ Hyett y Scott Chacon fundarían GitHub con Git como base, Github es una herramienta online con interfaz web que le permite a la comunidad compartir su código, modificarlo, y editarlo(Chen et al., 2025) lo cual es base fundamental para el desarrollo del software libre.

Las licencias de uso determinan hasta qué punto el código puede ser reutilizado,

modificado o distribuido. La licencia GPLv3, de tipo *copyleft*, permite modificar y redistribuir el software pero exige que los trabajos derivados se distribuyan bajo los mismos términos, es decir que todo el software que se genere derivado debe ser igual de libre. Esta licencia asegura que el código permanezca libre y ha inspirado modelos como Creative Commons(Bacon & Dillon, 2006). Existen otras licencias libres más permisivas que permiten incorporar el código en software privativo, o cerrando softwares derivados, favoreciendo la adopción industrial pero debilitando la reciprocidad.

Aunque estos términos de libre y cerrado se suelen asociar a gratuito o de pago, realmente no tienen una correlación de causalidad, ya que un software libre puede ser de pago, como los servicios de overleaf o el plan premium de github, a pesar de su código fuente este completamente disponible, o un software completamente cerrado como instagram puede ser gratuito.

**Evolución de los entornos de desarrollo en robótica**

A mediados de los 2000 apareció arduino, una iniciativa open source liderada por un profesor italiano la cual constaba en placas de desarrollo con microcontroladores (Chips programables capaces de seguir instrucciones en bucle), estas placas acompañadas de un software Arduino IDE el cual permitía programarlas e indicar diferentes acciones en sus pines de salida, haciéndola ideal para proyectos de electrónica y robótica. Debido a su origen arduino siempre tuvo un énfasis educativo, la idea de democratizar el conocimiento y la robótica, un estudio publicado en la ASEE(Nannim et al., 2025) demostró que los proyectos de robótica con arduino fomentaban y desarrollaban el pensamiento algorítmico junto con las capacidades lógico matemática.

Gracias a los bajos costos y facilidad de reproducción de las tarjetas sumado al ser una iniciativa opensource, tuvo una acogida global y hoy en día es la plataforma educativa de robótica mas grande a nivel mundial.

A medida que los proyectos de robótica exigían mayor potencia de cálculo y conectividad, surgieron placas como ESP32 y Raspberry Pi Pico las cuales soportan IA y frameworks de desarrollo. Aunque estas placas presentaban una iniciativa de hardware cerrado, mantenían un

bajo coste y compatibilidad con Arduino IDE, dándole aun mas fuerza a Arduino IDE pero este seguía siendo muy básico para explotar las nuevas capacidades de las placas de desarrollo mas modernas.

Surgieron nuevas iniciativas para explotar las ventanas que abrieron las placas modernas pero estas iniciativas fueron de software cerrado y/o pagas como MATLAB/Simulink o MRDS fueron estándares en robótica educativa para quienes tenian los recursos generando una brecha con los que no los poseian. Un estudio sobre educación en mecatrónica remarca que las licencias de productos comerciales representan costos elevados para las instituciones y que los estudiantes pierden acceso al software al graduarse(Lotfi et al., 2022). Además, la dependencia de plataformas privativas restringe la personalización y dificulta la transición hacia otras tecnologías. Por estas razones, la comunidad ha migrado hacia alternativas libres y multiplataforma. Plataformas como PlatformIO se posicionan como entornos de desarrollo modernos, aunque su adopción académica aún es limitada sin contar sus oportunidades de mejora como la adopción de frameworks de manera nativa.

**Middleware y frameworks en el desarrollo robótico**

La robótica moderna se apoya en patrones de diseño distribuidos en los que distintos módulos deben comunicarse. Un *middleware* actúa como “pegamento” que proporciona infraestructura de comunicación y gestión de hardware, permitiendo que componentes heterogéneos trabajen juntos, es decir nodos escritos en diferentes lenguajes y tipos de archivos interactuando en diversas placas, permitiendo la construcción de robots con lo mejor de cada herramienta. Un framework robótica es un conjunto de herramientas, bibliotecas y convenciones que simplifican el desarrollo de robots complejos e imponen principios arquitectónicos de micro servicios, mientras que el middleware se centra en la infraestructura de comunicación la cual es sumamente importante en el desarrollo robótico ya que permite que todo funcione con la precisión y efectividad diseñada(Tsardoulias & Mitkas, 2017).

Este panorama moderno propuso las condiciones para que surgieran middlewares como *micro-ROS* el cual permite llevar la flexibilidad de ROS2 a microcontroladores con recursos

limitados. La tesis de Mälardalen University explica que *micro-ROS* permite ejecutar cálculo y comunicación dentro de los microcontroladores, integrándose de forma transparente con ROS2 y ofreciendo un rendimiento comparable con baja latencia(Nguyen, 2022). Financiado por el proyecto europeo *OFERA*, este middleware abre la puerta a robots distribuidos en los que nodos de muy bajo consumo y coste participan en redes colaborativas.

