ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Diseño y desarrollo de una aplicación móvil para la recepción de datos mediante un sistema OCC

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Telemática

Presentado por:

Gustavo Gerardo Castillo Cusme

Daniel Alejandro Vargas Wong

GUAYAQUIL - ECUADOR

Año: 2024

DEDICATORIA

DANIEL VARGAS

Dedicado a mi mamá Jaqueline Wong, mi papá Julio Vargas, mi abuela Elena Nan, mis hermanos, Julio, Jaime, Marjorie y Verónica, pero mas aún, dedicada a mi hermana Priscila Palacios, por su apoyo incondicional y su persistencia y sus palabras cada vez que pasaba por momentos complicados en mi vida, también dedicada a mis sobrinos Natalia, Pablo, Samantha, Damián, Jaime, Julio, Mía, Mishelle y Anthony, por ser lo mas bello que me ha dado el mundo.

DEDICATORIA

GUSTAVO CASTILLO

Le dedico a Dios, fuente inagotable de amor y sabiduría que incondicionalmente me acompaña. A mis padres, Gustavo Castillo y Angela Cusme, cuyo esfuerzo me permitió iniciar en este viaje académico. A mi querida tía Carmen Cusme, que en su bondad y nobleza me supo apoyar en el proceso. A mi mentora Carmen Peña, a quien admiro por su dedicación y persistencia. Su guía ha sido un pilar fundamental que ha impulsado la culminación exitosa de esta etapa académica. Este logro está dedicado con profundo cariño y agradecimiento a cada uno de ustedes, quienes han sido mi soporte y luz en este significativo trayecto.

AGRADECIMIENTOS

DANIEL VARGAS

Agradecido enteramente con Dios, sin Él nada de esto sería posible, a mis padres por darme la mejor herramienta para la vida, la educación, a mi abuela, por ser parte fundamental de mi vida, a mis hermanos, por siempre apoyarme en todos mis momentos, también agradezco a mis amigos de la universidad Andrés Acosta, Lily Rodríguez, Arnold Terreros, Azael Ubillus, Henry Pérez, Bryan Machuca, Génesis Encalada, Saul Carvallo, Emilio Abril, Santiago Mogrovejo, Hamilton Baque, Jean Ramírez, Miguel Matute y mis amigos fuera de esta como lo son David Gavilánez, María Arce, Javier Quimis y Ericka Mena, también a mis docentes, María Ramírez e Ignacio Marín, y demás, por impartirme su enseñanza y brindarme lo mejor que pudieron durante mi vida universitaria.

GUSTAVO CASTILLO

A Dios, por abrirme las puertas que me permitieron cruzar esta etapa académica. A la Escuela Superior Politécnica del Litoral, por brindarme la oportunidad de adquirir conocimientos, desafiar mis límites y crecer como profesional. Agradezco a cada miembro de esta institución por su contribución a mi formación. A mi familia, A mi compañero Lincoln Sánchez y su familia, quienes han compartido este trayecto académico con generosidad y amistad. Finalmente, a mí mismo, por el esfuerzo constante, la perseverancia y el compromiso dedicado a alcanzar esta meta. A todos ustedes, gracias por ser parte esencial de este capítulo en mi vida. Su influencia ha dejado una marca indeleble en mi camino hacia el crecimiento y el éxito académico.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Daniel Alejandro Vargas Wong y Gustavo Gerardo Castillo Cusme damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual ”

Daniel Alejandro Vargas Wong

Gustavo Gerardo Castillo Cusme

EVALUADORES

Firmado electrónicamente por:
MARIA JOSE RAMIREZ PRADO

María José Ramírez Prado

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
IGNACIO MARIN GARCIA

Ignacio Marín García

PROFESOR TUTOR

RESUMEN

En el ámbito de las comunicaciones inalámbricas, la Comunicación por Luz Visible (VLC) y la Comunicación Óptica por Cámara (OCC) han ido ganando terreno gradualmente. Estas tecnologías se fundamentan en la utilización de la luz emitida por dispositivos electrónicos para establecer comunicación entre ellos, realizando una decodificación de señales lumínicas recibidas por el dispositivo móvil. Con el propósito de proporcionar una comprensión más profunda sobre el funcionamiento de las comunicaciones VLC y OCC, se ha desarrollado una aplicación exclusiva para dispositivos Android. Esta aplicación permite a los usuarios recibir y decodificar mensajes transmitidos a través de un transmisor VLC, demostrando así el proceso de cómo trabajan estas tecnologías. El enfoque principal de esta iniciativa es facilitar a los estudiantes, particularmente de la Escuela Superior Politécnica del Litoral, una comprensión más clara de cómo operan estas formas de comunicación. Se espera que este proyecto sirva como un punto de entrada para que los estudiantes se sumerjan más profundamente en el estudio de estas innovadoras tecnologías de comunicación. En conclusión, la aplicación no solo satisface los objetivos de diseño y decodificación, sino que también destaca por su potencial impacto en la educación. La importancia de OCC como herramienta tecnológica se manifiesta en su capacidad para transmitir información de manera novedosa y eficaz.

Palabras Clave: Comunicación por Luz Visible (VLC), Comunicación por Cámara Óptica (OCC), Decodificación de señales lumínicas.

ABSTRACT

In the field of wireless communications, Visible Light Communication (VLC) and Optical Camera Communication (OCC) have been gradually gaining ground. These technologies are based on using light emitted by electronic devices to establish communication between them, involving the decoding of light signals received by the mobile device. With the purpose of providing a deeper understanding of how VLC and OCC communications function, an exclusive application for Android devices has been developed. This application allows users to receive and decode messages transmitted through a VLC transmitter, thus demonstrating the process of these technologies. The main focus of this initiative is to facilitate clearer comprehension for students, particularly those from the Escuela Superior Politécnica del Litoral, regarding how these forms of communication operate. It is expected that this project will serve as a gateway for students to delve more deeply into the study of these innovative communication technologies. In conclusion, the application not only meets the design and decoding objectives but also stands out for its potential impact on education. The significance of OCC as a technological tool is evident in its ability to convey information in a novel and effective manner.

Keywords: Optical Camera Communication (OCC), Visible Light Communications (VLC), Luminescent signal decoding

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
OCC	Optical cammera communications / comunicación óptica por cámara
VLC	Visible light communications / Comunicación por luz visible
LED	Light emitting diode / Diodo emiso de luz
OWC	Optical Wireless Communication / Comunicación Optica Inalambrica
IoT	Internet of things / internet de las cosas
PWM	Pulse Width Modulation / Modulación por ancho de pulso
APK	Paquete de aplicación de Android
APP	Aplicación
RGB	Red, Green, Blue / Rojo, Verde, Azul

ÍNDICE DE FIGURAS

CAPÍTULO 1

1. INTRODUCCIÓN

En la era de la información y la conectividad, la tecnología móvil se ha convertido en una herramienta indispensable en la vida cotidiana de las personas. El constante avance de la tecnología móvil ha allanado el camino para la creación de aplicaciones innovadoras que no solo mejoran la eficiencia y la accesibilidad, sino que también ofrecen experiencias interactivas únicas y educativas (P. Nguyen et al., 2017). Dentro de este contexto, este proyecto se adentra en el mundo de la Comunicación Óptica por Cámara (OCC) con el objetivo de desarrollar una aplicación móvil Android de vanguardia.

La aplicación móvil propuesta aprovecha la convergencia de la óptica y la tecnología móvil, permitiendo a los usuarios recibir, decodificar y visualizar datos transmitidos a través de luz visible capturada por la cámara de sus dispositivos Android. Este enfoque innovador desafía los límites tradicionales de la comunicación y proporciona una experiencia de usuario única. Más allá de sus aplicaciones prácticas, la aplicación busca ofrecer a los usuarios una plataforma interactiva y educativa, permitiéndoles explorar y comprender la transmisión de datos a través de la luz visible.

En este proyecto, exploraremos en detalle el desarrollo, la implementación y las implicaciones de esta aplicación móvil OCC. A medida que avanzamos en nuestro análisis, se revelarán las oportunidades y desafíos inherentes a la comunicación óptica por cámara en el contexto de dispositivos móviles Android. Además, se destacará el potencial de esta tecnología en el ámbito de la educación y la comunicación, al tiempo que se examinará su viabilidad en el mundo real.

En última instancia, este proyecto tiene como objetivo contribuir al crecimiento y la comprensión de la OCC, ofreciendo una aplicación práctica y educativa que demuestra su potencial y apunta a ampliar las fronteras del conocimiento y la innovación tecnológica.

1.1 Definición de la problemática

En el ámbito educativo con enfoque en la tecnología OCC, se presenta una problemática fundamental. El principal desafío radica en la carencia de herramientas interactivas y accesibles que permitan a estudiantes y usuarios involucrarse de manera práctica con esta tecnología emergente. La comunicación a través de luz visible (VLC), al ser un concepto relativamente reciente, tiende a resultar abstracta y de difícil comprensión sin la oportunidad de una experiencia tangible (Saeed et al., 2019).

Adicionalmente, surgen dificultades en lo que concierne a la recolección y visualización de datos transmitidos mediante VLC. La información transmitida por medio de luz visible generalmente se encuentra más allá del espectro perceptible por el ojo humano y requiere el empleo de tecnologías específicas, como las cámaras de teléfonos inteligentes, para ser capturada y descifrada. La falta de una aplicación dedicada que facilite esta experiencia puede limitar de manera significativa la capacidad de los usuarios para interactuar y aprender de manera efectiva sobre VLC. La solución a este dilema radica en el desarrollo de una aplicación educativa que fomente la exploración práctica de OCC.

1.2 Justificación del problema

La justificación de este problema radica en la creciente importancia de la tecnología OCC en la educación y el aprendizaje. La OCC es una innovación tecnológica prometedora, pero su adopción se ve obstaculizada por la falta de herramientas interactivas y accesibles que permitan a estudiantes y usuarios comprenderla y experimentar con ella de manera práctica. La comunicación mediante luz visible es un concepto relativamente novedoso y, por lo tanto, abstracto para muchos, lo que dificulta su comprensión sin una experiencia tangible (Quintana Sánchez, 2013).

Además, la tecnología OCC implica la transmisión de datos a través de luz visible, lo cual está más allá del espectro perceptible por el ojo humano. Esto requiere el uso de dispositivos específicos, como cámaras de teléfonos inteligentes, para capturar y decodificar la información transmitida. La falta de una aplicación educativa dedicada limita la capacidad de los usuarios para interactuar y aprender efectivamente sobre esta

tecnología. Por lo tanto, el desarrollo de una aplicación educativa se justifica como una solución esencial para superar estos obstáculos y fomentar una comprensión más profunda y una adopción efectiva de la Comunicación por Cámara Óptica en entornos educativos.

1.3 Objetivos:

El objetivo general del proyecto propuesto es diseñar y desarrollar una aplicación móvil innovadora y funcional específicamente diseñada para dispositivos Android. Esta aplicación móvil tendrá la capacidad esencial de recibir, descifrar y presentar de manera comprensible los datos transmitidos a través de la tecnología VLC-OCC. La aplicación móvil no solo se enfocará en garantizar una recepción efectiva de datos, sino también en optimizar la experiencia del usuario y promover la adopción de la VLC-OCC, impulsando así la evolución en las comunicaciones móviles y la transmisión de datos, el proyecto se lo va a realizar mediante los siguientes objetivos específicos:

1. Implementar una interfaz que habilite a los usuarios configurar de manera precisa los parámetros de la cámara mediante la gestión de recursos del dispositivo, logrando la captura de imágenes aptas para el análisis y procesamiento.

2. Desarrollar el código fuente de la aplicación móvil mediante la integración eficiente de algoritmos de procesamiento de imágenes y un controlador de cámara Android, encapsulando de manera efectiva la recepción, procesamiento y exposición de información transmitida en el sistema OCC.

3. Optimizar el funcionamiento integral de la aplicación, enfocándose especialmente en la reducción significativa de los tiempos de respuesta logrando una visualización de datos recurrente.

1.4 Alcance y Limitaciones

El alcance de este proyecto abarca el desarrollo de una aplicación móvil exclusivamente diseñada para dispositivos Android. La aplicación tiene como finalidad la recepción, decodificación y visualización de un número entero dentro del rango del 1 al 15,

el cual es transmitido mediante un transmisor OCC ubicado en los laboratorios de telecomunicaciones de la Escuela Superior Politécnica del Litoral (ESPOL). El enfoque de este proyecto se centra en la creación y optimización de esta aplicación móvil, que permitirá a los usuarios, especialmente estudiantes, interactuar con la tecnología OCC y experimentar la transmisión de datos en un contexto controlado. El objetivo es facilitar la comprensión y el aprendizaje de esta tecnología emergente en un entorno académico.

- Enlazando con el **primer objetivo específico**, diseñar una interfaz de usuario para la aplicación móvil que permita a los usuarios visualizar los datos transmitidos a través del transmisor OCC.

  Este proyecto consiste en desarrollar una interfaz práctica, sencilla y accesible exclusivamente para usuarios de dispositivos Android. La finalidad de esta interfaz será proporcionar opciones claras e intuitivas para la utilización de la cámara del dispositivo móvil o la revisión de un video previamente almacenado en el mismo.

  Además, se busca optimizar la experiencia del usuario al reducir la cantidad de información presentada en la interfaz, facilitando así el acceso a la aplicación y permitiendo su utilización por una mayor variedad de usuarios. Esta simplificación de la información contribuirá a que la aplicación sea más amigable y fácil de usar.

  Cabe destacar que el diseño de la interfaz de usuario se enfrentará al desafío de ser compatible con una amplia gama de dispositivos Android. Las diferencias en tamaños de pantalla y capacidades de hardware entre estos dispositivos requieren una atención especial durante el proceso de diseño, con el objetivo de asegurar un rendimiento óptimo y una experiencia consistente para todos los usuarios.

