ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROPUESTA DE TRABAJO FIN DE MÁSTER

Elaboración de un algoritmo de descubrimiento de transmisores ópticos sub-píxel

Titulación: Máster Universitario en Ingeniería de Telecomunicación

Autora: Dña. Idaira Rodríguez Yánez

Tutores: Dr. José Alberto Rabadán Borges

Dr. Víctor Guerra Yánez

Fecha: Junio 2023

ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROPUESTA DE TRABAJO FIN DE MÁSTER

Elaboración de un algoritmo de descubrimiento de transmisores ópticos sub-píxel

HOJA DE EVALUACIÓN

Calificación: _____

Presidente

Fdo.:

Vocal **Secretario/a**

Fdo.: **Fdo.:**

Fecha: Julio de 2023

Resumen

En los distintos ámbitos de aplicaciones en los que se pueden encontrar las comunicaciones ópticas basadas en cámaras (OCC - *Optical Camera Communications*), los elementos transmisores no siempre se ubican a una distancia tan próxima de la cámara o tienen unas dimensiones tan elevadas, como para ocupar una proporción significativa de los píxeles de la imagen. De hecho, frecuentemente estas regiones se limitan a uno o unos pocos píxeles (condiciones sub-píxel). Para hacer frente a estas condiciones se han propuesto distintas soluciones que se pueden dividir entre aquellas que alteran la óptica de la cámara, y las que ajustan los parámetros internos de configuración de la misma. No obstante, estas alteraciones dificultan el uso de la cámara para otros fines distintos de la comunicación. En la práctica, la cantidad de trabajos que han tratado las condiciones sub-píxel sin realizar alteraciones de la cámara es reducido y se limitan a verificar la posibilidad de transmitir información en esas circunstancias.

En este trabajo se pretende desarrollar y evaluar un algoritmo de descubrimiento de transmisores ópticos sub-píxel, en escenarios *outdoor* e *indoor*, donde transmisores y receptores permanecen estáticos y sin realizar modificaciones en la cámara, lo que permitiría la utilización de las OCC en redes de sensores de Smart Cities, industrias, terrenos agrícolas, etc. El algoritmo debe ser capaz de detectar el píxel en el que se encuentran cada uno de los transmisores, descartando otras fuentes luminosas presentes en la imagen, así como presentar la información en formato binario recibida de los mismos. Para lograr esto, es necesario estudiar y usar técnicas de procesamiento de imágenes y de análisis de señales ópticas para identificar patrones en los datos de entrada de la cámara. Además, el algoritmo debe ser diseñado para poder operar con diferentes cantidades de transmisores, y adaptable a distintos escenarios ambientales.

Tras ser desarrollado y evaluado el algoritmo en diferentes escenarios de interés, se obtuvo una tasa de detección promedio del 87 % y una tasa de falsos positivos del 28 %. La omisión de positivos verdaderos se produjo en la mayor parte de los casos por la diferencia de potencia lumínica entre transmisores en un mismo escenario. Por su parte, el 97,5 % de los falsos positivos se corresponden con reflexiones de la luz de los transmisores. En cuanto a los tiempos de operación, en promedio la duración del algoritmo fue de 390 ms. Estos resultados se consideran aptos, pero mejorables, para la aplicación de esta tecnología. Para conseguirlo, diferentes soluciones se plantean como el uso de la inteligencia artificial o la alternancia periódica entre varios umbrales de luminosidad.

Abstract

In various fields of application where optical camera communications (OCC) can be found, the transmitting elements are not always located at a close distance from the camera or have dimensions large enough to occupy a significant proportion of the image pixels. In fact, these regions often encompass just one or a few pixels (sub-pixel conditions). Different solutions have been proposed to address these conditions, which can be divided between those that alter the camera optics and those that adjust internal configuration parameters of the image sensor. However, these alterations hinder the use of the camera for purposes other than communication. In practice, the number of works that have addressed sub-pixel conditions without altering the camera is limited and primarily focus on verifying the feasibility of transmitting information under such circumstances.

This work aims to develop and evaluate an algorithm for sub-pixel optical transmitter discovery in outdoor and indoor scenarios, where transmitters and receivers remain static without making modifications to the camera. This would enable the utilization of OCC in smart city sensor networks, industries, agricultural fields, etc. The algorithm should be capable of detecting the pixel location of each transmitter while discarding other light sources present in the image, as well as presenting the received information from them in binary format. To achieve this, it is necessary to study and employ image processing and optical signal analysis techniques to identify patterns in the camera's input data. Additionally, the algorithm should be designed to operate with varying numbers of transmitters and be adaptable to different environmental scenarios.

After being developed and evaluated in various relevant scenarios, the algorithm achieved an average detection rate of 87% and a false positive rate of 28%. True positive omissions mostly occurred due to differences in light power among transmitters within the same scenario. Additionally, 97.5% of false positives were attributed to light reflections from transmitters. On average, the algorithm operated for 390 ms. While these results are deemed suitable, there is room for improvement in implementing this technology. Potential solutions include employing artificial intelligence or periodic alternation of luminosity thresholds.

Agradecimientos

A lo largo de mi vida educativa me he enfrentado a una amplia variedad de retos y dificultades. No obstante, nada se me ha hecho tan difícil como escribir este documento a la par que afrontar tu partida. Por ello, este apartado de agradecimientos te lo quiero dedicar enteramente a ti, Norita Montesdeoca Ferrera, abuelita.

Hay muchos conocimientos valiosos en la vida, y los más importantes son sin duda los que tú enseñaste a todos los que tuvimos el placer de conocerte: saber estar, respeto, trabajo, generosidad, y mucho amor. Eras de esas personas que, lo poco que tenía, lo repartía. De esas que nunca tenían una mala palabra ni un mal gesto hacia nadie. De esas que siempre intentaba ayudar a quien lo necesitase sin esperar nada a cambio. De esas que llenaban la mesa de comida fuera cual fuera la visita. De esas que nunca se olvidaba de darte un abrazo, ni le pesaban los “te quiero”.

Ya no podré pasar más tardes a tus lados preguntándote por “los tiempos de antes” y descubriendo lo maravillosa que fuiste, ni tampoco te podré contar mi día a día, pero espero que fueras tan feliz como yo lo estuve a tu lado. Intentaré que sigas sintiéndote orgullosa de mí allá donde estés.

Índice general

Índice de Figuras

Índice de Tablas

Índice de Códigos

Índice de Ecuaciones

Acrónimos

ACO - OFDM - *Asymmetrically Clipped Optical - OFDM*

ADC - *Analog to Digital Converter*

APD - *Avalanche PhotoDiode*

BER - *Bit Error Rate*

CCD - *Charge-Coupled Device*

CMOS - *Complementary Metal-Oxide-Semiconductor*

COB - *Chip On Board*

CPU – *Central Processor Unit*

CSK - *Color Shift Keyin*

DCO - OFDM - *Direct Current - biased Optical* - OFDM

DFT - OFDM - *Discrete Fourier Transform - OFDM*

DIP - *Dual in-Package*

ECG - *Electrocardiogram*

EEG - *Electroencephalogram*

EMG - *Electromyography*

FPS - *Frames Per Second*

FSO - *Free Space Optic*s

GMM - *Gaussian Mixture Model*

HD - *High Definition*

IDE - *Integrated Development Environment*

IDeTIC - Instituto para el Desarrollo Tecnológico y la Innovación en Comunicaciones

IEEE - *Institute of Electrical and Electronics Engineers*

IGIC - Impuesto General Indirecto Canario

IoT - *Internet of Things*

IoV - *Internet of Vehicles*

ISI - *Inter Symbol Interference*

ISO - *International Organization for Standardization*

ITS - *Intelligent Transport System*

JPG - *Joint Photographic Experts Group*

GS - *Global Shutter*

LED - *Light Emitting Diode*

MJPEG - *Motion Joint Photographic Experts Group*

MUIT - Máster Universitario en Ingeniería de Telecomunicación

PAM - *Pulse Amplitude Modulation*

PCB - *Printed Circuit Board*

PIN - *Positive-Intrinsic-Negative*

PSF - *Potencial Spread Function*

PWM - *Pulse Width Modulation*

OCC - *Optical Camera Communications*

OFDM - *Orthogonal Frequency Division Multiplexing*

OLED - *Organic Light-Emitting Diode*

OOK - *On - Off Keying*

OpenCV - *Open Source Computer Vision Library*

OWC - *Optical Wireless Communications*

RF - RadioFrecuencia

RGB - *Red, Green, Blue*

ROI - *Region Of Interest*

RS - *Rolling Shutter*

SMD - *Surface Mounted Device*

SMPS - *Switching Mode Power Supply*

SNR - *Signal to Noise Ratio*

SSL - *Solid State Logi*

TFM - Trabajo de Fin de Máster

UFSOOK - *Undersampled Frequency Shift On-Off Keying*

ULPGC - Universidad de Las Palmas de Gran Canaria

UOCC - *Underwater Optical Camera Communications*

UPSOOK - *Undersampled Phase Shift On-Off Keying*

VLC - *Visible Light Communications*

VLP - *Visual Light Positioning*

VPPM - *Variable Pulse Position Modulation*

WSN - *Wireless Sensor Networks*

Parte I - Memoria

Capítulo 1

1. Introducción

En este capítulo de la memoria se realiza una contextualización del proyecto que envuelve el presente Trabajo de Fin de Máster (TFM). Para ello, se describen brevemente las comunicaciones ópticas basadas en cámara y sus ámbitos de aplicación, así como se explica la casuística de los transmisores sub-píxel y los trabajos desarrollados en esa línea. A continuación, se definen los objetivos y la motivación del trabajo y, finalmente, se especifica la estructura de la memoria.

1.1 Antecedentes

Los sistemas de comunicaciones ópticas basadas en cámaras (OCC - *Optical Camera Communications*) son aquellos que utilizan estos dispositivos como receptores de señales ópticas moduladas. Tienen un alto interés debido a que permiten proporcionar la capacidad de comunicación sin perder la capacidad de captura de vídeo y sin necesidad de realizar grandes cambios en el hardware de móviles, tablets, drones, sistemas de vigilancia… Sumado al bajo coste y alta durabilidad de los elementos utilizados como transmisores en estos sistemas, los diodos de emisión de luz (LED - *Light Emitting Diode*), los sistemas OCC se posicionan como un recurso emergente de comunicación y como uno de los campos de investigación de mayor proyección dentro de las comunicaciones ópticas por luz visible (VLC - *Visible Light Communications*).

El interés del ecosistema industria-academia de las OCC se manifiesta en los esfuerzos por su normalización. Esta tecnología se incluye en diferentes apartados del IEEE (*Institute of Electrical and Electronics Engineers*) 802.15.7 [1], donde se especifican niveles físicos basados en OCC y donde se contemplan las dos técnicas de adquisición de imágenes principales por parte de la cámara: los sistemas *Global Shutter* (GS) y *Rolling Shutter* (RS). Los primeros utilizan sensores *Charge-Coupled Device* (CCD) que realizan la captura simultánea de todos los píxeles. Por su parte, los sistemas Rolling Shutter (RS), basados principalmente en la tecnología de transistores *Complementary Metal-Oxide-Semiconductor* (CMOS), capturan los píxeles de forma secuencial en vez de todos a la vez, procesando la imagen fila a fila o columna a columna. Estos dos métodos de adquisición se pueden apreciar gráficamente en la Figura 1.1.1.

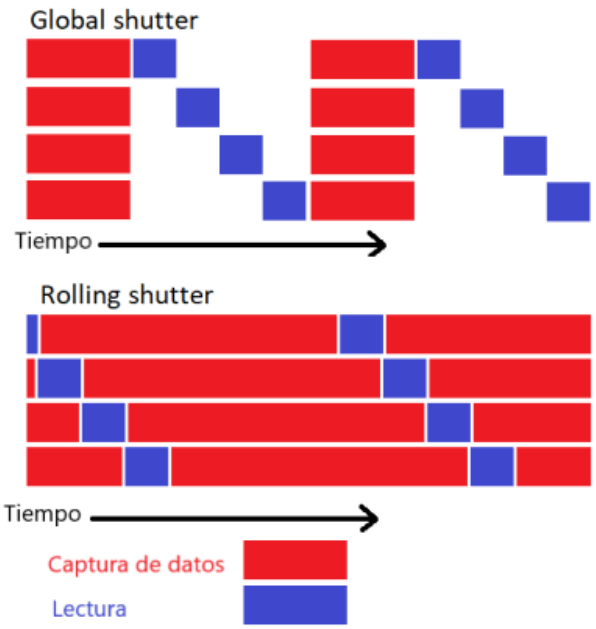

Figura 1.1.1: Diferencia de adquisición en GS y RS

El abanico de aplicaciones de las OCC es muy amplio. En el ámbito de las comunicaciones submarinas, donde las señales de radio sufren unas atenuaciones tan elevadas que dificultan su uso, donde los sistemas cableados reducen sustancialmente la distancia y maniobrabilidad de los nodos, y frente a la baja tasa de datos y la latencia de las ondas acústica en el agua, las OCC se posicionan como una opción óptima para el trabajo con vehículos operados a distancia, o para la comunicación entre vehículos marítimos autónomos, por ejemplo [2]. En el ámbito industrial y médico, donde el espectro radioeléctrico también es restringido debido a las interferencias que puede ocasionar con los equipos de trabajo o los dispositivos intracorporales de los pacientes, las OCC también permiten implementar soluciones de bajo coste y fácil instalación [3].

A diferencia de los sistemas de comunicación ópticos inalámbricos (OWC - *Optical Wireless Communications*) que utilizan como receptores fotodiodos individuales para cada transmisor, las OCC permiten la separación espacial de las fuentes de luz, como se puede observar en los trabajos [4 - 5], lo que las hace interesantes también para las redes de sensores inalámbricas (WSN - *Wireless Sensor Networks*) [6], el Internet de las cosas (IoT - *Internet of Things*) [7], el Farming 4.0 [8]... Debido a su alta precisión alcanzable, del orden de centímetros, otro ámbito de aplicación de las OCC son los sistemas de posicionamiento por luz visible (VLP - *Visual Light Positioning*) [9]. Esta aplicación a su vez se relaciona con otro campo de uso importante, los sistemas inteligentes de transporte (ITS - *Intelligent Transport System*) [10 - 12]. Las OCC pueden aumentar el rendimiento de los vehículos

autónomos mediante una mejora de las comunicaciones tanto entre ellos mismos como con la infraestructura de su entorno (semáforos, farolas, luces de otros vehículos…).

En los distintos ámbitos de aplicaciones en los que se pueden encontrar las OCC, los elementos transmisores no siempre se podrán encontrar a una distancia tan próxima de la cámara o tener unas dimensiones tan elevadas, como para ocupar una proporción significativa de los píxeles de la imagen. De hecho, lo más probable es que su proporción sea menor a unos pocos píxeles, por lo que el estudio específico de esta casuística es de gran interés. Para hacer frente a estas condiciones se han propuesto distintas soluciones que se pueden dividir entre aquellas que alteran la óptica de la cámara, y las que ajustan de forma efectiva los parámetros internos de configuración del sensor de imagen [13 - 14].

En cuanto a trabajos que no han realizado alteraciones de la cámara, en [15] se demuestra la viabilidad de un enlace GS en el que la comunicación sucede a nivel de sub-píxel gracias al hecho de que la energía emitida por el LED, debido a la distorsión introducida por el canal y por la cámara, afecta a los píxeles contiguos. En [6] se prueba nuevamente, esta vez en condiciones climáticas adversas y utilizando un enlace RS (Figura 1.1.2). No obstante, aún no se ha desarrollado ningún algoritmo que permita descubrir de forma automática los elementos transmisores en condiciones sub-píxel. En [6] o [15], por ejemplo, los transmisores se seleccionaban manualmente ampliando la imagen y clicando el píxel de interés.

Figura 1.1.2: Escenario del experimento efectuado en [6]

Tanto la tarea de descubrimiento como la del seguimiento del transmisor resultan fundamentales para seguir desarrollando esta tecnología. Sólo con el descubrimiento sería posible emplear las OCC en escenarios donde la cámara y el transmisor se mantienen estáticos, como en aplicaciones relacionadas con Smart Cities, Industria 4.0, Agricultura 4.0, etc.

1.2 Objetivos

El objetivo principal de este trabajo es el desarrollo y validación de un algoritmo de descubrimiento de transmisores ópticos sub-píxel para escenarios donde transmisores y receptores permanecen estáticos. Para ello será necesario estudiar previamente la estructura de trama con la que se va a trabajar, definir e implementar el sistema de comunicaciones, así como grabar las transferencias de datos en varios escenarios de prueba para poder evaluar posteriormente el algoritmo desarrollado. Este objetivo general se puede desglosar en las siguientes etapas de realización:

- O1: Analizar las soluciones de detección sub-píxel existentes.
- O2: Elegir y programar los transmisores y receptores del sistema de comunicación.
- O3: Implementar el sistema, y grabar y analizar las comunicaciones.
- O4: Desarrollar el algoritmo de detección sub-píxel.
- O5: Tomar medidas con el algoritmo implementado.
- O6: Analizar las medidas y valorar el algoritmo.

1.3 Motivación

La motivación principal de este Trabajo Fin de Máster es la finalización del Máster Universitario en Ingeniería de Telecomunicación (MUIT) que habilita el desempeño de la profesión de Ingeniera de Telecomunicación y permite acceder al mercado laboral con dicha acreditación oficial. Asimismo, también existe una motivación personal por aplicar los conocimientos adquiridos en distintas asignaturas del máster a un caso práctico, y por poder ampliar conocimientos en el sector de las comunicaciones ópticas inalámbricas.

Es importante destacar que estoy actualmente integrada en la División de Tecnología Fotónica y Comunicaciones del Instituto para el Desarrollo Tecnológico y la Innovación en Comunicaciones (IDeTIC), donde estoy comenzando a realizar una tesis doctoral. Este Trabajo de Fin de Máster servirá como punto de partida para los desarrollos que llevaré a cabo en dicha investigación doctoral, generando sinergias y contribuyendo al avance de este campo.

De igual modo, también toma un gran peso en la motivación, el potencial impacto comercial de los resultados de este trabajo. Las soluciones y avances propuestos podrían tener un efecto significativo en distintos ámbitos de aplicación en el futuro, lo cual añade un atractivo adicional a la realización de este proyecto.

1.4 Metodología

Para el desarrollo de este trabajo, en primer lugar, se estudiarán en profundidad los sistemas OCC, así como los posibles algoritmos de descubrimiento relacionados que puedan servir de utilidad. Este análisis permitirá tener un mayor entendimiento sobre las comunicaciones ópticas e identificar aquellas funciones o técnicas que sean más favorables para el propósito del proyecto.

Luego, en base a los conocimientos adquiridos, se diseñará e implementará un sistema OCC en distintos escenarios de interés. Las comunicaciones se grabarán en cada caso y se estudiarán para obtener conclusiones de cara al desarrollo del algoritmo. Durante el desarrollo del mismo se probará y estudiará el rendimiento de las posibles funciones y bibliotecas a utilizar para reducir los recursos consumidos por el programa.

Una vez desarrollado el algoritmo, se pondrá en funcionamiento utilizando los vídeos anteriormente grabados. Utilizando código adicional dedicado a la validación, se comprobará su correcta operación y se evaluará su rendimiento, así como la calidad de las comunicaciones en términos de relación señal a ruido (SNR - *Signal to Noise Ratio*) y tasa de error de bit (BER - *Bit Error Rate*).

Esta metodología, que se podría resumir en cuatro tipos de tareas (análisis, diseño, implementación y validación), se puede entender de forma gráfica mediante la Figura 1.4.1.

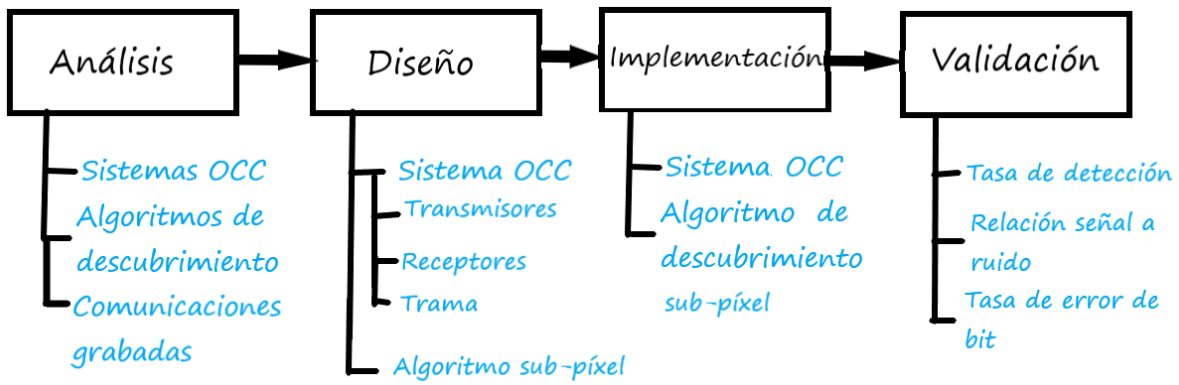

Figura 1.4.1: Metodología del TFM

1.5 Estructura de la memoria

Este documento está formado por una memoria descriptiva, una bibliografía, el presupuesto y la sección de anexos. La memoria sigue una estructura por capítulos, y en ella se hace referencia al presupuesto y a la sección de anexos. Esta última contiene aquella información que puede alejar al lector del contenido principal del TFM: algunos códigos secundarios y su explicación, códigos de gran longitud y las hojas Excel rellenadas durante la fase de experimentación.

En este primer capítulo del documento se contextualiza el marco de trabajo del TFM, así como se presentan los objetivos del mismo. En el segundo, se realiza una descripción detallada del funcionamiento y aplicaciones de las OCC, y de la casuística sub-píxel. En el tercero, se describe el sistema OCC diseñado y los distintos entornos y circunstancias en los que se ha implementado. En el cuarto, se explica la estructura del algoritmo elaborado, así como el código que lo compone, y los conocimientos en los que se fundamenta. En el quinto, se resumen las distintas estrategias y métricas utilizadas para probar la validez del algoritmo y evaluar la calidad de las comunicaciones, así como los principales resultados obtenidos. En el sexto, para finalizar, se realiza una valoración del trabajo realizado y se analizan las conclusiones obtenidas del mismo.

Capítulo 2

2. Los sistemas OCC

En este capítulo de la memoria se realiza una contextualización y descripción detallada de los sistemas OCC. Para ello, en primer lugar, se describen las distintas tecnologías existentes en el marco de las comunicaciones ópticas inalámbricas. Luego, se definen las formas de caracterizar el canal de comunicaciones ópticas inalámbricas y los elementos y fenómenos de mayor peso en los mismos. A continuación, se profundiza en las características y tipos de transmisores y receptores comúnmente utilizados en OCC, tras lo cual se describe la casuística sub-píxel. Finalmente, se analizan las posibles modulaciones usables con esta tecnología, y las principales aplicaciones de la misma.

2.1 Contextualización

Las comunicaciones ópticas inalámbricas son aquellas en las que se trabaja con ondas electromagnéticas en un rango de longitud de onda entre los 370 y los 940 nm, lo que comprende tanto el rango visible para el ojo humano como otras longitudes de onda como el infrarrojo o el ultravioleta. En las OWC, no se utiliza un medio que guíe la energía luminosa, como sí que ocurre con la fibra óptica. Esto hace que la comunicación sea más compleja y dependa del medio y sus cambios. No obstante, los sistemas OWC son más sencillos de instalar, no requieren grandes obras ni infraestructuras y facilitan la movilidad de sus nodos [16].

Cuando se habla de las OWC se suelen distinguir dos tipos de tecnología: los *Free Space Optics* (FSO) y las Visible Light Communication. Los primeros se refieren a conexiones punto a punto usando típicamente enlaces láser en la banda infrarroja (normalmente 1550 nm), lo que les proporciona una mayor directividad y por tanto les permiten alcanzar mayores distancias (hasta 5 kilómetros), así como la capacidad de penetrar a través del humo y la niebla. Son similares a los radioenlaces, pero son más baratos y tienen una mayor facilidad de instalación, por lo que son ideales para enlaces temporales o con altas necesidades de seguridad. Por su parte, las VLC se refieren típicamente a aquellas comunicaciones que utilizan dispositivos LED, arrays, o lámparas de los mismos como transmisores. Este tipo de tecnología alcanza menores distancias (hasta 10 metros) y velocidades más bajas pero es más segura para la vista, por lo que tiene mayor valor en zonas densas como interiores, exteriores o

entornos urbanos con fuertes interferencias de radiofrecuencia. Dado el potencial de esta tecnología, se prevé que el tamaño del mercado mundial de comunicaciones ópticas y de luz visible en el espacio libre pase de 6.500 millones de dólares en 2021 a 25.800 millones de dólares en 2030, con una tasa de crecimiento anual del 27% [17].

Dentro de las VLC, se utilizan principalmente dos tipos de receptores: los fotodiodos y los sensores de imagen. A su vez, como fotodiodos se suelen utilizar dos tipos: PIN (*Positive-Intrinsic-Negative*) y APD (*Avalanche PhotoDiode*).

El fotodiodo PIN es un dispositivo semiconductor que consta de tres regiones: una región p (positiva), una región i (*intrinsic*) y una región n (negativa). La región intrinsic, que es una capa no dopada, proporciona una región sensible a la luz. Cuando los fotones de luz inciden en esta región, generan pares electrón-hueco, que luego son separados por el campo eléctrico presente en el fotodiodo. La corriente generada como resultado de esta separación se puede medir y amplificar para obtener una señal útil. Los fotodiodos PIN son dispositivos de detección de luz de respuesta lineal, lo que significa que la corriente generada es proporcional a la intensidad de la luz incidente. Son los más extendidos por su reducido coste y fácil polarización.

Por otro lado, los fotodiodos APD son similares a los fotodiodos PIN en su estructura básica, pero incluyen una región de multiplicación adicional llamada región de avalancha. Esta región está diseñada para aprovechar el efecto de avalancha, que ocurre cuando un electrón acelerado por un campo eléctrico de alta intensidad colisiona con otros átomos y libera más electrones. Esto produce una ganancia interna en la señal generada, lo que resulta en una mayor sensibilidad en comparación con los fotodiodos PIN. Los fotodiodos APD tienen una respuesta no lineal, ya que la ganancia interna introducida por la región de avalancha causa una amplificación exponencial de la corriente generada. Esto los hace adecuados para aplicaciones que requieren una alta sensibilidad en condiciones de baja intensidad de luz. Si bien los fotodiodos de avalancha tienen una velocidad de respuesta, una sensibilidad y una relación señal a ruido mejores, su costo es mucho más elevado. La estructura de ambos tipos de fotodiodos se puede apreciar en la Figura 2.1.1 [18].

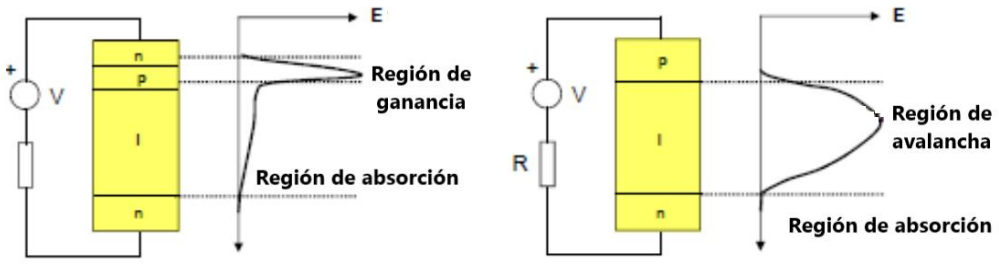

Figura 2.1.1: Estructura de fotodiodo APD a la izquierda, y PIN a la derecha

La tecnología de comunicaciones ópticas inalámbricas que usa fotodiodos como receptores es típicamente nombrada como VLC, y consigue mayores velocidades que la que usa sensores de imagen. En entornos de laboratorio se han alcanzado los 224 Gbps, y fuera de los mismos suele operar en torno a 1 Gbps [19]. Aquella tecnología VLC que además cumple con el estándar IEEE 802.15.7, en el que se regulan distintos aspectos clave de las OWC se denomina LiFi (*Light Fidelity*) [20]. Este tipo de tecnología es la que más éxito ha tenido hasta el momento en las VLC y se espera que alcance los 9.600 millones de dólares de ingresos en 2025.

Por su parte, las VLC que utilizan sensores de imagen como receptores son comúnmente conocidas como OCC. Si bien presentan una menor velocidad, este tipo de receptores son de menor coste y están presentes en mayor medida en la actualidad. Además, permiten la separación espacial de las fuentes, lo que a su vez abre la puerta a esquemas *Multiple - Input Multiple - Output* (MIMO). Es decir, a sistemas donde se aprovecha la propagación multitrayecto para incrementar la tasa de transmisión y reducir la tasa de error. Esta separación espacial de los transmisores también posibilita el desarrollo de sistemas de posicionamiento, que con los fotodiodos conllevaría una mayor complejidad y costo.

2.2 Canal

Los sistemas basados en cámaras presentan un comportamiento aceptable tanto en condiciones atmosféricas como submarinas. En ambos casos es necesario la existencia de una línea de visión directa entre transmisor y receptor, o bien entre este último y una reflexión de la señal transmitida. Por ello, el peor enemigo de los sistemas OCC son los obstáculos.

En ambos escenarios, atmosféricos y submarinos, las comunicaciones se pueden ver afectadas por tres fenómenos principales: la atenuación, la dispersión y las turbulencias. Estos afectan especialmente a enlaces de larga distancia como los FSO.

2.2.1 Condiciones atmosféricas

En condiciones atmosféricas homogéneas la atenuación atmosférica puede modelarse a través de la ley de Beer-Lambert, comúnmente usada para calcular las máximas distancias de un enlace óptico [21]. Esta ley obtiene la estimación total en base a las condiciones de absorción y dispersión molecular y de aerosol. También es posible estimar esa atenuación en base a la longitud de onda y las condiciones de visibilidad (definida como la máxima distancia a la que se puede distinguir un objeto oscuro sobre el horizonte). Por ejemplo, en la Figura 2.2.1.1 se pueden observar las máximas distancias alcanzables por un enlace óptico según distintas condiciones de visibilidad.

Figura 2.2.1.1: Edificio Chrysler ante nube de humo naranja con una máxima distancia de enlace óptico de 600 metros a la izquierda, y edificio Chrysler en noche despejada con una máxima distancia de 2,5 kilómetros a la derecha

Por su parte, la dispersión hace referencia a la modificación de la propagación de la luz al interactuar con las partículas del aire de diferentes dimensiones (agua, aerosoles, polvo…). Cuando las partículas que causan la dispersión tienen un tamaño inferior a 1/10 de la longitud de onda de la señal óptica se habla de dispersión Rayleigh. Este fenómeno es más propenso y afecta más para longitudes de onda corta, y produce una dispersión de energía similar en todas las direcciones (dispersión cuasi-uniforme). Por su parte, cuando las partículas que causan la dispersión son mayores que la longitud de onda de trabajo se trata de dispersión de Mie. Este tipo tiene una baja dependencia con la longitud de onda, e incrementa sus efectos cuanto mayor es el tamaño de las partículas. En este caso la energía no se distribuye de forma uniforme, sino que se propaga en mayor proporción en la dirección que seguía la luz originalmente [22]. Una comparativa de ambos tipos de dispersión se puede observar en la Figura 2.2.1.2.

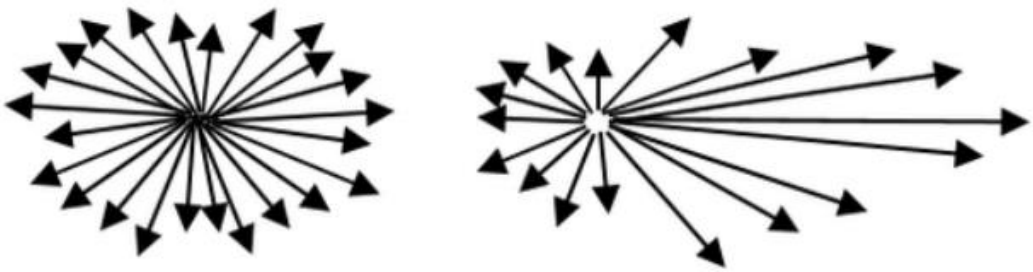

Figura 2.2.1.2: Dispersión de Rayleigh a la izquierda, dispersión de Mie a la derecha

Otro fenómeno a tener en cuenta en condiciones atmosféricas son las turbulencias, que son variaciones del índice de refracción debidas a diferencias de temperatura, originadas principalmente por los vientos y las nubes [23]. Estos sucesos pueden producir tres tipos de

fenómenos sobre la señal óptica recibida: el centelleo, la desviación del haz, y la fluctuación y variación de la amplitud y posición de la luz incidente. Estos dos últimos fenómenos se pueden apreciar en la Figura 2.2.1.3.

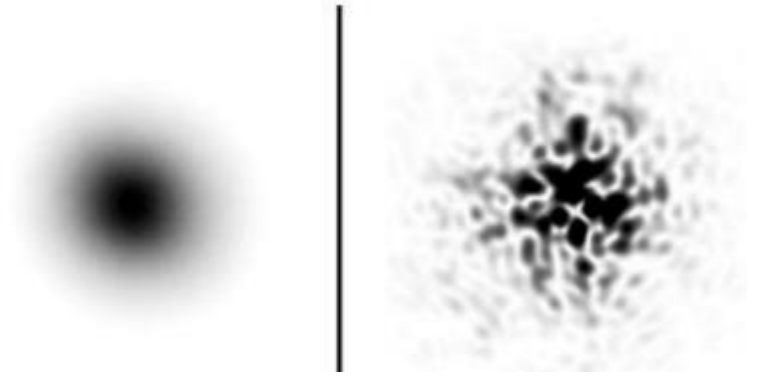

Figura 2.2.1.3: A la izquierda desviación del haz, y a la derecha variación de la amplitud y posición de la luz incidente

2.2.2 Condiciones submarinas

En escenarios submarinos los fenómenos de atenuación, dispersión y turbulencias tienen efectos mucho más perjudiciales sobre las señales ópticas. Esto se debe a que hay una mayor cantidad de factores que pueden ocasionar los mismos como son las corrientes marinas, la salinidad del agua, el material orgánico en suspensión, etc.

En este caso la atenuación se puede obtener mediante modificaciones de la ley de Beer-Lambert. Ahora, la constante de atenuación depende de otros factores que se agrupan según si producen absorción o dispersión [22]. Entre los elementos que influyen en la absorción se encuentran la longitud de onda, el tipo de agua (agua de mar pura, agua oceánica limpia, agua costera, agua turbia…), la cantidad de materia en suspensión, o la cantidad de fitoplancton. Si se combinan todas las atenuaciones que pueden causar los distintos elementos comentados en función de la longitud de onda, se obtiene la gráfica de la Figura 2.2.2.1 donde, como se puede observar, la luz azul es la que mejor comportamiento presenta.

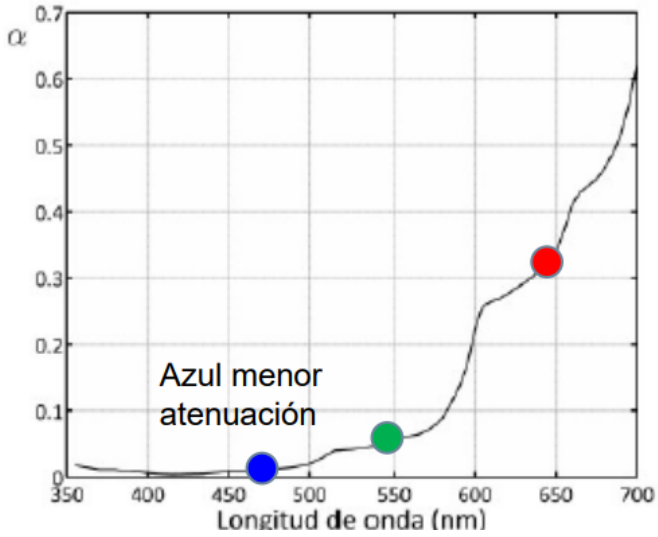

Figura 2.2.2.1: Absorción de las señales ópticas en condiciones submarinas en función de la longitud de onda (nm)

En cuanto a la dispersión, la más predominante es la dispersión de Mie, y es producida principalmente por la interacción de las señales ópticas con partículas de fitoplancton que tienen una longitud de onda similar a las longitudes típicamente usadas. La dispersión es el fenómeno atenuante predominante en condiciones submarinas donde el agua está turbia o próxima a la costa, tal y como se puede apreciar en la Tabla 2.2.1 que relaciona los valores del coeficiente de absorción, dispersión y atenuación (suma de coeficientes de absorción y dispersión) en función del tipo de agua [25].

Tipo de agua	**Absorción**	**Dispersión**	**Atenuación**
Agua de mar pura	0.0405	0.0025	0.043
Agua oceánica limpia	0.114	0.037	0.151
Agua costera	0.179	0.219	0.298
Agua turbia	0.266	1.824	2.19

Tabla 2.2.1: Comparativa de coeficientes de absorción, dispersión y atenuación en función del tipo de agua

2.3 Sistemas OCC

Como en los sistemas de comunicación convencionalmente utilizados en radiofrecuencias (RF), la transmisión y recepción en comunicaciones ópticas requiere de una serie de bloques de trabajo que permitan por una parte convertir la información en señales ópticas, y por otra detectar y extraer esa información. Del lado del transmisor los bloques que permiten estas tareas son el codificador, que permite representar la información sea cual sea su naturaleza y origen como combinación de unos y ceros, y el modulador, que modifica la señal óptica para transmitir esa información binaria. Tras estos bloques suele actuar un módulo de control que atiende a requisitos temporales, de sincronización, de ajuste de amplitud de la señal óptica en función de parámetros de tiempo real, etc. Por su parte, del lado del receptor es necesario en primer lugar el sensor de imagen que adquiera los fotogramas y un posible bloque de preprocesamiento que mejore la calidad de la señal adquirida. Tras estos módulos, es posible llevar a cabo las tareas de descubrimiento y seguimiento de transmisores, así como la demodulación y decodificación de la información transmitida por ellos. La aportación de este trabajo se centra precisamente en el bloque de descubrimiento y seguimiento de transmisores. Este esquema comentado se puede apreciar en la Figura 2.3.1.