El lenguaje MicroPython complementa esta tendencia al ofrecer una implementación eficiente de Python para microcontroladores. La documentación oficial destaca que MicroPython está escrito en C99, es de código abierto bajo licencia MIT y ofrece un intérprete interactivo con funciones avanzadas como enteros de precisión arbitraria y comprensiones de listas(George & contributors, 2025). Su reducido tamaño y compatibilidad con Python estándar permiten a programadores novatos trasladar sus conocimientos del escritorio a dispositivos embebidos sin una curva de aprendizaje pronunciada e incluso llegar a usar herramientas como métodos básicos de openCv para ejecutar visión artificial.

Un framework robótico proporciona herramientas, bibliotecas y convenciones que simplifican el desarrollo y suelen imponer principios arquitectónicos, mientras que un entorno de desarrollo integrado (IDE) se centra en ofrecer edición de código, depuración y compilación. El middleware suele ser un componente del framework, pero este último incluye además API de alto nivel y servicios que estructuran la aplicación(Tsardoulias & Mitkas, 2017), estos frameworks requieren instalaciones adicionales en los IDEs y sistemas operativos que pueden ser tardías y complejas, susceptibles a errores humanos en los pasos de instalación y compilación.

Theia es una plataforma modular y de código abierto para construir IDEs de escritorio o en la nube. La documentación oficial resalta que se concibió como un proyecto neutral que reutiliza tecnologías modernas como los protocolos LSP y DAP; su arquitectura permite ejecutar una misma base de código tanto en web como en escritorio. Empresas como Arduino, Red Hat y Samsung han adoptado Theia para crear IDEs personalizados(Eclipse Foundation, 2025). Este caso demuestra cómo las herramientas libres permiten desarrollar entornos adaptados a necesidades específicas sin perder compatibilidad con estándares.

**Aspectos pedagógicos: robótica y educación STEM**

La robótica se considera un elemento integrador dentro del aprendizaje STEM porque combina programación, electrónica y diseño mecánico en proyectos concretos lo cual fomenta el pensamiento algorítmico y desarrollo de capacidades logico matemáticas. Un estudio sobre aprendizaje STEM integrado destaca que las actividades robóticas desarrollan habilidades del siglo XXI como la comunicación, la colaboración, la creatividad y la resolución de problemas(Budiyanto et al., 2024). Asimismo, la iniciativa INBOTS de la Unión Europea sostiene que la robótica educativa debe ser inclusiva y emplearse como herramienta para enseñar ciencias y humanidades a todos los estudiantes desde la educación media, y no solo a futuros ingenieros(Alimisis, 2021), debido a los múltiples beneficios que trae.

La incorporación de la robótica en contextos académicos y profesionales implica una curva de aprendizaje pronunciada. Muchos planes de estudio escolares se limitan a guías paso a paso que impiden la creatividad y la experimentación, y subraya que se requieren recursos y formación docente para adoptar la robótica(Alimisis, 2021; Budiyanto et al., 2024). En el ámbito profesional, la rápida evolución de frameworks, lenguajes y hardware obliga a los ingenieros a actualizarse constantemente; la transición desde entornos educativos como Arduino hacia sistemas más complejos refleja esa curva continua de aprendizaje y resalta la importancia de la formación permanente.

Uno de los objetivos de la educación sobre la industria robótica es democratizar el acceso a estas tecnologías. El movimiento del software libre reduce los costos de licencias y fomenta la reutilización; esto se traduce en plataformas de bajo coste como Arduino y Raspberry Pi, así como en licencias GPL que protegen la libertad de uso(Lotfi et al., 2022). La iniciativa INBOTS aboga por que la robótica educativa sea inclusiva, desmitificando los robots y haciendo que niños y adultos se sientan capacitados para interactuar con ellos(Alimisis, 2021). No obstante, la democratización también requiere abordar desigualdades económicas y de infraestructura; programas de bajo coste y recursos comunitarios son clave para que la robótica deje de ser un privilegio y se convierta en una herramienta al alcance de todos.

## Análisis de requerimientos

A continuación se presenta un cuadro con los requerimientos del proyecto (Cuadro 1).