- Conectando con el **segundo objetivo específico**, que busca integrar algoritmos de captura y decodificación de señales OCC.

  Se busca la incorporación de algoritmos de decodificación específicamente diseñados para procesar las señales transmitidas, abarcando números del 1 al 9. Esta implementación tiene como finalidad mostrar directamente los resultados de la decodificación en la interfaz de la aplicación, mejorando así la experiencia del usuario.

  La decisión de integrar algoritmos existentes responde a la necesidad de optimizar el proceso de desarrollo. Al aprovechar soluciones ya establecidas, se reduce

significativamente el tiempo requerido para implementar estas funciones, lo que contribuye a un desarrollo más eficiente y rápido de la aplicación en su conjunto.

Es importante tener en cuenta que, en el curso de esta integración, podrían surgir consideraciones legales relacionadas con las licencias de los algoritmos preestablecidos. Algunos de estos algoritmos pueden estar sujetos a restricciones que limitan su uso o distribución en aplicaciones comerciales o proyectos específicos. En consecuencia, se llevará a cabo una evaluación cuidadosa y, en caso necesario, se buscarán alternativas viables para cumplir con las restricciones de licencia y garantizar la legalidad y sostenibilidad del proyecto.

- El **tercer objetivo específico** busca realizar pruebas de funcionamiento para evaluar la experiencia de usuario y realizar mejoras en la aplicación.

  Se llevará a cabo una evaluación exhaustiva de todos los elementos que componen la aplicación. Este proceso busca realzar la experiencia del usuario y perfeccionar cada aspecto de la aplicación para asegurar su funcionalidad óptima y satisfacción del usuario. Desde el diseño de la interfaz hasta la eficiencia del mismo; se realizará un análisis detallado para identificar áreas de mejora y realizar ajustes significativos.

  Es importante destacar que, como parte de la estrategia de implementación, se ha decidido no poner la aplicación en la plataforma Play Store. Esta elección se fundamenta en la consideración de costos sustanciales asociados con el desarrollo y la distribución de aplicaciones a través de dicha plataforma. Se explorarán alternativas para la distribución de la aplicación que sean más económicas y eficientes, sin comprometer la accesibilidad y utilidad para los usuarios finales. Esta decisión también estará respaldada por un análisis exhaustivo de las necesidades y preferencias de los usuarios potenciales.

1.5 Marco Teórico:

El presente trabajo tiene como objetivo proporcionar una base sólida de conocimientos y conceptos teóricos que respalden el desarrollo de una aplicación móvil para la recepción de datos mediante un sistema OCC para recibir, decodificar y mostrar datos transmitidos por un emisor LED. A continuación, se proporciona una descripción concisa de cada una

de estas tecnologías, estableciendo así el fundamento teórico necesario para comprender su aplicación en el contexto de este proyecto.

Una **aplicación móvil**, es un pequeño programa de software diseñado principalmente para su uso en dispositivos móviles como teléfonos inteligentes. Estas aplicaciones móviles presentan características específicas relacionadas con requisitos relacionados con hardware, software y conectividad de red. La creación de una aplicación móvil efectiva, considerando la experiencia del usuario, depende de las particularidades y características de la propia aplicación móvil. (Patidar and Suman, 2021) El incremento de la popularidad de las aplicaciones móviles se atribuye a las ventajas de las tecnologías móviles. Asimismo, la diversificación de dispositivos, como tabletas y otros dispositivos móviles con distintas formas, ha ampliado la variedad de aplicaciones disponibles. Estos desarrollos han generado una creciente demanda para que los desarrolladores de aplicaciones comprendan los patrones de uso de las aplicaciones móviles. No obstante, las investigaciones previas en este ámbito se han centrado principalmente en dispositivos instrumentados, lo que ha limitado su alcance. (Li et al., 2017)

La **Comunicación Óptica por Ondas (OWC)**, emerge como una tecnología que permite la transmisión de datos utilizando el aire como su medio de transmisión principal. Esta tecnología abarca varias subcategorías, siendo particularmente destacadas la Comunicación por Luz Visible (VLC) y OCC. La última de estas, OCC, se considera una variante específica de VLC y ha ganado importancia en el ámbito de la transmisión de datos a través de la luz visible. (Sagotra, 2018)

Comunicación VLC es una alternativa en constante crecimiento en el ámbito de tecnología inalámbrica de próxima generación. Ofrece ventajas como costos reducidos, el uso de un ancho de banda no regulado y un soporte omnipresente en términos de infraestructuras. Esta tecnología tiene un amplio espectro de aplicaciones tanto en entornos interiores como exteriores, y se basa en el empleo de diodos emisores de luz (LED). Lo destacado de estos sistemas radica en su capacidad para utilizar las infraestructuras de iluminación preexistentes como medios de transmisión de datos. Esta sinergia entre la iluminación y la transmisión de datos maximiza la eficiencia de los recursos disponibles, garantiza la seguridad de la información y acelera significativamente la velocidad de transmisión. La ventaja que aprovecha es usar las instalaciones de luces LED existentes es que se puede implementar una red de comunicación sin necesidad de

crear una infraestructura adicional. La luz visible representa la forma en que el cerebro humano interpreta la radiación electromagnética en un rango específico de longitudes de onda. El espectro de luz visible abarca longitudes de onda que van desde los 380 nm hasta los 750 nm. (Chowdhury, 2013)

La **comunicación OCC** surge de manera directa de la tecnología VLC, ya que al igual que esta última, aprovecha la luz visible como medio para transmitir datos a través del aire. Sin embargo, se distingue de otras tecnologías inalámbricas en su enfoque receptor, ya que emplea la cámara de un dispositivo electrónico para demodular y decodificar las señales emitidas por el transmisor. La capacidad de recibir información de manera simultánea representa una de las ventajas clave de estos sistemas, permitiendo que múltiples usuarios visualicen los datos transmitidos utilizando aplicaciones específicas en sus dispositivos. Una estrategia para mejorar la velocidad de transmisión de datos es definir los estados posibles en los LEDs. En el caso de los LEDs RGB, existe la posibilidad de asignar un mayor número de estados y definir múltiples canales de transmisión, lo que implica configurar una frecuencia en la cual el encendido y apagado de estos LEDs sea prácticamente imperceptible. Además, en estos sistemas, es viable implementar más de una lámpara para optimizar aún más la transmisión de datos. (CristoJV., 2020)

CAPÍTULO 2

2. Estado del arte y metodología

En el ámbito de tecnologías de comunicación por luz, la velocidad del cambio es notable, impulsada por avances tecnológicos y la creciente demanda de soluciones innovadoras. Esta revisión tiene como objetivo sumergirse en el tejido de este campo dinámico, explorando las investigaciones y desarrollos más recientes que han definido su trayectoria. Este estado del arte ofrece una visión integral de los logros alcanzados y los desafíos que aún persisten.

2.1 Estado del arte

Un hito fundamental en la historia de las telecomunicaciones inalámbricas fue el desarrollo del fotófono por Alexander Graham Bell en 1880. Este invento, que utilizaba un haz de luz solar modulado por un espejo vibratorio para transmitir sonido, marcó el inicio de lo que eventualmente sería conocido como Comunicación por Luz Visible (VLC, por sus siglas en inglés). Este avance temprano en las comunicaciones ópticas inalámbricas sentó las bases para futuros desarrollos en este campo. (Boubezari, 2018).

Posteriormente, a finales del siglo XIX, específicamente en 1895, Guglielmo Marconi, un ingeniero italiano, y Alexander Stepanovich Popov, un físico ruso, dieron un paso significativo en el desarrollo de las telecomunicaciones inalámbricas al realizar la primera transmisión telegráfica inalámbrica. Este logro, que atravesó la barrera de la atmósfera terrestre (Boubezari, 2018), condujo al desarrollo de las comunicaciones inalámbricas de corto alcance, también conocidas como “Short-range Communication”. Estas comunicaciones, caracterizadas por su limitado rango de transmisión, abarcan tecnologías notables como WiFi, Bluetooth y la comunicación por infrarrojos. Dentro de este espectro, las comunicaciones ópticas de corto alcance, especialmente las que

utilizan infrarrojos, representan una porción significativa de esta categoría.

Tras el innovador inicio de las comunicaciones ópticas inalámbricas en el siglo XIX, estas tecnologías han seguido evolucionando, adaptándose y encontrando aplicaciones prácticas en varios ámbitos. Un claro ejemplo de esto es el uso de tecnologías de comunicación óptica en aplicaciones militares desde el siglo XVII, donde los telégrafos ópticos representaron un avance significativo en las tácticas de comunicación. (Lorenzo Grandes, 2016)

En el contexto doméstico, un hito notable fue el desarrollo del control remoto de televisión en 1955. Este avance no solo popularizó las Comunicaciones Ópticas Inalámbricas (OWC, por sus siglas en inglés) en el uso diario, sino que también demostró la viabilidad y eficacia de las comunicaciones ópticas para aplicaciones cotidianas. Este evento marcó un precedente importante para futuras innovaciones en el campo de las OWC, facilitando el camino para el desarrollo de tecnologías más avanzadas como la Comunicación VLC (Gavrincea et al., 2014). El salto de las aplicaciones militares y domésticas a soluciones más sofisticadas como VLC y OCC ilustra la evolución y adaptabilidad de las tecnologías de comunicación óptica. Estas tecnologías más recientes, especialmente VLC, han logrado superar muchas limitaciones de los métodos anteriores, ofreciendo mayor eficiencia, seguridad y capacidad para transmitir datos a velocidades más altas. La integración de estas tecnologías en dispositivos de uso diario, como los smartphones, ha abierto un nuevo panorama de posibilidades en el campo de la comunicación inalámbrica. (Herrera Luque, 2017)

Durante el último siglo, las comunicaciones ópticas inalámbricas han experimentado una transformación significativa. Aunque inicialmente se caracterizaron por velocidades de transmisión de datos relativamente bajas, como se observa en los controles remotos para electrodomésticos, la incorporación de tecnologías avanzadas ha marcado un cambio drástico en su eficacia y aplicación. (T. Nguyen et al., 2018)

Un punto de inflexión en este desarrollo fue la integración de Diodos Emisores de Luz (LED) en VLC en 2003 en Japón (Unión internacional telecomunicaciones, 2018). Esta innovación no solo representó un avance considerable en la capacidad de transmisión de datos de VLC, sino que también ofreció ventajas significativas sobre las tecnologías de radiofrecuencia. Entre estas ventajas se encuentran una mayor resistencia a las interferencias electromagnéticas y una seguridad reforzada contra el

acceso no autorizado a los datos transmitidos. (Torres, 2016)

Esta evolución de VLC a través de la incorporación de LEDs mejoró notablemente el rendimiento de las comunicaciones ópticas inalámbricas. Los LEDs, al ser más eficientes y tener una mayor capacidad de modulación que las fuentes de luz anteriores, permitieron aumentar las velocidades de transmisión de datos y expandir el alcance de las aplicaciones de VLC. Esto ha abierto un abanico de posibilidades para su uso en diferentes entornos, desde sistemas de comunicación domésticos hasta complejas redes de datos en entornos corporativos y urbanos. (Haas et al., 2020)

La Comunicación VLC ha sido una de las mayores innovaciones en el campo de las comunicaciones ópticas inalámbricas. Sin embargo, una evolución aún más notable en esta área es OCC. A diferencia de los sistemas VLC tradicionales, que dependen exclusivamente de fotodiodos (PD) para la recepción de datos, los sistemas OCC aprovechan las cámaras integradas en dispositivos como los smartphones para realizar tanto la comunicación como la captura de imágenes o vídeo (T. Nguyen et al., 2018. Esta dualidad de funciones representa una mejora significativa en términos de versatilidad y utilidad. (Duque et al., 2018)

OCC no solo amplía las capacidades de VLC, sino que también introduce una nueva dimensión en la interacción entre dispositivos. Al utilizar cámaras de smartphones, OCC permite una forma de comunicación más integrada y accesible, aprovechando la ubicuidad y la sofisticación tecnológica de estos dispositivos. Además, la naturaleza de la comunicación mediante OCC ofrece ventajas adicionales, como la capacidad de funcionar en entornos con iluminación visible y la posibilidad de implementar comunicaciones direccionales y seguras. (Morales Marquez, 2021)

Esta sinergia entre VLC y OCC ha resultado en un sistema de comunicación óptica inalámbrica más robusto y adaptable, capaz de satisfacer las crecientes demandas de la era digital. La integración exitosa de OCC en dispositivos cotidianos no solo representa un avance tecnológico, sino que también señala un cambio en la forma en que concebimos la interacción entre dispositivos en el ámbito de la comunicación inalámbrica. (D. T. Nguyen et al., 2019)

El avance de los Diodos Emisores de Luz (LED) ha jugado un papel crucial en el progreso de la tecnología de Comunicación VLC. Los LED no solo han mejorado la eficiencia energética y la potencia lumínica, sino que también han extendido la vida útil

de los dispositivos de comunicación a un costo reducido, haciendo de la tecnología VLC una opción cada vez más viable y accesible. (Haas et al., 2020)

Esta mejora en la tecnología de los LED ha tenido un impacto significativo en el campo del Internet de las Cosas (IoT). En contraste con las tecnologías de radio tradicionales, que necesitan un chipset o antena adicionales para facilitar la comunicación, la comunicación LED-cámara (teléfono móvil) en el contexto de VLC y OCC representa una ventaja sustancial. Debido a que los smartphones modernos ya integran tanto LEDs como cámaras, se simplifica enormemente el proceso de convertir un dispositivo común en un nodo de comunicación dentro de una red de IoT, minimizando así los costos adicionales y la complejidad técnica. (Duque et al., 2018)

La importancia de esta integración se ve magnificada en el creciente ecosistema de IoT, donde la capacidad de conectar dispositivos de manera eficiente y segura es primordial. Los dispositivos LED, al ser utilizados en aplicaciones de VLC y OCC, no solo facilitan la comunicación entre dispositivos, sino que también abren nuevas vías para aplicaciones inteligentes y automatizadas en entornos domésticos, industriales y urbanos.