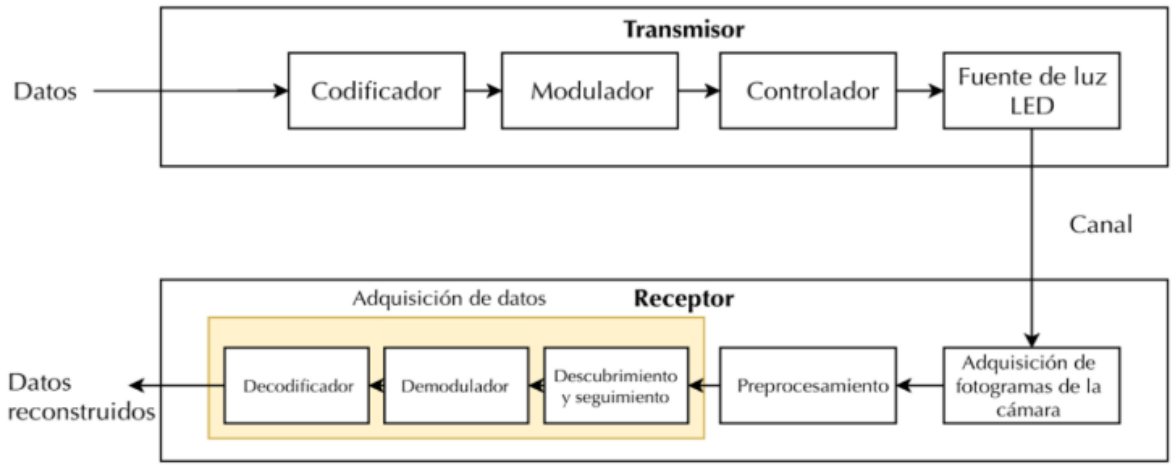

Figura 2.3.1: Diagrama de bloques de un sistema OCC

2.3.1 Transmisores

Como transmisores en comunicaciones ópticas, como ya se ha comentado a lo largo del documento, se usan fuentes luminosas. Estos elementos están presentes ampliamente en la actualidad y duplicar su funcionalidad apenas supone unos pequeños costes asociados a los bloques previos a las fuentes de luz comentados en la Figura 2.3.1. Como fuentes de iluminación típicamente se han usado [26]:

- Las lámparas incandescentes. Emiten luz al calentar un filamento muy fino hasta que empieza a brillar. Así, la cantidad de luz emitida a cada longitud de onda depende de la temperatura alcanzada por el filamento. En este tipo de lámparas la cantidad de luz visible emitida es muy pequeña ya que más del 90% de la electricidad gastada se pierde en forma de calor.
    - Las lámparas halógenas. Son una variante de las lámparas incandescentes que mejoran el rendimiento de las mismas a la par que aumentan la vida útil a las 2.000 - 4.000 horas de uso. Están formadas por un filamento de tungsteno dentro de un gas inerte y una pequeña cantidad de halógeno (como yodo o bromo). El filamento y los gases se encuentran en equilibrio químico, que es lo que hace que mejore el rendimiento del filamento y aumente su vida útil. El vidrio de las lámparas incandescentes se sustituye en estas lámparas por un compuesto de cuarzo, que soporta mucho mejor el calor, lo que permite lámparas de tamaño mucho menor para potencias altas.
- Las lámparas de descarga. - Consiguen emitir luz por excitación de un gas sometido a descargas eléctricas entre dos electrodos. Se pueden clasificar según el gas utilizado (vapor de mercurio o sodio) y según la presión a la que este se encuentre (alta o baja). Las lámparas de descarga más conocidas son las fluorescentes, que son aquellas que utilizan vapor de mercurio.

Dos de los tipos comentados se pueden observar gráficamente en la Figura 2.3.1.1.

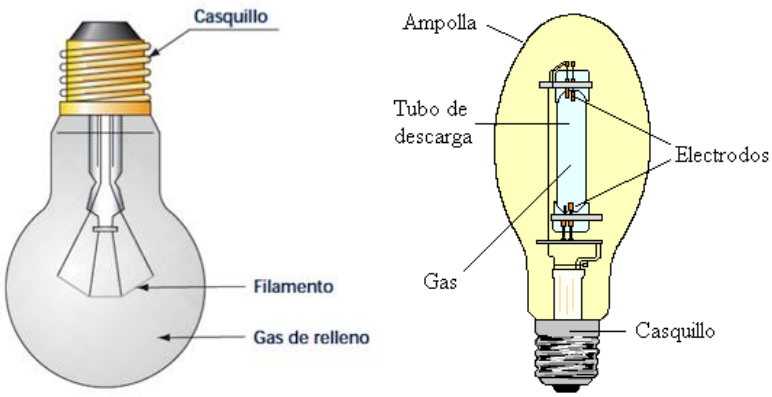

Figura 2.3.1.1: Lámpara halógena a la izquierda y lámpara de descarga a la derecha

No obstante, la mayor parte de ejemplos comentados no son aptos para la comunicación porque su capacidad de variación de la intensidad luminosa es muy pequeña y no permite transmisiones operativas. Además, incluso para fines de iluminación ya están en desuso debido a las múltiples

ventajas que proporciona la tecnología LED, la cual sí presenta una buena capacidad de variación luminosa.

Los LEDs basan su funcionamiento en el envío de energía a través de materiales conductores. Específicamente, consiste en el envío de un electrón a través de la banda de conducción a la de valencia. En este proceso se pierde energía y esa energía perdida puede manifestarse en forma de un fotón con amplitud, dirección y fase aleatoria. De esta manera, esa circulación de energía hace que se genere luz. Parte de la energía se desprende como calor, pero en una cantidad mucho menor a la de las opciones antes comentadas.

La estructura de LED más común es la llamada doble heterounión, una interfaz entre dos materiales semiconductores con diferentes bandas de energía (en oposición a lo que se llama homounión). Esta estructura se puede observar en la Figura 2.3.1.2. La capa p es la capa positiva del LED, dopada con impurezas de tipo p. Contiene huecos (deficiencia de electrones) como portadores de carga. Por su parte, la capa n es la capa negativa del LED, dopada con impurezas de tipo n, y que contiene electrones como portadores de carga. En la zona de depleción los electrones de la capa n y los huecos de la capa p se recombinan, creando un área con ausencia de portadores de carga.

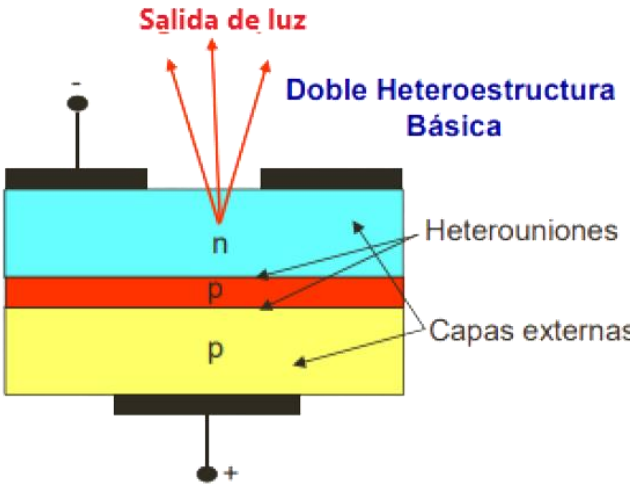

Figura 2.3.1.2: Estructura de un LED

Los LED son dispositivos casi lineales en su zona de trabajo. Esto es, que la potencia óptica emitida se incrementa de forma proporcional a la corriente de entrada del LED (casi lineal). No obstante, es muy dependiente de la temperatura. Para una determinada corriente la potencia óptica disminuye a medida que aumenta la temperatura. Este es uno de los principales factores a tener en cuenta junto a la respuesta espectral a la hora de evaluar lo favorable que es un LED para un diseño.

Debido a que los LEDs generan una menor cantidad de calor, presentan un comportamiento más eficiente y requieren de un menor consumo energético para producir la misma o mayor cantidad de potencia óptica que las lámparas incandescentes y de descarga. Esto conlleva a su vez una vida útil más elevada, superando las 50.000 horas. Asimismo, también suelen tener tiempos de encendido más rápidos que las tecnologías anteriores dado que no requiere del calentamiento de ningún filamento o la ionización de gases.

Otra ventaja de los LEDs es que son compactos y se pueden fabricar en diferentes formatos y formas, y combinar para formar estructuras más complejas. En una gran parte de los casos, son esas combinaciones de LEDs las que se utilizan en la práctica para aumentar la potencia óptica radiada y la región de interés. Entre las infraestructuras de LEDs más populares se encuentran [26]:

- LED DIP (*Dual In-Package*): Es un tipo de LED que se encuentra en un encapsulado con pines en ambos lados. Estos pines permiten una fácil soldadura y conexión a una placa de circuito impreso (PCB - *Printed Circuit Board*). Los LED DIP suelen ser más grandes y tienen una potencia menor en comparación con otros tipos de LED. Se utilizan en la mayoría de los electrodomésticos.

- LED SMD (*Surface Mounted Devic*e): Es un tipo de LED que se monta directamente sobre la superficie de una PCB. Los LED SMD no tienen pines, sino contactos metálicos en la parte inferior que se sueldan a la placa. Son más pequeños y planos en comparación con los LED DIP, lo que permite un montaje más compacto y eficiente.

- LED COB (*Chip On Board*): Es un tipo de LED en el que varios chips LEDs, generalmente con un tamaño inferior a 100 micrómetros (microLED), se montan directamente en una placa o sustrato. Los chips LED se colocan muy cerca unos de otros y se recubren con una capa de fósforo para crear una fuente de luz continua. Esta estructura ofrece un alto rendimiento lumínico (de hasta 120 lúmenes/vatio, dos veces más que un SMD), una mejor uniformidad y un mayor rendimiento térmico en comparación con otros tipos de LED. Además, soporta largos periodos de actividad, lo que la hace propicia para su instalación en lugares de trabajo donde se requieren largas horas de encendido.

Cabe destacar que, todas aquellas lámparas que utilizan la tecnología LED como fuentes de luz principales se denominan lámparas de estado sólido (SSL - *Solid State Logic*). En la Figura 2.3.1.3 se pueden observar algunos de los tipos de infraestructuras comentadas que se pueden utilizar para formar una SSL.

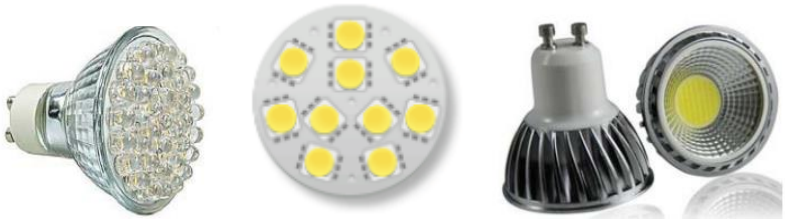

Figura 2.3.1.3: LED DIP a la izquierda, LED SMD en el centro y LED COB a la derecha

Adicionalmente a la tecnología LED, también cobra gran importancia en las OWC la tecnología OLED (*Organic Light-Emitting Diode*). Esta última tiene un principio de operación similar a la tecnología LED ya que produce radiación óptica a partir de la recombinación de electrones y huecos. No obstante, la tecnología OLED reduce los costes al trabajar con materiales orgánicos en lugar de los inorgánicos que usan los LEDs, a la par que conserva las ventajas comentadas de estos últimos. Los LEDs se utilizan ampliamente en iluminación y pantallas porque emiten luz puntual y direccional en diferentes colores, mientras que los OLEDs son populares en pantallas de alta calidad en dispositivos como televisores y teléfonos móviles debido a su mayor contraste y ángulos de visión más amplios [27]. Una comparativa de ambos tipos de iluminación se puede apreciar en la Figura 2.3.1.4.

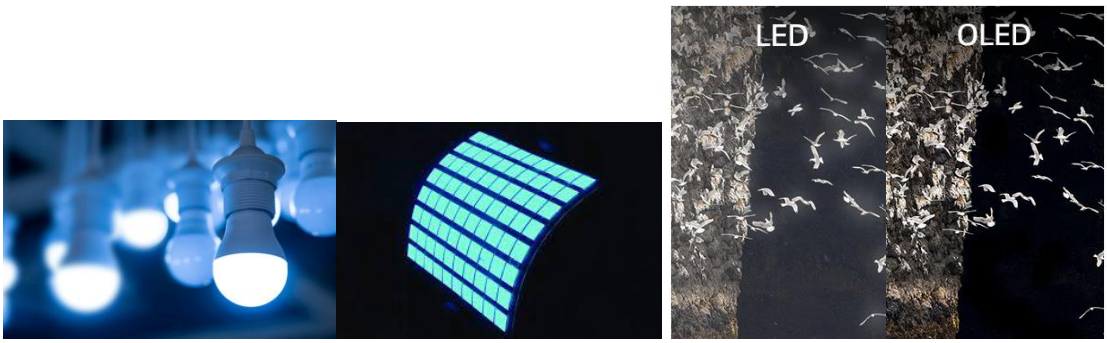

Figura 2.3.1.4: Fuente LED a la izquierda, fuente OLED al centro, y comparativa de televisión que trabaja con tecnología LED y una que trabaja con tecnología OLED

A la hora de hablar de las fuentes de luz de OCC se puede hacer referencia a distintos parámetros y fenómenos. Dentro del primer ámbito destacan [18]:

- Luminosidad: Es la percepción subjetiva que tiene un observador de la cantidad total de luz emitida por una fuente luminosa. Se mide en candelas por metro cuadrado (cd/m²).
- Flujo luminoso: Es la cantidad total de luz emitida por una fuente luminosa en todas las direcciones. Se mide en lúmenes (lm).

- Intensidad luminosa: Es la cantidad de luz emitida en una dirección específica, medida en candelas (cd).
- Emitancia: Es la cantidad de luz emitida por unidad de área de una superficie, medida en lúmenes por metro cuadrado (lm/m²).
- Irradiancia: Es la cantidad de energía radiante incidente en una superficie por unidad de área, medida en vatios por metro cuadrado (W/m²).

En lo relativo a la comunicación, cobran relevancia dos fenómenos que, debido a las necesidades de transmisión pueden afectar negativamente a la tarea de iluminación: el *flickering* y el *dimming*. El flickering se refiere al parpadeo (perceptible por el ojo humano) de la luz emitida por una fuente luminosa, debido a que presenta fluctuaciones con una frecuencia similar o inferior a 100 Hz (frecuencia de corte del ojo humano [28]). Esto no solo está relacionado con la tasa de transmisión, sino también con la estructura de las tramas. Por ejemplo, se puede transmitir con tasas de datos muy superiores a 100 Hz, pero que por la estructura de las tramas las fluctuaciones se produzcan con una frecuencia igual o inferior a esa cifra.

Por su parte, el dimming es una técnica que se utiliza para controlar la intensidad de la luz en una fuente de iluminación. Consiste en ajustar el nivel de brillo de una luz, ya sea reduciéndolo o aumentándolo, para crear diferentes ambientes, ahorrar energía o adaptarse a las necesidades específicas. El dimming se logra mediante la regulación de la corriente eléctrica o el voltaje suministrado a la fuente de luz. El problema reside en que las características de la transmisión puedan aumentar o disminuir ese nivel de brillo de forma indeseada.

2.3.2 Receptores

Como se ha comentado anteriormente, los elementos usados típicamente como receptores en las OCC son las cámaras. Estos dispositivos están presentes de forma masiva en la actualidad. Atendiendo sólo a cámaras de seguridad se pueden encontrar cifras que superan las 70 por kilómetro cuadrado en ciudades como Barcelona, tal y como se puede apreciar en los datos de la Figura 2.3.2.1 [30]. Teniendo en cuenta además la cantidad de vehículos con cámara frontal o trasera y la cantidad de *smartphones* presentes hoy día, la cifra de cámaras podría superar los 55 millones solo en España [31].

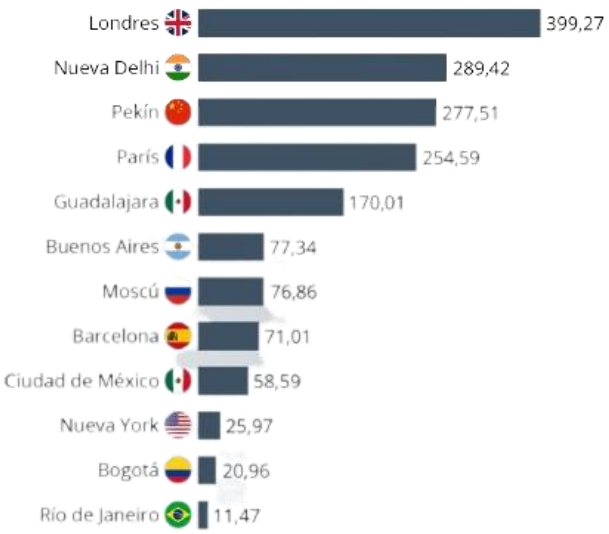

Figura 2.3.2.1: Cantidad de cámaras de seguridad por kilómetro cuadrado en 2020 en algunas de las ciudades más pobladas del mundo

Existen tanto tipos de cámaras como aplicaciones se les pueden dar a las mismas: videovigilancia, fotografía, asistencia a la conducción, etc. No obstante, pese a su apariencia externa, su principio de funcionamiento y arquitectura son bastante similares en todos los casos.

El principio de operación es bastante sencillo. La cámara utiliza una óptica para capturar los rayos de luz que provienen de diferentes direcciones dentro de su campo de visión (lo que la cámara puede ver). Estos rayos de luz son proyectados en diferentes puntos de un sensor de imagen. Antes de llegar a los píxeles del sensor, los rayos de luz atraviesan filtros ópticos que dividen sus longitudes de onda en tres canales utilizados para formar la imagen: rojo, verde y azul. Luego, cada píxel del sensor integra la intensidad de la luz recibida en cada canal y genera una carga eléctrica. La cantidad de carga eléctrica generada depende de la sensibilidad del píxel a diferentes longitudes de onda de luz y de la duración de la exposición.

Después, una circuitería electrónica lee la carga acumulada por cada celda del sensor. Esta circuitería adapta la señal para que pueda ser convertida de analógica a digital utilizando un convertidor analógico-digital (ADC - *Analog to Digital Converter*). Antes de dicha conversión, se aplican etapas de amplificación analógica lineal y no lineal, como la transformación *gamma*. Luego, el ADC cuantifica la señal electrónica y genera una imagen con valores restringidos por la resolución del convertidor. Por ejemplo, si el ADC tiene una resolución de 8 bits, los valores de cada canal de la imagen estarán comprendidos entre 0 (sin intensidad) y 255 (máxima intensidad) [32].

La arquitectura de la cámara se podría dividir en tres secciones principales [33]: el sistema óptico, el sensor de imagen y el circuito de salida. El sistema óptico proyecta la luz en el sensor de imagen. El sensor de imagen, formado múltiples celdas fotosensibles (píxeles), genera por cada celda un voltaje proporcional al número de fotones recibidos en la misma. A su vez, los píxeles están conectados a un circuito externo que convierte el voltaje de los píxeles en datos binarios.

2.3.2.1 Sistema óptico

El sistema óptico de la cámara se compone de lentes y filtros ópticos que descomponen la luz y permiten la formación de la imagen en la superficie del sensor. Este sistema se puede subdividir en varias etapas.

La primera etapa del sistema óptico consta de una serie de lentes convergentes y divergentes, junto con un elemento limitante llamado diafragma. Estas lentes y el diafragma trabajan en conjunto para redirigir los haces de luz incidentes y enfocarlos en la superficie del sensor. El diafragma también tiene la función de regular la cantidad de luz que llega al sensor, controlando así la intensidad luminosa.

La segunda parte del sistema óptico está formada por una matriz de microlentes ubicada en la superficie del sensor. Cada píxel del sensor tiene una microlente asociada que determina su ángulo de visión específico. Además, esta matriz de microlentes incorpora filtros ópticos que descomponen la luz en los tres canales utilizados para la formación de la imagen: rojo, verde y azul. Estos filtros se organizan en un patrón simétrico que puede variar entre los diferentes sensores de imagen.

El patrón de filtros más comúnmente utilizado es conocido como filtro de Bayer, el cual se repite en un patrón de 2x2. Este patrón consiste en dos filtros verdes, uno rojo y uno azul para cada conjunto de cuatro píxeles adyacentes. Esta configuración se elige para otorgar a la cámara una mayor sensibilidad en el rango espectral verde, ya que el ojo humano percibe mejor las variaciones de intensidad en esa región del espectro. El filtro de Bayer junto al resto de partes del sistema óptico comentadas se pueden apreciar en la Figura 2.3.2.1.1.

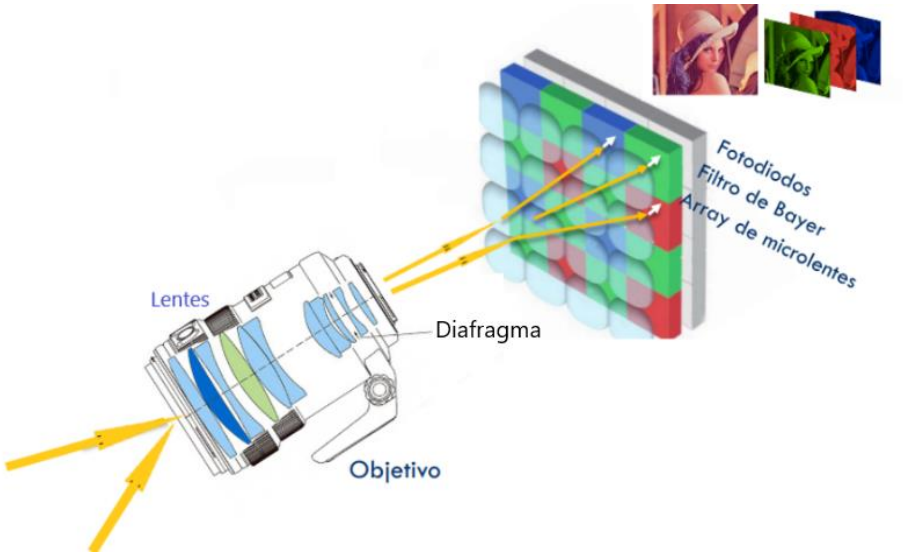

Figura 2.3.2.1.1: Sistema óptico de una cámara

2.3.2.2 Sensor de imagen

El sensor de imagen genera un mapa de valores a partir de la intensidad luminosa de la luz incidente en cada píxel. Este elemento también se puede descomponer en tres partes fundamentales [32 - 33]:

- La parte fotorreceptora, que mediante los fotodiodos que la forman, convierte la luz incidente en carga eléctrica (conversión fotoeléctrica).
- La parte de transmisión de carga, que transfiere la carga eléctrica acumulada y la transforma en niveles de tensión.
- La parte de conversión, que genera la señal digital que representa la imagen. Para ello, hace uso de de convertidores ADC y de los niveles de tensión obtenidos en la parte anterior.

Hay dos tipos principales de sensores de imagen atendiendo a la forma en la que realizan la transmisión de carga desde los fotodiodos hasta el ADC: los sensores CCD y los CMOS. Usar un tipo u otro conlleva usar el modo de adquisición GS o RS y por tanto funcionalidades y aplicaciones distintas.

2.3.2.2.1 CCD

Los sensores CCD consisten en una malla compacta de electrodos de polisilicio dispuestos en la superficie de un chip. Cuando los fotones de luz inciden sobre el silicio de cada píxel, generan electrones que pueden ser almacenados temporalmente. Luego, se produce una transferencia en paralelo de los electrones desde cada píxel hacia una estructura llamada registro de desplazamiento vertical. Esta estructura está formada por una serie de electrodos que se activan secuencialmente para desplazar los electrones almacenados en cada píxel hacia el siguiente píxel adyacente. De esta

manera, la carga eléctrica generada por la luz se transfiere en serie desde el primer píxel hasta el último píxel en el sensor. Esta estructura se puede observar en la Figura 2.3.2.2.1.1.

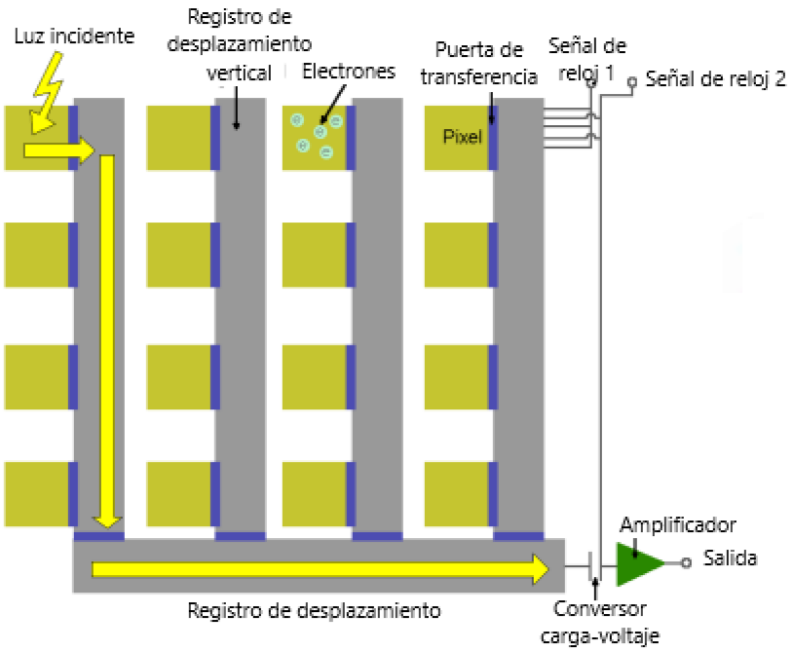

Figura 2.3.2.2.1.1: Sensor de imagen CCD

Una vez que se ha completado la transferencia de carga mediante los registros de desplazamiento verticales, un último registro de desplazamiento horizontal dirige la carga a un convertidor analógico a digital para su cuantificación y posterior procesamiento. La lectura de la señal en cada píxel se realiza rápidamente, permitiendo que todos los píxeles capturen la luz simultáneamente.

Este método de captura simultánea en todos los píxeles es beneficioso en muchas aplicaciones, especialmente en aquellas donde se necesita capturar objetos en movimiento o escenas con cambios rápidos. Al capturar la señal al mismo tiempo se evitan problemas como la distorsión de objetos en movimiento, que se produciría si se capturase de forma secuencial. Además, proporciona una representación más precisa y coherente de la escena en su totalidad.

No obstante, por su forma de operación y traslado de cargas, los sensores CCD son más propensos a la propagación de carga entre píxeles. Es decir, cuando una región es muy brillante, puede ocurrir que parte de esa carga se acabe “desbordando” y afectando a los píxeles aledaños. Este fenómeno es conocido como *blooming*, y ocasiona en la imagen distorsión y áreas con brillo excesivo. Este fenómeno se puede apreciar en la Figura 2.3.2.2.1.2.

Figura 2.3.2.2.1.2: Blooming

2.3.2.2.2 CMOS

En cuanto a los sensores de imagen CMOS, son los más utilizados debido a sus bajos costes y sencillez de implementación [33]. La principal diferencia con los sensores CCD es que ahora cada píxel está compuesto por un fotodiodo tipo PIN, un condensador para el almacenamiento de carga, y un amplificador de bajo ruido. Esto permite a cada píxel gestionar su propia energía, poder ser seleccionado de forma independiente, y evitar problemas como el blooming.

Otro aspecto destacable de los sensores CMOS es que no usan un único ADC para todos los píxeles, sino que disponen de uno por cada columna de píxeles, lo que permite tiempos de lectura más cortos y mayores tasas de fotogramas que los CCD. Asimismo, para maximizar aún más las tasas de captura, cada fila comienza su exposición mientras se completa la lectura de la anterior. Este proceso se puede observar gráficamente en la Figura 2.3.2.2.2.1.

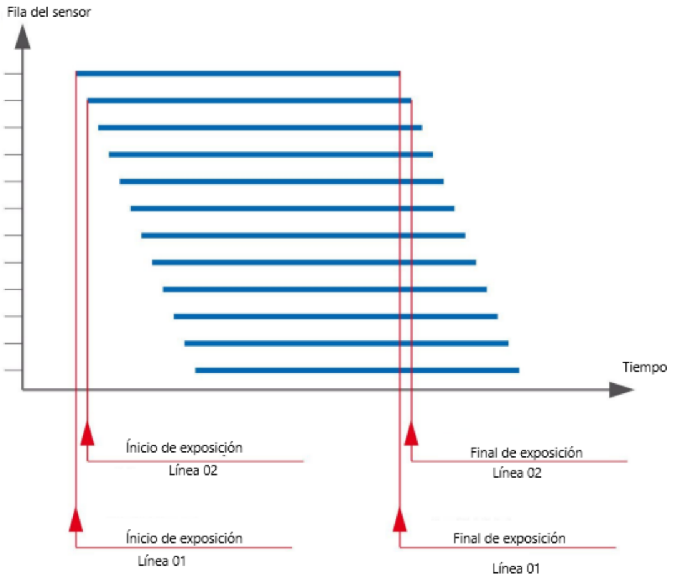

Figura 2.3.2.2.2.1: Captura CMOS

Esta exposición y lectura secuencial de las filas en el sensor de imagen dan lugar a los modos de adquisición RS. El problema de este tipo de exposición y lectura es que introducen distorsión en aquellas imágenes que cambian en un tiempo inferior al tiempo que tarda en operar cada fila. Esa distorsión introducida debido al cambio de la imagen en tiempos inferiores al de la captura de cada línea se denomina efecto RS. Ese efecto se puede apreciar en la Figura 2.3.2.2.2.2.

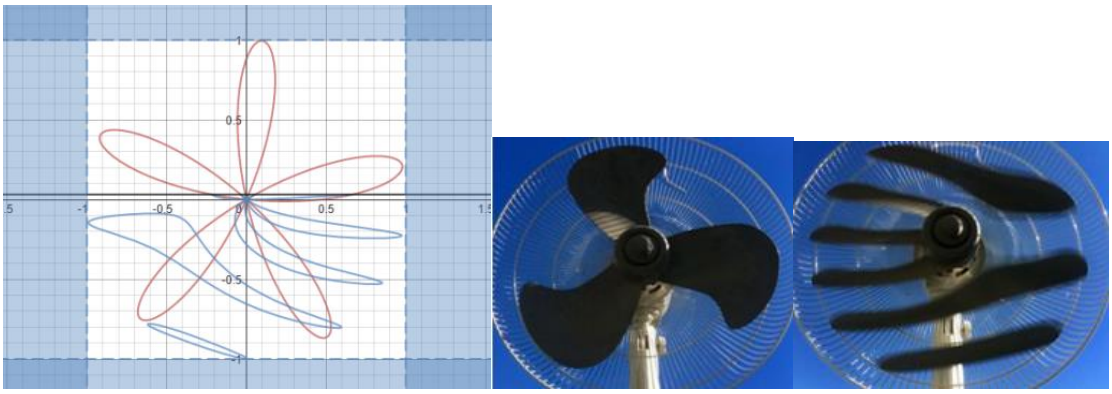

Figura 2.3.2.2.2.2: Efecto RS

En cuanto a comunicaciones se refiere, el usar sensores CMOS permite aumentar la tasa de datos debido no solo al aumento de los *Frames Per Second* (FPS), sino a que cada línea o agrupación de líneas puede utilizarse para recibir un símbolo. Como el tiempo que se tarda en capturar una línea o grupo de las mismas es menor que en procesar una imagen completa, mayor es la cantidad de datos que se puede procesar.

Por su parte, usar sensores CCD implica tasas de transmisión más bajas, ya que mayor es el tiempo que tarda en realizar la captura de cada imagen y menor es la cantidad de tramas por segundo. Esta reducción de la cantidad de FPS implica un mayor tiempo en el envío de cada símbolo y por tanto es más probable que se produzca flickering en estos casos. Para evitarlo, es necesario usar modulaciones más complejas que la comúnmente usada en OCC: la *On - Off Keying* (OOK). No obstante, en fuentes luminosas que no están destinadas a la iluminación sino a la comunicación u otros fines como la decoración, ese efecto de parpadeo puede ser aceptado.

2.4 Casuística sub-píxel

Cuando se habla de la casuística sub-píxel en comunicaciones ópticas basadas en cámara, se hace referencia a que la proyección de la fuente luminosa transmisora sobre el sensor de imagen tiene una ocupación inferior a un píxel. Aunque la documentación relativa a esta casuística es escasa, se sabe

que esto depende tanto del tamaño de la fuente luminosa, como de la distancia a la misma y las características del sensor de imagen, especialmente su resolución [14].

No obstante, es necesario considerar que, aunque la proyección del píxel en el sensor de imagen sea inferior a un píxel, la proyección de luz sobre la misma puede ocupar una mayor cantidad de celdas del sensor de imagen. Esto depende de varios factores como son la luminosidad media del escenario, la luminosidad y directividad de la fuente o las distorsiones que pueda introducir el canal. A mayor luminosidad media del escenario, menor es la cantidad de luz de la fuente captada, ya que esta se ve opacada por la luz ambiente. A mayor luminosidad de la fuente, mayor probabilidad existe de que su emisión afecte a varios píxeles debido a la dispersión introducida por el canal.

Por último, las imperfecciones del sistema óptico también suelen ser responsables de que una fuente acabe ocupando más de un píxel en el sensor de imagen, aunque su proyección sea inferior a uno de los mismos. Esto se debe a que cuando la luz de una fuente puntual pasa a través de una lente o un sistema óptico, experimenta modificaciones y dispersión debido a imperfecciones en la lente y características del sistema óptico. Esto resulta en una extensión o difusión del punto original en la imagen proyectada.

Para describir cómo un sistema óptico o una lente desenfoca un punto de luz en una imagen se utiliza generalmente la *Potencial Spread Function* (PSF), también conocida como Función de Difusión Potencial. Por lo general, se representa como una función bidimensional, donde cada punto en la imagen de salida está relacionado con un valor que indica la intensidad de luz en un punto de entrada y el volumen de la región iluminada [15]. La PSF es importante a la hora de determinar la posición de origen de una fuente luminosa. Conociendo la PSF del sistema óptico específico, es posible aplicar algoritmos de procesamiento de imágenes para deshacer o corregir el efecto de la difusión y mejorar la calidad de la imagen.

2.5 Modulaciones

En el ámbito de las OCC, las modulaciones juegan un papel crucial para codificar y transmitir información de manera efectiva a través de fuentes de luz. Entre las modulaciones más empleadas en OCC, como ya se comentó, se encuentra la OOK. Esta destaca por su simplicidad, ya que la información se transmite mediante la alternancia de la intensidad de luz entre dos valores, típicamente encendido y apagado. Aunque en RS es posible conseguir tasas de datos elevadas con esta modulación gracias a la lectura secuencial, en GS implica bajas tasas de datos y que se produzca flickering debido a la baja frecuencia que es necesario usar para que la cámara pueda detectar los cambios correctamente. En

la Figura 2.5.1 se puede apreciar una comparativa de qué es lo que transmite la fuente y qué es lo que detecta cada tipo de sensor de imagen (GS o RS) en cada caso.

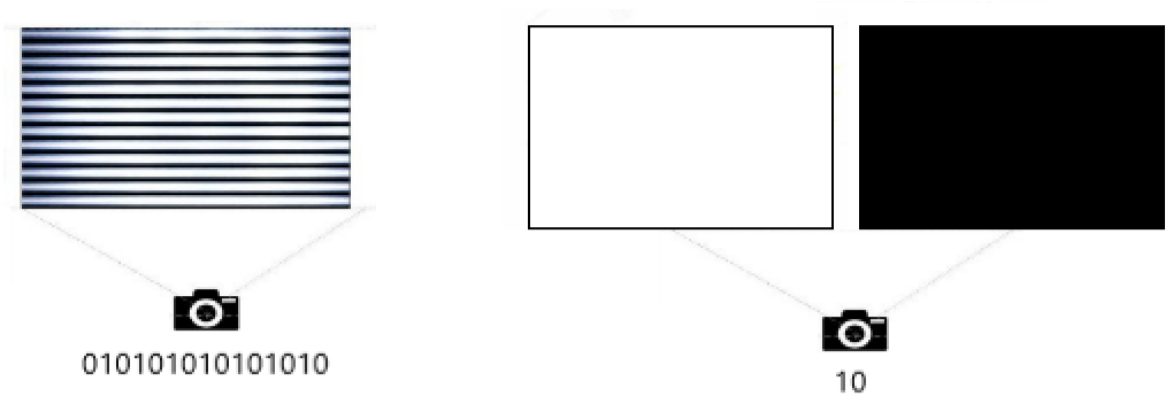

Figura 2.5.1: Lámpara transmitiendo para sensor RS y trama detectada a la izquierda, y lámpara transmitiendo para sensor GS y trama detectada a la derecha

Para solventar los problemas asociados al uso de la modulación OOK en GS existen dos soluciones. La primera consiste en usar cámaras con mayor tasa de frames por segundos (mayor a 200 FPS) para que la transmisión no se detecte por el ojo humano y para poder aumentar la tasa de datos por segundo. No obstante, esta opción aumenta considerablemente los costes del sistema. La segunda solución consiste en usar técnicas de undersampling, que son aquellas en las que se codifican los datos con frecuencias superiores a las de la cámara y se analiza la señal submuestreada para detectar los datos. Esto permite evitar el problema del flickering, pero no el de las bajas tasas de datos.

Existen dos técnicas principales de undersampling dentro de la modulación OOK: la *Undersampled Frequency Shift On-Off Keying* (UFSOOK) y la *Undersampled Phase Shift On-Off Keying* (UPSOOK) [34]. En la UFSOOK, se utilizan dos frecuencias distintas para representar los bits de datos 0 y 1. Estas frecuencias, conocidas como frecuencias de espacio (0 lógico) y frecuencias de marca (1 lógico), son siempre superiores a 100 Hz para evitar el parpadeo. Si se observa el mismo valor en las muestras sucesivas, indica que se está transmitiendo un bit de 0 (frecuencia de espacio). Si se observa una alternancia en los valores de las muestras, indica que se está transmitiendo un bit de 1 (frecuencia de marca). Un ejemplo se puede observar en la Figura 2.5.2, donde se transmite la secuencia 01. Esto se puede saber teniendo en cuenta que se toman dos muestras de cada bit, y que las dos primeras muestras toman valores consecutivos (ON, ON) y las dos siguientes valores alternos (ON, OFF) [28]. En esa imagen se trabaja con una frecuencia de 120 Hz para el cero lógico (línea discontinua, 8 ciclos de OOK) y 105 Hz para el uno lógico (línea continua, 7 ciclos de OOK).

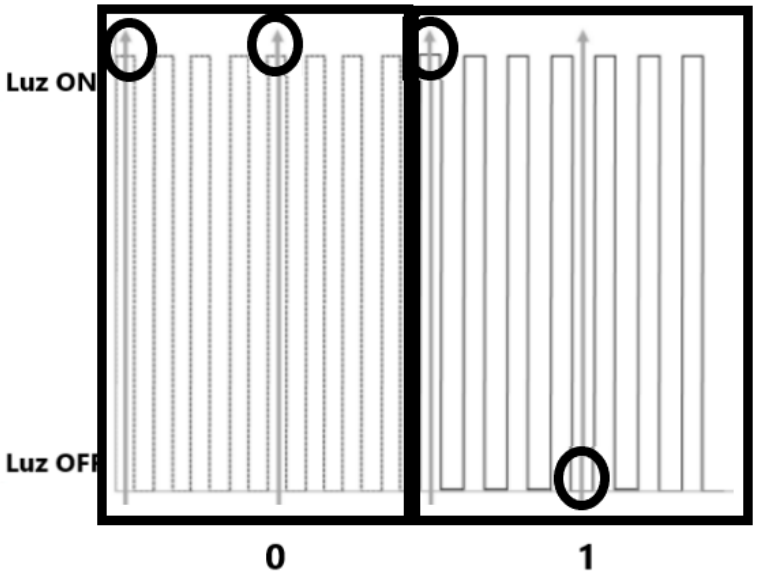

Figura 2.5.2: UFSOOK transmitiendo 01

Por su parte, la UPSOOK utiliza la misma frecuencia para transmitir 0 y 1 lógicos. En este caso lo que se realiza para cambiar el estado sobre el que se submuestrea es cambiar la fase de la señal [34]. Cuando se submuestrea un estado ON se considera que se recibe un 1 y cuando se submuestrea un estado OFF se considera que se recibe un 0. Un ejemplo se puede apreciar en la Figura 2.5.3 donde se transmite la combinación 110.