**Cuadro 1**

*Requerimientos del proyecto*

| ID | Requerimiento | Tipo |
|---|---|---|
| RQ-01 | Acceso al modelo open source Theia como núcleo para el IDE. | Software |
| RQ-02 | Acceso a Python y a middlewares especializados (micro-ROS, microPython) para automatizar sus rutinas de carga a placas de desarrollo. | Software |
| RQ-03 | Conocimiento en entornos Linux y uso de la terminal para automatizar rutinas. | Conocimiento |
| RQ-04 | Conocimiento en control de versiones con Git/GitHub para la documentación. | Conocimiento |
| RQ-05 | Equipo con sistema operativo Windows 10/11 para validación del IDE. | Hardware |
| RQ-06 | Equipo con sistema operativo Linux (Ubuntu/Debian) para desarrollo principal y validación del IDE. | Hardware |
| RQ-07 | Placa de desarrollo ESP32 para validar la carga y funcionamiento de micro-ROS y microPython. | Hardware |
| RQ-08 | Placa de desarrollo con Raspberry Pi Pico como hardware alternativo de validación. | Hardware |

| ID | Requerimiento | Tipo |
|---|---|---|
| RQ-09 | Sensores y actuadores básicos (LEDs, servo motores y cámara) para validación. | Hardware |
| RQ-10 | Disponibilidad de conectividad WiFi y Bluetooth para pruebas de IoT. | Hardware |
| RQ-11 | Disponibilidad de licencia open source GPLv3 para distribución libre. | Legal/Operativo |

## Restricciones

### Factores políticos

El gobierno Colombiano maneja políticas de compra y alianzas con empresas privadas pueden condicionar la tecnología utilizada en las aulas. El informe de la OCDE señala que la política digital colombiana sufre de falta de alineación entre las iniciativas de puntos digitales y los programas de formación, lo que evidencia vacíos regulatorios y de gobernanza(for Economic Co-operation & Development, 2019). En ausencia de marcos jurídicos específicos, las entidades educativas deben navegar una normativa general sobre adquisiciones y protección de datos que a veces favorece software comercial. Por tanto, aunque no existan restricciones políticas explícitas contra el software educativo, la normativa vigente y las prácticas de contratación pública pueden actuar como barreras al uso de plataformas alternativas.

### Factores económicos

En el ámbito económico nuevamente no encontramos restricciones directas ya que para la realización del proyecto solo se necesita un equipo que maneje sistema operativo linux y windows con acceso a internet, pero se encontró una barrera de entrada para la adopción de Robot Angel la cual radica en los costos asociados al hardware, la conectividad y el acceso a internet. Aunque el IDE sea de libre distribución, el uso de placas de desarrollo, sensores y componentes básicos implica una inversión inicial que puede resultar significativa para instituciones educativas públicas o estudiantes en contextos de bajos recursos. Además, la brecha digital en Colombia se refleja en la desigualdad en la calidad y costo de los servicios de conectividad (Lotfi et al., 2022). En consecuencia, el componente económico se presenta como una barrera.

No se puede negar la existencia de esta barrera, sin embargo hay que aclarar que es algo inherente a la robótica, no hay una opción de desarrollo y prototipado sin inversión de hardware y al hacer este IDE compatible con las placas de desarrollo mas económicas del mercado, esta barrera se reduce

**Factores sociales**

Desde la perspectiva social, la percepción de las plataformas libres y abiertas puede constituir un obstáculo. En muchos entornos académicos y profesionales se asocia lo libre y mas aun cuando es gratuito con baja calidad o falta de soporte técnico, lo que genera resistencia frente a la adopción de alternativas como Robot Angel frente a softwares comerciales consolidados. A esto se suma la necesidad de formación docente y/o empresarial: los profesores u empleados requieren tiempo y capacitación para apropiarse de nuevas herramientas, lo cual no siempre está contemplado en los programas de enseñanza o cronogramas empresariales. Estas condiciones sociales generan una barrera cultural y de aceptación que puede ralentizar la implementación del IDE, aun cuando este ofrezca ventajas técnicas y pedagógicas claras (Patiño-Toro et al., 2022).

**Factores tecnológicos**

En cuanto a la tecnología la principal restricción se relaciona con las capacidades de los dispositivos utilizados en contextos educativos. Aunque el IDE esté diseñado para ejecutarse en sistemas Linux y Windows, su rendimiento depende del acceso a computadores con suficiente capacidad de procesamiento, memoria y conectividad. De igual forma, la diversidad de placas de desarrollo y la falta de estandarización en protocolos o librerías puede derivar en incompatibilidades que afecten la experiencia de uso. Este panorama técnico obliga a implementar procesos de validación y adaptación constantes, así como a garantizar una curva de actualización para incorporar nuevos middlewares o frameworks que surjan en el ecosistema robótico (Nguyen, 2022).

**Factores ambientales**

No existe ninguna restricción ambiental directa hacia la elaboración y distribución de IDEs o software libre, pero si existe una barrera de adopción con consecuencias ambientales la cual debe de tomarse en cuenta para el desarrollo del proyecto.