En conclusión, el desarrollo de las comunicaciones ópticas inalámbricas ha sido marcado por hitos significativos a lo largo de la historia. Desde el fotófono de Alexander Graham Bell en 1880 hasta las tecnologías más recientes como VLC y OCC, hemos presenciado una evolución continua, las primeras transmisiones telegráficas inalámbricas a fines del siglo XIX llevaron al surgimiento de comunicaciones de corto alcance, incluyendo WiFi, Bluetooth y la comunicación por infrarrojos. Las aplicaciones militares y el desarrollo del control remoto de televisión en 1955 contribuyeron a la popularización de las Comunicaciones OWC, la evolución de VLC, potenciada por la mejora en la tecnología de LED, llevó a aplicaciones más amplias en entornos domésticos, corporativos y urbanos. La Comunicación VLC, junto con la innovadora Comunicación OCC, ha creado un sistema inalámbrico robusto y adaptable.

La integración exitosa de OCC en dispositivos cotidianos, especialmente en smartphones, ha cambiado la forma en que concebimos la interacción entre dispositivos en la comunicación inalámbrica. Además, la combinación de VLC y OCC ha proporcionado un sistema más versátil, capaz de satisfacer las demandas de la era digital.

2.2 Metodología

En la configuración establecida, el teléfono móvil actúa como el dispositivo receptor. La cámara integrada en el dispositivo desempeña la función crítica de capturar las señales luminosas emitidas por diodos emisores de luz (LEDs) que operan en el espectro RGB. Para la decodificación efectiva de estas señales, se implementa un algoritmo avanzado desarrollado en el lenguaje de programación *Java*, que incorpora las capacidades de la biblioteca de visión por computadora *OpenCV*. Esta biblioteca es esencial, ya que proporciona un conjunto de funciones para el procesamiento avanzado de imágenes, optimizando así la interpretación de los datos visuales capturados. Posteriormente, el sistema facilita la representación visual de los datos decodificados en la interfaz de usuario del dispositivo móvil, permitiendo una interacción directa con la información procesada, todo este esquema se puede identificar en la figura 2.1, donde se explica la arquitectura del proyecto.

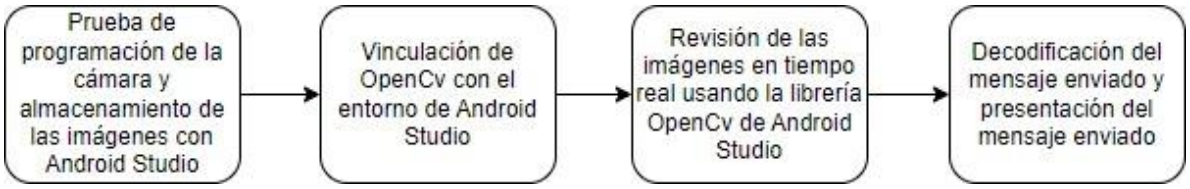

Figura 2.1: Arquitectura del proyecto

Eficiencia Computacional

El procesamiento de imágenes binarias ofrece ventajas computacionales notables frente a las imágenes a color o en escala de grises, particularmente en entornos donde los recursos son escasos, como es el caso de los dispositivos móviles. (Rosales Mora and García Capulín, 2018)

Dichas imágenes, compuestas únicamente por píxeles en blanco y negro (valores binarios 0 o 1), reducen drásticamente la complejidad de los cálculos necesarios, en contraste con las imágenes a color, que procesan múltiples canales (rojo y azul), y las imágenes en escala de grises, que manejan diversos niveles de intensidad.

La simplicidad inherente a las imágenes binarias optimiza la implementación de algoritmos y procesos computacionales, resultando en una menor demanda de recursos

como la capacidad de procesamiento y la memoria, ambos frecuentemente limitados en dispositivos móviles. Por ende, la utilización de operaciones binarias no solo acelera el rendimiento, sino que también promueve una gestión más eficaz de los recursos disponibles en dichos dispositivos. (Molina Edgar and Díaz Julia, 2018)

Análisis y Extracción de Características

Dentro del análisis de imagen y la visión por computadora, la binarización es fundamental para simplificar la representación de la información visual y mejorar la eficiencia en la extracción de características y la identificación de patrones específicos. La binarización convierte una imagen a una forma binaria donde cada píxel adopta un valor de 0 o 1, lo cual se logra a través de la implementación de un umbral que clasifica los píxeles en blanco o negro basándose en su intensidad de color.

La adopción de métodos de binarización, tales como el algoritmo de Otsu para umbralización automática, permite una segmentación efectiva de elementos de interés, facilitando tareas críticas como el reconocimiento de objetos, la segmentación de imágenes y la detección de bordes. Además, al prescindir de la variabilidad de tonos, las imágenes binarias reducen la complejidad de los datos y mejoran la velocidad de procesamiento, resultando en una manipulación de datos más eficiente.

Cabe destacar que, a pesar de sus ventajas en cuanto a simplicidad y robustez frente a ruido o variaciones de iluminación, la binarización puede conllevar la pérdida de información detallada, lo cual es un factor a considerar en el análisis preciso de imágenes. Por tanto, su aplicación debe ser evaluada cuidadosamente en contraste con otras técnicas de procesamiento de imágenes.

La binarización ha demostrado ser una técnica valiosa en una variedad de aplicaciones, desde el procesamiento de documentos hasta la visión robótica, al proporcionar una representación simplificada que facilita la identificación de estructuras y patrones clave. En definitiva, la binarización sigue siendo una herramienta esencial en el arsenal de la visión por computadora, optimizando el análisis de imagen en diversos contextos aplicativos.

Configuración y calibración de la Cámara del dispositivo Android

La calibración de la cámara en dispositivos Android es esencial para la correcta captura y diferenciación de colores, especialmente cuando se utilizan transmisores basados en luz, como los LEDs que emiten colores rojo y azul. Una configuración precisa de la cámara es fundamental para distinguir estos colores de manera individual.

Primero, es necesario otorgar a la aplicación el permiso para acceder a la cámara. Posteriormente, se puede programar una instancia de la cámara en *Java*, utilizando las *API* de Android, para ajustar la velocidad de obturación de manera que permita la adecuada segmentación de los canales de color. Un valor de obturación optimizado es crucial para evitar mezclas indeseadas de color que puedan confundir el análisis de los datos transmitidos.

El ISO es otro parámetro crítico; aumentarlo puede mejorar la claridad de las imágenes, especialmente cuando se modifica la velocidad de obturación para capturar los colores precisamente. Es fundamental considerar las especificaciones de cada dispositivo móvil, ya que las capacidades de la cámara varían, incluso entre modelos con características similares. Se debe trabajar dentro de un rango de velocidad de obturación e ISO que el dispositivo pueda soportar y elegir un valor específico que brinde la mejor calidad de imagen para el análisis.

La elección de los parámetros de configuración también debe tener en cuenta el entorno de captura. Las condiciones de iluminación fluctuantes, como la luz del día en exteriores frente a una habitación iluminada artificialmente por la noche, exigen ajustes en la configuración de ISO y velocidad de obturación para obtener resultados óptimos en el procesamiento de la imagen.

Además, la calibración debe incluir ajustes al balance de blancos para garantizar la precisión del color y la corrección de la distorsión de la lente, lo que ayuda a mantener la integridad de las formas y patrones capturados. Estos factores son críticos para la precisión del análisis de imagen posterior y deben ser calibrados cuidadosamente según el contexto de uso.

Captura de las imágenes

Una vez determinados los valores óptimos de velocidad de obturación e ISO, como se observa en la figura 2.2 se procede a la captura de imágenes utilizando la instancia activa de la cámara. A través de un método definido en el código de la aplicación, se genera un arreglo de datos que representa la imagen capturada. Este proceso se inicia al interactuar con el botón 'CAPTURE' presente en la interfaz de usuario, el arreglo resultante contiene la información completa de la imagen, la cual puede ser temporalmente almacenada en el dispositivo para su revisión y asegurarse de que la captura refleja fielmente lo visualizado en la interfaz de la aplicación.

No obstante, desde el punto de vista de la funcionalidad, no es imprescindible guardar la imagen en el almacenamiento interno. Es crucial gestionar eficientemente la memoria asignada al arreglo en *Java*, liberando los recursos después de su uso para mantener el rendimiento óptimo de la aplicación y evitar el consumo excesivo de memoria, Es importante mencionar que la calidad de la imagen capturada puede verse afectada por el formato de archivo utilizado y por las condiciones de procesamiento de la imagen post-captura. Por lo tanto, se deben considerar las estrategias de post-procesamiento que optimicen la calidad visual de las imágenes para su posterior análisis o visualización.

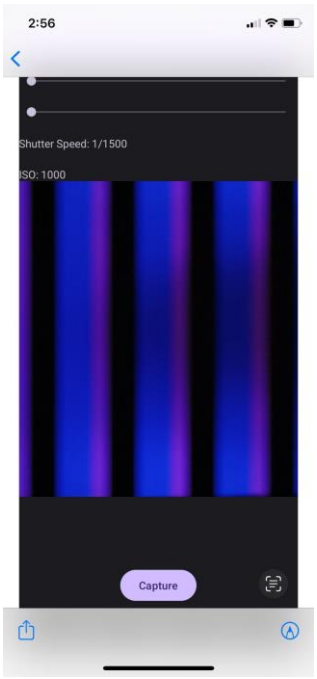

Figura 2.2: Demostración de la captura de imágenes con dispositivo móvil

Integracion de *OpenCV* con el entorno Android Studio

Aunque *OpenCV* se asocia comúnmente con el lenguaje de programación Python, su integración con Android Studio es una estrategia poderosa para el desarrollo de aplicaciones de visión por computadora en dispositivos móviles. *OpenCV* es una biblioteca avanzada que brinda más de 2500 algoritmos optimizados, esenciales para el procesamiento eficiente de imágenes y la gestión de información visual compleja, la integración efectiva de *OpenCV* en Android Studio capacita a las aplicaciones para analizar y procesar las imágenes capturadas por la cámara de manera eficaz. La plataforma Android, conocida por su extensa comunidad de desarrolladores y sus herramientas de desarrollo avanzadas, ofrece soporte excepcional para bibliotecas especializadas como *OpenCV*, elevando la calidad y las capacidades de las aplicaciones de procesamiento de imágenes.

La selección de *Java* como lenguaje de programación se justifica por su portabilidad, gestión de memoria eficiente, y capacidades multihilo, así como por la amplia comunidad de desarrolladores que lo respaldan. Estas cualidades convierten a *Java* en una opción adecuada para aplicaciones que demandan un procesamiento avanzado y la visualización de datos de imágenes, debido a la necesidad de procesar imágenes con alta eficiencia y realizar tareas de visión por computadora en tiempo real, *OpenCV* es la biblioteca elegida para el desarrollo de aplicaciones de captura de imágenes. Sus capacidades de procesamiento en tiempo real, su soporte multiplataforma y su integración sin problemas con diversas tecnologías la hacen indispensable para el desarrollo de aplicaciones complejas en Android.

Procesamiento de las imágenes en Android con *OpenCV*

El procesamiento de imágenes en Android mediante *OpenCV* implica la aplicación de técnicas de filtrado y segmentación para aislar y mejorar las áreas de interés dentro de una imagen capturada. Estos procesos son cruciales cuando la información relevante está contenida en segmentos específicos de la imagen, como ciertos rangos de color o regiones definidas espacialmente, La imagen original, representada como una matriz de píxeles, contiene toda la información visual necesaria. El objetivo inicial es descomponer esta matriz en componentes que reflejen características homogéneas para

una interpretación más detallada. Por ejemplo, la separación de la imagen en canales de color rojo y azul permite analizar cada canal de forma independiente.

Utilizando *OpenCV*, el primer paso en el algoritmo de procesamiento consiste en dividir estos canales en sus respectivas matrices. Cada una de estas matrices será sometida a una serie de transformaciones para decodificar la información que contienen. Estas operaciones no solo segregan los datos relevantes, sino que también aumentan la claridad y la precisión de los resultados obtenidos, es importante mencionar que técnicas adicionales como la umbralización adaptativa, la detección de bordes de Canny, y las operaciones morfológicas pueden ser utilizadas para refinar aún más la segmentación y el procesamiento de la imagen. Estas técnicas avanzadas, facilitadas por *OpenCV*, son esenciales para la extracción eficaz de características y para la preparación de datos para la etapa de decodificación.

Decodificación del Mensaje Enviado y Presentación del Mensaje en la Aplicación móvil

Tras separar la imagen en canales (Rojo y Azul) y convertir cada uno en matrices distintas, se procede a binarizar los canales. Mediante un método en *Java* integrado en el código de la aplicación, y utilizando funciones específicas de *OpenCV*, cada canal se binariza según un umbral predefinido, lo que simplifica la representación de la imagen a blanco y negro para facilitar el manejo de la información relevante, sin embargo, la binarización puede resultar en transiciones difusas entre el blanco y el negro, lo que dificulta una decodificación precisa. Para remediar esto, se aplica el algoritmo k-means, estructurado también como un método en la aplicación, que agrupa los píxeles por similitud, mejorando así la distinción entre segmentos o franjas.

