UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMATICA, CIENCIAS DE LA COMPUTACION E INNOVACION TECNOLOGICA

CARRERA DE SOFTWARE

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL CONTROL DE UN ROBOT AUTÓNOMO QUE RESUELVE CUBOS RUBIK 3X3 UTILIZANDO IMPRESIÓN 3D Y ALGORITMOS DE VISIÓN COMPUTACIONAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

AUTOR: BORIS ADRIAN LOPEZ ROJAS

DIRECTOR: ING. JUAN PABLO PAZMIÑO PIEDRA, MGS.

CUENCA – ECUADOR

2025

DIOS, PATRIA, CULTURA Y DESARROLLO

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMATICA, CIENCIAS DE LA COMPUTACION E INNOVACION TECNOLOGICA

CARRERA DE SOFTWARE

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL CONTROL DE UN ROBOT AUTÓNOMO QUE RESUELVE CUBOS RUBIK 3X3 UTILIZANDO IMPRESIÓN 3D Y ALGORITMOS DE VISIÓN COMPUTACIONAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

AUTOR: BORIS ADRIAN LOPEZ ROJAS

DIRECTOR: ING. JUAN PABLO PAZMIÑO PIEDRA, MGS.

CUENCA – ECUADOR

2025

DIOS, PATRIA, CULTURA Y DESARROLLO

DECLARATORIA DE AUTORÍA Y RESPONSABILIDAD

Boris Adrián López Rojas portador(a) de la cédula de ciudadanía N.° **0106741978**. Declaro ser el autor de la obra: "**DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL CONTROL DE UN ROBOT AUTÓNOMO QUE RESUELVE CUBOS RUBIK 3X3 UTILIZANDO IMPRESIÓN 3D Y ALGORITMOS DE VISIÓN COMPUTACIONAL**", sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Cuenca, **25 de abril de 2025**

F: .....

Boris Adrián López Rojas

C.I. **0106741978**

Universidad Católica de Cuenca

CERTIFICO

Certifico que el presente Articulo Científico fue desarrollado por **Boris Adrian Lopez Rojas**, con el Tema "**DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL CONTROL DE UN ROBOT AUTÓNOMO QUE RESUELVE CUBOS RUBIK 3X3 UTILIZANDO IMPRESIÓN 3D Y ALGORITMOS DE VISIÓN COMPUTACIONAL**", bajo mi supervisión.

Ing. Juan Pablo Pazmiño P.
C.I: 0102826195

DEDICATORIA

Dedico este trabajo a quienes me amaron incluso en mis silencios.

A mi familia, por ser el faro en cada tormenta.

A quienes ya no están, pero siguen presentes en mi memoria y en mi corazón, porque de alguna forma, también caminaron conmigo hasta aquí.

Y a mí mismo, por no rendirme cuando el camino se volvió incierto. Este logro también es un acto de amor propio.

AGRADECIMIENTOS

A lo largo de este camino no he estado solo, y este logro no sería posible sin la presencia, el apoyo y el amor de muchas personas que me acompañaron, creyeron en mí y me sostuvieron en los momentos difíciles.

Gracias, primero, a mi familia, por ser mi base, mi motor y mi refugio. A mis padres, por enseñarme con el ejemplo que la constancia y la humildad abren caminos. A [nombre de hermanos/as, si aplica], por sus palabras sencillas que a veces sanaron más que cualquier teoría.

A mis docentes y mentores, por compartir su conocimiento con paciencia y pasión. Gracias por despertar en mí la curiosidad y por alentarme a ir siempre un paso más allá.

A mis amigos y compañeros, con quienes compartí desvelos, risas, dudas y certezas. Este trabajo lleva también algo de ustedes: una conversación, una mirada de aliento, una taza de café a deshoras.

A todos los que creyeron en mí cuando yo dudaba, gracias por recordarme quién soy.

RESUMEN

En estos momentos muchos avances tecnológicos están en el auge de sus objetivos, el realizar la vida de los seres humanos menos compleja, resolviendo problemas y dando conclusiones de su uso y sus beneficios en una sociedad cada vez más automatizada. Para tomar como ejemplo realizamos este proyecto para abordar la falta de capacidades en un ser humano que cuenta con discapacidades ya sea de aprendizaje o en sus extremidades, es decir no poder desarrollar movimientos que requieren de ayuda motriz como los dedos al resolver un cubo Rubik 3x3.

Para aquello optamos por realizar este prototipo junto a la metodología Modelo en V + SCRUM, detallando los resultados como correctos al momento de usar este artefacto para que resuelva nuestro juguete Rubik 3x3; detallando como objetivo la retroalimentación que obtiene el público usuario, incluso cuando tenemos una discapacidad como falta de extremidades o algún otro contratiempo para completar completamente el puzle del cubo Rubik, dando como resultado un cubo Rubik resuelto solamente con colocarlo en el prototipo de frente a una cámara web y presionar un botón.

Palabras clave: *automatización, discapacidad, prototipo robótico, cubo Rubik 3x3, Visión por Computadora.*

ABSTRACT

Currently, many technological advancements are reaching their peak objectives: to simplify human life, solve problems, and provide insights into their use and benefits in an increasingly automated society. For example, we undertook this project to address the lack of capabilities in individuals with disabilities, whether related to learning or their extremities; that is, the inability to perform movements requiring motor assistance, such as the fingers when solving a 3x3 Rubik's Cube.

To that end, we chose to develop this prototype using the V-Model + SCRUM methodology, with results considered correct when the device is used to solve our 3x3 Rubik's Cube. Our objective is to get feedback from the users, including those with disabilities such as missing limbs or other setbacks that prevent them from fully completing the Rubik's Cube puzzle, resulting in a solved Rubik's Cube by simply placing it in the prototype in front of a webcam and pressing a button.

Keywords: *Automation, disability, robotic prototype, 3x3 Rubik's Cube, computer vision.*

INDICE DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

LISTA DE ANEXOS

CAPITULO I

1. MARCO REFERENCIAL

En este capítulo se presenta el marco referencial que sustenta el desarrollo del presente proyecto. Se describen el problema abordado, su justificación, los objetivos propuestos, el alcance del trabajo, los conceptos fundamentales relacionados con la temática, así como los trabajos similares existentes que aportan al análisis y validación del enfoque adoptado, incluso cuando tenemos dificultades e intentamos concretar lo impulsado por el conocimiento tecnológico.

1.1. INTRODUCCION

En un mundo donde la tecnología avanza a pasos agigantados, surge un reto fascinante: combinar la visión artificial, la mecánica de precisión y la inteligencia algorítmica para resolver pequeños desafíos como el cubo Rubik de manera totalmente automatizada. Este proyecto nace de la pasión por la robótica, la programación y los rompecabezas, con el sueño de diseñar un sistema que pueda capturar imágenes del cubo, reconocer cada color con precisión y manipularlo usando servomotores hasta resolverlo, todo sin intervención humana. Como mencionan algunos expertos en el área, “la integración de la robótica y la visión artificial nos permite enfrentar problemas complejos de forma eficiente y autónoma, abriendo nuevas oportunidades en distintos campos” [6].

Hoy en día, el uso de tecnologías emergentes ocupa un lugar clave en la educación. La robótica, la programación y la electrónica se han vuelto herramientas esenciales para desarrollar en los estudiantes habilidades como el pensamiento lógico, la creatividad y la resolución de problemas. El cubo Rubik, conocido tanto por su dificultad como por su valor educativo, ha sido utilizado en diferentes espacios para enseñar conceptos como algoritmos, secuencias y estructuras. Con este proyecto, se busca llevar el clásico desafío del cubo Rubik a otro nivel, integrándolo con sistemas automatizados que combinan visión por computadora y componentes electrónicos, fomentando así un aprendizaje activo, creativo e interdisciplinario.

1.2. PLANTEAMIENTO DEL PROBLEMA

Durante mucho en nosotros, existe una necesidad grande de inducir estrategias pedagógicas que integren en la cotidianidad tecnología, esto de forma agigantada y mucho más cuando tenemos un entorno en la que personas discapacitadas necesitan de aquella para mejorar su calidad de vida. Muchas personas enfrentan dificultades al comprender conceptos abstractos en áreas como matemáticas, física y programación. El uso de recursos tradicionales no siempre permite abordar estos temas de manera práctica ni despertar el interés de los demás.

Esta herramienta novedosa como tal es el cubo Rubik, nos ayuda incluso desde un enfoque pedagógico distinto ya que el poder implementar esta metodología de poder resolver algoritmos cuando detectamos un puzle complejo como este ya sea para practicar y mejorar o simplemente lo aplicamos en la vida cotidiana; es por eso, que se implementa este juego de ingenio; es muy importante tener en cuenta que esta herramienta novedosa, es compleja, y sin embargo; automatizar esta tarea conlleva retos complejos relacionados con la adquisición y procesamiento de imágenes bajo condiciones variables de iluminación y ángulo, así como la sincronización mecánica de movimientos precisos. "La robótica educativa ofrece un enfoque innovador para el aprendizaje, permitiendo a los estudiantes interactuar con conceptos abstractos de manera tangible y práctica" [7].

Según el Consejo Nacional para la Igualdad de Discapacidades (CONADIS), en 2018 había 5.917 estudiantes con discapacidad matriculados en universidades y escuelas politécnicas, lo que representa aproximadamente el 1,29% de la población con discapacidad registrada en ese año.

Además, entre 2017 y 2019, 394 personas con discapacidad accedieron a becas otorgadas por la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT), siendo Administración de Empresas y Derecho las carreras más demandadas por este grupo poblacional. Esto sugiere que las carreras tecnológicas, incluidas las relacionadas con la programación, podrían tener una representación aún menor entre los estudiantes con discapacidad.

Según datos del Instituto Nacional de Estadística y Censos (INEC), solo el 48,3% de las personas con discapacidad en Ecuador completaron la educación general básica, y apenas el 23,5% alcanzaron el nivel de bachillerato. En cuanto a la educación superior, únicamente una de cada diez personas con discapacidad ha obtenido un título universitario, y solo el 2,6% ha alcanzado una maestría.

Debido a que, según el Ministerio de Educación, en 2024, 50.676 estudiantes con discapacidad reciben atención en 7.822 instituciones educativas a nivel nacional, representando aproximadamente el 1,21% de la población estudiantil. Sin embargo, la tasa de culminación de estudios superiores entre personas con discapacidad sigue siendo baja; apenas uno de cada diez ha obtenido un título universitario.

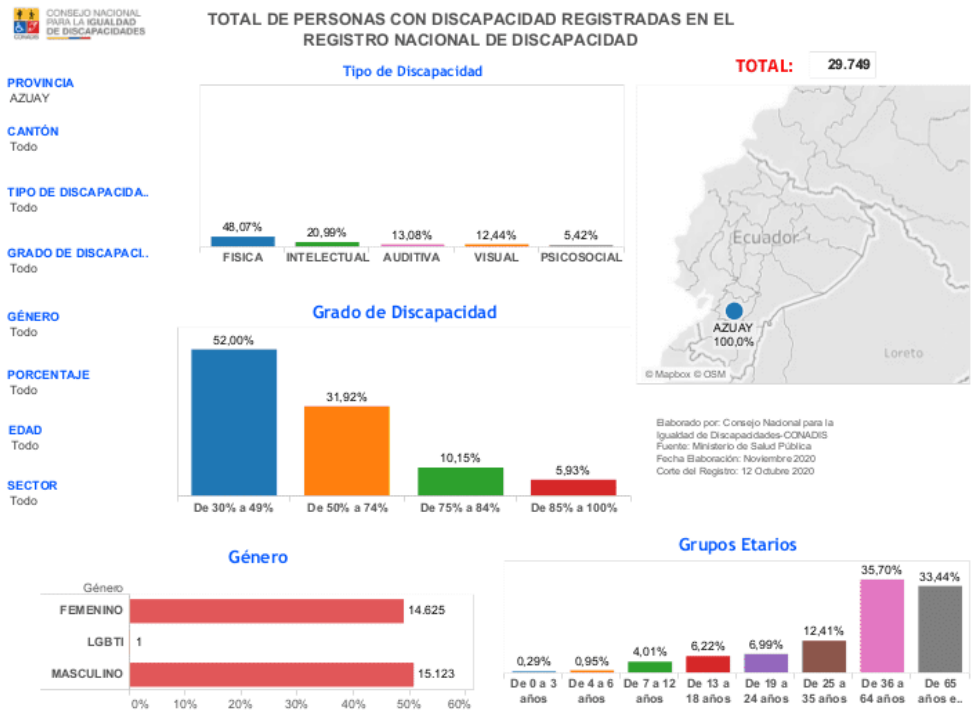


Figura 1: Estadísticas de estudiantes discapacitados, cursando carreras universitarias

A pesar de la limitada participación general, existen esfuerzos para promover la inclusión en áreas tecnológicas. Por ejemplo, en diciembre de 2024, el Tecnológico Universitario Pichincha graduó a la primera cohorte de magísteres tecnológicos que incluyó a personas con discapacidad, como Ángel Sánchez, una persona sorda que completó sus estudios con el apoyo de intérpretes en lengua de señas.

Además, en julio de 2024, el Banco Interamericano de Desarrollo (BID) aprobó un proyecto en Ecuador destinado a fomentar el acceso a la educación superior y la inserción laboral de jóvenes con discapacidad en campos técnicos y tecnológicos, con el apoyo de Fe y Alegría Ecuador

Dado por hecho que de lleno, pensamos en como abarcar las siguientes fases donde proporcionamos la ayuda necesaria para el desarrollo del individuo como tal, de manera; que se logre desarrollar un sistema muy integro y preciso a la hora de mostrarnos como se va desarrollando la resolución de este novedoso souvenir, en este punto; el objetivo a lograr es que prácticamente tengamos un sistema que tanto pueda ser pedagógicamente accesible a todas las personas incluyendo sin importar su condición, debido a que muchas de las veces las personas discapacitadas, requieren a veces de ayuda o herramientas extracurriculares para su desarrollo intelectual en el mundo del aprendizaje cognitivo.

Durante la etapa experimental, se destacó por concepto de que dado que puede ser utilizado como herramienta extracurricular para un estudiante con características de discapacidad, ya que como nos referimos este proyecto tiene como finalidad lograr resolver el cubo Rubik, para poder tener intuitivamente una solución eficiente, y consistente a la hora de querer aprender los algoritmos en el área computacional, física, o matemática, dado que; el juguete en si es una representación de puzle que combina movimientos o secuencias detallando así; un conjunto completo de pasos a los que les podemos llamar como algoritmos, entonces es por esa cuestión que para desarrollar esto se concretó los procesos pertinentes para que sea una herramienta fiable y de calidad para el usuario, durante ese procedimiento nos encontramos que se requería ajuste dinámico de umbrales en valores HSV y validaciones adicionales, y que la integración con servomotores no presente latencias que afectan la fluidez entre captura y giro del cubo, con sus movimientos sincronizados y el funcionamiento de la cámara web.

Esto conduce a la interrogante central: ¿Cómo desarrollar un sistema robusto que combine visión por computadora y control de servomotores para resolver un cubo Rubik con alta precisión, velocidad y fácil de comprender para el usuario?

1.3. JUSTIFICACION

El proyecto en si ayuda no solo a tener conocimientos generales de cómo resolver un cubo Rubik si no también, integra conocimientos de visión por computadora para interpretación de imagen en tiempo real, control de servomotores en tiempo real. Se espera que esta herramienta sirva como base didáctica para estudiantes de Ingeniería en Sistemas, Mecatrónica o áreas afines, fomentando la comprensión de la intersección entre teoría y práctica, y por más importancia, sobre todo cuando tenemos el entorno de discapacidad pedagógicamente hablando; pues debido a que requerimos a veces de ayuda extra cuando queremos resolver un puzle como el cubo Rubik, es conveniente y mucho más optimo tener una herramienta que nos facilite este proceso, detallando así el funcionamiento y aprendizaje concreto de lo que involucra resolver un cubo Rubik.

Además, la tecnología desarrollada puede extenderse a sistemas de clasificación automática por colores en caso de requerir ayuda visual o mejor manejo motriz de las tecnologías. Como indica un estudio reciente, "los proyectos de robótica educativa fomentan el desarrollo de habilidades STEM y promueven el aprendizaje activo y colaborativo" [8].

El desarrollo de un sistema automatizado para la detección y resolución del cubo Rubik representa una oportunidad para acercar a la gente como usuario a disciplinas STEM (Ciencia, Tecnología, Ingeniería y Matemáticas).

A través de este proyecto, se busca demostrar que es posible aprender conceptos complejos de manera lúdica y experimental, utilizando tecnologías disponibles y asequibles para el fin de beneficiar a otros usuarios que requieran de necesidad integra de herramientas para su vida cotidiana, ya que hoy en día el ser humano busca como poder tener a la mano lo más viable para solucionar sus problemas en su diario vivir.

Además, se pretende fomentar el trabajo en equipo, la planificación de proyectos y el aprendizaje, elementos esenciales en la formación integral de las personas que intenten solventar la necesidad de mejorar su entorno con la tecnología, fomentando el desarrollo conceptual de ideas preparadas para el mejor rendimiento de la sociedad actual.