La mayor barrera en el ámbito ambiental radica en el mismo desarrollo de robótica que fomenta el proyecto, ya que este genera muchos desechos electrónicos como placas de desarrollo, componentes, e insumos (Sáenz & otros, 2024) y sin un debido manejo estos pueden contribuir a

problemas graves de contaminación ambiental.

**Factores legales**

En Colombia no existe ninguna restricción legal directa hacia la elaboración y distribución de IDEs o software libre, pero si existe una legislación vigente sobre la protección de datos y distribución digital, las cuales establecen un marco claro en el que Robot Angel se tiene que ubicar.

La ley 581 (Ley 1581 de 2012, 2012) establece una política de protección de datos personales general, la cual se debe tener en cuenta a la hora de redactar y garantizar que se cumpla el consentimiento informado de privacidad en caso de que se recolecte algún tipo de dato.

La ley 527 (Ley 527 de 1999, 1999) asegura la validez jurídica de la licencia de uso para su distribución, si esta esta bien definida, en el caso del proyecto la GPLv3.

**Conclusión**

Al analizar las restricciones propuestas observamos que Robot Angel no cuenta con una restricción directa en ninguno de los ámbitos propuestos por el PESTEL, pero si se encuentra con varias barreras de entrada, las cuales podrían ser superadas con un debido manejo.

## Metodología

Este proyecto abordara una metodología de acción-resultado, en la que se propone una acción que al llevar a cabo su ejecución da como resultado el cumplimiento de cada uno de los objetivos específicos del proyecto.

A continuación se presenta un cuadro con las acciones que se realizaran para cumplir cada uno de los objetivos del proyecto (Cuadro 2).

**Cuadro 2**

*Acciones metodológicas y resultados esperados del proyecto*

| **Objetivo específico** | **Acción metodológica** | **Resultado esperado** |
|---|---|---|
| Integrar Theia como núcleo del IDE | Configuración y adaptación de Theia como núcleo del IDE, con una capa de personalización, que incluye extensiones para microcontroladores e interfaz gráfica, añadiendo la validación del correcto funcionamiento en Linux y Windows | Prototipo funcional del editor y compilador |
| Establecer función de carga automática de middlewares | Desarrollo de scripts de instalación para microROS y Micro-python en placas de desarrollo a demás de las respectivas pruebas en ESP32 y Raspberry Pi Pico | Automatización de carga de micro-ROS y MicroPython |
| Publicar el proyecto bajo GPLv3 | Creación de repositorio en GitHub y publicación bajo licencia GPLv3 de código, manuales y guías de uso | IDE disponible con documentación completa open source |

## Costos

La ejecución de Robot Angel implica una inversión en recursos humanos y técnicos necesarios para la materialización del IDE. Si bien no se busca un retorno económico directo debido a su ideología de software libre, los costos asociados se justifican como parte del proceso de investigación, desarrollo y validación técnica del proyecto.

El desarrollo se realizó empleando un equipo Lenovo Legion 5, con un costo de mercado aproximado de 5 000 000 COP, utilizado para las pruebas en sistemas Linux y Windows mediante un dual boot. El proyecto demandó aproximadamente 100 horas de desarrollo, distribuidas entre programación, diseño de interfaz y pruebas de integración de middlewares especializados en robótica.

Los costos de desarrollo del proyecto fueron estimados en:

1. **Consumo energético:** considerando una tarifa promedio de energía eléctrica de 700 COP/kWh para estrato 3 en Colombia y un consumo aproximado de 0.12 kWh por hora, se obtiene un costo total de 6 720 COP para las 80 horas de trabajo.
2. **Conectividad y espacio de trabajo:** el acceso a Internet y el uso de un entorno adecuado de programación representan un costo aproximado de 600 000 COP, incluyendo servicios básicos y mantenimiento del espacio de trabajo.
3. **Salario del desarrollador:** valorando la hora técnica del desarrollador principal en 50 000 COP, correspondiente a un estimado mensual de 5 000 000 COP, el costo total asociado a las 100 horas de trabajo asciende a 5 000 000 COP.

A continuación, se presenta un resumen de los costos estimados (Cuadro 3).

**Cuadro 3**

*Justificación de costos del proyecto*

| **Concepto** | **Costo estimado (COP)** |
|---|---|
| Equipo Lenovo Legion 5 (infraestructura de desarrollo) | 5 000 000 |
| Energía eléctrica (80 h) | 6 720 |
| Conectividad y espacio de trabajo | 600 000 |
| Desarrollo (100 h $\times$ 50 000 COP) | 5 000 000 |
| **Total estimado** | **10 606 720** |

Si bien el proyecto no contempla la recuperación de la inversión de cerca de 10 millones de pesos colombianos mediante mecanismos comerciales, su valor se traduce en un impacto social, educativo y tecnológico, al contribuir a la democratización de la robótica, fomentar el acceso equitativo a la tecnología y fortalecer la formación práctica en ingeniería y software libre. En este sentido, los costos representan una inversión en desarrollo académico e innovación abierta del país, lo cual es coherente con la filosofía del movimiento *open source* y los principios de la educación STEM.