Una vez perfeccionado el agrupamiento mediante k-means, se procede a recortar la imagen en función de los contornos identificados, permitiendo analizar y compensar las áreas de interés, debido a que cada mensaje enviado corresponde a distintos valores en el porcentaje de pixeles, es necesario que la información de los porcentajes de pixeles de interés sean compensados a un valor específico según un rango, es decir, existirán distintos rangos de porcentajes de pixeles y para cada rango se establece un valor único de salida, por ejemplo si disponemos un rango de porcentajes de 10 a 30 y el porcentaje

de pixeles objeto tiene 25 el cual está entre 10 y 30 el valor compensado se establece en 20.

Para finalmente decodificar la información se utilizo el valor de compensación para el canal rojo y azul, son dos valores de compensación necesarios a utilizar en la ecuación 2.1 Donde ValorRed representa el valor compensado del porcentaje de pixeles en canal rojo y ValorBlue representa el valor compensado del porcentaje de pixeles en el canal azul.

$$s = \frac{ValorRed}{5} + \frac{ValorBlue}{20} - 5 \tag{2.1}$$

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

Como seguimiento de las investigaciones teóricas y prácticas discutidas en el Capítulo 2, se implementaron pruebas rigurosas para validar la funcionalidad de nuestra aplicación. Este capítulo detallará las pruebas ejecutadas, describirá la metodología de evaluación empleada y ofrecerá un análisis exhaustivo de los parámetros críticos que influyen en la efectividad de la comunicación establecida por nuestra aplicación.

El propósito de esta sección es proporcionar evidencia empírica del rendimiento óptimo de nuestro sistema, con una presentación de los datos recolectados y su interpretación adecuada. Se espera que los resultados demuestren no solo la viabilidad de la aplicación sino también cómo se compara su rendimiento con los estándares actuales.

A lo largo de este análisis, se identificarán puntos fuertes y posibles limitaciones en la implementación del sistema, ofreciendo una perspectiva realista de su eficacia y áreas de mejora. Dicha información será invaluable para respaldar la confiabilidad y la calidad de nuestra solución tecnológica.

3.1 Presentación de los Resultados

La fase de desarrollo del receptor ha sido crucial, atravesando una serie de adaptaciones significativas que no se reflejaron en el transmisor, pero que fueron esenciales para alcanzar un rendimiento óptimo. A continuación, se describirá detalladamente el proceso evolutivo del receptor, resaltando las mejoras implementadas y cómo estas han contribuido a la solución final. Se presentarán y discutirán los datos de rendimiento, demostrando cómo cada ajuste y modificación ha mejorado la eficiencia y la eficacia del sistema de recepción de nuestra aplicación.

Inicialización de la cámara con *CameraX*

Durante la fase inicial de desarrollo, se consideró utilizar *OpenCV* debido a su capacidad de interactuar directamente con las cámaras. Sin embargo, se encontró que *OpenCV* no gestionaba eficientemente este recurso de hardware, creando instancias que limitaban el control directo sobre funciones específicas del hardware de la cámara. Por tanto, se decidió utilizar *CameraX* como se oberva en la figura 3.1, una *API* de Android Studio especializada en el manejo de cámaras. *CameraX* se destacó por su facilidad de uso y la conveniencia que brinda a los desarrolladores en diversas funciones. A pesar de su utilidad, se identificó un problema significativo con *CameraX* relacionado con la configuración de parámetros específicos como el ISO y la velocidad de obturación. Esta limitación tuvo un impacto negativo en la definición precisa de las imágenes y en la recepción de datos, afectando la exactitud de la información transmitida por las señales PWM. Esta dificultad resultó en interpretaciones erróneas de los ciclos de trabajo y lecturas defectuosas por parte de la cámara. Para abordar estos desafíos, se exploraron alternativas para mejorar el control sobre los parámetros de la cámara. Se investigaron enfoques que permitieran una configuración más detallada, manteniendo al mismo tiempo la estabilidad y eficiencia del sistema. Este análisis detallado y la adaptación de la tecnología fueron fundamentales para optimizar la captura de imágenes y asegurar la precisión en la interpretación de los datos.

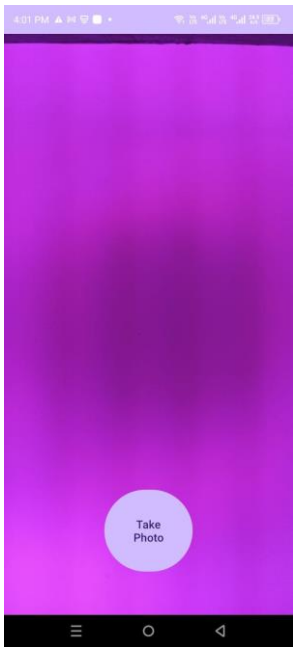

Figura 3.1: *Pruebas realizadas con CameraX*

Utilización de *Camera2 API*

Como se muestra en la figura 3.2, se optó por implementar la *API Camera2* en el entorno de desarrollo de nuestra aplicación. Esta decisión fue motivada por la necesidad de un control más refinado sobre la cámara del dispositivo, especialmente para ajustar la forma en que captura la luz emitida por el transmisor. A diferencia de otras *APIs* disponibles, *Camera2* ofrece una flexibilidad significativa en la configuración de parámetros críticos como el ISO y la velocidad de obturación. Esta capacidad es crucial para asegurar una captura de imágenes precisa y de alta calidad, que es esencial para la fase de decodificación de los mensajes transmitidos.

El uso de la *API Camera2* nos ha permitido manipular y optimizar la captura de imágenes de manera más eficaz, facilitando así el proceso de decodificación y la visualización precisa del mensaje emitido por el transmisor. Durante la implementación de esta *API*, nos enfrentamos a algunos desafíos, como la complejidad de su configuración y la necesidad de ajustar los parámetros de la cámara para diferentes condiciones de iluminación. Estos desafíos fueron abordados mediante un enfoque detallado y una configuración cuidadosa, lo que nos permitió aprovechar al máximo las capacidades avanzadas de la *Camera2 API*.

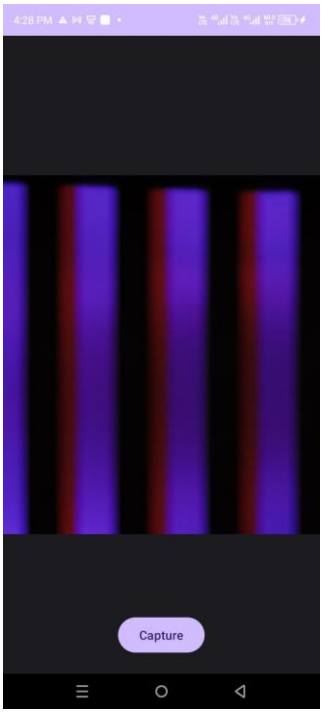

Figura 3.2: *Pruebas realizadas con Camera2 API*

Configuración Manual de la Cámara

Como se muestra en la figura 3.3, la *API Camera2* permite a los usuarios de la aplicación ajustar manualmente los parámetros de la cámara, lo que es crucial para adaptarse a las variaciones en los dispositivos, el entorno y las condiciones de iluminación. El ajuste personalizado de valores como el ISO y la velocidad de obturación es fundamental para capturar con precisión la luz emitida por el transmisor y obtener la mejor respuesta visual posible.

Esta funcionalidad de ajuste manual brinda a los usuarios el control sobre cómo desean capturar la luz, lo que resulta especialmente importante en situaciones donde las condiciones de iluminación son variables o inusuales. Por ejemplo, en entornos con luz baja, aumentar el ISO puede ser necesario para capturar detalles, mientras que en situaciones de alta luminosidad, reducir la velocidad de obturación puede evitar la sobreexposición.

Adicionalmente, sería útil proporcionar a los usuarios de la aplicación directrices o recomendaciones sobre cómo ajustar estos parámetros en diferentes escenarios. Esto podría incluir consejos sobre el equilibrio entre ISO y velocidad de obturación para diferentes niveles de luz ambiental o sugerencias para minimizar el ruido y maximizar la claridad en las imágenes capturadas.

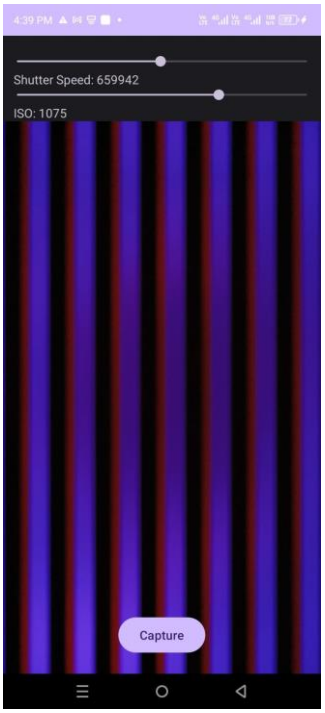

Figura 3.3: Demostración del ISO y Velocidad de obturación variable

Procesamiento de la Imagen

Tras la segmentación de los canales (Rojo y Azul) en dos matrices distintas, procedemos a la binarización de cada canal. Este proceso, desarrollado y codificado específicamente para nuestra aplicación, pretende optimizar la calidad de los datos de la imagen antes de su conversión a blanco y negro. La binarización, como se ilustra en la figura 3.4, implica transformar la imagen según un umbral predefinido, de manera que los píxeles representen blanco o negro basándose en este umbral, simplificando así la gestión de la información relevante del canal.

No obstante, este procedimiento puede generar imprecisiones, especialmente en áreas con franjas de color blanco, donde las transiciones entre negro y blanco pueden resultar difusas. Para mejorar la exactitud en la decodificación, es crucial diferenciar claramente los píxeles negros de los blancos, segregándolos en segmentos o franjas definidos. Para alcanzar este objetivo, se ha integrado un algoritmo *k-means* en el código de la aplicación. Este algoritmo agrupa píxeles similares, lo que facilita una separación más nítida y precisa de las franjas de color, mejorando la decodificación de la imagen.

Además, sería beneficioso detallar cómo el algoritmo *k-means* se ajusta y optimiza para las características específicas de las imágenes en nuestra aplicación. Esto podría incluir discusiones sobre la selección del número de clusters en *k-means*, la manipulación de umbrales para distintas condiciones de iluminación, y cómo estos ajustes afectan la calidad final de la imagen y la precisión de la decodificación.

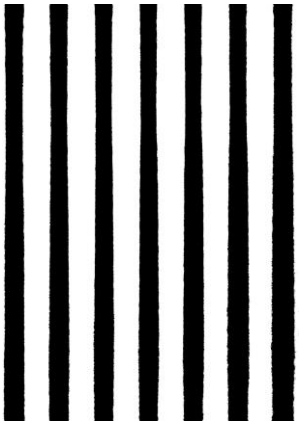

Figura 3.4: Imagen Binarizada con umbral

Una vez que se ha configurado la cámara, se procede a realizar el análisis y procesamiento de las imágenes capturadas en tiempo real. Para llevar a cabo este

proceso, se inicia separando los canales (rojo y azul) como se observa en la figura 3.5.

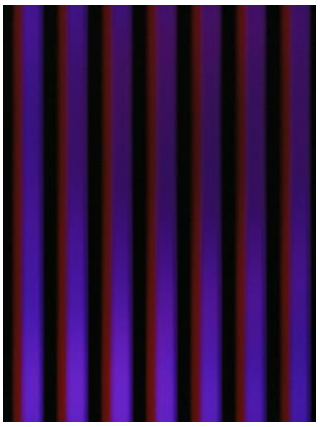

Figura 3.5: Imagen con dos canales

La implementación adecuada del algoritmo *k-means* culmina en la obtención de una imagen con una delineación más nítida y definida de las franjas en blanco y negro. Posteriormente, se efectúa un recorte meticuloso de la imagen, prestando especial atención a los contornos, específicamente a las franjas ubicadas en los bordes, como se muestra en la Figura 3.6. Este paso es crucial para una evaluación detallada de la compensación de la imagen. Finalmente, se procede a la fase de decodificación, un proceso analítico esencial que transforma la información visual en un formato interpretable y significativo como se observa en la figura 3.7, el mensaje toma un momento para procesar esta información.

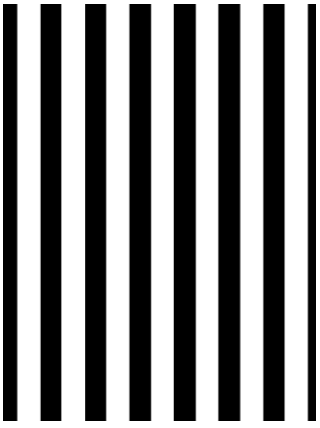

Figura 3.6: Imagen un solo canal binarizado y mejorado con *kmeans*

Figura 3.7: Procesamiento de información para luego ser presentado

Como se puede apreciar en la figura 3.8, el número decodificado se visualiza en la pantalla del teléfono móvil. La visualización se debe a que se utilizó las funciones propias para usar cámara en android, las cuales permiten presentar cualquier información a manera de notificaciones dentro de la aplicación creada.

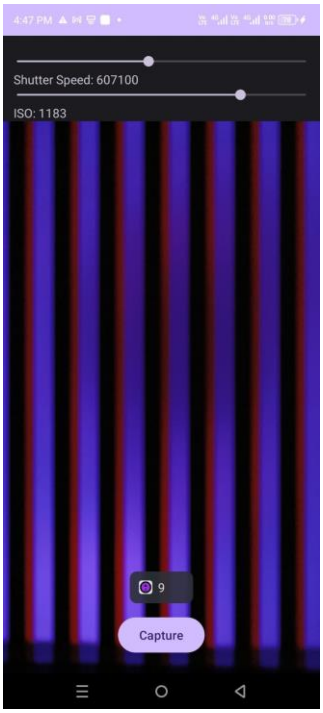

Figura 3.8:

Demostración del mensaje recibido, y decodificado con algoritmos de *OpenCV*

3.2 Análisis de Resultados

La eficiencia de nuestra aplicación se mide por la rapidez con la que procesa y muestra la información emitida por el transmisor al usuario tras la captura de la imagen. Por ello, es crucial contar con un algoritmo de procesamiento de imágenes que sea tanto conciso como eficiente. El proceso inicia con la binarización de la imagen, una técnica clave para realzar el contraste entre zonas claras y oscuras. Las zonas claras son de especial interés, ya que representan la información relevante, mientras que este proceso también ayuda a diferenciar y segmentar la información útil del ruido de fondo.