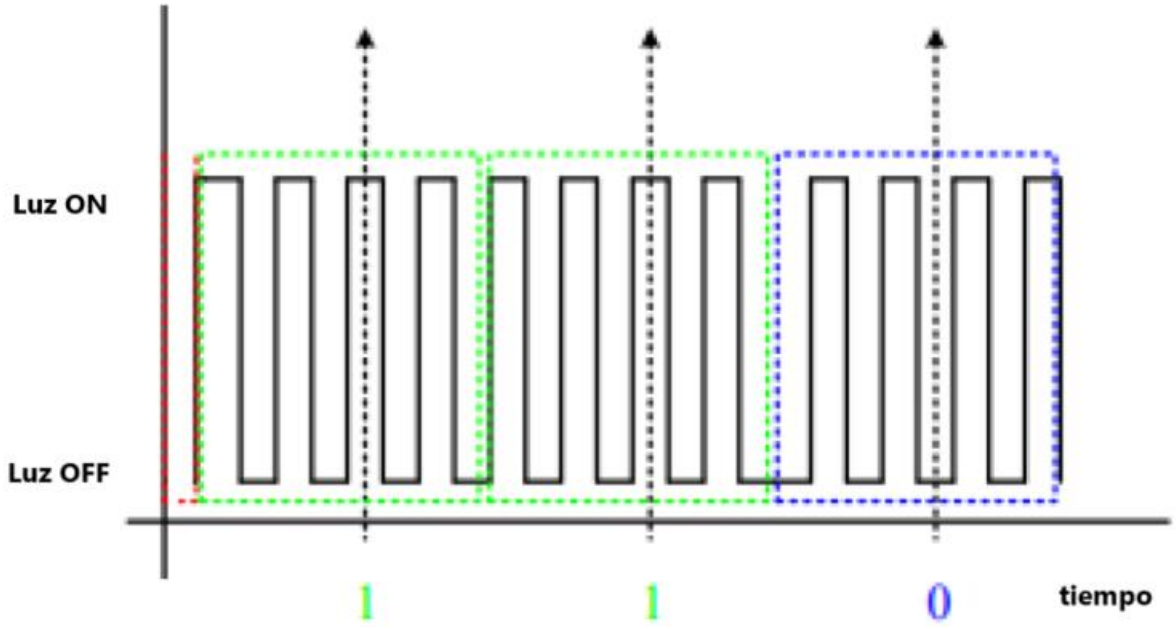

Figura 2.5.3: UPSOOK transmitiendo 110

Además de las opciones comentadas existen otras más complejas que suelen tener cabida en las OCC, sobre todo cuando se usan sensores de imágen CMOS y el modo de adquisición RS. Un ejemplo es la

Variable Pulse Position Modulation (VPPM) que, al igual que OOK, trabaja con pulsos de luz. No obstante, en este caso incorpora una variación en la posición de los pulsos para codificar información adicional y una variación en el ancho de los mismos para controlar el dimming de la lámpara a la vez que se realiza la transmisión. Como se puede apreciar en la Figura 2.5.4, aquellos pulsos que sólo están presentes al comienzo del periodo representan un 0, y los que sólo están presentes al final del mismo representan un 1 [36].

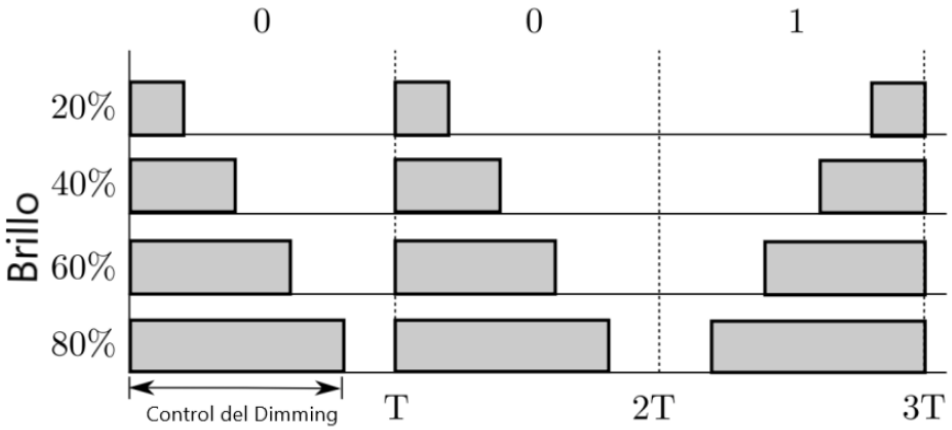

Figura 2.5.4: Modulación VPPM

De igual modo, las modulaciones por ancho de pulso (PWM - *Pulse Width Modulation*) y amplitud de pulso (PAM - *Pulse Amplitude Modulation*) han sido ampliamente utilizadas en la tecnología OCC con modos de adquisición RS. Esta última modulación también juega un papel importante al usar el modo de adquisición GS. No obstante, PAM requiere una mayor complejidad que conlleva un incremento en los costes y PWM suele presentar problemas de tasa de error de bit [36].

En situaciones donde se utilizan fuentes de luz RGB (*Red, Green, Blue*), la modulación *Color Shift Keying* (CSK) es relevante. Esta consiste en la combinación de las componentes RGB para formar los diferentes símbolos de una constelación con los que se codifican los datos. Un ejemplo de la misma se puede observar en la Figura 2.5.5.

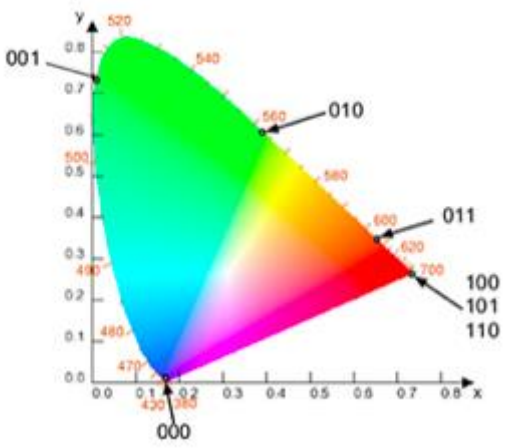

Figura 2.5.5: Modulación CSK

Otra importante modulación dentro del marco de las OWC, especialmente en el caso de las VLC y las OCC que trabajan con RS es la *Orthogonal Frequency Division Multiplexing* (OFDM). El principio básico de la OFDM es dividir el ancho de banda disponible en múltiples subportadoras ortogonales. Cada subportadora transporta una parte de los datos que se van a transmitir. Al utilizar subportadoras ortogonales, se minimizan las interferencias entre ellas, lo que permite una transmisión eficaz de los datos. OFDM ofrece varias ventajas en las comunicaciones ópticas, como una alta eficiencia espectral, resistencia a las interferencias, ecualización mejorada del canal, utilización eficiente del espectro y mitigación de la interferencia entre símbolos (ISI - *Inter Symbol Interference*) [37].

Existen varias variantes de OFDM como son la *Asymmetrically Clipped Optical - OFDM* (ACO-OFDM), la *Direct Current - biased Optical - OFDM* (DCO - OFDM) o la *Discrete Fourier Transform - OFDM* (DFT-OFDM), ampliamente probadas en VLC. Sin embargo, el uso de esta tecnología en la comunicación óptica por cámara (OCC) es reciente, y aún se están desarrollando y probando nuevas variantes como el *Rolling* OFDM, para sensores de imagen CMOS [37].

2.6 Aplicaciones

Las aplicaciones que presentan las OCC son tan amplias como las que poseen las radiofrecuencias. Sin embargo, como se comentó anteriormente, hay escenarios donde su operación cobra más importancia y otros en los que su operación es innecesaria por otras tecnologías existentes.

Este es el caso por ejemplo de las comunicaciones indoor, donde las radiofrecuencias se ven limitadas por las atenuaciones que sufren al atravesar paredes. Si bien en comunicaciones indoor con velocidades elevadas la tecnología OCC no tiene cabida frente a VLC, es ideal para aquellas con bajas

tasas de datos, necesidades de bajo costo, y requerimientos de seguimientos de nodos. Esto la convierte en una tecnología idónea para el desarrollo de sistemas de posicionamiento en interiores.

Varios son los estudios que han tratado el uso de las OCC en entornos indoor. Por ejemplo, en [38] un ejemplo de sistema posicionamiento indoor es probado utilizando fuentes puntuales como balizas, consiguiendo precisiones de en torno a 7.4 cm. En [39] técnicas de fotogrametría son utilizadas para desarrollar un sistema de posicionamiento indoor alcanzándose en ese caso precisiones en torno a 10 cm. En [40] redes neuronales son utilizadas alcanzando precisiones del orden de 10 mm. En [41] se planifica y plantea la utilización de las luces LED en salas de concierto para obtener y proporcionar información de posición a los usuarios a través de sus cámaras. En [42] se propone un esquema híbrido donde se combina la tecnología VLC y OCC para ofrecer tanto altas como bajas velocidades de datos usando una única lámpara transmisora.

Debido a su típica baja tasa de datos, otra de las principales aplicaciones de las OCC tanto en interiores como exteriores son las redes de sensores, donde el sector industrial es el principal interesado. En este ámbito las OCC proporcionan también soluciones de fácil instalación, movilidad y bajo costo. En [43] por ejemplo, se implementa un sistema OCC para el monitoreo de temperatura de válvulas en el interior de una fábrica, y en [8] se evalúa la viabilidad de implementar una red de sensores en un campo al aire libre en un entorno agrícola emulado.

En entornos indoor, el otro principal sector interesado en la tecnología OCC es el médico, donde permiten llevar a cabo tareas de monitorización sin afectar al equipo sensible a radiofrecuencias frecuente en esos entornos, y proporcionan mayores garantías de seguridad [44]. En estos casos, por ejemplo, los sensores asociados a electroencefalogramas (EEG - *Electroencephalogram*), electromiografías (EMG - *Electromyography*) y electrocardiogramas (ECG - *Electrocardiogram*) podrían transmitir su información mediante LEDs infrarrojos y parches sobre la piel, permitiendo reducir además la cantidad de cables necesarios [44]. En [45], por ejemplo, se diseña e implementa un sistema de comunicación por cámara óptica para la monitorización remota en tiempo real de los datos de frecuencia cardiaca y saturación de oxígeno del paciente. Asimismo, en [46] se propone una arquitectura híbrida OCC/RF para la monitorización remota de la salud de los pacientes en hospitales. En [47] se realiza un diseño con un propósito similar utilizando únicamente la tecnología OCC y las redes de acceso 5G.

Por último, también se están llevando a cabo múltiples estudios sobre los beneficios que las OCC podría aportar al turismo. En este ámbito se podrían usar para localización y seguimiento de turistas,

IoT, métodos de pago y acceso seguros, proporción de información contextual o automatización de funciones, entre otros [48].

Por otro lado, en situaciones outdoor, la aplicación más prometedora de las OCC está asociada al sector automovilístico, donde no solo permitiría una comunicación eficiente, de bajo coste y con buena precisión entre vehículos, sino también entre los mismos e inmobiliario público (farolas, semáforos, etc.). El término acuñado a este sector emergente es Internet de los Vehículos (IoV - *Internet of Vehicles*), que se refiere a la interconexión de vehículos, infraestructuras y usuarios a través de Internet para permitir la comunicación y el intercambio de información en tiempo real. Esta es una de las dos aplicaciones más ampliamente meditadas junto al sector médico. En la práctica, algunos de los estudios que se han llevado a cabo incluyen la integración teórica de sistemas OCC con la Nube [49], la discusión de los beneficios y aplicaciones de utilizar las comunicaciones entre vehículos y entre estos y las infraestructuras de la carretera [50], y simulaciones de estas comunicaciones [51]. Asimismo, en ambientes outdoor también cabe destacar el uso de las OCC en Smart Cities, donde se podría aprovechar la luminaria ya presente en las mismas para trabajar con información sensórica y posicionamiento, principalmente [14].

Por último, otra de las potenciales aplicaciones de las OCC se encuentra en el sector submarino. Esta rama de conocimiento es denominada *Underwater Optical Camera Communications* (UOCC). Gracias a su elevado ancho de banda y su baja latencia en estos escenarios, podría posibilitar las comunicaciones subacuáticas con fines de exploración y vigilancia de actividades marinas. En este sentido, la mayor parte de trabajos se dividen entre aquellos que se dedican a caracterizar el canal, como pueden ser [24 - 25, 52], y los que implementan enlaces en esas circunstancias. En [2] por ejemplo, se prueba experimentalmente un sistema UOCC donde se alcanza una distancia de 2 m con una velocidad binaria de 8 bps por canal (24 bps en total). En [53] por su parte, se implementa un enlace unidireccional de UOCC, en el que se obtienen velocidades de 100 bps para distancias entre 20 y 100 cm.

Capítulo 3

3. Sistema de comunicaciones

Para poder desarrollar el algoritmo de detección es necesario definir, en primer, lugar el sistema de comunicaciones sobre el que operará. Por ello, a continuación se describen todas las decisiones tomadas relativas a este aspecto. Se comentará la trama de comunicación y la modulación utilizadas, los equipos y software asociados a transmisores y receptores, y los escenarios y condiciones en los que se ha implementado el sistema.

3.1 Trama

El estándar IEEE 802.15.7r1 define estructuras de trama para los sistemas VLC y OCC. Esta estructura es jerárquica y conforma una supertrama [1]. Sin embargo, el estándar no ha sido adoptado por el mercado ya que es muy complejo y no presenta ventajas significativas frente a otro tipo de soluciones más sencillas.

Por ello, en este trabajo, se ha optado por definir una trama de corta duración que facilite la detección de las fuentes de luz. En ella se reduce la cantidad de *payload*, así como se omite información de control o direccionamiento en la cabecera, para favorecer la transmisión de cantidades pequeñas y puntuales de datos. Así, el sistema definido está principalmente orientado a las redes de sensores.

Precisamente por estar orientada a redes de sensores, se presupone que los transmisores pudieran estar a distancias lejanas del receptor. Si el transmisor solo ocupa uno o unos pocos píxeles del sensor de imagen, no se pueden usar distintas líneas del sensor de imagen para recibir distintas bandas de información en una modulación OOK. Por ello, las ventajas que pudiera tener usar un modo de adquisición RS no están presentes en estos escenarios.

Como resultado del poco área que los transmisores pueden proyectar sobre el sensor de imagen (ROI - *Region Of Interest*), en estos casos se podrían utilizar tanto cámaras en modo de adquisición RS, como en modo GS. Usar ambos tipos es de interés porque, aunque las cámaras con RS están más presentes debido a su bajo precio, las cámaras GS están presentes en multitud de cámaras de videovigilancia debido a su mayor calidad de imagen. No obstante, dado que las cámaras GS trabajan

con tasas de FPS generalmente más bajas, esto obliga a que los tiempos de transmisión de cada bit en la trama tengan que ser significativamente más amplios para garantizar su correcta detección. A mayor duración de cada bit, mayor probabilidad de realizar una correcta detección, pero menor es la tasa de datos.

La trama utilizada en este trabajo utiliza una duración para cada bit de 133.33 ms. Teniendo en cuenta que las cámaras típicamente operan con una tasa de 30 FPS, el tiempo que tardan en captar una imagen es de en torno a 33.33 ms. Es decir, de cada bit se toman unas 4 muestras.

Para facilitar la identificación del comienzo de la trama se establece que como máximo pueden haber cinco bits a 1 seguidos, y estos cinco bits a 1 se corresponden con la cabecera de la trama. Para garantizar esto, tras la cabecera y cada cuatro bits de payload se ubica un bit a 0 como banda de guarda (a excepción de cuando se alcanza el final de trama).

Dado que en redes de sensores típicamente se envían pocas cantidades de información, la trama únicamente contendrá 8 bits de datos. Dependiendo de la aplicación, parte de esos datos se pueden dedicar a identificación, verificación, o al envío de datos. Así, la estructura final de la trama, de duración total 2 segundos, se puede observar en la Figura 3.1.1. Esta estructura de trama fue diseñada en el trabajo [6], y ha sido utilizada en otros experimentos como [15].

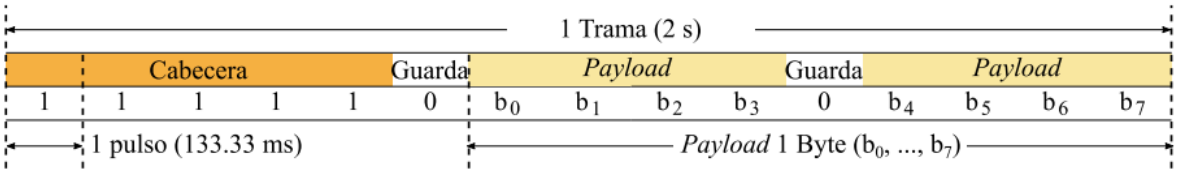

Figura 3.1.1: Trama de comunicaciones utilizada

3.2 Modulación

Como modulación se ha decidido utilizar la OOK. Debido a la trama utilizada, está modulación conlleva que el usuario note y perciba el efecto de parpadeo. No obstante, dado que este sistema está orientado a operar en redes de sensores, no se considera que las fuentes luminosas a utilizar vayan a servir como fuentes de iluminación. Es decir, en lugar de utilizarse grandes pantallas, los transmisores consistirán en LEDs o arrays de los mismos puntuales cuya función única será la de transmitir.

Si bien hubiera sido posible utilizar otras técnicas de modulación como UFSOOK o UPSOOK para reducir ese parpadeo, ello solo acarrearía una complejidad innecesaria al sistema. Además, aumentaría la probabilidad de error al comprobarse que algunas de las cámaras con las que se ha

experimentado tienen desviaciones elevadas en cuanto a tiempo entre captura de imágenes se refiere.

3.3 Transmisor

Para llevar a cabo las transmisiones se han utilizado dos elementos principales: dispositivos Arduino Nano 33 como plataformas de control, y LEDs blancos y RGB como fuentes luminosas.

En cuanto a los Arduino, se han seleccionado estos dispositivos por su amplia disponibilidad en el mercado y su flexibilidad de programación. Esta plataforma cuenta con su propio Entorno de Desarrollo Integrado (IDE - *Integrated Development Environment*) que proporciona diferentes herramientas para facilitar el desarrollo de código para este tipo de dispositivos. Si bien soportan lenguajes de programación como C, Wiring o Processing, normalmente para desarrollar en Arduino se utiliza el lenguaje de programación propio del mismo, basado en C++ y con muchas similitudes con el mismo [54].

El código en Arduino sigue una estructura preestablecida que se divide en dos partes principales: la función *setup*() y la función *loop*(). La función setup() se ejecuta una vez al principio del programa y se utiliza para configurar los pines de entrada/salida, establecer variables globales y realizar cualquier inicialización necesaria para que el programa funcione correctamente. Por su parte, la función loop() se ejecuta continuamente después de la función setup() y es la que ejecuta el comportamiento del programa. Es en este método donde se escriben las instrucciones para leer sensores, enviar datos, o cualquier otra acción que necesite realizar el programa.

En cada uno de los Arduinos se ha programado la transmisión de todas las posibles tramas de forma consecutiva e indefinida. Para ello, se comienza definiendo el PIN 6 como salida en la función *setup*, tal y como se puede observar la línea 4 en el Código 3.3.1. El polo positivo del LED irá conectado a este pin, y el negativo a cualquiera de las tomas a tierra presentes en los dispositivos.

Luego, en la función *loop*, se define un array de tipo *int* de 15 elementos llamado *trama* (línea 15) que representa la estructura de la trama a transmitir, que está formada por 15 bits. Inicialmente se asignan a este array los valores de la primera trama de la secuencia a transmitir. Es decir, una trama con payload igual a todo ceros. Luego, se lleva a cabo una cuenta decimal de 0 a 255. En cada una de las iteraciones de esta cuenta se realiza la conversión del número a formato binario (línea 15), luego se asigna ese valor a los espacios asociados al payload de la trama (posición 6 - 9 para la primera mitad de byte, y posición 11 - 14 para la segunda mitad), y se transmite la trama respetando los tiempos

asignados a cada bit (133.33 ms). El diagrama de flujo de este código se puede observar en la Figura 3.3.1.

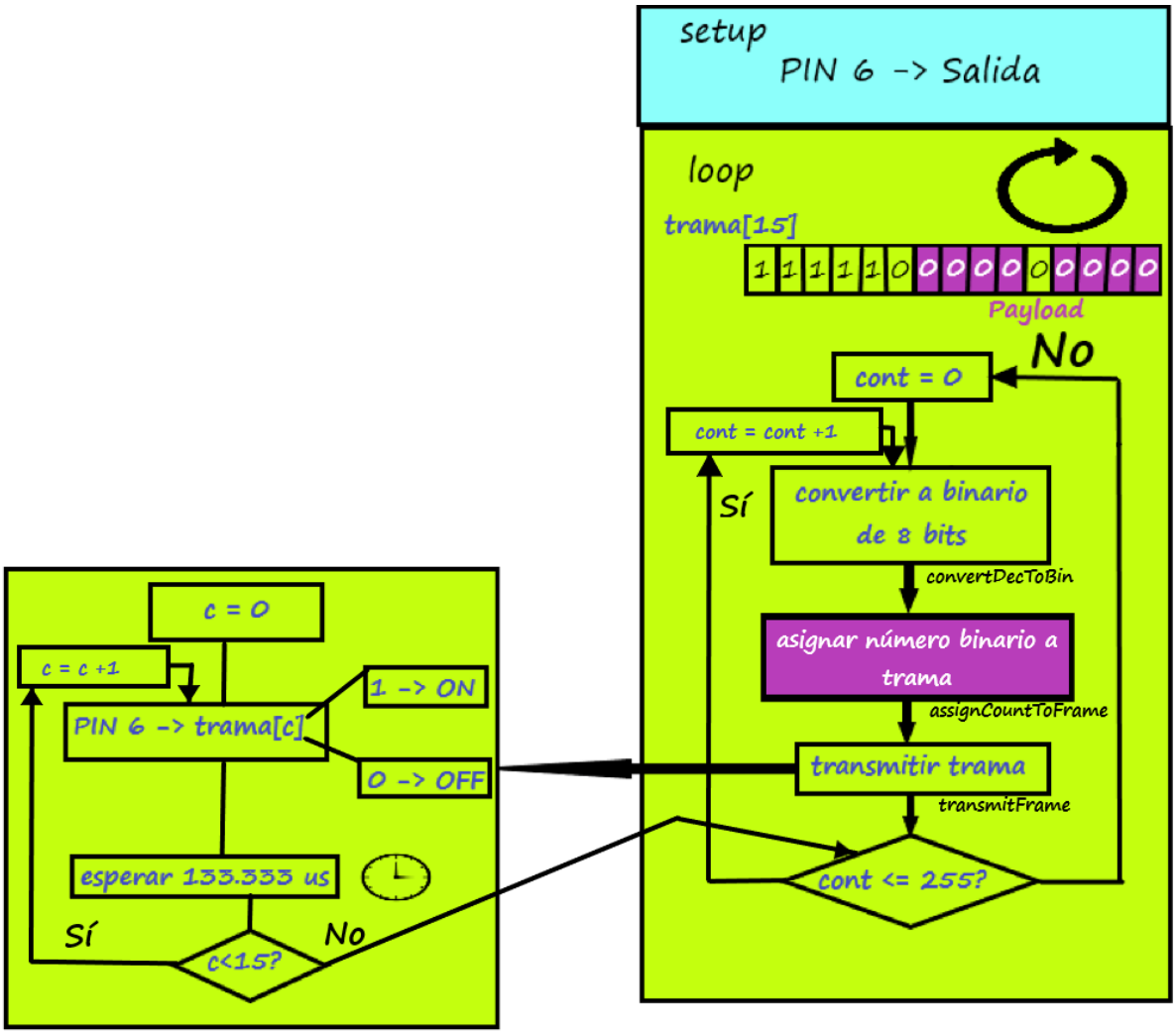

Figura 3.3.1: Diagrama de flujo del código del transmisor

Para simplificar el entendimiento de las tareas a realizar en la función *loop* se han definido tres métodos:

- *convertDecToBin*: Recibe por parámetro un entero denominado *decimal* y devuelve el puntero a un array de tipo int de 8 elementos denominado *binaryCont*. Dicho array contiene la conversión a binario del número en formato decimal pasado por parámetro. Para realizar esa conversión se usa un bucle *for* en el que, en cada iteración, se desplazan los bits del número decimal hacia la derecha utilizando el operador de desplazamiento a la derecha (>>) con un desplazamiento *i*. Este desplazamiento equivale a dividir el número decimal entre 2 elevado a *i*. Luego, sobre el resultado de ese desplazamiento se aplica la operación lógica AND (&) con un 1 para obtener el valor del bit más a la derecha después del desplazamiento. Si el resultado

de la operación es un 1, significa que el resultado de la división es un número impar y se asigna 1 al elemento correspondiente del array binaryCont[i]; de lo contrario, significa que el resultado de la división es un número par y se asigna 0. Este proceso de conversión se conoce como *método de la escalera* y se puede entender gráficamente mediante la Figura 3.3.2.

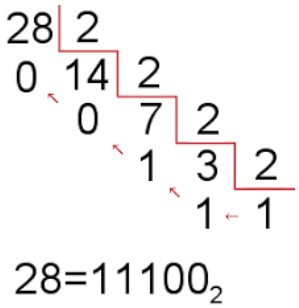

Figura 3.3.2: Ejemplo de conversión de decimal a binario

- *assignCountToFrame*: Recibe por parámetro el puntero al array que contiene el contador en formato binario (*BinaryCont*), así como el puntero al array que contiene la estructura de la trama a transmitir (*trama*). Esta función se encarga de asignar los distintos valores del contador en formato binario, a las posiciones apropiadas de la trama para que transmita esa información como *payload* de la misma. Como se puede observar en la Figura 3.3.3, estas posiciones se corresponden con la 6 - 9 y la 11 - 14. Dado que el resultado de la conversión a binario está invertido, se asignan índices del *binaryCont* más bajos a los índices más altos asociados al payload de *trama*.

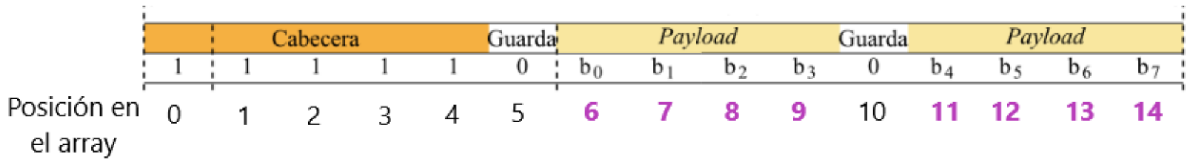

Figura 3.3.3: Posiciones de la trama

- *transmitFrame*: Este método recibe por parámetro el puntero al array *trama*. Su función es recorrer todas las posiciones de ese array pasado por parámetro y encender o apagar el LED (línea 46 del Código 3.3.1), en función de si el bit se corresponde con un 1 o un 0. Cada vez que se representa un bit se esperan los 133.33 ms estipulados para cada bit mediante la función *delayMicroseconds* para conseguir una mayor precisión.

```
1   /*
2   Codigo para transmitir con el Arduino todas las tramas posibles
3   */
4   const int TRANSMIT_PIN = 6;
5   /* Configuracion del PIN que se usara como transmisor*/
6   void setup(){
```

```
7       pinMode(TRANSMIT_PIN, OUTPUT);
8     }
9
10    void loop(){
11      int trama[15] = {1,1,1,1,1,0,0,0,0,0,0,0,0,0,0};
12
13      /* Contador de 0 a 255*/
14      for(int cont=0; cont<=255;cont++){
15        int* binaryCont = convertDecToBin(cont);
16        assignCountToFrame(binaryCont, trama);
17        transmitFrame(trama);
18      }
19    }
20
21    /* Conversion del contador a binario*/
22    int* convertDecToBin(int decimal){
23      static int binaryCont[8];
24      for(int i=7; i>=0;i--){
25        if(decimal >> i & 1){
26          binaryCont[i] = 1;
27        }else {
28          binaryCont[i] = 0;
29        }
30      }
31      return binaryCont;
32    }
33
34    /* Asignacion del Payload a la trama*/
35    void assignCountToFrame(int* binaryCont, int* trama){
36      trama[6] = binaryCont[7];
37      trama[7] = binaryCont[6];
38      trama[8] = binaryCont[5];
39      trama[9] = binaryCont[4];
40      trama[11] = binaryCont[3];
41      trama[12] = binaryCont[2];
42      trama[13] = binaryCont[1];
43      trama[14] = binaryCont[0];
44    }
45    /* Alternar estado del LED para transmitir la trama*/
46    void transmitFrame(int* trama){
47      for(int c=0; c<15;c++){
48        digitalWrite(TRANSMIT_PIN, trama[c]);
49        delayMicroseconds(133333); // 133.333 ms
50      }
51    }
```

Código 3.3.1: Transmisión de todas las posibles tramas

Como fuentes luminosas se han utilizado dos tipos de LED: blancos y RGB. Como LED blanco se ha utilizado el que tiene por número de referencia C513A-WSN-CY0Z0232 del fabricante Cree. Tiene un tamaño de 5 mm y debe ser alimentado con una tensión de 3.2 V (máximo 4 V) y una corriente de 20

mA. En estas condiciones, con una temperatura de 25 grados, es capaz de proporcionar una intensidad luminosa de unos 5 cd. Tiene un montaje en orificio pasante con dos pines, lo que facilita su conexión a la placa Arduino.

Cómo LED RGB se ha empleado el número de referencia L-154A4SURKQBDZGW del fabricante Kingbright. Tiene un tamaño de 5 mm y un montaje en orificio pasante con cuatro pines. Uno de los mismos se corresponde con el polo negativo y cada uno de los otros permite seleccionar el componente R, G o B del LED. Opera con una tensión de 3.3 V (máximo 4.1 V) y una corriente de 20 mA. La componente roja opera típicamente en la longitud de onda de 630 nm, la verde en la de 525 nm y la azul en 465 nm. La intensidad luminosa asociada respectivamente es de 700 mcd, 1300 mcd, y 300 mcd.

En total se usaron 4 Arduinos Nano 33, 4 LEDs RGB y uno blanco. Este tipo de Arduino proporciona por defecto una tensión de salida en sus pines de 3.3 (V) y una corriente máxima de 15 mA. En los distintos escenarios se probaron distintas combinaciones de colores para poder evaluar cómo se comportaba el sistema para cada longitud de onda.

En último lugar, cabe destacar que para alimentar las distintas placas de Arduino se usaron baterías recargables Li-ion, con referencia 2347-4003-520 del fabricante ANSMANN. Tienen un voltaje nominal de 3.6 V y una capacidad nominal de 26000 mAh, lo que es más que suficiente para mantener las transmisiones durante los escenarios de prueba que se definirán más adelante en este documento. Una muestra gráfica de estos transmisores se puede observar en la Figura 3.3.4. Como se puede apreciar, como base de las plataformas Arduino se han usado protoboards.

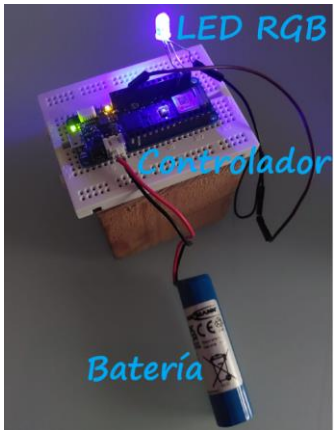

Figura 3.3.4: Transmisor usando LED RGB

3.4 Receptor

Para recibir la información y aplicar sobre la misma el algoritmo de detección se utilizaron tres elementos: el ordenador de placa única Raspberry Pi versión 3, un ordenador portátil Acer, y un disco duro externo de 1 TB de la marca Toshiba. Esto se debe a que, para poder analizar cuantas veces se requiera las comunicaciones, no se va a trabajar con las mismas en tiempo real, sino que se van a grabar y posteriormente postprocesar. Además, ello permite tener evidencias y poder corroborar el trabajo realizado.

Como elemento grabador se utilizó la Raspberry Pi v3 por su amplia disponibilidad en el mercado, y su fácil movilidad, configuración y programación. Asimismo, se disponía en el laboratorio de un módulo de cámara V2 de altas prestaciones, de *displays* para Raspberry Pi v3, y de soportes fabricados con impresora 3D que agilizaban y facilitaban aún más su uso y transporte. Para alimentar este dispositivo se utilizaron dos medios: las tomas de corriente cuando estas eran fácilmente accesibles, y una batería externa del fabricante Varta con número de referencia 57976 101 111 cuando las tomas de corriente no eran cercanas.

Por su parte, el ordenador se utilizó para aplicar el algoritmo desarrollado sobre los vídeos grabados y obtener tanto los píxeles asociados a cada transmisor como la información enviada por los mismos. Para ejecutar dicho código se utilizó el mismo IDE que se empleó en su desarrollo: Visual Studio Code. No se decidió utilizar la Raspberry Pi v3 para aplicar el algoritmo de detección sobre los vídeos porque sus prestaciones no lo permitieron. De igual modo, sus capacidades de almacenamiento no eran suficientes para mantener las grabaciones de vídeo, por lo que durante las mismas fue necesario usar el disco duro externo anteriormente comentado. Las características principales de los tres elementos comentados, así como de la cámara utilizada con la Raspberry se resumen en la Tabla 3.4.1. Asimismo, el conjunto utilizado para grabar y almacenar los vídeos (Raspberry Pi v3, teclado, ratón, memoria externa y enchufe) se puede apreciar en la Figura 3.4.1.

Figura 3.4.1: Equipo de grabación de las comunicaciones

Raspberry Pi	Modelo	Raspberry Pi 3 Model B
	Fabricante	Raspberry Pi Foundation
	Sistema Operativo	Raspian - 32 bits
	Procesador	Broadcom BCM2837 SoC (System-on-a-Chip) con CPU ARM Cortex-A53 de 64 bits de cuatro núcleos a 1.2 GHz
	Almacenamiento	1 GB de memoria RAM
	Alimentación	5 V , 2.5 A
	Conectividad inalámbrica	Wi-Fi 802.11n y Bluetooth 4.2
Cámara Módulo V2	Modelo	IMX219 - V2 module
	Fabricante	Raspberry Pi Foundation
	Modo de adquisición	Rolling Shutter
	Resolución máxima	3280x2464
	Ganancia máxima	16 B
	Frame rate máximo	30 FPS
Memoria Externa	Capacidad	1.000 GB
	Velocidad de transferencia	5.000 Mbits/s
	Fabricante	Toshiba
Ordenador portátil	Modelo	Acer Aspire 3 A315-34
	Procesador	Intel Celeron N4020/8, 2 núcleos, 1.1 GHz
	Memoria	8 GB DDR4-SDRAM, 256 GB SSD

Tabla 3.4.1: Características de los principales medios materiales usados para la recepción

3.4.1 Configuración de la Raspberry Pi v3

Para poner operativa la Raspberry Pi v3, lo primero fue conectar a la misma el display y la cámara mediante los conectores en formato de banda y los puertos adecuados. Luego, se debió insertar en la

misma la tarjeta SD con el sistema operativo Raspbian - 32 bits, y encajar y atornillar la placa a su carcasa, fabricada en las instalaciones del IDeTIC mediante la impresora 3D y la cortadora de metacrilato.

Una vez hecho esto, se configuró la fecha y hora de la Raspberry mediante el comando *sudo date -s "YYYY-MM-DD HH:MM:SS"*. Sin este ajuste no hubiera sido posible usar la conexión a Internet. Luego, se habilitó la cámara accediendo a la ventana de comandos, y ejecutando *raspi-config*. Tras esto, se seleccionó *Interfacing Options* (Figura 3.4.1.1)*, P1 Camera* (Figura 3.4.1.2), *Yes* y *Finish*. Una vez hecho esto, se reinició la Raspberry para que se efectuaran los cambios.

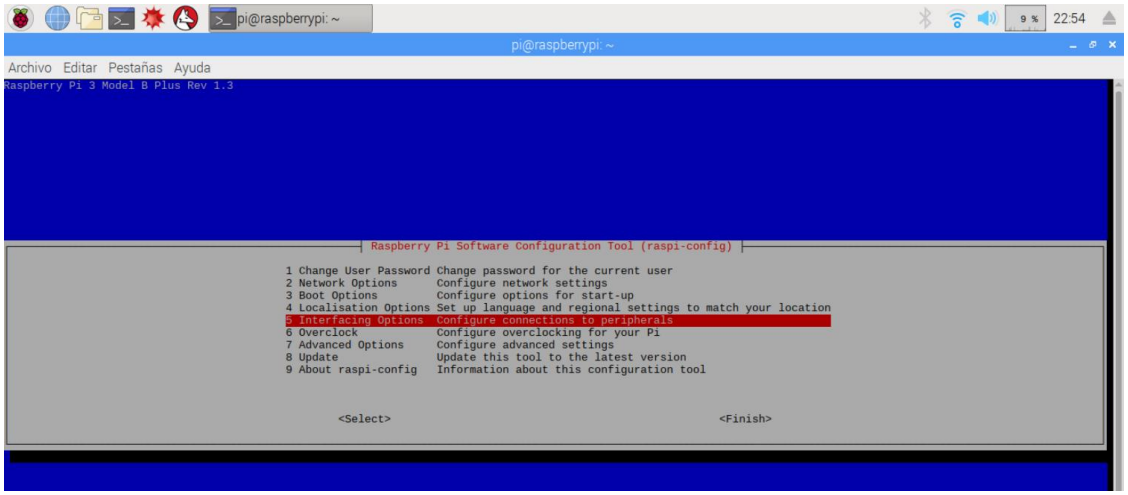

Figura 3.4.1.1: Habilitación de la cámara (I)

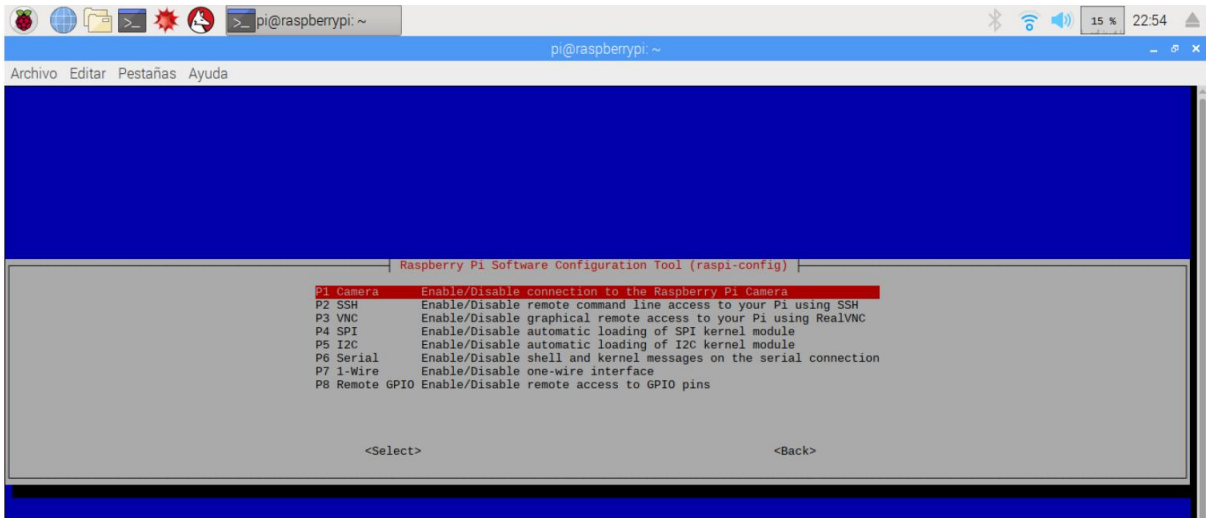

Figura 3.4.1.2: Habilitación de la cámara (II)

3.4.1.1 Programación de la grabación

Para grabar el vídeo en la Raspberry Pi se utilizó el lenguaje Python 3 dada la elevada cantidad de librerías del mismo que facilitan la operación con este tipo de dispositivos. Además, por defecto Python 3 viene incluido en el sistema operativo Raspbian, lo que se puede comprobar ejecutando el comando *python3 –version*. Por su parte, como entorno de desarrollo se decidió utilizar Tommy, también instalado por defecto en el sistema operativo de Raspberry Pi.