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL

Desarrollar un sistema automatizado basado en visión por computadora y control electrónico que permita detectar y resolver un cubo Rubik, con el fin de proporcionar herramientas pedagógicas al usuario.

1.4.2. OBJETIVOS ESPECIFICOS

- Diseñar e implementar un prototipo funcional del sistema utilizando hardware accesible para que lo pueda replicar el usuario.
- Integrar una solución de software que permita la captura y análisis de imágenes del cubo Rubik para que lleve a su resolución del mismo.
- Aplicar una metodología de desarrollo ágil que facilite el seguimiento y evaluación del proyecto.
- Validar la utilidad del sistema como herramienta educativa en entornos pedagógicos de ser necesario con personas carentes de conocimiento o práctica.

1.5. ALCANCE

El proyecto se enfoca en el cubo Rubik clásico 3x3, utilizando una cámara estática y movimientos de servomotores definidos mediante un Pololu Servo Controller de 18 canales.

No se implementaron redes neuronales ni versiones de cubos más complejas. Se priorizó la claridad y replicabilidad del sistema, con vistas a futuras extensiones que incluyan aprendizaje y facilidad para el usuario.

Este contempla el desarrollo de un prototipo funcional capaz de detectar los colores del cubo Rubik mediante visión por computadora, procesar la información y ejecutar los movimientos necesarios para resolverlo utilizando servomotores.

La solución se orienta hacia el ámbito pedagógico, por lo que se prioriza la simplicidad, la comprensión y la facilidad de implementación.

1.6. CONCEPTOS RELACIONADOS

- **Impresión 3D:** En la impresión 3D enfoca la fabricación de soportes, pinzas, bases, engranajes, y estructuras que ayudan a ajustarse correctamente los componentes electrónicos como servomotores, microcontroladores, cámaras y sensores.

  Si bien podemos utilizar filamentos como PLA, PVA, TPU, entre otras que son aleaciones de componentes que ayudan a la resistencia y calidad del objeto que sea impreso, esta facilita la integración del hardware con el software como por ejemplo los soportes necesarios para la webcam, manteniendo la integridad y las dimensiones exactas del cubo Rubik estándar (3x3x3).

- **Visión por Computadora:** Es el enfoque basado en la inteligencia artificial que ayuda a permitir que las máquinas interpreten el entorno visual, es decir lo analicen y regrese un resultado. Emulando la visión humana. Los sistemas de visión por computadora usan algoritmos de procesamiento de imagen y video para identificar, procesar y analizar patrones, objetos, colores, entre otras funciones.

  Lo cual se emplea en una variedad de aplicaciones, desde cuando queremos detectar objetos hasta la navegación autónoma en si de un entorno. "La visión por computadora se ha convertido en una herramienta esencial en la robótica, permitiendo a los robots percibir y comprender su entorno de manera autónoma" [9].

- **Modelo HSV (Hue, Saturation, Value):** Este modelo de colores es utilizado en visión por computadora muy por lo general, debido a que descompone el color en tres componentes principales: Tono (H), Saturación (S) y Valor (V).

  Ya que ayuda al procesamiento de imágenes, guiando sus valores como una ayuda o guía para tener una mejor percepción humana de los colores optando por la mayor flexibilidad en los análisis de imágenes, inclusive cuando tenemos algunas condiciones de iluminación variables, o algún tipo de hardware que no es constante en su percepción actual del entorno.

- **Metodología en V:** Este es una metodología que se estructura para desarrollar proyectos, especialmente en ingeniería de software, que asocia de manera directa cada fase de desarrollo con una fase correspondiente de pruebas. La metodología en V enfatiza la verificación y validación de los sistemas en cada etapa del proceso, simplemente para mejor calidad cuando desarrollamos dicho producto; es particularmente útil en proyectos que requieren alta fiabilidad por parte del usuario y documentación rigurosa que guíen para que el proyecto sea todo un éxito.

- **Metodología SCRUM:** Es una metodología de tipo framework, más conocida como un marco en este caso debe ser ágil para realizar gestión de proyectos utilizado principalmente en desarrollo de software o codificación, pero se puede usar en otras áreas. SCRUM se basa en ciclos de trabajo iterativos e incrementales, llamados sprints, esto solamente ayuda a mejorar la eficiencia y la flexibilidad con la que el equipo de trabajo llegue a manejarse, por eso esto permite adaptarse a cambios rápidos y responder de manera efectiva con las necesidades del cliente, apoyándonos en fortalecer la colaboración y la mejora continua.

- **Multihilo y Multiproceso en Visión por Computadora:** Esto es por definición un proceso que se usa en la programación como estrategia para mejorar la eficiencia en el procesamiento de grandes volúmenes de datos o en la realización de tareas paralelas.

El procesamiento de imagen en tiempo real se ayuda mucho de esta técnica, ya que permite que se hagan múltiples ejecuciones simultáneamente, haciendo que el tiempo de respuesta del sistema se incremente y sea más optimo.

- **Control de Movimiento y Cinemática Inversa:** Para entender que es el control de movimiento, lo maneja mucho la ciencia de la cinemática es decir se guía a lo mecánico para facilitar las cosas, esto es fundamental en proyectos donde se manipulan servomotores o sistemas robóticos.

  La cinemática inversa se utiliza para determinar ya sea el giro debido que debe hacer los servomotores ayudando a que estos se mantengan en una posición específica, permitiendo que el motor cumpla con el parámetro que se le envía. En el contexto de un cubo Rubik, esta técnica es útil para mover el cubo de forma precisa y controlada según los movimientos de los servomotores.

- **Segmentación de Imágenes:** Consiste en dividir una imagen en regiones o segmentos que tienen características homogéneas, como color o textura; o inclusive cuando queremos separar zonas específicas en esa imagen.

  Es ampliamente utilizada en aplicaciones de reconocimiento de patrones y en la identificación de las caras del cubo Rubik, siendo los stickers de colores que contiene el cubo como prioridad. “Para el procesamiento de imágenes y la detección de colores, se utilizó la biblioteca OpenCV, una herramienta fundamental en visión por computadora" [1]

- **Control de Servomotores con PWM (Modulación por Ancho de Pulso):** Técnica para controlar servomotores mediante la modulación de la duración de los pulsos enviados al motor, que permite controlar con precisión el ángulo de rotación de los servomotores, facilitando movimientos exactos,

  “El control preciso de los servomotores se logró mediante el uso del Pololu Micro Maestro Servo Controller, siguiendo las guías y ejemplos proporcionados por Pololu" [2].

- **Planificación de Tareas en Proyectos de Software:** Gestionar y planificar un software es muy importante para planificar de manera efectiva las tareas y recursos. Sin embargo, eso ayuda a que la planificación conste de la asignación de responsabilidades, la estimación de tiempos y la identificación de los hitos importantes para garantizar el éxito del proyecto en tiempo y forma.

- **Transmisión de Comandos desde Aplicación Android:** Cuando deseo controlar remotamente un sistema robótico mediante una aplicación desarrollada en sistemas Android.

  La aplicación ayuda comunicándose con el sistema a través de interfaces como Bluetooth, Wi-Fi o USB, en este caso es Wi-Fi por usar un servidor para enviar los comandos en tiempo real para ejecutar acciones específicas como mover servomotores, o iniciar algoritmos.

  Esta puede ser unidireccional (de la app al sistema) o bidireccional. “La creación de aplicaciones Android permite interfaces intuitivas y móviles para controlar sistemas de robótica y visión por computadora en tiempo real” [10].

1.7. TRABAJOS RELACIONADOS

Se han desarrollado múltiples proyectos que utilizan el cubo Rubik como una herramienta creativa para enseñar algoritmos y lógica de programación. Gracias a su estructura y complejidad, el cubo se convierte en una excusa perfecta para adentrarse en campos como la robótica, la visión por computadora y la inteligencia artificial. Entre los proyectos más interesantes destacan:

- **OpenAI – Dactyl:** El equipo de OpenAI presentó "Dactyl", una mano robótica capaz de resolver un cubo Rubik. Este logro no solo demostró lo que puede alcanzar el aprendizaje profundo, sino que también expuso algunas de las dificultades al llevar una simulación virtual al mundo real.

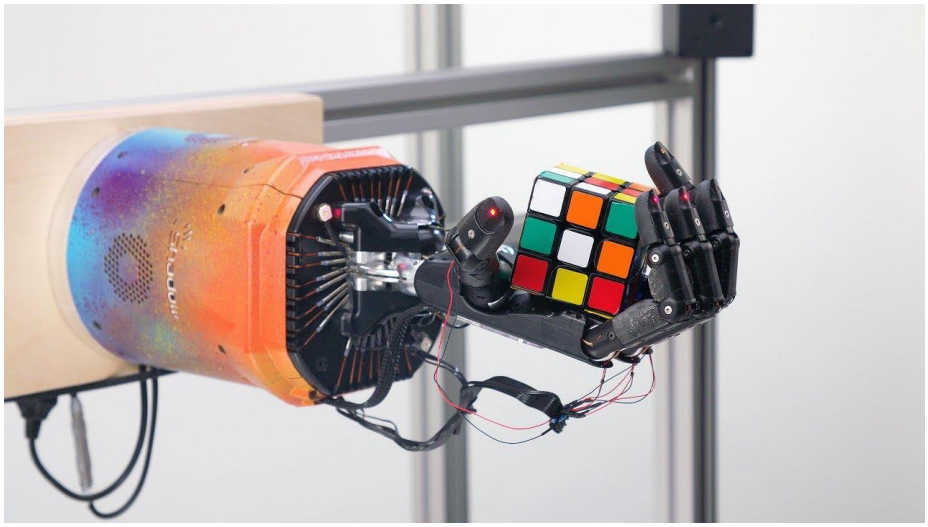


Figura 2: Dactyl, una mano robótica desarrollada por OpenAI

- **CubeStormer III:** Este robot, construido con piezas de LEGO Mindstorms y controlado por un smartphone, rompió récords al resolver el cubo en tan solo 3.25 segundos. Su secreto: una combinación precisa de visión artificial y algoritmos de optimización de movimientos.

Figura 3: CubeStormer III, un robot construido con LEGO Mindstorms

- **RuBot II (The Cubinator):** Creado por Pete Redmond, este robot utiliza cámaras para "ver" el cubo y el famoso algoritmo de dos fases de Kociemba para calcular la solución. Puede completar el reto en menos de 50 segundos, e incluso ha llegado a marcar 21 segundos en demostraciones.

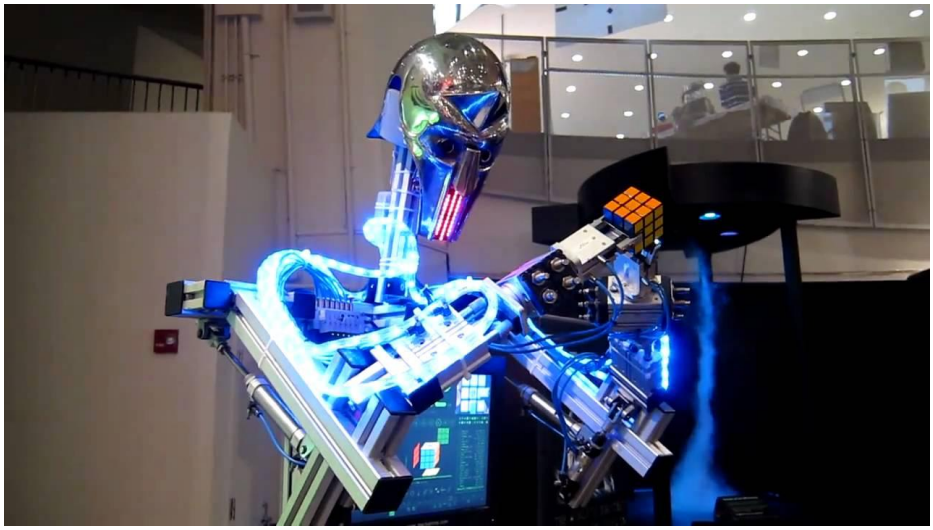


Figura 4: RuBot II, un robot hecho para solucionar cubos Rubik con movimientos mecánicos más precisos.

- **SIGMATEK – Rubik's Cube Solver:** Un grupo de estudiantes de la HTL Saalfelden en Austria, apoyados por SIGMATEK, diseñó un robot que logró resolver el cubo en apenas 385 milisegundos. Su sistema se basa en reconocimiento de imágenes y una cuidadosa planificación de movimientos.

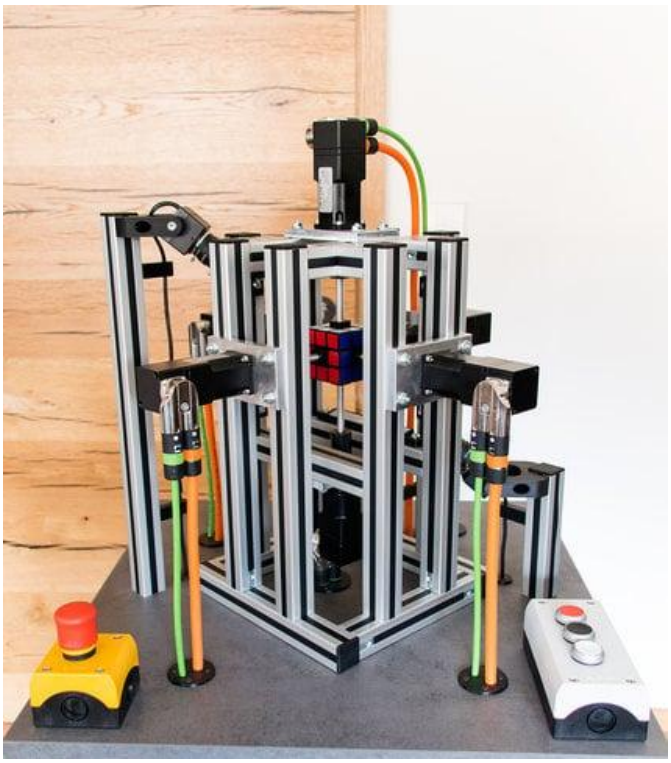


Figura 5: SIGMATEK, sistema para resolver cubos Rubik que crearon estudiantes de la HTL Saalfelden en una comunidad de Austria

- **Q-Bot:** Pensado como un proyecto accesible para quienes quieren construir su propio robot solucionador de cubos Rubik, el Q-Bot combina motores paso a paso y una webcam en una propuesta de código abierto que invita a aprender haciendo.

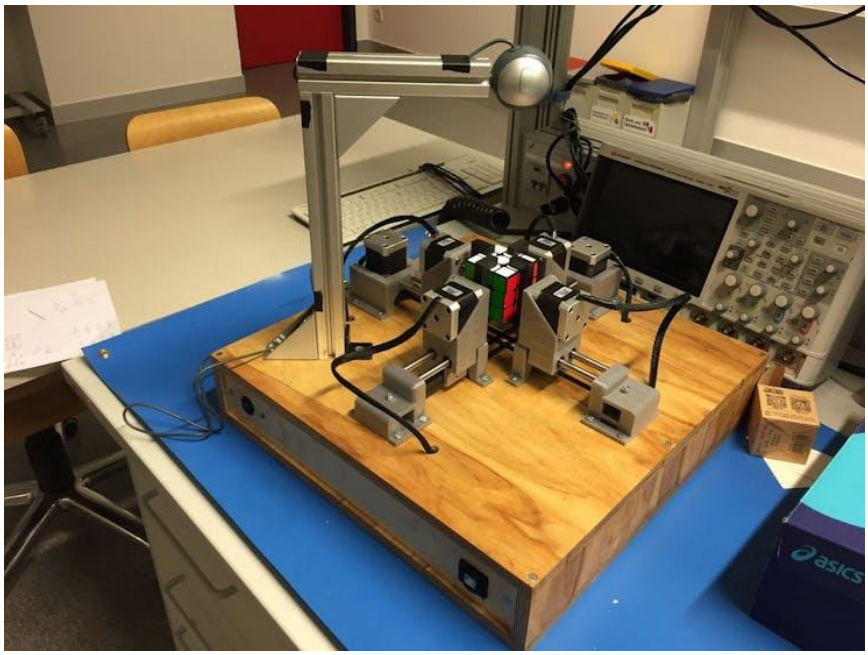


Figura 6: Q-Bot, un prototipo diseñado para resolver cubos Rubik, diseñado por Jocobs.

- **CUBOTino Autonomous:** Este pequeño robot, totalmente impreso en 3D, utiliza una Raspberry Pi Zero 2 y una PiCamera. Su diseño es simple y práctico, logrando resolver el cubo en alrededor de 90 segundos. Es una opción perfecta para quienes buscan un proyecto eficiente y compacto.

Figura 7: CUBOTino, un pequeño robot que resuelve cubos Rubik, desarrollado por Andrea Favero

- **RCR3D – Fully 3D-Printed Rubik's Cube Robot:** Pensado para ser una herramienta educativa, el RCR3D se ha convertido en un éxito entre familias y escuelas. Utiliza servomotores, visión artificial y una Raspberry Pi, todo montado sobre una estructura completamente impresa en 3D.

Figura 8: RCR3D, robot diseñado para resolver cubos Rubik

Estos proyectos destacan la versatilidad del cubo Rubik como herramienta educativa y de investigación en el campo de la robótica y la inteligencia artificial. Cada uno de ellos aporta enfoques innovadores en áreas como la visión por computadora, el control de movimiento y la programación, ofreciendo valiosas lecciones y avances tecnológicos.