En una etapa preliminar, las imágenes binarizadas se almacenaron como archivos para evaluar la calidad del procesamiento. Aunque la binarización inicial logró una representación efectiva en blanco y negro, se observó una falta de precisión en la delimitación de los objetos blanco y negro. Esta imprecisión requería una revisión y mejora adicional del proceso. Por lo tanto, se realizó un segundo procesamiento para afinar la claridad y definición de las franjas blancas y negras, con el objetivo de alcanzar una segmentación más precisa y una calidad de imagen mejorada.

La calibración de la cámara con el ISO y la velocidad de obturación está influenciada por el entorno en donde están el transmisor y el receptor. En cada dispositivo las características del hardware varían y con ello los rangos de valores permitidos en ISO y velocidad de obturación, a modo de ejemplo el rango ISO para el dispositivo con el que se hizo las pruebas esta entre 100 ns a 16000 ns, y la velocidad de obturación entre 100000 ns a 30000000000 ns.

CAPÍTULO 4

4. Conclusiones, recomendaciones y líneas futuras

Después de haber finalizado la implementación de la aplicación para la recepción y decodificación de información a través de comunicación OCC, se llegó a las siguientes conclusiones y recomendaciones.

4.1 Conclusiones

- Gestionar los recursos del dispositivo móvil puede ser un desafío sin la herramienta correcta, Android ofrece varias que permiten trabajar con la cámara, pero la que es realmente útil es aquella que dispone de métodos que pueden hacer variar el comportamiento del sensor y la obturación, sin este paso es inaccesible filtrar la información de interés, durante las pruebas la única herramienta que permitió lograr la configuración de la cámara fue Camera 2 API, este controlador pudo establecer valores de ISO y velocidad de obturación que permitían dejar lista una imagen para que pueda ser procesada.

- Integrar tanto la instancia de la cámara configurada (ISO y velocidad de obturación) con los métodos de procesamiento de imágenes de openCV requiere la implementación de la programación orientada a objetos para que distintas clases puedan interactuar y converger en un resultado, fue así como el procesamiento de la imagen tiene su propia clase con métodos multihilo para lograr aprovechar el procesamiento del dispositivo, y la clase principal es la que gestiona la instancia de la cámara, no obstante puede haber una caída del rendimiento del dispositivo por lo mencionado anteriormente sobre el procesamiento.

- La implementación del algoritmo k-means desempeña un papel fundamental en la consecución de la precisión requerida para decodificar los datos presentados en la pantalla. Es importante destacar que este algoritmo ha sido adaptado específicamente para mejorar la calidad de los patrones de las franjas en los canales. En este sentido, fue necesario determinar la forma óptima de integrarlo en el código fuente. Aunque este proceso añade una capa adicional al tiempo de respuesta, se ha concluido que es imprescindible para garantizar respuestas correctas y, por lo tanto, se considera un compromiso necesario para alcanzar los estándares de precisión deseados.

4.2 Recomendaciones

- Para futuras investigaciones se debe ofrecer al usuario dos modos de capturar información. Uno de ellos sería manual, donde la información se captura solo cuando se presiona un botón. El segundo modo consistiría en una captura automática a intervalos de tiempo, brindando la opción de obtener información de forma periódica, simulando una lectura en tiempo real.

- Basado en que el rendimiento de la cámara puede modificar valores capturados al ser utilizada por largos periodos de tiempo, se sugiere analizar los factores que demandan consumo de procesamiento en el análisis de las imágenes con el fin de optimizar y obtener una aplicación mas ligera al utilizar la cámara del dispositivo.

- En última instancia, se sugiere llevar a cabo una gestión proactiva de los recursos del dispositivo antes de iniciar la aplicación, con el propósito de mejorar significativamente la eficiencia operativa. Este enfoque busca optimizar el rendimiento de la aplicación, reduciendo la carga sobre el dispositivo y mitigando posibles inconvenientes que podrían surgir durante su uso continuo. Se recomienda realizar acciones como cerrar aplicaciones en segundo plano, liberar memoria cache y desactivar funciones innecesarias para garantizar un entorno óptimo que permita a la aplicación ejecutarse de manera más fluida y sin interrupciones. Esta práctica contribuirá a una experiencia de usuario más satisfactoria y a un funcionamiento consistente de la aplicación a lo largo del tiempo.

4.3 Líneas Futuras

Como continuación de esta investigación, se propone una exploración más profunda y amplia en el ámbito de los algoritmos de aprendizaje automático destinados a la decodificación de información encapsulada en imágenes capturadas por cámaras ópticas. El enfoque futuro se centrará en la evaluación detallada de algoritmos específicos, considerando su aplicabilidad a diversas formas de información encapsulada y la adaptabilidad a entornos cambiantes.

Se sugiere profundizar en el análisis de la robustez de los algoritmos frente a distorsiones comunes en imágenes capturadas, como ruido, desenfoque o variaciones en la iluminación. Además, se explorará la capacidad de los algoritmos para manejar información codificada en diferentes formatos y la mejora de su rendimiento mediante la inclusión de técnicas de preprocesamiento específicas.

La investigación futura también podría abordar la optimización de algoritmos para su desempeño en dispositivos móviles o con recursos computacionales limitados, garantizando así su viabilidad práctica en aplicaciones del mundo real. Además, se propondrán evaluaciones comparativas entre diferentes enfoques de aprendizaje automático, considerando métricas específicas de rendimiento y eficiencia.

BIBLIOGRAFÍA

Boubezari, R. (2018). Smartphone to smartphone visible light communications. doctoral thesis. http://nrl.northumbria.ac.uk/id/eprint/36194

Chowdhury, R. A. (2013). Visible light communication. *International Journal of Computer Trends and Technology (IJCTT)*. https : / / citeseerx . ist . psu . edu / document ? repid = rep1 & type = pdf & doi = ca7d4a34e2efaa7f1cb0fd569baf43dec3cef080

CristoJV. (2020). Explain like i'm five: Optical camera communications (part 1). https://cristojv.com/explain-like-im-five-optical-camera-communications-part-1.html

Duque, A., Stanica, R., Rivano, H., & Desportes, A. (2018). Decoding methods in led-to-smartphone bidirectional communication for the iot.

Gavrincea, C., Baranda, J., & Henarejos, P. (2014). Rapid prototyping of standard-compliant visible light communications system. *Communications Magazine, IEEE*, *52*, 80–87. https://doi.org/10.1109/MCOM.2014.6852087

Haas, H., Elmirghani, J., & White, I. (2020). Optical wireless communication. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, *378*, 20200051. https:// doi. org/ 10. 1098/rsta.2020.0051

Herrera Luque, D. (2017). *Estudio y simulación de un sistema flip-ofdm para sistemas de comunicación por luz visible (vlc)*.

Li, X., Zheng, C., Zhang, C., Li, S., Guo, L., & Xu, J. (2017). Understanding usage behaviors of mobile apps by identifying app package in network traffic. *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, 1037–1041. https:// doi . org / 10 . 1109 / ICCSN . 2017.8230268

Lorenzo Grandes, B. (2016). *Estudio del estado del arte de los sistemas de comunicaciones por luz visible (vlc)*.

Morales Marquez, A. (2021). *Estudio experimental de un sistema de comunicación por luz visible (vlc) empleando como receptores sensores de imagen.*

Nguyen, D. T., Park, S., Chae, Y., & Park, Y. (2019). Vlc/occ hybrid optical wireless systems for versatile indoor applications. *IEEE Access*, *7*, 22371–22376. https : / / doi . org / 10 . 1109 / ACCESS . 2019 . 2898423

Nguyen, P., Le, N. T., & Jang, Y. M. (2017). Challenges issues for occ based android camera 2 api. *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 669–673. https://doi.org/10.1109/ICUFN.2017.7993875

Nguyen, T., Islam, A., Yamazato, T., & Jang, Y. M. (2018). Technical issues on ieee 802.15.7m image sensor communication standardization. *IEEE Communications Magazine*, *56*(2), 213–218. https://doi.org/10.1109/MCOM.2018.1700134

Patidar, A., & Suman, U. (2021). Towards analyzing mobile app characteristics for mobile software development. *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, 786–790.

Quintana Sánchez, C. (2013). Transmisión de datos por medio de sistemas vlc. *Vector Plus*. https://doi.org/http://hdl.handle.net/10553/11881

Rosales Mora, A. J., & García Capulín, C. H. (2018). Plataforma de desarrollo de procesamiento de imágenes en dispositivos móviles. *JÓVENES EN LA CIENCIA*, *4*(1), 2410–2414. https : / / www . jovenesenlaciencia.ugto.mx/index.php/jovenesenlaciencia/article/view/2672

Saeed, N., Guo, S., Park, K.-H., Al-Naffouri, T. Y., & Alouini, M.-S. (2019). Optical camera communications: Survey, use cases, challenges, and future trends. *Physical Communication*, *37*, 100900. https://doi.org/https://doi.org/10.1016/j.phycom.2019.100900

Sagotra, R. (2018). A comparative survey of optical wireless technologies: Architectures and applications. *IEEE Access*, *6*, 9819–9840. https://doi.org/10.1109/ACCESS.2018.2792419

Torres, J. F. R. (2016). *Contribución al estudio de servicios soportados sobre redes vlc (visible light communications)*.

Unión internacional telecomunicaciones, l. (2018). *Luz visible para las comunicaciones de banda ancha* (tech. rep. No. 2422-0). Unión internacional de telecomunicaciones. https://doi.org/https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-SM.2422-2018-PDF-S.pdf

APÉNDICES

A Anexos

Recepción del “1” visto desde la cámara del teléfono.

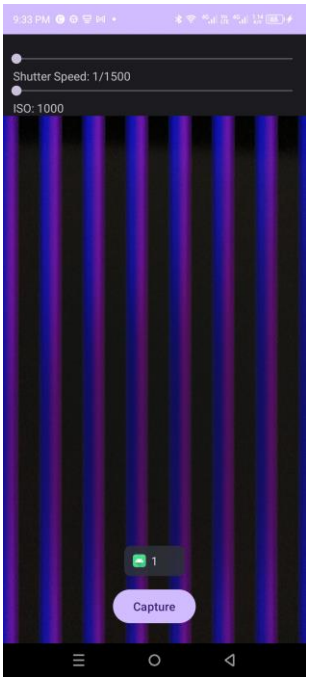

Figura 1: En la imagen se muestra la recepción del número 1

Recepción del “2” visto desde la cámara del teléfono.

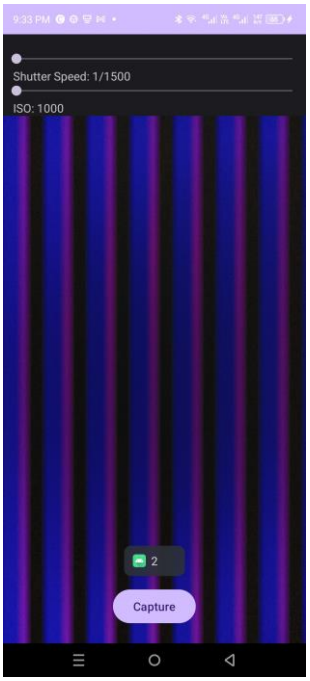

Figura 2: En la imagen se muestra la recepción del número 2

Recepción del “3” visto desde la cámara del teléfono.

Figura 3: En la imagen se muestra la recepción del número 3

Recepción del “4” visto desde la cámara del teléfono.

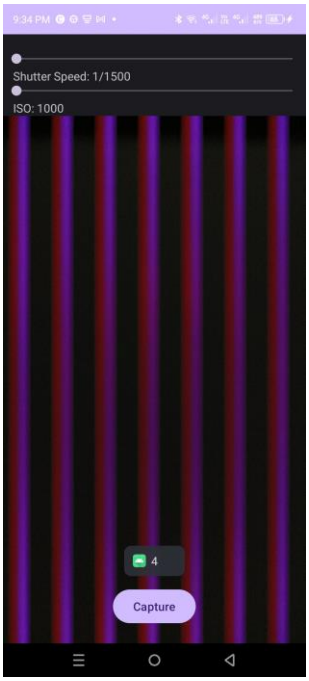

Figura 4: En la imagen se muestra la recepción del número 4

Recepción del “5” visto desde la cámara del teléfono.

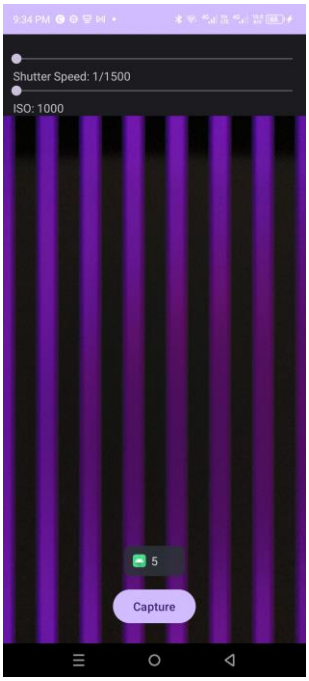

Figura 5: En la imagen se muestra la recepción del número 5

Recepción del “6” visto desde la cámara del teléfono.