## Resultados

En este apartado se presentan los resultados obtenidos sobre Robot Angel.

En primera instancia tenemos el cumplimiento total del objetivo planteado junto a sus tres objetivos específicos, logrando así desarrollar un IDE open source especializado en el desarrollo robótica.

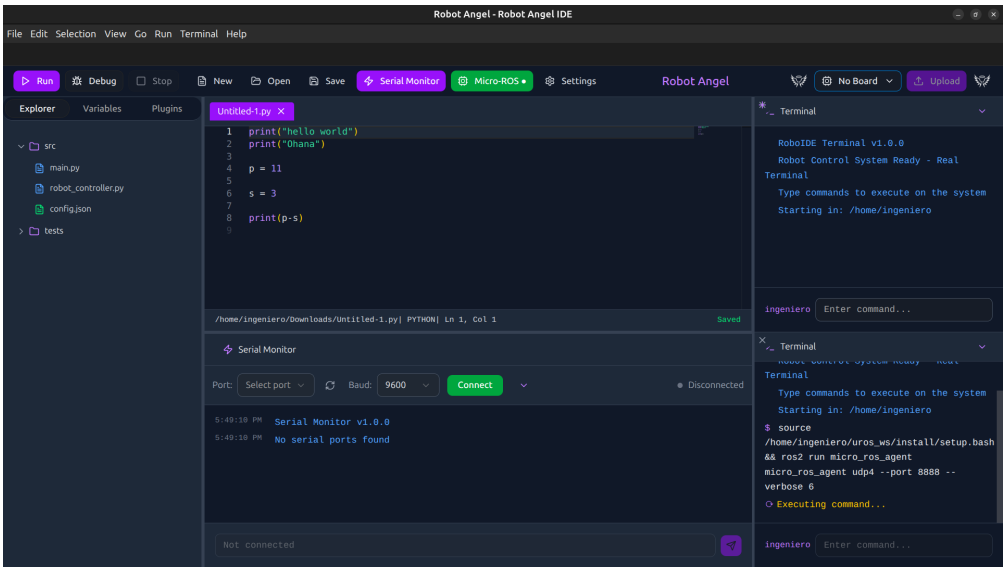


**Figura 1**

*Robot Angel IDE completamente funcional*

En orden con los objetivos específicos se logro integrar el entorno Theia como núcleo del compilador y editor de código fuente dentro del IDE, obteniendo soporte multi lenguaje.

