UNIVERSIDAD OBERTA DE CATALUNYA

Trabajo Fin de Máster Universitario en Software Libre

POSICIONAMIENTO EN INTERIORES DE UN DRON POR MÉTODO MULTICÁMARA

Alumno/a: José Antonio Calvillo Ardila
Dirigido por: Dr. Pedro Luis Galindo Riaño

CURSO 2016-17 (Septiembre/Febrero)

Índice de contenido

Índice de ilustraciones

Índice de tablas

Resumen

El presente trabajo tiene por objetivo ofrecer una solución tecnológica que permite obtener la posición de un dron en el interior de una habitación, a partir de odometría visual, basada únicamente en dos cámaras y sin emplear ningún tipo de sensor adicional.

La implementación se ha realizado utilizando Software Libre y dispositivos de bajo coste. El lenguaje de programación que se ha usado ha sido Python, junto a diferentes módulos especializados para el manejo de las cámaras, para el tratamiento de imágenes y gestión de los interfaces. En cuanto al hardware, se ha seleccionado la plataforma *Raspberry Pi* debido a la gran comunidad que tiene, el uso de protocolos de comunicación estándar y su precio reducido.

Además, para la realización del proceso de reconstrucción 3D del dron, se le añadió un led encendido en su parte superior y se implementó un sistema de reconocimiento basado en la identificación del punto de máxima luminosidad. La posición 3D se obtuvo a partir de dos imágenes capturadas con cámaras diferentes en el mismo instante.

Por último, hay que mencionar que fue necesario crear un dispositivo óptico de medición del tiempo con precisión suficiente (1 milésima de segundo) para poder medir el grado de sincronización en la captura simultánea de imágenes desde dos cámaras diferentes.

Abstract

This paper aims to provide a technological solution for indoor positioning of a dronbased on the use of multiple cameras without using any type of additional sensors.

The implementation has been made using free software and low cost devices. Software was developed using the Python programming language along with different specialized Python modules for camera management, network interfaces configuration, image processing and computer vision. Regarding the hardware, we choose the Raspberry Pi platform because of the large community behind this platform, the use of standard communication protocols and its reduced price.

3D position is recovered in real time by identifying the point of maximum luminosity in images taken simultaneously by different cameras, and using its 2D positions along with the intrinsics and extrinsics parameters of each camera to reconstruct the 3D position. In order to precisely identify the dron in captured images, a high brightness led was added on its upper part.

Finally, it should be mentioned that it was necessary to create an optical device of time measurement having a precision of 1 thousandth of a second to be able to measure the degree of synchronization in the simultaneous capture of images from different cameras.

1. Introducción.

Este trabajo representa la memoria final del Trabajo de Fin de Máster en Software Libre (en adelante TFM) titulado *Posicionamiento en interiores de un dron por método multicámara*. *(Universitat Oberta de Cataluña).*

En la actualidad, cada vez es más amplio el catálogo de funciones y aplicaciones de los vehículos aéreos no tripulados o drones. La mayoría de estos usos se realizan en espacios abiertos, donde pueden hacer uso de un sistema de posicionamiento global como el GPS.

Este proyecto aborda el posicionamiento de un dron pero en un espacio cerrado donde resulta imposible el uso de sistemas de posicionamiento *GPS*. Para ello, se utiliza una tecnología visual basada en varias cámaras, conectadas cada una a un ordenador de placa única (en adelante *SBC*).

El objetivo principal de este trabajo ha consistido en implementar una solución tecnológica que permite obtener la posición de un dron en el interior de una habitación, a partir de odometría visual. Otros objetivos han sido los siguientes:

- Conseguir un sistema de posicionamiento basado únicamente en cámaras y sin emplear ningún tipo de sensor adicional.
- Lograr un sistema lo suficientemente preciso y rápido para que el dron se mueva de forma autónoma.
- Aportar una solución basada completamente en Software Libre.

Son muchas las aplicaciones que pueden implementarse para fines específicos en el ámbito de los drones en espacios cerrados. A continuación, se exponen algunas aplicaciones interesantes:

- En cualquier tipo de emergencia producida dentro de un edificio, en la que se necesita evaluar y localizar rápidamente donde se ha producido un incidente y/o resulta complicado acceder a los equipos de rescate. Por ejemplo: localización de un incendio, búsqueda de heridos, ...
- En labores de vigilancia ya que facilitaría la supervisión constante de edificios que alojan puntos sensibles.
- Para construir un modelo 3D de un espacio interior, a partir de fotografías.

Las principales ventajas del uso de drones en interiores serían:

- Agilizar la localización donde se ha producido el incidente facilitando a los equipos de rescate su acceso.
- Evitar exponer a cualquier persona a un incidente del cual se desconoce su magnitud y posición exacta.
- Reducir el número de personas implicadas en la realización de determinadas labores.
- Disminuir el tiempo de trabajo.

Para llevar a cabo este trabajo, se ha utilizado un ordenador central conectado, a través de conexiones series, a varios ordenadores de placa única (*SBC*), los cuales disponen de una cámara conectada, capaz de realizar fotos y videos.

Cada cámara, se ha colocado y orientado de forma precisa, realizando una serie fotos de forma continua a la sala donde se encuentra el vehículo en movimiento, se ha calculado su posición en el plano fotográfico y se la ha proporcionado al ordenador central. Éste, a partir de las distintas posiciones que le proporcionan cada *SBC*, ha reconstruido la posición en el espacio 3D. Una vez obtenida la posición del dron en el espacio, ya se estaría en disposición de poder desplazarlo por un trazado guiado.

Tras el desarrollo del proyecto, se ofrece un sistema que permite posicionar y desplazar un dron dentro de un espacio cerrado de forma autónoma con un elevado nivel de precisión, aunque este trabajo se ha centrado únicamente en la obtención precisa y lo más rápida posible de la posición del dron dentro de una habitación.

Este documento se ha dividido en diferentes capítulos, en el primer capítulo se realiza una introducción a TFM. Posteriormente en el segundo capítulo se ha realizado el estado del arte

A continuación, en el tercer capítulo se indica el análisis, diseño e implementación de la solución implementada.

Más adelante en el cuarto capítulo, se describe el dispositivo óptico de medición del tiempo creado.

Para continuar, en el capítulo quinto se detalla la información relacionada con la interconexión y sincronizado de las cámaras.

A continuación, en el sexto capítulo se exponen aspectos teóricos y experimentales relacionados con: Visión artificial, Calibración de cámaras y Reconstrucción 3D.

Posteriormente, en el séptimo capítulo se recogen las conclusiones de este trabajo.

2. Revisión del estado del arte.

Este proyecto pretende abordar el posicionamiento de un Vehículo Aéreo (UAV) o dron pero en un espacio cerrado donde resulta imposible el uso de sistemas de posicionamiento *GPS*. Para ello, se utilizará una tecnología visual, basada en varias cámaras conectadas cada una a un ordenador de placa única (en adelante *SBC*).

Mientras que el *GPS* domina el posicionamiento y navegación en espacios abiertos, en la actualidad no existe ninguna tecnología estandarizada para espacios interiores, incluso tampoco existe un estándar que establezca el sistema de coordenadas a usar en áreas interiores. En la mayoría de las aplicaciones de posicionamiento en interiores, se emplean coordenadas locales o coordenadas relativas para posicionar un dispositivo o un ser humano.

Para comprender el alcance de este tipo de problema, resulta interesante realizar un acercamiento al panorama actual y analizar las técnicas empleadas para resolver este tipo de cuestiones, como es la creación de un sistema de posicionamiento para espacios interiores.

Aunque para la realización de este trabajo se empleará una tecnología visual basada en cámaras fotográficas, es necesario comenzar revisando y describiendo las técnicas más usadas hoy en día [1], como son:

- *Tecnologías basadas en ondas de radio*: WIFI, Bluetooth, RFID y Telefonía móvil.
- *Tecnologías no basadas en ondas de radio*: acústicas, navegación por estima e híbridas que surgen como combinación de las anteriores.

Cada tecnología utiliza un enfoque diferente para localizar un usuario o dispositivo.

El posicionamiento basado en tecnología WIFI consta de varios Puntos de Acceso (APs) instalados en los lugares adecuados y conocidos dentro de un espacio cerrado.

En términos generales, se podría decir que las principales ventajas de este tipo de infraestructuras inalámbricas son su flexibilidad, ya que permiten realizar tareas diferentes de su propósito original y su fácil y rápido despliegue, ya que muchos edificios cuentan con un sistema WIFI. Sin embargo, también habría que decir que proporcionan poca precisión (más de un metro de error dependiendo del número de AP) y tienen un elevado coste. El coste podría reducirse significativamente, si se

aprovechara una infraestructura (WIFI) existente.

El posicionamiento basado en tecnología de telefonía móvil (GSM, 3G, 4G) lleva incluida la funcionalidad de posicionamiento como parte de su normativa (E-911). En concreto, se establece una precisión de al menos 50 metros para el 67% de las llamadas de emergencia.

En cuanto al posicionamiento usando tecnología Bluetooth, habría que indicar que se trata de una tecnología de red de corto alcance (10 metros aproximadamente). Aun así, se consigue precisión de 2 metros .

El posicionamiento basado en RFID ha ganado importancia en los últimos años pero presenta una desventaja y es que esta tecnología no se encuentra integrada en todos los teléfonos móviles de hoy en día.

En recintos interiores, resulta bastante complejo la creación de un modelo para determinar la propagación de las ondas de radio, debido principalmente a dos motivos: multicamino de la señal y la dificultad de tener visión directa. El uso de tecnologías inalámbricas en espacios interiores, también se ve afectado por otro tipo de factores como son: tráfico de red, factores ambientales, estructuras usadas en el edificio, otros equipamientos, etc. Todo ello tiene un impacto negativo en la precisión y exactitud a la hora de calcular la posición de una persona u objeto.

En cuanto a otras tecnologías que no usan ondas de radio, por ejemplo, el posicionamiento acústico, habría que decir que presentan dos desventajas: la primera es que los ultrasonidos no pueden penetrar las paredes y además necesitan visión directa. Como ventaja habría que destacar que pueden alcanzar una precisión de 2 cm el 99% del tiempo.

En el caso de la tecnología de navegación por estima, se trata de un enfoque de posicionamiento relativo que comienza desde una ubicación conocida e intenta tener en cuenta el movimiento del objeto. La mayoría de los teléfonos móviles cuentan con un acelerómetro que permite contar los pasos de una persona desde una localización previa.

Las tecnologías híbridas combinan varias de las tecnologías anteriores alcanzando mayor exactitud, precisión y latencia.
El uso extendido de teléfonos inteligentes, tabletas y ordenadores portátil de pequeñas dimensiones, que incluyen múltiples sensores (giróscopo, acelerómetro, etc) y

tecnologías (WIFI, Bluetooth, 4G, NFC, etc) inducen a que se investiguen nuevos sistemas de posicionamiento utilizando este tipo de dispositivos muy extendidos en los últimos años.

Otro factor importante a evaluar en un sistema de posicionamiento en interiores, es el tiempo que tarda el sistema en estimar la posición (latencia). En un sistema de posicionamiento como el que se pretende abordar en este trabajo, no sólo es suficiente con que la posición obtenida sea lo más exacta y precisa posible, sino que además se calcule muy rápidamente. En los sistemas WIFI que son uno de los más extendidos, la localización conlleva un retraso que será mayor cuanto más puntos de acceso existan en el sistema. Teniendo en cuenta que una de las principales funcionalidades del sistema de posicionamiento en interiores está orientado a su aplicación en emergencias y desastres, donde el tiempo de respuesta es primordial, se hace necesario, al menos para este tipo de tareas, una tecnología más eficiente.

Algunos autores [2], clasifican las técnicas de posicionamiento en interiores de la siguiente forma:

- Técnicas fundamentadas en el análisis del escenario.
- Técnicas fundamentadas en la triangulación.
- Técnicas fundamentadas en proximidad.

A partir de la bibliografía seleccionada [2] [6] para la realización de este trabajo, se puede comprobar cómo las técnicas de análisis de escenario, también denominadas "*fingerprinting*", son las más usadas. Este tipo de técnicas son capaces de calcular la posición de un individuo u objeto en movimiento, comparando las magnitudes en el instante actual con otras realizadas previamente. Son técnicas que usan una fase de entrenamiento (*offline*) en la que se crea una base de datos con los valores de la potencia de la señal recibida de los distintos Puntos de Accesos incluidos dentro de un área de interés. Cada medida tomada corresponde a las coordenadas donde se ha tomado la medición.

En este tipo de sistemas, la siguiente fase (*online*) que consiste en obtener la posición en tiempo real, comparando los valores medidos de la señal en la fase anterior (offline) con las medidas tomadas justo en el instante en el que se quiere obtener la posición. Probablemente, ambas medidas no coincidan con lo que se necesita estimar la posición a partir de los datos recogidos en ambas fases. Para estimar la posición de la forma más precisa se recurre a técnicas matemáticas como son: los Métodos de Clasificación Bayesianos, Redes Neuronales, Vecino más cercano y Árboles de decisión.

Las principales ventajas de este sistema consisten en que no necesitan conocer la posición de cada antena y el sistema puede seguir operando, aunque alguna de las antenas deje de funcionar.

Además del análisis del escenario, existen otras técnicas en las que no se necesita fase de entrenamiento, son los denominados "Modelos de pérdida de propagación RF". En este caso, se hace necesario conocer previamente la posición exacta de los Puntos de Accesos para tenerlos como puntos de referencia para estimar la posición del objeto o individuo. La posición se calcula realizando cálculos matemáticos de triangulación.

En cuanto a las técnicas de proximidad, se trata de obtener la posición relativa a partir de una red de antenas, de forma que un objeto móvil se posicionaría en la ubicación de la antena más cercana, que en definitiva será de la que reciba la señal con mayor potencia.

Este tipo de algoritmo es relativamente fácil de implementar. Puede ser implementado sobre diferentes medios físicos. Los sistemas basados en infrarojos (IR) o sistemas de identificación por frecuencias de radio (RFID) [3] están a menudo basados en este método alcanzando una precisión de 2.1 metros. Al estar cada antena separa por una distancia considerable, no existe un espacio continuo de posibles ubicaciones.

Independientemente de la tecnología empleada para lograr el posicionamiento, resulta interesante mencionar las métricas que determinarán el rendimiento del sistema. A continuación, se indican estos factores [6]:

- *Exactitud*: determina la proximidad entre la posición medida y la posición real.
- *Precisión*: viene dada por la tolerancia del sistema de posicionamiento. La tolerancia representa la proximidad entre la posición estimada por el sistema y la posición real. Cuanto menor sea la tolerancia del sistema, mayor precisión tendrá.
- *Complejidad*: para alcanzar el posicionamiento en tiempo real, es necesario que el sistema no requiera mucho tiempo de computación. Cuanto más sencillo sea el algoritmo de posicionamiento, mayor será el número de veces que se podrá ejecutar en un intervalo de tiempo establecido.
- *Robustez*: refleja la estabilidad del sistema, ante incidencias en la infraestructura del sistema.
- *Escalabilidad*: el sistema de posicionamiento debe permitir crecer y mantener su correcto funcionamiento en un escenario de mayores dimensiones.

- *Coste*: no sólo es necesario que el sistema de posicionamiento sea lo suficientemente preciso y exacto, sino que sea fácil de desplegar y a un precio asequible y razonable.

En este trabajo, se pretende aportar una solución al problema de posicionamiento en interiores, pero usando odometría visual, en lugar de tecnologías inalámbricas. A continuación, se justificará el uso de este tipo de técnicas para el cálculo de posición de un Vehículo Aéreo (UAV) en un espacio interior.

Para la localización de un UAV dentro de un espacio cerrado, las cámaras pueden aportar mayor precisión comparado con otros tipos de sensores (láser) utilizados para este mismo fin [7] . Los medidores láser deberían de ir colocados en el propio vehículo para poder identificar la distancia a la que se encuentra un obstáculo. Todo ello añadiría complejidad al sistema e incrementaría considerablemente el coste al sistema. Por el contrario, existen en el mercado una amplia gama de cámaras y resulta relativamente fácil encontrar modelos con prestaciones suficientes para la realización de este trabajo y a un coste muy bajo.

Un ejemplo muy interesante del empleo de este tipo de técnicas visuales, se puede comprobar en la bibliografía citada anteriormente [7]. Se trata de un experimento, en donde dos cámaras se conectan directamente a un ordenador que posteriormente realiza el procesado de las imágenes capturadas. El ordenador conectado a las cámaras analiza cada fotograma, detectando el vehículo y calculando su posición en tiempo real. Los autores determinan que 10Hz es una frecuencia suficiente para capturar fotografías y posicionar el vehículo en tiempo real, es decir, establecen que 10 fotogramas por segundo (10 f.p.s.) es una velocidad de captura suficiente para lograr el posicionamiento de un UAV en tiempo real.