Como librerías, sólo se utilizó *picamera*, para poder detectar si existe algún elemento de tipo cámara conectado a la Raspberry, y la librería *time*, para poder establecer esperas durante tiempos determinados. La biblioteca *time* viene instalada por defecto, pero para añadir *picamera* es necesario ejecutar el comando *sudo apt-get install python-picamera*. Previamente a la ejecución de ese comando es recomendable ejecutar *sudo apt update* para actualizar la lista de paquetes disponibles y *sudo apt upgrade* para descargar e instalar las actualizaciones disponibles de los paquetes ya instalados en el sistema.

El software desarrollado para realizar la grabación se puede observar en el Código 3.4.1.1.1. En el mismo, una vez definidas las propiedades de configuración del vídeo (línea 4 - línea 7), se inicializa la cámara. Para ello es necesario crear una instancia de *Picamera*, que se asigna a la variable *camera* (línea 11). Luego, se configuran la resolución, los FPS, el tiempo de exposición y la sensibilidad de la cámara (línea 12 - línea 17). Dado que realizar estos cambios requiere un cierto tiempo, se establece una espera de 4 segundos antes de iniciar el proceso de captura (línea 21).

Una vez transcurrido el tiempo de espera, se establece el nombre (*video*) y el formato del vídeo deseado, y se inicia la grabación (líneas 24 - 25). Finalmente, se configura la duración del vídeo (línea 28) y que tras pasar ese tiempo se pare la grabación (línea 30).

```
1   # Codigo para grabar las comunicaciones con la Raspberry Pi
2   import time
3   import picamera
4
5   # Definir las propiedades del video
6   fps = 30
7   exposure_time = 20000 # en microsegundos, max 30000
8   frame_size = (1920, 1088) # tamaño del fotograma (ancho, alto)
9
10  # Inicializar la cámara
11  with picamera.PiCamera() as camera:
12    camera = picamera.PiCamera()
13    camera.resolution = frame_size
14    camera.framerate = fps
```

```
15
16      # Configurar el tiempo de exposición fijo
17      camera.shutter_speed = exposure_time
18      camera.iso = 100 # sensibilidad, con baja luz se sube, con mucha luz se baja
19
20      # Esperar a que se estabilice la exposición
21      time.sleep(4)
22
23      # Crear el objeto de codificación de video
24      video_file = 'video.mjpeg'
25      camera.start_recording(video_file)
26
27      # Esperar la duración del video
28      camera.wait_recording(560)
29
30      # Finalizar la grabación de video
31      camera.stop_recording()
```

Código 3.4.1.1.1: Software de grabación de las comunicaciones

3.4.1.2 Configuración de la grabación

Como se puede observar en el Código 3.4.1.1.1, como FPS se decidió utilizar el máximo que permite la cámara, 30, para tomar la mayor cantidad de muestras posibles de cada trama. Por su parte, como resolución de la cámara (cantidad de píxeles que conforman el sensor de imagen) se usó un valor menor al máximo que permite, concretamente la resolución asociada a una cámara Full HD (*High Definition*), 1920x1080. Se tomó esta decisión para verificar la funcionalidad del sistema con cámaras de prestaciones usuales.

En cuanto al tiempo de exposición, este parámetro es el período de tiempo durante el cual el obturador de la cámara está abierto y el sensor de imagen está expuesto a la luz que ingresa a la cámara. Usar un tiempo de exposición bajo permite disminuir la cantidad de luz que entra al sensor de imagen, por lo que es recomendable para condiciones de alta luminosidad. Del mismo modo, para condiciones de baja luminosidad permite aumentar la cantidad de luz que entra al sensor de imagen.

En algunos de los escenarios sobre los que se probó el algoritmo se tomaron varios vídeos en lugar de uno, usando en cada uno de los mismos un tiempo de exposición diferente. Esto se realizó para poder evaluar el comportamiento del sistema al variar las condiciones de luminosidad de forma artificial, algo que puede llevarse a cabo frecuentemente, por ejemplo en cámaras de seguridad, para aumentar la nitidez de la imagen. Estos valores de tiempo de exposición no fueron los mismos para los distintos escenarios de prueba, ya que en los mismos existieron a su vez distintas condiciones de luminosidad.

En todos los casos se debió cumplir que el tiempo de exposición máximo fuera igual o inferior a 32.000 µs, dado que al usar una tasa de 30 FPS el tiempo entre capturas es de 33.333 µs. Si se usase un tiempo de exposición mayor, se reduciría la tasa de FPS.

En cuanto a la sensibilidad de la cámara, es una medida de la capacidad del sensor de imagen para capturar luz. Un valor de sensibilidad alto permite aumentar la cantidad de fotoelectrones captada, y uno bajo disminuirla. Establecer valores muy elevados puede introducir ruido y distorsiones. Al igual que ocurría con el tiempo de exposición, estos valores se modificaron en función de la intensidad lumínica del escenario y en función del tiempo de exposición de forma manual, observando qué valores permitían obtener los resultados de vídeo deseados.

Como formato de vídeo se decidió utilizar *Motion Joint Photographic Experts Group* (MJPEG) ya que aplica la compresión de forma individual (sin tener en cuenta la redundancia temporal) a cada imagen teniendo en cuenta la redundancia entre cuadros consecutivos (redundancia espacial). La mayoría de formatos de vídeo aplican, además de la redundancia espacial, la redundancia temporal, que es aquella que comprime en función de valores similares en el tiempo de un píxel. Esto no es deseado desde la perspectiva de la detección porque se puede omitir variaciones pequeñas pero significativas de la intensidad luminosa. Sin embargo, la redundancia espacial no tiene un efecto tan notorio porque las fuentes luminosas se intentan situar en sitios con poca luminosidad ambiente para reducir el ruido en las comunicaciones. Si bien lo ideal sería grabar las imágenes sin ningún tipo de compresión, esto afecta enormemente tanto a los requisitos de almacenamiento como a los tiempos de ejecución del algoritmo. Por ello, entre las opciones de grabación posibles en la Raspberry Pi v3 se ha elegido la que menos puede afectar a las comunicaciones.

Por último, como duración del vídeo se especificaron 560 segundos para asegurar que se detecten todas las posibles tramas transmitidas por los Arduinos; hay 256 tramas posibles, y estas tienen una duración de 2 segundos cada una.

3.4.2 Almacenamiento de los datos

Dado que en el algoritmo a desarrollar no se va a trabajar con el vídeo como tal, sino con la secuencia de imágenes grabadas, resulta fundamental almacenar en el disco duro estas mismas para no realizar el proceso de extracción de imágenes de forma constante. De esta forma se ahorran tiempos en la fase de evaluación.

Para extraer las imágenes de los vídeos se usó el Código 3.4.2.1. En el mismo se carga el vídeo creando una instancia de la clase *VideoCapture* del módulo *cv2*. Luego, se extraen una a una las imágenes del

vídeo mediante la función *read*, que tiene la clase VideoCapture, y se almacenan como imágenes en formato JPG (*Joint Photographic Experts Group*) mediante la función de cv2 *imwrite*. Para mantener el orden en la que son extraídas en el almacenamiento, a la par que se lleva a cabo la extracción se incrementa un contador que se va asignando como parte del nombre con el que las imágenes son almacenadas. La extracción se realiza hasta que el parámetro *success* que devuelve la función *read* da como valor *False*, lo que indica que se ha producido algún error, como que no hay más imágenes que extraer. Finalmente se presenta en el Shell la cantidad de imágenes extraídas, que debería coincidir con la multiplicación de la cantidad de segundos grabados con la tasa de FPS (30). Este código fue ejecutado en el ordenador, haciendo uso del disco externo para el almacenamiento, debido a las limitaciones de memoria de la Raspberry Pi.

```
1   # Codigo que recupera todos los fotogramas que componen un video
2   import cv2 as cv
3   import time
4
5   cap= cv.VideoCapture('outdoorhigh/video.mjpeg')
6
7   # Definimos contador
8   count = 0
9
10  # Se extrae primer fotograma y se comprueba si la extraccion ha sido
11  correcta
12  success, image = cap.read()
13
14  while success:
15
16      # Guarda el fotograma con indice "count" en el directorio especificado
17      cv.imwrite("outdoorhigh/frame%d.jpg" % count, image)
18
19      # Se extrae fotograma
20      success, image = cap.read()
21
22      count += 1
23
    print("La cantidad total de imagenes en el video es de ", count)
```

Código 3.4.2.1: Software de extracción y almacenamiento de imágenes de un vídeo

3.5 Escenarios

A modo de resumen de los apartados anteriores, el sistema de comunicaciones consiste por una parte, en el envío de tramas de 2 segundos por parte de dispositivos Arduinos alimentados por baterías de Li-ion que, mediante el encendido y apagado de los LEDs (modulación OOK), envían la información. La

parte receptora por su parte consiste en la grabación de 10 minutos de esas comunicaciones mediante el uso de una Raspberry Pi y un disco duro externo, y en la aplicación del algoritmo de detección a esas grabaciones mediante ordenador. Un diagrama resumen se puede observar en la Figura 3.5.1.

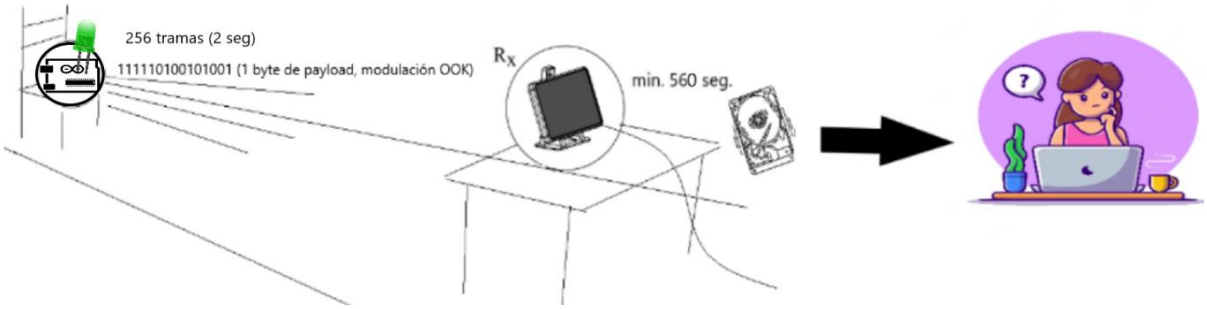

Figura 3.5.1: Esquema del sistema de comunicaciones

Este sistema ahora comentado ha sido implementado en tres escenarios distintos: uno *indoor* y dos *outdoor*. El escenario indoor se corresponde con el pasillo de la planta 2 del edificio Polivalente II del Parque Científico Tecnológico de la Universidad de Las Palmas de Gran Canaria (ULPGC). En total se grabaron tres vídeos en el mismo: uno con un tiempo de exposición de 9 ms, otro con un tiempo de exposición de 14 ms y otro con un tiempo de exposición 19 ms. La sensibilidad ISO (*International Organization for Standardization*) en los tres casos fue de 100. Las grabaciones fueron realizadas en torno a las 12 del mediodía, y en las mismas destaca la presencia de varios objetos reflectantes (suelo y paredes) y la presencia de obstáculos (escalera y personas). En este caso, la intensidad lumínica del entorno no era elevada al tratarse de un escenario *indoor*. Los transmisores se ubicaron sobre sillas en un extremo del pasillo, y el receptor se colocó sobre otra silla en el otro extremo. La distancia aproximada del enlace fue de 32 metros. En cuanto a los colores utilizados, dos transmisores trabajaron con el color verde, uno con el azul y otro con el blanco. Un esquema de este escenario se puede apreciar en la Figura 3.5.2.

Figura 3.5.2: Escenario indoor

En cuanto a los escenarios *outdoor*, en uno de los casos las medidas fueron tomadas durante el día, en torno a las 4 de la tarde, y en el otro caso de noche, en torno a las 10.

En cuanto al escenario diurno, se ubica en la azotea no cubierta de una casa, sin objetos reflectantes, ni obstáculos presentes. Los transmisores se ubicaron en un extremo de la misma, y el receptor en el lado contrario. En esta ocasión uno de los transmisores trabajó con el color rojo, dos con el azul y uno con el blanco. La distancia entre el receptor y cada uno de los transmisores es similar y es de unos 8 metros. En este caso se trabajó con una sensibilidad ISO de 40 y un tiempo de exposición de 0,8 y 0,45 ms. Es decir, en este caso se grabaron dos vídeos.

En cuanto al escenario nocturno, se ubica en una colina ascendente, sin obstáculos ni objetos reflectantes. En este caso los transmisores se ubicaron a distintas distancias del receptor, colocado en la parte baja de la montaña. Todos ellos trabajaron con el color azul dado que eran los que más permitían distinguirlos a simple vista, para garantizar la detección de los mismos. La distancia de los enlaces osciló entre los 65 y los 72 metros, y se trabajó con una sensibilidad ISO de 500 y un tiempo de exposición de 32 ms. En este caso solo se grabó un vídeo con un único valor de tiempo de exposición y sensibilidad dado que la variación de estos parámetros no produjo cambios notorios. Una imagen de los dos escenarios *outdoor* comentados se puede observar en la Figura 3.5.3. En el caso del escenario diurno, se marcan con un círculo rojo los transmisores.

Figura 3.5.3: Escenario outdoor diurno a la izquierda y nocturno a la derecha

Capítulo 4

4. Descripción del algoritmo

En este capítulo de la memoria se realiza una descripción del algoritmo de detección de transmisores sub-píxel elaborado. Para ello primero se estudian los distintos aspectos teóricos considerados de forma previa a su desarrollo, o que han surgido tras modificaciones del mismo en el análisis de resultados. Tras ello, se describe de forma general la funcionalidad y forma de operación del algoritmo. Finalmente, se describen con un mayor grado de complejidad técnica las distintas partes que componen el diseño final.

4.1 Análisis preliminar

Para llevar a cabo la detección de los transmisores, es necesario tener en cuenta lo que diferencia a cualquier píxel o conjunto de píxeles de la imagen de aquel en el que está presente la información transmitida por el LED. Los tres principales elementos diferenciadores son la luminosidad del píxel, que suele ser considerablemente superior a la media de todos los píxeles de la imagen, el tamaño de la agrupación de píxeles que tiene esa luminosidad, que suele ser reducido (sobretodo en la casuística sub-píxel), y la variabilidad de esta luminosidad en intervalos con una duración teórica máxima y mínima.

4.1.1 Luminosidad

La luminosidad es determinante a la hora de filtrar qué píxeles pueden ser potenciales transmisores. Si se establece un valor umbral de luminosidad bajo, mayor cantidad de píxeles serán considerados potenciales transmisores, y consecuentemente aumentará el tiempo de procesamiento y, probablemente, la tasa de falsos positivos. Por el contrario, si se establece un valor umbral de luminosidad elevado, aquellos transmisores cuya luminosidad sea baja en comparación con la luz ambiente serán ignorados. El valor de luminosidad depende directamente de las características de los transmisores y, sobre todo, de las condiciones lumínicas del escenario considerado.

En un escenario nocturno, la cantidad de luz que se recibe de un transmisor será mayor, ya que se aumenta la sensibilidad del sensor y el tiempo de exposición para mejorar la calidad de imagen. No obstante, durante el día ocurrirá todo lo contrario, se recibirá más luz del entorno y, para evitar la sobreexposición de los sensores de imagen y poder tener una imagen nítida, se reducirá la sensibilidad y el tiempo de exposición, lo que hará que se reciba menos cantidad de luz de los transmisores.

Para modelar este comportamiento se analizaron los valores medios de luminosidad de al menos tres imágenes de cada vídeo tomado y los valores medios de luminosidad del transmisor que presentaba peores características en cada caso. Es decir, se obtuvo la luminosidad media del escenario y el umbral máximo de luminosidad que se podía establecer para poder detectar incluso al transmisor con peor comportamiento.

Para obtener estos valores se desarrolló un script que permite obtener una señal óptica de 4 segundos de aquel píxel sobre el que se haya clicado con el ratón en una de las imágenes del vídeo bajo estudio. Adicionalmente, este script también obtiene el valor promediado de las componentes R, G y B (por separado) tanto del píxel central como de sus vecinos con un desplazamiento igual a 2 a izquierda, derecha, arriba y abajo. Esto se realiza para poder determinar el color del LED y la distorsión que experimenta la señal conforme nos alejamos del píxel central de la ROI. Además, el script también permite obtener la luminosidad media de dicha imagen. Este software se explica y presenta en el Anexo A, en el apartado A.1 del mismo, y fue utilizado también durante la fase de resultados.

Un ejemplo de la ejecución de este código se observa en la Figura 4.1.1.1. Como se puede apreciar, la luminosidad de los píxeles vecinos es considerablemente menor que la del píxel central, lo que provoca que las componentes RGB tengan una amplitud mucho menor que la componente de luminosidad del píxel central tras la imagen ser convertida a escala de grises. Esta conversión sigue la Ecuación I de acuerdo con la documentación de cv2 [55], donde R es el valor de luminosidad de la componente roja, G de la componente verde, B de la componente azul, e Y el resultado de la conversión a escala de grises.

$$Y = 0.299*R + 0.587*G + 0.114*B \quad (I)$$

Por otra parte, en la Figura 4.1.1.1 también se observa que aplicar el promediado con píxeles vecinos suaviza la señal y que las diferencias entre las distintas componentes son más o menos notorias dependiendo del color, especialmente cuando se trata de un LED verde, debido a la Ecuación I. A la izquierda se aprecia notablemente con mayor amplitud la componente G, ya que se corresponde con un LED verde, y a la derecha se aprecia ligeramente más la componente B, ya que se trata de un LED

azul. Asimismo, también cabe destacar la diferencia de luminosidad del píxel central en ambos transmisores: el verde oscila en los 180, y el azul en los 160.

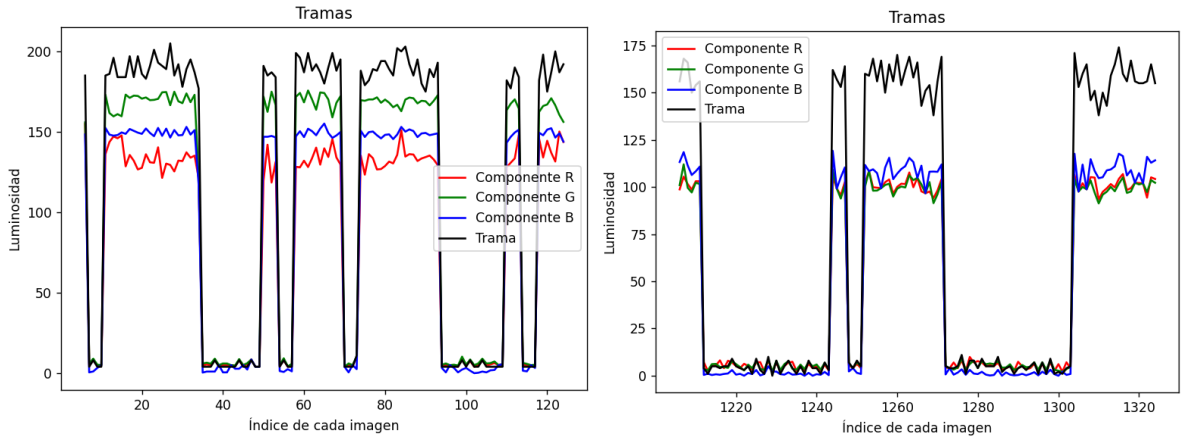

Figura 4.1.1.1: Ejecución del Código A.1 sobre imágenes del escenario indoor con mínimo tiempo de exposición, y selección de píxel asociado a LED verde (izquierda) y azul (derecha)

Tras obtenerse por cada escenario el valor medio de luminosidad del LED con peor comportamiento, y la luminosidad media de cada imagen en escala de grises, se observó que, en valores muy bajos de la luminosidad media de la imagen, la intensidad del LED es hasta 80 veces superior. Por su parte, para medias muy elevadas, esta cifra disminuye exponencialmente y la luminosidad del LED llega incluso a 0,7 la luminosidad media. No obstante, para valores intermedios de luminosidad media, el valor de umbral máximo no siempre es inversamente proporcional a la luminosidad media de la imagen.

Para conseguir un mayor grado de precisión, se calculó la expresión matemática que describe la relación entre la luminosidad media de la imagen, y el umbral máximo permisible en cada caso. Para ello se desarrolló otro script de Python que se presenta y comenta en el apartado A.2 del Anexo A. En este script se utiliza un conjunto de puntos que describen los umbrales máximos de luminosidad para cada luminosidad media (datos obtenidos mediante el Código A.1.1 y los vídeos en los diferentes escenarios y tiempos de exposición) para obtener una función polinómica. Así, en dicha función “x” representa el valor de luminosidad media de una imagen, e “y” el valor de luminosidad máxima que se debería establecer como umbral en esa imagen para que se pudieran detectar todos los transmisores.

La función polinómica inicialmente obtenida provocaba que para algunos puntos no se respetase su umbral máximo (curva toma valores en “y” superiores a los que toman los puntos obtenidos en la práctica). Si se establece un umbral máximo superior al máximo obtenido en la práctica, el LED con peores condiciones de luminosidad no se detectaría en los escenarios con esa luminosidad media.

Para solucionarlo, se decide reducir levemente algunos de los valores en “y” obtenidos en la práctica y utilizar un grado polinómico igual a 5, puesto que ajustaba mejor la función. Asimismo, también fue necesario modificar la señal a partir de valores de “x” igual a 195, ya que, si no se alcanzarían valores de umbral máximo ilógicos, tales como valores superiores a 255. Las condiciones de luminosidad media y umbral máximo para cada escenario utilizados finalmente se pueden apreciar en la Tabla 4.1.1, mientras que la función polinómica final y su representación gráfica se pueden observar en la Ecuación II y la Figura 4.1.1.2 respectivamente.

Valores de x (luminosidad media)	2,4	17,5	81,42	97,3	146	193,5
Valores de y (umbrales máximos de luminosidad)	195	85	165	215	85	128

Tabla 4.1.1.1: Valores de luminosidad media de la imagen y correspondientes umbrales máximos de luminosidad permisibles para poder detectar todos los transmisores

$$(5.503e-08)x^5-(2.038e-05)x^4+(0.001936)x^3+0.01908x^2-5.868x+202.6, si\ x<195$$
$$195, si\ x\geq 195\ \ (II)$$

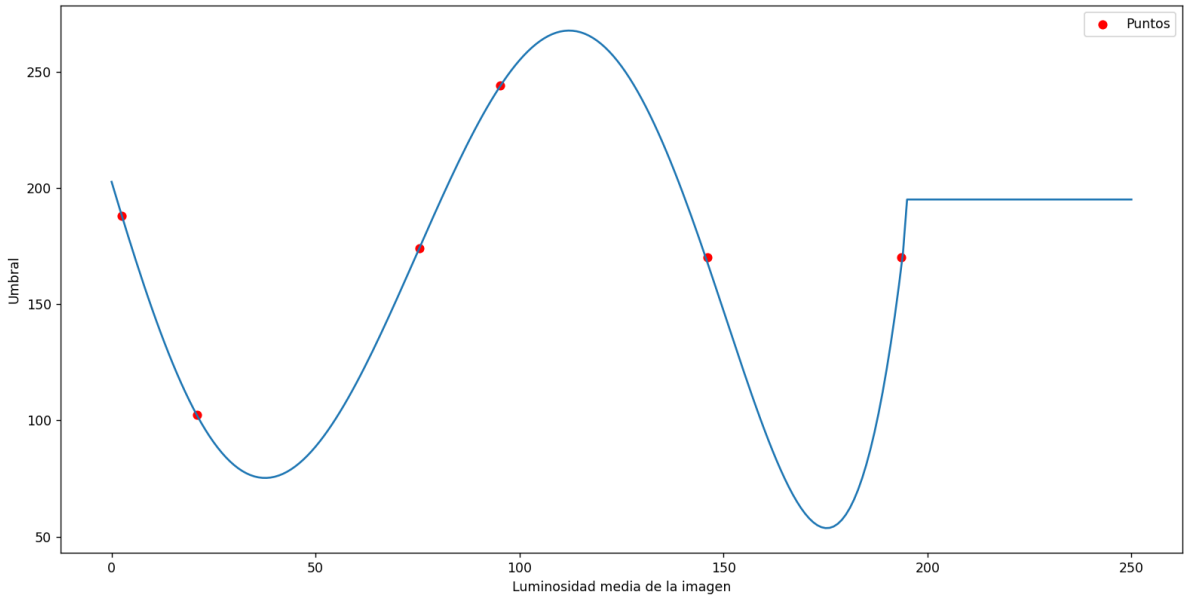

Figura 4.1.1.2: Función polinómica discontinua que representa la relación entre la luminosidad media de la imagen y el umbral máximo de luminosidad de la misma, junto a los puntos a partir de los que se ha elaborado el ajuste polinómico

4.1.2 Tamaño de las regiones luminosas

Otro parámetro útil a la hora de distinguir posibles transmisores de otros elementos es el tamaño de la región luminosa bajo estudio. Es decir, el tamaño del conjunto de píxeles que tienen una intensidad luminosa semejante. Al tratarse específicamente en este trabajo la temática sub-píxel, se puede adelantar que el tamaño de esas regiones será pequeño en comparación con la resolución de la imagen (como se comentó anteriormente, Full HD).

Este parámetro depende de cuatro factores. El tamaño de la fuente luminosa, que como se comentó anteriormente es de 5 mm, la distancia del receptor a la misma, la PSF de la cámara, la potencia lumínica de la fuente, y sobre todo, nuevamente, la luminosidad media del entorno. En función de este último parámetro una fuente de un mismo tamaño y potencia lumínica, y a una misma distancia, puede ocupar regiones de 80 píxeles o ser imperceptible.

Para encontrar la relación entre la región máxima que pueden ocupar los transmisores utilizados, y la luminosidad media de la imagen, se siguió un proceso similar al apartado anterior. En primer lugar, se desarrolló un script que permitiera obtener el tamaño del área luminosa de cada región seleccionada con el ratón. Este script se explica en el apartado A.3 del Anexo A, y sigue una estructura similar al Código A.1.1. Luego, se obtuvo la región máxima de cada escenario y los valores de luminosidad media en cada caso. Finalmente, usando el Código A.2.1, se obtuvo la relación polinómica de esos dos parámetros. Es decir, la relación entre luminosidad media y tamaño de luminosidad máxima.

A mayor tamaño de región máxima, más posibles transmisores se consideran y por tanto mayor es el tiempo de procesamiento y la cantidad de posibles falsos positivos. A menor tamaño de región máxima, mayor es la probabilidad de que se omita algún positivo porque su potencia lumínica sea muy elevada en comparación con la luminosidad media de la imagen, o debido a la PSF de la cámara.

Tras la obtención de las distintas regiones máximas del total de transmisores en cada escenario, y relacionarlos con la luminosidad en cada caso, se obtuvo la Tabla 4.1.2.1 y la Ecuación III, que se representa como la Figura 4.1.2.1.

Valores de x (luminosidad media)	2,4	17,5	81,42	97,3	146	193,5
Valores de y (umbrales mínimos de tamaño de región)	130	24	57	38	26	30

Tabla 4.1.2.1: Valores de luminosidad media de la imagen y correspondientes umbrales máximos de tamaño de región luminosa para poder detectar todos los transmisores

$$(3.649e-10)x^6 - (2.628e-07)x^5 + (7.255e-05)x^4 - (0.00955)x^3 + 0.6035x^2 - 16.07x + 168.2 \ (III)$$

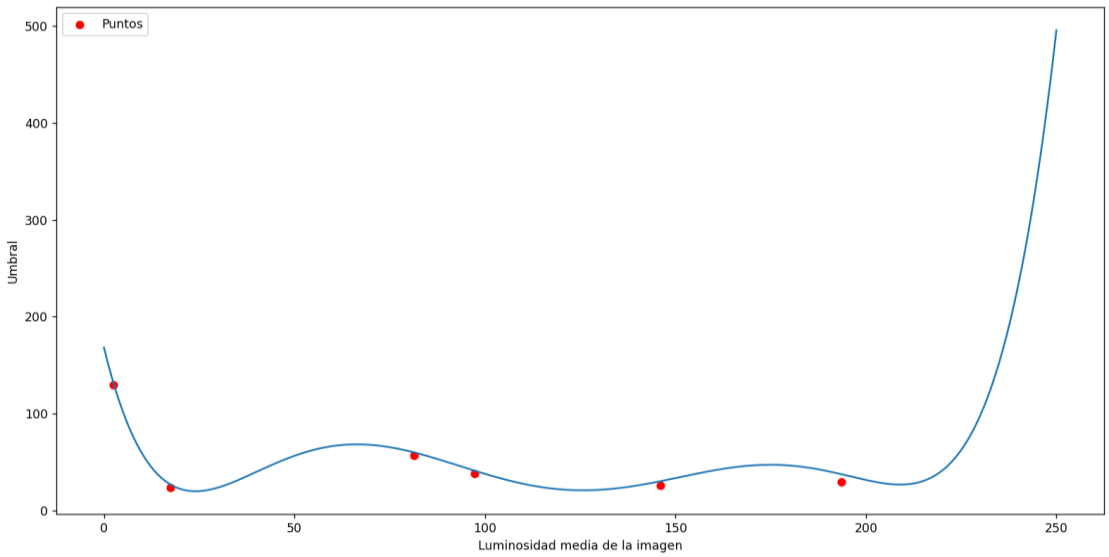

Figura 4.1.2.1: Función polinómica que representa la relación entre la luminosidad media de la imagen y el umbral máximo de tamaño de las ROI de la misma, junto a los puntos a partir de los que se ha elaborado el ajuste polinómico

4.1.3 Desviación estándar de la señal óptica

Otra forma de distinguir una señal óptica procedente de un transmisor, de una procedente de cualquier otro elemento es mediante la desviación típica. La desviación típica es una medida estadística que se utiliza para cuantificar la dispersión o variabilidad de un conjunto de datos con respecto a su media. Representa la medida promedio de la distancia entre cada valor del conjunto de datos y la media.

Una señal óptica modulada mediante OOK presenta generalmente desviaciones típicas superiores a 30, dado que la luminosidad oscila constantemente respecto a su media y la diferencia entre esos valores (unos y ceros) es notoria. En cambio, en otro tipo de elementos, tales como todos aquellos que simplemente reflejan la luz natural, esta variación es pequeña dado que la fuente principal (el Sol o una pantalla) no presenta variaciones tan significativas. Es decir, no varía considerablemente respecto a la media y por tanto no presentan desviaciones superiores a 30.

En la imagen de la izquierda de la Figura 4.1.3.1 se puede observar la señal óptica de un punto que simplemente refleja la luz ambiente, cuya desviación típica es de 4.21. Por su parte, en la imagen de la derecha de la misma Figura se puede observar la señal óptica de un transmisor del mismo escenario, cuya desviación típica es de 83.77.

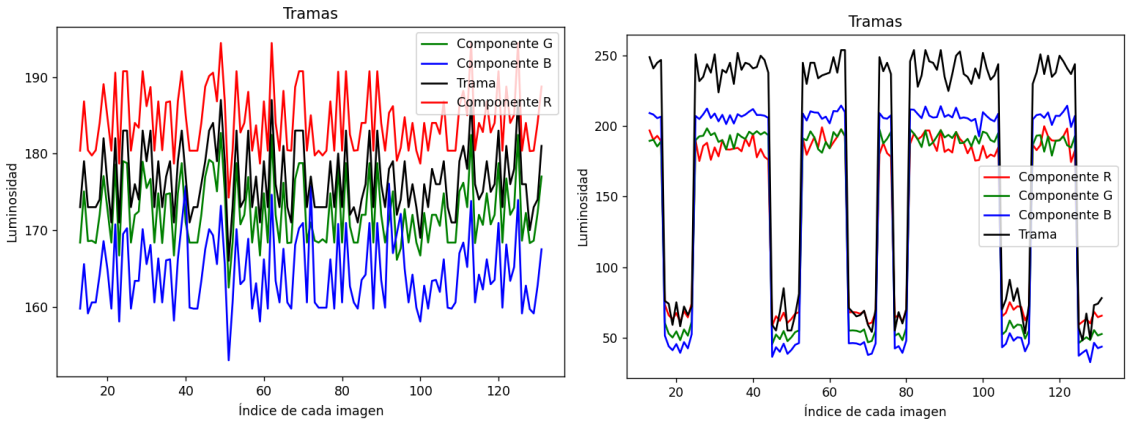

Figura 4.1.3.1: Señal óptica recibida de un punto de una puerta (izquierda) y de un transmisor azul (derecha) en el escenario indoor con tiempo de exposición medio

Esto también se puede observar de forma más evidente a través del histograma de la señal recibida de un transmisor óptico y de un punto cualquier. Un ejemplo de ello se puede apreciar en la Figura 4.1.3.2, donde se ha seleccionado uno de los transmisores (imagen izquierda) y un punto cualquiera de la pared (imagen derecha) en el escenario indoor, y se ha obtenido el histograma de la señal óptica asociada a cada caso. Como se puede apreciar, en la imagen de la derecha la dispersión de los valores principales es pequeña, mientras que en la imagen de la izquierda la dispersión es tan elevada que hay dos conjuntos principales, equidistantes a la media.

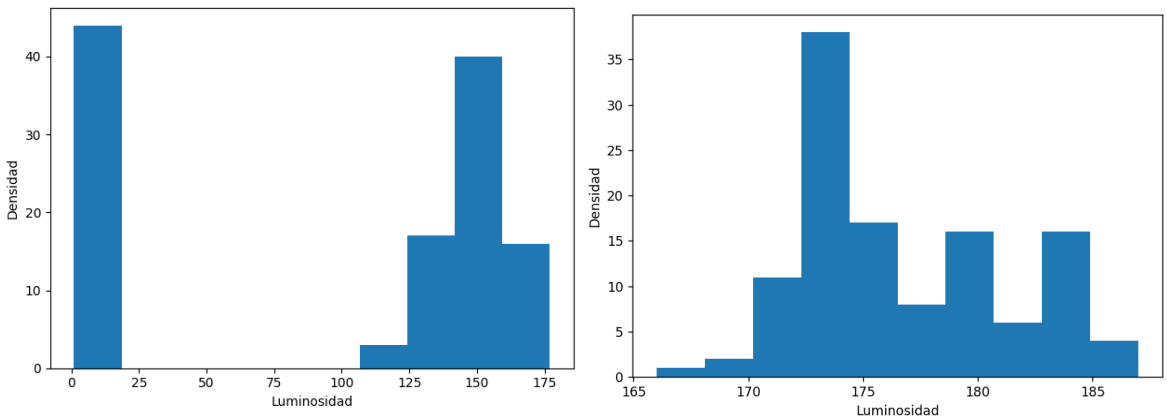

Figura 4.1.3.2: A la izquierda, histograma de un positivo verdadero. A la derecha histograma de un falso positivo

Para obtener los valores de desviación típica comentados, se presenta en el *Shell* el parámetro *trama* del Código A.1 al clicar elementos transmisores y no transmisores, y se copia y pega ese dato en el Código 4.1.3.1. En el mismo simplemente se utiliza la función *std* de la librería *numpy*, y se presenta el resultado por pantalla. De esta manera, fue posible cuantificar las diferencias entre las señales ópticas procedentes de transmisores y procedentes de otro tipo de elementos.

```python
import numpy as np

# Datos proporcionados
data = [252, 225, …, 251, 230]

# Convertir la lista de arrays en un array bidimensional
data = np.array(data)

# Calcular la desviación estándar
std_deviation = np.std(data)

# Imprimir el resultado
print("La desviación estándar de los datos es:", std_deviation)
```

Código 4.1.3.1: Script *check_std*

4.1.4 Estructura de la trama

Los intervalos de tiempo máximos y mínimos durante los que deben mantenerse los niveles de luminosidad de los píxeles vienen fijados por la estructura de la trama de datos, que puede verse en la Figura 3.1.1. Dada esta estructura, un píxel asociado a información modulada puede identificarse en función del número de flancos ascendentes de su forma de onda, o en función de la longitud de un periodo, por ejemplo. Cuantas más tramas de datos se estudien a la hora de identificar transmisores, menor será la probabilidad de falsos positivos, pero mayor será el tiempo de procesamiento, por lo que en el algoritmo sólo se analizan dos tramas para realizar la detección. Esto equivale a un vídeo de 4 segundos de duración.

En cuanto al número máximo y mínimo de flancos ascendentes y descendentes, estos parámetros son interesantes debido a la sencillez para obtenerlos. Para ello simplemente es necesario restar la trama (señal óptica convertida a unos y ceros tras aplicar un umbral), con ella misma desplazada a la derecha. El resultado será una matriz formada por ceros, unos (que indican flancos descendentes), y menos unos (que indican flancos ascendentes). Atendiendo a la Figura 3.1.1 se podrían dar los siguientes casos:

- Que el payload de las tramas consideradas fuera 255 (11111111), en cuyo caso hay 5 flancos ascendentes y 4 descendentes.
- Que el payload de las tramas consideradas fuera 0 (00000000), en cuyo caso hay 2 flancos ascendentes y 2 descendentes.
- Que el payload de las tramas consideradas fuera 170 (10101010), en cuyo caso hay 10 flancos ascendentes y 10 descendentes.