CAPITULO II

2. MARCO TEORICO

La manera en que los estudiantes aprenden hoy en día está cambiando gracias a la incorporación de nuevas tecnologías, sobre todo en áreas como la robótica, la programación y la electrónica.

Entre las herramientas más interesantes para estimular el pensamiento lógico, la resolución de problemas y la comprensión de algoritmos se encuentra el cubo Rubik.

Este popular rompecabezas tridimensional, cuando se combina con tecnologías modernas, abre la puerta a proyectos educativos sumamente enriquecedores.

En este trabajo se plantea desarrollar un sistema automatizado que, mediante visión por computadora y componentes electrónicos, sea capaz de escanear y resolver un cubo Rubik con un propósito claramente educativo.

La idea principal es promover un aprendizaje activo, en el que los estudiantes experimenten, exploren y construyan su conocimiento de manera significativa.

Para el diseño y construcción del prototipo, se tomó como referencia el modelo presentado en la plataforma RCR3D, que ofrece planos, listas de materiales y guías útiles para crear un robot resolutor de cubos. Sin embargo, el enfoque de este proyecto va mucho más allá de lo técnico: se busca aprovechar su potencial como herramienta de aprendizaje.

Al integrar tecnologías como OpenCV para reconocer los colores, servomotores para realizar los movimientos mecánicos, y aplicar principios de desarrollo ágil, se logra una experiencia educativa completa. Esto permite a los estudiantes acercarse de manera práctica a temas como la visión por computadora, la programación estructurada y la lógica de resolución de problemas. De esta manera, el proyecto no solo enseña tecnología, sino que también impulsa la creatividad, el trabajo en equipo y la autonomía en el aprendizaje.

2.1. MARCO METODOLOGICO

El proyecto fue desarrollado bajo un enfoque ágil, aplicando la metodología Scrum para organizar el trabajo de forma flexible y adaptativa. La naturaleza del proyecto, que mezcla hardware, software y visión computacional, requería una estructura que permitiera iterar, corregir y mejorar de manera continua.

Scrum permitió dividir el proyecto en ciclos de trabajo llamados sprints, donde cada uno tenía objetivos claros, como el diseño del prototipo, la programación de la visión por computadora y la integración de los servomotores.

Este método favoreció la entrega de avances funcionales de manera regular, permitiendo ajustes sobre la marcha y una documentación organizada del progreso.

En este proyecto, cada sprint estuvo enfocado en cumplir metas específicas, como el diseño del prototipo, la integración de los componentes electrónicos, la programación de la visión por computadora y la validación del sistema educativo.

Esto permitió adaptar los avances del proyecto en función de los resultados obtenidos, mantener una documentación detallada del progreso y corregir errores a tiempo.

Además, se aplicó una estrategia de investigación-desarrollo-validación, en la cual primero se recopilaron antecedentes técnicos (por ejemplo, el diseño físico propuesto por RCR3D), luego se adaptaron y desarrollaron nuevas funcionalidades, y finalmente se evaluó el impacto educativo del sistema mediante pruebas funcionales.

El enfoque metodológico combina conocimientos de ingeniería, informática y pedagogía, para crear un recurso que, además de ser técnicamente efectivo, sea útil en procesos de enseñanza-aprendizaje centrados en el estudiante.

2.2. SCRUM

Scrum es una metodología ágil que permite gestionar proyectos complejos a través de una estructura ligera, flexible y adaptativa.

Se basa en la organización del trabajo en ciclos cortos e iterativos denominados sprints, los cuales permiten entregar resultados funcionales en periodos regulares de tiempo.

Este enfoque promueve la colaboración del equipo, la transparencia en los procesos, la mejora continua y la rápida adaptación a los cambios o retroalimentaciones.

2.2.1. CICLO DE VIDA DE SCRUM

El ciclo de vida de Scrum es altamente adecuado para la gestión de proyectos ágiles y adaptativos, y se puede combinar con el modelo en V para aportar una estructura más formal a las fases de desarrollo.

En este contexto, trabajamos como único desarrollador, pero las prácticas de Scrum siguen siendo relevantes y útiles para mantener un flujo continuo de trabajo iterativo, organizado y flexible.

2.2.1.1. Product Backlog

El Product Backlog es una lista priorizada de tareas, características, mejoras, y correcciones necesarias para desarrollar un producto.

El Product Owner (propietario del producto) es responsable de mantener y priorizar el Product Backlog, asegurando que las tareas más importantes estén al frente de la lista para que el equipo de desarrollo pueda enfocarse en ellas.

Historia de Usuario	Descripción	Nro.
Como desarrollador, quiero controlar los servomotores, para poder mover las piezas del robot automáticamente.	Instalación y prueba de servomotores.	1
Como desarrollador, quiero capturar y procesar la cámara web con OpenCV, para detectar correctamente los colores de las caras del cubo.	Calibración, HSV, exclusión de interferencias.	2
Como desarrollador, quiero ver una vista previa con los 54 colores detectados, para verificar que todo se detectó correctamente.	Representación visual del cubo.	3
Como usuario, quiero tener una interfaz gráfica intuitiva, para iniciar el proceso y ver el estado del cubo Rubik.	GUI con botones y vistas.	4
Como usuario, quiero que el sistema resuelva el cubo Rubik, para que me muestre los pasos exactos de cómo se está solucionando también mediante servomotores.	Integrar algoritmo Kociemba, traducir solución.	5
Como usuario, quiero usar una app Android para controlar el sistema, para operar el cubo remotamente sin usar el PC.	Comunicación entre app y sistema base.	6
Como desarrollador, quiero revisar y consolidar el código del sistema, para tener un sistema listo para escalar o presentar.	Refactorización y limpieza.	7

Tabla 1: Tabla de Product Backlog definido, para realizar el método SCRUM.

2.2.1.2. Sprint Planning (Planificación del Sprint)

Duración: 4 Semanas (1 Semana por Sprint)

2.2.1.3. Sprint (1, 2, 3, 4)

- ✓ Sprint 1: Hardware & Captura
  - Historias:
    - Historia 1: Control de servomotores
    - Historia 2: Captura y procesado de cámara

- Tareas clave:
    - Configurar librería serial y calibrar servos.
    - Inicializar OpenCV y parámetros HSV, filtrar interferencias.

✓ Sprint 2: Visualización & GUI básica

- Historias:
    - Historia 3: Vista previa de colores detectados
    - Historia 4: Interfaz gráfica intuitiva
- Tareas clave:
    - Generar componente visual 3×3×6 con colores.
    - Construir GUI con botones de control y vista previa.

✓ Sprint 3: Resolución Automática

- Historias:
    - Historia 5: Resolución automática del cubo
- Tareas clave:
    - Integrar librería Kociemba, mapear notación a funciones de giro.
    - Añadir modo de "previa animada" y ejecución física de la solución.

✓ Sprint 4: App Móvil & Refactor

- Historias:
    - Historia 6: App Android de control remoto
    - Historia 7: Revisión y consolidación del código

- Tareas clave:

    - Desarrollar app Android, definir protocolo de mensajes JSON.

    - Refactorizar código y documentar.

2.2.1.4. Daily Scrum (Reunión Diaria)

Como protagonista de este proyecto, se puede seguir esta práctica para mantener la claridad y organización.

Al final de cada día, se hace una breve revisión personal, respondiendo a las siguientes preguntas:

- ✓ ¿Qué hice ayer?
    - ¿Logré completar las tareas planificadas?
    - ¿Hubo obstáculos?
- ✓ ¿Qué voy a hacer hoy?
    - ¿Qué tareas me gustaría abordar hoy?
    - ¿Necesito modificar el plan?
- ✓ ¿Hay algún impedimento?
    - ¿Algo que me esté retrasando?
    - ¿Necesito más recursos o tiempo?

2.2.1.5. Sprint Review (Revisión del Sprint)

- ✓ Revisar si los objetivos del Sprint se han cumplido.
- ✓ Evaluar el avance en el cubo Rubik, verificando si la plantilla o vista previa final refleja correctamente los colores del cubo.
- ✓ Recopilar retroalimentación para saber si el producto está alineado con las expectativas.

2.2.1.6. Sprint Retrospective (Retrospectiva del Sprint)

Esta es una fase interna donde se reflexiona sobre el Sprint. Al ser un solo desarrollador, te haces preguntas como:

- ✓ ¿Qué salió bien durante el Sprint? ¿Qué tareas se completaron sin problemas?
- ✓ ¿Qué podría haberse hecho mejor? ¿Hubo dificultades que debieron haberse abordado antes?
- ✓ ¿Qué cambios podrías implementar en el siguiente Sprint para mejorar tu productividad o la calidad del código?

La retrospectiva nos permite mejorar constantemente, asegurándose de que el siguiente Sprint sea más eficiente y fluido.

2.2.1.7. Repetición del ciclo

Con el ciclo terminado, se repite el proceso comenzando con una nueva planificación para el siguiente Sprint. A medida que se avanza, el ciclo se repite hasta que el proyecto esté completo o los objetivos definidos se alcancen y el producto final se ajuste hasta cumplir con todos los criterios de calidad.

2.2.2. SCRUM TEAM

El marco de trabajo Scrum establece una estructura organizacional compuesta por roles definidos que trabajan en conjunto para lograr los objetivos de un proyecto. Tradicionalmente, el Scrum Team se conforma por el Product Owner, el Scrum Master y el Development Team, cada uno con funciones específicas dentro del ciclo ágil.

Sin embargo, en el contexto de este proyecto individual de tesis, se adoptaron los principios de Scrum desde una perspectiva adaptativa, combinándolos con la rigurosidad secuencial del modelo en V para lograr una gestión equilibrada entre planificación estructurada y flexibilidad ágil.

En este proyecto me tocó asumir, de manera simultánea, todos los roles que normalmente conforman un equipo Scrum. Como Product Owner, fui el encargado de definir y priorizar los requerimientos del sistema, siempre alineándolos con los objetivos académicos y pedagógicos planteados desde el inicio.

En el rol de Scrum Master, me ocupé de organizar el flujo de trabajo, resolver obstáculos técnicos y asegurar que el proyecto se mantuviera fiel a los principios ágiles.

Y como parte del Development Team, diseñé e implementé todas las soluciones de hardware y software necesarias para lograr la automatización del cubo Rubik, utilizando visión por computadora y servomotores.

Este enfoque de trabajo permitió que las actividades se estructuraran en ciclos de desarrollo cortos e iterativos, favoreciendo la evaluación constante del progreso y permitiendo realizar mejoras de manera continua.

La combinación de metodologías, tanto en V como Scrum aportó una base sólida al proyecto, equilibrando el rigor técnico con la flexibilidad necesaria para aprender, adaptarse y evolucionar a lo largo del desarrollo.

2.2.3. EVENTOS DE SCRUM

El scrum está estructurado en torno a una serie de eventos que garantizan el orden y la transparencia durante el desarrollo de un proyecto. Estos eventos permiten una planificación efectiva, un seguimiento continuo del progreso y la mejora constante del proceso de trabajo.

En este proyecto, aunque se trató de una tesis desarrollada individualmente, los eventos de Scrum fueron utilizados de forma adaptada, permitiendo una organización disciplinada y ágil del tiempo y los recursos disponibles.

- ✓ **Sprint:** Los sprints cumplieron el tiempo de desarrolló en 4 semanas, en esta tesis; se definieron jornadas de trabajo con objetivos precisos para alcanzar finalizar el proyecto, se tomó en cuenta el diseño del prototipo, la programación, la integración del sistema con los servomotores, y el diseño y programación de la aplicación Android que funciona remotamente con el proyecto.

- ✓ **Sprint Planning:** Al inicio de cada ciclo, se realizó una planificación, estableciendo las tareas y los objetivos a lograr.

  Esta actividad permitió a enfocarse en metas concretas y manejables, enfocándose en la complejidad y el impacto.

- ✓ **Daily Scrum:** Tradicionalmente se realiza en equipo, pero en este caso se lo hizo individual y se implementó como una autoevaluación diaria.

  En donde hubo un espacio al inicio de cada jornada para hacer una revisión de lo que se había hecho, y reorganizar las actividades en caso de que exista problemas.

- ✓ **Sprint Review:** En cada Sprint, se hizo una revisión de lo que se hizo en el sprint. Para esto se utilizó pruebas funcionales del sistema y verificación de los avances comparando con los objetivos planeados.

  De la misma forma, se dieron observaciones y ajustes requeridos.

- ✓ **Sprint Retrospective:** Finalmente, hubo un espacio de evaluación critica en cada cierre. Esto nos ayudó a identificar errores y oportunidades de mejorar en la gestión del proyecto.

  Es un momento clave para fortalecer la autogestión SCRUM.

Esto quiere decir que, los eventos de Scrum fueron usados para mantener un enfoque iterativo, ordenado y flexible. Siempre adaptado a los requerimientos de un proyecto académico desarrollado, pero con visión profesional y metodológica.

2.2.4. ARTEFACTOS DE SCRUM

Aquí presentamos una descripción detallada de los cuatro artefactos de Scrum: Product Backlog, Sprint Backlog, Incrementos y Definition of Done; estos se aplican de la siguiente manera en nuestro proyecto del cubo Rubik.

1.Product Backlog

- ✓ **Historia 1:** Control de servomotores

"Como desarrollador, quiero controlar los servomotores, para poder mover las piezas del robot automáticamente."

Descripción: Instalación, calibración y prueba de servos en rangos 0–90°.

✓ **Historia 2:** Captura y procesado de cámara

“Como desarrollador, quiero capturar y procesar la cámara web con OpenCV, para detectar correctamente los colores de las caras del cubo.”

Descripción: Calibración de la cámara, definición de rangos HSV, exclusión de interferencias.

✓ **Historia 3:** Vista previa de colores detectados

“Como desarrollador, quiero ver una vista previa con los 54 colores detectados, para verificar que todo se detectó correctamente.”

Descripción: Representación visual de las seis caras en una matriz 3×3×6.

✓ **Historia 4:** Interfaz gráfica intuitiva

“Como usuario, quiero tener una interfaz gráfica intuitiva, para iniciar el proceso y ver el estado del cubo Rubik.”

Descripción: GUI con botones “Iniciar”, “Pausar”, “Reiniciar”, panel de cámara y vista previa.

✓ **Historia 5:** Resolución automática del cubo

“Como usuario, quiero que el sistema resuelva el cubo Rubik, para que me muestre los pasos exactos de cómo se está solucionando también mediante servomotores.”

Descripción: Integración de algoritmo Kociemba y traducción a movimientos físicos.

✓ **Historia 6:** App Android de control remoto

“Como usuario, quiero usar una app Android para controlar el sistema, para operar el cubo remotamente sin usar el PC.”

Descripción: Comunicación bidireccional entre app y sistema base (Servidor Flask).

✓ **Historia 7:** Revisión y consolidación del código

“Como desarrollador, quiero revisar y consolidar el código del sistema, para tener un sistema listo para escalar o presentar.”

Descripción: Refactorización, limpieza y documentación.

2.Sprint Backlog

Como el Sprint Planning tiene una duración de 4 semanas, se realizarán 4 Sprints haciendo uno por semana.

Aquí, las fases del Modelo en V se combinan con el enfoque de Scrum:

- ✓ Sprint 1 – Hardware & Captura

  El objetivo de este primer Sprint es garantizar que el sistema físico responda correctamente a comandos de posición y que el flujo de vídeo sea lo suficientemente estable para soportar la detección de color.

  Historias de Usuario implicadas:

  - Historia de Usuario 1: Control de servomotores

  "Como desarrollador, quiero controlar los servomotores, para poder mover las piezas del robot automáticamente."

  Control básico de servomotores (0–90°): Se diseña e implementa la lógica de transmisión de ángulos a los servomotores, asegurando que cada instrucción (0°, 45°, 90°) resulte en un movimiento preciso y repetible.

  Criterios de aceptación:

  - El controlador acepta y envía comandos de ángulo (0°, 45°, 90°) sin errores de comunicación.
  - Cada servo alcanza la posición solicitada con una tolerancia máxima de ±2°.
  - El tiempo de respuesta desde el comando hasta el inicio del movimiento es ≤ 150 ms.
  - Se pueden ejecutar de forma consecutiva al menos 10 movimientos sin pérdida de sincronía.
  - Existe un procedimiento de calibración automatizado que guarda y recupera offset de posición.

- Historia de Usuario 2: Captura y procesado de cámara

"Como desarrollador, quiero capturar y procesar la cámara web con OpenCV, para detectar correctamente los colores de las caras del cubo."

Captura y calibración HSV en OpenCV, exclusión de interferencias: Se configura la lectura de la cámara web con OpenCV, se establecen rangos iniciales de HSV para cada color del Cubo Rubik y se añade un mecanismo de filtrado que excluya objetos o tonos ajenos (por ejemplo, un objeto rojo que pueda interferir con la detección).