En este caso [7], se emplean dos cámaras alineadas en paralelo en una posición fija. Ambas cámaras tienen los mismos ajustes y han sido calibradas para corregir la distorsión de la lente.

La posición del UAV se obtiene a partir de la zona de solapamiento de captura entre ambas cámaras. Las imágenes capturadas se convierten de RGB a escala de grises para facilitar su posterior tratamiento, evitando así el uso de los tres canales incluidos en cada imagen capturada. Además, a las imágenes se le aplica un filtro de mediana (que es bastante rápido) para eliminar el ruido.
Una vez realizado el preprocesado de las imágenes, se procede a la localización del

UAV, para ello se resta la imagen del espacio interior sin ningún vehículo, a las imágenes capturadas en la que aparece el vehículo, simplemente restando el valor de los píxeles entre ambas fotos. En este modelo se asume que el fondo es el primer fotograma en el que todavía no aparece ningún vehículo.

Hasta el momento, todo el procesamiento se ha realizado en paralelo con las imágenes capturadas desde cada cámara. La posición del vehículo en el espacio 3D se obtendrá a partir de ambas imágenes por medio de una ecuación matemática.

En este artículo [7], no se mencionan los detalles relacionados con la sincronización de las cámaras para conseguir realizar las capturas de las fotografías simultáneamente, tampoco se hace referencia al proceso de calibración de las cámaras, ni tampoco se especifica el algoritmo matemático empleado para obtener la posición 3D.

En cambio, en este otro artículo [10] sí que se trata el aspecto de la sincronización multicámara, incluso se llega a describir formalmente los efectos negativos de cuando no se consigue la sincronización entre capturas desde diferentes cámaras.

A través de otro artículo [5] más reciente (2016), se va a ir profundizando en aspectos necesarios dentro de los métodos de posicionamiento en interiores usando tecnología visual.

En este caso, los autores presentan un sistema de posicionamiento en interiores basado en la creación de un modelo 3D a partir de las imágenes capturadas en su interior y empleando también, los planos del edificio donde se realizan las capturas de fotos.

Al igual que los sistemas de posicionamiento basados en sistemas de radio frecuencia se ven afectados por una serie de inconvenientes (estructura del edificio, factores ambientales, interferencias con otros dispositivos, etc), los sistemas de posicionamiento que usan tecnología visual se enfrentan a otros inconvenientes como son las distorsiones producidas por las cámaras en las capturas. Esto hace que inicialmente sea necesario realizar un preproceso conocido como calibración de las cámaras.

El proceso de calibración básicamente consiste en obtener los parámetros geométricos (coordenadas del punto central y distancia focal) y físicos [11] (distorsión radial y tangencial) de la cámara que se empleará.

Teniendo en cuenta que los componentes ópticos de una cámara son imperfectos, la

proyección entre la imagen y el objeto real no es lineal y repercuten negativamente en la conversión de 2D a 3D y viceversa.

La distorsión tangencial provoca un desplazamiento de los puntos de forma perpendicular a la línea radial y está provocada por la alineación de los componentes ópticos, en cambio, la distorsión radial los desplaza de forma radial desde el centro y viene provocada por las imperfecciones de la propia lente.

El proceso de calibración inicialmente considera dos tipos de distorsiones: tangenciales y radiales, aunque la primera de ellas se despreciará al no ser significativa en el caso de las cámaras digitales.

Son muchos los posibles métodos de calibración de una cámara, en la bibliografía de este trabajo se recogen dos de ellos [4] [5]. Para la elaboración de este trabajo, se evaluaron ambas técnicas.

En la primera de ellas [4] se presenta una técnica flexible y sencilla, que consiste en que la cámara observa un determinado patrón formado por un conjunto de rectángulos (8x8) todos del mismo tamaño y separados por la misma distancia. El algoritmo identifica la posición de los vértices de cada cuadrado para calcular la distorsión de la lente y por último estima los parámetros físicos de la cámara y la distorsión radial de la lente.

La segunda opción [5], se trata de un método en el que dado los parámetros de la cámara y conociendo la posición de las cámaras, es posible proyectar con precisión cualquier punto en 3D en un punto de la imagen 2D y viceversa.

Además de las técnicas de calibración anteriores, en este trabajo se evaluarán los métodos que provee el módulo *OpenCV* [9] para la calibración de las cámaras. Esta librería, dispone de una variedad de métodos que facilitan la obtención de los parámetros intrínsecos y extrínsecos de las cámaras y también facilitan, por ejemplo, la obtención de los vértices de patrones formado por cuadrados, como los descritos anteriormente. Todo ello agilizó parte del objetivo de este trabajo.

Sería interesante resaltar dos importantes aspectos que en la bibliografía de referencia no se describen y que en este trabajo sí se profundizará:

- Un aspecto importante y complejo, consiste en la realización de múltiples capturas fotográficas desde varias cámaras, pero en el mismo instante. La

sincronización de las capturas fotográficas será una de las cuestiones más relevantes dentro de este trabajo.

- Otro aspecto destacado, consistirá en alcanzar el mayor número de capturas por segundo, fijado en un mínimo de 10 f.p.s. [7].

Además, habría que añadir que para llevar a cabo este trabajo se evaluarán dispositivos de bajo coste, como por ejemplo, una *SBC* ampliamente extendida como es *Raspberry Pi.* Ésta dispone de un bus específico denominado *CSI* (*Camera Serial Interface*) para conectar la cámara (que también comercializa el fabricante) la cual dispone de su propia librería para desarrollar con ella [8].

Referencias bibliográficas incluidas en el Estado del Arte.

[1] Karimi, H. A. (Ed.). (2015). Indoor wayfinding and navigation. CRC Press.

[2] Liu, H., Darabi, H., Banerjee, P., & Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6), 1067-1080.

[3] Seco, F., Koutsou, K., Ramos, F., & Jiménez, A. R. (2013). Localización personal en entornos interiores con tecnología RFID. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 10(3), 313-324

[4] Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE *Transactions on pattern analysis and machine intelligence, 22*(11), 1330-1334.

[5] Bjornstad, S., Maveus, N., & Pang, T. (2016). Camera Calibration and Optimization for Indoor Location.

[6] Kul, G., Özyer, T., & Tavli, B. (2014). IEEE 802.11 WLAN based real time indoor positioning: Literature survey and experimental investigations. *Procedia Computer Science, 34,* 157-164.

[7] Mustafah, Y. M., Azman, A. W., & Akbar, F. (2012). Indoor uav positioning using stereo vision sensor. *Procedia Engineering, 41*, 575-579.

[8] *Pi camera documentation* [en línea]
http://picamera.readthedocs.io/en/latest/index.html

[9] *Open Source Computer Vision (OpenCV)* [en línea]
http://opencv.org/

[10] Vibeck A. (2015) Synchronization of a multi camera system [en línea]
http://liu.diva-portal.org/smash/get/diva2:822340/FULLTEXT01.pdf

[11] Weng J., Cohen P., Herniou M. (1992) Camera Calibration with Distortion Models and Accuracy Evaluation. IEEE Ttransactions on pattern analysis and machine intelligence. Volume 14, Number 14

3. Análisis, diseño e implementación.

A continuación, se pasará a profundizar en las etapas del ciclo de vida de desarrollo.

Se ha dividido el contenido de este capítulo en dos partes bien diferenciadas:

- *Parte 1*: destinada a recopilar las decisiones previas a la realización del diseño e implementación.
- *Parte 2*: Incluye la descripción de la Estructura de la solución.

3.1 Análisis

En esta etapa se comenzará explicando las principales decisiones tomadas en esta fase de análisis.

A continuación, y a través de los diagramas *UML* de distribución y de casos de uso, se irá describiendo la solución propuesta.

3.1.1 Requisitos no funcionales

A continuación, se indican las decisiones tomadas más importantes durante esta fase de análisis:

- Se determina el tipo de dispositivos a usar: cámaras y *SBC*. También se determina el número de equipos *SBC* necesarios y su forma de interconexión.
- Se descarta el uso de Sistemas en tiempo real por la complejidad que añadiría.
- Se descubre la necesidad de crear un dispositivo óptico para medir el tiempo y que pueda ser fotografiado para determinar el grado de sincronización entre capturas realizadas desde dos cámaras distintas.
- Se seleccionan los módulos en *Python* que ayudarán a la implementación: como por ejemplo para el tratamiento de imágenes, gestión de puertos GPIO y serie, y representación gráfica.

3.1.2 Diagrama de distribución

El siguiente diagrama de distribución UML, muestra la arquitectura física del sistema de posicionamiento de un dron en un espacio interior.

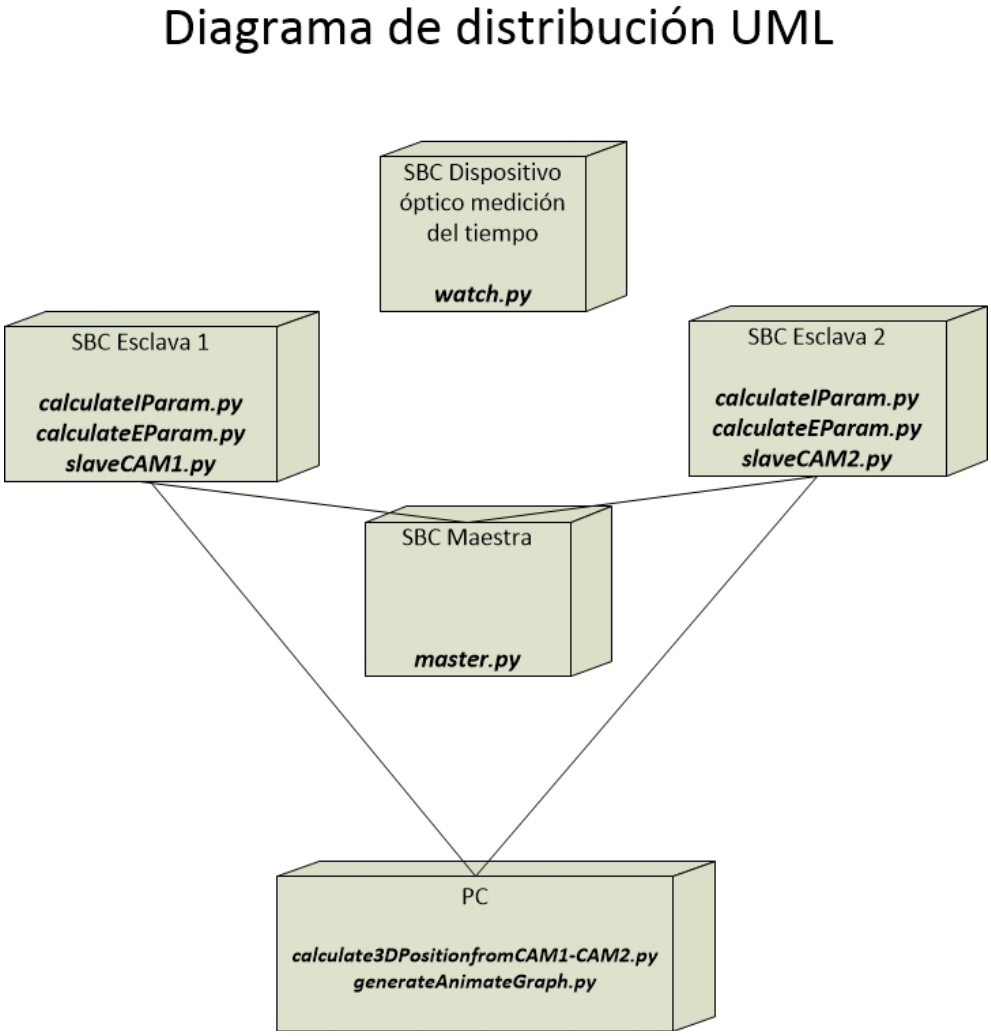


Figura 1: Diagrama de distribución UML

En el diagrama anterior, se representan los dispositivos y equipos que integran el sistema de posicionamiento. Cada cubo representa un ordenador: *SBC dispositivo óptico*, *SBC Esclava 1*, *SBC Esclava 2*, *SBC Maestra y PC*.

Las líneas de conexión que relacionan los cubos identifican las conexiones reales existentes entre los diferentes dispositivos. Por último, en cada cubo se identifica el software implementado que se ejecutará en cada máquina.

3.1.3 Diagrama de casos de uso

En la imágenes siguientes, se muestra como existen dos sistemas independientes:

- El primero representa los casos de uso del dispositivo óptico para la medición del tiempo.
- El segundo representa la solución implementada para el posicionamiento de un dron en un espacio interior.

En la primera figura, se puede comprobar como el sistema del dispositivo óptico para la medición del tiempo tiene un único actor y sus acciones están relacionadas con la gestión de un conjunto de leds. Se utilizará para medir el grado de sincronización de las capturas realizadas desde las dos cámaras que se utilizarán en este trabajo.

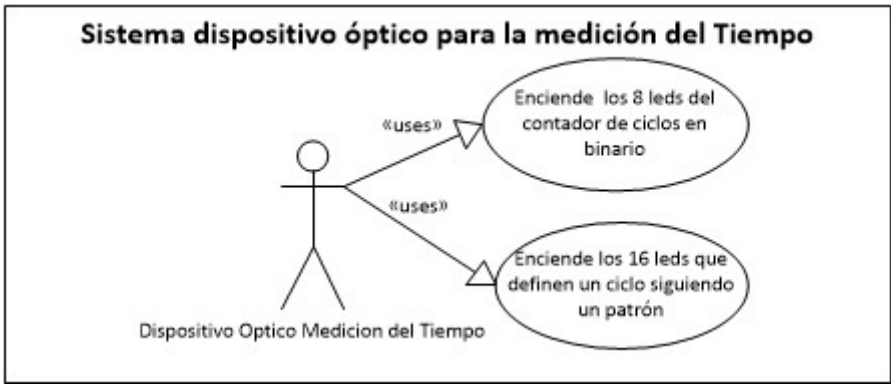


Figura 2: Diagrama de casos del dispositivo óptico para la medición del tiempo

En la segunda figura, se puede comprobar cómo el sistema de posicionamiento de un dron en un espacio interior tiene cuatro actores, que coinciden con los tres ordenadores *SBC* más el PC que integra nuestro sistema .

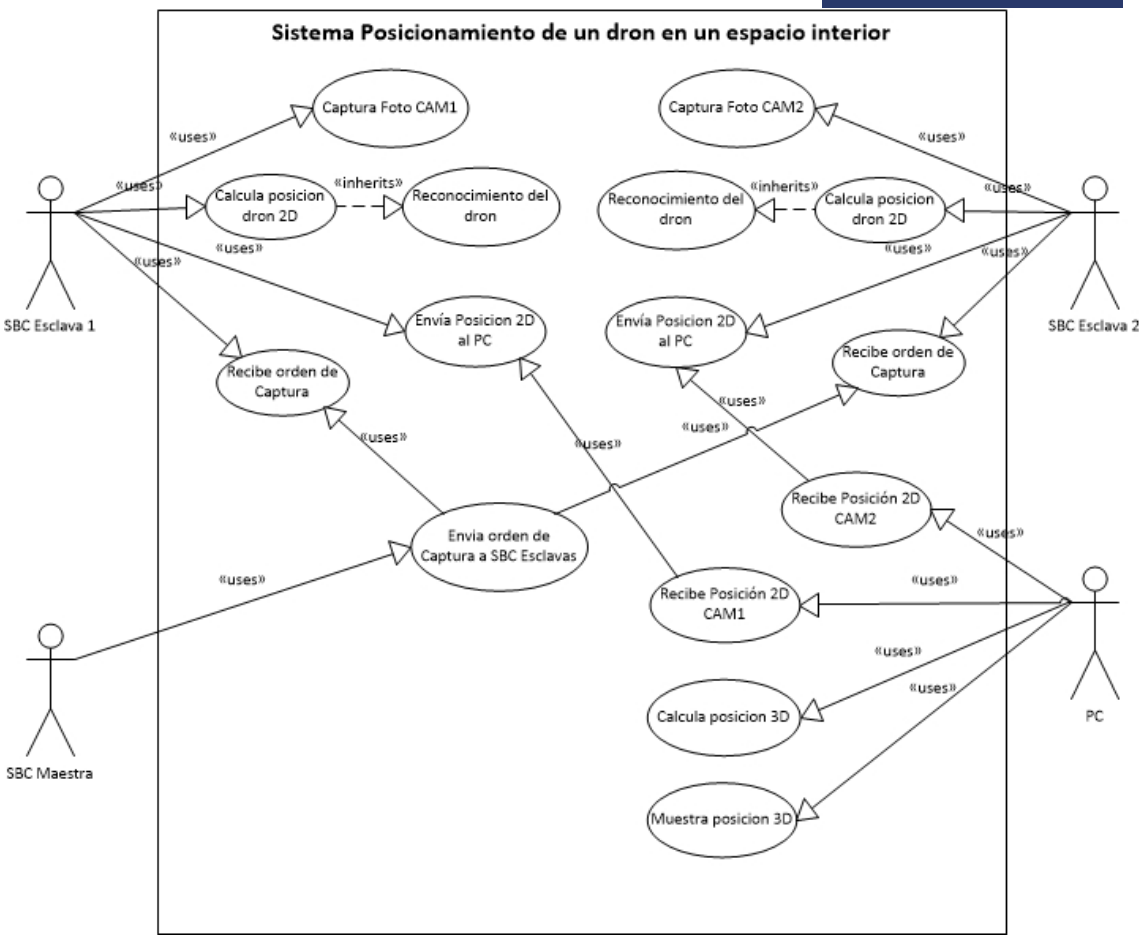


Figura 3: Diagrama de casos de uso del sistema de posicionamiento en un espacio interior

3.2 Diseño e implementación.

Se ha dividido el contenido de este capítulo en dos partes bien diferenciadas:

- *Parte 1*: destinada a recopilar las decisiones previas a la realización del diseño e implementación.