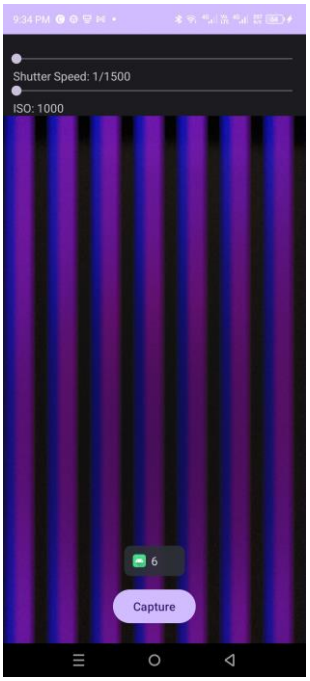

Figura 6: En la imagen se muestra la recepción del número 6

Recepción del “7” visto desde la cámara del teléfono.

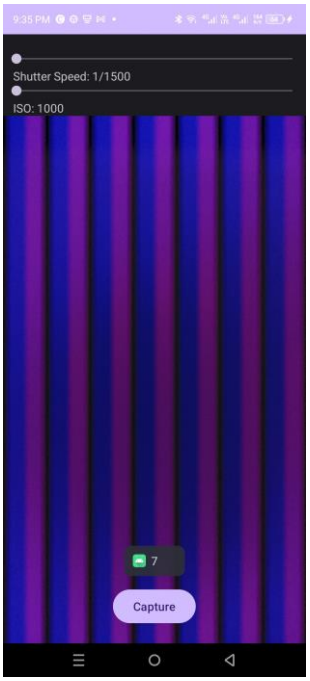

Figura 7: En la imagen se muestra la recepción del número 7

Recepción del “8” visto desde la cámara del teléfono.

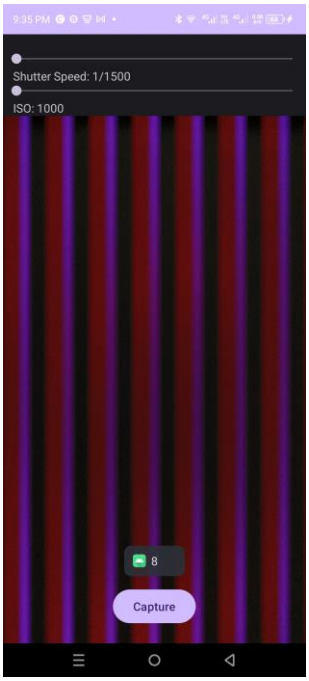

Figura 8: En la imagen se muestra la recepción del número 8

Recepción del “9” visto desde la cámara del teléfono.