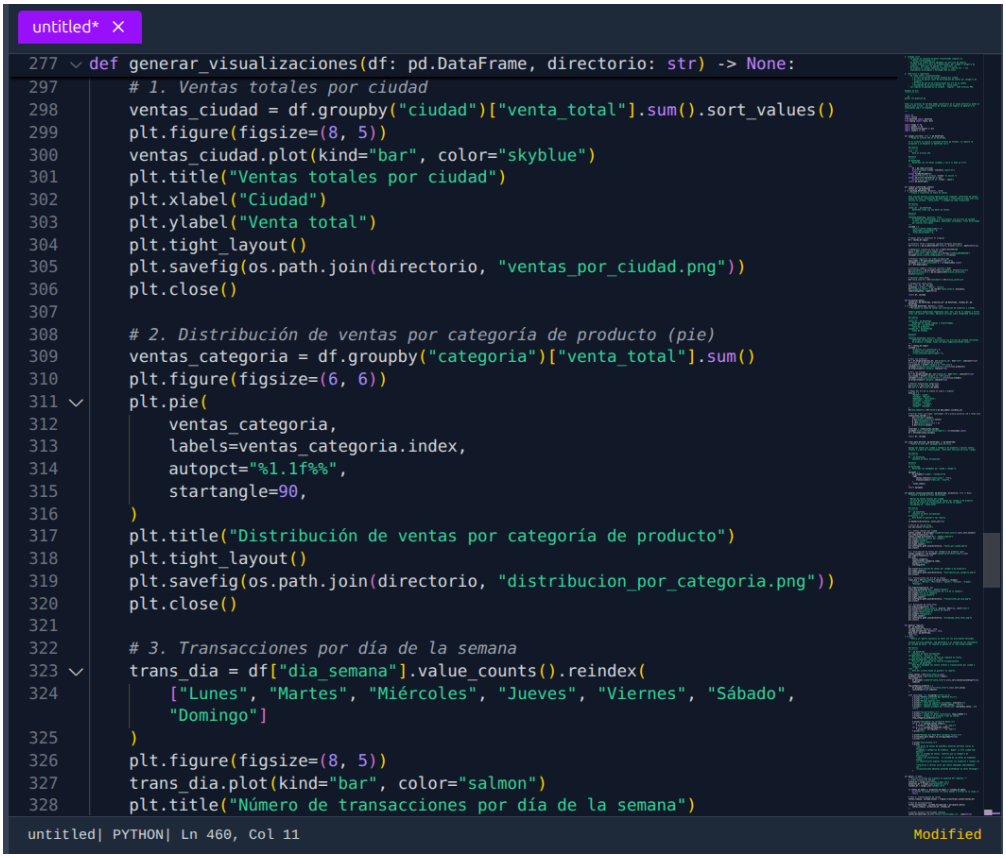


**Figura 2**

*Editor de código multi lenguaje*

En consecuencia del objetivo especifico de establecer una función que permita la carga automática de middlewares como micro-ROS y MicroPython en placas de desarrollo, encontramos la función automatizada de la carga.

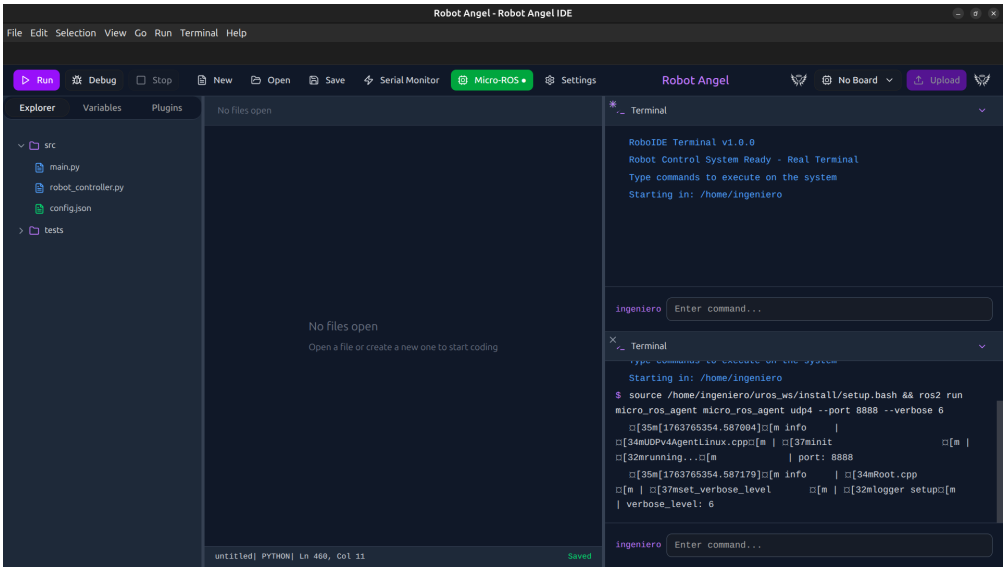


**Figura 3**

*Agente de micro ROS activado*

Por ultimo en consecuencia al objetivo publicar el proyecto con una documentación completa y open source bajo una licencia GPLv3 presentamos un repositorio en github el cual puedes visitar en este enlace para más información, donde se subio todo el desarrollo bajo una licencia GPLv3 con sus respectivos archivos de licencia y .md para que la comunidad pueda descubrir el proyecto, aprender a usarlo y contribuir para mejorarlo.