- *Parte 2*: Incluye la descripción de la Estructura de la solución.

3.2.1 Especificaciones.

En esta primera parte se describen las decisiones más importantes que se han tomado para realizar este trabajo.

Se ha optado por utilizar un conjunto de librerías en *Python* para agilizar la implementación de este trabajo. Las librerías utilizadas han sido:

- ***OpenCV***: Se trata de una librería diseñada para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real.
- ***Picamera***: Es una librería específicamente creada para interactuar con la cámara *Raspberry Pi Camera,* modelo de cámara usado en este trabajo.
- ***RPi.GPIO***: Se trata de una librería que permite gestionar los 40 pines del puerto *GPIO (General Purpose Input Output)* de la las *Raspberry Pi* y ha permitido la interacción entre la SBC Maestra y las esclavas.
- ***Numpy***: Es una librería orientada a la computación científica con Python.
- ***Matplotlib***: Se trata de una librería para la representación gráfica en 2D desde Python.
- ***Serial:*** Es un módulo para *Python* creado para la gestión del puerto serie.

Tras detectarse la necesidad de elaborar nuestro propio dispositivo óptico de medición del tiempo, basado en un ordenador de placa única (*SBC*) y un conjunto de leds (*light emitting diode*) fue necesario implementar un programa en *Python*, que no interactuará con ningún otro dispositivo pero que permitirá medir el grado de sincronización en la captura simultánea de imágenes desde dos cámaras diferentes.

Para lograr el posicionamiento de un dron en movimiento en un espacio cerrado, se implementó primeramente, un programa para realizar el proceso de calibrado de las cámaras y determinar las características ópticas, geométricas y digitales de cada cámara. También, fue necesario implementar el cálculo de los parámetros extrínsecos de las cámaras.

Además, para la realización del proceso de reconstrucción 3D del dron se implementó un sistema de reconocimiento basado en la identificación del punto de máxima luminosidad ya que previamente al dron se le añadió un led encendido en su parte superior.

Por último, se implementó el proceso de reconstrucción de una posición 3D a partir de dos imágenes capturadas (con cámaras diferentes) en el mismo instante

3.2.2 Estructura de la implementación

A continuación, se describen cada uno de los ficheros que contienen el código fuente de

la solución propuesta:

3.2.2.1 Implementación del Dispositivo Óptico para la medición del tiempo

Se trata de un componente implementado en Python que gestiona los 24 leds del dispositivo óptico. En el grupo de los 8, se contabilizarán (en formato binario) los ciclos hasta un máximo de 256.

Se ha cuantificado el retraso óptimo entre el encendido y apagado de los 16 leds, de forma que se ha determinado el tiempo de cada ciclo.

El fichero que contiene la implementación se denomina ***watch.py***:

3.2.2.2 Implementación del Dispositivo Maestro.

Se trata de un componente implementado en Python que activa las dos cámaras de forma sincronizada para la realización de capturas de forma consecutiva.

El fichero que contiene la implementación se denomina ***master.py***:

3.2.2.3 Implementación de los Dispositivo Esclavos.

Se trata un módulo implementado en Python, que se ejecuta en las SBC esclavas. Desde este módulo se gestiona la orden de captura de foto recibida por parte de la SBC Maestra. Además, este módulo incluye el reconocimiento del dron y de su posición, para ello se identifica el led que lleva situado en la parte superior. La identificación del led se realiza obteniendo el punto de mayor luminosidad que establecerá la posición del dron.

El fichero que contiene la implementación se denomina ***slaveCAM1.py*** que es el módulo que se ejecuta en la Raspberry Pi que tiene la Cámara 1 conectada:

El fichero que contiene la implementación se denomina ***slaveCAM2.py*** que es el módulo que se ejecuta en la Raspberry pi que tiene la Cámara 2 conectada:

3.2.2.4 Implementación del PC.

El fichero que contiene la implementación se denomina

calculate3DPositionfromCAM1-CAM2.py que es el módulo que se ejecuta en el PC para recoger las posiciones que las SBC esclavas envían después de haber identificado la posición del dron. Además, una vez recogidas las posiciones se reconstruye la posición 3D y se va mostrando por pantalla.

3.2.2.5 Implementación realizada para el cálculo de los parámetros intrínsecos de las cámaras.

El fichero que contiene la implementación se denomina ***calculateIParam.py*** que es el módulo que se ejecuta en cada *SBC* con cámara para calcular los parámetro intrínsecos a partir de las fotos realizadas al damero desde cada cámara.

3.2.2.6 Implementación realizada para el cálculo de los parámetros extrínsecos de las cámaras.

El fichero que contiene la implementación se denomina ***calculateEParam.py*** que es el módulo que se ejecuta en cada *SBC* con cámara, para calcular los parámetros extrínsecos. Estos parámetros se calculan a partir de los parámetros intrínsecos (ya calculados anteriormente) y de una serie de puntos cuyas coordenadas son conocidas en el espacio y en cada imagen tomada por cada cámara.

3.2.2.7 Generar el gráfico animado.

Se trata del módulo que se ejecuta en el PC. Desde este módulo se genera un gráfico animado que representa la trayectoria que ha seguido el drone obtenida a partir de las capturas fotográficas realizadas desde ambas cámaras.

El fichero que contiene la implementación se denomina ***generateAnimateGraph.py***

4. Dispositivo óptico de medición del tiempo.

4.1 Introducción

En este capítulo, se describe una solución a uno de los problemas más complejos abordados en este proyecto: medir el grado de sincronización en la captura de fotos realizadas desde dos cámaras en el mismo instante.

Inicialmente se buscó y analizó referencias bibliográficas relacionadas con este tema para evaluar cómo otros autores lo habían resuelto. Aunque existe constancia de que este problema ya fue abordado en 1923 [1], se ha comprobado que la bibliografía sobre este tema es muy reducida. Además, se observa que en las referencias bibliográficas [2], [3] de mayor interés para este trabajo, se emplean dispositivos ópticos preconstruidos (basados en leds) para la medición del tiempo. En concreto, usan 40 y 128 leds respectivamente para alcanzar una precisión de 1 milisegundo. También habría que mencionar que los autores no aportan muchos detalles sobre el dispositivo utilizado para la medición del tiempo.

Antes de comenzar a describir la solución original aportada en este trabajo de investigación, resulta interesante exponer los distintos prototipos que se fueron creando hasta llegar al dispositivo óptico de medición del tiempo final.

Se ha tratado de encontrar una solución al problema de la medición del tiempo pero de una forma sencilla de implementar y de bajo coste, evitando utilizar complejos dispositivos especializados.

Para ello, se ha creado nuestro propio dispositivo óptico de medición del tiempo, basado en un ordenador de placa única (*SBC*) y un conjunto de leds (*light emitting diode*). En nuestro caso, 24 leds han sido suficientes para alcanzar una precisión de 1 milisegundo. El dispositivo creado, todavía permite reducir la precisión significativamente de dos formas: reduciendo el retraso de encendido y apagado de cada led, o bien, incrementando el número de leds.

Al realizar simultáneamente fotografías desde dos cámaras, a nuestro dispositivo óptico de medición del tiempo, el estado de los leds en ambas capturas identificará de una forma precisa el desfase existente entre cada imagen tomada.

4.2 Prototipos

A continuación, y debido a la importancia de este dispositivo dentro de este trabajo, se procederá a ir describiendo cada uno de los prototipos que se fueron construyendo hasta llegar al prototipo final.

4.2.1 Solución 1: Uso de dispositivos móviles.

En un principio, para la realización de los tests de sincronización (entre capturas desde dos cámaras) se emplearon los cronómetros incluidos en los dispositivos móviles como son los teléfonos y tabletas.

Motivación:
Las principales razones que hicieron comenzar a realizar pruebas con este tipo de dispositivos fueron las siguientes:

- Fácil acceso a este tipo de dispositivos.
- Se dispone de una gran pantalla en un dispositivo de reducido tamaño
- Los dígitos del cronómetro se representaban en un tamaño bastante grande lo que beneficiaba la realización de fotos.
- Por último, alguno cronómetros contaban con una precisión de milésimas de segundos.

Problemas encontrados:

Tras diversos tests realizando fotografías a diferentes cronómetros, con distinto hardware y distinto sistema operativo, se comprobó la imprecisión con la que contaban todos ellos en la representación por pantalla que realizan del tiempo (a nivel de milésismas y centésimas de segundo). Se observó que los cronómetros internamente miden el tiempo correctamente pero no lo representan por pantalla de manera precisa.

En la secuencia de imágenes siguientes, realizadas a alta velocidad, se puede comprobar cómo las centésimas de segundos comienzan en 0 y el valor siguiente es 4, es decir, se puede afirmar que en la representación de las centésimas de segundo, se omiten valores intermedios.

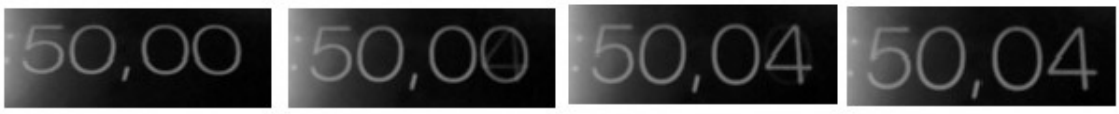


Figura 4: Serie de fotos realizadas al cronómetro de un dispositivo móvil

Además, se comprobó que el intervalo de imprecisión en la representación por pantalla

no era constante. Todo ello, hizo que hubiera que descartar este tipo de dispositivos para el cálculo de la diferencia de tiempo entre dos imágenes tomadas desde dos cámaras.

4.2.2 Solución 2: Uso de circuitos LCD.

A continuación, se realizaron nuevos tests de sincronización empleando un circuito LCD conectado a un ordenador de placa única (SBC).

Motivación:
Las principales razones que hicieron comenzar a realizar pruebas con este tipo de dispositivos fueron las siguientes:

- Fácil acceso a este tipo de dispositivos y bajo coste.
- Se trata de un dispositivo programable que permite la representación de texto y números. Conectado a un SBC se podría programar un contador que fuera tan rápido como permitiera el SBC.

Problemas encontrados:

Tras diversos tests realizando fotografías al dispositivo, se comprueba de nuevo que la representación por pantalla sigue siento también lenta, a nivel de centésimas y milésimas de segundo.

Al tratar de usar un circuito LCD para representar un contador que variaba cada milésima de segundo, se observó que la tasa de refresco del dispositivo LCD era muy baja, lo que hacía que se sobrescribieran los dígitos. En la imagen siguiente se puede comprobar la representación del contador en el LCD al realizarle una foto a alta velocidad:

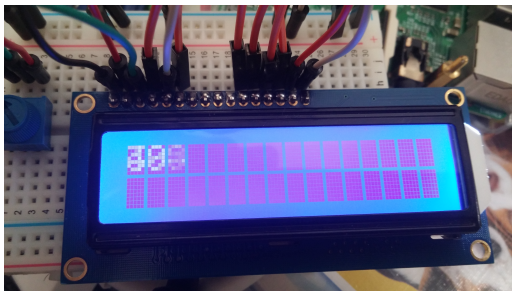


Figura 5: Circuito LCD mostrando un contador

También se implementó un contador en formato binario. El resultado fue el mismo pero esta vez sólo en los bits situados más a la derecha, que son los que cambian con mayor frecuencia. En la imagen siguiente se puede comprobar lo descrito anteriormente:

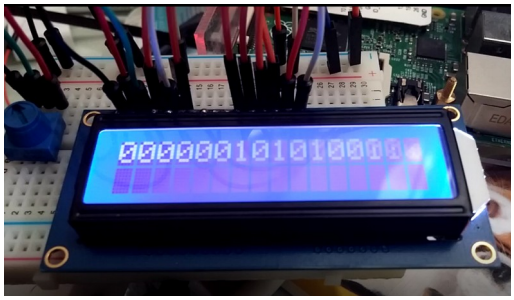


Figura 6: Circuito LCD mostrando un contador binario

4.2.3 Solución 3: Uso de matriz de leds.

A continuación, para la realización de nuevos tests de sincronización se utilizó un dispositivo que incluía una matriz de leds.

Un led tiene la capacidad de entenderse en un tiempo muy corto (menor de 1 milisegundo), controlando el encendido secuencial de los leds de una matriz, se podría medir el tiempo y al poderse fotografiar fácilmente, se podría obtener la diferencia de tiempo entre dos fotografías.

Inicialmente se plantea la idea de que la forma más sencilla de mostrar la evolución del tiempo transcurrido empleando leds, consistiría en encender 10 leds secuencialmente cada milisegundo, otros 10 leds cada centésima de segundo y otros 10 leds cada décima de segundo.

Para la realización de los nuevos tests, se utilizó una matriz de 64 leds (8x8)

Motivación:
Las principales razones que hicieron comenzar a realizar pruebas con este tipo de dispositivos fueron las siguientes:

- Dispositivo de reducido tamaño y fácil conexión al SBC
- La matriz de leds viene con una librería que facilita su programación.

Problemas encontrados:

Tras diversos tests realizando fotografías a la matriz de leds, se comprueba que el comportamiento de la matriz de leds (a niveles de centésimas y milésimas de segundo) era bastante impreciso, ya que las fotografías mostraban como los leds no representaban

la secuencia de encendido y apagado que se les estaba ordenando.

Para realizar la medición del tiempo a través de este dispositivo, lo que se hizo fue, usar dos columnas de leds de la matriz para representar los milisegundos (8 leds de una columna y los otros dos de la siguiente columna). Lo mismo se hizo para las centésimas de segundo y décimas de segundo. En total, se usaron 6 columnas de la matriz, y se dejaron dos filas libres para separar milisegundos de las centésimas de segundo, y otra para separar las centésimas de las décimas.

En la imagen siguiente, se puede comprobar los leds que se emplearon para representar las décimas, centésimas y milésimas de segundo, así como la dirección de encendido secuencial de los leds.

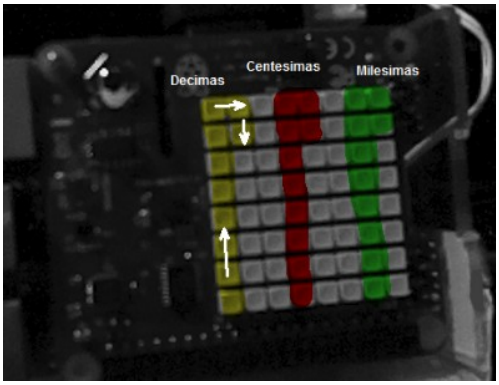


Figura 7: Matriz de leds identificando milésimas, centésimas y décimas de segundo

Tras del encendido de cada led dentro de sus respectivas columnas, se introduce un retraso de una milésima de segundo a los leds que identifican los milisegundos, 1 centésima de segundo a los leds encargados de representar centésimas y una décima de segundo a los leds encargados de representar las décimas.

Se verifica que añadiendo un retardo superior a 0.025 segundos el patrón comienza a ser el esperado, con retardos inferiores el encendido y apagado de los leds es incorrecto.

En la imagen siguiente se puede comprobar cómo donde debería haber encendido al menos 3 leds (y en diferentes columnas: décimas, centésimas y milésimas), aparecen dos leds en la columna que identifica las milésimas de segundo y el resto permanecen apagadas.