Criterios de aceptación:

- El feed de cámara se mantiene a ≥ 15 FPS durante 5 minutos continuos sin caídas de frame.
- Cada uno de los seis colores del cubo (blanco, amarillo, rojo, naranja, verde, azul) se detecta con ≥ 95 % de acierto en zonas de prueba.
- El rango HSV de cada color es configurable en tiempo real y persiste entre ejecuciones.
- Se excluyen correctamente interferencias (otros objetos de color) en al menos el 99 % de los casos de prueba.
- Los fotogramas corruptos o con enfoque insuficiente se identifican y se descartan automáticamente.

Durante la fase de Diseño y Desarrollo (Modelo en V), se codifica el controlador de servos, se calibra el mapeo ángulo–PWM, se crea la rutina de captura de vídeo y se parametriza el espacio HSV.

Se documenta la interfaz de configuración de rangos de color en un archivo de parámetros.

En la fase de Validación (Modelo en V), se realizan pruebas de precisión de servos midiendo el ángulo real recorrido frente al solicitado en al menos diez ciclos por posición.

Se evalúa la estabilidad del feed de la cámara (sin pérdidas de frames durante cinco minutos continuos) y la exactitud de detección de un patrón de color estándar. Se registran incidencias y se ajustan umbrales de HSV para reducir falsos positivos.

✓ Sprint 2 – Visualización & GUI básica

Este Sprint tiene como objetivo ofrecer al usuario una vista previa que combine el feed de la cámara y la plantilla de los 54 stickers detectados, además de proporcionar controles mínimos para arrancar y detener el proceso.

Historias de Usuario implicadas

- Historia de Usuario 3: Vista previa de colores detectados

"Como desarrollador, quiero ver una vista previa con los 54 colores detectados, para verificar que todo se detectó correctamente."

Plantilla visual 3×3×6 con los colores detectados: Se desarrolla una componente gráfica que represente, en dos dimensiones, cada una de las seis caras del cubo como una matriz 3×3, pintando cada "sticker" con el color detectado en tiempo real.

Criterios de aceptación:

- La GUI muestra una plantilla 3×3 por cara (6 caras) pintada con los colores detectados, sin superposiciones.
- La actualización de la plantilla tras cada captura sucede en ≤ 200 ms.
- Los stickers cuya detección sea dudosa (< 80 % de píxeles coincidentes) quedan resaltados en rojo para reintento.
- Al hacer clic sobre un sticker, se despliega un tooltip con el valor HSV promedio y coordenadas de la región.

- Se valida visualmente que, en un test de 10 escenarios distintos, la plantilla concuerde al 100 % con la configuración esperada.

- Historia de Usuario 4: Interfaz gráfica intuitiva

"Como usuario, quiero tener una interfaz gráfica intuitiva, para iniciar el proceso y ver el estado del cubo Rubik."

GUI con botones "Iniciar", "Pausar", "Reiniciar" y panel de cámara + plantilla: Se implementa una ventana de interfaz con botones claramente etiquetados y dispuestos, un área que muestre el vídeo capturado y, junto a ella, la representación de la plantilla de colores.

Criterios de aceptación:

- Existen claramente los botones "Iniciar", "Pausar", "Reiniciar" y "Resolver", operativos en todo momento.

- El feed de cámara y la plantilla de colores conviven en la misma ventana, con un layout responsivo.

- Cada acción de botón genera una notificación visual o sonora que confirme su ejecución.

- El estado del sistema ("Conectado", "Procesando", "Listo", "Error") se muestra en un área dedicada.

- Todos los elementos de la GUI pasan un test de usabilidad con al menos 3 usuarios no desarrolladores, obteniendo una puntuación $\geq 4/5$ en claridad.

En la fase de Diseño y Desarrollo, se elabora la arquitectura de la GUI (por ejemplo, usando Qt o Tkinter), se integran las librerías gráficas para dibujar la plantilla, y se codifica la lógica de sincronización entre captura de frames y actualización de la vista de stickers.

Durante la Validación, se comprueba que cada botón invoque correctamente la funcionalidad esperada (arrancar, pausar, reiniciar) sin bloquear la interfaz ni perder datos.

Se realiza un test de usabilidad básico con un usuario externo para asegurar que la plantilla y el feed sean claros y que la actualización ocurra con un retardo inferior a 200 ms.

✓ Sprint 3 – Resolución Automática

El foco de este Sprint es integrar el algoritmo de Kociemba para el cálculo de la solución óptima y traducir esa solución en instrucciones para los servomotores.

Historia de Usuario implicada

- Historia de Usuario 5: Resolución automática del cubo

"Como usuario, quiero que el sistema resuelva el cubo Rubik, para que me muestre los pasos exactos de cómo se está solucionando también mediante servomotores."

Cálculo de la secuencia de movimientos óptima y ejecución animada/manual: Se incorpora la librería Kociemba, se obtiene la lista de movimientos (hasta 20 en el peor caso), y se implementa un motor que traduzca cada movimiento ("F", "L", "B", "R", etc.) en un par de ángulos de servomotor para realizar el giro físico.

Criterios de aceptación:

- Dado un estado válido del cubo, el algoritmo Kociemba produce una secuencia ≤ 20 movimientos.
- Cada movimiento ("F", "L", "B", "U", etc.) se traduce en uno o más comandos de servo que ejecutan físicamente el giro.
- En modo "Previa", la GUI muestra la lista de pasos con su notación y un diagrama animado de cada giro.
- En modo "Ejecución", la acción física se dispara paso a paso, con confirmación visual de cada movimiento completado.

- Tras aplicar la secuencia completa, el cubo queda resuelto en hardware, verificado por una captura final y recálculo de colores.

En Diseño y Desarrollo, se encapsula la llamada al resolver dentro de una clase de "Gestor de Solución", se mapea cada notación de cubo a funciones existentes (letter_u(), letter_r(), etc.) respetando sus movimientos predefinidos, y se añade un modo de "previa animada" donde la GUI muestra cada paso antes de enviarlo al hardware.

En la Validación, se generan diez estados iniciales aleatorios válidos del cubo (usando un secuenciador de scramble) y se verifica que, tras aplicar la serie de comandos, el cubo quede resuelto. Se comprueba, asimismo, en modo animado, que la interfaz muestre correctamente los pasos y que la sincronía entre visualización y movimiento físico no supere 100 ms de desfase.

✓ Sprint 4 – App Móvil & Refactor

El último Sprint busca habilitar el control remoto desde un dispositivo Android y dejar el código listo para escalar o presentar, sin modificar las funciones manuales de movimiento.
Historias de Usuario implicadas

- Historia de Usuario 6: App Android de control remoto

"Como usuario, quiero usar una app Android para controlar el sistema, para operar el cubo remotamente sin usar el PC."

App Android para enviar comandos ("Iniciar", "Resolver", etc.) y recibir feedback: Se desarrolla una aplicación con botones equivalentes a los de la GUI de PC y un área de texto o indicadores que muestren el paso actual y el estado del cubo. Se implementa comunicación Wi-Fi o Bluetooth según disponibilidad de hardware.

Criterios de aceptación:

- La app detecta y se empareja con el dispositivo base (por Wi-Fi o Bluetooth) en ≤ 10 s.
- Desde la app se pueden enviar los comandos "Iniciar", "Pausar", "Reiniciar" y "Resolver" y recibir confirmación en < 150 ms.

- El estado actual del cubo (cara procesada, paso en ejecución, errores) se muestra en pantalla de la app.
- Se manejan desconexiones inesperadas mostrando un mensaje claro y ofreciendo reintento de conexión.
- La app opera correctamente en al menos tres dispositivos Android de tamaños de pantalla distintos (5″ a 7″).

- Historia de Usuario 7: Revisión y consolidación del código

"Como desarrollador, quiero revisar y consolidar el código del sistema, para tener un sistema listo para escalar o presentar"

Refactorizar todo el sistema (módulos, pruebas, documentación) sin alterar las funciones específicas de movimientos: Se reorganiza el código en paquetes para captura, procesamiento, GUI, resolución y comunicaciones; se añaden pruebas unitarias para cada módulo y se redacta la documentación técnica y de usuario.

Criterios de aceptación:

- El código está dividido en módulos claros (captura, procesamiento, GUI, resolución, comunicaciones)
- Se alcanzan ≥ 80 % de cobertura de pruebas unitarias e integración en el repositorio principal.
- Todos los métodos públicos y principales clases incluyen docstrings o comentarios de al menos 3 líneas.
- La documentación de despliegue (README) explica paso a paso cómo instalar y ejecutar el sistema en un entorno limpio.

En la parte de Diseño y Desarrollo, se define un protocolo ligero de mensajes (por ejemplo, JSON sobre socket) para la app Android, se genera la APK y se ajusta el receptor en el sistema base.

A continuación, se aplica una refactorización general siguiendo principios SOLID y se integran pruebas unitarias automatizadas.

La fase de Validación incluye tests de compatibilidad de la app en al menos tres modelos de teléfono diferentes, medición de latencia de comandos remotos (< 150 ms) y un ensayo final de flujo completo: desde la captura inicial hasta la ejecución remota de la solución.

Finalmente, se revisa y corrige la documentación, y se preparan los anexos para la defensa de la tesis.

Con esta descripción textual y detallada, cada Sprint queda plenamente justificado tanto en su parte ágil (Scrum) como en las fases clásicas de diseño y validación del Modelo en V, enlazando explícitamente cada actividad con las Historias de Usuario del Product Backlog.

3. Incrementos

Al finalizar cada Sprint obtenemos un incremento potencialmente entregable:

- ✓ **Sprint 1:** Servomotores controlables en software y flujo de vídeo calibrado listo para procesar colores.
- ✓ **Sprint 2:** GUI en PC que muestra en tiempo real los 54 stickers detectados y permite iniciar/pausar/reiniciar.
- ✓ **Sprint 3:** Generación de la solución Kociemba y ejecución paso a paso, tanto en pantalla como en servos.
- ✓ **Sprint 4:** Control remoto estable desde Android, código refactorizado y documentación completa para demostración.

Cada incremento se integra, prueba y revisa antes de la demo de fin de Sprint.

4. Definition of Done (DoD)

Para garantizar calidad, consistencia y preparación para escalado, cada Historia de Usuario sólo se considera terminada cuando cumple TODOS los criterios siguientes:

- ✓ **Calidad de Código**
    - Estándares y estilo: Código formateado según la guía PEP8 (Python) o convenciones establecidas, que los nombres de variables y funciones sean semánticos y consistentes. [5]
    - Revisión de código: Todo cambio ha pasado por un pull request revisado por al menos un compañero, y se han cerrado todos los comentarios de revisión.
- ✓ **Pruebas Automatizadas**
    - Cobertura mínima: $\geq 80\,\%$ de cobertura en el módulo afectado por la historia de usuario.
    - Tipos de pruebas
        - ✓ Unitarias: cubren lógica aislada (p.ej. mapeo rotación, conversiones HSV).
        - ✓ De integración: validan interacciones cámara, procesador de color, GUI, control de servos.
        - ✓ De extremo a extremo: flujos críticos (captura, detección, solución, movimiento).
- ✓ **Documentación**
    - Comentarios inline: Todas las funciones públicas cuentan con docstrings descriptivos.
    - Guías de usuario y desarrollador: Manual de uso para PC y Android, con capturas de pantalla.
    - Guía de despliegue y configuración del entorno.
- ✓ **Rendimiento y Estabilidad**
    - Captura de vídeo: Sin pérdida de frames durante al menos 5 minutos de uso continuo, y latencia de procesamiento ≤ 100 ms por frame.

- Movimientos de servos: Tiempo de respuesta servo ≤ 150 ms desde comando a movimiento, y precisión de ángulo $\leq \pm 2°$ en 10 ciclos de prueba.

✓ **Experiencia de Usuario (UX)**

- Interfaz PC: Botones operativos y claramente etiquetados (“Iniciar”, “Abrir”, “Cerrar”, “Resolver”, “STOP”). Y Retroalimentación visual de estados (procesando, listo, error).

- App Android: Diseño responsive que se adapte a pantallas de 5″ a 7″, con indicadores de conexión y notificaciones de error.

✓ **Integración Continua y Despliegue**

- Entrega de artefactos: Paquete instalable para PC (wheel/exe) y APK para Android disponibles en el repositorio.

✓ **Seguridad y Robustez**

- Manejo de errores: Captura y registro de excepciones críticas (p.ej. cámara no detectada, servo desconectado), y mensajes de error claros al usuario.

- Validación de datos: Comprobación de formatos de imagen, rangos HSV y comandos de ángulo antes de procesar.

Con este nivel de detalle en la Definition of Done, cada Historia de Usuario quedará asegurada en calidad, rendimiento, usabilidad, seguridad y mantenibilidad, cumpliendo los estándares académicos y profesionales.

2.3. PROYECTOS DE VINCULACION

Con el tiempo, este proyecto se transformó en una verdadera herramienta de conexión con la comunidad, la educación y el sector tecnológico. Basado en el uso de visión por computadora, servomotores y principios de trabajo ágil, el proyecto encontró aplicaciones más amplias de las inicialmente previstas.

Uno de sus principales logros fue servir como inspiración para talleres prácticos de robótica y visión computacional en colegios e institutos de Guayaquil. Allí, los estudiantes no solo aprendieron conceptos teóricos, sino que tuvieron la oportunidad de construir su propio robot usando piezas impresas en 3D [8], viviendo de primera mano la experiencia de ensamblar y programar tecnología real.

El alcance del proyecto también llegó a escenarios más grandes: fue presentado en la Feria Tecnológica SOLACYT 2024 en Quito. Durante el evento, se mostró cómo una propuesta de automatización puede hacer que aprender temas complejos sea más sencillo y divertido, atrayendo el interés de estudiantes, docentes y profesionales.

Además, se generaron colaboraciones con la carrera de Ingeniería Electrónica, enfocadas en perfeccionar aspectos técnicos como la sincronización entre la captura de imágenes y el movimiento mecánico del cubo. Este trabajo conjunto ayudó a mejorar la precisión del sistema y optimizar su funcionamiento [2].

Un aspecto especialmente valioso de este proyecto fue su enfoque inclusivo. Desde el principio, se pensó en un sistema que no solo estuviera dirigido a entusiastas de la tecnología, sino también a personas que enfrentan retos físicos o cognitivos. Considerando que, según datos de CONADIS, solo el 1,29% de los estudiantes universitarios ecuatorianos tiene una discapacidad [7], contar con herramientas educativas accesibles es crucial para fomentar una educación más equitativa.

Más allá de la educación, el prototipo tiene el potencial de ser aplicado en contextos donde se necesita asistencia visual o motriz, abriendo nuevas oportunidades de inclusión social. Como se resalta en la tesis, "la visión por computadora y la robótica educativa no solo impulsan el aprendizaje, sino que también ayudan a derribar barreras físicas y cognitivas" [9].

En resumen, este proyecto no solo fortaleció el perfil profesional de su creador, sino que también dejó una huella significativa en la comunidad educativa y tecnológica. Su carácter escalable y adaptable confirma su potencial para seguir creciendo y generando impacto en distintos espacios del país.

2.4. EDITOR DE CODIGO

El editor de código es la herramienta principal para el desarrollo de software, ya que facilita la escritura, edición, depuración y organización del código fuente. Lo que caracteriza este proyecto es:

Depuración integrada: Puntos de seguimiento de variables en donde verificamos los módulos de captura y detección de colores.

Extensiones y plugins: Instalación de extensiones específicas para Python, control de versiones (Git), y formateo (Black).

Terminal integrada: Ejecución de scripts de prueba y despliegue del servidor backend sin salir del editor.

2.4.1. VISUAL STUDIO CODE

Visual Studio Code (VS Code) es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Sus ventajas en este proyecto ayudan a desarrollar en si lo siguiente:

Ligereza y velocidad: Se inicia rápidamente y maneja proyectos con muchos archivos sin retardos, aunque en esta ocasión solamente utilizamos un archivo único.

Marketplace de extensiones: Disponibilidad de extensiones como Python, Pylance, etc. para trabajar ágilmente.

2.5. BACKEND

El desarrollo del Backend del sistema de detección, control y resolución automática de un cubo Rubik, el cual interactúa con una interfaz gráfica desarrollada en Tkinter (Python) y una aplicación Android escrita en Java.

El backend fue implementado utilizando el microframework Flask, el cual permite crear un servidor ligero y eficiente capaz de recibir comandos desde distintos clientes y actuar sobre el hardware, incluyendo cámaras y servomotores.

Cada comando enviado desde la GUI o la app móvil es interpretado por el servidor Flask, que a su vez ejecuta las acciones correspondientes, como hacer el inicio de procesar colores, mover los motores, o iniciar la resolución del cubo.

Flask fue seleccionado por las siguientes razones:

- Su simplicidad y ligereza, ideal para proyectos embebidos o con requerimientos de respuesta rápida.
- Su compatibilidad con múltiples clientes HTTP, incluyendo interfaces desarrolladas en Python, Android o navegadores.
- Su facilidad de integración con scripts de control de hardware.
- Permite una estructura modular y escalable del sistema backend.

El servidor Flask actúa como puente de comunicación entre:

- La interfaz gráfica en Tkinter.
- La app Android.
- El sistema de visión por computadora (detección de colores del cubo).
- El sistema de control de hardware (servomotores para rotar el cubo).