- Que el payload de las tramas consideradas fuera 85 (01010101), en cuyo caso hay 9 flancos ascendentes y 9 descendentes.

Por tanto, los valores mínimos y máximos de flancos ascendentes y descendentes son idénticos, 2 y 10, por lo que con comprobar uno de los mismos es suficiente.

En cuanto a la duración de los periodos, es un factor determinante que estos cumplan con los criterios de duración máxima y mínima para poder considerar que esa señal óptica proviene de un transmisor. El periodo máximo y mínimo de los mismos dependerá de si se está considerando una secuencia de unos o de ceros. De acuerdo con la Figura 3.1.1.1, en la trama puede haber un máximo de 9 unos seguidos y de 10 ceros seguidos. Por su parte, la cantidad mínima de ambas opciones es una. Teniendo en cuenta que cada bit es representado mediante unas cuatro imágenes, cada periodo debería estar comprendido entre una cantidad de 4 y 36 unos seguidos, o 4 y 40 ceros seguidos. Intervalos considerablemente inferiores o superiores en cualquiera de los periodos de la trama implicaría que no se pudiera considerar una señal óptica modulada. Para estudiar un periodo de unos o de ceros simplemente es necesario considerar los bits de la trama comprendidos entre las posiciones de un flanco de subida y bajada o bajada y subida. Suponiendo que las dos tramas a estudiar cuenten con un payload igual a 0, en total podrá haber como máximo 4 periodos.

También se podrían considerar otros parámetros relacionados con la estructura de la trama, como la cantidad total de imágenes que contienen un 0 o un 1 en las mismas, pero este dato es más impreciso ya que el rango de posibles valores es más amplio.

4.2 Descripción general

Dada la elevada cantidad de factores que podrían ser tenidos en cuenta a la hora de identificar un transmisor, y la elevada cantidad de píxeles que se podrían considerar, el algoritmo ha sido desarrollado mediante varios bloques de trabajo. En cada uno de los mismos se aplica alguna de las restricciones que marcan una distinción entre píxeles potencialmente transmisores y otros píxeles. Como resultado de cada uno, se mantiene o reduce la cantidad de candidatos a ser transmisores, lo que simplifica las tareas totales que es necesario llevar a cabo.

Los cinco bloques principales que se definen en el código son:

I. El bloque de obtención de imágenes (*loader*). En este se obtienen una cantidad especificada de imágenes asociadas a un vídeo, concretamente 120 (4 segundos de vídeo grabado a 30 FPS). Estas imágenes están almacenadas en un directorio y están numeradas siguiendo el

software de extracción de muestras presentado en el Código 3.4.2.1, por lo que es posible seleccionar el rango de imágenes que se desea tomar indicando el índice inicial de la secuencia. Además, en este bloque las imágenes son convertidas a escala de grises antes de ser enviadas a la siguiente etapa.

II. El bloque simplificador (*simplifier*). En este se analizan las imágenes asociadas a 2 segundos de vídeo (60 imágenes), tiempo suficiente para detectar al menos cinco unos (tiempo durante el cual el transmisor ilumina). En este análisis, una a una, se lleva a cabo la aplicación del umbral de luminosidad a las imágenes y el etiquetado de las regiones luminosas resultantes, tal y como se realizó en el Código A.3. Tras ello, se obtienen los datos de tamaño de región de cada área luminosa etiquetada, y se seleccionan como píxeles asociados a potenciales transmisores, solo el píxel central de todas aquellas regiones que tuvieran un tamaño inferior al obtenido mediante la Ecuación III. Un esquema de lo comentado se puede observar en la Figura 4.2.1.

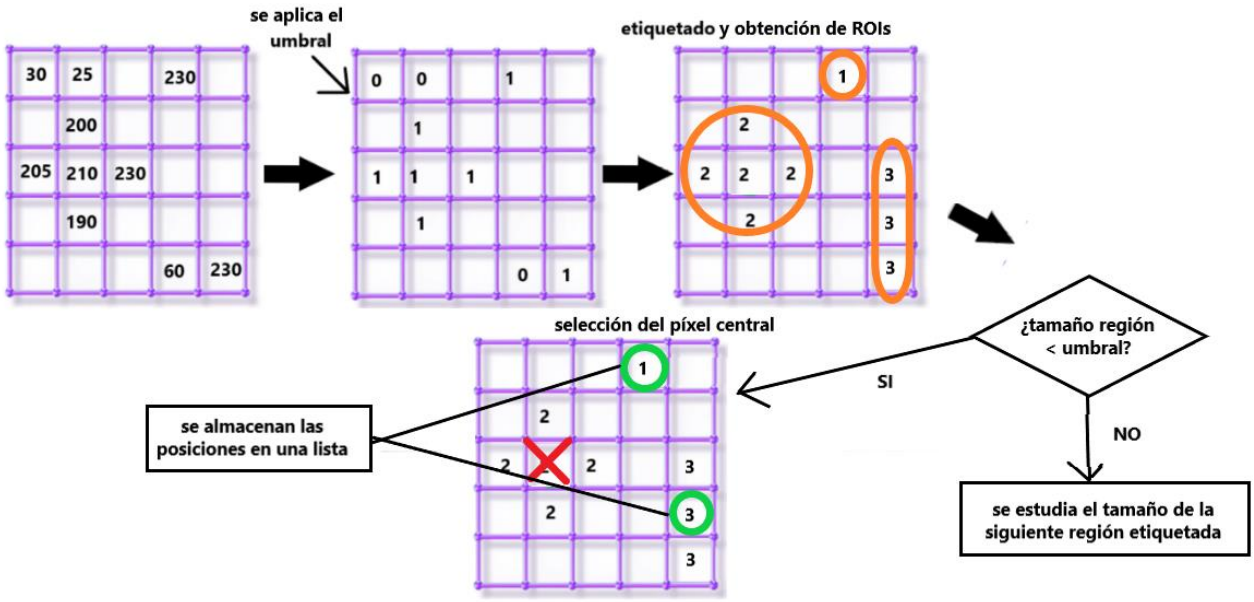

Figura 4.2.1: Representación gráfica de la forma de operar del bloque simplificador

III. El bloque correlador (*correlator*). Este recibe como parámetros de entrada las imágenes asociadas a 4 segundos de vídeo (2 tramas de datos), y los potenciales transmisores obtenidos del bloque simplificador. Luego, obtiene la trama de datos de cada uno de los píxeles potenciales mediante los valores de luminosidad de los 4 segundos de vídeo, calcula la desviación estándar de la misma y comprueba si ese valor es superior a 30. Sólo en ese caso, se aplica la umbralización a dicha trama, y se obtienen los flancos de subida y bajada de la misma, restando dos versiones de la señal desplazadas una posición entre sí. Una vez hecho

esto, se cuentan la cantidad de flancos de subida, y se compara con los valores teóricos estudiados anteriormente. Sólo en caso de que se satisfaga esa condición, se obtiene la longitud del segundo periodo de la trama y se comprueba que también cumpla con esos valores teóricos. Para calcular la longitud de dicho periodo se buscan los índices del primer flanco de subida y bajada y se restan en valor absoluto. Primero se aplica el condicionante de la desviación típica porque, sobre todo en escenarios indoor, existen muchos elementos reflectantes que pueden ser confundidos con potenciales transmisores en el bloque simplificador, y esta operación permite descartarlos de forma directa y sencilla. En segundo lugar, se decide comprobar la cantidad de flancos por la simplicidad de la operación. Por último, aunque la longitud del segundo periodo sea un factor bastante diferencial, el proceso de obtención del mismo es más costoso, por lo que es preferible que solo se realice en pequeñas ocasiones. El diagrama de flujo de este bloque se puede observar en la Figura 4.2.2.

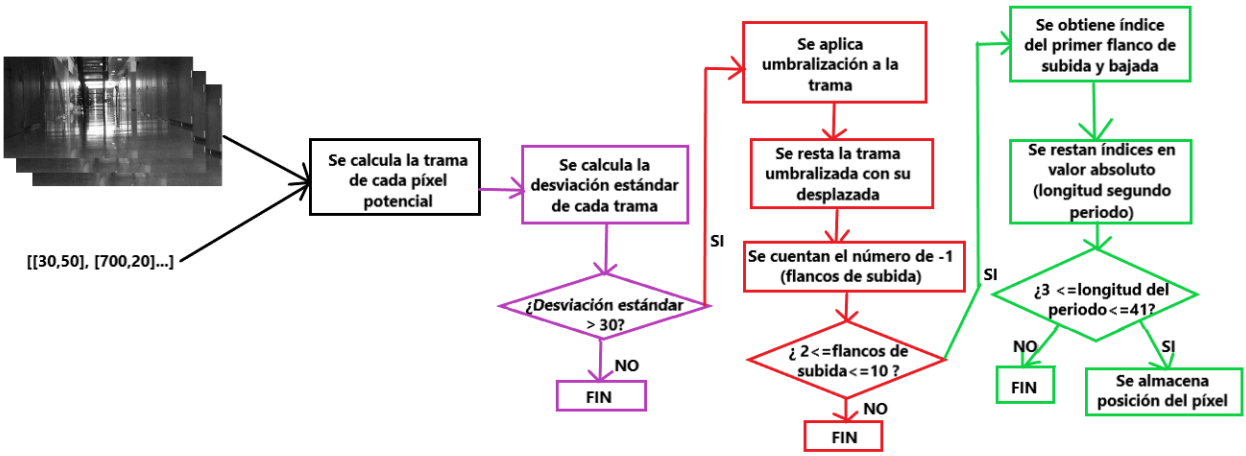

Figura 4.2.2: Diagrama de flujo del bloque correlador

IV. El bloque de eliminación de píxeles vecinos (*delete_neighbours*). Este recibe el resultado del bloque de correlación, y compara cada elemento de la lista con los restantes y elimina aquellos que se encuentren a una distancia inferior a 12 píxeles al sumar las separaciones en coordenadas “x” e “y”. De esta manera se evita que un transmisor sea considerado múltiples veces, algo que podría ocurrir dado que el píxel central obtenido en el bloque simplificador podría no ser el mismo en todas las imágenes estudiadas. Este bloque opera tras el proceso de correlación porque es costoso computacionalmente y hacer que funcione tras el bloque simplificador elevaría demasiado los tiempos de operación del algoritmo.

V. El bloque de presentación de resultados. Este presenta en el *Shell* la cantidad de transmisores, el tiempo de procesamiento de los distintos bloques, los valores de la señal óptica asociada a

cada uno de los transmisores, y el píxel asociado a cada señal óptica. Además, representa gráficamente la señal óptica, y una de las imágenes estudiadas, en la cual marca la ubicación de cada píxel considerado transmisor.

La estructura general del software comentado se puede observar en la Figura 4.2.3.

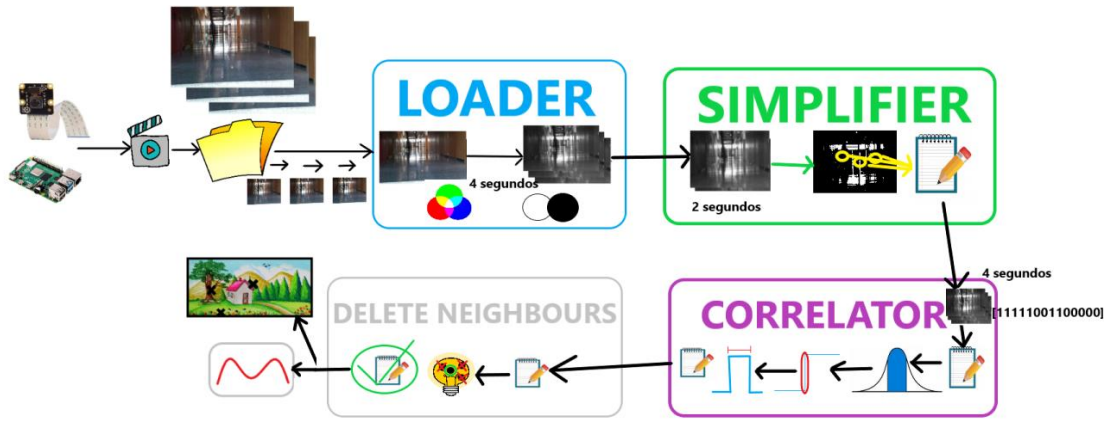

Figura 4.2.3: Estructura general del software desarrollado

4.3 Descripción técnica

El software de descubrimiento está formado por un total de cuatro scripts que permiten su funcionamiento. Hay un script principal en el que se definen métodos simples y en el que se incluye la función *main*, donde se llaman de forma secuencial a las distintas funciones fundamentales. Estas funciones están escritas en tres scripts diferentes debido a su complejidad, para simplificar el entendimiento del código. Estas tres funciones fundamentales son la obtención de imágenes (bloque I), la simplificación de píxeles potenciales (bloque II), y el estudio de las señales ópticas (bloque III). Por su parte, los bloques IV y V se llevan a cabo dentro del script principal dado que no conllevan una gran dificultad.

4.3.1 Multiprocessing

Para reducir los tiempos de procesamiento del software, y facilitar la paralelización de tareas se hace uso del módulo *multiprocessing* de Python. Este permite la ejecución de procesos en paralelo aprovechando los múltiples núcleos de la CPU (*Central Processor Unit*). Este módulo a su vez dispone de varias clases y métodos para el trabajo y ejecución de estos procesos en paralelo. Las clases y métodos utilizados son:

- Queue: Es una implementación de cola diseñada para la comunicación y el intercambio de datos entre procesos. Proporciona métodos como *put()* y *get()* para insertar y extraer objetos

de la cola. Estos métodos son seguros para ser utilizados por múltiples procesos al mismo tiempo, ya que se encargan de la sincronización y la gestión de bloqueos internamente. En el software desarrollado, en el script principal *main*, se crea una instancia de Queue. Esta cola es pasada por parámetro al método de carga de imágenes, que añade las mismas a la cola mediante el método put. De esta manera, es posible ir recibiendo en la función principal main las imágenes a medida que se obtienen, sin necesidad de tener que esperar a la finalización del método de carga.

- cpu_count: Función que devuelve el número de núcleos de CPU disponibles en el sistema. Se utiliza para determinar cuántos procesos se pueden ejecutar en paralelo.
- Pool: Es una clase que administra automáticamente la asignación de tareas y la distribución de la carga de trabajo entre un grupo de procesos. Cuando se crea una instancia de la clase Pool se crea un grupo de procesos. Por defecto, la clase utiliza el número de CPU disponibles en el sistema, pero se puede configurar un número menor indicando en la instancia de la clase el parámetro *processes*. Una vez creada una instancia de la clase, se pueden aplicar sobre el grupo de procesos varios métodos como *map*, *apply_async*, *map_async*, *imap*, *imap_unordered*, *join*, etc. De ellos únicamente se ha utilizado la función map.
    - map(func, iterable): Este método se utiliza para aplicar la función *func* a cada elemento del *iterable* y devuelve una lista de resultados en el mismo orden que los elementos del iterable. El orden de los resultados en la lista de salida es el mismo orden en el que aparecen los elementos en el iterable de entrada. Es decir, el resultado correspondiente al primer elemento del iterable estará en la primera posición de la lista de salida, el resultado correspondiente al segundo elemento estará en la segunda posición, y así sucesivamente. Esta función se ha utilizado únicamente para ejecutar de forma paralela el método de simplificación a lo largo de la lista de imágenes captadas. Con ello se reducen considerablemente los tiempos de procesamiento en comparación con si se ejecutase el método de forma concurrente por cada una de las 60 imágenes de estudio.

      La función *func* se ejecuta en tantos procesos como se haya configurado la instancia de Pool, dividiendo automáticamente la carga de trabajo entre los procesos del grupo. Si la cantidad de iterables no es exactamente igual a la cantidad de procesos de Pool, se dividen los elementos iterables para que cada proceso tenga la misma cantidad de trabajo. Como los iterables con los que se ha trabajado al usar esta función son imágenes, y no es deseable que se subdividan porque entonces la información de posición de los potenciales transmisores sería incorrecta, se han pasado como

iterables tantas imágenes como procesos forman la instancia Pool. En la Figura 4.3.1 se puede apreciar a la izquierda los errores que se podrían producir si se pasa una cantidad de imágenes distinta de la cantidad de procesos, y a la derecha lo que ocurriría si se pasa la cantidad exacta.

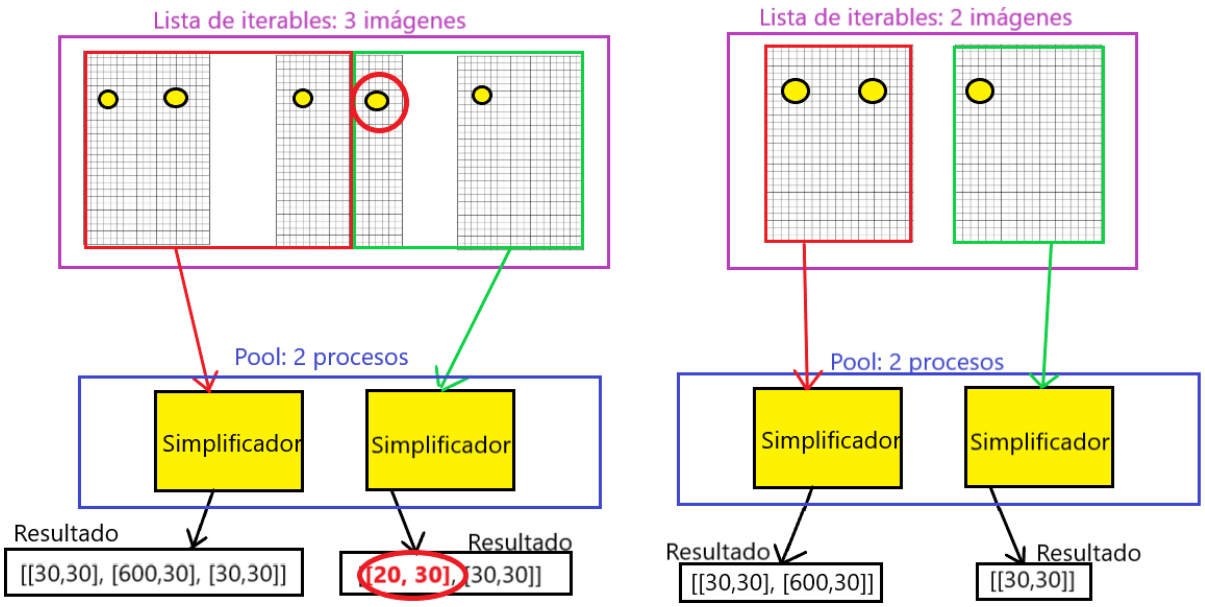

Figura 4.3.1: Comparativa de resultados al aplicar la función map sobre una instancia Pool siendo la cantidad de elementos de la lista iterable distinta (izquierda) e igual (derecha) al número de procesos

Si la función especificada recibe por parámetro otros elementos distintos del iterable principal, es necesario usar la función *partial* del módulo *functool*, que permite crear funciones parciales. Una función parcial en Python es aquella que se ha creado a partir de otra función original, de la cual se fijan algunos de sus parámetros. Al crear una función parcial es posible pasar al método map un método que tenga parámetros que no deban ser iterados, junto con el o los parámetros iterables. Es posible proporcionar varios iterables a la función map siempre cuando todos tengan la misma longitud. La función proporcionada debe aceptar tantos argumentos como la cantidad de parámetros fijos e iterables proporcionados.

- apply_async(func, args=()): Este método se utiliza para aplicar la función func a los argumentos args de forma asíncrona. Esto significa que su ejecución no bloquea el flujo principal del programa. En lugar de esperar a que se complete una tarea antes de continuar, el programa puede seguir ejecutando otras tareas mientras la tarea asíncrona se realiza en segundo plano. Los argumentos pasados por parámetro

pueden ser tanto fijos como iterables. La función devuelve un objeto AsyncResult que representa el resultado de la llamada. Se puede utilizar el método get() del objeto AsyncResult para obtener el resultado, pero esto no se puede realizar hasta que hayan finalizado todos los procesos.

- Process: Es una clase que representa un proceso individual. Se utiliza para tener un control más granular sobre la creación y ejecución de procesos específicos. Entre los parámetros que se pueden indicar al crear una instancia de esta clase se encuentran *target*, que permite indicar la función a ejecutar en ese proceso, o *args*, que permite proporcionar el conjunto de parámetros que puede necesitar esa función para poder operar. También se puede asignar un nombre al proceso mediante el parámetro *name*, o indicar el grupo de procesos al que se agregará mediante el parámetro *group*. Para iniciar la ejecución del proceso se debe llamar a la función *start* sobre la instancia del proceso. Para esperar a que termine se debe usar la función *join*. Para parar el proceso forzosamente se usa *terminate*. Para comprobar si el proceso está en ejecución se puede llamar a la función *is_alive*, que devuelve un booleano con el resultado. De los métodos comentados solo se ha usado la función *start*.

  En el software desarrollado la instancia Process es utilizada para comenzar a ejecutar el método de carga de imágenes como un proceso independiente. Como a la instancia de Pool se asignan todas las CPU del sistema, la simultaneidad real inicial de los procesos dependerá de cómo asigne el sistema operativo los recursos. No obstante, según finalice el proceso de carga de imágenes, que es temprano, la instancia Pool podrá utilizar todos los recursos para trabajar con *cpu_count* procesos.

4.3.2 Script principal

En el script principal se especifican todos los parámetros principales del programa: los procesos, las colas, los parámetros que se pueden configurar (directorio de imágenes, e índice inicial), la llamada a las distintas funciones… Dada la cantidad de tareas que se ejecutan en este script, se instancian una gran cantidad de módulos: multiprocessing, functools y socket para la paralelización de tareas; loader, simplifier y correlator para poder operar con las funciones de los bloques principales definidos, y numpy y cv2 para el trabajo con las imágenes.

Por el mismo motivo, también se definen una gran cantidad de variables: los parámetros que se pueden configurar del software; una instancia de cola para paralelizar los procesos de carga de imágenes y obtención de transmisores potenciales mediante el bloque simplificador; la cantidad de procesadores de las que dispone el sistema, para poder pasarle cantidades de imágenes exactas o

proporcionales a esa cifra a la función *map* de *Pool*, y los parámetros con los que se ha configurado la grabación, para poder obtener el número de imágenes necesarias en función del número de tramas que se desean visualizar.

En cuanto a los métodos, sin contar el principal (*main*), se definen dos:

- *delete_neighbours_pixels*: recibe por parámetros una lista con la posición de los píxeles en los que se puede encontrar potencialmente un transmisor, la trama temporal de duración 4 segundos (120 muestras) de cada uno de esos píxeles, y el valor de distancia máxima que puede tomar la suma de separaciones en "x" e "y" para que se consideren pertenecientes a una misma fuente luminosa. En este método se utilizan dos bucles anidados para obtener constantemente el valor de restar en "x" e "y" cada píxel de la lista con los restantes. Tras obtenerse el valor de cada resta se realiza la suma de las separaciones en las dos coordenadas y se compara dicho valor con el umbral pasado como tercer parámetro. Si el valor obtenido tras la suma es menor al límite, se añade el índice de ese segundo píxel a la lista de posiciones a eliminar (*position_to_delete*). Una vez finalizados los bucles anidados, se quitan de la lista de posiciones a eliminar aquellos valores repetidos, y se ordenan en orden decreciente para que las eliminaciones se realicen de forma correcta (cuando se eliminan elementos de una lista cambia la posición de los elementos restantes). Luego, se eliminan tanto los elementos de la lista de *array*, como de *tramas* en base a los índices de *position_to_delete*, y se retornan los dos arrays.
- *calculate_light_umbral*: este método recibe por parámetro la luminosidad media de una imagen, y retorna los umbrales de luminosidad y ROI correspondientes aplicando las Ecuaciones II y III.

Por su parte, en cuanto al método principal *main*, es el de mayor longitud de todo el software. En el mismo, en primer lugar, se crea una instancia de Pool. Luego, se calcula y se presenta la cantidad de imágenes a solicitar al método *get_frames* del script *loader*. Acto seguido, se crea y se inicia una instancia de un *Process* de *multiprocessing* en el que se ejecuta el método de carga de imágenes (Bloque I).

Una vez se ha iniciado el método *get_frames*, ya es posible recuperar las imágenes convertidas a escala de grises de la cola. Por ello, se establece un bucle usando el condicional *while* y parámetros contadores, en el que se obtienen las imágenes una a una mediante el método *get*, especificándose en el mismo un tiempo de espera de 20 segundos para recibir la imagen o abortar la misión de obtener

elementos de la cola. Una vez se obtiene la primera imagen, y únicamente en ese caso, se usa el método *calculate_light_umbral* para obtener tanto el umbral de luminosidad como de ROI.

Luego, en el resto de las iteraciones del bucle, se van anexando las distintas imágenes a dos listas: la lista que se pasará al bloque simplificador solo contendrá las 60 primeras imágenes, y la lista que se pasará al bloque correlador contendrá las 120. Para evitar el problema de que las imágenes se fraccionen al pasarlas al método de *get_possible_tx* del script *simplifier*, se establece como condicionante antes de llamar al método *map*, que la lista de imágenes del simplificador sea proporcional a la cantidad de procesos de *Pool* y mayor a cero. Para ello se usa el operador *%*, que permite obtener el resto de la división de ambos elementos. Si se cumple la condición de proporcionalidad, se inicia la paralelización de procesos (líneas 111 - 115), se resetea la lista de imágenes (ya que estas ya han sido analizadas), y se almacenan los resultados (líneas 119 – 128). Dado que el método *map* se llamará en varias ocasiones, es necesario definir un segundo contador para que, en caso de que sea el primer resultado se copie el resultado en una lista almacén, y en caso de que no sea el primero se vayan concatenando a dicha lista.

Una vez finalicen de captarse todas las imágenes, y de realizarse todos los procesos de simplificación, se transforma el array que es una lista de lista de resultados, en simplemente una lista de resultados (líneas 135 - 144). Una vez hecho esto, se eliminan los elementos repetidos que puedan existir (líneas 146 - 151).

Finalmente, se pasa la lista resultante al bloque correlador para reducir aún más la cifra de potenciales transmisores (líneas 158 - 160), y se usa la lista de píxeles resultantes a su vez para pasarla al método *delete_neighbours_pixels* (línea 163). Tras esto, se presenta por el Shell la trama y la posición de cada píxel, y se representa gráficamente tanto la señal óptica asociada a la trama, como una imagen de la muestra de vídeo estudiada con una marca textual en la posición de cada píxel.

Para representar las imágenes de los escenarios se usa la función *imshow* de cv2, *putText* de cv2 para añadir texto a la imagen y *waitKey(0)* para que no se ejecute ninguna acción hasta que se pulse una tecla. Por su parte, para representar la señal óptica se usa el módulo *matplotlib.pyplot*. Por último, para representar la información por el Shell se usa la función *print*.

El software comentado en este apartado se puede observar al completo en el Anexo B.

4.3.3 Bloque de obtención de imágenes

En el script asociado a este bloque, observable en el Código 4.3.3.1, simplemente es necesario usar el módulo OpenCV2 (*Open Source Computer Vision Library*), que se utiliza para cargar las imágenes del

directorio y convertirlas a escala de grises. El método definido en este script, *get_frames*, recibe por parámetro el nombre del directorio en el que se buscará la secuencia de imágenes a cargar, el índice de la imagen inicial que se carga, la cantidad de imágenes que se cargan y el elemento de tipo *multiprocessing queue* al que se irán añadiendo las imágenes. Para obtener las imágenes se utiliza un bucle de tipo *for* que permite iterar en la búsqueda, obtención, conversión a escala de grises y adición a la cola. Todo este proceso se realiza dentro de una estructura *try - except*, ya que se podrían producir errores tales como que el nombre del directorio o el índice inicial no se correspondan con elementos válidos.

```
1   import cv2 as cv
2
3   # Método que añade a la cola del programa las imágenes asociadas
4   # a un rango de imágenes en una base de datos
5   def get_frames(directory, init_frame,
6                  number_of_frames, frames_queue):
7       try:
8           for count in range(init_frame,
9                              int(init_frame + number_of_frames)):
10            image = cv.imread(f"{directory}/frame{count}.jpg")
11            # se convierte la imagen a escala de grises
12            gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
13            frames_queue.put(gray) # se añade imagen a Queue
14      except:
15          print('frame inicial demasiado elevado')
```

Código 4.3.3.1: Script *loader*

4.3.4 Bloque de simplificación

En este script, que se puede observar en el Código 4.3.4.1, se utilizan el submódulo *measure* de la biblioteca *scikit-image* para poder etiquetar las regiones de las imágenes, la biblioteca *numpy* para poder eliminar elementos repetidos de las listas, así como obtener su tamaño, y el módulo *cv2* de la biblioteca OpenCV para poder aplicar umbrales de luminosidad a la imagen.

El script consta de un único método *get_possible_tx*, que recibe por parámetro la luminosidad y la ROI umbral, y una imagen en escala de grises. El funcionamiento del método es similar a lo comentado en el Anexo A.3. No obstante, en este caso, la luminosidad y la ROI se pasan por parámetro en vez de calcularse *in situ*, y en vez de devolverse el tamaño de la ROI del pixel clicado, se obtienen las ROI de todas las regiones etiquetadas, se comprueba si el valor de ROI de cada una de las regiones es menor al umbral y sólo en ese caso se devuelve el píxel central.

En el código, en primer lugar, se aplica el umbral de luminosidad a la imagen en escala de grises mediante el método *threshold* del módulo *cv2*. La justificación de la máscara utilizada en este método se puede observar en el apartado A.3 del Anexo A. Luego, se etiquetan cada una de las regiones luminosas mediante la función *label* del submódulo *measure*. Esta función asigna un mismo número a todos los píxeles contiguos con mismo valor. Para recorrer las distintas regiones etiquetadas se utiliza un bucle *for* y el método de numpy *unique* para eliminar los elementos repetidos de la imagen etiquetada *label*, y obtener así la lista de etiquetas que se han utilizado en la misma. En cada iteración del bucle se obtiene el tamaño de la región haciendo uso de la función de numpy *count*_nonzero, contando la cantidad de píxeles en la imagen que tienen el valor de esa etiqueta. Asimismo, se obtienen los índices de todos los píxeles con esa etiqueta mediante la función de numpy *where*. Si el tamaño de la región etiquetada bajo estudio es menor al umbral máximo de ROI, se almacena el píxel central de la misma. El píxel central se obtiene dividiendo entre 2 el tamaño de la región en cada eje ("x" e "y"), y aplicando al resultado la función *round* que aproxima el resultado entero más cercano. En caso de que la región solo ocupe un píxel no es necesario calcular ese píxel central.

```
1  from skimage import measure
2  import numpy as np
3  import cv2
4
5  # Método que obtiene la posición en el fotograma de posibles tx en base a
6  # la luminosidad y tamaño de grupo de píxeles
7  def get_possible_tx(light_umbral, roi_umbral, gray):
8      # Array que contiene los pares x,y asociados con posibles tx
9      index_pixels = []
10     # Se añade el elemento inicial para evitar errores relacionados
11     # con elementos nulos
12     index_pixels.append([0,0])
13
14     # Se umbraliza la imagen para revelar regiones de luz en
15     # la imagen gris
16     (T, thresh) = cv2.threshold(
17         gray, light_umbral, 255, cv2.THRESH_BINARY_INV & cv2.THRESH_OTSU)
18
19     # Se etiquetan las regiones contiguas
20     labels, num = measure.label(thresh, background=0, return_num=True)
21     # Se recorren las distintas regiones
22     for label in np.unique(labels):
23         # Se cuentan los píxeles con esa etiqueta
24         numPixels = np.count_nonzero(labels == label)
25         # Se obtiene una una lista de dos arrays con las
26         # posiciones x, y que cumplen la condicion
27         index_region = np.where(labels == label)
28         # Si el tamaño de la region es menor al umbral,
29         # almacena pixel central
30         if numPixels <= roi_umbral: # si numero de pixeles es menor al max
```

```
31              if np.shape(index_region)[1] == 1: # si la ROI es 1 pixel
32                  index_pixels.append(
33                      [index_region[0][0],index_region[1][0]])
34              else: # si 1 < ROI <= umbral
35                  index_pixels.append([index_region[0][round((
36                      np.shape(index_region)[1])/2)],index_region[1][round((
37                      np.shape(index_region)[1])/2)]])
38
39      return index_pixels
```

Código 4.3.4.1: Script *simplifier*

4.3.5 Bloque de correlación

El script asociado a este bloque se puede observar en el Código 4.3.5.1 En el mismo, se define el método *get_tx_pixels*, que recibe por parámetro un array con 120 imágenes en escala de grises, una lista de posiciones *[x, y]* asociadas a posibles transmisores, y el umbral de luminosidad. En dicho método se recorre cada valor *[x,y]* de la lista, y se obtiene su trama asociada recorriendo la lista de imágenes y almacenando el valor de luminosidad que toma ese píxel en cada imagen. Una vez obtenida la trama, se usa la función *std* de numpy para calcular la desviación estándar de la trama y el condicional *if* para ver si se encuentra entre los umbrales aceptables. Solo en ese caso, se aplica un umbral a la trama mediante una estructura condicional *if-else* para convertir la misma a formato de "unos" y "ceros". Este umbral aplicado es el 70% del calculado por la Ecuación *II* ya que es necesario considerar la variabilidad de la señal óptica.

Una vez calculada la trama en formato de "unos" y "ceros" se resta el valor de cada posición de la trama con el de la siguiente posición para obtener los flancos de subida (los que toman el valor -1) y bajada (los que toman el valor 1). Luego, se cuentan la cantidad de flancos positivos con la función de numpy *count*, y se usa nuevamente una estructura condicional *if* para comprobar si se encuentra en los valores esperados.

Finalmente, se busca el índice del primer máximo y el primer mínimo en la trama mediante los métodos de numpy *argmax* y *argmin*, y se calcula la diferencia en valor absoluto mediante el método *abs* de numpy. Luego, se comprueba si se encuentra entre los valores esperados y solo en ese caso se almacenan tanto la posición del píxel como su trama mediante la función de numpy *append*.

```
1   import numpy as np
2
3   # Metodo que obtiene la posición en la imagen de posibles tx en base
```

```
4   # a la relacion de sus valores en el tiempo y el formato de la trama
5
6   def get_tx_pixels(images, posible_tx, light_umbral):
7       pixels = []
8       tramas = []
9       threshold =  0.7*light_umbral
10
11      # Recorrer la matriz de píxeles potencialmente válidos
12      for posible_tx_count in range(0, len(posible_tx)):
13          trama = []
14          trama_10 = []
15          # Recorrer el conjunto de imágenes en escala de grises
16          for images_count in range(0, len(images)):
17              # Se crea una trama con todas las imágenes
18              trama.append(
19                  images[images_count][
20                      posible_tx[posible_tx_count][0],
21                      posible_tx[posible_tx_count][1]])
22          # Se calcula la desviación estándar de la trama
23          desviacion_estandar = np.std(trama)
24
25          # Se verifica si la desviación estándar es mayor o igual a 30
26          if 30 <= desviacion_estandar:
27              # Se aplica el umbral a la trama
28              trama_10 = [0 if x<threshold else 1 for x in trama]
29              # Se obtienen los flancos de bajada (1) y subida (-1)
30              for r in range(0, len(trama_10)-1):
31                  trama_10[r] = trama_10[r] - trama_10[r+1]
32
33              rises = trama_10.count(-1) # flancos de subida
34              # Chequear cantidad de flancos de subida
35              if (2 <= rises <= 10):
36                  # SEGUNDO PERIODO
37                  # Se obtiene el índice del primer flanco de bajada
38                  ind_down = np.argmax(trama_10)
39                  # Se obtiene el índice del primer flanco de subida
40                  ind_rise = np.argmin(trama_10)
41                  # Se obtiene el segundo periodo
42                  period2 = np.abs(ind_down-ind_rise)
43                  max = 40
44                  min = 4
45                  # Se comprueba que el periodo se encuentra entre
46                  # los umbrales teoricos
47                  if ((min - 1  <= period2 <= max + 1) ):
48                      pixels.append(posible_tx[posible_tx_count])
49                      tramas.append(trama)
50
51      return [pixels, tramas]
```

Código 4.3.5.1: Script *correlator*

Capítulo 5

5. Caracterización del algoritmo

En este capítulo de la memoria se realiza una descripción de las métricas utilizadas para comprobar la validez del algoritmo desarrollado. Luego, se describen los scripts o modificaciones de los mismos llevadas a cabo para evaluar las métricas comentadas. Finalmente, se resumen y presentan los resultados obtenidos durante la fase de experimentación.

5.1 Métricas

El objetivo de la fase de experimentación es tanto demostrar la calidad y el correcto funcionamiento del algoritmo desarrollado, como evaluar la viabilidad y calidad de los sistemas OCC en los distintos escenarios probados. Para ello, se pueden utilizar diferentes parámetros que demuestren empíricamente que los píxeles indicados al ejecutar el algoritmo se corresponden con los emisores de la escena, y que permitan evaluar la calidad de las señales recibidas. En este trabajo, se han utilizado seis métricas diferentes.

1. La forma de la onda de la señal óptica recibida: Si se trata de un verdadero positivo, la forma de la onda debe ser similar a la estructura de trama de datos definida en la Figura 3.1.1; no hay periodos excesivamente largos o cortos de unos y ceros y se detecta la cabecera. Esta forma de onda se puede obtener mediante el Código A.1.1 utilizado durante el análisis preliminar.
2. La posición estimada del píxel asociado al transmisor: Las posiciones estimadas se obtienen ejecutando también el Código A.1.1, que permite obtener manualmente, haciendo clic, la posición en la imagen de cada píxel seleccionado. Comparando las posiciones del transmisor devueltas por el algoritmo con las posiciones estimadas seleccionadas, es posible comprobar si cada valor devuelto corresponde a un falso positivo, como por ejemplo un objeto reflectante, o a un verdadero positivo.
3. El histograma de la trama de datos del píxel: El histograma asociado a un transmisor tiene una forma similar a una mezcla de dos gaussianas, una gaussiana y una gaussiana truncada,

o dos gaussianas truncadas. Como los "unos" y los "ceros" se repiten constantemente, hay dos valores de luminosidad que se repiten más que el resto de forma considerable. En el caso de píxeles que no estén asociados a elementos transmisores, el histograma de los casos suele mostrar valores concentrados en torno a un mismo punto (gaussiana única). Esto se puede comprobar en la Figura 4.1.3.2.