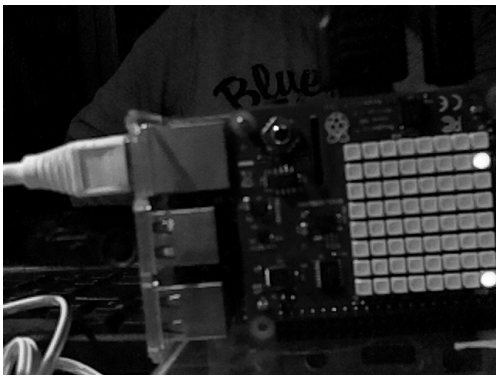


Figura 8: Matriz de Led mostrando un estado incorrecto

4.2.4 Solución 4: Uso de leds.

Finalmente para la realización de los tests de sincronización de capturas entre dos cámaras, se decide utilizar un conjunto de leds independientes. En la imagen siguiente se puede apreciar el diseño del dispositivo empleando 16 leds.

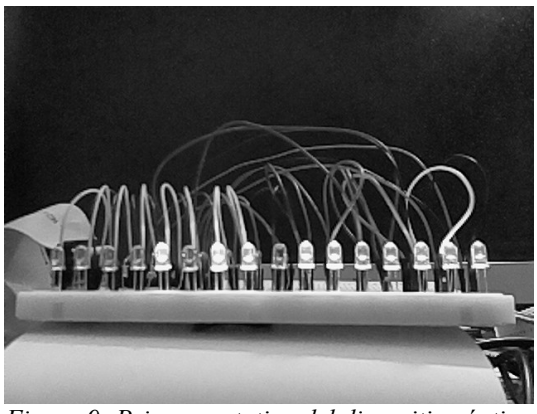


Figura 9: Primer prototipo del dispositivo óptico creado para la medición del tiempo

Motivación:
Las principales razones que hicieron comenzar a realizar pruebas con este tipo de dispositivos fueron las siguientes:

- Solución totalmente flexible y de bajo coste.
- Fácil acceso al material necesario.

Problemas encontrados:

Se implementa un contador en formato binario con 16 leds donde el valor máximo que se puede representar será 65.535. Tras diversos tests se observa que los 4 leds situados

más a la derecha permanecen siempre encendidos, ya que son los que cambian con mayor frecuencia. Se comprueba que este comportamiento es debido a que el tiempo de apagado de los leds es mucho mayor al de encendido.
En la siguiente serie de imágenes, se puede comprobar cómo, de los 16 leds, los 4 más a la derecha parpadean tan rápido que parece que siempre permanecen encendidos

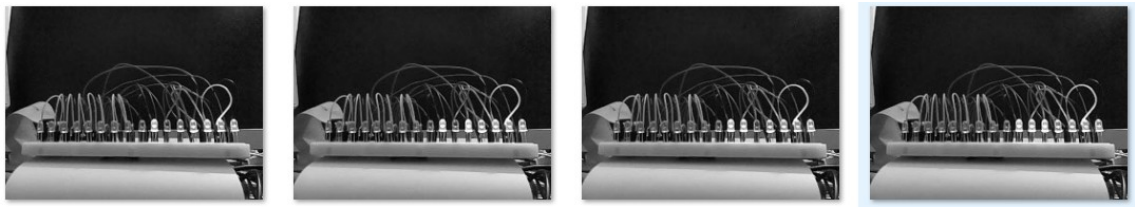


Figura 10: Serie de fotos mostrando los 4 últimos leds siempre encendidos

4.3 Implementación final del Dispositivo Óptico para la medición del Tiempo

A partir de todos los problemas descritos anteriormente, se llega a la conclusión que hay que incrementar el número de leds hasta conseguir que los leds más a la derecha del contador binario parpadeen más lentamente.

Finalmente, para la implementación del dispositivo óptico de medición del tiempo, se han utilizado 24 leds. En este caso, los leds se dividen en dos grupos: uno con 8 leds y otro de 16 leds.

El ciclo de reloj vendrá determinado por el encendido y apagado secuencial de cada uno de los leds del grupo de los 16. En el grupo de los 8, se contabilizarán (en formato binario) los ciclos hasta un máximo de 256.

Teniendo en cuenta, todo lo descrito anteriormente, el reloj que se ha implementado tiene 2^{12} estados, es decir, un total de 4096 estados.

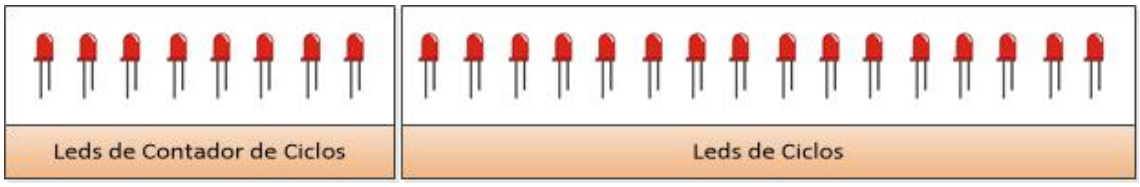


Figura 11: Diseño del dispositivo óptico final para medir el tiempo

Se ha cuantificado el retraso óptimo entre el encendido y apagado de los 16 leds, de

forma que se ha determinado el tiempo de cada ciclo.

Para determinar la velocidad del sistema, inicialmente se ha medido el tiempo en cambiar el led más a la izquierda del grupo de los 8, determinado así, el número de veces que cambia el led por segundo. El siguiente led de este grupo, cambiará el valor anterior multiplicado por 2, el siguiente se multiplicará por 4 y así consecutivamente hasta el led 8.

En otras palabras, supongamos que el led más situado a la izquierda, en M segundos cambia N veces. Eso indica que cambia N/M veces cada segundo. Eso quiere decir que el segundo led cambia 2*N/M veces cada segundo. Y el tercer led cambia 4*N/M veces cada segundo, y así sucesivamente hasta el led 16 que cambia 2^15*N/M veces cada segundo. Todo ello ofrece como resultado la frecuencia del reloj de leds.

Si el led más a la izquierda parpadea cada 2.27 segundos, el segundo lo hace cada 2.27/2, el tercero cada 2.27/4, el cuarto cada 2.27/8, el quinto cada 2.27/16, el sexto cada 2.27/32, el séptimo cada 2.27/64 y el octavo cada 2.27/128. Y como este parpadea cada vez que haces un recorrido a través de los 16 Leds, el intervalo entre led y led es de (2.27/128)/16 = 0.0011084 segundos. O sea, aproximadamente una milésima.

Se ha añadido un retraso controlado en el apagado de los leds de *0.00089333 sg* para que el retardo entre led y led del grupo de los 16 sea de un *0.001 sg*. Aproximadamente.

En la figura siguiente, se representa de manera gráfica el retardo provocado.

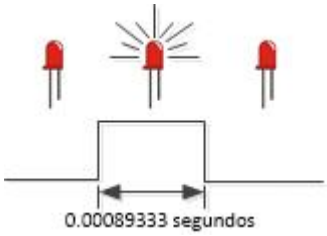


Figura 12: Representación gráfica del retardo controlado provocado en los leds del grupo de los 16

Para ello, el led más a la izquierda del contador de vueltas parpadeará cada 2,048 segundos.

Se ha elegido un M suficientemente grande (10), ya que cuanto mayor sea, más preciso

serás en el cálculo del cociente M/N (frecuencia de cambio del LED a la izquierda del todo)

Además, en el contador del grupo de los 16 leds hay que mirar cual es el último led encendido (si hay varios encendidos, se escoge el que está situado más a la derecha de los que están encendidos, .esto aporta un valor de uno a 16. que hay que sumar al computo total

El tiempo total se calculará a partir de la siguiente fórmula:

$$tiempo = Contador\ de\ Ciclos * 16 + Contador\ grupo\ de\ los\ 16\ leds$$

A continuación, se aplicará la fórmula anterior a un ejemplo real. Para ello, se realizarán dos imágenes a la vez del reloj de leds con dos cámaras diferentes. El reloj permitirá, conocer el desfase que hay entre ellas y la precisión del sistema.

Cámara 1	**Cámara 2**

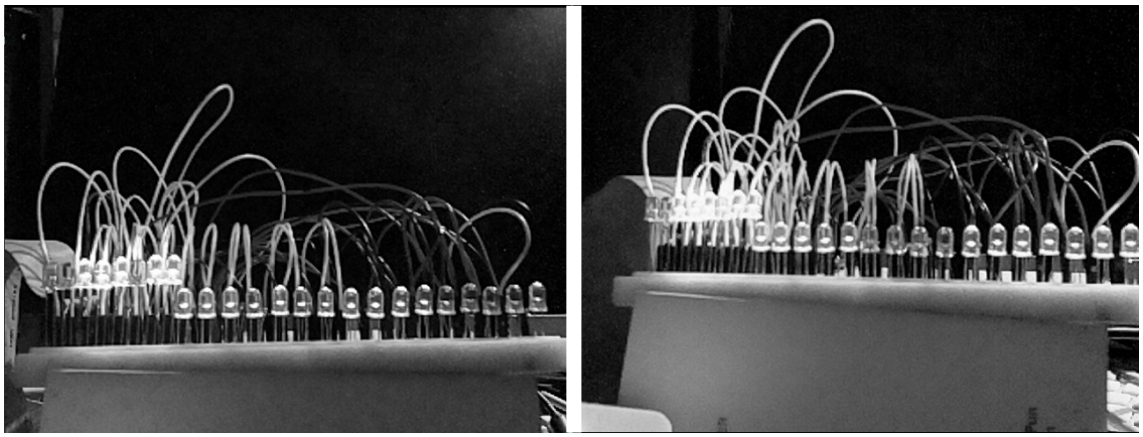


Figura 13: Ejemplo de capturas realizadas al dispositivo con las dos cámaras en el mismo instante

A continuación, se representará en binario el estado de los leds capturados por cada cámara. Por un lado se representará el contador de ciclos y por otro lado el grupo de los 16 leds.

Cámara 1	Cámara 2
00111011-*1111111111110001*	**00111111-*1111100001111111***

Ahora, se pasa a decimal el contador de ciclos.

Cámara 1	Cámara 2
00111011 en decimal es 59	**00111111 en decimal es 63**

A continuación, teniendo en cuenta que cada ciclo equivale a 16ms, se realiza la operación.

Cámara 1	Cámara 2
(59 ciclos * 16 ms)=944 ms	**(63 ciclos *16 ms)=1008**

Para continuar, se analiza el estado de los leds del grupo de los 16, y tal cómo se describió anteriormente, se contabiliza el último led encendido antes del primer led apagado comenzando desde el primer led encendido desde la izquierda.

Cámara 1	Cámara 2
1111111111110001 ***El último led encendido está en la posición 12*** ***12x 1ms = 12ms***	***1111100001111111*** ***El último led encendido está en la posición 5*** ***5x1ms=5ms***

Por último, se suma el valor que resulta del contador de ciclos y el valor obtenido analizando el grupo de los 16 leds. Repitiendo la operación por separado para cada cámara.

Cámara 1	Cámara 2
944ms + 12ms = 956 ms.	**1008 ms + 5 ms= 1013ms**
La diferencia de tiempo entre captura es: 1013-956= 57ms	

Además, analizando una secuencia de imágenes y comprobando los cambios en el reloj de leds, se puede obtener el número real de fotogramas por segundo que cada cámara es capaz de obtener.

4.4 Diseño físico del dispositivo.

Cada led se encuentra conectado a uno de los pines del puerto GPIO de la Raspberry. GPIO (General Purpose Input/Output). Se trata de un sistema de Entrada y Salida, es decir, una serie de conexiones que se pueden usar como entradas o salidas para cualquier propósito. La Raspberry cuenta 40 pines. En la figura siguiente se puede comprobar la localización y aspecto del puerto GPIO.

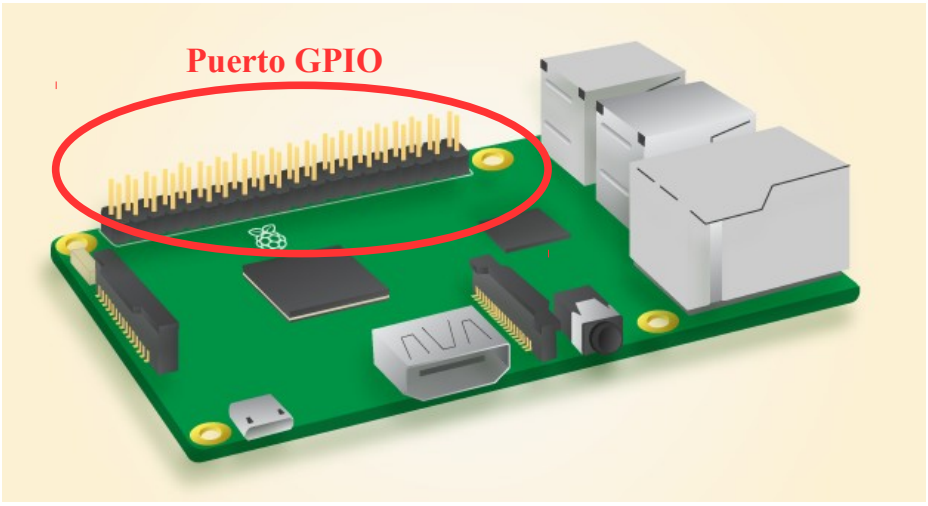


Figura 14: Localización y aspecto del Puerto GPIO de la Raspberry Pi 3

Para la elaboración del dispositivo óptico de medición del tiempo sólo se usarán 24 pines para manejar los leds. Cada uno de estos 24 pines estará conectado físicamente a un led. En la figura siguiente se puede ver el conexionado entre los leds y el puerto GPIO.

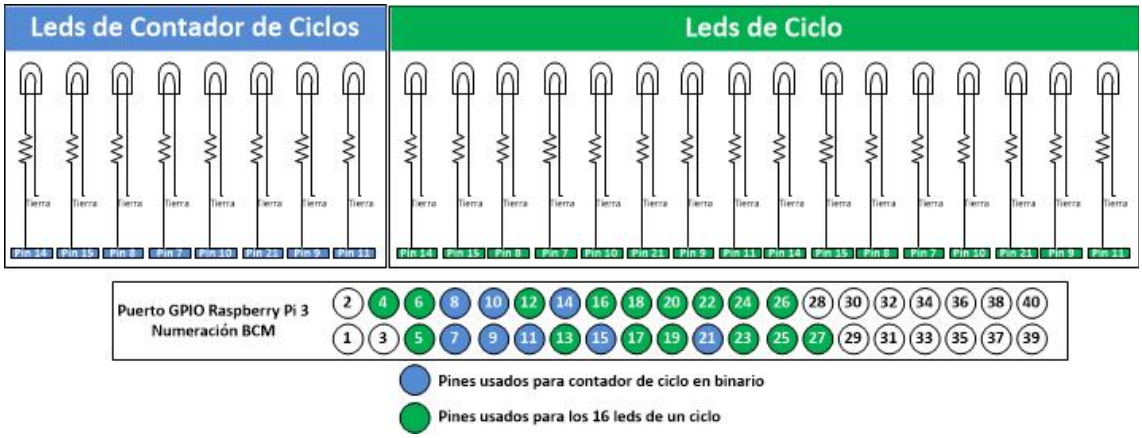


Figura 15: Esquema básico de conexión entre los leds del dispositivo y la Raspberry

Para la activación/desactivación de los pines se ha implementado un script en Python. A través de sus órdenes correspondientes, se realizarán los impulsos necesarios (y a intervalos regulares) para conseguir el encendido/apagado de los leds siguiendo una secuencia establecida que permitirá la medición del tiempo, de forma equivalente a un péndulo o balancín en un reloj mecánico. Se considera que se ha implementado un reloj estable ya que su frecuencia es constante.

4.5 Análisis errores.

Para el tratamiento de los errores seguiremos el riguroso protocolo establecido en la bibliografía especializada [4].

A continuación, se describirá el proceso para el cálculo del error cometido por el dispositivo creado para la medición del tiempo. Para ello, examinando los 8 leds que cuentan el número de ciclos, se medirá el tiempo que el dispositivo emplea en pasar desde su estado inicial (todos los leds apagados) hasta su estado final (todos los leds encendidos). Para obtener mayor precisión, cada test medirá el tiempo transcurrido en realizar diez veces consecutivas el proceso anterior.

En primer lugar, se procederá a determinar el ***Error aleatorio*** o ***desviación típica*** que vendrá determinado por la siguiente expresión:

$$\partial_a \equiv \sigma_{N-1} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\bar{x}-x_i)^2}$$

Donde:

- $\bar{x}$ es el valor medio de las medidas de tiempo realizadas que viene dado por la siguiente expresión:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

- N es el número de mediciones realizadas.
- x_i es cada medición de tiempo realizada.