Código BACKEND:

```
flask_app = Flask(__name__)
@flask_app.route("/abrir")
def abrir_flask():
   command_queue.put('abrir')
   return jsonify({"status": "ok"})
@flask_app.route("/cerrar")
def cerrar_flask():
   command_queue.put('cerrar')
   return jsonify({"status": "ok"})
@flask_app.route("/iniciar")
def iniciar_flask():
   command_queue.put('iniciar')
   return jsonify({"status": "ok"})
@flask_app.route("/resolver")
def resolver_flask():
   command_queue.put('resolver')
   return jsonify({"status": "ok"})
@flask_app.route('/parar')
def emergencia_flask():
   command_queue.put('parar')
```

```
    return jsonify({"status": "ok"})
@flask_app.route("/ping")
def ping_flask():
    return "CUBIBOT_HERE"
def run_flask():
     flask_app.run(host='0.0.0.0', port=8080, threaded=True)
if __name__ == "__main__":
    threading.Thread(target=run_flask, daemon=True).start()
```

Aunque este proyecto se enfoca en un entorno controlado, se tomaron algunas medidas básicas:

- Validación de parámetros recibidos.
- Separación de responsabilidades entre servidor y controlador de hardware.
- Restricción del acceso al puerto 5000 a dispositivos autorizados.

En un entorno de producción, se recomendaría incluir autenticación y uso de HTTPS.

El servidor backend desarrollado en Flask permite la integración fluida entre los distintos módulos del sistema, facilitando la detección de colores, el control de hardware y la resolución automática del cubo Rubik. Su arquitectura modular, flexible y liviana fue clave para el correcto funcionamiento del proyecto, logrando una comunicación eficiente con los clientes Tkinter y Android, así como una gestión robusta de las acciones internas.

CAPITULO III

3. MARCO METODOLOGICO

El desarrollo de este proyecto se realizó de forma iterativa, aplicando la metodología ágil SCRUM como base para organizar y estructurar el trabajo, junto a ello se implementó la gestión de proyectos monitoreado por GitLab lo cual facilita la obtención de organización en el análisis de requerimientos a medida que va avanzando el desarrollo.

A lo largo del proceso, se fueron abordando las necesidades técnicas y funcionales para construir un sistema capaz de detectar, interpretar y resolver un cubo Rubik 3x3 de forma automática. La solución integra visión por computadora para el reconocimiento de colores, algoritmos para resolver el cubo, y control de hardware para ejecutar físicamente los movimientos necesarios, todo coordinado desde un entorno de desarrollo en Python y en Android Studio.

"El algoritmo de dos fases de Kociemba se implementó para calcular la secuencia óptima de movimientos para resolver el cubo Rubik" [3].

3.1. DESARROLLO

Para llevar adelante este proyecto, se decidió trabajar siguiendo un enfoque metodológico basado en principios ágiles. Esta elección permitió organizar el trabajo en etapas más pequeñas, manejables y flexibles, entregando avances funcionales en cada ciclo de desarrollo, por lo que construir un sistema que uniera componentes físicos como la cámara y los servomotores, el procesamiento digital, visión por computadora, algoritmos de resolución y visualización gráfica, trajo consigo varios retos técnicos y de integración.

Precisamente por eso, resultó indispensable utilizar una metodología que ofreciera suficiente flexibilidad para avanzar de forma progresiva, incorporar mejoras de manera continua y resolver cualquier imprevisto que surgiera, sin comprometer la estructura ni el objetivo general del proyecto.

3.2. ANALISIS DE REQUERIMIENTOS

Los requisitos se centraron en lograr que el sistema pudiera detectar el estado de un cubo Rubik 3x3 y resolverlo de forma autónoma, integrando captura de imagen, análisis de colores, cálculo de movimientos y ejecución física.

Requerimientos funcionales:

- Capturar imágenes del cubo desde múltiples ángulos.
- Generar una representación digital del estado del cubo.
- Calcular la solución utilizando un algoritmo confiable.
- Ejecutar la solución a través de movimientos físicos del cubo.
- Mostrar una vista visual del proceso de resolución.

Requerimientos no funcionales:

- Alta precisión en la detección de colores.
- Tiempo razonable de resolución.
- Capacidad de adaptación a distintas condiciones de iluminación.

3.3. IMPLEMENTACION DE LA METODOLOGIA SCRUM

Scrum permitió planificar y ejecutar el proyecto de forma ordenada. Cada etapa se gestionó mediante sprints, en los cuales se realizaron reuniones de planificación, revisión y retroalimentación.

Cada sprint incluyó planificación, ejecución, revisión de avances y retroalimentación. De esta forma se mantuvo un ritmo constante y se pudieron tomar decisiones iterativas con base en los resultados obtenidos en cada fase.

Además, se emplearon herramientas visuales como tableros de tareas y gráficos burndown, que facilitaron el seguimiento del progreso y ayudaron a mantener la organización del proyecto.

"La gestión del proyecto se llevó a cabo utilizando la metodología SCRUM, siguiendo las directrices establecidas en The Scrum Guide." [4]

3.4. DEFINICION DE LOS SPRINTS

Definir a los Sprints, significa que todos aquellos son relacionados con metodologías ágiles, como Scrum, y se utiliza para explicar cómo se organizó el proyecto en los siguientes 4 Sprints, es decir, en ciclos iterativos y cortos, para realizarlo en el tiempo optimo y necesario para su presentacion.

En el marco de trabajo Scrum, el desarrollo del proyecto se organizó en Sprints, que son períodos de tiempo fijos (generalmente de una a cuatro semanas) durante los cuales se planifica, desarrolla y entrega un conjunto de funcionalidades completamente funcionales y probadas del sistema.

La estructura del proyecto se organizó en cuatro sprints principales:

Sprint 1: Preparación del entorno físico, configuración de motores y pruebas básicas de movimiento.

Elemento	Detalle
Objetivo	Asegurar que el sistema pueda enviar comandos a los servomotores y capturar vídeo estable desde la cámara para detección de color.
Historias de Usuario	- Historia de usuario 1: Como desarrollador, quiero controlar los servomotores, para poder mover las piezas del robot automáticamente. - Historia de usuario 2: Como desarrollador, quiero capturar y procesar la cámara web con OpenCV, para detectar correctamente los colores de las caras del cubo.
Tareas Técnicas	- Conectar servomotores al controlador. - Configurar librería serial. - Calibrar servos (0–90°). Mapear movimientos de servomotores. - Inicializar OpenCV. - Crear ventana para la webcam. - Definir rangos HSV de colores del cubo Rubik. - Realizar condición para excluir casos especiales. - Finalizar detección de colores en 9 regiones específicas.
Criterios de Aceptación	- Los servos responden sin fallo a comandos de 0° y 90°. - El feed de cámara mantiene ≥ 15 FPS durante 5 minutos. - Los rangos HSV detectan correctamente un patrón de prueba de color sin falsos positivos.
Duración estimada	1 semana

Tabla 2: Tabla de preparación para el Sprint 1

Sprint 2: Desarrollo de la funcionalidad de captura de imágenes y control de la cámara.

Elemento	Detalle
Objetivo	Proporcionar una vista previa que muestre los 54 stickers detectados y una interfaz mínima para arrancar, pausar y reiniciar el sistema.
Historias de Usuario	- Historia de usuario 3: Como desarrollador, quiero ver una vista previa con los 54 colores detectados, para verificar que todo se detectó correctamente. - Historia de usuario 4: Como usuario, quiero tener una interfaz gráfica intuitiva, para iniciar el proceso y ver el estado del cubo Rubik.
Tareas Técnicas	- Crear módulo de dibujo 3×3×6 para stickers. - Mapear como estado, las caras del cubo. - Mapear según kociemba los centros únicos para la plantilla. - Usar los colores detectados con OpenCV, y colocarlos en la plantilla. - Desarrollar GUI con botones “Iniciar”, “Pausar”, “Reiniciar”. - Integrar feed de cámara con la vista de colores. - Realizar ejecución en hilos para que se actualice tanto motores y cámara. - Implementar ejecución de programa en un solo botón.
Criterios de Aceptación	- La ventana muestra simultáneamente vídeo y plantilla sin bloqueos. - Cada sticker refleja correctamente el color detectado en tiempo real. - Los botones responden en < 200 ms y cambian el estado del proceso.
Duración estimada	1 semana

Tabla 3: Tabla de preparación para el Sprint 2

Sprint 3: Implementación del sistema de reconocimiento de colores y generación de la plantilla digital del cubo.

Elemento	Detalle
Objetivo	Integrar el algoritmo de Kociemba y traducir la solución resultante en movimientos físicos de los servomotores.
Historias de Usuario	- Historia de usuario 5: Como usuario, quiero que el sistema resuelva el cubo Rubik, para que me muestre los pasos exactos de cómo se está solucionando también mediante servomotores.
Tareas Técnicas	- Instalar e integrar librería Kociemba. - Mapear notación (“F”, “L”, “B”, “R”, etc.) a funciones de servomotores. - Realizar una vista previa del manejo de solución una vez detecto los colores. - Guardar e imprimir la solución. - Crear script de ejecución automática para realizar solución con servomotores. - Implementar solución a las funcionalidades del GUI
Criterios de Aceptación	- El algoritmo resuelve cualquier scramble válido en ≤ 20 movimientos. - Cada movimiento de servo coincide con el paso calculado (± 2° de tolerancia). - La GUI muestra la secuencia paso a paso correctamente.
Duración estimada	1 semana

Tabla 4: Tabla de preparación para el Sprint 3

Sprint 4: Cálculo de la solución, ejecución de movimientos físicos y visualización del proceso de resolución.

Elemento	Elemento | Detalle
Objetivo	Habilitar control remoto mediante una app Android y consolidar/refactorizar el código para escalabilidad y presentación.
Historias de Usuario	- Historia de usuario 6: Como usuario, quiero usar una app Android para controlar el sistema, para operar el cubo remotamente sin usar el PC. - Historia de usuario 7: Como desarrollador, quiero revisar y consolidar el código del sistema, para tener un sistema listo para escalar o presentar.
Tareas Técnicas	- Desarrollar app Android con botones “Iniciar”, “Resolver”, “Pausar” y panel de estado. - Definir protocolo de conexión, servidor-celular. - Refactorizar código en módulos: captura, visión, GUI, resolución, comunicaciones para conexión con Flask. - Escribir pruebas unitarias y de integración. - Actualizar documentación de despliegue y usuario.
Criterios de Aceptación	- La app Android controla el sistema en < 150 ms de latencia. - Comunicación remota estable durante 10 minutos sin desconexiones. - Cobertura de pruebas ≥ 80 % en módulos refactorizados. - Documentación actualizada y PDF de manual disponible.
Duración estimada	1 semana

Tabla 5: Tabla de preparación para el Sprint 4

Esta organización por etapas permitió desarrollar el proyecto de manera progresiva, resolviendo primero lo esencial antes de avanzar a lo más complejo.

3.5. PLANIFICACION DE LOS SPRINTS

Antes de iniciar cada sprint, se realizó una planificación que incluyó la selección de tareas prioritarias desde el Product Backlog.

Cada tarea fue descompuesta en subtareas más pequeñas, se estimó su complejidad en puntos de historia y se asignaron responsables.

Durante cada sprint se utilizó un tablero de tareas (taskboard) para visualizar el avance, y un gráfico burndown para hacer seguimiento al ritmo de trabajo. Esto ayudó a mantener el enfoque y a identificar bloqueos a tiempo.

3.5.1. TASKBOARD INICIAL Y BURNDOWN CHAT INICIAL

El taskboard inicial trata de un gráfico que tiene columnas: "Por hacer", "En progreso", "En revisión" y "Completado". A medida que se avanza, las tareas se ven moviendo de una columna a otra.

El burndown chart muestra cómo se cumple y se baja la cantidad de tareas pendientes, ayudando a ver ajustes necesarios para cuando existan dificultades técnicas o algunas nuevas ideas para el proyecto, pero en este caso como se trabajó en GitLab debemos de usar la versión Premium para mostrar el burndown chart.

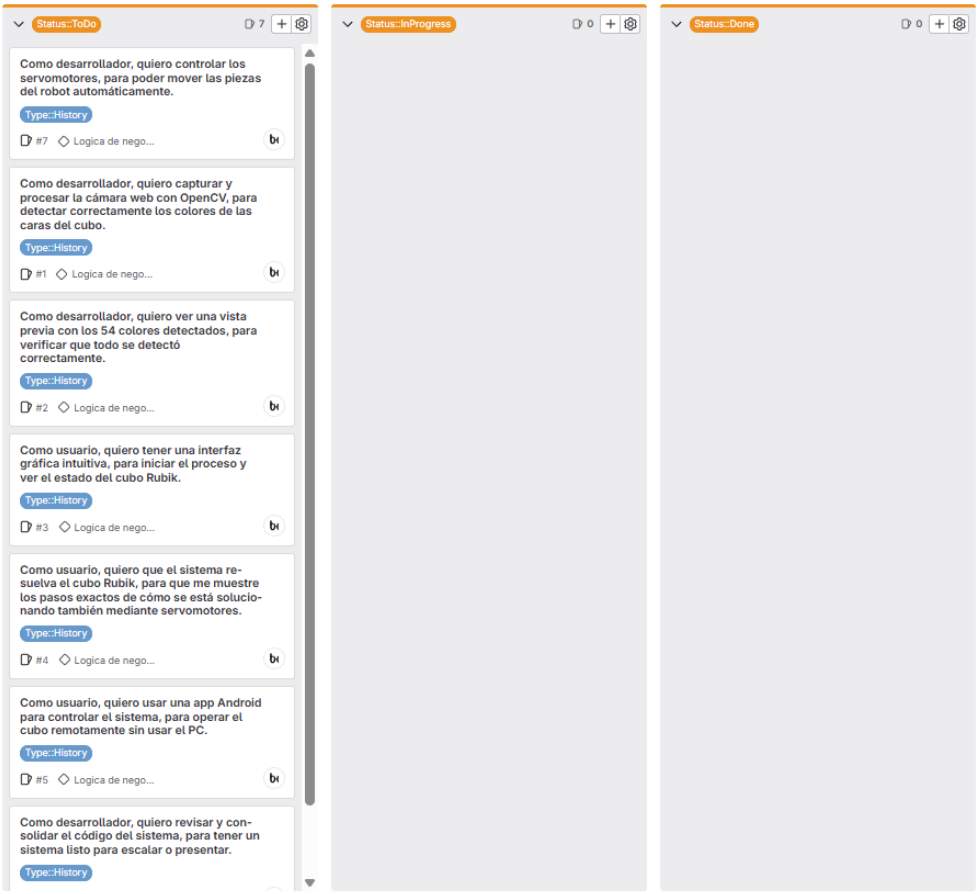


Figura 9: Representación de la tabla de tareas ´Por hacer', 'En progreso' y 'Hecho' programadas como historias de usuario.

3.6. DESARROLLO DE LA APP

A continuación, se presenta un resumen de los avances durante cada sprint:

```
PS C:\Users\lopez\OneDrive\Desktop\cubibot> git clone https://gitlab.com/imborix/cubibot.git
```

Figura 10: Antes de desarrollar, se creo una carpeta planificada de GitLab en el escritorio, cumpliendo con el repositorio en línea, para trabajar los avances.

3.6.1. SPRINT 1

En esta primera fase del proyecto se pusieron los cimientos sobre los cuales se desarrollaría todo el sistema. Se procedió a configurar el entorno de trabajo, asegurando que tanto el hardware como el software estuvieran listos para comenzar.

Durante este sprint, también se conectaron los primeros componentes físicos, entre ellos los servomotores y la estructura de soporte, y se realizaron las pruebas iniciales de movimiento para garantizar su correcto funcionamiento. Aunque en esta etapa aún no se abordaron funcionalidades complejas, su importancia fue crucial: permitió verificar la estabilidad básica del sistema y preparar el terreno para añadir nuevas capas de funcionalidades en los siguientes ciclos de desarrollo.

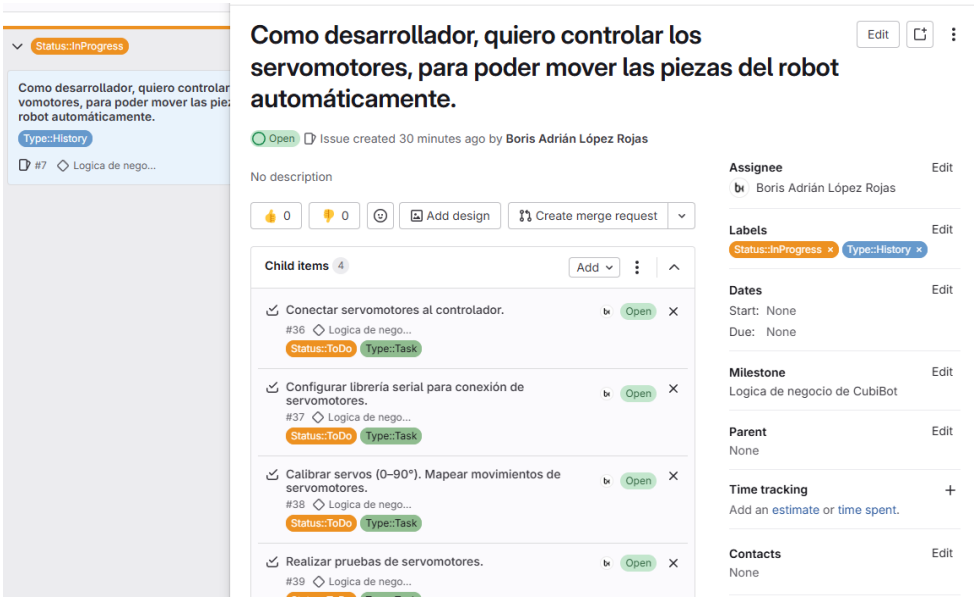


Figura 11: Desarrollo del primer sprint en GitLab, en este caso es la representación de una Historia de Usuario.