Para calcular y representar el histograma de los transmisores detectados se añadió el Código 5.1.1 al final del script principal (*main*). Como se puede observar, para obtener y representar el histograma simplemente se usa la función *hist* del módulo *pyplot* de la librería *Matplotlib*, que devuelve el número de contenedores (*bins*) en los que se divide el rango de valores, la frecuencia de repetición de cada contenedor (*n*) y los objetos de tipo *Rectangle* que representan los rectángulos que forman el histograma (*patches*). Finalmente, se usan nuevamente funciones como *title*, *xlabel* o *ylabel* para etiquetar la nueva figura.

```
# OBTENEMOS Y REPRESENTAMOS HISTOGRAMA
      (counts, bins, patches) = plt.hist(y)
      plt.title("Frame graph {}".format(id))
      plt.xlabel("Luminosidad")
      plt.ylabel("Densidad")
      plt.show()
```

Código 5.1.1: Cálculo y representación de histogramas

4. El tiempo de ejecución del algoritmo: Medir la rapidez del algoritmo en diferentes escenarios es fundamental para evaluar sus posibles aplicaciones y limitaciones. Para llevar a cabo esta medida se añadió al script principal (*main*) el módulo *time*, que dispone de distintos métodos que permiten controlar este parámetro. Concretamente se hizo uso del método *time*, que devuelve el tiempo actual del sistema en segundos desde el *epoch*. Estos valores de tiempo se almacenan en distintas variables antes y después de ejecutar cada uno de los dos bloques principales (simplifier y correlator) y se presentan por pantalla sus respectivas diferencias. Es decir, se presenta el tiempo que tardan en ejecutarse cada uno de estos dos bloques.
5. La relación señal a ruido: La SNR es una medida de la relación entre la intensidad de la señal y el nivel de ruido de fondo presente en un sistema de comunicación. En este trabajo, la SNR se calculó utilizando un enfoque de ventana deslizante y un Modelo de Mezcla Gaussiana (GMM - *Gaussian Mixture Model*) para estimar la SNR esperada. La fórmula para

calcular la SNR en este estudio se puede observar en la Ecuación IV, donde μ0 y σ0 son el valor esperado y la desviación estándar de la primera Gaussiana, μ1 y σ1 son el valor esperado y la desviación estándar de la segunda Gaussiana, y α es la ponderación o contribución relativa de la primera Gaussiana al cálculo de la SNR. La primera Gaussiana representa la señal y la segunda Gaussiana representa el ruido de fondo. Por tanto, la proporción α indica qué porcentaje de la mezcla total corresponde a la señal y qué porcentaje corresponde al ruido de fondo [6].

$$SNR = \frac{1}{2} \cdot \frac{|\mu 1 - \mu 0|}{\alpha \sigma_0^2 + (1 - \alpha) \cdot \sigma_1^2} \quad (IV)$$

En la práctica, para obtener este dato se añadió al script principal (main) el módulo *sklearn.mixture*, de la biblioteca *scikit-learn*, que dispone de la clase *GaussianMixture* que permite modelar datos utilizando una mezcla de distribuciones gaussianas. El software utilizado se puede observar en el Código 5.1.2. Este se añadió también al final del script principal, main.

En el mismo, los datos de la trama (y) se convierten en una matriz bidimensional para su procesamiento. Luego, se crea una instancia del modelo de mezcla gaussiana (GaussianMixture) con "*n_components = 2*", lo que implica que se ajustará una mezcla de dos componentes gaussianas a los datos. Tras ello, se lleva a cabo el ajuste mediante el método *fit*.

A continuación, se obtienen los parámetros estimados del modelo de mezcla gaussiana. Esto incluye el peso (alpha) de la primera componente gaussiana, la media (mu0 y mu1) de las dos componentes gaussianas y la desviación estándar (sigma0 y sigma1) de las dos componentes gaussianas. Una vez conocidos estos datos, finalmente, se calcula la SNR siguiendo la Ecuación IV.

```
    # CALCULAMOS SNR
       # Se convierte el array que contiene la trama (y)
       # en una matriz bidimensional
       y = np.array(y).reshape(-1, 1)
       # Se crea un objeto GaussianMixture con 2 componentes
       gmm = GaussianMixture(n_components=2)
       # Se ajusta el modelo de mezcla gaussiana a los datos de y
       gmm.fit(y)


```

```
        # Se obtienen los parámetros estimados del modelo GMM
        # Ratio de la primera componente gaussiana
        alpha = gmm.weights_[0]
        # Media de la primera componente gaussiana
        mu0 = gmm.means_[0][0]
        # Desviación estándar de la primera componente gaussiana
        sigma0 = np.sqrt(gmm.covariances_[0][0][0])
        # Media de la segunda componente gaussiana
        mu1 = gmm.means_[1][0]
        # Desviación estándar de la segunda componente gaussiana
        sigma1 = np.sqrt(gmm.covariances_[1][0][0])

        # Se calcula la SNR utilizando la formula proporcionada
        SNR = (1 / 2) * ((mu1 - mu0)**2) / \
        (alpha * (sigma0**2) + (1 - alpha) * (sigma1**2))

        SNR_dB = 10 * math.log10(SNR)


        # Se imprimen el resultado
        print("SNR (dB):", SNR_dB)
```

Código 5.1.2: Cálculo y representación de la SNR

6. La tasa de error: La BER es una medida del número de bits erróneos recibidos en comparación con el número total de bits transmitidos en un sistema de comunicación. En este trabajo, se calculó la tasa de error teórica para cada SNR estimada experimentalmente utilizando la función de error complementaria (erfc). La fórmula para calcular la BER en este estudio se puede observar en la Ecuación V, donde erfc(×) es la función de error complementaria [6].

$$BER = \frac{1}{2} \cdot erfc\left(\sqrt{\frac{SNR}{2}}\right) (V)$$

Este dato se calcula tras obtenerse la SNR utilizando el Código 5.1.3. Para ejecutar el mismo simplemente es necesario importar la función *erfc* de la biblioteca *SciPy* y ejecutar la misma siguiendo la Ecuación V.

```
    # Se calcula la tasa de error de bits (BER) utilizando la SNR
    BER = (1 / 2) * erfc(np.sqrt(SNR / 2))

    # Se imprime el resultado
    print("BER:", BER)
```

Código 5.1.3: Cálculo y representación de la BER

5.2 Resultados

En este apartado se resumen los resultados obtenidos durante la fase de experimentación. Estos resultados han sido anotados en un documento Excel, adjunto a este escrito y consultable en el Anexo C. Los mismos se clasifican en función del escenario donde se han tomado los vídeos.

5.2.1 Escenario indoor

Como se comentó anteriormente, en este caso se grabaron tres vídeos utilizando una sensibilidad ISO de 100 y tres tiempos de exposición diferentes: 9, 14 y 19 ms. Una imagen asociada a cada caso se puede apreciar en la Figura 5.2.1.1.

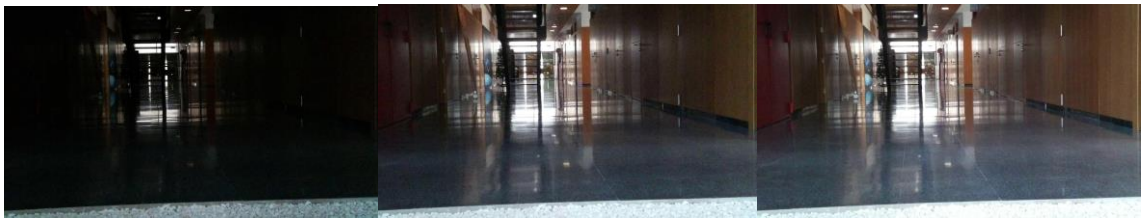

Figura 5.2.1.1: Escenario indoor con tiempo de exposición de 9 (izquierda), 14 (medio) y 19 ms (derecha)

En total se analizaron 45 muestras de vídeo en las que se detectaron en promedio el 88 % de los transmisores. Los mejores resultados se obtuvieron al utilizar el tiempo de exposición más elevado, seguido del tiempo de exposición más bajo y el tiempo de exposición medio. En la Tabla 5.2.1.1 se puede observar un resumen de las tasas de detección en cada caso.

Casos	Tiempo de exposición alto	Tiempo de exposición medio	Tiempo de exposición bajo
Se detectaron todos	79 %	29 %	71 %
Se detectaron 3/4	21 %	57 %	29 %
Se detectaron 2/4	0 %	14 %	0 %

Tabla 5.2.1.1: Tasa de detección en escenario indoor

Como se observa, en el mejor de los casos se detectan todos los transmisores en el 79 % de las muestras analizadas, mientras que en el peor solo en el 29 %. En el 32 % de los casos, los transmisores no son detectados porque una o varias personas los obstaculizan. En el resto de

ocasiones, el transmisor no detectado es el mismo, y se corresponde con el que mayor potencia lumínica presenta, uno de los dos LED verdes. Este se ha omitido en tantas ocasiones debido a que sus niveles de luminosidad asociados a los ceros, superan en algunas ocasiones el umbral utilizado para distinguir los unos y ceros, lo que provoca que su trama sea comprendida como una larga secuencia de unos. Esto se debe a que tiene un nivel de luminosidad tan elevado que satura el sensor de imagen, y el nivel de luminosidad de los ceros no desciende lo que debería. No obstante, si se aumentase el umbral de luminosidad, esto provocaría que los LEDs con menor potencia sean omitidos en más ocasiones que las que se obvia el LED de mayor potencia.

En cuanto a los falsos positivos, el escenario indoor es el que peores resultados presenta. Para todos los tiempos de exposición casi se detectan la misma cantidad de positivos verdaderos que de falsos positivos. No obstante, al analizar las tramas de dichos falsos positivos se pudo comprobar que en el 97,5 % de las muestras de vídeo estudiadas todos esos falsos positivos se corresponden en realidad con reflexiones de las señales ópticas de los transmisores. Esto trae consigo múltiples ventajas, como el hecho de que en la mayor parte de los casos en los que no se detectaba un transmisor, se podía recibir su información a partir de la reflexión de su señal óptica en el suelo. Un ejemplo de ello se puede apreciar en las Figuras 5.2.1.2 - 5.2.1.4. Como se puede apreciar en las mismas, la forma de la señal se conserva de forma fiel en las reflexiones, lo único que las diferencia es el nivel de luminosidad.

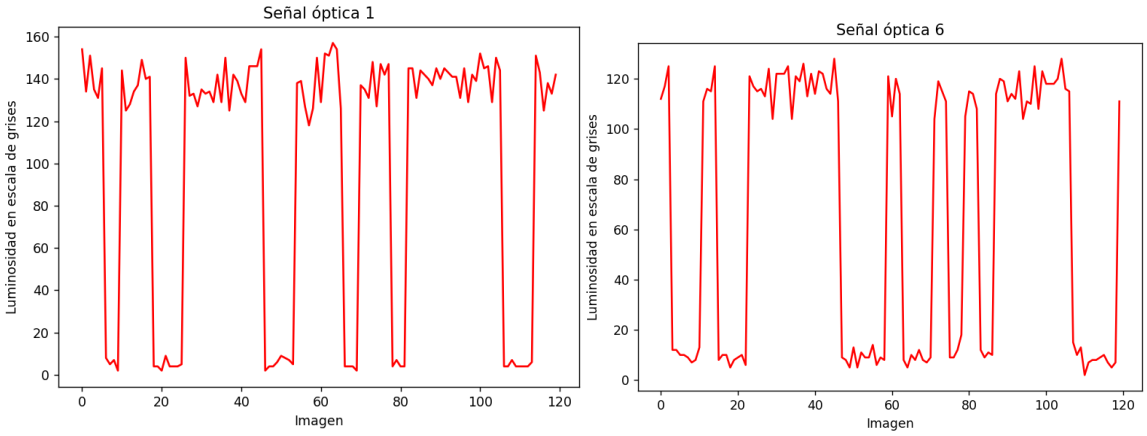

Figura 5.2.1.2: Señal óptica recibida de un transmisor y de una superficie reflectante que refleja una señal del mismo

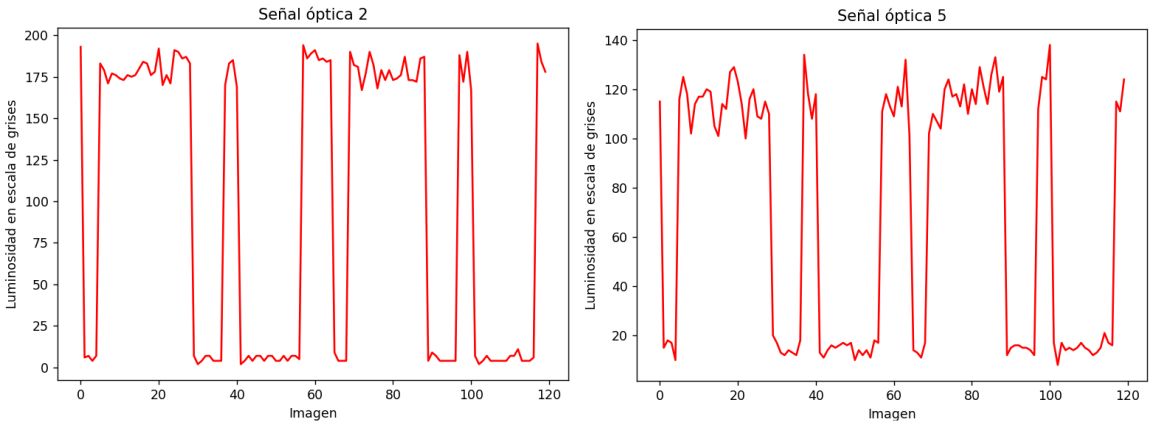

Figura 5.2.1.3: Señal óptica recibida de un transmisor y de una superficie reflectante que refleja una señal del mismo

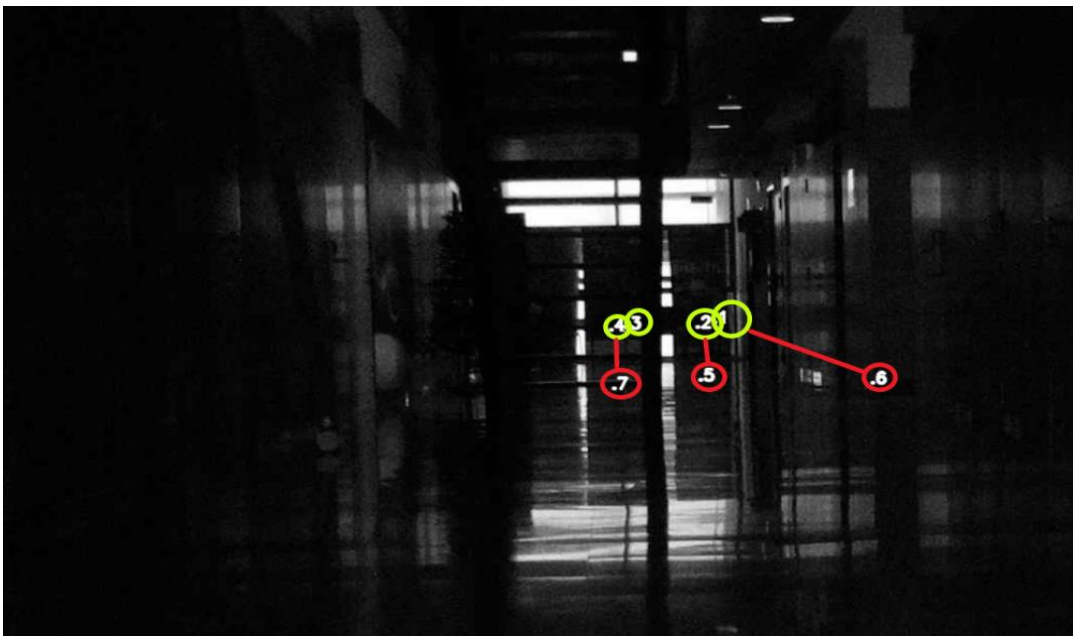

Figura 5.2.1.4: Transmisores verdaderos en verde y puntos reflectantes de los mismos en rojo

En el resto de muestras estudiadas, los falsos positivos que aparecieron se produjeron de forma puntual y abundante debido al paso de personas. Un ejemplo práctico se puede observar en la Figura 5.2.1.5.

Figura 5.2.1.5: Falsos positivos debido al paso de una persona

En cuanto a la relación señal a ruido y la tasa de error de bit, los LEDs verdes han presentado un mejor comportamiento en los tres escenarios. Esto se debe a que, como se comentó en el apartado de cámaras cuando se habló del filtro de Bayer, o como se pudo observar en la Ecuación I, cuando se trabajan con cámaras se trata de dar mayor peso a la componente verde porque el ojo humano es más sensible a cambios en la misma. En segundo lugar, el LED que ha presentado un mejor comportamiento ha sido el blanco, y en último lugar el azul, algo que es congruente con la Ecuación I.

Estos resultados han sido más favorables en el escenario de tiempo de exposición medio, seguido del tiempo de exposición alto. Esto se debe a que, si bien a mayor tiempo de exposición se recibe más señal, también se incrementa considerablemente el ruido de luminosidad ambiental. No obstante, las diferencias no son significativas entre los 3 tiempos de exposición. Sí que son más notorias en función de la longitud de onda. En la Figura 5.2.1.6 se puede observar la evolución de la SNR en el caso optimista para cada uno de los colores con los que se ha trabajado (se ha tomado el LED verde con mejor comportamiento de los dos), y en la Figura 5.2.1.7 la variación de dicha curva para el caso del LED verde que más veces es detectado en función del tiempo de exposición con el que se ha grabado el vídeo.

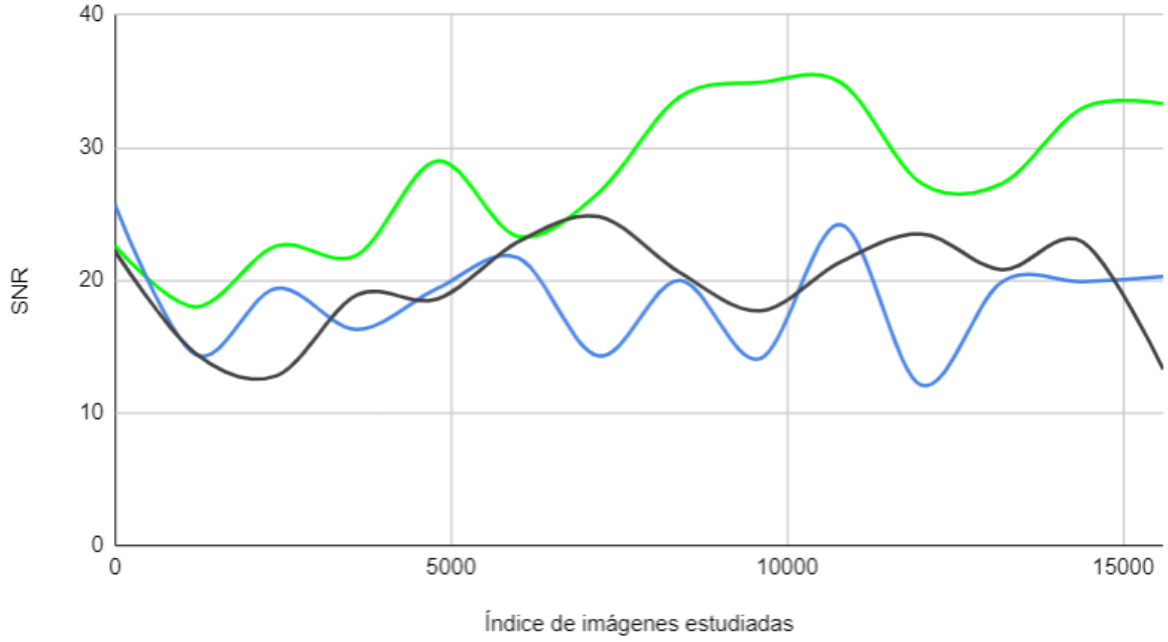

Figura 5.2.1.6: Valores de SNR en el caso indoor con tiempo de exposición elevado para los LEDs verdes (curva verde), azul (curva azul) y blanco (curva negra)

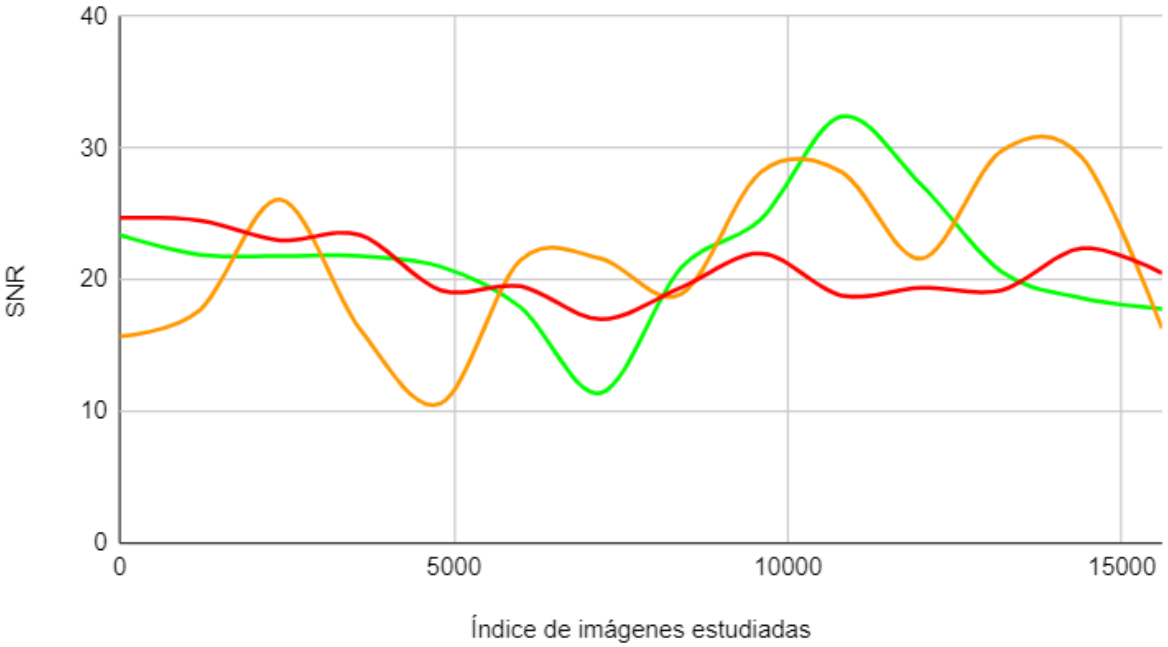

Figura 5.2.1.7: Valores de SNR de uno de los LEDs verdes del escenario indoor en el vídeo tomado con tiempo de exposición elevado (curva verde), bajo (curva roja) y media (curva naranja)

Como se aprecia en las gráficas, la SNR varía en menor medida al usar un tiempo de exposición bajo. Esto se debe a que al obturador estar abierto un menor tiempo, menor es la cantidad de luz

ambiental que incide en el sensor de imagen y por tanto menor es la influencia de la luz ambiental en los resultados. En promedio, para el LED verde en el mejor de los casos la SNR ha sido de 28, 15 dB, y la tasa de error de bit de 4,88E-33. Por su parte, el LED azul en el peor de los casos ha tenido una SNR promedio de 19,3 dB y una tasa de error de bit media de 1,64E-20. Un mayor detalle de los resultados obtenidos se puede apreciar en el Anexo A.

En cuanto a los tiempos de procesamiento, la fase de simplificación ha tenido un tiempo promedio de 62 ms. Por su parte, los tiempos de correlación medios fueron de 600 ms. Estos tiempos fueron mayores en el caso del bajo tiempo de exposición, seguidos del tiempo de exposición medio y alto. Esto se debe a que conforme disminuye el tiempo de exposición, disminuye el tamaño de las regiones luminosas que podrían considerarse transmisores, y por tanto mayor es la cantidad de candidatos que salen del bloque simplificador a menor tiempo de exposición. Además, estos tiempos son tan considerablemente altos debido a la elevada cantidad de elementos reflectantes en el escenario. Los candidatos resultantes del bloque simplificador oscilan en este escenario los 17.000.

5.2.2 Escenario outdoor diurno

Como se comentó anteriormente, en el caso outdoor diurno se grabaron dos vídeos utilizando una sensibilidad ISO de 40 y dos tiempos de exposición diferentes: 0,45 y 0,8 ms. Una imagen asociada a cada caso se puede apreciar en la Figura 5.2.2.1. Además, en el caso de la imagen de la derecha, la posición de cada transmisor ha sido marcada por el algoritmo.

Figura 5.2.2.1: Escenario outdoor diurno con tiempo de exposición de 0,45 ms (izquierda) y 0,8 ms (derecha)

En total se analizaron 24 muestras de vídeo en las que se detectaron todos los transmisores en el 27,3 % de los casos, solo el 75% de los transmisores en el 31,8 % y la mitad en el 40,9 % de los casos.

Los mejores resultados se obtuvieron en el caso del tiempo de exposición más alto, tal y como se puede apreciar en la Tabla 5.2.2.1.

Casos	Tiempo de exposición alto	Tiempo de exposición bajo
Se detectaron todos	54,55 %	0 %
Se detectaron 3/4	45,45 %	18,18 %
Se detectaron 2/4	0 %	81,82 %

Tabla 5.2.2.1: Tasas de detección en escenario outdoor diurno

El motivo de estas tasas de omisión de positivos tan elevadas se debe a las mismas razones que en el caso del escenario indoor. La diferencia de potencia lumínica entre transmisores, sobretodo en el caso del vídeo con tiempo de exposición bajo, conlleva que usando un único umbral de luminosidad una porción de los transmisores no se vaya a detectar. En este escenario ha habido una clara distinción de nivel de intensidad luminosa entre los dos LEDs centrales, de color azul, y los LEDs laterales, de color rojo. Esto se puede observar graficamente en la Figura 5.2.2.2, donde se representa gráficamente la señal óptica recibida del LED azul (izquierda) y rojo (derecha) en el vídeo con tiempo de exposición elevado. Como se observa, la fuente óptica roja tiene menor luminosidad y es más ruidosa. Por ello, dado que la luminosidad de los LEDs rojos era muy cambiante e inestable, en la Ecuación III se consideraron únicamente los valores de luminosidad de los LEDs azules para obtener la función. Con ello se buscaba garantizar al menos la detección de los LEDs azules, dado que su comportamiento era más fiable. No obstante, como consecuencia, en el vídeo con tiempo de exposición bajo casi no se detectan los LEDs rojos en ninguna ocasión. El único en ser detectado de los dos fue el más cercano al receptor.

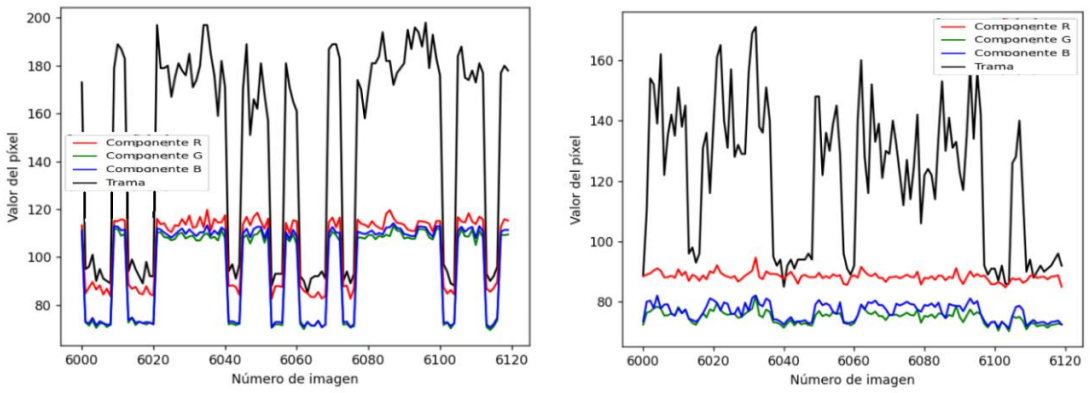

Figura 5.2.2.2: Diferencia de la señal óptica recibida del LED azul (izquierda) y rojo (derecha) en el vídeo con tiempo de exposición elevado

En cuanto a los falsos positivos, la tasa es nula en ambos vídeos gracias a la baja cantidad de elementos reflectantes y a la ausencia de personas que pudieran obstaculizar.

En lo referente a la relación señal a ruido y la tasa de error de bit, como cabe esperar por lo comentado, los LEDs azules son los que presentaron un mejor comportamiento. Este comportamiento fue más favorable en el caso del tiempo de exposición bajo. Esto se debe a que en este escenario el ruido de la luz ambiente es notorio, lo que produce un deterioro de la señal conforme más luz se deja pasar a la cámara. No obstante, usar un bajo tiempo de exposición provoca que ni siquiera algunos transmisores puedan ser detectados, como se pudo comprobar en la Tabla 5.2.2.1. En promedio, la SNR de los LEDs azules fue de 27,8 dB en el caso del tiempo de exposición bajo, y de 20,8 dB en el caso del tiempo de exposición alto. Por su parte, la tasa de error media en el caso del tiempo de exposición bajo fue de 5,5E-76, y de 4,51E-30 en el caso del tiempo de exposición alto.

En cuanto a los tiempos de procesamiento, debido a la menor cantidad de superficies y objetos reflectantes, se redujo a los 29 ms en el caso del bloque simplificador, y los 580 ms en el caso del bloque correlador. En este caso las desviaciones de esos tiempos de operación fueron menores debido a que los cambios en el entorno eran mínimos.

5.2.3 Escenario outdoor nocturno

En el escenario outdoor nocturno únicamente se grabó un vídeo con una sensibilidad ISO de 500 y un tiempo de exposición de 30 ms. En este caso se analizaron 13 fragmentos de vídeo, en los que se detectaron el 100 % de los transmisores en el 100 % de los casos. Además, la tasa de falsos positivos fue de 0, debido a la nula cantidad de otros elementos luminosos o reflectantes en el escenario. En la Figura 5.2.3.1 se pueden observar todos los transmisores etiquetados en este escenario.

Figura 5.2.3.1: Ejecución del algoritmo en el resultado oudoor nocturno

En cuanto a la relación señal a ruido y la tasa de error de bit, en este caso ambos parámetros apenas varían a lo largo del vídeo debido a la ausencia de ruido (luz ambiental). Los valores medios fueron bastante similares entre transmisores, siendo la SNR promedio de 21,3 dB y la BER promedio de 5,13 E-11.

Por último, en lo referente a los tiempos de procesamiento, debido a la ausencia de otras fuentes luminosas, este fue el escenario en el que mejores resultados se obtuvieron, siendo el tiempo tanto del bloque de simplificación como de correlación de 20 ms.

Capítulo 6

6. Conclusiones

En este capítulo de la memoria se realiza una evaluación de los resultados y se presentan las conclusiones obtenidas a partir de los mismos. Finalmente, se explican posibles maneras de mejorar el algoritmo.

6.1 Conclusiones

En el presente trabajo se ha desarrollado y validado experimentalmente el desarrollo de un algoritmo de detección de transmisores ópticos sub-píxel, en el marco de las comunicaciones ópticas basadas en cámaras. El algoritmo desarrollado se divide en cuatro bloques principales: adquisición de imágenes, simplificación de posibles transmisores en base a su luminosidad y tamaño de la región luminosa, correlación de valores de la señal óptica con valores teóricos esperables según la trama utilizada, y eliminación de píxeles vecinos. Para reducir los tiempos de ejecución del algoritmo, en los dos primeros bloques comentados se utilizaron funciones de multiprocessing, y para facilitar el desarrollo del código se programó aprovechando las múltiples bibliotecas y módulos del lenguaje Python.

Para validarlo, se implementó un sistema de comunicaciones OCC en distintos escenarios y se grabaron las comunicaciones utilizando tiempos de exposición diferentes. Se utilizaron cuatro transmisores en total, y se trabajó con distintas longitudes de onda en los distintos escenarios. De las grabaciones se estudiaron múltiples fragmentos en cada escenario para mostrar la validez del algoritmo independientemente del payload transmitidos en cada trama.

Los resultados mostraron una mayor tasa de detección de positivos verdaderos cuanto menor era la cantidad de luz ambiental presente en el escenario, detectándose el 100%, el 89% y el 72% de los transmisores en el escenario nocturno, diurno indoor, y diurno outdoor respectivamente. La tasa de omisión de positivos estuvo directamente relacionada con esa luz ambiente, que produjo que la potencia recibida por los distintos transmisores presentase fluctuaciones elevadas con el tiempo. Como consecuencia de ello, los valores de SNR y BER fueron más constantes cuanto menor fue el tiempo de exposición y la luz ambiente en los distintos escenarios, tal y como se pudo observar en la

Figura 5.2.1.7. Además, estos valores fueron más favorables en el caso del tiempo de exposición bajo en el escenario outdoor diurno debido al enorme peso en la SNR del ruido ambiental en esas condiciones. En el caso indoor, los mejores resultados en esos dos parámetros se obtuvieron en el caso de tiempo de exposición medio, que supuso un compromiso entre permitir captar la luz del transmisor y evitar demasiada luz ambiental.

La SNR y la tasa de error de bit también tuvieron diferencias notables en función de la longitud de onda de trabajo, obteniéndose los mejores resultados al utilizar longitudes de ondas próximas al verde, dada la predisposición de las cámaras a captar este color debido al comportamiento del ojo humano. También cabe destacar positivamente los resultados obtenidos con los LED de color azul, pese a que estos no tengan gran peso en la Ecuación I.

En cuanto a las tasas de falsos positivos, estas estuvieron principalmente relacionadas con la presencia de superficies u objetos reflectantes. Ello produjo que en los escenarios outdoor no se detectaran falsos positivos, pero que en el escenario indoor esta cifra fuera similar a la de positivos verdaderos. No obstante, se pudo comprobar cómo este hecho no es negativo ya que permite disponer de mayor redundancia en la recepción de datos y obtener los mismos aunque haya algo que impida recuperar la información directamente del transmisor.

En lo referente al tiempo de procesamiento, está principalmente relacionado también con la presencia de superficies u objetos reflectantes. Una mayor cantidad de los mismos conlleva una mayor cantidad de elementos de salida del bloque simplificador, y unos tiempos muy elevados de la etapa de correlación. Si bien es posible aplicar también técnicas de multiprocessing a este último bloque, las pruebas llevadas a cabo al respecto produjeron tiempos de procesamiento aún mayores debido a la necesidad de dividir la lista en múltiplos de la cantidad de procesadores, y el ordenamiento posterior de los diferentes resultados del multiprocessing. Sin embargo, los tiempos se consideran aceptables dado que el algoritmo está principalmente orientado a redes de sensores, donde no suelen predominar las exigencias a nivel temporal. En promedio, el tiempo de ejecución del algoritmo fue de unos 390 ms.

Si bien los resultados del algoritmo desarrollado se consideran aceptables, estos pueden ser mejorados. Para ello, es necesario la grabación de las comunicaciones en nuevos escenarios con nuevas condiciones ambientales. El estudio de esas grabaciones permitiría un mejor ajuste de las funciones polinómicas definidas en II y III, así como la obtención de resultados con mayor precisión en nuevos entornos. Asimismo, permitiría definir con mayor certeza qué longitudes de ondas son más favorables en función del entorno, y que distancias máximas pueden definirse en función de

las condiciones ambientales.

Disponer de una mayor cantidad de grabaciones de las comunicaciones podría posibilitar el uso de inteligencia artificial en el algoritmo. Se podría utilizar, o bien para estimar los umbrales utilizados en el algoritmo, o bien para realizar directamente el descubrimiento. En este sentido, las redes neuronales convolucionales (CNN - *Convolutional Neural Networks*) han demostrado ser capaces de extraer características relevantes de las imágenes y aprender patrones complejos [56].

También resultaría beneficioso tomar vídeos en un mismo escenario y con un mismo tiempo de exposición, variando la distancia entre transmisor y receptor para encontrar una expresión matemática que, además de relacionar el umbral de ROI con la luminosidad media de la imagen, lo hiciese también con la distancia mínima del enlace. No obstante, como en este trabajo únicamente se trata la casuística sub-píxel, no se ha llevado a cabo esa parte de la experimentación.

En base a los resultados, se definen como entornos más favorables para la implementación de este tipo de sistemas, aquellos en los que las condiciones lumínicas sean bajas y no presenten cambios considerables con el tiempo. Asimismo, la existencia de superficies y objetos reflectantes puede resultar beneficioso, especialmente si se estipula la presencia de obstáculos en el escenario. Por lo tanto, los entornos indoor y outdoor nocturnos son ideales para este tipo de sistemas de comunicaciones. Por su parte, trabajar con LEDs del mismo tipo y longitud de onda (preferiblemente verdes o azules por los resultados obtenidos), también es recomendable para evitar que el umbral usado los descarte por la diferencia de potencia lumínica entre los mismos. No obstante, también es posible usar otro tipo de colores y ajustar las Ecuaciones II y III al caso de interés.

Dependiendo del entorno y de la probabilidad de obstaculización de los transmisores, podría ser aconsejable que los dispositivos transmisores enviaran su información en múltiples ocasiones para aumentar la probabilidad de que sean detectados. En base a ello, del lado del receptor se podría intentar realizar el descubrimiento aplicando distintos umbrales en distintos instantes de tiempo. Por ejemplo, en primer lugar, se podrían seguir las Ecuaciones II y III, y posteriormente probar con valores del $\pm$ 20 % de ese valor. En cada uno de los intentos de descubrimiento se podrían encontrar transmisores cuyas potencias de recepción presentasen diferencias marcadas.

Para finalizar, recordar algunas aplicaciones prácticas del algoritmo desarrollado, como pueden ser la monitorización sensórica en invernaderos, hospitales, museos, hoteles, supermercados, o en las calles de Smart Cities.

Parte II - Bibliografía

Bibliografía

[1] "IEEE Standard for Local and metropolitan area networks--Part 15.7: Short-Range Optical Wireless Communications," in IEEE Std 802.15.7-2018 (Revision of IEEE Std 802.15.7-2011) , vol., no., pp.1-407, 23 April 2019, doi: 10.1109/IEEESTD.2019.8697198.

[2] B. Majlesein, V. Matus, C. Jurado-Verdu, V. Guerra, J. Rabadan and J. Rufo, "Experimental characterization of sub-pixel underwater optical camera communications," 2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), 2022, pp. 150-155, doi: 10.1109/CSNDSP54353.2022.9907903.

[3] Durai Rajan Dhatchayeny and Yeon Ho Chung, "Optical extra-body communication using smartphone cameras for human vital sign transmission," Appl. Opt. 58, 3995-3999 (2019) https://opg.optica.org/ao/abstract.cfm?URI=ao-58-15-3995.

[4] Teli, S.R.; Zvanovec, S.; Perez-Jimenez, R.; Ghassemlooy, Z. Spatial frequency-based angular behavior of a short-range flicker-free MIMO–OCC link. OSA Appl. Opt. 2020, 59, 10357–10368.

[5] Teli, S.R.; Matus, V.; Zvanovec, S.; Perez-Jimenez, R.; Vitek, S.; Ghassemlooy, Z. The First Study of MIMO Scheme Within Rollingshutter Based Optical Camera Communications. In Proceedings of the 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Porto, Portugal, 20–22 July 2020; pp. 1–5.

[6] Matus V, Guerra V, Jurado-Verdu C, Zvanovec S, Perez-Jimenez R. "Wireless Sensor Networks Using Sub-Pixel Optical Camera Communications: Advances in Experimental Channel Evaluation." Sensors (Basel). 2021 Apr 13;21(8):2739. doi: 10.3390/s21082739. PMID: 33924508; PMCID: PMC8069996.