Esta cantidad mide la repetibilidad de los datos, cuanto mayor sea la desviación típica de los valores, mayor será el error y menor será el grado de coincidencia.

En segundo lugar, se procederá a determinar el grado de dispersión de las medidas. El número de medidas necesario para que se pueda considerar preciso dependerá de la

repetibilidad de los valores obtenidos.

Para continuar, se determinarán el número de medidas a realizar siguiendo los criterios establecidos en la bibliografía [4]:

"*Se toman medidas x1, x2 y x3 de la magnitud y se calcula el valor medio y la dispersión de los valores, definida como la diferencia entre los valores extremos, es decir.* $D = x_{max} - x_{min}$"

A continuación, se calcula la ***dispersión relativa*** de los valores que vendrá determinado por la siguiente expresión:

$$d = (D/\bar{x})\,x\,100$$

Donde:

- D es la dispersión de las mediciones, que viene definida como la diferencia entre los valores extremos, es decir:

$$D = (x_{max} - x_{min})$$

La ***dispersión relativa*** vendrá dada finalmente por la siguiente expresión:

$$d = 100 * (max(x) - min(x))/\bar{x}$$

Donde:

- $max(x)$ hace referencia al valor máximo de todas las mediciones realizadas
- $min(x)$ hace referencia al valor mínimo de todas las mediciones realizadas
- $\bar{x}$ es el valor medio de las medidas de tiempo realizadas.

Si el valor de d es menor al 2%, se considera que realizando tres mediciones es suficiente para estimar el error.

A continuación, se exponen en la siguiente tabla, el tiempo de ejecución consumido por nuestro dispositivo en pasar del estado inicial (todos los leds apagados) al estado final (todos los leds encendidos) en tres pruebas diferentes:

	Tiempo de ejecución
1	*40.940611 sg.*
2	*40.974310 sg.*
3	*40.936757 sg.*

Por último, se indican los resultados obtenidos

El valor medio es:	**0.00099977**
La desv. Típica :	**5.0436e-07**
La dispersion relativa es :	**0.091703 por ciento**
El valor medido es:	**0.001 +/- 1e-05**
El error relativo es:	**1 por ciento**

En el siguiente histograma de resultados, se puede comprobar cómo tras la ejecución del reloj en 20 ocasiones, la diferencia de tiempo está distribuida en torno a una milésima de segundo.

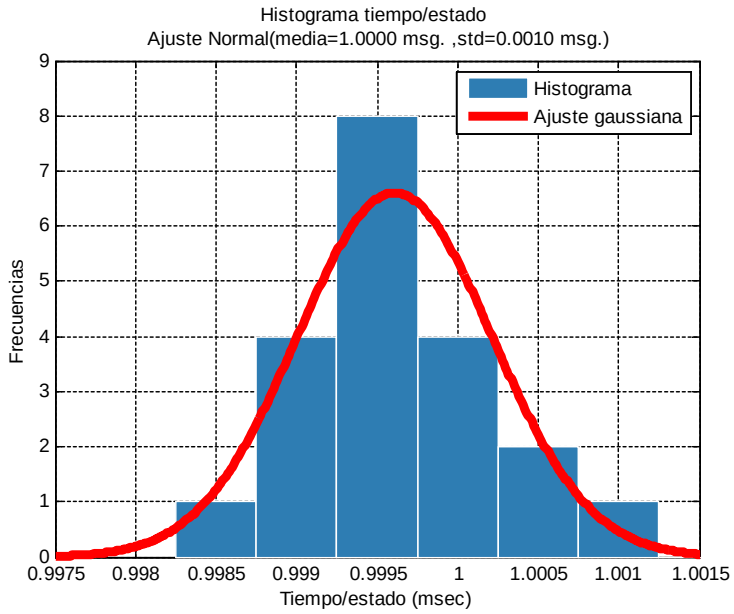


Figura 16: Histograma que muestra la distribución de la diferencia de tiempo después de la ejecución del reloj implementado en 20 ocasiones

4.6 Referencias bibliográficas incluidas en este capítulo

[1] Richardson, F.H. (1923, October). Importance of Synchronizing Taking and Camera Speeds. Transactions of S.M.P.E.

[2] Litos, G., Zabulis, X., & Triantafyllidis, G. (2006, June). Synchronous image acquisition based on network synchronization. In 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06) (pp. 167-167). IEEE.

[3] Svoboda, T., Hug, H., & Van Gool, L. (2002, September). ViRoom—low cost synchronized multicamera system and its self-calibration. In *Joint Pattern Recognition Symposium* (pp. 515-522). Springer Berlin Heidelberg.

[4] Sánchez del Río, C. (1989) Análisis de errores, Ed. Eudema

5. Interconexión y sincronizado de las cámaras.

5.1 Introducción

En este capítulo, se muestran las diferentes soluciones dadas a la interconexión de las cámaras, sus ventajas e inconvenientes y las razones que nos han permitido elegir una de estas soluciones. Además se mostrará la importancia que tiene la sincronización en la captura, así cómo los métodos desarrollados para ello.

5.2 Interconexión de dispositivos.

Actualmente, existen diferentes posibilidades de interconexión tanto para el envío de la señal de captura, cómo para la recepción de información. Estas posibilidades se podrían agrupar en dos tipos:

- *Sistemas alámbricos:* Conexión serie, interfaz GPIO, USB, etc.
- *Sistemas inalámbricos:* Wifi, Bluetooth, infrarojos, etc.

La principal ventaja de los sistemas alámbricos es su velocidad, robustez y fiabilidad. Por el contrario, su principal desventaja es que añaden complejidad en el despliegue de la infraestructura utilizada en nuestro sistema.

En cuanto a los sistemas inalámbricos, su principal ventaja es su fácil despliegue. Por el contrario, su principal desventaja es su fragilidad a factores externos como pueden ser las interferencias provocadas por otros dispositivos, o también ambientales, como por ejemplo la humedad.

En este trabajo se ha optado por un sistema alámbrico que utiliza el interfaz GPIO de los SBC y el puerto serie del PC.

En primer lugar, resulta interesante comenzar presentado un esquema básico de conexionado entre los dispositivos implicados en este trabajo.

En la siguiente figura 5.1 se puede comprobar cómo se han utilizado 4 equipos: tres *SBC* (*Raspberry 3*) y un Pc portátil. También se puede observar que dos de los *SBC* tienen una cámara (*Pi Camera*) conectada a través del interfaz *CSI (Camera Serial Interface)* de las respectivas *Raspberrys*.

Además, las dos *Raspberrys* con cámara se encuentran conectadas a través de su interfaz *GPIO (General Purpose Input/Output)* a una tercera *Raspberry.*

Para terminar la descripción, habría que especificar el tipo de conexión existente entre el PC y las dos Raspberrys realizada a través del puerto serie.

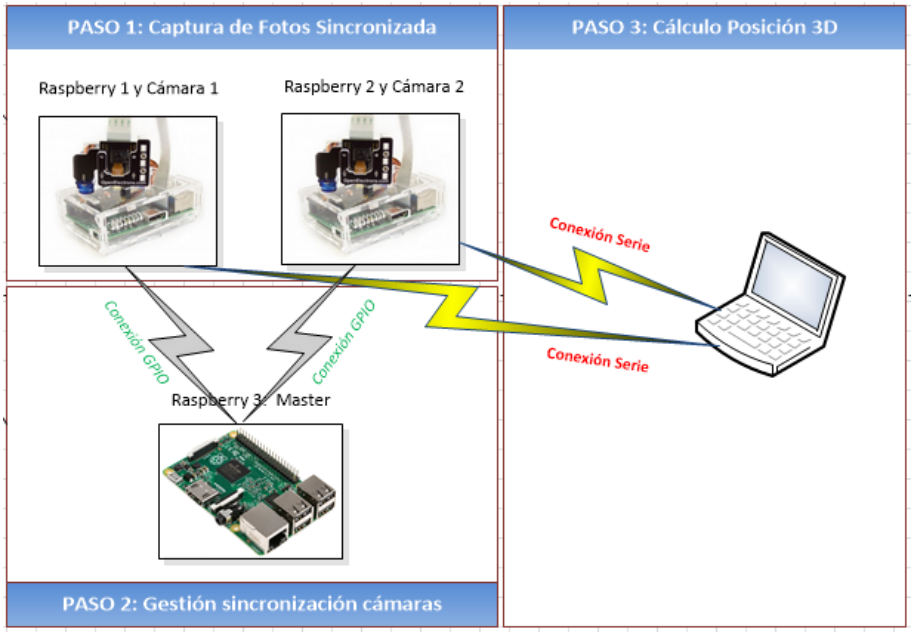


Figura 17: Esquema básico de conexión de los dispositivos implicados en este trabajo

A continuación, se profundizará sobre el interconexionado utilizado entre los distintos dispositivos. Para ello, por un lado, se describirá el conexionado implicado en la sincronización de las capturas desde las dos cámaras, y por otro lado, se describirá el conexionado relacionado con la interacción con el PC.

En la figura 5.2, se puede ver con mayor detalle cómo están relacionadas los tres dispositivos de placa única implicados en la sincronización de las capturas. También, se puede comprobar cómo entre las tres Raspberrys existe una relación que sigue el modelo Maestro-Esclavo, es decir, una de las tres (identificada en la figura como Maestra) es la encargada de mandar la orden de captura a las Raspberrys que tienen la cámara conectada (Esclavas).

Tras diversos tests, se comprobó que conectando un mismo pin del interface GPIO de cada Raspberry esclava a un único pin de la Maestra, la señal es recibida en el mismo

instante por las que tienen la cámara conectada.

Los equipos esclavos ejecutan una rutina en la que quedan a la espera de recibir la orden de captura del equipo maestro. En el instante, en el que reciben la señal de la Maestra, se inicia la captura fotográfica en cada cámara.

En la figura 5.2, se puede comprobar cómo los equipos maestro y esclavos están conectados a través del interfaz GPIO y la sincronización la consiguen a través del PIN 39, de color rojo en el diagrama. Además, también se encuentran conectadas por una toma de tierra a través del PIN 34, de color negro en el diagrama.

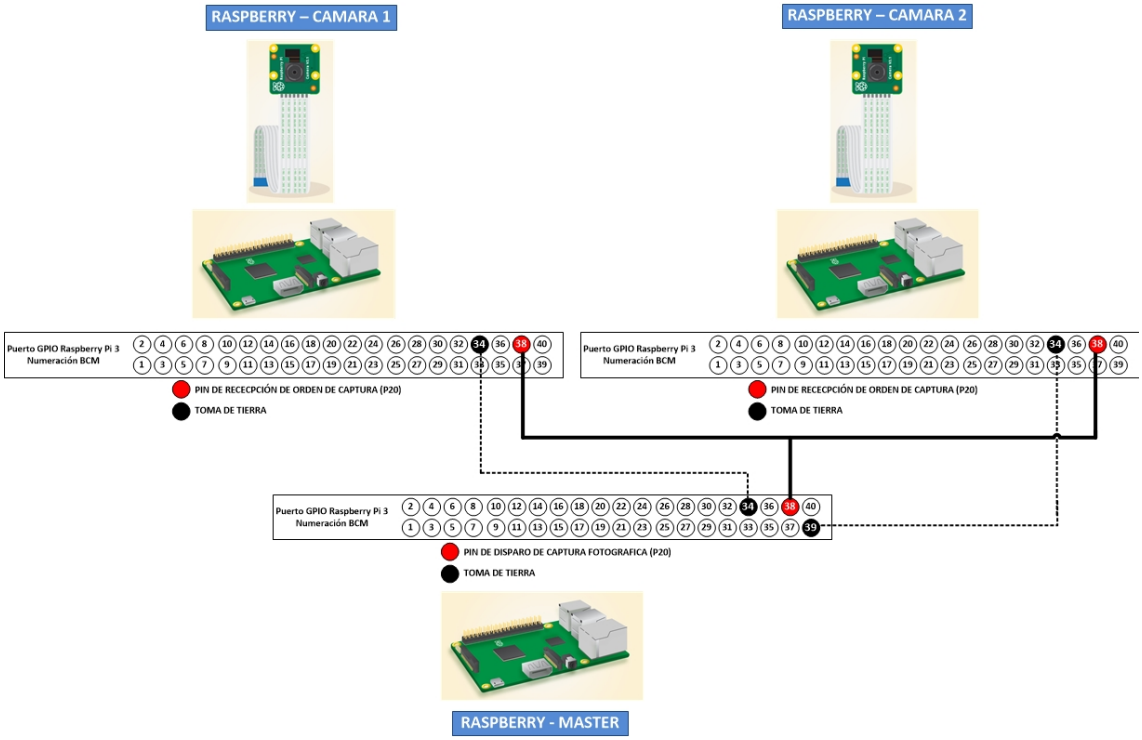


Figura 18: Esquema de conexionado entre las cámaras y la Raspberry maestra

Para continuar, se describirá el conexionado existente entre los dispositivos con cámara y el PC. Este conexionado se utiliza para el envío de la posición 2D *(x,y)* del drone desde los dispositivos con cámara al PC. Éste, a partir de ambas posiciones calcula la posición 3D *(x,y,z)*.

En la figura 5.3, se puede comprobar cómo los equipos que tienen la cámara conectada y el PC se encuentran conectados a través del puerto serie. El PC incluye dos puertos series destinados cada uno a recibir la posición del drone que le proporcionan los dispositivos con cámara. La interfaz serie de las Raspberrys se encuentra incluida dentro

los pines del *GPIO*, en concreto, en los pines **6** *(Tierra: identificado de color negro)*, **8** *(Transmisión: identificado de color Amarillo)* y ***10*** *(Recepción: identificado de color Verde)*.

Por último, indicar que el PC no disponía de puerto serie y se le añaden dos adaptadores que transforman dos puertos USB a serie.

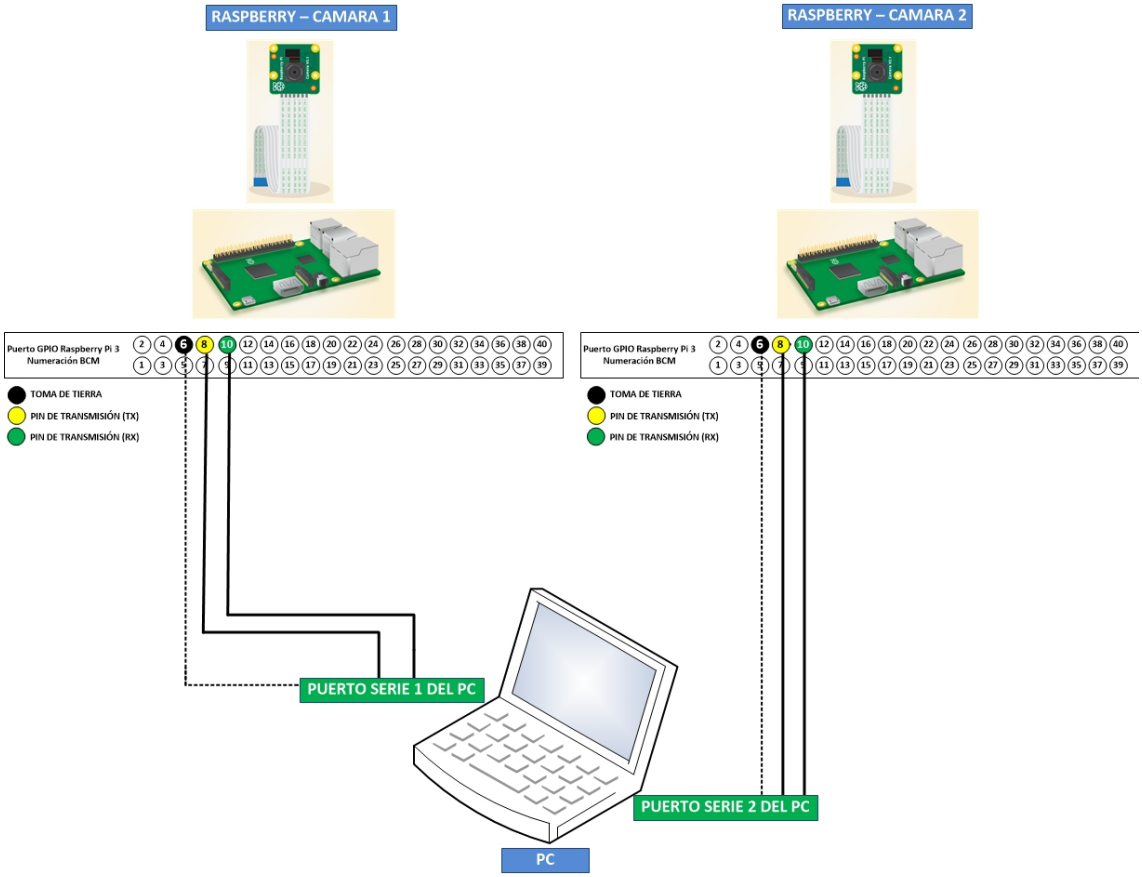


Figura 19: Esquema de conexionado entre las SBCs con cámara y el PC

5.3 Sincronización de las cámaras.

A continuación, y debido a la relevancia de esta cuestión dentro de este trabajo, se procederá a describir las dos soluciones diferentes que se implementaron para alcanzar el mayor grado de sincronización entre capturas.