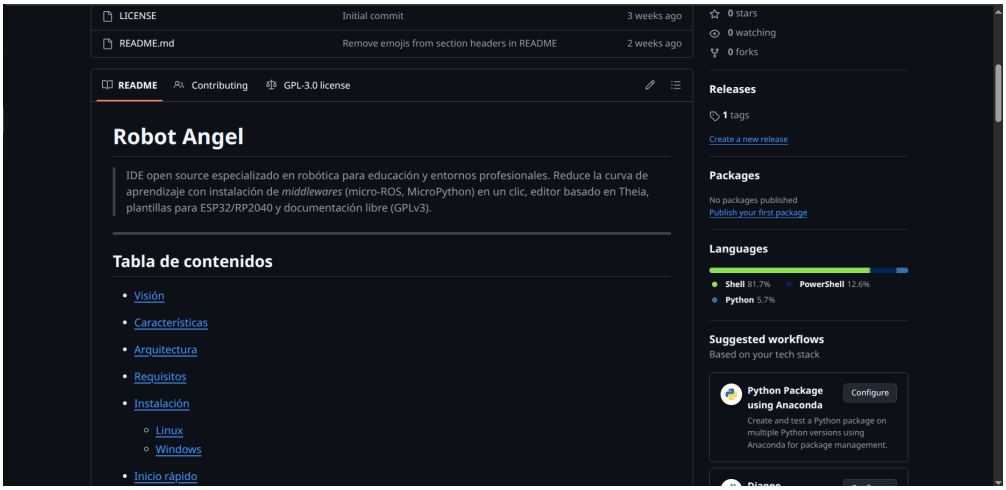


**Figura 4**

*Repositorio de github con toda la documentación.*

Paralelo a todos los resultados se desarrollo una landing page sobre el proyecto para una mayor difusión y que los usuarios finales tanto de windows como linux puedan descargar sus instaladores con un solo clic en este enlace.

## Conclusiones

En primer lugar el desarrollo de Robot Angel permitió cumplir plenamente los objetivos planteados en el proyecto, consolidando un IDE open source funcional, multiplataforma y especializado en robótica. Se integró Theia como núcleo del editor, se automatizaron procesos complejos como la instalación y carga de middlewares (micro-ROS y MicroPython), y se publicó la plataforma bajo licencia GPLv3 con documentación completa, evidenciando rigurosidad técnica y coherencia con el marco teórico del movimiento open source. Más allá de un simple prototipo, Robot Angel demostró ser una solución viable y académicamente sólida para reducir la curva de aprendizaje en robótica, aportando un entorno moderno, accesible y alineado con las necesidades actuales de la educación STEM.

Si bien los resultados alcanzados constituyen un avance significativo, el proyecto abre oportunidades claras para su evolución. Entre ellas, se encuentra la integración de un visor URDF, que permitiría inspeccionar modelos robóticos en 3D directamente desde el IDE, así como el avance hacia una versión completamente web, coherente con el potencial de Theia para entornos cloud. Estas mejoras no solo ampliarían la funcionalidad del IDE, sino que lo convertirían en un ecosistema integral comparable a plataformas como RDS, pero completamente libre. Las ideas presentadas en la landing del proyecto —como la expansión modular, nuevos toolchains y un catálogo de plantillas para ESP32 y Raspberry Pi Pico— marcan una hoja de ruta clara para seguir fortaleciendo el impacto educativo y profesional del IDE. Teniendo la oportunidad de convertirse en el IDE especializado para robótica número 1 a nivel mundial si se le da 1 o 2 años de desarrollo y la difusión correcta.

Robot Angel aporta directamente al fortalecimiento de la ingeniería al promover la democratización del conocimiento, la robótica y la construcción colectiva de herramientas abiertas. En un contexto donde gran parte del software especializado es cerrado o de alto costo, ofrecer un IDE libre, documentado y extensible representa un aporte tangible para futuros ingenieros, instituciones educativas y desarrolladores autodidactas. Como proyecto nacido desde la comunidad STEM, recuerda que la ingeniería avanza cuando cada integrante retorna a la

comunidad parte de lo aprendido: compartiendo código, documentando, educando y manteniendo vivo el espíritu del software libre. Robot Angel no es solo un IDE, es un llamado a construir una comunidad que mejore, enseñe y expanda el proyecto para que nuevas generaciones puedan ir más lejos aún.

## Referencias

Alimisis, D. (2021). Technologies for an inclusive robotics education. *Open Research Europe*, *1*, 40. https://doi.org/10.12688/openreseurope.13321.2

Bacon, S., & Dillon, T. (2006). *The potential of open source approaches for education* (inf. téc.) (Opening Education series). Futurelab. https://www.nfer.ac.uk/media/z0anmnpv/futl58.pdf

Balich, N., Balich, F., Ocampo, T., & Balich, B. (2024). Desarrollo de simulador de robótica cloud multipresencia para enseñanza de programación. *XXX Congreso Argentino de Ciencias de la Computación (CACIC)(La Plata, 7 al 11 de octubre de 2024)*.

Bell, C. (2017). MicroPython for the Internet of Things. *Apress: New York, NY, USA*.

Bone, M. A. P., Bravo, A. V. S., Herrera, V. M. S., & Vera, L. R. M. (2022). Software libre vs Software privativo: su implicación en la educación del siglo XXI. *Revista Científica Multidisciplinaria Ogma*, *1*(2), 62-73.