[7] P. Chávez, J. Rabadan and R. Perez-Jimenez, "Optical Camera Communication for Internet of Things in Urban Environments", 2021, ULPGC, Tesis Doctoral.

[8] V. Matus, V. Guerra, C. Jurado-Verdu, S. Zvanovec, J. Rabadan and R. Perez-Jimenez, "Design and Implementation of an Optical Camera Communication System for Wireless Sensor Networking in Farming Fields," 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2021, pp. 1-6, doi: 10.1109/PIMRC50174.2021.9569653.

[9] N. Chaudhary, O. I. Younus, L. N. Alves, Z. Ghassemlooy, S. Zvanovec, and H. Le-Minh, "An indoor visible light positioning system using tilted leds with high accuracy," Sensors, vol. 21, no. 3, 2021.

[10] M. Karbalayghareh, F. Miramirkhani, H. B. Eldeeb, R. C. Kizilirmak, S. M. Sait, and M. Uysal, “Channel modelling and performance limits of vehicular visible light communication systems,” IEEE Transactions on Vehicular Technology, vol. 69, no. 7, pp. 6891–6901, 2020.

[11] A. L. R. Gonçalves, Á. H. A. Maia, M. R. Santos, D. A. de Lima, and A. de Miranda Neto, “Visible light positioning and communication methods and their applications in the intelligent mobility,” IEEE Latin America Transactions, vol. 100, no. 1e, 2021.

[12] R. M. Marè, C. L. Marte, C. E. Cugnasca, O. G. Sobrinho, and A. S. dos Santos, “Feasibility of a testing methodology for visible light communication systems applied to intelligent transport systems,” IEEE Latin America Transactions, vol. 100, no. 1e, 2020.

[13] E. Eso, S. Teli, N. B. Hassan, S. Vitek, Z. Ghassemlooy, and S. Zvanovec, “400 m rolling-shutter-based optical camera communications link,” Opt. Lett., vol. 45, pp. 1059–1062, Feb 2020

[14] P. Chavez-Burbano, V. Guerra, J. Rabadan, and R. Perez-Jimenez, “Optical camera communication for smart cities,” in 2017 IEEE/CIC International Conference on Communications in China (ICCC Workshops), pp. 1–4, 2017.

[15] V. Matus, V. Guerra, C. Jurado-Verdu, J. Rabadan and R. Perez-Jimenez, "Demonstration of a Sub-Pixel Outdoor Optical Camera Communication Link," in IEEE Latin America Transactions, vol. 19, no. 10, pp. 1798-1805, Oct. 2021, doi: 10.1109/TLA.2021.9477281.

[16] D. Sisodiya and D. Sipal, "Long Range and Low Cost WDM OWC System Based on Hybrid Configuration of Optical Amplifiers," 2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2022, pp. 1-5, doi: 10.1109/ICAECT54875.2022.9807650.

[17] Dataintelo. (2021). Free Space Optics (FSO) and Visible Light Communication (VLC) Market Research Report 2021-2028. [Online]. Available: https://dataintelo.com/report/free-space-optics-fso-and-visible-light-communication-vlc-market/#:~:text=a%2025%25%20share.-,Free%20Space%20Optics%20(FSO)%20and%20Visible%20Light%20Communication%20.

[18] R. Pérez Jiménez, "Tema 2: Transmisores", en Comunicaciones Ópticas, Universidad de Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, 2021.

[19] Iberdrola, "LiFi Technology," Iberdrola Innovation, [Online]. Available: https://www.iberdrola.com/innovation/lifi-technology#:~:text=With%20LiFi%2C%20however%2C%20its%20band%20frequency%20of%20200%2C000,the%20best%20wifi%20would%20barely%20reach%20300%20Mbit%2Fs.

[20] R. Arellano, "Título del TFG", TFG, Universidad Oberta de Catalunya, Barcelona, España, junio de 2019. [Online]. Disponible: https://openaccess.uoc.edu/bitstream/10609/95748/7/rarellanoTFG0619memoria.pdf

[21] F. González del Tánago Landín, "Título del documento", Tesis, Universidad de Vigo, Vigo, España. [Online]. Disponible:

http://calderon.cud.uvigo.es/bitstream/handle/123456789/273/Gonz%C3%A1lez%20del%20T%C3%A1nago%20Land%C3%ADn%2C%20Fernando%20-%20Memoria.pdf?sequence=1&isAllowed=y

[22] "Mirando al cielo: fenómenos ópticos atmosféricos (y II)", Revista al Aficionado a la Meteorología (RAM), no. 3, septiembre 2002. [Online]. Disponible: https://www.tiempo.com/ram/numero3/pdf/fenopticos.pdf

[23] J. D. Sánchez López, "Estudio teórico-experimental de un sistema de comunicaciones ópticas homodino utilizando el canal inalámbrico turbulento", Tesis doctoral, Centro de Investigación Científica y de Educación Superior de Ensenada, Programa de Posgrado en Ciencias en Electrónica y Telecomunicaciones, Ensenada, Baja California, México, octubre 2009.

[24] B. M. Cochenour, L. J. Mullen and A. E. Laux, "Characterization of the Beam-Spread Function for Underwater Wireless Optical Communications Links," in IEEE Journal of Oceanic Engineering, vol. 33, no. 4, pp. 513-521, Oct. 2008, doi: 10.1109/JOE.2008.2005341.

[25] H. M. Oubei, R. T. ElAfandy, K. -H. Park, T. K. Ng, M. -S. Alouini and B. S. Ooi, "Performance Evaluation of Underwater Wireless Optical Communications Links in the Presence of Different Air Bubble Populations," in IEEE Photonics Journal, vol. 9, no. 2, pp. 1-9, April 2017, Art no. 7903009, doi: 10.1109/JPHOT.2017.2682198.

[26] V. K. Yadav, V. Sharma, G. Prajapat, J. Kundu, A. Sharma and N. K. Gupta, "A Review on Light-Emitting-Diodes Characteristics," 2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP), Hyderabad, India, 2022, pp. 1-6, doi: 10.1109/ICICCSP53532.2022.9862375.

[27] M. Wan, "OLED frente a iluminación LED", Sitio web, Actualizado: 28 de noviembre. [Online]. Disponible: https://www.ledyilighting.com/es/oled-vs-led-lighting/#:~:text=Los%20OLED%20son%20diodos%20emisores,iluminaci%C3%B3n%20brillante%20sobre%20un%20%C3%A1rea.

[28] R. D. Roberts, "Undersampled frequency shift ON-OFF keying (UFSOOK) for camera communications (CamCom)," 2013 22nd Wireless and Optical Communication Conference, Chongqing, China, 2013, pp. 645-648, doi: 10.1109/WOCC.2013.6676454.

[29] NVC Lighting, "What is Dimming?", Technical Support, NVC Lighting, [Online]. Disponible: https://www.nvcuk.com/technical-support/view/what-is-dimming-11

[30] S. Chevalier Naranjo, "¿Qué tan vigiladas están las grandes metrópolis del mundo? Vigilancia por CCTV", Statista, 2021. [Online]. Disponible: https://es.statista.com/grafico/23979/camaras-de-seguridad-por-km2-en-una-seleccion-de-ciudades/

[31] R. Fernández, "La telefonía móvil en España - Datos estadísticos", Statista, diciembre de 2022. [Online]. Disponible: https://es.statista.com/temas/5458/la-telefonia-movil-en-espana/#topicOverview.

[32] Kuroda, T. (2015). Essential Principles of Image Sensors (1st ed.). CRC Press. https://doi.org/10.1201/b17411

[33] C. M. Jurado Verdú, J. A. Rabadán Borges, V. Guerra Yanez, "Diseño, caracterización e implementación de un sistema de comunicaciones ópticas basado en cámara", Trabajo final de máster, Escuela de Ingeniería de Telecomunicación y Electrónica, Departamento de Señales Y Comunicaciones, Universidad de Las Palmas de Gran Canaria, 2019. [Online]. Disponible: http://hdl.handle.net/10553/77565.

[34] P. Luo, Z. Ghassemlooy, H. Le Minh, X. Tang and H. -M. Tsai, "Undersampled phase shift ON-OFF keying for camera communication," 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP), Hefei, China, 2014, pp. 1-6, doi: 10.1109/WCSP.2014.6992043.

[35] C.-S. Gong, Y.-C. Lee, J.-L. Lai, C.-Y. Yang, et al., "The High-efficiency LED Driver for Visible Light Communication Applications," Aug. 2016. [En línea]. Disponible: https://www.researchgate.net/figure/Variable-pulse-position-modulation-PPM-scheme-with-different-PWM-dimming-levels_fig8_306009195.

[36] J. He, Y. Yang and J. He, "Artificial Neural Network-Based Scheme for 4-PWM OCC System," in IEEE Photonics Technology Letters, vol. 34, no. 6, pp. 333-336, 15 March15, 2022, doi: 10.1109/LPT.2022.3153692.

[37] H. Nguyen, M. D. Thieu, T. Nguyen and Y. M. Jang, "Rolling OFDM for Image Sensor Based Optical Wireless Communication," in IEEE Photonics Journal, vol. 11, no. 4, pp. 1-17, Aug. 2019, Art no. 6500817, doi: 10.1109/JPHOT.2019.2926394.

[38] M. Rêgo and P. Fonseca, "OCC Based Indoor Positioning System Using a Smartphone Camera," 2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Santa Maria da Feira, Portugal, 2021, pp. 31-36, doi: 10.1109/ICARSC52212.2021.9429782.

[39] M. T. Hossan, M. Z. Chowdhury, A. Islam and Y. M. Jang, "Simplified photogrammetry using optical camera communication for indoor positioning," 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 2017, pp. 126-130, doi: 10.1109/ICUFN.2017.7993761.

[40] M. S. Ifthekhar, N. Saha and Y. M. Jang, "Neural network based indoor positioning technique in optical camera communication system," 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Busan, Korea (South), 2014, pp. 431-435, doi: 10.1109/IPIN.2014.7275513.

[41] E. Jeong, B. -s. Seo, N. Kim, D. You, D. H. Kim and Y. H. Lee, "An indoor positioning method in a concert hall using optical camera communication (OCC) technology," 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2017, pp. 970-972, doi: 10.1109/ICTC.2017.8190828.

[42] D. T. Nguyen, S. Park, Y. Chae and Y. Park, "VLC/OCC Hybrid Optical Wireless Systems for Versatile Indoor Applications," in IEEE Access, vol. 7, pp. 22371-22376, 2019, doi: 10.1109/ACCESS.2019.2898423.

[43] M. F. Ahmed, M. K. Hasan, M. Z. Chowdhury, N. C. Hoan and Y. M. Jang, "Continuous Status Monitoring of Industrial Valve Using OCC-Enabled Wireless Sensor Network," in IEEE Transactions on Instrumentation and Measurement, vol. 71, pp. 1-10, 2022, Art no. 5501010, doi: 10.1109/TIM.2021.3130292.

[44] M. Shahjalal, M. K. Hasan, M. Z. Chowdhury and Y. M. Jang, "Future Optical Camera Communication Based Applications and Opportunities for 5G and Beyond," 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Okinawa, Japan, 2019, pp. 492-495, doi: 10.1109/ICAIIC.2019.8669075.

[45] M. F. Ahmed, M. K. Hasan, M. Shahjalal, M. M. Alam and Y. M. Jang, "Design and Implementation of an OCC-Based Real-Time Heart Rate and Pulse-Oxygen Saturation Monitoring System," in IEEE Access, vol. 8, pp. 198740-198747, 2020, doi: 10.1109/ACCESS.2020.3034366.

[46] M. K. Hasan, M. Shahjalal, M. Z. Chowdhury and Y. M. Jang, "Access Point Selection in Hybrid OCC/RF eHealth Architecture for Real-Time Remote Patient Monitoring," 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2018, pp. 716-719, doi: 10.1109/ICTC.2018.8539582.

[47] M. Z. Chowdhury, M. T. Hossan, M. Shahjalal, M. K. Hasan and Y. M. Jang, "A New 5G eHealth Architecture Based on Optical Camera Communication: An Overview, Prospects, and Applications," in IEEE Consumer Electronics Magazine, vol. 9, no. 6, pp. 23-33, 1 Nov. 2020, doi: 10.1109/MCE.2020.2990383.

[48] L. Aguiar, P. de Saa, V. Guerra and R. Perez-Jimenez, "Survey of VLC and OCC Applications on Tourism Industry: Potentials & Challenges," 2020 South American Colloquium on Visible Light Communications (SACVC), Santiago, Chile, 2020, pp. 1-6, doi: 10.1109/SACVLC50805.2020.9129867.

[49] C. H. Nguyen, V. H. Nguyen and Y. M. Jang, "Optical camera comunication (OCC) applications for Internet of Vehicle (IoV)," 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2019, pp. 512-514, doi: 10.1109/ICTC46691.2019.8939837.

[50] M. K. Hasan, M. O. Ali, M. H. Rahman, M. Z. Chowdhury and Y. M. Jang, "Optical Camera Communication in Vehicular Applications: A Review," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6260-6281, July 2022, doi: 10.1109/TITS.2021.3086409.

[51] N. Devulapalli, V. Matus, E. Eso, Z. Ghassemlooy and R. Perez-Jimenez, "Lane-cross Detection using Optical Camera-based Road-to-Vehicle Communications," 2021 17th International Symposium on Wireless Communication Systems (ISWCS), Berlin, Germany, 2021, pp. 1-5, doi: 10.1109/ISWCS49558.2021.9562196.

[52] B. Majlesein, A. Gholami and Z. Ghassemlooy, "The Channel Impulse Response of SIMO Underwater Optical Wireless Communication Link based on Monte Carlo Simulation," 2019 2nd West Asian Colloquium on Optical Wireless Communications (WACOWC), Tehran, Iran, 2019, pp. 69-73, doi: 10.1109/WACOWC.2019.8769998.

[53] M. S. M. Akram, L. G. D. Aravinda, M. K. P. D. Munaweera, G. M. R. I. Godaliyadda and M. P. B. Ekanayake, "Camera based visible light communication system for underwater applications," 2017 IEEE International Conference on Industrial and Information Systems (ICIIS), Peradeniya, Sri Lanka, 2017, pp. 1-6, doi: 10.1109/ICIINFS.2017.8300377.

[54] SoftZone, "Programar Arduino: Guía completa para principiantes y expertos," Recuperado de https://www.softzone.es/programas/lenguajes/programar-arduino/

[55] "OpenCV: Color conversions," OpenCV, Disponible en: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

[56] W. Bao, X. Zhang, S. Yan and Z. Gao, "Iterative convolutional neural network for noisy image super-resolution," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 4038-4042, doi: 10.1109/ICIP.2017.8297041.

[57] Universidad de Las Palmas de Gran Canaria, "Retribuciones del Personal Investigador Contratado 2022," Universidad de Las Palmas de Gran Canaria, 2022. [En línea]. Disponible en: https://www.ulpgc.es/sites/default/files/ArchivosULPGC/transparencia/Personal/2022/Retribuciones/2022_retribuciones_pi_contratado_ulpgc.pdf.

Parte III – Presupuesto

Presupuesto

P.1 Introducción

En esta parte del documento se describe el coste económico asociado al desarrollo de este TFM.
Los aspectos del trabajo que han contribuido a dicho coste son:

- Recursos materiales.
    - Recursos hardware.
    - Recursos software.
- Recursos humanos.

P.2 Recursos materiales

Como recursos materiales se han utilizado distintos elementos físicos y software. En cuanto a los primeros incluyen los equipos utilizados en la fase de experimentación (una Raspberry Pi, cuatro Arduinos Nano 33, cuatro LEDs, una cámara Full HD, un teclado, y un ratón) y los equipos utilizados para la fase de búsqueda de documentación, postprocesamiento y redacción (ordenador portátil). En cuanto a los recursos software únicamente se ha utilizado el IDE Visual Studio Code para la programación y evaluación del código, y el paquete Office 365 para la redacción del documento y el manejo de datos.

En la Tabla P.2.1 se resumen los costes asociados a los elementos comentados. En la misma se usa la Ecuación VI para obtener los costes finales prorrateados, donde *Vad* es el valor de adquisición, *Vres* el valor residual (valor de los elementos al transcurrir su vida útil), *Aútil* los años de vida útil del equipo y *Autilización* el tiempo en años durante el cual se ha utilizado.

$$Cf = \frac{(Vad - Vres) \cdot Autilización}{Aútil} (VI)$$

Recursos Materiales							
Recursos físicos							
Descripción	Cantidad	Coste unidad (€)	Tiempo de uso (años)	Coste total (€)	Valor residual (€)	Vida útil (años)	Total (€)
Raspberry Pi v3	1	85	0,08	85	20	4	1,35
Arduino Nano 33	3	31	0,08	93	10	4	1,31
LED RGB	3	0,85	0,08	2,55	0,1	3	0,06
LED blanco	1	0,7	0,08	0,7	0,1	3	0,02
Teclado	1	18,78	0,08	18,78	6	3	0,36
Ratón	1	11,25	0,08	11,25	4	3	0,20
Cámara Full HD módulo V2	1	10,2	0,08	10,2	3	4	0,15
Ordenador portátil Acer Extensa	1	450	0,33	450	100	4	29,17
Recursos software							
Descripción	Cantidad	Coste unidad (€)	Tiempo de uso (meses)	Coste total (€)	Valor residual (€)	Vida útil (años)	Total (€)
Visual Studio Code	1	0	3	0	-	-	
Paquete Office 365	1	7	4	7	-	-	28
Total							60,62

Tabla P.2.1: Costes asociados a recursos materiales

El coste total de los recursos materiales es de SESENTA EUROS CON SESENTA Y DOS CÉNTIMOS.

P.3 Recursos humanos

Los recursos humanos hacen referencia al sueldo asociado al personal encargado del proyecto. Dado que el trabajo ha sido realizado dentro del entorno de investigación de la ULPGC, el coste mensual por los trabajos realizados se debe obtener de los documentos oficiales que regulan este tipo de contratos en la universidad. Concretamente, el dato se obtiene de la tabla ”CLASIFICACIÓN Y RETRIBUCIÓN DEL PERSONAL CONTRATADO CON CARGO A PROYECTOS, PROGRAMAS, CONVENIOS Y CONTRATOS” del BOULPGC del 3 de febrero de 2023 [57]. Según dicha tabla, el sueldo mensual del personal técnico con formación de grado o equivalente que trabaja 30 horas semanales es el que se indica en la Tabla P.3.1. Los costes asociados a recursos humanos se obtienen multiplicando los meses de trabajo por el coste mensual correspondiente.

Recursos Humanos			
Personal	Coste total mensual	Tiempo (meses)	Total
1	1.090,50€	4	4.362,00€

Tabla P.3.1: Costes asociados a recursos humanos

El coste total de los recursos humanos es de CUATRO MIL TRESCIENTOS SESENTA Y DOS EUROS.

P.4 Aplicación de impuestos y coste final

A los gastos totales considerados es necesario añadir el Impuesto General Indirecto Canario (IGIC). En la Tabla P.4.1 se recogen los gastos comentados y los gastos totales finales.

Presupuesto final	
Partidas	**Totales**
Recursos hardware	32,62
Recursos software	28,00
Recursos humanos	4362,00
Total parcial	4422,62
IGIC (7%)	309,58
Total parcial	4732,20

Tabla P.4.1: Costes totales finales tras impuestos

Los costes totales finales del TFM son de CUATRO MIL SETECIENTOS TREINTA Y DOS EUROS CON VEINTE CÉNTIMOS.

P.5 Declaración jurídica

Doña Idaira Rodríguez Yánez, autora del presente Trabajo Fin de Máster, declara que: El Trabajo Fin de Máster con título “Elaboración de un algoritmo de descubrimiento de transmisores ópticos sub-píxel”, desarrollado en la Escuela de Ingeniería de Telecomunicación y Electrónica de la Universidad de Las Palmas de Gran Canaria, con un peso de 18 ECTS, correspondiente a 450 horas, de trabajo presencial y no presencial, tiene un coste total de cuatro mil setecientos treinta y dos euros con veinte céntimos (4732, 20 €), correspondiente a la suma de las cantidades consignadas y justificadas en los

apartados anteriores. Firmando la presente para que así conste a los efectos oportunos. Las Palmas de Gran Canaria, 25 de julio de 2023.

Autora del TFM:

Idaira Rodríguez Yánez

Parte IV – Anexos

Anexos

En esta sección de la memoria se añade toda aquella información que podría alejar al lector del hilo conductor de la misma. Concretamente, en primer lugar se presentan los códigos más complejos utilizados en el análisis preliminar y la explicación de los mismos. En segundo lugar, se añade el código del script principal del software desarrollado. Por último, se adjuntan las hojas Excel completadas durante la fase de resultados.

A. Códigos del análisis preliminar

A.1 Luminosidad

Para evaluar la luminosidad media de las imágenes y de los píxeles asociados a los transmisores, se utilizó el Código A.1.1, cuyo diagrama de flujo se puede observar en la Figura A.1.1.

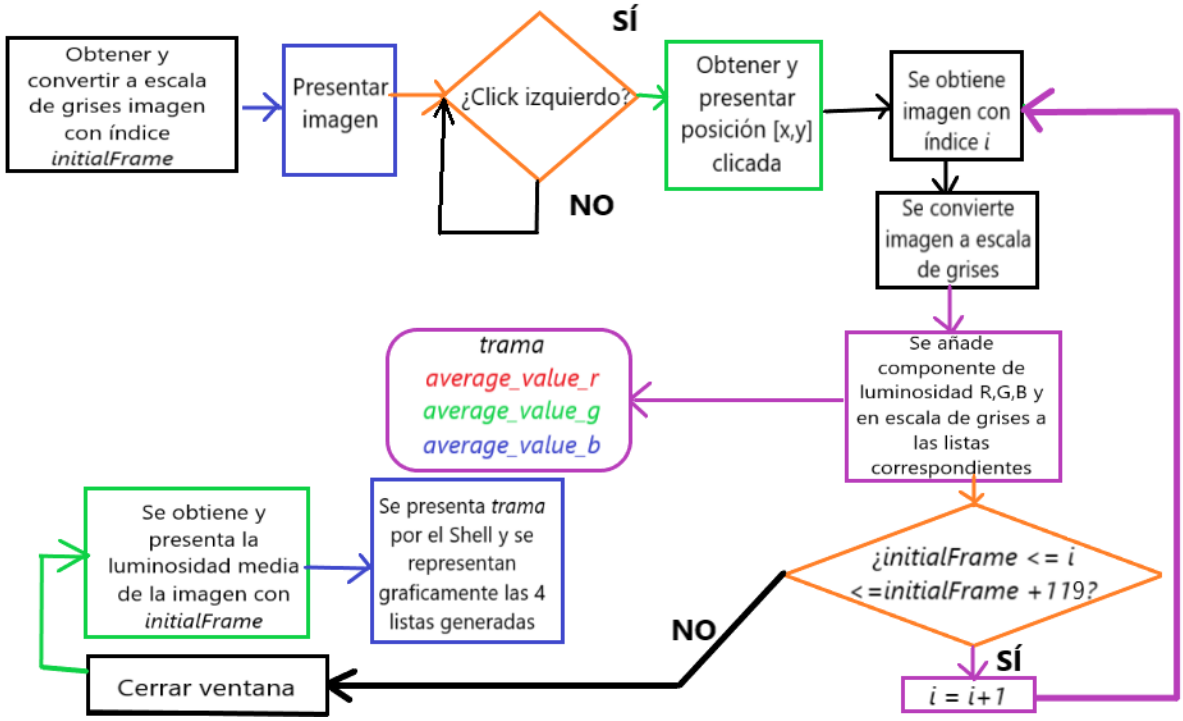

Figura A.1.1: Diagrama de flujo del script *check_frame*

Al comienzo del Código A.1.1, en las líneas 6 y 7 se definen los dos únicos parámetros a configurar: el directorio donde se encuentra el conjunto de imágenes que se desea estudiar y el índice de la imagen

inicial a partir del cual se desea llevar a cabo el estudio. El índice de la imagen final se obtiene teniendo en cuenta que la tasa de adquisición de los vídeos era de 30 FPS, y que se desean estudiar 4 segundos de información. Es decir, se deben considerar 119 imágenes adicionales al índice inicial configurado (línea 10).

Cuando se ejecute el script lo primero que se realizará será presentar la imagen inicial (línea 93) convertida a escala de grises (línea 14) mediante la función *cvtColor* de *cv2*, el módulo principal de OpenCV2. Esto se realiza para que sea más sencillo identificar los LEDs en la imagen. Tras ello, se establece que cada vez que haya un evento de ratón sobre la imagen desplegada se ejecute la función *click_event* (línea 96). En la misma, solo si se ha producido un evento de click izquierdo pulsado (línea 26), se presentan las coordenadas [x, y] del píxel (línea 28) y se obtiene la señal óptica asociada a dicha celda del sensor de imagen. Esto se realiza recorriendo el conjunto de imágenes entre los índices indicados, convirtiendo cada una a escala de grises (línea 36) y añadiendo ese valor al array denominado *trama* (línea 39).

Adicionalmente, también se obtiene el valor promediado de las componentes RGB tanto del píxel central como de sus vecinos con un desplazamiento igual a 2 a izquierda, derecha, arriba y abajo (líneas 44 – 49). Es decir, de cada imagen sin convertir a escala de grises, se obtiene la media que toma la región formada por los píxeles vecinos con una distancia de dos o inferior al píxel seleccionado, de los datos de intensidad luminosa de las componentes roja, verde y azul. Esto se realiza para poder determinar el color del LED y la distorsión que experimenta la señal conforme nos alejamos del píxel central de la ROI.

Tras obtener la señal óptica (trama) y las componentes promedio RGB, se cierra la imagen en escala de grises inicialmente presentada (línea 52), se calcula su luminosidad media (línea 55) y se presenta este valor en pantalla (línea 58). Finalmente, se representan la señal óptica y las componentes RGB en una misma figura, y en la función main se establece que una vez se pulse cualquier botón del teclado, se cerrará dicha ventana y finalizará la ejecución del programa (líneas 99 - 102).

```
1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # Configurar
6  initFrame = 13
7  directory = "indoormedium"
8
9  # Calcular ultima imagen
10 finalFrame = initFrame+119
11 # Obtener primera imagen
```

```
12  frame0 = cv2.imread("{}/frame{}.jpg".format(directory, initFrame))
13  # Convertir a escala de grises
14  gray0 = cv2.cvtColor(frame0, cv2.COLOR_BGR2GRAY)
15
16  # Inicializar listas
17  trama = []
18  average_value_r = []
19  average_value_g = []
20  average_value_b = []
21  shift = 2
22
23  # Configurar la función de callback del ratón
24  def click_event(event, x, y, flags, params):
25      # Chequear si es click del botón izquierdo
26      if event == cv2.EVENT_LBUTTONDOWN:
27          # Presentamos pixel clicado
28          print(x, ' ', y)
29          p = [y, x]
30
31          # Bucle en el que se recorre cada imagen
32          # y se obtienen valores del pixel pulsado
33          for i in range(initFrame, finalFrame):
34              # Se carga la imagen con índice i
35              frame = cv2.imread("{}/frame{}.jpg".format(directory, i))
36              gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
37              # Se crea una trama con los valores de
38              # luminosidad de cada imagen
39              trama.append(gray[y, x])
40
41              # Se calcula el valor medio de las regiones
42              # con un desplazamiento shift alrededor del
43              # pixel pulsado de las componentes R,G,B
44              average_value_r.append(np.mean(
45                  frame[y - shift:y + shift, x - shift:x + shift, 2]))
46              average_value_g.append(np.mean(
47                  frame[y - shift:y + shift, x - shift:x + shift, 1]))
48              average_value_b.append(np.mean(
49                  frame[y - shift:y + shift, x - shift:x + shift, 0]))
50
51          # Se cierran todas las ventanas
52          cv2.destroyAllWindows()
53
54          # Se calcula la media de luminosidad
55          media_luminosidad = cv2.mean(gray)[0]
56
57          # Se muestra la media de luminosidad
58          print("La media de luminosidad es:", media_luminosidad)
59          print(trama)
60
61          # Representamos average_value_r, average_value_g, average_value_b
62          # y trama en una misma figura
63
```

```
64          plt.plot(range(initFrame, finalFrame),
65                   average_value_g, color='green', label='Componente G')
66          plt.plot(range(initFrame, finalFrame),
67                   average_value_b, color='blue', label='Componente B')
68          plt.plot(range(initFrame, finalFrame),
69                   trama, color='black', label='Trama')
70          plt.plot(range(initFrame, finalFrame),
71                   average_value_r, color='red', label='Componente R')
72
73          plt.xlabel("Índice de cada imagen")
74          plt.ylabel("Luminosidad")
75          plt.title("Tramas")
76          plt.legend()
77
78          plt.tight_layout()
79          plt.show()
80
81          # Representamos HISTOGRAMA
82          plt.rcParams["figure.autolayout"] = True
83          (counts, bins, patches) = plt.hist(trama)
84          plt.title("Histograma")
85          plt.xlabel("Luminosidad")
86          plt.ylabel("Densidad")
87          plt.show()
88
89
90
91  if __name__=="__main__":
92      # Se presenta la imagen para que se pueda clicar sobre la misma
93      cv2.imshow('image', gray0)
94
95      # Se configura la función de evento ante un click sobre la imagen
96      cv2.setMouseCallback('image', click_event)
97
98      # Se espera a que se clique una tecla
99      cv2.waitKey(0)
100
101     # Se cierran todas las ventanas
102     cv2.destroyAllWindows()
```

Código A.1.1: Script *check_frames*

A.2 Representación polinómica

Para realizar el ajuste polinómico de los valores obtenidos mediante el Código A.1.1 y el A.3.1, se utilizó el Código que se puede observar a continuación (Código A.2.1).

```python
import numpy as np
import matplotlib.pyplot as plt

# Función que permite obtener el ajuste polinonmico mas apropiado
# dado unos valores x, y, y el grado del polinomio
def polynomial_fit(x, y, degree):

    coefficients = np.polyfit(x, y, degree)
    polynomial = np.poly1d(coefficients)

    return polynomial

if __name__=="__main__":
    # Configurar
    points = [(2.53, 187.9), (20.9, 102.3),
              (75.4, 174.1), (95.2, 244), (146, 170), (193.5, 170)]
    degree = 5

    # Se reformatean los puntos
    x = np.array([point[0] for point in points])
    y = np.array([point[1] for point in points])

    # Se obtiene la función polinómica
    polynomial = polynomial_fit(x, y, degree)

    # Se obtienen los valores de x generando 100 valores de 0 a 250
    x_fit = np.linspace(0, 250, 100)
    # Se obtiene el valor de y asociado a cada valor de x
    y_fit = polynomial(x_fit)

    # Se representa la función graficamente
    plt.scatter(x, y, label='Puntos')
    plt.plot(x_fit, y_fit, label='Ajuste polinómico', color='red')
    plt.xlabel('Luminosidad media de la imagen')
    plt.ylabel('Umbral')
    plt.legend()
    plt.show()

    # Se presenta la función polinómica
    print("La función polinómica es:")
    print(polynomial)
```

Código A.2.1: Script *create_function*

La función principal del Código A.2.1 es *polynomial_fit*. Esta recibe la lista de puntos y el grado del polinomio al que se desea convertir esa secuencia de puntos. La lista de puntos se obtiene mediante dos listas separadas: una contiene los valores de "x" (valor de luminosidad media) y otra los correspondientes valores de "y" (umbral de luminosidad o de ROI). Esas dos listas de valores, junto

con el grado del polinomio, se pasan a la función de *numpy*, *polyfit* (línea 8), que permite obtener los coeficientes de dicho polinomio. Aplicando a continuación la función *poly1d* se consigue el polinomio (línea 9).

Dentro de la función principal *main*, simplemente se genera la lista de puntos (líneas 15 - 16), se especifica el grado deseado del polinomio (línea 17), se separan los valores de "x" e "y" de la lista de puntos en dos listas (líneas 20 - 21), y se llama a la función *polynomial_fit* (línea 24). Luego, se obtienen los valores de x, que oscilan de 0 a 250, siendo 0 el valor que se obtendría en una imagen completamente oscura y 250 el valor que se obtendría si se tuvieran unas condiciones de luminosidad máxima en cada una de las celdas. Finalmente, se obtienen los valores de "y" asociados a esos valores de "x" aplicando la función polinómica obtenida (línea 29), y se representa la función final (líneas 31 - 37).

A.3 Tamaño de la región luminosa

El script desarrollado para obtener la región de interés de cada transmisor se puede observar en el Código A.3.1, y sigue una estructura similar al Código A.1.1. Consta de una función principal en la que se presenta la imagen convertida a escala de grises para facilitar el encuentro de los transmisores, de una función a la que se llama cuando se clica sobre dicha imagen desplegada (método *click_event*), y de una función denominada *find_region_size* que recibe por parámetro las coordenadas de un píxel y devuelve su tamaño asociado. Esta última función es llamada dentro del método *click_event*, tras obtenerse y presentarse por pantalla la posición del click izquierdo del ratón. A diferencia del Código A.1.1, tras ejecutarse la función de obtener la región no se elimina la imagen inicialmente desplegada, sino que se mantiene y se presenta en la misma la información útil: el píxel, y el tamaño de la región a la que pertenece. De esta manera, es mucho más sencillo obtener la región de los 4 transmisores en cada escenario de forma simultánea.

Para obtener el tamaño de cada región luminosa, primero se obtiene la imagen con índice especificado del directorio configurado al comienzo del script. Luego, se convierte esa imagen a escala de grises mediante la función cvtColor de cv2, y se aplica la función *threshold* del mismo módulo de openCV. Esta función permite umbralizar la imagen. Es decir, convertirla a un formato binario formada por aquellos que cumplen un determinado umbral, y los que no. Como parámetros esta función recibe, la imagen de origen en escala de grises a la que se aplicará la umbralización, el valor umbral que se utilizará para clasificar los píxeles en blanco o negro (que se obtiene a partir de la luminosidad media y la función de la Ecuación II), el valor máximo que se asignará a los píxeles que superen el umbral (por lo general se establece en 255), y el tipo de umbralización que se aplicará a la imagen. Este último

parámetro se refiere a la forma en la que se asignan los valores a la umbralización y a qué partes de la misma. Por ejemplo, algunos tipos comunes son:

- cv2.THRESH_BINARY: En este tipo de umbralización, los píxeles de la imagen que superan el umbral se establecen en un valor máximo (definido por maxval), y los píxeles restantes se establecen en 0.
- cv2.THRESH_BINARY_INV: Al contrario que en cv2.THRESH_BINARY, en este tipo de umbralización los píxeles que superan el umbral se establecen en 0, y los píxeles restantes se establecen en un valor máximo.
- cv2.THRESH_TRUNC: En este tipo de umbralización, los píxeles que superan el umbral se mantienen con el valor del umbral, y los píxeles restantes no se modifican.
- cv2.THRESH_TOZERO: Los píxeles que superan el umbral se mantienen sin cambios, y los píxeles restantes se establecen en 0.
- cv2.THRESH_TOZERO_INV: Al contrario que en cv2.THRESH_TOZERO, los píxeles que superan el umbral se establecen en 0, y los píxeles restantes se mantienen sin cambios.

También son comunes la combinación de los anteriores tipos de umbralización con el método de Otsu, cuyo objetivo es encontrar el umbral que maximice la varianza entre las clases de píxeles en una imagen, y que minimice la varianza entre los píxeles de una misma clase. Es ampliamente utilizado en tareas de segmentación de imágenes, como la detección de bordes, la segmentación de objetos y la binarización de imágenes.

Para seleccionar una de las opciones de umbralización comentadas, se representaron las imágenes tras aplicar los distintos umbrales a las mismas. Como se puede observar en la Figura A.3.1 y la A.3.2, la combinación del método Otsu con la binarización inversa es la opción que minimiza el ruido (regiones pequeñas luminosas no asociadas a fuentes y que aumentan el tiempo de procesamiento) y que optimiza la detección de las fuentes luminosas.

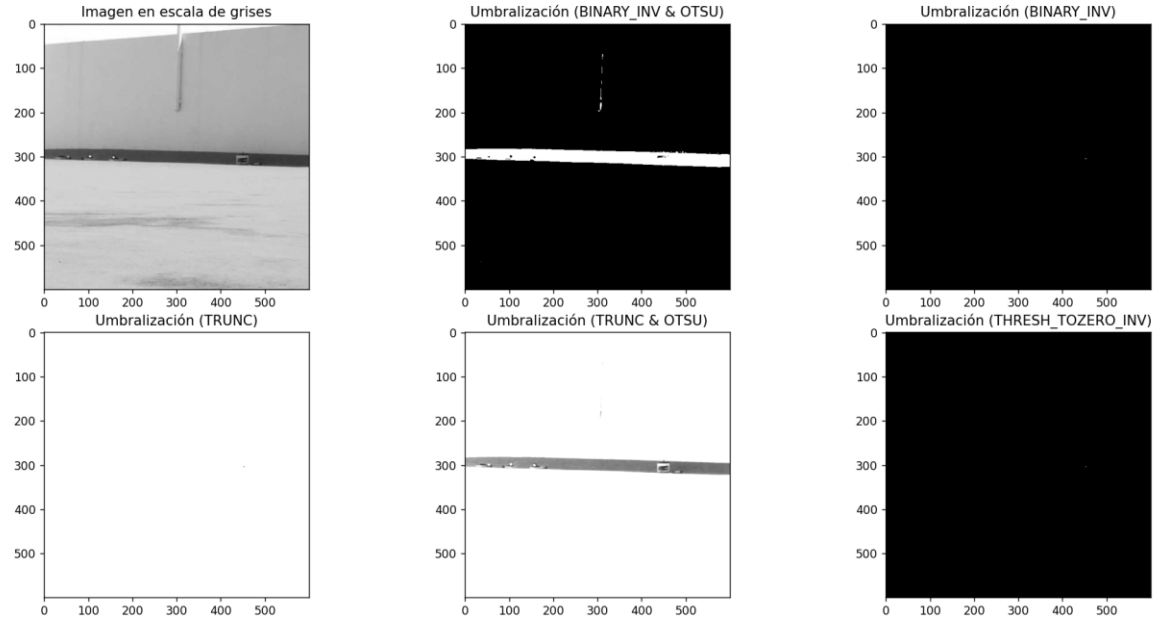

Figura A.3.1: Métodos de umbralización para escenario outdoor con tiempo de exposición más alto

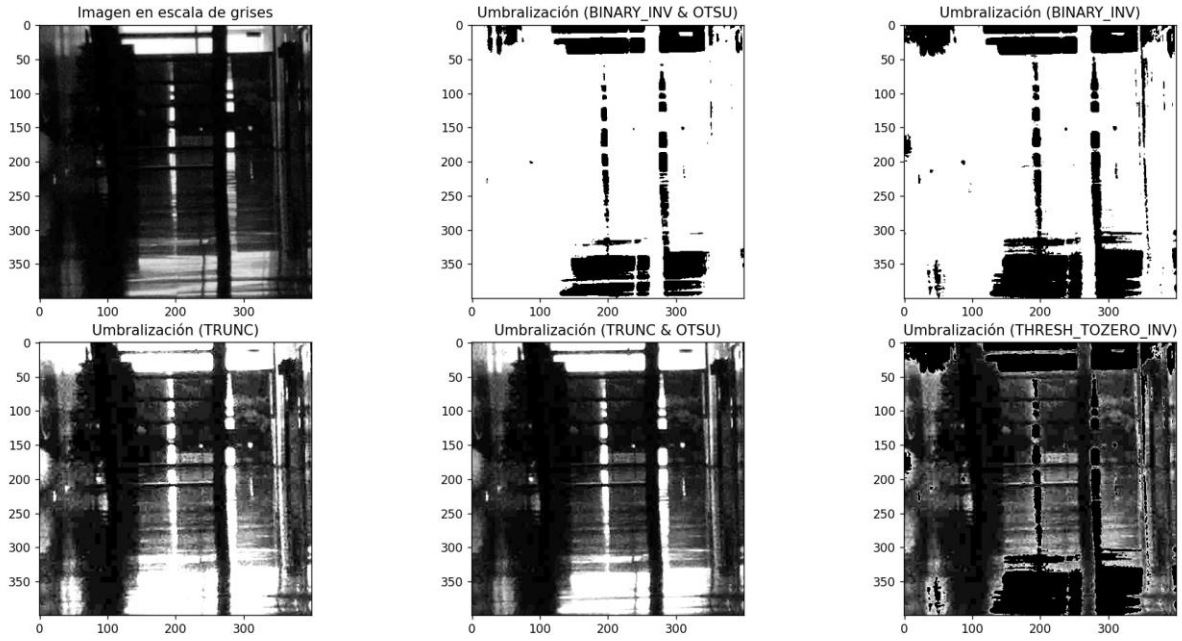

Figura A.3.2: Métodos de umbralización para escenario outdoor con tiempo de exposición más bajo

Como resultado, la función threshold devuelve el valor de umbral utilizado y la imagen resultante de aplicar la umbralización, que estará formada por los valores 255 y 0 únicamente en este caso. Esta imagen binarizada se pasa por parámetro a su vez a la función *label* del módulo *measure*. Esta función se utiliza para etiquetar todas aquellas regiones conectadas en la imagen binaria *thresh*. Es decir, todos aquellos píxeles o conjuntos de píxeles con un valor contiguo igual (0 o 255). A cada región se asigna un valor de etiqueta único. El método devuelve la variable *labels* que almacena la imagen de etiquetas resultante, donde cada píxel está etiquetado con un número de región, y la variable *num*, que contiene el número total de regiones encontradas.