Para mostrar la importancia y dimensión del problema que queremos resolver, comenzaremos haciendo una pequeña analogía. Suponiendo que un dron se mueva a la velocidad máxima de una persona corriendo (*100 m*. en *9,58 sg.*, *Usain Bolt*) recorre *100/9.58 m/s = 10,44 m/s* .Esto quiere decir que en 1 décima de segundo el dron ha recorrido *1,044 metros*, y en una centésima de segundo habrá recorrido *0,1044 metros*, es decir, *10,44 centímetros*.

Esta pequeña diferencia de tiempo es crítica a la hora de realizar la localización 3D de un dron usando varias cámaras y métodos de triangulación, ya que un pequeño retardo entre las cámaras puede producir un error enorme en el posicionamiento.

En la figura 5.4, se muestra de manera gráfica el problema descrito anteriormente y se puede observar dos casos diferentes:

1. Caso ideal en el que no hay retardo *(Retardo=0)*: los disparos de ambas cámaras coinciden en un punto P que pertenece a la trayectoria del dron.

2. Caso en el que existe un retardo *(Retardo > 0)*: se puede comprobar como este retardo provoca que exista un desajuste entre los disparos de las cámaras y el posicionamiento obtenido P* quede alejado de la trayectoria del dron.

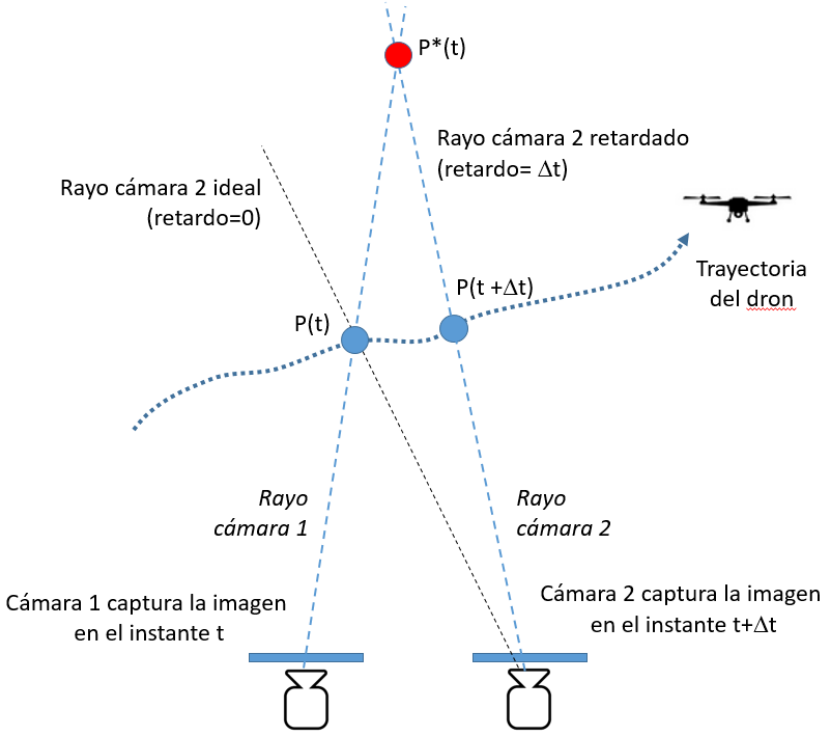


Figura 20: Gráfico explicativo sobre el problema de sincronización entre dos cámaras

5.3.1 Solución 1: Implementación que utiliza sockets a través de tecnología Ethernet.

En un principio, se consiguió realizar una aplicación usando el módulo de Python Pyro que permitía activar las cámaras desde el equipo Maestra en el mismo instante.

Motivación:
Las principales razones que hicieron comenzar a realizar pruebas con este tipo de dispositivos fueron las siguientes:

- Heredar las ventajas del uso de la tecnología Ethernet: cableado robusto, fiable y permite longitudes de 100 metros.
- Solución bastante escalable ya que para añadir más cámaras, sólo es necesario tener conectado el equipo dentro de la misma red y ejecutar el mismo script.

Problemas encontrados:

Tras diversos tests utilizando un conmutador de 5 puertos Gigabytes, se comprobó que

el desfase máximo entre las capturas de cada cámara estaba en 57 milisegundos y el mínimo en 4 milisegundos.

Tras diversos tests, se analizaron series de 10 fotos realizadas con ambas cámaras a nuestro dispositivo óptico de medición del tiempo. Para calcular el desfase entre cámara se siguió el procedimiento descrito en el apartado 4.3. En la tabla 5.1 se puede comprobar la diferencia de tiempo entre las capturas realizadas desde cámara.

Diferencia de tiempo entre capturas										
Foto	1	2	3	4	5	6	7	8	9	10
Tiempo (msg.)	*8*	*26*	*4*	*10*	*57*	*38*	*26*	*26*	*23*	*12*

Tabla 1: Diferencia de tiempos entre capturas con la Solución 1

Se comprobó que el desfase era muy variable y que los valores eran demasiado elevados. Se estimó que los retrasos probablemente estaban provocados por:

- la propia electrónica de red y las operaciones llevadas a cabo desde el nivel de Aplicación al nivel Físico dentro del modelo OSI.
- el propio módulo en *Python* utilizado: *Pyro (Python Remote Objects)*.

Se consideró que eran valores elevados y que resultaba necesario mejorar los tiempos de sincronización entre capturas

5.3.2 Solución 2: Implementación que usa el Interfaz GPIO y cableado especializado.

A continuación, se realizaron nuevos tests siguiendo el esquema indicado en la figura 5.2 y ya descrito anteriormente

Motivación:
Las principales razones que hicieron comenzar a realizar pruebas con este tipo de dispositivos fueron las siguientes:

- Más fácil de implementar.
- Mayor grado de sincronización.
- No sería necesario utilizar un conmutador de red, ni tampoco ninguna infraestructura de red.

Problemas encontrados:

Tras diversos tests, se comprobó que el desfase entre las capturas de cada cámara se redujo considerablemente, a valores del orden de un milisegundo y con muy poca variabilidad, salvo en algunos caso, como veremos en el apartado siguiente.

Los problemas que se identificaron fueron:

- Los cables de interconexión entre las interfaces GPIO de las distintas SBC fueron creados y soldados manualmente, esto añadió un mayor grado de fragilidad al conexionado.
- Se comprobó que la longitud de los cables de interconexión, facilitaba la aparición de ruidos eléctricos e interferencias pero fueron solucionados colocando varios núcleos de ferrita a lo largo de los cables.

5.4 Resultados Sincronización

Tras diversos tests, se analizaron series de 10 fotos realizadas con ambas cámaras a nuestro dispositivo óptico de medición del tiempo. Para calcular el desfase entre cámara se siguió el procedimiento descrito en el apartado 4.3.

En la tabla siguiente se puede comprobar la diferencia de tiempo entre las capturas realizadas desde cámara.

Diferencia de tiempo entre capturas										
Foto	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Tiempo (msg.)	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*22*	*1*	*1*

Tabla 2: Diferencia de tiempos entre capturas con la Solución 2

A continuación, se muestra la serie de fotos realizadas por ambas cámaras al dispositivo óptico para la medición del tiempo, donde se pueden comprobar los resultados de la tabla anterior 5.2

Este análisis permite determinar que vamos a disponer una reconstrucción 3d de la posición del dron cada m = 150.7903 ms de media. Asumiendo que los datos estén distribuidos según una gaussiana podemos determinar que el 99.73% de la diferencia entre dos reconstrucciones consecutivas será inferior a m+3*s = 267.3793 mseg.

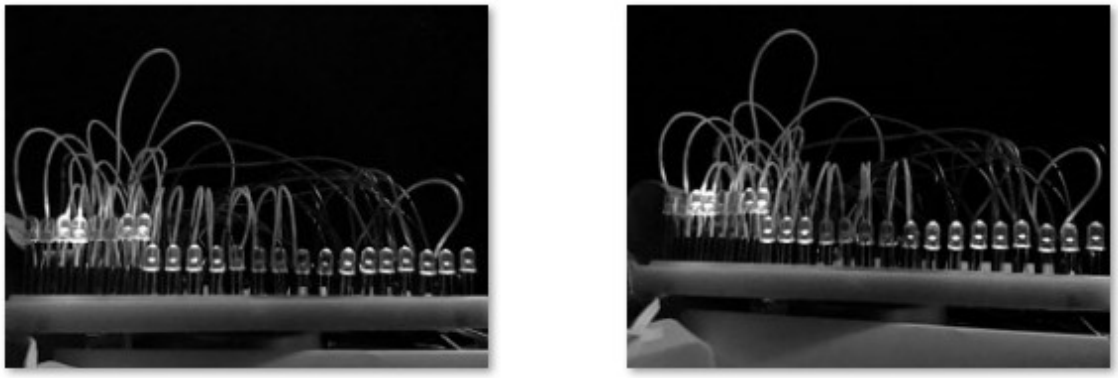


Figura 21: Foto 1 realizada al Reloj desde las cámaras 1 y 2 respectivamente

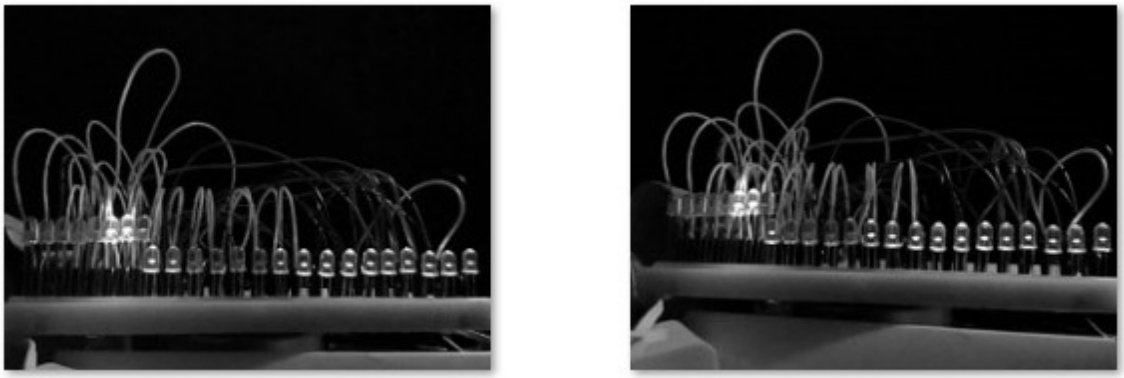


Figura 22: Foto 10 realizada al Reloj desde las cámaras 1 y 2 respectivamente

Como se puede observar en la tabla 5.2 existe un valor mayor que el resto (22 ms.). Algunas de las razones que pueden provocar este desfase entre capturas, pueden tener un origen muy diverso:

- Todas las *SBC* trabajan con un kernel de *Linux* que no es adecuado para aplicaciones en tiempo real. Al trabajar en multitarea, cualquier proceso puede tener prioridad sobre la CPU y causar un retraso en la captura.
- La propia electrónica de las cámaras, al recibir invocaciones de captura de forma rápida y continua.
- La gestión de las interrupciones que hacen los *SBC* que tienen la cámara que están a la espera de la orden de captura.

Pese a que el motivo de esta desincronización es desconocido, se observa que la

desincronización en el disparo entre ambas cámaras (valores anormalmente altos) se produce únicamente de forma esporádica y se ha considerado como un problema menor, ya que la precisión alcanzada cubre nuestras expectativas. Además, dado que la trayectoria del dron es continua, se podría realizar un procedimiento correctivo analizando las distancias entre posiciones consecutivas.

5.5 Análisis del tiempo entre disparos consecutivos

Analizando el tiempo entre disparos consecutivos de la misma cámara a lo largo del tiempo, tomando un total de 62 fotografías y realizando el proceso completo.

Este análisis permite determinar que vamos a disponer una reconstrucción 3D de la posición del dron cada m = 150.7903 mseg. de media. Asumiendo que los datos están distribuidos según una gaussiana podemos determinar que el 99.73% de la diferencia entre dos reconstrucciones consecutivas será inferior a m+3*s = 267.3793 mseg.

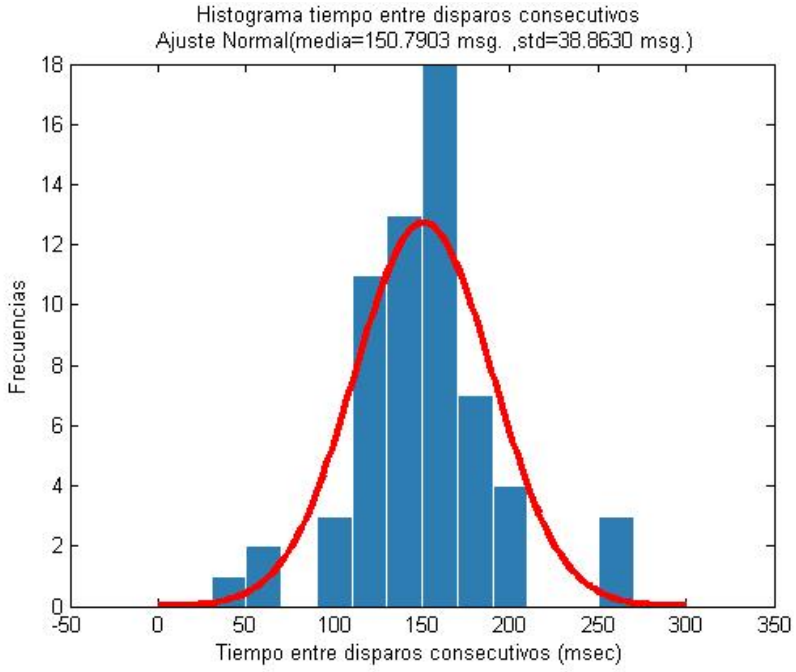


Figura 23: Histograma que muestra el tiempo medio entre disparos

5.6 Referencias bibliográficas incluidas en este capítulo

[1] Richardson, F.H. (1923, October). Importance of Synchronizing Taking and Camera Speeds. Transactions of S.M.P.E.

[2] Litos, G., Zabulis, X., & Triantafyllidis, G. (2006, June). Synchronous image acquisition based on network synchronization. In 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06) (pp. 167-167). IEEE.

[3] Svoboda, T., Hug, H., & Van Gool, L. (2002, September). ViRoom—low cost synchronized multicamera system and its self-calibration. In *Joint Pattern Recognition Symposium* (pp. 515-522). Springer Berlin Heidelberg.

[4] Vibeck A. (2015) Synchronization of a multi camera system [en línea] http://liu.diva-portal.org/smash/get/diva2:822340/FULLTEXT01.pdf

[5] Sánchez del Río, C. (1989) Análisis de errores, Ed. Eudema

6. Visión artificial, calibración de las cámaras y reconstrucción 3D.

6.1 Introducción

De la misma forma que en un sistema de posicionamiento basado en sistemas de radio se ve perjudicado por inconvenientes (estructura del edificio, interferencias con otros dispositivos, factores ambientales, etc), los sistemas de posicionamiento basados en tecnología visual se enfrentan a otros factores como son las distorsiones producidas por las cámaras en las capturas. Esto hace que sea necesario la realización de un procedimiento conocido como calibración de las cámaras.

6.2 Visión Artificial

Antes de describir el proceso realizado para la calibración de las cámaras, resulta necesario explicar algunos conceptos sobre Visión Artificial [1].