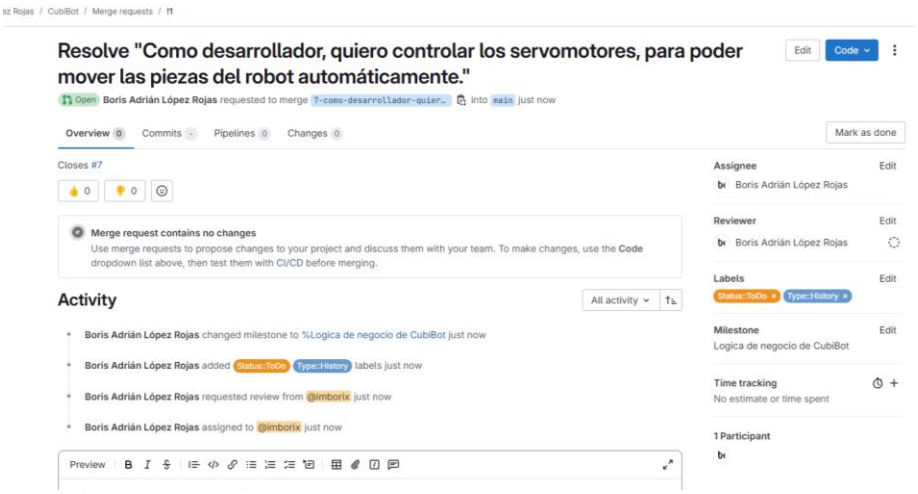


Figura 12: Representación de un merge request, definido por GitLab en la historia de usuario planteada anteriormente, para subir los cambios de ese repositorio.

Figura 13: Revisión de las ramas existentes, en este caso la rama main principal y la realizada para el merge requerst.

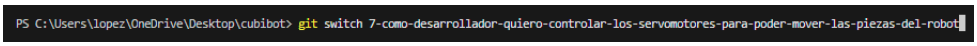


Figura 14: Ejecución de salto hacia la rama que está definida como primera para realizar las tareas definidas por la Historia de Usuario.

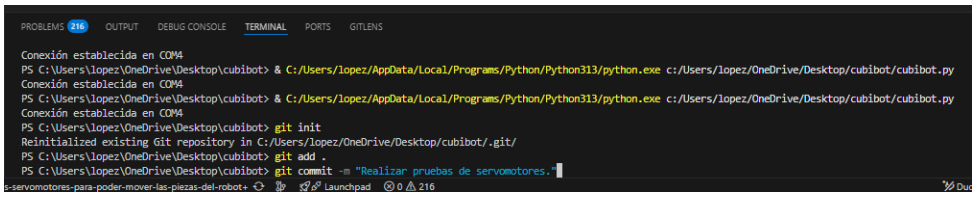


Figura 15: Proceso de agregado y commit de una de las tareas de la Historia de Usuario.

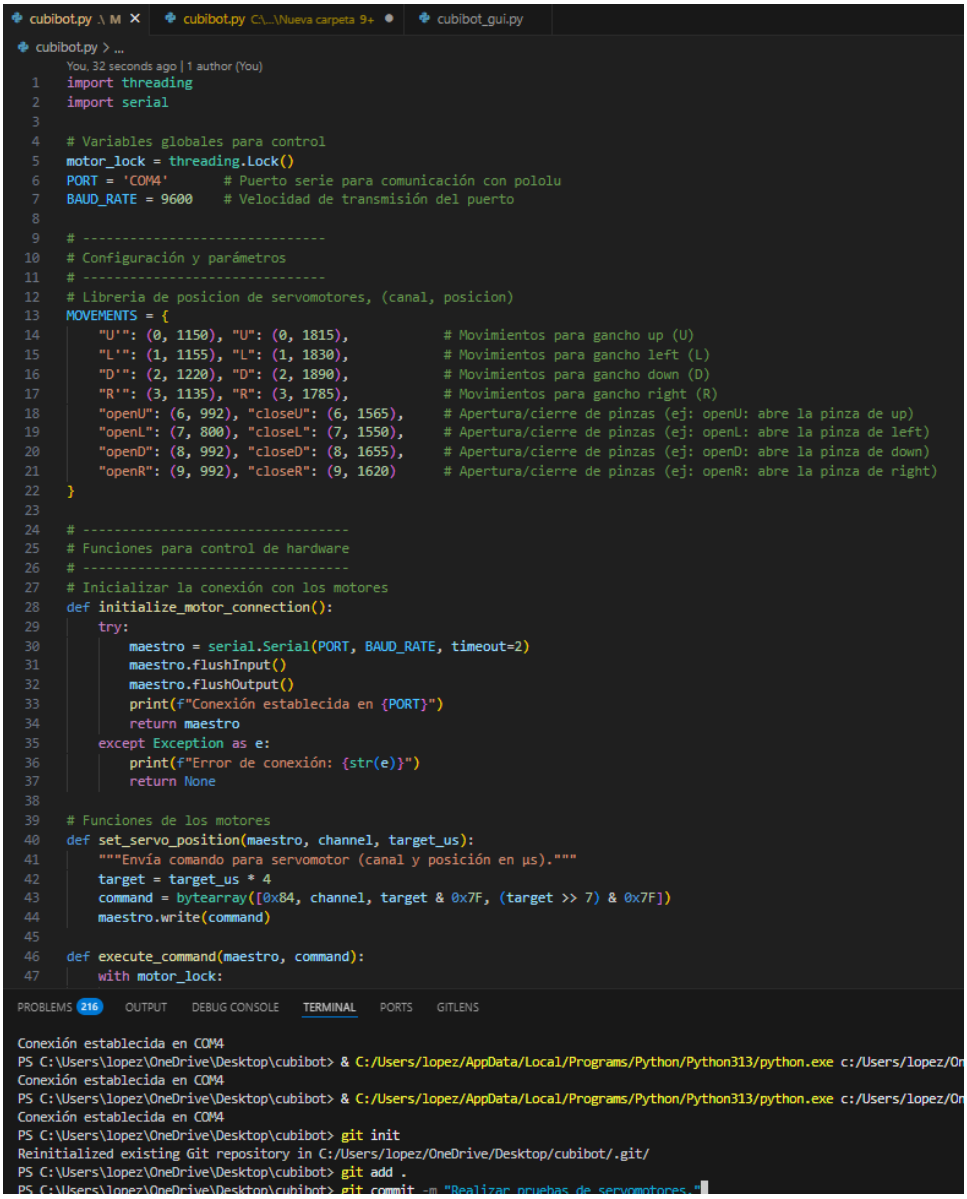


Figura 16: Representación del código commitido y subido al repositorio para revisión con el merge request hacia el main del proyecto.

3.6.2. SPRINT 2

Una vez que la base del sistema estuvo establecida, el siguiente paso fue integrar la cámara al conjunto. Se trabajó en desarrollar toda la lógica que permitiría capturar imágenes de manera controlada y confiable.

Se incorporó además una ventana de vista previa que mostraba en tiempo real la imagen captada por la cámara, permitiendo al usuario asegurarse de que las capturas fueran de buena calidad y estuvieran correctamente enfocadas.

Durante este sprint, se realizaron numerosas pruebas para garantizar que las imágenes capturadas fueran nítidas y libres de desenfoques, requisito fundamental para el éxito de los procesos de análisis de color que se aplicarían posteriormente.

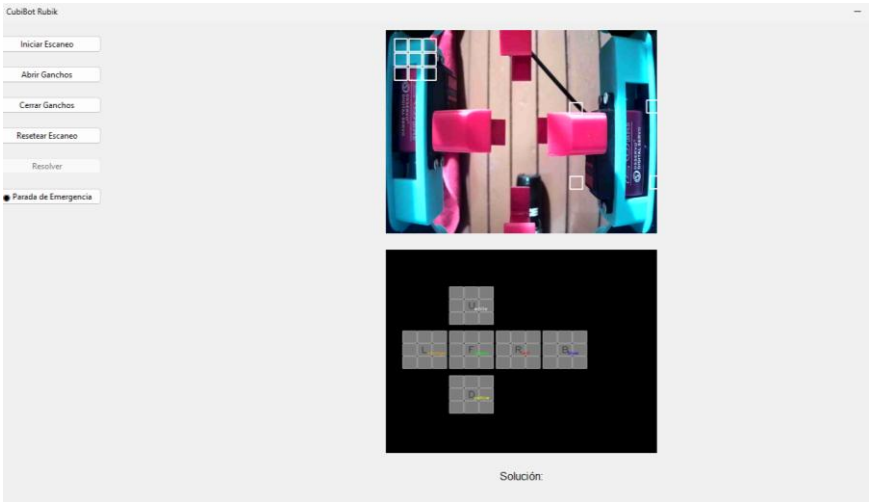


Figura 17: Vista de la representación grafica realizada para manejar funciones del proyecto como un GUI general.

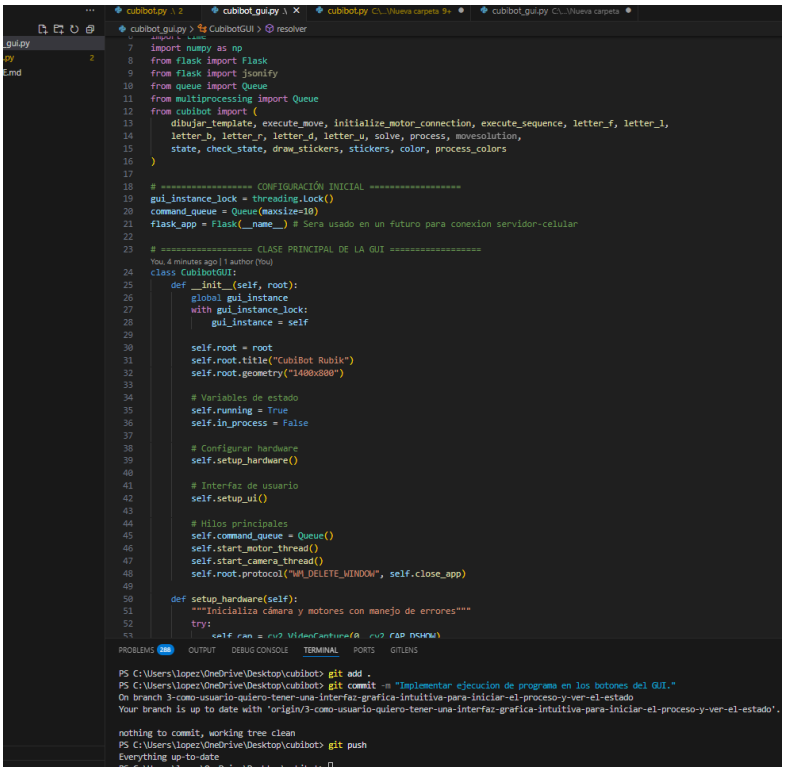


Figura 18: Vista del código desarrollado en la sección de Sprint 2 lo cual se realizo sus respectivas commits y actualizaciones de tareas, según las Historias de Usuario.

3.6.3. SPRINT 3

Con la capacidad de capturar imágenes ya implementada y validada, se avanzó hacia la detección de colores en las casillas del cubo Rubik.

Para ello, se diseñó un sistema que trabaja sobre imágenes previamente validadas, utilizando el modelo de color HSV. Este modelo resulta más confiable que el modelo RGB tradicional, especialmente en situaciones donde la luz puede variar, lo cual ayudó a obtener una detección de colores mucho más precisa.

Además, se implementó la capacidad de comparar dos imágenes de una misma cara del cubo, para asegurarse de reconocer correctamente los colores incluso en áreas parcialmente cubiertas o afectadas por reflejos.

Gracias a este proceso, se pudo construir una plantilla digital que reflejaba con exactitud el estado actual del cubo Rubik, sirviendo de base para calcular su solución.

Figura 19: Vista de una nueva solicitud de merge request para realizar una tarea nueva que solicite las nuevas Historias de Usuario.

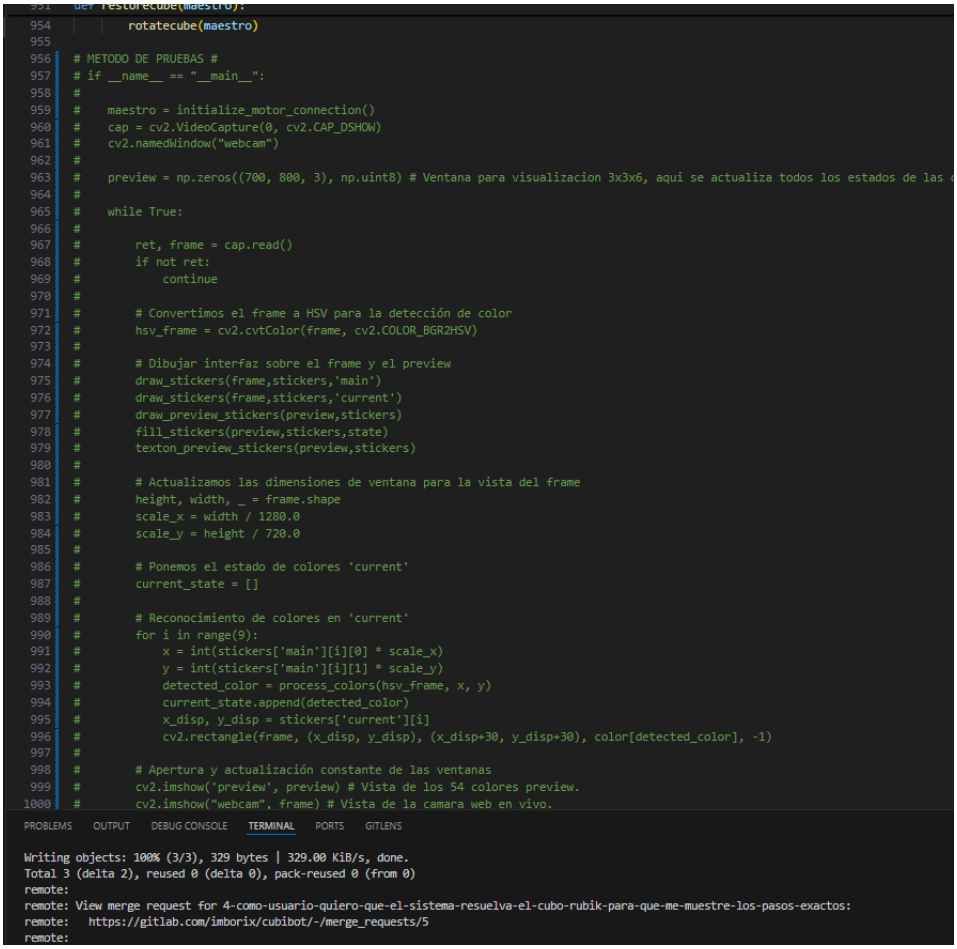


Figura 20: Vista del código realizado por las tareas actuales del Sprint según las Historias de usuario establecidas en este determinado paso.

3.6.4. SPRINT 4

Con la plantilla del cubo completa, el siguiente objetivo fue calcular la solución de manera eficiente. Para ello, se utilizó un algoritmo optimizado basado en el método de Kociemba, ampliamente reconocido por su capacidad de encontrar soluciones rápidas y eficaces para el cubo Rubik.

Una vez generada la solución, los movimientos necesarios se tradujeron en comandos que podían ser ejecutados por los servomotores, permitiendo que el sistema resolviera físicamente el cubo de forma automática.

Adicionalmente, se integró una visualización gráfica paso a paso, para que el usuario pudiera observar de manera clara cómo se desarrollaba todo el proceso de resolución, haciendo que la experiencia fuera no solo funcional, sino también didáctica y fácil de seguir.

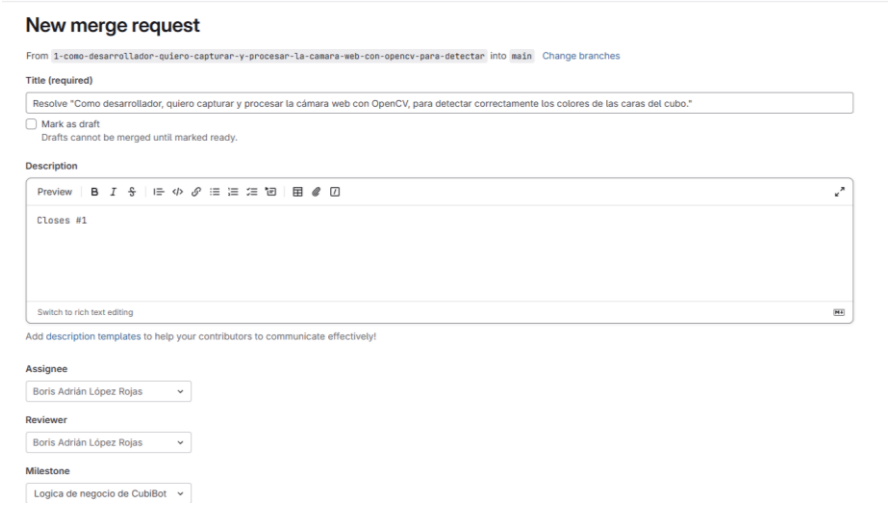


Figura 21: Vista de una nueva solicitud de merge request para realizar una tarea nueva que solicite las nuevas Historias de Usuario para el Sprint 4.