Budiyanto, C. W., Arafat, M. H., Yuana, R. A., & Fenyvesi, K. (2024). The Role of Educational Robotics in Integrated STEM Learning Towards the Formation of 21st-Century Skills [EP2024 regular session]. *Proceedings of the Asian Technology Conference in Mathematics*. https://atcm.mathandtech.org/EP2024/regular/22157.pdf

Chen, K. Y., Toro-Moreno, M., & Subramaniam, A. R. (2025). GitHub enables collaborative and reproducible laboratory research. *PLoS Biology*, *23*(2), e3003029. https://doi.org/10.1371/journal.pbio.3003029

Eclipse Foundation. (2025). *Theia — Open and extensible platform for desktop and cloud IDEs* [Accessed 2025-09-11]. https://theia-ide.org/

for Economic Co-operation, O., & Development. (2019). *OECD Reviews of Digital Transformation: Going Digital in Colombia* (Disponible en https://doi.org/10.1787/781185b1-en). OECD Publishing. Paris.

George, D. P., & contributors. (2025). *MicroPython — Python for microcontrollers* [Accessed 2025-09-11]. https://micropython.org/

Ley 1581 de 2012, Colombia (2012). https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

Ley 527 de 1999, Colombia (1999). https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=340

Lotfi, N., Auslander, D., Rodriquez, L., Mbanisi, K., & Berry, C. A. (2022). Use of open-source software in mechatronics and robotics engineering education – Part I: Model simulation and analysis [Accessed 2025-09-11]. *ASEE Computers in Education Journal*, *12*(1). https://coed.asee.org/2021/11/30/use-of-open-source-software-in-mechatronics-and-robotics-engineering-education-part-i-model-simulation-and-analysis/

Mamani-Saico, A., & Yanyachi, P. R. (2023). Implementation and performance study of the micro-ros/ros2 framework to algorithm design for attitude determination and control system. *IEEE Access*, *11*, 128451-128460.

Mochi Alemán, P. Ó. (2002). El movimiento del software libre. *Revista Mexicana de Ciencias Políticas y Sociales*, *45*(185), 73-89.

Munera, J. M., Jimenez, A., Botero, M. A., Rivas, K. Y., & Lopez, J. (2020). La educación moderna al alcance de arduino. *Revista Espacios*, *798*, 1015.

Nannim, F. A., Ibezim, N. E., Mosia, M., & Oguguo, B. C. E. (2025). Project-based learning with Arduino robots: impact on undergraduate students' achievement and task persistence in robotics programming. *Frontiers in Robotics and AI*, *12*, 1615427. https://doi.org/10.3389/frobt.2025.1615427

Newton, I. (1687). *Philosophiæ Naturalis Principia Mathematica* ["Si he logrado ver más lejos ha sido porque he subido a hombros de gigantes."]. Royal Society.

Nguyen, P. (2022). *Micro-ROS for Mobile Robotics Systems* [Master's thesis]. Mälardalen University. https://mdh.diva-portal.org/smash/get/diva2:1670378/FULLTEXT01.pdf

Palasí Lallana, V.-R. (2004). Modelos de desarrollo iterativos. *Realidad y Reflexión, 2004, Año. 4, núm. 12, p. 61-67.*

Patiño-Toro, O. N., Valencia-Arias, A., Gomez-Molina, S., & Bermeo-Giraldo, M. C. (2022). Open-Source software adoption among university students in emerging countries. *IEEE Revista Iberoamericana de Tecnologias Del Aprendizaje*, *17*(2), 185-196.

Pittí Patiño, K., Moreno, I., Muñoz, L., Serracín, J. R., Quintero, J., Quiel, J., et al. (2012). La robótica educativa, una herramienta para la enseñanza-aprendizaje de las ciencias y las tecnologías.

Sáenz, J., & otros. (2024). Automated disassembly of e-waste — requirements on modeling of disassembly processes and product states. *Frontiers in Robotics and AI*, *11*(1303279). https://doi.org/10.3389/frobt.2024.1303279

Tinizaray, F. S. Z., & Juarros, V. I. M. (2025). Evaluación del aprendizaje basado en proyectos STEAM mediado por robótica educativa para el desarrollo de habilidades del siglo XXI. *RiiTE Revista interuniversitaria de investigación en Tecnología Educativa*, (18), 68-90.

Tsardoulias, E., & Mitkas, P. (2017). Robotic frameworks, architectures and middleware comparison. *arXiv preprint arXiv:1711.06842*. https://doi.org/10.48550/arXiv.1711.06842

Vega-Moreno, D., Solé, X. C., Rueda, M. J., Llinás, D., et al. (2016). Integración de robótica educativa de bajo coste en el ámbito de la educación secundaria para fomentar el aprendizaje por proyectos. *IJERI: International Journal of Educational Research and Innovation*, (6), 162-175.

Zorrilla-Puerto, J., Lores-Gómez, B., Martínez-Requejo, S., & Ruiz-Lázaro, J. (2023). El papel de la robótica en Educación Infantil: revisión sistemática para el desarrollo de habilidades. *RiiTE Revista interuniversitaria de investigación en Tecnología Educativa*, 188-194.