6.2.1 El modelo de Pinhole

La imagen obtenida en una cámara, según el modelo Pinhole se define según el siguiente diagrama:

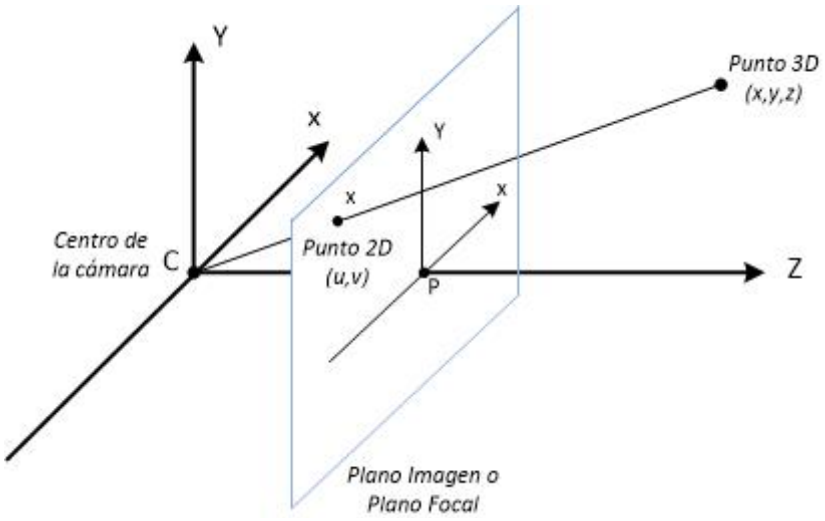


Figura 24: Modelo de Pinhole

El punto *C* es el centro de la cámara. Un punto *X* del mundo real se proyecta en lo que se llama el *plano focal* de la cámara, que está a una distancia *f* del centro de la cámara. A esta distancia *f* se le llama *distancia focal*.

Como convenio, consideraremos lo siguiente:

- En el *plano imagen* (también llamado *plano focal*), el eje x es horizontal y creciente hacia la derecha, el eje y es vertical y creciente hacia arriba, y el eje z es perpendicular y creciente en la dirección del centro de la cámara hacia el plano focal.

- El *eje óptico* es la línea perpendicular al plano focal que pasa por el centro de la cámara C

- El punto *P* es la intersección entre el eje óptico y el plano imagen. En una foto, el punto *P* está cerca del centro de la imagen, pero no suele estar exactamente en el centro.

- Si tomamos como origen del sistema de referencia en 3D *(X,Y,Z)* el centro de la cámara (*C*), y los ejes paralelos a *x*, *y*, y siendo Z el eje óptico, se cumple que:

$$X/Z = x/f$$

$$Y/Z = y/f$$

6.2.2 Coordenadas homogéneas

Las coordenadas homogéneas son un artificio matemático que se utilizan para poder describir la proyección de un punto en el plano imagen de forma lineal.

Para ello, se expanden las coordenadas 2D y 3D con una dimensión adicional, cuyo valor es 1.

Así, en 2D, el punto [x;y] pasaría a ser [x;y;1]. En 3D, el punto [X;Y;Z] pasaría a ser [X;Y;Z;1]

6.2.3 Matriz de cámara

La matriz de cámara es una matriz de dimensiones *3x4* que convierte un punto del

espacio 3D a un punto del espacio 2D usando el *modelo Pinhole*, asumiendo coordenadas homogéneas y que el origen de coordenadas está en el centro de la cámara *C*.

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = [P] \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

Podemos observar que la matriz de cámara depende exclusivamente de la distancia focal f.

Dado un punto (X_c,Y_c,Z_c) del espacio 3D, para obtener su proyección basta ponerlo en coordenadas homogéneas añadiendo un 1, y resolver el sistema anterior definiendo:

$$X_c = X/Z$$

$$Y_c = Y/Z$$

$$Z_c = Z/Z = 1$$

Ello hace que:

$$x = f*Xc = f * X/Z$$

$$y = f*Yc = f * Y/Z$$

Como la solución es única, basta definir la matriz de cámara usando el valor de *f*, y usar coordenadas homogéneas para proyectar un punto (X, Y, Z) en el correspondiente (x,y).

6.2.4 Parámetros extrínsecos de la cámara: traslación y rotación

El centro de la cámara puede no coincidir con el origen de coordenadas del mundo real (3D). La cámara estará desplazada, y rotada en el espacio 3D.

a. Traslación

Para describir un desplazamiento en 3D, basta con definir los desplazamientos a lo largo del eje X, del eje Y, y del eje Z. Por tanto, el vector desplazamiento es un vector columna con los valores:

$$t=[t_x;t_y;t_z]$$

b. Rotación

Para describir la rotación en el espacio 3D se necesitan 3 parámetros (alpha, beta, gamma) y para describir la rotación otros 3 parámetros (*incrx*, *incry*, *incrz*). Estos 6 parámetros se denominan parámetros Extrínsecos de la cámara, y describen su posición y orientación en el espacio 3D.

La rotación R en el espacio 3D se describe mediante la matriz de rotación:

$$R_x(\theta)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\theta) & -\sin(\theta)\\ 0 & \sin(\theta) & \cos(\theta)\end{bmatrix} R_y(\theta)=\begin{bmatrix}\cos(\theta) & 0 & \sin(\theta)\\ 0 & 1 & 0\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix} R_z(\theta)=\begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\\ \sin(\theta) & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix}$$

Así, para rotar un vector en torno al eje X, se aplica:

$$[X';Y';Z'] = Rx * [X;Y;Z]$$

Cualquier rotación en el espacio 3D se puede descomponer en 3 rotaciones, una por cada eje.

$$[X',Y',Z'] = Rx*Ry*Rz*[X;Y;Z] = R * [X;Y;Z]$$

Por tanto R, aunque tiene 9 valores, realmente se puede describir con solo 3, los 3 ángulos de esta descomposición.

c. Posición de la cámara

Cualquier posición de la cámara se puede descomponer en una traslación *t* y un giro *R*, por lo que para calcular transformar un punto del espacio 3D del mundo real al espacio 3D con origen en el centro de la cámara *C*, podemos realizar la siguiente transformación:

$$\begin{bmatrix}X_c\\ Y_c\\ Z_c\\ 1\end{bmatrix}=[R]\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix}+t=\begin{bmatrix}R\,t\\ 0\,1\end{bmatrix}=\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix}=[T]\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix}$$

La matriz T tiene dimensiones 4x4, y se cumple que:

$$R = T(1:3,1:3)$$

$$t = T(1:3,4)$$

La fila cuarta de T es igual a 0, salvo el valor T(4,4) que vale 1.

6.2.5 Conversión de metros a píxeles

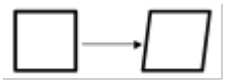
La última transformación necesaria es convertir las unidades del mundo real (x,y, en metros) a las unidades del sensor de la cámara (u,v en píxeles).

Para ello es necesario conocer:

a. El desplazamiento del eje óptico respecto a su lugar teórico (centro de la imagen) medido en píxeles, que se define mediante dos parámetros (c_x, c_y)

b. El número de píxeles por metro en la dirección x e y (k_x, k_y)

c. El término conocido como "*skew factor*" mide el ángulo real que forma el eje x con la perpendicular al eje y, por el que los píxeles no se pueden considerar cuadrados. Recordar que cot(a)=tan(pi/2-a).



Con estos términos, la transformación de metros(x,y) en píxeles(u,v) viene dada por:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_x k_x \cot(\theta) & c_x + c_y \cot(\theta) \\ 0 \; k_y \sin(\theta) & c_y \sin(\theta) \\ 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = [A] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Normalmente el *skew factor* es muy pequeño, y se asume que tiene el valor de 90 grados, en cuyo caso, la ecuación se simplifica:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_x \; 0 \; c_x \\ 0 \; k_y c_y \\ 0 \; 0 \; 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = [A_{simplificada}] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

6.2.6 Parámetros intrínsecos de la cámara

Definen las características ópticas, geométricas y digitales de la cámara:

a. Distancia focal (f): distancia del centro de la cámara al plano imagen de la misma

b. Parámetros necesarios para transformar las coordenadas del plano imagen con las coordenadas en pixeles *(k_x, k_y, c_x, c_y)*

   - *kx y ky*: factores de escala que permite establecer una correspondencia para la distancia entre pixeles de la imagen (expresada en metros y en pixeles)
   - *cx y cy* : coordenadas medidas en pixels del centro de la imagen, definido como la intersección del eje óptico (perpendicular al plano imagen que pasa por el centro de la cámara) con el plano imagen

c. Parámetros que definen la distorsión introducida por las lentes (se tienen en cuenta tanto la distorsión radial como tangencial)

6.2.7 Modelo Pinhole completo

El modelo *Pinhole* completo se puede definir como:

$$(X\,Y\,Z)\underset{\rightarrow}{T}(X_c\,Y_c\,Z_c)\underset{\rightarrow}{P}(x\,y)\underset{\rightarrow}{A}(u\,v)$$

donde T convierte el sistema de coordenadas del mundo real al sistema de coordenadas de la cámara, P convierte de 3D a 2D, y A convierte de metros a pixeles.

De esta forma:

$$[u;v;w] = A * P * T * [X;Y;Z;1]$$

Para obtener las coordenadas en 2D basta con dividir el vector obtenido [u,v,w] para garantizar coordenadas homogéneas

$$u = u/w$$

$$v = v/w$$

Normalmente, A*P se une en una única matriz:

$$K = AP = \begin{bmatrix} k_x k_x \cot(\theta) & c_x + c_y \cot(\theta) \\ 0 \; k_y \sin(\theta) & c_y \sin(\theta) \\ 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} f\,0\,0\,0 \\ 0\,f\,0\,0 \\ 0\,0\,1\,0 \end{bmatrix} = \begin{bmatrix} f_x f_x \cot(\theta) & c_x + c_y \cot(\theta) & 0 \\ 0 \; f_y \sin(\theta) & c_y \sin(\theta) & 0 \\ 0 \quad 0 & 1 & 0 \end{bmatrix}$$

cuyo valor, en el caso de un *skew factor* de 90º es:

$$K = AP = \begin{bmatrix} k_x & 0 & c_x \\ 0 & k_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f\,0\,0\,0 \\ 0\,f\,0\,0 \\ 0\,0\,1\,0 \end{bmatrix} = \begin{bmatrix} f\,k_x & 0 & c_x & 0 \\ 0 & f\,k_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Es frecuente encontrar los valores de la distancia focal como $f_x = f * k_x$, es decir la distancia focal medida en pixels en la dirección x, e igual con $f_y = f * k_y$, en la dirección y.

Asimismo, es frecuente llamar **parámetros intrínsecos** a los 5 términos c_x, c_y, f_x, f_y y *skew factor.*

6.2.8 Matriz de proyección

Si combinamos la matriz de parámetros intrínsecos (K=A*P), y la de parámetros extrínsecos (T), obtenemos la ecuación siguiente, siendo M de dimensiones 3x4 la matriz de proyección definida como:

$$M_{3x4} = A_{3x3} * P_{3x4} * T_{4x4} = K_{3x4} * T_{4x4}$$

$$\begin{bmatrix} x_h \\ y_h \\ w \end{bmatrix} = M \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} m_{12} m_{13} m_{14} \\ m_{21} m_{22} m_{23} m_{24} \\ m_{31} m_{32} m_{33} m_{34} \end{bmatrix} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Por tanto, la matriz de proyección nos permite convertir un punto en el espacio 3D real a un punto en la imagen en pixels.

Para obtener las coordenadas en el espacio 2D es preciso convertir el vector obtenido (x_h,y_h,w) a coordenadas homogéneas, dividiendo por w:

$$u = x_h/w$$

$$v = y_h/w$$

Se puede comprobar que si skew factor=90º, $f_x=f*k_x$, $fy=f*k_y$:

$$t=\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix};[R]=\begin{bmatrix} r_{11} r_{12} r_{13} \\ r_{21} r_{22} r_{23} \\ r_{31} r_{32} r_{33} \end{bmatrix}$$

La operación matricial se reduce a:

$$u=f_x\frac{r_{11}X+r_{12}Y+r_{13}Z+t_x}{r_{31}X+r_{32}Y+r_{33}Z+t_z}+c_x$$

$$v=f_x\frac{r_{21}X+r_{22}Y+r_{23}Z+t_y}{r_{31}X+r_{32}Y+r_{33}Z+t_z}+c_y$$

6.3 Procedimiento de calibración de las cámaras

Un proceso de calibración fundamentalmente consiste en la obtención de los parámetros geométricos y físicos de las cámaras que se usarán [2][3].

Se definen como *parámetros geométricos*:

- las coordenadas del punto central y
- la distancia focal

Se definen como *parámetros físicos* **[7]** :

- la distorsión radial y
- la distorsión tangencial

Considerando que los componentes ópticos de cualquier cámara no son perfectos, la proyección entre la imagen capturada y el objeto real no es lineal y esto afecta de forma negativa a la conversión 2D a 3D y viceversa.

Existen muchos métodos de calibración de una cámara, en este trabajo se empleará una técnica que consiste en realizar una serie de fotografías a un determinado patrón formado por un conjunto de rectángulos (28x40, en nuestro caso) todos del mismo

tamaño y separados por la misma distancia. El algoritmo identifica la posición de los vértices de cada cuadrado para calcular la distorsión de la lente y por último estima los parámetros físicos de la cámara y la distorsión radial de la lente.

Para la implementación de la calibración de las cámaras se usará el módulo para python denominado *OpenCV*. Esta librería, dispone de una variedad de métodos que facilitan la obtención de los parámetros intrínsecos y extrínsecos de las cámaras y también agilizan, por ejemplo, la obtención de los vértices de patrones formado por cuadrados, como los descritos anteriormente.

A continuación, se describen los pasos seguidos para realizar la calibración de las cámaras.

1. El primer paso consistió en la creación e impresión de un damero, es decir, un patrón formado por cuadrados del mismo tamaño. La impresión se realizó en una hoja de tamaño A3 para facilitar la realización de fotografías. El damero utilizado tenía 40 rectángulos de alto y 28 rectángulos de ancho. En nuestro caso, el tamaño fijado para cada rectángulo fue de 9 mm.

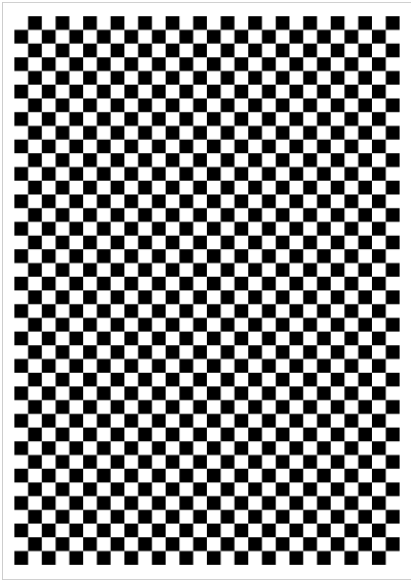


Figura 25: Aspecto del damero utilizado para la calibración de las cámaras

2. El segundo paso consistió en el pegado del damero a un cristal, ya que para la

realización de las fotografías resulta importante que el papel no tuviera ninguna deformación. Para el pegado se utilizó un pegamento especial en spray para repartirlo uniformemente sobre la superficie del cristal y evitar la formación de grumos.

Figura 26: Aspecto del damero pegado al cristal

3. El tercer paso consistió en la realización de las fotos al damero desde distintas posiciones con cada cámara. El proceso que se siguió fue el mismo con cada cámara de forma independiente. La cámara se montó sobre un trípode para evitar movimientos a la hora de la realización de las capturas. También, se tuvo en cuenta la iluminación para que ésta, estuviera homogéneamente distribuida sobre toda la superficie del damero. Además, se implementó un pequeño script en python que permitió mostrar en una pantalla la previsualización de la captura, con el objetivo de realizar el encuadre cómodamente, y finalmente a través de la pulsación de una tecla realizar la fotografía.

   En la imagen siguiente, se puede observar el escenario y todos los dispositivos utilizados (lámparas, trípodes, raspberry pi, pantalla y damero) para la realización de las series de fotos al damero.

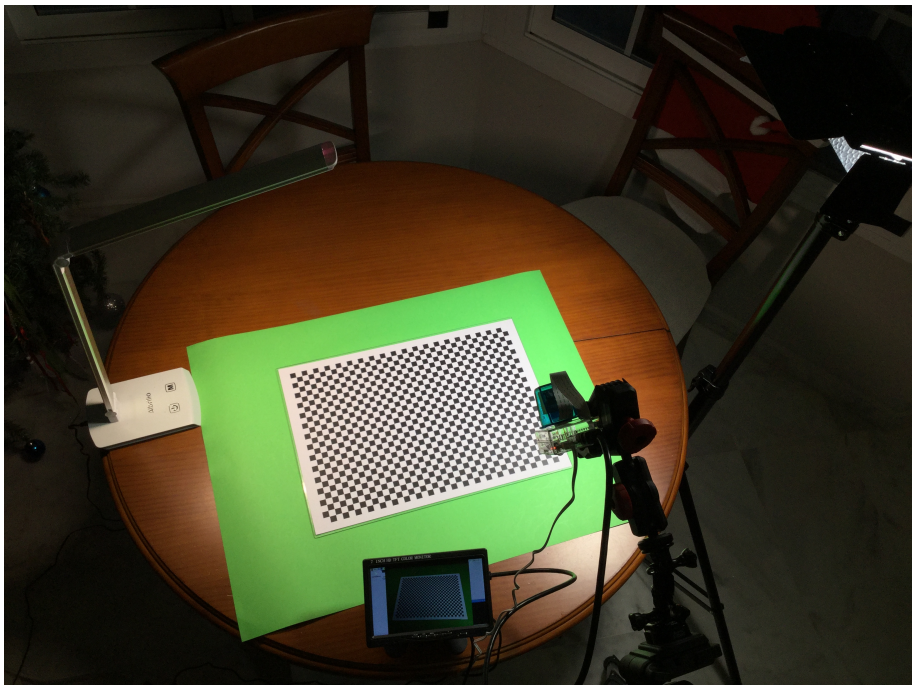


Figura 27: Escenario usado para la realización de las fotografías del damero.