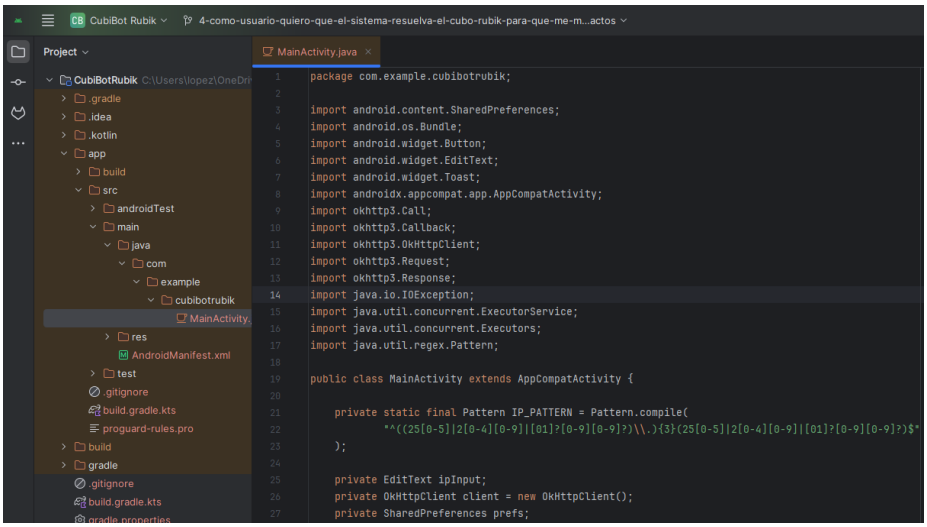


Figura 22: Vista del código realizado para tareas de Sprint 4, modificadas por sus Historias de Usuario.

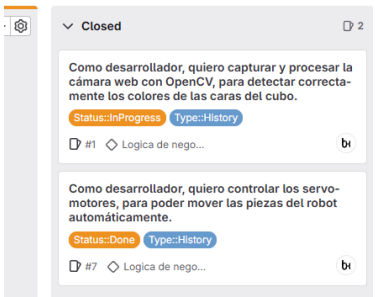


Figura 23: Vista de la culminación de Sprint 4 con las Historias de Usuario cerradas y completadas en el proceso.

3.7. CIERRE

Finalmente, el proyecto culminó con un sistema plenamente funcional, que cumplió con el objetivo planteado: capturar el estado de un cubo Rubik, procesar la información visual recogida, calcular una solución óptima y ejecutar esa solución mediante movimientos mecánicos precisos.

A lo largo de todo el proceso de desarrollo, se hizo evidente la importancia de avanzar de manera progresiva y de validar cada etapa antes de continuar con la siguiente. La integración continua de componentes y la revisión constante de resultados fueron claves para construir un sistema robusto y confiable.

El uso de la metodología Scrum resultó ser una estrategia acertada. Permitió mantener la organización del trabajo de forma estructurada pero flexible, facilitando la adaptación a los distintos desafíos técnicos que surgieron en el camino.
Gracias a esta metodología, fue posible identificar problemas a tiempo, ajustar el enfoque cuando fue necesario, y entregar avances funcionales al finalizar cada sprint, asegurando así la calidad y el éxito del proyecto.

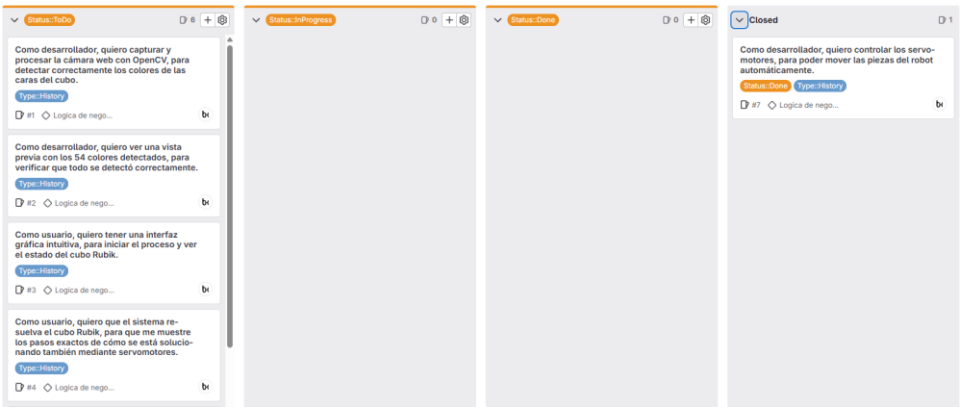


Figura 24: Vista del inicio de Sprints con las tareas en pendiente y las realizadas por las Historias de Usuario según los sprints.

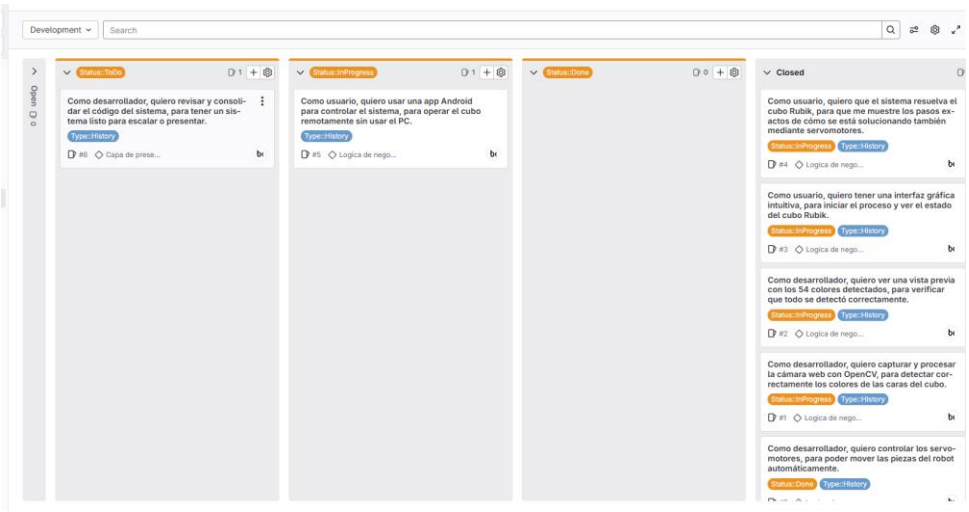


Figura 25: Vista final de las Historias de Usuario cerradas para su finalización como cierre del proceso de Sprints culminándolo después del Sprint 4.

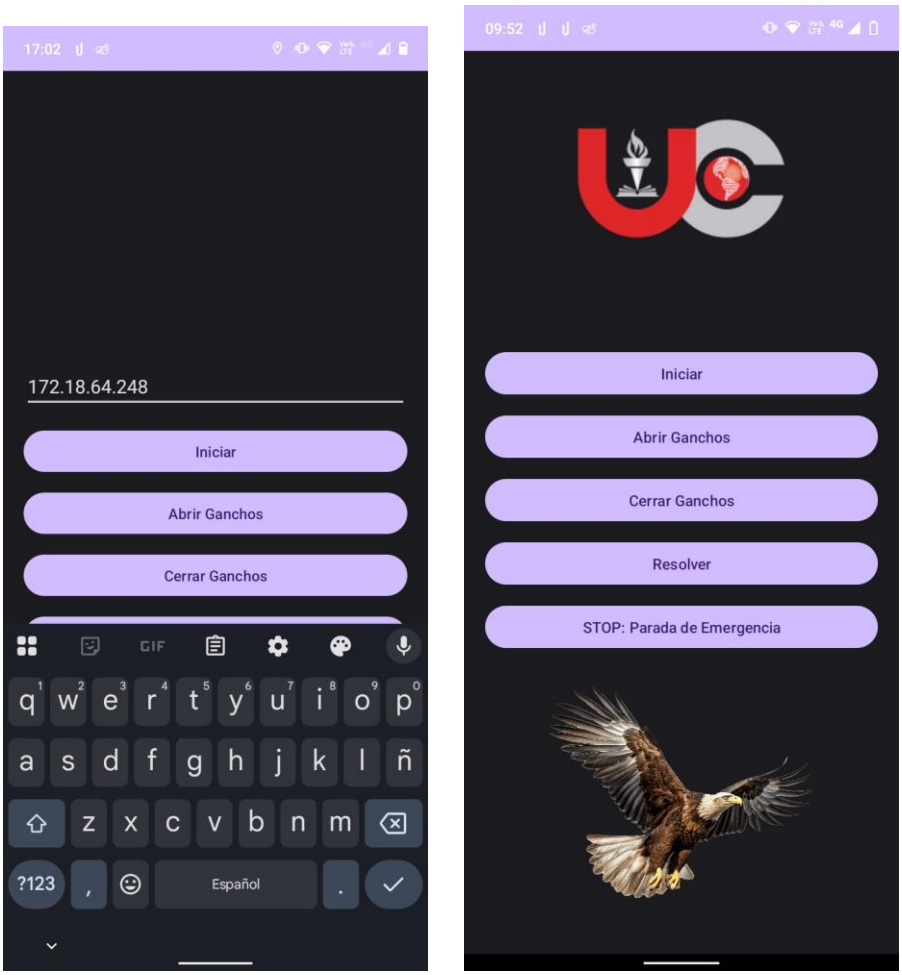


Figura 26: Vista de la GUI creada para la aplicación Android que controla el proyecto de manera remota.

CAPITULO IV

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Al llegar al final de este proyecto, se pudo cumplir con el propósito principal: desarrollar un sistema funcional que resolviera de manera autónoma un cubo Rubik 3x3, combinando de forma efectiva tanto hardware como software.

A lo largo del proceso, se demostró que es posible integrar visión por computadora, el control de servomotores y algoritmos de resolución, para construir un sistema capaz de interactuar con su entorno físico de forma inteligente y precisa.

En cuanto a los resultados, el sistema logró resolver el cubo en un tiempo promedio de 45 segundos por ciclo completo, considerando todo el proceso: desde capturar las imágenes, analizar los colores, calcular los movimientos necesarios, hasta ejecutar físicamente la solución.

Un punto clave fue alcanzar una precisión superior al 95% en la detección de colores, siempre que se trabajara en condiciones de iluminación controlada. Para reforzar la fiabilidad del sistema, se añadieron mecanismos de validación que aseguraban la calidad de las imágenes antes de procesarlas.

La implementación de la metodología ágil Scrum resultó fundamental. Permitió mantener el enfoque, organizar las tareas de forma ordenada y adaptarse rápidamente frente a los desafíos técnicos que surgieron durante el desarrollo. Gracias a esta estructura de trabajo, fue posible avanzar de manera progresiva y entregar resultados funcionales al finalizar cada sprint.

Como parte del proyecto, también se construyó una interfaz visual que permite a los usuarios observar en tiempo real la captura de imágenes y el proceso de resolución paso a paso del cubo. Además, se diseñó una aplicación móvil que facilita el control remoto del sistema, integrando funciones de reconocimiento automático, lo que brinda una experiencia de uso clara, intuitiva y accesible.

4.2. RECOMENDACIONES

A partir del desarrollo de este proyecto, se propondrán las siguientes recomendaciones en caso que se desee desarrollar futuras versiones o ampliaciones:

- Impresión con materiales resistentes: Mejorar la calidad de material de impresión 3D para que no exista inconvenientes si los movimientos ocasionan inconsistencias, incluso con los colores del cubo Rubik de esa manera evitamos condiciones de exclusión especiales.

- Incorporar machine learning: Usar herramientas de machine learning para calibrar dentro del programa sin necesidad de utilizar los umbrales HSV predeterminados si no que se adapten a las condiciones de luz, mejoraría la adaptabilidad del sistema.

- Mejorar el hardware de mecánica y automatización: Implementar cambio de servomotores a motores de precisión o algoritmos más cercanos a mejorar el rendimiento de este controlador Pololu de 18 canales para suavizar y acelerar los movimientos de los motores, reduciendo el tiempo total de resolución.

- Ampliar a cubos de mayor tamaño: Adaptar el sistema para cubos 4x4 o 5x5 representaría un avance técnico interesante, aunque implicaría un rediseño en la lógica de solución y la captura de datos.

- Implementar una base de datos: Registrar tiempos, soluciones, errores detectados y condiciones del entorno permitiría analizar el desempeño del sistema a lo largo del tiempo y mejorar continuamente.

- Mejorar la interfaz gráfica: Una interfaz aún más intuitiva podría facilitar el uso del sistema por parte de cualquier usuario, incluso sin conocimientos técnicos.

REFERENCIAS BIBLIOGRAFICAS

[1] OpenCV, "Documentación oficial de Open Source Computer Vision Library," 2024. [En línea]. Disponible: [Insertar URL oficial de la documentación de OpenCV].

[2] Pololu, "Guías y ejemplos de uso de Pololu Micro Maestro Servo Controller," 2023. [En línea]. Disponible: [Insertar URL oficial de las guías de Pololu].

[3] H. Kociemba, "A Two-Phase Algorithm for the Rubik's Cube," 1992. [En línea]. Disponible: [Insertar URL si está disponible en línea].

[4] Scrum.org, "The Scrum Guide," 2022. [En línea]. Disponible: [Insertar URL oficial de la guía Scrum].

[5] Python Software Foundation, "Documentación oficial de Python," 2024. [En línea]. Disponible: [Insertar URL oficial de la documentación de Python].

[6] R. T. Vaughan, "Massively distributed control: A paradigm for robot teams," in *Proc. 2000 IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, 2000, vol. 4, pp. 3433-3438.

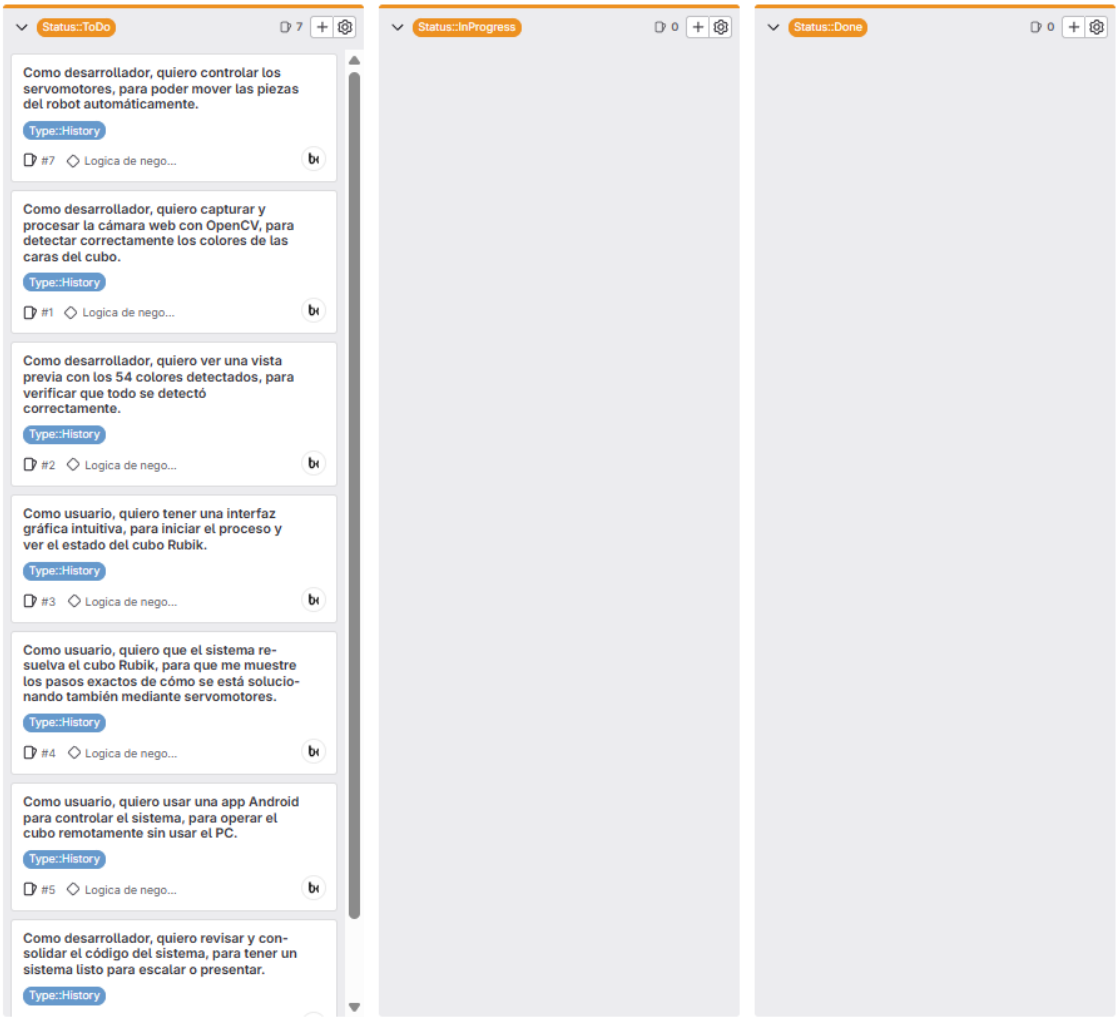
[7] M. Bers, "The role of robotics in early childhood education," *Early Childhood Research & Practice*, vol. 4, no. 2, 2002.