Por último, la función *find_region_size* simplemente busca entre todas las etiquetas de la imagen *labels*, aquella que contenga el píxel seleccionado. Es decir, obtiene la etiqueta en la que se encuentra ese píxel. Luego, finalmente, con la función *count_nonzero* del módulo *numpy* cuenta la cantidad de píxeles que en la imagen etiquetada, *labels*, toman el valor de la etiqueta en la que se encuentra el píxel seleccionado, *label*. La cantidad retornada por la función se corresponde con la cantidad de píxeles que componen la región en la que se encuentra el píxel pulsado.

El diagrama de flujo del Código A.3.1 se puede observar en la Figura A.3.3.

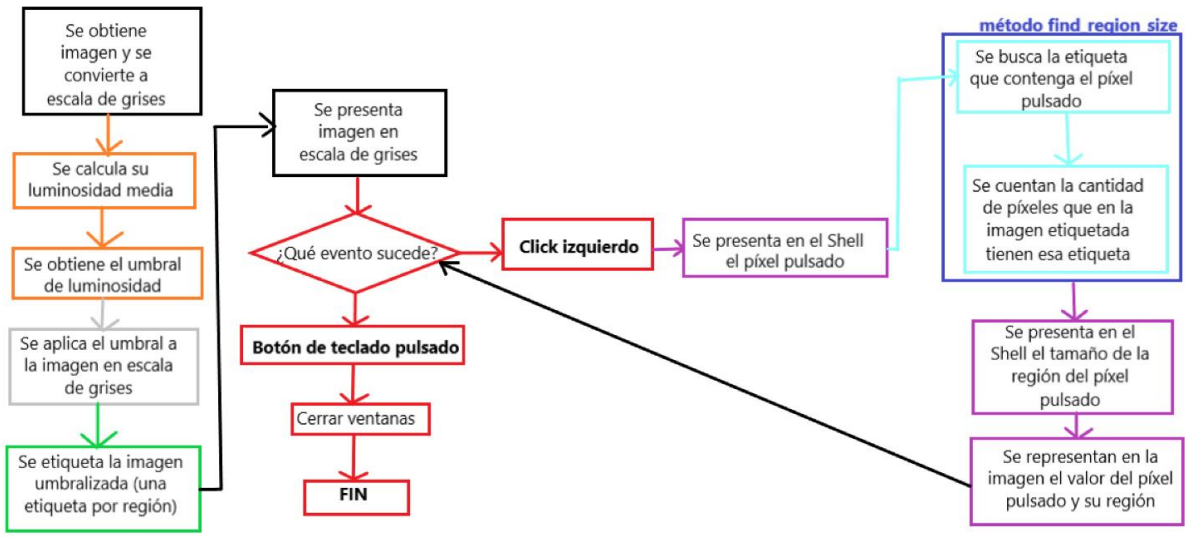

Figura A.3.3: Diagrama de flujo del Código A.3.1

```
import cv2
import numpy as np
from skimage import measure

# Configurar
img = cv2.imread("indoorhigh/frame10.jpg")

# Se convierte la imagen a escala de grises
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
x = (np.mean(image))
if x < 200:
    light_umbral = (5.99e-08)*(x**5)
    - (2.062e-05)*(x**4) + (0.001533)*(x**3)
    + 0.08777*(x**2) - 9.452*x + 221.2
else:
    light_umbral = 80
print('el umbral de luminosidad es: ', light_umbral)

# Se aplica el umbral a la imagen para obtener regiones luminosas
(T, thresh) = cv2.threshold(
    gray, light_umbral, 255, cv2.THRESH_BINARY_INV & cv2.THRESH_OTSU)
```

```
23
24  # Se etiquetan las regiones
25  labels, num = measure.label(thresh, background=0, return_num=True)
26
27  # Metodo que muestra las posiciones y ROIs de los pixeles pulsados
28  def click_event(event, x, y, flags, params):
29      # Se chequea que se hizo click izquierdo
30      if event == cv2.EVENT_LBUTTONDOWN:
31          # Se representan coordenadas en el Shell
32          print(x, ' ', y)
33
34          # Se calcula su ROI
35          region_size = find_region_size(x, y)
36          # Se representa la ROI en el Shell
37          print('Region size:', region_size)
38
39          # Se representan las coordenadas y tamaño en la imagen
40          font = cv2.FONT_HERSHEY_SIMPLEX
41          p = [y, x]
42          cv2.putText(
43              gray, "{}".format(p), (x, y), font, 0.75, (255, 0, 0), 2)
44          cv2.putText(
45              gray, "Size: {}".format(region_size),
46              (x, y+20), font, 0.75, (0, 255, 0), 2)
47          cv2.imshow('image', gray)
48
49  # Funcion que calcula el tamaño de la region clicada
50  def find_region_size(x, y):
51      # Se obtiene la etiqueta de region correspondiente al pixel (x, y)
52      label = labels[y, x]
53      numPixels = np.count_nonzero(labels == label)
54      return numPixels
55
56
57  if __name__=="__main__":
58      # Se muestra la imagen
59      cv2.imshow('image', gray)
60
61      # Se establece funcion por defecto a click de raton
62      cv2.setMouseCallback('image', click_event)
63
64      # Se espera a que se pulse una tecla
65      cv2.waitKey(0)
66
67      # Se cierran todas las ventanas
68      cv2.destroyAllWindows()
```

Código A.3.1: Script *check_ROI*

B. Código del script principal

```
from multiprocessing import Pool, TimeoutError
import multiprocessing as mp
from functools import partial
from socket import timeout
import time
import loader
import simplifier
import correlator
import numpy as np
import cv2
import matplotlib.pyplot as plt
from scipy.special import erfc
from sklearn.mixture import GaussianMixture
import math

#-----------------------------------------
# Variables
# Variables - multiprocessing
frames_queue = mp.Queue()
amount_of_subregion = mp.cpu_count()

# Variables - estructura de trama
tiempo_bit = 4/30
num_bits_trama = 15
fps = 30
num_tramas = 2 # fijo

# Variables - procesado de imagen
index_unified = []
images_simplifier = []
images_correlator = []

# CONFIGURAR
directory = "noche"
init_frame = 3600
#-----------------------------------------
# Función para calcular el valor de light_umbral según la ecuación
def calculate_light_umbral(x):
    if x < 195:
      light_umbral = (5.503e-08) * (x**5) - (2.038e-05) * (x**4) \
               + (0.001936) * (x**3) + (0.01908) * (x**2) \
               - 5.868 * x + 202.6
    else:
      light_umbral = 195
    roi_umbral = (3.649e-10)*(x**6) - (2.628e-07)*(x**5) \
             + (7.255e-05)*(x**4) - (0.00955)*(x**3) \
             + 0.6035*(x**2) - 16.07*x + 168.2

    return light_umbral, roi_umbral

```

```
51  # Método que elimina los pixeles vecinos (asi como su trama
52  # asociada), dejando unicamente el primero que aparezca en
53  # la lista definitiva de pixeles
54  def delete_neighbouring_pixels(array, tramas, pixel_limit):
55      position_to_delete = []
56      # Obtenemos los indices de los pixeles vecinos
57      for i in range(0, len(array)-1):
58          for j in range(i+1, len(array)):
59              x = array[i][0] - array[j][0]
60              y = array[i][1] - array[j][1]
61
62              if np.abs(x+y) < pixel_limit:
63                  position_to_delete.append(j)
64
65      # Eliminamos las posiciones de los pixeles vecinos
66      # Eliminamos posiciones repetidas de la lista
67      position_to_delete = [*set(position_to_delete)]
68      # Ordenamos en orden decreciente la lista
69      position_to_delete.sort(reverse = True)
70      for i in position_to_delete:
71          array.pop((i))
72          tramas.pop((i))
73      return array, tramas
74
75  #----------------------------------------
76  if __name__ == '__main__':
77    number_of_images = int(tiempo_bit*num_bits_trama*num_tramas*fps)
78    print('The number of frames are: '+str(number_of_images))
79    pool = mp.Pool(processes = amount_of_subregion)
80
81    # CAMARA
82    # Inicias proceso de captura
83    capture_process = mp.Process(
84       target=loader.get_frames,
85       args=(directory, init_frame, number_of_images, frames_queue))
86    capture_process.start()
87
88    count = 0
89    second_count = 0
90    # SIMPLIFIER
91    while count < number_of_images:
92      timref = time.time()
93      # Espera por elemento de la camara
94      image = frames_queue.get(timeout=20)
95      if (count == 0):
96        light_umbral, roi_umbral = calculate_light_umbral(
97           np.mean(image))
98        print('Light umbral is: ', light_umbral)
99        print('ROI umbral is: ', roi_umbral)
100
101     # Almacenamos solo la mitad de los frames porque con ellos
102     # es mas que suficiente para detectar un "1" en transmision
```

```
103       if (count < 60):
104         images_simplifier.append(image)
105       images_correlator.append(image)
106       if (len(images_simplifier) % amount_of_subregion == 0) \
107       & (count < 60) \
108       & (len(images_simplifier) > 0):
109         # Se obtienen los pixeles potencialmente tx de la imagen
110         # capturada
111         index_per_image = pool.map(partial
112                                    (simplifier.get_possible_tx,
113                                     light_umbral,
114                                     roi_umbral),
115                                     iterable = images_simplifier)
116         # Se resetea almacen de imagenes para el simplificador
117         images_simplifier = []
118         second_count = second_count + 1
119         # anexamos arrays de pixeles
120         if second_count == 1:
121           # no se pueden concatenar arrays a uno vacio,
122           # asi que en el primer caso se copia
123           index_unified = index_per_image.copy()
124         else:
125           for c in range (0, amount_of_subregion):
126             # concatenamos todos los pixeles considerados potenciales txs
127             index_unified[c] = np.concatenate((
128                index_unified[c], index_per_image[c]))
129       # incrementamos cuenta
130       count = count + 1
131     index_possible_tx = index_unified.copy()
132
133
134     try:
135       # Se transforma el array de subsecciones debido al multiprocessing
136       # en una unica lista de valores [x,y]
137       # Se pasa de [[[a,b] ...],[[c,d] ...]] a [[a,b], [c,d] ...]
138       array = [index_possible_tx[0][1]]
139       # accedemos a cada uno de los resultados del multiprocessing
140       for c in range(0,len(index_possible_tx)):
141         # escogemos todos los elementos de cada resultado
142         # del multiprocessing, menos el primer elemento [0,0]
143         for d in range(1, len(index_possible_tx[c])):
144           array.append(index_possible_tx[c][d])
145
146       # eliminamos los elementos repetidos
147       index_possible_tx = np.unique(array, axis = 0)
148       timout = time.time()
149       print('Tiempo de simplificación:')
150       print(timout-timref)
151       print('Hay '+ str(np.shape(index_possible_tx)[0]) + ' candidatos')
152
153
154       # CORRELATOR
```

```
155       tima = time.time()
156       # Se obtienen los pixeles transmisores y sus tramas
157       # en base a caracteristicas de std, y periodos
158       [pixels, tramas] = correlator.get_tx_pixels(images_correlator,
159                                                   index_possible_tx,
160                                                   light_umbral)
161
162       # ELIMINAMOS PIXELES VECINOS
163       [pixels, tramas] = delete_neighbouring_pixels(pixels, tramas, 12)
164
165       # Presentamos resultados
166       print('Hay ' + str(np.shape(pixels)[0]) + ' transmisores')
167       print(pixels)
168       timb = time.time()
169       print('Tiempo de correlación:')
170       print(timb-tima)
171
172       ########## OBSERVACION DE RESULTADOS ###########
173       # GRABAMOS IMAGEN PARA PODER USARLA CON CHECKPXL
174       cv2.imwrite('imagen.jpg', images_correlator[0])
175
176       # REPRESENTAMOS PIXELES
177       # Recuperamos imagen
178       gray = images_correlator[0]
179       id = 0
180       # Presentamos posicion de los pixeles en la imagen
181       for p in pixels:
182         [y,x] = p
183         id += 1
184         cv2.putText(gray, ".{}".format(id), (x, y - 2),
185                         cv2.FONT_HERSHEY_SIMPLEX, 0.55,
186                         (255, 0, 0), 2, cv2.LINE_AA)
187       cv2.imshow("Possible TXs labeled", gray)
188       cv2.waitKey(0)
189
190       # REPRESENTAMOS TRAMA
191       id = 1
192       for y in tramas:
193         print(y)
194         plt.rcParams["figure.autolayout"] = True
195         x = list(range(0,number_of_images))
196
197         plt.title("Señal óptica {}".format(id))
198         plt.ylabel('Luminosidad en escala de grises')
199         plt.xlabel('Imagen')
200         plt.plot(x, y, color="red")
201         plt.show()
202
203         cv2.waitKey(0)
204         cv2.destroyAllWindows()
205
206         # OBTENEMOS Y REPRESENTAMOS HISTOGRAMA
```

```
207         plt.rcParams["figure.autolayout"] = True
208         (counts, bins, patches) = plt.hist(y)
209         plt.title("Histograma {}".format(id))
210         plt.xlabel("Luminosidad")
211         plt.ylabel("Densidad")
212         plt.show()
213
214         # CALCULAMOS SNR
215         # Se convierte el array que contiene la trama (y)
216         # en una matriz bidimensional
217         y = np.array(y).reshape(-1, 1)
218         # Se crea un objeto GaussianMixture con 2 componentes
219         gmm = GaussianMixture(n_components=2)
220         # Se ajusta el modelo de mezcla gaussiana a los datos de y
221         gmm.fit(y)
222
223
224         # Se obtienen los parámetros estimados del modelo GMM
225         # Ratio de la primera componente gaussiana
226         alpha = gmm.weights_[0]
227         # Media de la primera componente gaussiana
228         mu0 = gmm.means_[0][0]
229         # Desviación estándar de la primera componente gaussiana
230         sigma0 = np.sqrt(gmm.covariances_[0][0][0])
231         # Media de la segunda componente gaussiana
232         mu1 = gmm.means_[1][0]
233         # Desviación estándar de la segunda componente gaussiana
234         sigma1 = np.sqrt(gmm.covariances_[1][0][0])
235
236         # Se calcula la SNR utilizando la formula proporcionada
237         SNR = (1 / 2) * ((mu1 - mu0)**2) / \
238         (alpha * (sigma0**2) + (1 - alpha) * (sigma1**2))
239
240         SNR_dB = 10 * math.log10(SNR)
241
242         # Se calcula la tasa de error de bits (BER) utilizando
243         # la SNR
244         BER = (1 / 2) * erfc(np.sqrt(SNR / 2))
245
246         # Se imprimen los resultados
247         print("SNR (dB):", SNR_dB)
248         print("BER:", BER)
249
250
251         id += 1
252
253         cv2.waitKey(0)
254         cv2.destroyAllWindows()
255     except:
256       print('No hay transmisores')
```

Código B.1: Script *main*

C. Resultados

En este apartado de la sección de anexos se adjuntan las hojas Excel desarrolladas durante la fase de experimentación, que resumen los distintos resultados obtenidos. En total hay 7 hojas: una por cada escenario y tiempo de exposición (Figura C.2 - Figura C.13), y una que resume los resultados globales (Figura C.1).

En la hoja que resume los resultados globales se puede observar el tiempo de procesamiento y la SNR promedio, los datos de ROI máximos y mínimos, las tasas de detección de positivos verdaderos, de falsos positivos y de omisión de positivos de cada escenario y tiempo de exposición, y en cómputo global.

Por su parte, en cada hoja individual de cada escenario y tiempo de exposición se pueden observar: los índices de inicio de las distintas muestras estudiadas; el tiempo que tardó en procesar el bloque de simplificación y correlación con esas muestras; la tasa de detección de positivos y falsos positivos; la ROI promedio y el color de cada transmisor, y la SNR y BER presentados por cada uno de los mismos por cada muestra de vídeo.

Tasa de detección promedio		Tasa de omisión promedio	SNR promedio (dB)	ROI max	ROI min
Indoor high	0,95	0,05	20,3	38	9
Indoor medium	0,79	0,21	20,5	31	1
Indoor low	0,93	0,1	21,4	34	3
Outdoor high	0,89	0,09	20,8	10	9
Outdoor low	0,55	0,48	27,8	6	5

Tasa de detección promedio		SNR promedio (dB)	ROI max	ROI min	Tiempo total	Tasa de falsos positivos	Tasa de omisión promedio
Indoor	0,89	22,6	38	1	0,60	0,83	0,11
Outdoor noche	1,00	21,3	97	3	0,05	0	0
Outdoor día	0,72	20,8	6	5	0,52	0	0,28
General	0,87	21,6	97	1	0,39	0,28	0,13

Tasa de omisión promedio	
Indoor	0,11
Outdoor noche	0,00
Outdoor día	0,28
General	0,13

Casos	
Todos detectados	0,50
50% detectados	0,85
75% detectados	0,72

Figura C.1: Resultados globales

Tiempo de exposición (ms)	9
Sensibilidad ISO	100

Índice vídeo	Cantidad de tx reales	Cantidad de tx detectados que son tx	Cantidad de tx detectados que no son tx	Cantidad de tx no detectados	Tiempo simplicación
0	4	4	4	0	0,11
1200	4	3	3	1	0,11
2400	4	4	3	0	0,20
3600	4	4	2	0	0,20
4800	4	3	3	1	0,09
6000	4	4	3	0	0,04
7200	4	4	11	0	0,02
8400	4	4	3	0	0,03
9600	4	3	3	1	0,03
10800	4	4	3	0	0,02
12000	4	4	3	0	0,03
13200	4	4	3	0	0,04
14400	4	4	3	0	0,03
15600	4	3	3	1	0,03
Total	56	52	50	4	

Tasa de detección	0,93	Se detectaron todos	10
Tasa de omisión de positivo	0,07	Se detectaron 3	4
Tasa de falsos positivos	0,89	Se detectaron 2	0

Figura C.2: Resultados del escenario indoor con tiempo de exposición bajo (I)

		Transmisores											
		1 [346,876]			2[351,858]			3 [352,786]			4[353,772]		
Tiempo correlación	Tiempo total	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI
0,7	0,81	18,4	5,36E-17		22,4	3,56E-40		24,3	9,70E-61		24,7	2,52E-66	
0,8	0,91	19,5	1,10E-21					26,3	2,23E-95		24,5	5,86E-63	
0,9	1,10	22,3	4,40E-39		26,9	5,45E-110		20,7	1,85E-27		23	3,38E-45	
0,8	1,00	23,7	1,91E-53		25,9	4,01E-86		22,3	8,73E-39		23,4	5,17E-50	
0,9	0,99	18,5	2,39E-17					21,5	1,33E-32		19,2	3,40E-20	
0,5	0,50	21,6	6,02E-34		22,7	2,26E-42		28,9	6,33E-173		19,5	1,34E-21	
0,8	0,90	21,9	5,36E-36	2	30,1	1,14E-223	21	29,6	7,60E-200	11	17	8,73E-13	6
0,9	0,80	23,3	2,02E-48		39	1,64E-248		24,1	1,88E-57		19,4	3,24E-21	
0,4	0,44	18,6	7,88E-18					19,1	7,45E-20		22	2,23E-36	
0,7	0,75	19,7	1,31E-22		20,8	5,62E-28		35,3	1,00E-248		18,8	1,50E-18	
0,6	0,67	20,4	5,10E-26		28,2	3,63E-145		31,8	1,00E-298		19,4	3,74E-21	
0,6	0,64	19,3	1,90E-20					21,6	8,86E-34		19,2	2,80E-20	
1,2	1,25	18,5	1,40E-17					30,2	4,15E-231		22,4	7,76E-40	
0,7	0,75	18,6	6,10E-18					22,6	2,00E-41		20,5	2,73E-26	
MAX	1,25	23,7	5,36E-17		39	5,62E-28		35,3	7,45E-20		24,7	8,73E-13	
MIN	0,44	18,5	1,91E-53	2	20,8	1,64E-248	21	19,1	1,00E-298	11	17	5,86E-63	6
MEDIA	0,81	20,05	6,55E-23		26,9	5,45E-110		23,35	1,00E-41		19,45	2,29E-21	
DESVIACIÓN TÍPICA	0,22	1,90	0,00		5,79	0,00		4,84	0,00		2,38	0,00	

Figura C.3: Resultados del escenario indoor con tiempo de exposición bajo (II)

Tiempo de exposición (ms)	14
Sensibilidad ISO	100

Índice vídeo	Cantidad de tx reales	Cantidad de tx detectados que son tx	Cantidad de tx detectados que no son tx	Cantidad de tx no detectados	Tiempo simplicación
0	4	3	2	1	0,03
1200	4	4	4	0	0,03
2400	4	3	1	1	0,05
3600	4	4	1	0	0,02
4800	4	4	3	0	0,03
6000	4	4	3	0	0,06
7200	4	3	1	1	0,06
8400	4	3	4	1	0,04
9600 [PERSONA OBSTACULIZA]	4	3	3	1	0,05
10800 [PERSON OBSTACULIZA]	4	2	1	2	0,07
12000 [PERSONA OBSTACULIZA]	4	2	14	2	0,09
13200	4	3	1	1	0,11
14400	4	3	3	1	0,06
15600	4	3	1	1	0,07
Total	56	44	42	12	

Tasa de detección	0,79	Se detectaron todos	4
Tasa de omisión de positivo	0,21	Se detectaron 3	8
Tasa de falsos positivos	0,75	Se detectaron 2	2
		Total	14

Figura C.4: Resultados del escenario indoor con tiempo de exposición medio (I)

		Transmisores											
		1 [346,876]			2[351,858]			3 [352,786]			4[353,772]		
Tiempo correlación	Tiempo total	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI
0,54	0,57	24,2	2,81E-59					18,5	1,92E-17		15,7	5,63E-10	
0,9	0,93	18,7	3,80E-18		17,9	2,14E-15		16,2	6,51E-11		17,7	1,05E-14	
0,6	0,65	21,9	1,00E-35					11,4	1,05E-06		26,1	2,47E-90	
0,37	0,39	18,2	3,10E-16		19,3	1,52E-20		22,2	1,60E-38		16,2	5,20E-11	
0,4	0,43	29,4	3,80E-193		29,9	4,60E-214		23,6	1,62E-51		10,6	3,50E-04	
0,25	0,31	30,1	4,97E-224		30,1	1,41E-223		22,6	1,50E-41		21,5	6,37E-33	
0,3	0,36	18,2	2,01E-16	12			53	13	4,16E-06	21	21,6	4,88E-33	23
0,4	0,44	21,5	4,23E-33					20,6	7,91E-27		18,9	4,29E-19	
0,6	0,65	29,2	2,43E-182					19,9	3,55E-23		28,2	5,82E-145	
0,7	0,77	20,8	5,64E-28					16,5	1,33E-11				
0,9	0,99							16,9	1,30E-12		21,6	2,03E-33	
1	1,11	16,8	1,97E-12					13,4	1,59E-06		29,8	1,27E-208	
0,8	0,86	19,9	3,87E-23					19,3	2,40E-20		29,3	5,20E-188	
0,7	0,77	21	1,13E-29					19,3	8,87E-21		16,3	2,70E-11	
MAX	1,11	30,1	1,97E-12		30,1	2,14E-15		23,6	4,16E-06		29,8	3,50E-04	
MIN	0,31	16,8	4,97E-224	12	17,9	1,41E-223	53	11,4	1,62E-51	21	10,6	1,27E-208	23
MEDIA	0,65	21	1,13E-29		29,9	4,60E-214		19,3	1,64E-20		21,6	4,88E-33	
DESVIACIÓN TÍPICA	0,25	4,56	0,00		6,61	0,00		3,73	0,00		5,93	0,00	

Figura C.5: Resultados del escenario indoor con tiempo de exposición medio (II)

Tiempo de exposición (ms)	19
Sensibilidad ISO	100

Índice video	Cantidad de tx reales	Cantidad de tx detectados que son tx	Cantidad de tx detectados que no son tx	Cantidad de tx no detectados	Tiempo simplicación
0	4	4	3	0	0,02
1200	4	4	4	0	0,02
2400	4	4	2	0	0,02
3600 [PERSONA OBSTACULIZA]	4	3	11	1	0,15
4800	4	4	2	0	0,01
6000	4	4	2	0	0,01
7200	4	4	3	0	0,05
8400	4	4	4	0	0,02
9600	4	4	4	0	0,02
10800	4	3	3	1	0,02
12000	4	3	3	1	0,05
13200	4	4	3	0	0,03
14400	4	4	2	0	0,23
15600	4	4	2	0	0,2
Total	56	53	48	3	

Tasa de detección	0,95	Se detectaron todos	11
Tasa de omisión de positivo	0,05	Se detectaron 3	3
Tasa de falsos positivos	0,86	Se detectaron 2	0
		Total	14

Figura C.6: Resultados del escenario indoor con tiempo de exposición alto (I)

		Transmisores											
		1 [346,876]			2[351,858]			3 [352,786]			4[353,772]		
Tiempo correlación (s)	Tiempo total (s)	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI
0,4	0,42	22,2	5,30E-38		22,6	2,55E-42		25,7	1,40E-83		23,4	2,78E-56	
0,31	0,33	14,5	5,05E-08		18	1,30E-15		14,4	7,75E-08		21,9	1,00E-35	
0,4	0,42	12,8	6,43E-06		22,6	1,93E-41		19,4	6,70E-21		21,8	1,50E-34	
0,3	0,45	18,9	7,40E-19		21,9	3,62E-36		16,3	2,90E-11				
0,2	0,21	18,6	1,11E-17		29	7,67E-177		19,4	3,11E-21		21	8,37E-30	
0,2	0,21	22,9	3,91E-44		23,3	1,24E-48		21,7	1,54E-34		17,9	1,99E-15	
0,3	0,35	24,8	2,58E-68	2	26,6	7,52E-102	34	14,3	9,80E-08	11	11,4	1,10E-05	9
0,2	0,22	20,6	3,43E-27		33,8	1,00E-300		20	5,13E-24		20,9	6,44E-29	
0,3	0,32	17,7	8,58E-15		34,9	1,00E-300		14,1	2,17E-07		24,6	1,95E-65	
0,3	0,32	21,4	4.17E-32					24,2	1,55E-59		32,4	1,00E-298	
0,2	0,25	23,5	1,00E-50					12,1	2,56E-05		27,2	2,89E-116	
0,5	0,53	20,8	1,60E-26		27,3	1,36E-118		19,9	2,05E-23		20,6	3,60E-27	
0,18	0,41	22,9	9,78E-45		33	1,00E-300		19,9	2,79E-23		18,6	1,20E-17	
0,6	0,8	13,3	2,14E-06		33,3	1,00E-300		20,3	2,15E-25		17,8	4,03E-15	
MAX	0,8	24,8	6,43E-06		34,9	1,30E-15		25,7	2,56E-05		32,4	1,10E-05	
MIN	0,21	12,8	2,58E-68	2	18	1,00E-300	34	12,1	1,55E-59	11	11,4	1,00E-298	9
MEDIA	0,34	20,7	1,60E-26		28,15	6,80E-119		19,65	1,57E-21		20,9	6,44E-29	
DESVIACIÓN TÍPICA	0,16	3,86	0,00		5,62	0,00		3,97	0,00		5,03	0,00	

Figura C.7: Resultados del escenario indoor con tiempo de exposición alto (II)

Tiempo de exposición (ms)	30
Sensibilidad ISO	500

Índice vídeo	Cantidad de tx reales	Cantidad de tx detectados que son tx	Cantidad de tx detectados que no son tx	Cantidad de tx no detectados	Tiempo simplicación
0	4	4	0	0	0,02
1200	4	4	0	0	0,03
2400	4	4	0	0	0,02
3600	4	4	0	0	0,02
4800	4	4	0	0	0,02
6000	4	4	0	0	0,02
7200	4	4	0	0	0,02
8400	4	4	0	0	0,03
9600	4	4	0	0	0,01
10800	4	4	0	0	0,03
12000	4	4	0	0	0,03
13200	4	4	0	0	0,03
14400	4	4	0	0	0,03
15600	4	4	0	0	0,01
Total	56	56	0	0	

Tasa de detección	1,00
Tasa de omisión de positivo	0,00

Figura C.8: Resultado del escenario outdoor nocturno (I)

		Transmisores											
		1[382,878]			2[418,867]			3 [434, 319]			4 [538, 392]		
Tiempo correlación (s)	Tiempo total (s)	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI
0,02	0,04	17,4	5,30E-14		18.2	2,00E-16		21	2,52E-29		27,7	1,72E-129	
0,02	0,05	21,6	2,47E-33		18,7	5,70E-18		38,2	1,00E-298		23,9	1,20E-54	
0,02	0,04	19,5	2,02E-21		9,6	1,30E-03		23,5	2,14E-50		19,9	3,50E-23	
0,03	0,05	21,6	2,50E-33		21,4	2,15E-32		22,8	1,61E-43		22,3	6,90E-39	
0,02	0,03	23,7	9,50E-53		27	5,46E-112		38,4	1,00E-298		24,3	3,10E-60	
0,03	0,05	21,4	7,82E-32		16,6	8,20E-12		24,8	5,90E-68		25,3	1,80E-76	
0,03	0,05	21,6	8,60E-34	29	13,6	8,80E-07	2	15,3	3,45E-09	127	15,2	4,60E-09	64
0,01	0,03	19,7	2,70E-22		20	1,13E-23		23,1	2,75E-46		23	2,93E-46	
0,05	0,06	25,6	4,73E-82		18	6,90E-16		34,7	1,00E-298		20,8	1,60E-28	
0,02	0,05	16,2	4,50E-11		18,9	5,20E-19		36,9	1,00E-298		18,5	2,21E-17	
0,03	0,06	15,6	8,10E-10		19,1	6,60E-20		24,3	7,90E-61		23,4	1,95E-49	
0,01	0,04	26	2,30E-89		16,4	2,02E-11		24	1,35E-56		20	7,50E-24	
0,01	0,04	18,7	4,47E-18		19,3	1,23E-20		20,7	8,32E-28		21,2	1,64E-30	
0,05	0,06	16,4	1,76E-11		24,4	1,63E-62		37,6	1,00E-298		21,7	1,30E-34	
MAX	0,06	26	8,10E-10		27	1,30E-03		38,4	3,45E-09		27,7	4,60E-09	
MIN	0,03	15,6	2,30E-89	29	9,6	5,46E-112	2	15,3	1,00E-298	127	15,2	1,80E-76	64
MEDIA	0,05	20,55	1,35E-22		19	2,93E-19		24,15	6,75E-57		21,45	8,20E-31	
DESVIACIÓN TÍPICA	0,01	3,33	0,00		6,53	0,00		7,84	0,00		3,09	0,00	

Figura C.9: Resultados del escenario outdoor nocturno (II)

Tiempo de exposición (ms)	0,8
Sensibilidad ISO	40

Índice vídeo	Cantidad de tx reales	Cantidad de tx detectados que son tx	Cantidad de tx detectados que no son tx	Cantidad de tx no detectados	Tiempo simplicación (s)
0	4	3	0	1	0,02
600	4	3	0	1	0,02
1200	4	3	0	1	0,02
1800	4	4	0	0	0,01
2400	4	4	0	0	0,01
3000	4	4	0	0	0,02
3600	4	4	0	0	0,02
4200	4	4	0	0	0,02
4800	4	4	0	0	0,02
5400	4	3	0	0	0,02
6000	4	3	0	1	0,02
Total	44	39	0	4	

Tasa de detección	0,89	Se detectaron todos	6
Tasa de omisión de positivo	0,09	Se detectaron 3	5
		Se detectaron 2	0
		Total	11

Figura C.10: Resultados del escenario outdoor diurno con tiempo de exposición alto (I)

		Transmisores											
		1 [295, 739]			2 [298, 805]			3 [301, 856]			4[301, 1152]		
Tiempo correlación (s)	Tiempo total (s)	SNR (dB)	BER	ROI	SNR (dB)	BER	PSF	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI
0,3	0,6	17,7	6,12E-15		20,6	3,36E-27		24,8	5,30E-67				
0,4	0,5	16,6	6,52E-12		30,4	3,54E-240		36,2	1,00E-298				
0,4	0,42	16,1	1,09E-10		31,2	7,12E-291		33,6	1,00E-298				
0,6	0,61	18,2	1,94E-16		19,7	2,02E-22		24,6	1,17E-64		18,7	3,73E-18	
0,5	0,51	21,3	2,87E-31		20,6	3,25E-27		33,6	1,00E-298		16,5	1,37E-11	
0,4	0,42	21,2	5,93E-31	3	16,3	2,95E-11	26	21,6	1,32E-33	21	20,2	4,42E-25	6
0,6	0,52	22,5	9,55E-41		16,9	1,60E-12		19,6	4,11E-22		17,6	1,21E-14	
0,4	0,42	20,6	4,59E-27		33,6	1,00E-300		18,9	5,40E-19		21,4	3,26E-32	
0,55	0,57	18,2	1,87E-16		33	1,00E-300		21	9,30E-39		18,4	4,99E-17	
0,6	0,62				26,7	1,33E-103		20,9	5,34E-29		17,9	2,13E-15	
0,7	0,72				19,7	1,68E-32		21	9,00E-30		16,5	1,26E-11	
MAX	0,72	22,5	1,09E-10		33,6	2,95E-11		36,2	5,40E-19	21	21,4	1,37E-11	
MIN	0,42	16,1	9,55E-41	3	16,3	1,00E-300	26	18,9	1,00E-298	0	16,5	3,26E-32	6
MEDIA	0,52	21,2	5,93E-31		20,6	1,68E-32		21	9,00E-30	#¡NUM!	17,9	2,13E-15	
DESVIACIÓN TÍPICA	0,10	2,29	0,00		6,64	0,00		6,33	0,00	#¡DIV/0!	1,71	0,00	

Figura C.11: Resultados del escenario outdoor diurno con tiempo de exposición alto (II)

Tiempo de exposición (ms)	0,45
Sensibilidad ISO	40

Índice vídeo	Cantidad de tx reales	Cantidad de tx detectados que son tx	Cantidad de tx detectados que no son tx	Cantidad de tx no detectados	Tiempo simplicación
0	4	3	0	2	0,06
600	4	3	0	1	0,03
1200	4	2	0	2	0,02
1800	4	2	0	2	0,2
2400	4	2	0	2	0,01
3000	4	2	0	2	0,02
3600	4	2	0	2	0,02
4200	4	2	0	2	0,02
4800	4	2	0	2	0,02
5400	4	2	0	2	0,02
6000	4	2	0	2	0,02
Total	44	24	0	21	

Tasa de detección	0,55
Tasa de omisión de positivo	0,48

Se detectaron todos	0
Se detectaron 3	2
Se detectaron 2	9
Total	11

Figura C.12: Resultados del escenario outdoor diurno con tiempo de exposición bajo (I)

		Transmisores											
		1			2 [803, 293]			3 [295, 857]			4[296, 1152]		
Tiempo correlación	Tiempo total	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI	SNR (dB)	BER	ROI
2	2,06				37,8	E-298		20	6,64E-24		20,4	6,00E-26	
0,6	0,63				35,2	E-298		18	7,97E-16		20,3	1,31E-25	
0,5	0,52				24	2,00E-56		20,1	3,34E-24				
0,5	0,7				20,9	9,63E-29		22,5	4,04E-41				
0,4	0,22				31,3	1,15E-295		25,8	2,29E-85				
0,4	0,42			2	27,7	2,94E-131	22	22,5	3,49E-41	20			4
0,5	0,52				30,5	1,88E-246		25,3	1,10E-75				
0,5	0,52				30,4	3,55E-238		29,5	3,69E-197				
0,7	0,72				26,9	2,36E-107		27,7	2,89E-130				
0,7	0,72				30,3	4,30E-295		24,9	3,25E-69				
0,6	0,62				17,3	9,89E-14		24,7	9,44E-67				
MAX	2,06	0	0		37,8	9,89E-14		29,5	7,97E-16		20,4	1,31E-25	
MIN	0,22	0	0		17,3	1,15E-295		20,1	3,69E-197		0	0	
MEDIA	0,52	#¡NUM!	#¡NUM!	2	30,3	3,55E-238	22	25,3	1,10E-75	20	#¡NUM!	#¡NUM!	4
DESVIACIÓN TÍPICA	0,50	0,58	#¡DIV/0!		9,98	0,00		9,46	0,00		11,75	0,00	

Figura C.13: Resultados del escenario outdoor diurno con tiempo de exposición bajo (II)