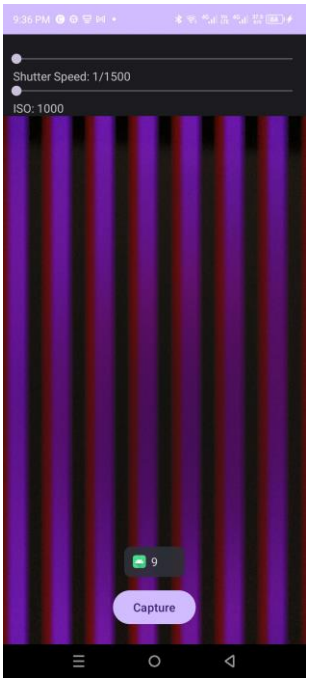

Figura 9: En la imagen se muestra la recepción del número 9

B MainActivity.java

```
package com.example.occ_opencv;


import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.ImageFormat;
import android.graphics.Matrix;
import android.graphics.RectF;
import android.graphics.SurfaceTexture;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraCaptureSession;
import android.hardware.camera2.CameraCharacteristics;
import android.hardware.camera2.CameraDevice;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.CameraMetadata;
import android.hardware.camera2.CaptureRequest;
import android.hardware.camera2.TotalCaptureResult;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.media.Image;
import android.media.ImageReader;
import android.media.MediaScannerConnection;
import android.os.Environment;
import android.os.Handler;
import android.os.HandlerThread;

import android.os.Bundle;
import android.os.Looper;
import android.os.Message;
import android.util.Log;
```

```
import android.util.Range;
import android.util.Size;
import android.util.SparseIntArray;
import android.view.Surface;
import android.view.Textureview;
import android.view.view;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.Textview;
import android.widget.Toast;

import org.opencv.android.OpenCvLoader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.Arrays;

import java.util.List;
import java.util.UUID;


public class MainActivity extends AppCompatActivity {
    private Button btnCapture;
    private Textureview textureview;
    //Check state orientation of output image
    private static final SparseIntArray ORIENTATIONS = new SparseIntArray2);
    static{
        ORIENTATIONS.append2Surface.ROTATION_0,90);
        ORIENTATIONS.append2Surface.ROTATION_90,0);
        ORIENTATIONS.append2Surface.ROTATION_180,270);
        ORIENTATIONS.append2Surface.ROTATION_270,180);
    }
    private String cameraId;
    private CameraDevice cameraDevice;
    private CameraCaptureSession cameraCaptureSessions;
```

```
private CaptureRequest.Builder captureRequestBuilder;
private Size imageDimension;
private ImageReader imageReader;
//Save to FILE
private File file;
private static final int REQUEST_CAMERA_PERMISSION = 200;
private Handler mBackgroundHandler;
private HandlerThread mBackgroundThread;
private static final int REQUEST_EXTERNAL_STORAGE_PERMISSION = 123;
private int ISO_vALUE = 1000;
private long SHUTTER_SPEED = 1000000000L / 1500;
private Range<Integer> isoRange;
private Range<Long> speedRange;

//vARIABLES DE SEEKBAR
private SeekBar shutterSpeedSeekBar, isoSeekBar;
private Textview shutterSpeedLabel, isoLabel;
//vARIABLES DE SEEKBAR
private ImageProcessor imageProcessor;
private Handler uiHandler;
private ProgressDialog progressDialog;


CameraDevice.StateCallback stateCallback = new CameraDevice.StateCallback2) {
    @Override
    public void onOpened2@NonNull CameraDevice camera) {
        cameraDevice = camera;
        createCameraPreview2);
    }

    @Override
    public void onDisconnected2@NonNull CameraDevice cameraDevice) {
        cameraDevice.close2);
    }

    @Override
    public void onError2@NonNull CameraDevice cameraDevice, int i) {
        cameraDevice.close2);
        //cameraDevice=null;
```

```
    }
};

@Override
protected void onCreate2Bundle savedInstanceState) {
    super.onCreate2savedInstanceState);
    setContentview2R.layout.activity_main);
    textureview  =  2Textureview)findviewById2R.id.textureview);

    //From Java 1.4 , you can use keyword 'assert' to check expression true or false
    assert textureview != null;
    textureview.setSurfaceTextureListener2textureListener);
    btnCapture  =  2Button)findviewById2R.id.btnCapture);

    //BARRA INICIALIzACION
    shutterSpeedSeekBar  =  findviewById2R.id.shutterSpeedSeekBar);
    isoSeekBar  =  findviewById2R.id.isoSeekBar);
    shutterSpeedLabel = findviewById2R.id.shutterSpeedLabel);
    isoLabel  =  findviewById2R.id.isoLabel);
    //BARRA INICIALIzACION


    //SET vALUES
    shutterSpeedSeekBar.setMin21OOOOO);
    shutterSpeedSeekBar.setMax21333333);
    isoSeekBar.setMin21OO);
    isoSeekBar.setMax215OO);

    btnCapture.setOnClickListener2new view.OnClickListener2) {
        @Override
        public void onClick2view view) {
            takePicture2);
        }
    });
    shutterSpeedSeekBar.setOnSeekBarChangeListener2new  SeekBar.OnSeekBarChangeListener2)  {
        @Override
        public void onProgressChanged2SeekBar seekBar, int progress, boolean fromUser) {
            // Calcula el valor de shutter speed y actualiza el texto correspondiente
```

```
        //SHUTTER_SPEED = 2long) 21OOOOOOOOOL / Math.pow22, progress / 1O.O));
        SHUTTER_SPEED=progress;
        shutterSpeedLabel.setText2"Shutter Speed: " + SHUTTER_SPEED);
        updatePreview2);
    }

    @Override
    public void onStartTrackingTouch2SeekBar seekBar) {
        // No es necesario implementar en este caso
    }

    @Override
    public void onStopTrackingTouch2SeekBar seekBar) {
        // No es necesario implementar en este caso
    }
});


isoSeekBar.setOnSeekBarChangeListener2new SeekBar.OnSeekBarChangeListener2) {
    @Override
    public void onProgressChanged2SeekBar seekBar, int progress, boolean fromUser) {
        // Actualiza el valor de ISO y el texto correspondiente
        ISO_vALUE = progress;
        isoLabel.setText2"ISO: " + ISO_vALUE);
        updatePreview2);
    }

    @Override
    public void onStartTrackingTouch2SeekBar seekBar) {
        // No es necesario implementar en este caso
    }

    @Override
    public void onStopTrackingTouch2SeekBar seekBar) {
        // No es necesario implementar en este caso
    }
});
```

```
    //MENSAJE DECODIFICADO
    uiHandler = new Handler2Looper.getMainLooper2)) {
        @Override
        public void handleMessage2Message msg) {
            // Muestra un Toast con el mensaje en medio de la pantalla
            String mensaje = 2String) msg.obj;
            Toast.makeText2getApplicationContext2), mensaje, Toast.LENGTH_SHORT).show2);
        }
    };

    // Inicializa ImageProcessor
    imageProcessor = new ImageProcessor2);









}

private void takePicture2){
    if2cameraDevice == null)
        return;
    CameraManager manager = 2CameraManager)getSystemService2Context.CAMERA_SERvICE);
    try {
        CameraCharacteristics characteristics = manager
        .getCameraCharacteristics2cameraDevice.getId2));
        Size]] jpegSizes = null;
        if2characteristics != null)
            jpegSizes = characteristics.get2CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP)
                    .getOutputSizes2ImageFormat.JPEG);

        //Capture image with custom size
        int width = 640;
        int height = 480;
        if2jpegSizes != null && jpegSizes.length > 0)
        {
            // Ordenar los tamaños en orden descendente 2mayor a menor)
```

```
    Arrays.sort2jpegSizes,  new  CompareSizesByArea2));

    // Seleccionar el tamaño más grande
    width = jpegSizes]O].getWidth2);
    height = jpegSizes]O].getHeight2);
}

imageReader = ImageReader.newInstance2width,height,ImageFormat.JPEG,1);
List‹Surface› outputSurface = new ArrayList‹›22);
outputSurface.add2imageReader.getSurface2));
outputSurface.add2new      Surface2textureview.getSurfaceTexture2)));

//calibrationvalues2characteristics);


final CaptureRequest.Builder captureBuilder = cameraDevice
.createCaptureRequest2CameraDevice.TEMPLATE_STILL_CAPTURE);
captureBuilder.addTarget2imageReader.getSurface2));
//captureBuilder.set2CaptureRequest.CONTROL_MODE,  CameraMetadata.CONTROL_MODE_AUTO);
// configure CAMERA CALIBRATION
captureBuilder.set2CaptureRequest.CONTROL_MODE, CameraMetadata.CONTROL_MODE_OFF);
captureBuilder.set2CaptureRequest.SENSOR_SENSITIvIT7,   ISO_vALUE);
captureBuilder.set2CaptureRequest.SENSOR_EXPOSURE_TIME, SHUTTER_SPEED);
captureBuilder.set2CaptureRequest.CONTROL_AE_MODE, CameraMetadata.CONTROL_AE_MODE_OFF);
//CALIDAD NEW
captureBuilder.set2CaptureRequest.JPEG_QUALIT7, 2byte) 1OO);


//CONFIGURE RESOLUTION


//Check orientation base on device
int rotation = getWindowManager2).getDefaultDisplay2).getRotation2);
captureBuilder.set2CaptureRequest.JPEG_ORIENTATION,ORIENTATIONS.get2rotation));
File picturesDirectory = Environment
.getExternalStoragePublicDirectory2Environment.DIRECTOR7_PICTURES);
file = new File2picturesDirectory, UUID.randomUUID2).toString2) + ".jpg");
ImageReader.OnImageAvailableListener readerListener = new ImageReader
.OnImageAvailableListener2) {
```

```
@Override
public void onImageAvailable2ImageReader imageReader) {
    Image image = null;
    // Mostrar el ProgressDialog antes de iniciar el procesamiento


    try{

        image = imageReader.acquireLatestImage2);
        ByteBuffer buffer = image.getPlanes2)]O].getBuffer2);
        byte]] bytes = new byte]buffer.capacity2)];
        buffer.get2bytes);

        // Continúa con el resto de tu lógica, por ejemplo, guardar la imagen procesada


        if 2OpenCvLoader.initDebug2)) {
            Log.d2"LOADTAG", "OpenCv initialized on Image");
            progressDialog = ProgressDialog.show2MainActivity.this, "Cargando informació

            // Crear una instancia de ImageProcessor si aún no existe
            if 2imageProcessor == null) {
                imageProcessor = new ImageProcessor2);
            }

            // Llamar al método processImage de ImageProcessor y pasar la imagen
            imageProcessor.processImage2bytes, progressDialog, uiHandler);




            //save2bytes);

        } else {
            Log.e2"LOADTAG", "OpenCv initialization failed");
        }


        // Guardar las imágenes procesadas
```

```
            //save2ImageAnalysis.binarizar2canalRojo,  UMBRAL_ROJO),  "rojo");
            //save2ImageAnalysis.binarizar2canalAzul,  UMBRAL_AzUL),  "azul");

            Log.d2"TAG","Esto es proceso de imagen");



            //save2bytes);
        }
        /*catch  2FileNotFoundException  e)
        {
            e.printStackTrace2);
        }
        catch 2IOException e)
        {
            e.printStackTrace2);
        }
        finally {
            {
                if2image != null)
                    image.close2);
            }
        }*/
        catch 2Exception e) {
            // Capturar y manejar excepciones generales
            Log.e2"ERROR", "Error al procesar la imagen: " + e.getMessage2));
        } finally {
            // Asegurarse de liberar la imagen en el bloque finally
            if 2image != null) {
                image.close2);
            }
        }
    }

    private void save2byte]] bytes) throws IOException {
        OutputStream outputStream = null;
        try{
            outputStream = new FileOutputStream2file);
            outputStream.write2bytes);
```

```
        }finally {
            if2outputStream != null)
                outputStream.close2);
        }
    }

    private void saveImage2Bitmap bitmap, String nombre) {
        File picturesDirectory = Environment.
        getExternalStoragePublicDirectory2Environment.DIRECTOR7_PICTURES);
        File file = new File2picturesDirectory, UUID.randomUUID2)
        .toString2) + "_" + nombre + ".jpg");
        OutputStream outputStream = null;
        try {
            outputStream = new FileOutputStream2file);
            bitmap.compress2Bitmap.CompressFormat.JPEG, 1OO, outputStream);
        } catch 2IOException e) {
            e.printStackTrace2);
        } finally {
            if 2outputStream != null) {
                try {
                    outputStream.close2);
                } catch 2IOException e) {
                    e.printStackTrace2);
                }
            }
        }
    }

};


MediaScannerConnection.scanFile2
        this,
        new String]]{file.getAbsolutePath2)},
        null,
        2path, uri) -› {
            Log.i2"TAG", "Scanned file " + path);
            // Puedes realizar acciones adicionales después del escaneo, si es necesario
        }
```

```
        );

        imageReader.setOnImageAvailableListener2readerListener,mBackgroundHandler);
        final CameraCaptureSession.CaptureCallback captureListener = new CameraCaptureSession.Capture
            @Override
            public void onCaptureCompleted2@NonNull CameraCaptureSession session, @NonNull CaptureReq
                super.onCaptureCompleted2session, request, result);
                //Toast.makeText2MainActivity.this, "Saved "+file, Toast.LENGTH_SHORT).show2);
                createCameraPreview2);
            }
        };

        cameraDevice.createCaptureSession2outputSurface, new CameraCaptureSession.StateCallback2) {
            @Override
            public void onConfigured2@NonNull CameraCaptureSession cameraCaptureSession) {
                try{
                    cameraCaptureSession.capture2captureBuilder.build2),captureListener,mBackgroundHan
                } catch 2CameraAccessException e) {
                    e.printStackTrace2);
                }
            }

            @Override
            public void onConfigureFailed2@NonNull CameraCaptureSession cameraCaptureSession) {

            }
        },mBackgroundHandler);


    }catch 2CameraAccessException e) {
        e.printStackTrace2);
    }


}
private void calibrationvalues2CameraCharacteristics characteristics){
    isoRange = characteristics.get2CameraCharacteristics.SENSOR_INFO_SENSITIvIT7_RANGE);
    speedRange = characteristics.get2CameraCharacteristics.SENSOR_INFO_EXPOSURE_TIME_RANGE);
    Log.d2"ATAG CAL:", "Rango ISO=" + isoRange);
```

```
    Log.d2"ATAG CAL:", "Rango SPEED=" + speedRange);
}

private void createCameraPreview2){
    try{
        SurfaceTexture texture = textureview.getSurfaceTexture2);
        assert texture != null;
        texture.setDefaultBufferSize2imageDimension.getWidth2),imageDimension.getHeight2));
        Surface surface = new Surface2texture);
        captureRequestBuilder = cameraDevice.createCaptureRequest2CameraDevice.TEMPLATE_PREvIEW);
        captureRequestBuilder.addTarget2surface);
        cameraDevice.createCaptureSession2
        Arrays.asList2surface), new CameraCaptureSession.StateCallback2) {
            @Override
            public void onConfigured2@NonNull CameraCaptureSession cameraCaptureSession) {
                if2cameraDevice == null)
                    return;
                cameraCaptureSessions = cameraCaptureSession;
                updatePreview2);

            }

            @Override
            public void onConfigureFailed2@NonNull CameraCaptureSession cameraCaptureSession) {
                Toast.makeText2MainActivity.this, "Changed", Toast.LENGTH_SHORT).show2);
            }
        },null);
    } catch 2CameraAccessException e) {
        e.printStackTrace2);
    }
}
private void updatePreview2) {
    if 2cameraDevice == null)
        Toast.makeText2this, "Error", Toast.LENGTH_SHORT).show2);

    captureRequestBuilder.set2CaptureRequest.CONTROL_MODE, CameraMetadata.CONTROL_MODE_OFF);
    captureRequestBuilder.set2CaptureRequest.SENSOR_SENSITIvIT7, ISO_vALUE);
    captureRequestBuilder.set2CaptureRequest.SENSOR_EXPOSURE_TIME, SHUTTER_SPEED);
    captureRequestBuilder.set2CaptureRequest.CONTROL_AE_MODE, CameraMetadata.CONTROL_AE_MODE_OFF)
```

```
    try {
        cameraCaptureSessions.setRepeatingRequest2captureRequestBuilder
        .build2), null, mBackgroundHandler);
    } catch 2CameraAccessException e) {
        e.printStackTrace2);
    }
}

private void openCamera2){
    CameraManager manager = 2CameraManager)getSystemService2Context.CAMERA_SERvICE);
    try{
        cameraId = manager.getCameraIdList2)]O];
        CameraCharacteristics characteristics = manager.getCameraCharacteristics2cameraId);
        StreamConfigurationMap map = characteristics
        .get2CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
        assert map != null;
        imageDimension = map.getOutputSizes2SurfaceTexture.class)]O];
        //Check realtime permission if run higher API 23
        if2ActivityCompat.checkSelfPermission2
        this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED)
        {
            ActivityCompat.requestPermissions2this,new String]]{
                    Manifest.permission.CAMERA,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE
            },REQUEST_CAMERA_PERMISSION);
            return;

        }
        // Configuración de la captura con nuevos valores de ISO y velocidad de obturación


        manager.openCamera2cameraId,stateCallback,null);
        calibrationvalues2characteristics);

    } catch 2CameraAccessException e) {
        e.printStackTrace2);
    }
}
```

```
Textureview.SurfaceTextureListener textureListener = new Textureview.SurfaceTextureListener2) {
    @Override
    public void onSurfaceTextureAvailable2SurfaceTexture surfaceTexture, int width, int height) {
        openCamera2);
        //configureTransform2width, height); // Llama a configureTransform aquÕ

    }

    @Override
    public void onSurfaceTextureSizeChanged2SurfaceTexture surfaceTexture, int i, int i1) {

    }

    @Override
    public boolean onSurfaceTextureDestroyed2SurfaceTexture surfaceTexture) {
        return false;
    }

    @Override
    public void onSurfaceTextureUpdated2SurfaceTexture surfaceTexture) {

    }
};

@Override
public void onRequestPermissionsResult2int requestCode, @NonNull String]] permissions,
@NonNull int]] grantResults) {
    super.onRequestPermissionsResult2requestCode, permissions, grantResults);
    if 2requestCode == REQUEST_EXTERNAL_STORAGE_PERMISSION) {
        if2requestCode == REQUEST_CAMERA_PERMISSION)
```

```
        {
            if2grantResults]O]   !=   PackageManager.PERMISSION_GRANTED)
            {
                Toast.makeText2this, "7ou can't use camera without permission",
                Toast.LENGTH_SHORT).show2);
                finish2);
            }
        }
    }
}
@Override
protected void onResume2) {
    super.onResume2);
    startBackgroundThread2);
    if2textureview.isAvailable2)) {
        openCamera2);

    }else{
        textureview.setSurfaceTextureListener2textureListener);}
}
@Override
protected void onPause2) {
    stopBackgroundThread2);
    super.onPause2);
}
private void startBackgroundThread2) {
    mBackgroundThread = new HandlerThread2"Camera Background");
    mBackgroundThread.start2);
    mBackgroundHandler = new Handler2mBackgroundThread.getLooper2));
}
private void stopBackgroundThread2) {
    mBackgroundThread.quitSafely2);
    try{
        mBackgroundThread.join2);
        mBackgroundThread= null;
        mBackgroundHandler = null;
    } catch 2InterruptedException e) {
        e.printStackTrace2);
    }
```

```
}

// ... 2código existente)

private Matrix matrix = new Matrix2); // Declaración de la matriz como variable de instancia

// Método configureTransform
private void configureTransform2int viewWidth, int viewHeight) {
    if 2imageDimension == null II textureview == null) {
        return;
    }

    int rotation = getWindowManager2).getDefaultDisplay2).getRotation2);
    float textureviewWidth = viewWidth;
    float textureviewHeight = viewHeight;

    if 2rotation == Surface.ROTATION_9O II rotation == Surface.ROTATION_27O) {
        textureviewWidth = viewHeight;
        textureviewHeight = viewWidth;
    }

    RectF viewRect = new RectF2O, O, textureviewWidth, textureviewHeight);
    RectF bufferRect = new RectF2O, O, imageDimension.getHeight2), imageDimension.getWidth2));
    float centerX = viewRect.centerX2);
    float center7 = viewRect.center72);

    bufferRect.offset2centerX - bufferRect.centerX2), center7 - bufferRect.center72));
    matrix.setRectToRect2viewRect, bufferRect, Matrix.ScaleToFit.FILL);

    float scale = Math.max2
            2float) textureviewHeight / imageDimension.getHeight2),
            2float) textureviewWidth / imageDimension.getWidth2)
    );

    matrix.postScale2scale, scale, centerX, center7);

    switch 2rotation) {
        case Surface.ROTATION_O:
            matrix.postRotate2O, centerX, center7);
```

```
                break;
            case Surface.ROTATION_90:
                matrix.postRotate290, centerX, center7);
                break;
            case Surface.ROTATION_180:
                matrix.postRotate2180, centerX, center7);
                break;
            case Surface.ROTATION_270:
                matrix.postRotate2270, centerX, center7);
                break;
        }

        textureview.setTransform2matrix);
    }


}
```