[8] V. Kumar, "Robotics and automation for STEM education," in *Proc. 2014 IEEE Int. Conf. on Robotics and Automation*, Hong Kong, China, 2014, pp. 622-627.

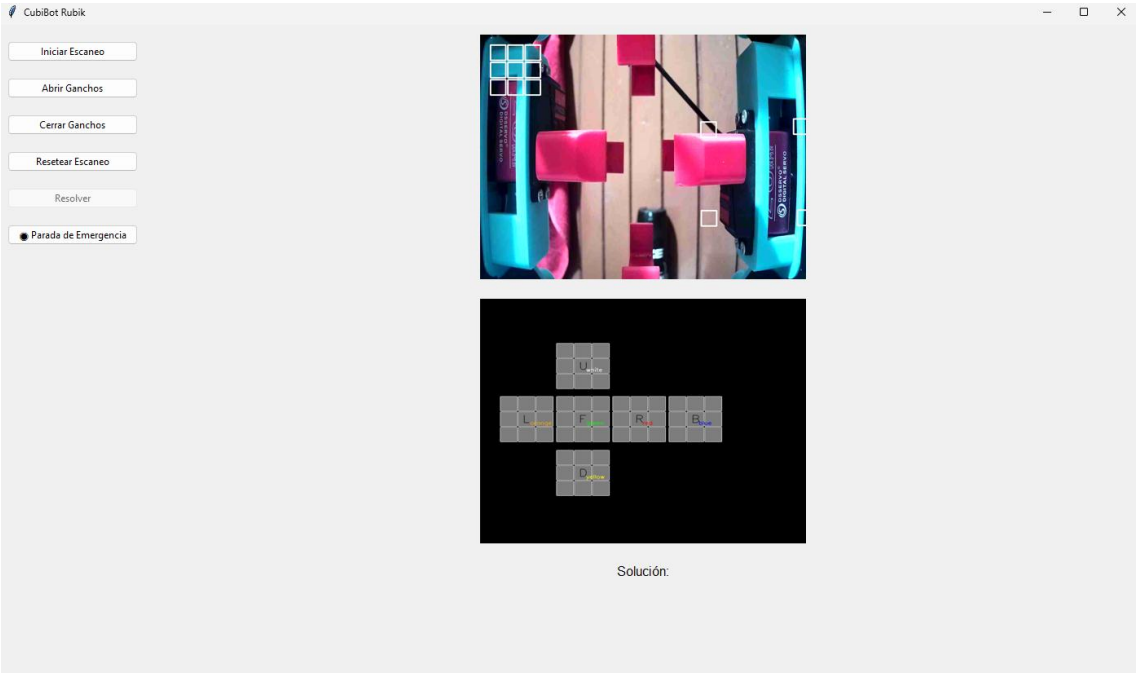
[9] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco, CA: W.H. Freeman, 1982.

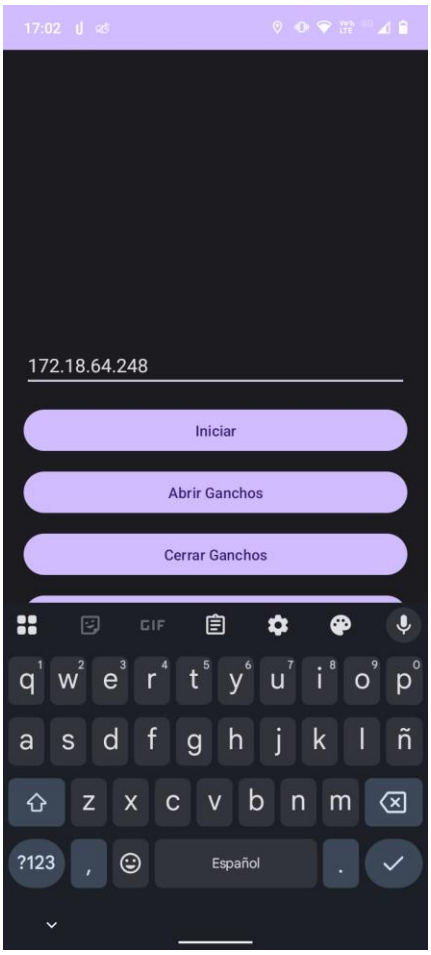
ANEXOS

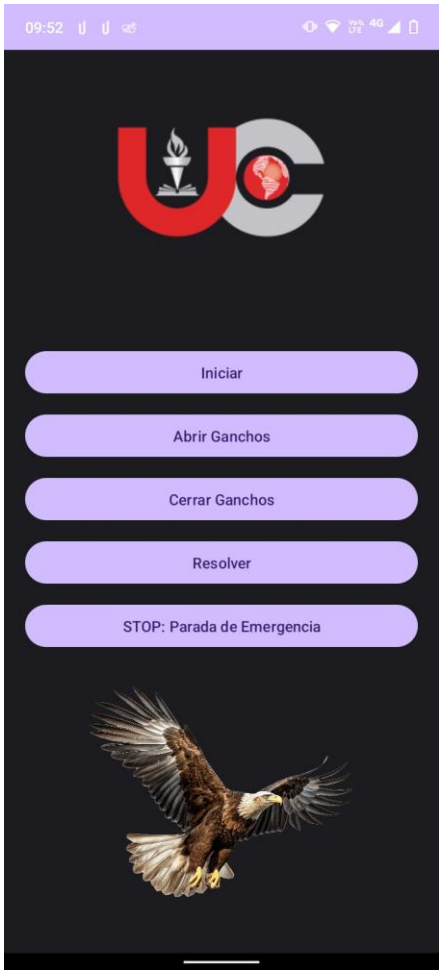
Anexo A. Taskboard inicial



Anexo B. Captura de la interfaz visual



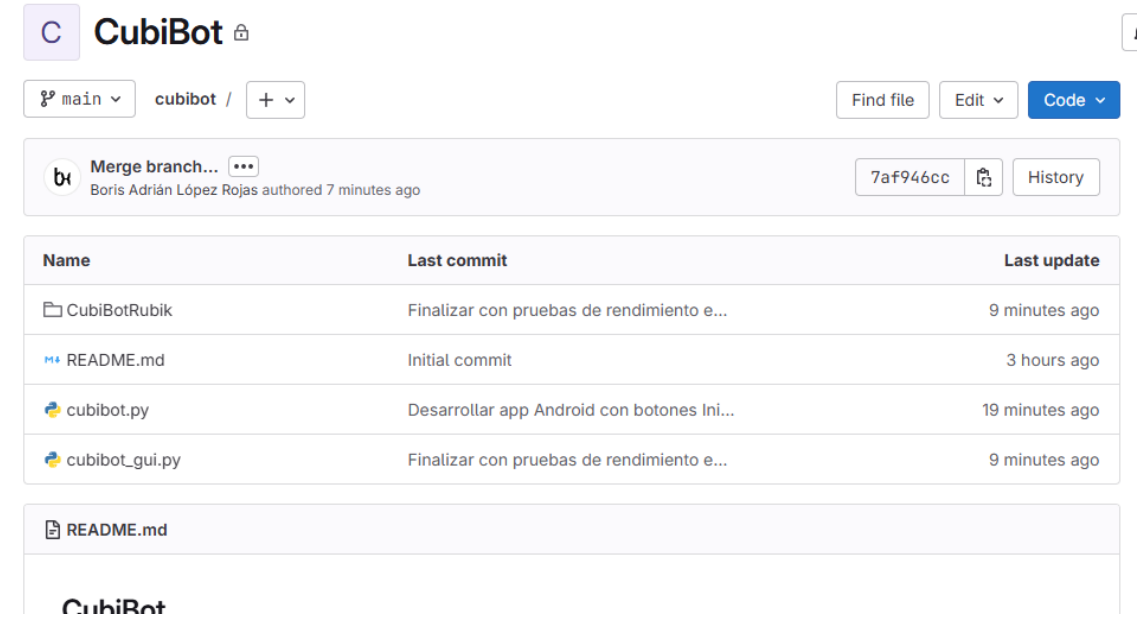


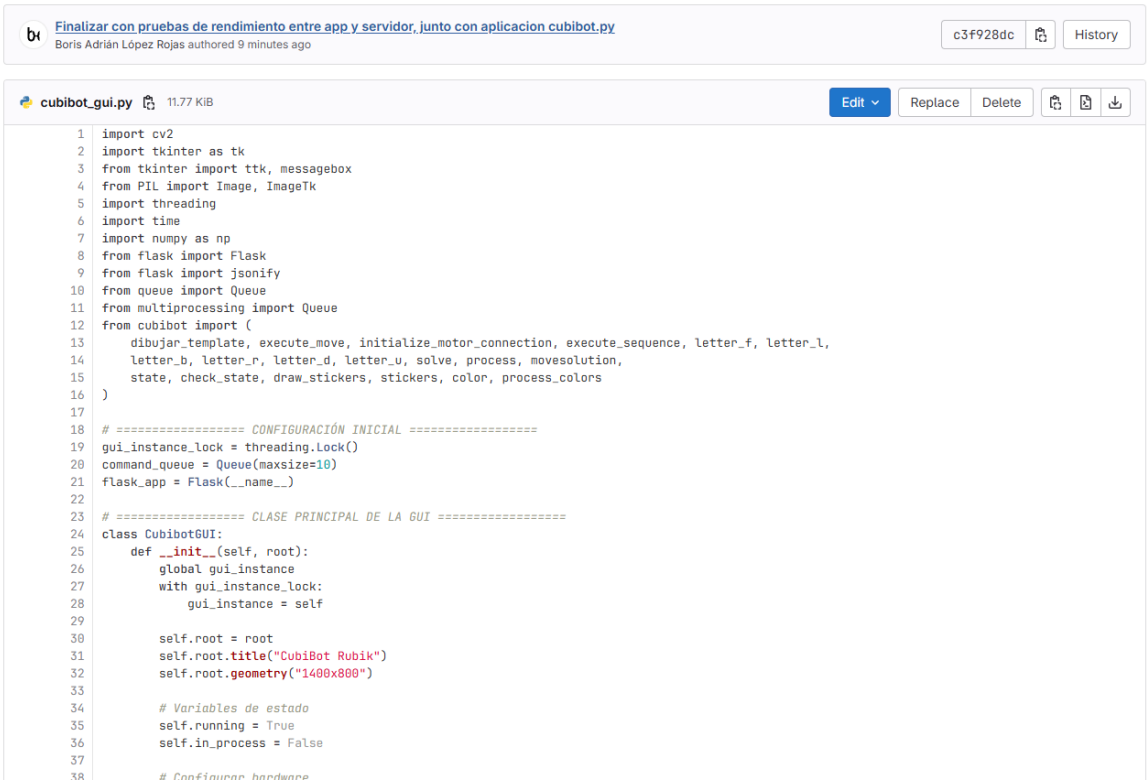


Anexo C. Secuencia de resolución del cubo

Anexo D. Prototipo finalizado, funcionando

Anexo E. Código fuente organizado






```python
import cv2
import tkinter as tk
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
import threading
import time
import numpy as np
from flask import Flask
from flask import jsonify
from queue import Queue
from multiprocessing import Queue
from cubibot import (
    dibujar_template, execute_move, initialize_motor_connection, execute_sequence, letter_f, letter_l,
    letter_b, letter_r, letter_d, letter_u, solve, process, movesolution,
    state, check_state, draw_stickers, stickers, color, process_colors
)

# ================== CONFIGURACIÓN INICIAL ==================
gui_instance_lock = threading.Lock()
command_queue = Queue(maxsize=10)
flask_app = Flask(__name__)

# ================== CLASE PRINCIPAL DE LA GUI ==================
class CubibotGUI:
    def __init__(self, root):
        global gui_instance
        with gui_instance_lock:
            gui_instance = self

        self.root = root
        self.root.title("CubiBot Rubik")
        self.root.geometry("1400x800")

        # Variables de estado
        self.running = True
        self.in_process = False

        # Configurar hardware
```

cubibot.py 60.69 KiB — Edit | Replace | Delete

```python
import threading
import time
import serial
import cv2 # Libreria OpenCV
import numpy as np
import kociemba as Cube # Libreria Kociemba

# Variables globales para control
motor_lock = threading.Lock()
PORT = 'COM4'       # Puerto serie para comunicación con pololu
BAUD_RATE = 9600    # Velocidad de transmisión del puerto
motortiempo = 1     # Tiempo de espera entre movimientos (segundos)
check_state=[]      # Lista de estados, donde definimos como una serie de funciones en una letra, determinando que se refiere a una cara del cubo.
font = cv2.FONT_HERSHEY_SIMPLEX # Fuente para texto en OpenCV
solved=False    # Variables para futuras actualizaciones de solucion.
solution=[]     # Variable para detectar la lista de pasos de soluciones.

# ------------------------------
# Configuración y parámetros
# ------------------------------
# Libreria de posicion de servomotores, (canal, posicion)
MOVEMENTS = {
    "U'": (0, 1150), "U": (0, 1815),            # Movimientos para gancho up (U)
    "L'": (1, 1155), "L": (1, 1830),            # Movimientos para gancho left (L)
    "D'": (2, 1220), "D": (2, 1890),            # Movimientos para gancho down (D)
    "R'": (3, 1135), "R": (3, 1785),            # Movimientos para gancho right (R)
    "openU": (6, 992), "closeU": (6, 1565),     # Apertura/cierre de pinzas (ej: openU: abre la pinza de up)
    "openL": (7, 800), "closeL": (7, 1550),     # Apertura/cierre de pinzas (ej: openL: abre la pinza de left)
    "openD": (8, 992), "closeD": (8, 1655),     # Apertura/cierre de pinzas (ej: openD: abre la pinza de down)
    "openR": (9, 992), "closeR": (9, 1620)      # Apertura/cierre de pinzas (ej: openR: abre la pinza de right)
}

# Coordenadas de stickers, util para definir las regiones de deteccion de colores.
stickers = {
    'main': [[433, 171], [615, 165], [793, 161], [434, 345], [623, 344], [794, 343], [434, 526], [622, 527], [796, 526]],
    'current': [[20, 20], [54, 20], [88, 20], [20, 54], [54, 54], [88, 54], [20, 88], [54, 88], [88, 88]],
    'preview': [[20, 130], [54, 130], [88, 130], [20, 164], [54, 164], [88, 164], [20, 198], [54, 198], [88, 198]],
    'left': [[50, 280], [94, 280], [138, 280], [50, 324], [94, 324], [138, 324], [50, 368], [94, 368], [138, 368]],
    'front': [[188, 280], [232, 280], [276, 280], [188, 324], [232, 324], [276, 324], [188, 368], [232, 368], [276, 368]],
    'right': [[326, 280], [370, 280], [414, 280], [326, 324], [370, 324], [414, 324], [326, 368], [370, 368], [414, 368]],
    'up': [[188, 128], [232, 128], [276, 128], [188, 172], [232, 172], [276, 172], [188, 216], [232, 216], [276, 216]],
    'down': [[188, 434], [232, 434], [276, 434], [188, 478], [232, 478], [276, 478], [188, 522], [232, 522], [276, 522]],
    'back': [[464, 280], [508, 280], [552, 280], [464, 324], [508, 324], [552, 324], [464, 368], [508, 368], [552, 368]]
```

CubiBot

MainActivity.java 4.55 KiB — Edit | Replace | Delete

```java
package com.example.cubibotrubik;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import java.io.IOException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.regex.Pattern;

public class MainActivity extends AppCompatActivity {

    private static final Pattern IP_PATTERN = Pattern.compile(
            "^((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$"
    );

    private EditText ipInput;
    private OkHttpClient client = new OkHttpClient();
    private SharedPreferences prefs;
    private ExecutorService executor = Executors.newSingleThreadExecutor();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        prefs = getSharedPreferences("CubiBotPrefs", MODE_PRIVATE);
        initializeUI();
        loadSavedIP();
    }

    private void initializeUI() {
        ipInput = findViewById(R.id.editTextIP);

        Button btnIniciar = findViewById(R.id.btnIniciar);
```

AUTORIZACIÓN DE PUBLICACIÓN EN EL REPOSITORIO INSTITUCIONAL

Boris Adrián López Rojas portador(a) de la cédula de ciudadanía N.° **0106741978**. En calidad de autor/a y titular de los derechos patrimoniales del proyecto de titulación **"DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL CONTROL DE UN ROBOT AUTÓNOMO QUE RESUELVE CUBOS RUBIK 3X3 UTILIZANDO IMPRESIÓN 3D Y ALGORITMOS DE VISIÓN COMPUTACIONAL"** de conformidad a lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos y no comerciales. Autorizo además a la Universidad Católica de Cuenca, para que realice la publicación de este proyecto de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

Cuenca, **25 de abril de 2025**

F: .....

Boris Adrián López Rojas

C.I. **0106741978**