C ImageProcessor.java

```
package com.example.occ_opencv;


import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Message;
import android.util.Log;


import org.opencv.android.CameraActivity;
import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
```

```
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Rect;
import org.opencv.core.Size;
import org.opencv.core.TermCriteria;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import android.os.AsyncTask;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Date;
import java.util.List;
import java.util.UUID;

public class ImageProcessor extends CameraActivity {
    private static final String LOADTAG = "Open_LOG";

    public void onCreate2Bundle savedInstanceState) {
        super.onCreate2savedInstanceState);
        setContentview2R.layout.activity_main);


    }


    public void processImage2byte]] imageData, ProgressDialog progressDialog, Handler handler) {
        new ImageProcessingTask2progressDialog, handler).execute2imageData);
    }


    private class ImageProcessingTask extends AsyncTask‹byte]], void, String› {
        private String textoMensaje = "";
        private Handler mHandler;
```

```
private ProgressDialog progressDialog;

public ImageProcessingTask2ProgressDialog progressDialog, Handler handler) {
    this.progressDialog = progressDialog;
    this.mHandler = handler;
}

@Override
protected String doInBackground2byte]]... params) {
    byte]] imageData = params]O];

    // Convertir el array de bytes a una matriz de OpenCv
    Mat inputImage = Imgcodecs
    .imdecode2new MatOfByte2imageData), Imgcodecs.IMREAD_UNCHANGED);

    // verificar si la matriz es válida
    if 2inputImage.empty2)) {
        Log.e2LOADTAG, "Error al decodificar la imagen");
        return null;
    }

    // Dividir la matriz en canales de color 2Blue, Green, Red)
    List‹Mat› channels = new ArrayList‹›2);
    Core.split2inputImage, channels);

    // Obtener la matriz del canal azul
    Mat blueChannel = channels.get2O);

    // Obtener la matriz del canal rojo
    Mat redChannel = channels.get22);


    Mat redBinary = binarizeChannel2redChannel, 6O);
    Mat blueBinary = binarizeChannel2blueChannel, 8O);
    //saveBinaryImage2blueBinary, "imgBLUEbinary");
   //saveBinaryImage2redBinary, "imgREDbinary");

    Mat redROI = recortarROI2redBinary, 1OO);
    Mat blueROI = recortarROI2blueBinary, 1OO);
```

```
//saveBinaryImage2blueROI, "imgBlueROI");
//saveBinaryImage2redROI, "imgRedROI");

int percentageRed ;
int percentageBlue;
int simbolo;
if 2redROI != null && blueROI != null) {
    System.out.println2"Cantidad de filas en redROI: " + redROI.rows2));
    System.out.println2"Cantidad de Columnas en redROI: " + redROI.cols2));
    double]] porcentajesRojo = porcentajevaloresPixeles2redROI);
    double]] porcentajesAzul = porcentajevaloresPixeles2blueROI);

    percentageRed = compensarvalor2porcentajesRojo]1]);
    percentageBlue = compensarvalor2porcentajesAzul]1]);
    System.out.println2"%RED Obj: " + percentageRed);
    System.out.println2"%BLUE Obj: " + percentageBlue);
    simbolo = decodificarSimbolo2percentageRed, percentageBlue);
    System.out.println2"Simbolo=" + simbolo);
    // Convierte el número entero a una cadena de texto
    textoMensaje = String.valueOf2simbolo);

    //saveBinaryImage2blueROI, "imgBlueROI");
    //saveBinaryImage2redROI, "imgRedROI");
} else {
    // Manejar el caso en el que las matrices recortadas son nulas
    System.out.println2"¡Error al recortar las imágenes!");
}


runOnUiThread2new Runnable2) {
    @Override
    public void run2) {
        //redBinaryTask.executeOnExecutor2AsyncTask.THREAD_POOL_EXECUTOR);
        //blueBinaryTask.executeOnExecutor2AsyncTask.THREAD_POOL_EXECUTOR);
        //Log.d2LOADTAG, "Matriz binaria del canal azul:\n" );

        Log.d2"ImageProcessor", "Información del canal azul: " + blueChannel.toString2));
        Log.d2"ImageProcessor", "Información del canal rojo: " + redChannel.toString2));
```

```
            System.out.println2"SIMBOLO THREAD=" + textoMensaje);
            //saveBinaryImage2blueBinary,  "imgBLUEbinary");
            //saveBinaryImage2redBinary,  "imgREDbinary");


            //Log.d2LOADTAG, "Matriz binaria del canal rojo:\n" + redBinary.dump2));
            // Realiza cualquier operación en la interfaz de usuario aquÕ
        }
    });

    // Liberar la memoria de la lista de matrices de canales
    for 2Mat channel : channels) {
        channel.release2);
    }

    inputImage.release2);

    return textoMensaje;
}

@Override
protected void onPostExecute2String result) {
    super.onPostExecute2result);
    // Cierra el ProgressDialog
    if 2progressDialog != null && progressDialog.isShowing2)) {
        progressDialog.dismiss2);
    }

    // Enviar un mensaje al Handler con el texto resultante
    if 2mHandler != null) {
        Message message = mHandler.obtainMessage2O, result);
        mHandler.sendMessage2message);
    }
}

private void saveBinaryImage2Mat binaryMat, String nombre) {
    // Crea un Bitmap a partir de la matriz binaria
    Bitmap bitmap = Bitmap
```

```
        .createBitmap2binaryMat.cols2), binaryMat.rows2), Bitmap.Config.ARGB_8888);
        Utils.matToBitmap2binaryMat, bitmap);
        File picturesDirectory = Environment
        .getExternalStoragePublicDirectory2Environment.DIRECTOR7_PICTURES);
        File file = new File2picturesDirectory, UUID.randomUUID2)
        .toString2) + "_" + nombre + ".jpg");
        OutputStream outputStream = null;

        // Guarda el Bitmap como archivo de imagen utilizando FileOutputStream
        try {
            outputStream = new FileOutputStream2file);
            bitmap.compress2Bitmap.CompressFormat.JPEG, 1OO, outputStream);
            System.out.println2"Imagen binaria guardada exitosamente en: " + Environment
            .getExternalStorageDirectory2).getPath2));
        } catch 2IOException e) {
            System.err.println2"Error al guardar la imagen binaria: " + e.getMessage2));
        } finally {
            // Libera la memoria de la matriz binaria y el Bitmap
            binaryMat.release2);
            bitmap.recycle2);
        }
    }

    public Mat kMeanCluster2Mat frame) {
        Mat floatFrame = new Mat2);
        // Convierte la imagen a 32 bits de punto flotante
        frame.convertTo2floatFrame, CvType.Cv_32F);
        System.out.println2"Cantidad de filas en framekmean: " + frame.rows2));
        System.out.println2"Cantidad de Columnas en framekmean: " + frame.cols2));
        TermCriteria criteria = new TermCriteria2TermCriteria.MAX_ITER
        + TermCriteria.EPS, 1O, 1.O);
        int pixels = 2;

        Mat label = new Mat2);
        Mat center = new Mat2);
        Core.kmeans2floatFrame, pixels, label, criteria, 1O, Core.KMEANS_RANDOM_CENTERS, center);

        System.out.println2"Cantidad de filas en center: " + center.rows2));
        System.out.println2"Cantidad de Columnas en center: " + center.cols2));
```

```
        center.convertTo2center,  CvType.Cv_8U);


        System.out.println2"Cantidad de filas en label: " + label.rows2));
        System.out.println2"Cantidad de Columnas en label: " + label.cols2));
        System.out.println2"Cantidad de filas en floatFrame: " + floatFrame.rows2));
        System.out.println2"Cantidad de columnas en floatFrame: " + floatFrame.cols2));
        Mat res = new Mat2frame.rows2), frame.cols2), center.type2));
        for 2int i = O; i < frame.rows2); i++) {
            center.rowRange2O, 1).copyTo2res.rowRange2i, i + 1));
        }

        System.out.println2"Cantidad de filas en res: " + res.rows2));
        System.out.println2"Cantidad de Columnas en res: " + res.cols2));

        //imprimirMatrizEnLogcat2LOADTAG,  res,  "CENTER");

        return res;
    }


    private Mat binarizeChannel2Mat channel, int thresholdvalue) {
        // Convertir la imagen a escala de grises si es una imagen en color
        Imgproc.GaussianBlur2channel, channel, new Size25, 5), O);
        System.out.println2"Cantidad de filas en Channel: " + channel.rows2));
        System.out.println2"Cantidad de Columnas en Channel: " + channel.cols2));
        // Realizar la umbralización inicial en el canal
        Mat thresholded = new Mat2);
        Mat filteredFrame = new Mat2);
        Imgproc.threshold2channel,  thresholded,  thresholdvalue,  255,  Imgproc.THRESH_BINAR7);
        int kernelSize = 3;
        Mat kernel = Imgproc.getStructuringElement2Imgproc.MORPH_RECT
        , new Size2kernelSize, kernelSize));
        Imgproc.erode2thresholded, thresholded, kernel);
        Imgproc.dilate2thresholded, thresholded, kernel);
        System.out.println2"Cantidad de filas en thresholded: " + thresholded.rows2));
        System.out.println2"Cantidad de Columnas en thresholded: " + thresholded.cols2));
        // Realizar la umbralización automática 2Otsu) en el canal después de aplicar kMeans
        Mat binaryk = kMeanCluster2thresholded);
```

```
        Imgproc.threshold2binaryk, filteredFrame, 127, 255, Imgproc.THRESH_BINAR7
        + Imgproc.THRESH_OTSU);
        //imprimirMatrizEnLogcat2LOADTAG,binaryk,"binaryk");
        System.out.println2"Cantidad de filas en binaryk: " + binaryk.rows2));
        System.out.println2"Cantidad de Columnas en binaryk: " + binaryk.cols2));
        thresholded.release2);
        binaryk.release2);
        System.out.println2"Cantidad de filas en filteredFrame: " + filteredFrame.rows2));
        System.out.println2"Cantidad de Columnas en filteredFrame: " + filteredFrame.cols2));

        return filteredFrame;
    }

    private Mat recortarROI2Mat imagenBinaria, double areaMinima) {
        // Encontrar los lÕmites superior, inferior, izquierdo y derecho de las franjas blancas
        List‹MatOfPoint› contornos = new ArrayList‹›2);
        Imgproc.findContours2imagenBinaria, contornos, new Mat2),
        Imgproc.RETR_EXTERNAL,
        Imgproc.CHAIN_APPROX_SIMPLE);

        // Filtrar contornos pequeños 2área menor a 'areaMinima')
        List‹MatOfPoint› contornosFiltrados = new ArrayList‹›2);
        for 2MatOfPoint contorno : contornos) {
            if 2Imgproc.contourArea2contorno) ›= areaMinima) {
                contornosFiltrados.add2contorno);
            }
        }

        if 2contornosFiltrados.isEmpty2)) {
            return null;
        }

        int min_y = Integer.MAX_vALUE;
        int max_y = Integer.MIN_vALUE;
        int min_x = Integer.MAX_vALUE;
        int max_x = Integer.MIN_vALUE;

        for 2MatOfPoint contorno : contornosFiltrados) {
            Rect boundingRect = Imgproc.boundingRect2contorno);
```

```
            int x = boundingRect.x;
            int y = boundingRect.y;
            int w = boundingRect.width;
            int h = boundingRect.height;

            min_y = Math.min2min_y, y);
            max_y = Math.max2max_y, y + h);
            min_x = Math.min2min_x, x);
            max_x = Math.max2max_x, x + w);
        }

        // Ordenar los contornos de izquierda a derecha según su posición en el eje x
        contornosFiltrados.sort2Comparator.comparingInt2c -› Imgproc.boundingRect2c).x));

        // Obtener el Ōndice de inicio de la segunda franja y el Ōndice de inicio de la última franja
        if 2contornosFiltrados.size2) ‹ 3) {
            return null;
        }

        int segundoInicioX = Imgproc.boundingRect2contornosFiltrados.get21)).x;
        int ultimoInicioX = Imgproc.boundingRect2contornosFiltrados.get2contornosFiltrados.size2) -

        // Recortar la imagen original utilizando los lŌmites encontrados
        Rect roiRect = new Rect2segundoInicioX, min_y,
        ultimoInicioX - segundoInicioX, max_y - min_y);
        Mat imagenRecortada = new Mat2imagenBinaria, roiRect);

        if 2imagenRecortada.rows2) == 0 II imagenRecortada.cols2) == 0) {
            System.out.println2"imagen recortada tiene 0 filas o columnas");
            return null;
        }

        return imagenRecortada;
    }

    private double]] porcentajevaloresPixeles2Mat imagenBinaria) {
        // Contar la cantidad de pŌxeles con valor 0 2fondo) y 255 2objeto)
        //imprimirMatrizEnLogcat2LOADTAG,imagenBinaria,"%Matriz ");
        int totalPixeles = imagenBinaria.rows2) * imagenBinaria.cols2);
```

```
    int pixelesFondo = totalPixeles - Core.countNonzero2imagenBinaria);
    int pixelesObjeto = Core.countNonzero2imagenBinaria);
    System.out.println2"TotalPixeles=" + totalPixeles);
    System.out.println2"pixelesFondo=" + pixelesFondo);
    System.out.println2"pixelesObjeto=" + pixelesObjeto);

    // Calcular los porcentajes
    double porcentajeFondo = 2double) pixelesFondo / totalPixeles * 100;
    double porcentajeObjeto = 2double) pixelesObjeto / totalPixeles * 100;

    return new double]]{porcentajeFondo, porcentajeObjeto};
}

public int compensarvalor2double porcentaje) {
    int valorCompensado = -1;

    if 2porcentaje >= 10 && porcentaje < 30) {
        valorCompensado = 20;
    } else if 2porcentaje >= 30 && porcentaje < 50) {
        valorCompensado = 40;
    } else if 2porcentaje >= 50 && porcentaje < 70) {
        valorCompensado = 60;
    } else if 2porcentaje >= 70 && porcentaje < 90) {
        valorCompensado = 80;
    }

    return  valorCompensado;
}

public int decodificarSimbolo2int valorRed, int valorBlue) {
    int s = -1;

    // Compensar los valores
    valorRed = compensarvalor2valorRed);
    valorBlue = compensarvalor2valorBlue);

    // Calcular el resultado según la fórmula
    if 2valorRed > 0 && valorBlue > 0) {
        s = 2valorRed / 5) + 2valorBlue / 20) - 5;
```

```
        }

        return s;
    }


    // Método para imprimir una matriz en Logcat
    private void imprimirMatrizEnLogcat2String tag, Mat matriz, String tipo) {
        int rows = matriz.rows2);
        int cols = matriz.cols2);

        // Imprimir la matriz en Logcat
        for 2int i = O; i ‹ rows; i++) {
            StringBuilder rowStringBuilder = new StringBuilder2);
            for 2int j = O; j ‹ cols; j++) {
                // Obtener el valor del pŌxel en la posición 2i, j)
                double pixelvalue = matriz.get2i, j)]O];
                // Agregar el valor a la cadena de la fila
                rowStringBuilder.append2pixelvalue).append2" ");
            }
            // Imprimir la fila en Logcat
            //Log.e2LOADTAG, "Fila"+tipo + i + ": " + rowStringBuilder.toString2));
            System.out.println2tag + " - Fila " + tipo + i + ": "
            + rowStringBuilder.toString2));
        }
        System.out.println2tag + " - MATRIz impresa ");
    }

}

}
```

D activitymain.xml

```
‹RelativeLayout     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<SeekBar
    android:id="@+id/shutterSpeedSeekBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_marginTop="20dp"/>

<SeekBar
    android:id="@+id/isoSeekBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/shutterSpeedSeekBar"
    android:layout_marginTop="20dp"/>

<Textview
    android:id="@+id/shutterSpeedLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/shutterSpeedSeekBar"
    android:text="Shutter Speed: 1/1500"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="12dp"
    android:layout_alignTop="@id/shutterSpeedSeekBar"/>

<Textview
    android:id="@+id/isoLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/isoSeekBar"
    android:text="ISO: 1000"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="12dp"
    android:layout_alignTop="@id/isoSeekBar"/>
```

```
    ‹Textureview
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/textureview"
        android:layout_below="@id/isoLabel"/›


    ‹Textview
        android:id="@+id/messageTextview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:text=""
        android:textColor="#FFFFFF"
        android:textSize="18sp"
        android:visibility="gone"/›

    ‹Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Capture"
        android:id="@+id/btnCapture"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp"/›

‹/RelativeLayout›
```

E AndroidManifest.xml

```
‹?xml version="1.0" encoding="utf-8"?›
‹manifest    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"›
    ‹uses-feature    android:name="android.hardware.camera.any"    /›
    ‹uses-permission    android:name="android.permission.CAMERA"    /›
    ‹uses-permission    android:name="android.permission.WRITE_EXTERNAL_STORAGE"    android:maxSdkversion="28"
```

```
    <uses-permission  android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <uses-permission android:name="android.hardware.camera2.full"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.OCCOpenCv"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```