En las siguientes imágenes, se pueden ver algunas de las capturas realizadas al damero. Para la realización del proceso de calibración, se utilizaron 30 capturas con cada cámara de diferentes perspectivas del damero. Todas las capturas se realizaron con un encuadre en el que el patrón se veía completamente y tratando de que en cada captura se cubriera un área diferente de la pantalla para conseguir que no quedara ningún área de la pantalla sin calibrar.

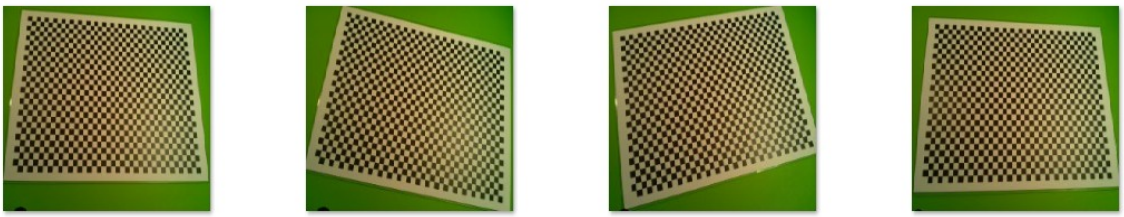


Figura 28: Ejemplos de fotografías realizadas al damero

4. El siguiente paso consiste en el cálculo de los parámetros intrínsecos de cada cámara. Para ello utilizaremos la función *calibrateCamera()* de la librería *OpenCV*.

   Los parámetros de entrada a esta función son:

1) Un conjunto de coordenadas 3D de un damero *(objpoints)*: contiene las coordenadas del damero en el espacio 3D, y se representa mediante una matriz de dimensiones Nx3, donde N es el número de esquinas consideradas del damero.

2) Conjunto de coordenadas en el espacio 2D de cada cámara de cada punto del damero *(imgpoints)*.

3) Tamaño de la imagen *(imgSize)*: vector de tamaño 1x2 con el tamaño, medido en pixels, de cada imagen.

4) Un vector de flags y otro de condiciones de parada, que dejaremos con sus valores por defecto

Los parámetros de salida de esta función son:

1) Matriz de parámetros intrínsecos*(mtx)* :

$$Matriz\,de\,cámara = \begin{bmatrix} k_x & 0 & c_x \\ 0 & k_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Donde:

kx,ky: factores de escala

cx,xy: coordenadas en pixels del centro de la imagen

2) Vector de coeficientes de distorsión*(dist)*: define la distorsión de la lente.

3) Matrices de rotación de cada cámara *(rvecs):* define la posición de la cámara.

4) Vectores de traslación de cada cámara *(tvecs)*. Define la posición de la cámara.

Un ejemplo de llamada es:

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, imgSize, None, None)

5. Para el cálculo de los parámetros extrínsecos de una cámara en cualquier posición y sin tener que usar el damero, utilizaremos la función *solvePnP()* de la librería *OpenCV*.

   Los parámetros de entrada a esta función son:

   1) Un conjunto de coordenadas 3D *(worldPoints)*: contiene las coordenadas de varios puntos del espacio 3D con coordenadas conocidas.

   2) Conjunto de coordenadas en el espacio 2D de cada cámara de los puntos conocidos anteriores *(photoPoints)*.

   3) Matriz de parámetros intrínsecos*(mtx)*: ya descritos anteriormente

   4) Vector de coeficientes de distorsión(*dist*): define la distorsión de la lente.

   Los parámetros de salida de esta función son:

   1) Un vector de flags y otro de condiciones de parada, que dejaremos con sus valores por defecto

   2) Matrices de rotación de cada cámara *(rvecs):* define la posición de la cámara.

   3) Vectores de traslación de cada cámara *(tvecs)*. Define la posición de la cámara.

   Un ejemplo de llamada es:

   ret, rvec, tvec = cv2.solvePnP(worldPoints, photoPoints, mtx, dist)

6.4 Reconstrucción 3D

Este apartado se encuentra dividido en dos partes: una primera parte dedicada al reconocimiento del drone en una captura de foto y una segunda parte dedicada a explicar el proceso de cálculo de una posición 3D, a partir de sus proyecciones en dos cámaras.

6.4.1 Reconocimiento del drone.

En primer lugar, se expondrán los fundamentos teóricos empleados para la identificación del drone y a continuación, se describirá de una forma gráfica el experimento realizado.

Aspectos teóricos.

El objetivo es conocer la posición del dron en las distintas imágenes obtenidas con cada cámara.

Para realizar la reconstrucción se utilizarán procedimientos geométricos que suponen el dron en una posición puntual en el espacio (x,y,z)

Por tanto, resulta necesario que el aparato tenga una referencia puntual para facilitar la identificación del dron en el espacio 3D. Para ello, se ha utilizado un elemento lo más pequeño posible y que fuese fácil de identificar en las imágenes, ya que resulta necesario que pueda ser fotografiado desde cualquier perspectiva. Se ha utilizado un Led cuyas dos principales características son:

- Ofrecer una potente fuente de luz, considerando su reducido tamaño
- Tiene muy bajo consumo de energía, lo que permite añadirle una fuente de energía también de reducido tamaño y peso para no afectar al vuelo.

Los leds se fabrican de distintos colores, tras diversas pruebas se decide utilizar el led de color blanco por sus mejores resultados y también por la simplificación en la implementación de la identificación. Se ha comprobado que este algoritmo no sobrecarga la CPU de la SBC a la que se encuentra conectada cada cámara.

La identificación del punto de máxima luminosidad en una fotografía es un proceso muy sencillo, ya que consiste en buscar el valor máximo en la matriz RGB capturada por la cámara.

El proceso de identificación del punto de máxima luminosidad implementado es el siguiente:

1. Realización de la captura fotográfica
2. Conversión de la imagen a Escala de grises.
3. Aplicación de un filtro *Gaussiano* a la imagen con el objetivo de producir un suavizado uniforme a la imagen.
4. Localización de las coordenadas *x,y* del punto de mayor luminosidad.
5. El resultado es la posición del dron (x,y) en el plano imagen (2D) de cada cámara.

Aspectos experimentales.

A continuación, se describe de forma gráfica el experimento realizado para el reconocimiento del drone.

Tal y cómo se indicó en el apartado anterior, para facilitar la identificación del drone en una captura fotográfica, se añadió un led junto a una pequeña batería en la parte superior del drone.

En la imagen siguiente, se puede ver como queda el led con su respectiva batería sobre la carcasa del drone. Para aumentar la luminosidad del led se comprobó que rascando la superficie del led y limando la parte superior del led mejora la difusión de la luminosidad del led, lo que facilita el reconocimiento.

Figura 29: Drone con el led encendido en su parte superior

Se ha tratado de buscar un sistema simple y a la vez efectivo, que no afecte al vuelo del drone.

Para el cálculo de la posición 3D del aparato, cada *SBC* tomará fotografías a través de la cámara que tiene conectada y a partir de cada fotografía irá identificando el drone.

En las dos imágenes siguientes, se puede comprobar la identificación del drone, a partir del led. En este caso, para comprobar que se ha identificado correctamente el led, desde el código implementado, se ha colocado un círculo azul sobre el punto de máxima luminosidad identificado.

Figura 30: Identificación del drone en las fotos realizadas por las dos cámaras

Una vez identificado el drone, el sistema devuelve la posición x,y en píxeles dentro de

la correspondiente foto.

A partir de las posiciones x,y identificadas en cada foto y después de ser transferidas a un PC, éste realizará la reconstrucción y obtendrá la posición 3D.

En el apartado siguiente, se describe el proceso de reconstrucción.

6.4.2 Reconstrucción 3D.

En primer lugar, se expondrán los fundamentos teóricos empleados para la reconstrucción 3D [5] y a continuación, se describirá de una forma gráfica el experimento realizado.

Aspectos teóricos.

En este apartado se describirá el proceso de cálculo de reconstrucción de una posición 3D a partir de dos imágenes capturadas por dos cámaras diferentes en el mismo instante.

Supongamos que la proyección de un punto 3D en una cámara con matriz de proyección P1 sea (x1,y1) y que la proyección del mismo punto 3D en otra cámara con matriz de proyección P2 sea (x2,y2), compondremos la matriz

auxiliar A de la siguiente forma:

```
A =   [x1*P1(3,:) - P1(1,:)
       y1*P1(3,:) - P1(2,:)
       x2*P2(3,:) - P2(1,:)
       y2*P2(3,:) - P2(2,:)];
```

donde

P1(1,:) indica la primera fila de P1
P1(2,:) indica la segunda fila de P1
P1(3,:) indica la tercera fila de P1
P2(1,:) indica la primera fila de P2
P2(2,:) indica la segunda fila de P2
P2(3,:) indica la tercera fila de P2

Sea (S,V,D) la descomposición en valores singulares de A [4]. La última columna de D

contiene el punto deseado, pero al estar expresado en coordenadas homogéneas su última componente ha de ser 1, por lo que hay que dividir dicha columna por el último elemento, cuyo valor pasa a ser 1.

En las tres primeras posiciones se encuentra a reconstrucción 3D del punto de interés.

Aspectos experimentales.

A continuación, se describe de forma gráfica el experimento realizado para la reconstrucción 3D a partir de dos capturas fotográficas.

En la imagen siguiente, se puede comprobar la ubicación de las cámaras que se utilizarán:

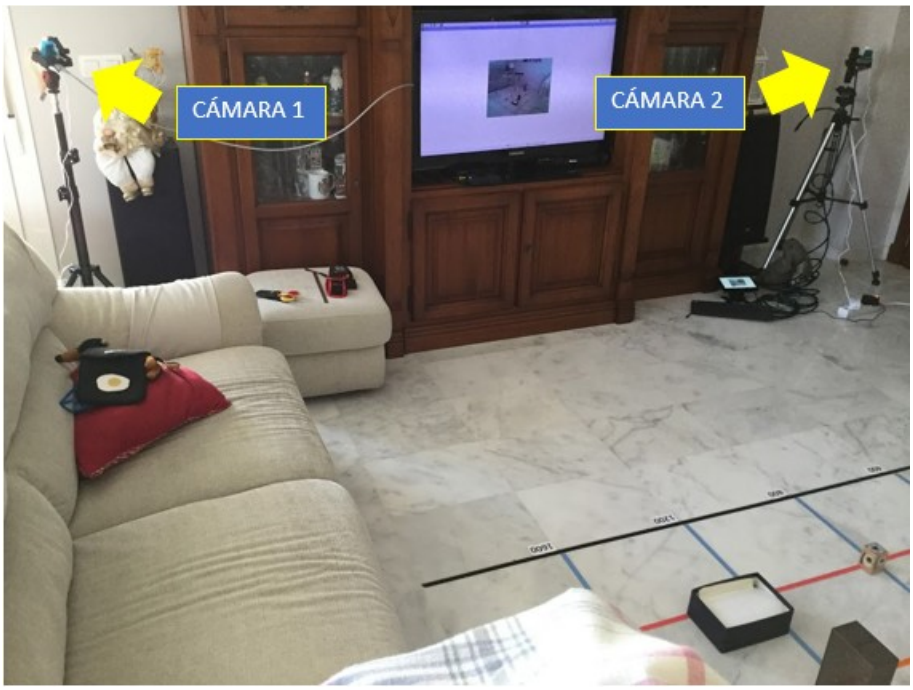


Figura 31: Ubicación de las cámaras para la realización de la reconstrucción 3D

En las imágenes siguientes, se pueden observar cómo cada cámara está pegada a una Raspberry y mediante un soporte unida a un trípode.

Figura 32: Aspecto de la cámara 1 sobre trípode

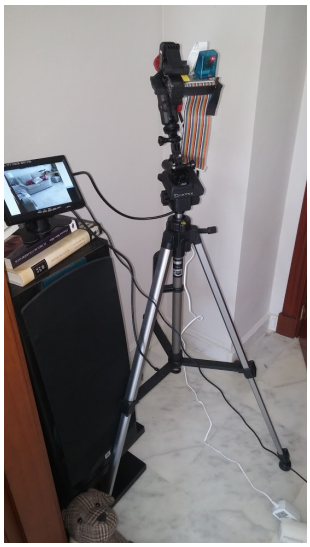


Figura 33: Aspecto de la cámara 2 sobre trípode

Figura 34: Vista de cerca del soporte, cámara y Raspberry

En la imagen siguiente, se puede observar el sistema de referencia creado para poder realizar la reconstrucción 3D. Se puede apreciar los ejes de coordenadas, así cómo su correspondiente origen.

Figura 35: Sistema de referencia real creado para la reconstrucción 3D

En la imagen anterior, se puede apreciar cómo hay situados una serie de objetos que servirán para el cálculo de los *parámetros extrínsecos*, así cómo para la comprobación de que la reconstrucción 3D se ha realizado correctamente.

Para el cálculo de los *parámetros extrínsecos* se usaron 10 puntos y en concreto fueron los que se muestran en la tabla siguiente:

Puntos utilizados para el cálculo de los parámetros extrínsecos		
x	**y**	**z**
0	300	67
800	300	67
400	900	67
400	1200	135
155	1966	1295
1355	1966	1295
800	0	0
0	900	0
1600	1200	0
400	600	0

Tabla 3: Puntos utilizados par el cálculo de los parámetros extrínsecos

A continuación se procederá a realizar la reconstrucción 3D de un punto conocido para comprobar el funcionamiento del algoritmo implementado.

El punto al que señala la flecha amarilla, identifica el vértice superior izquierdo del objeto cuyas coordenadas reales son *(1200, 600, 190)*.

En la imagen siguiente, se puede comprobar las coordenadas reales 3D del punto que se pretender reconstruir

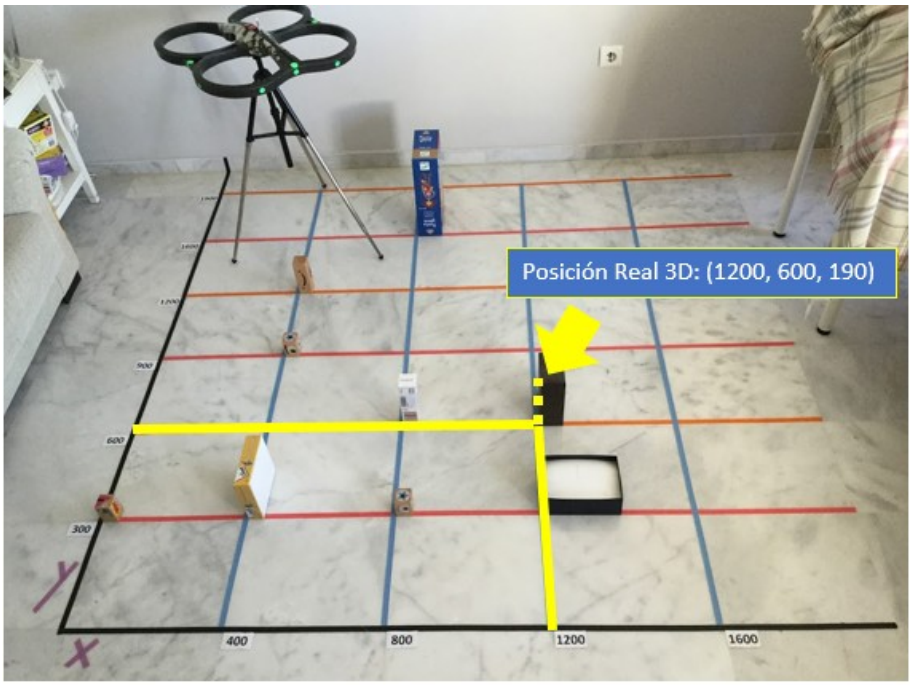


Figura 36: Representación gráfica de la posición real del punto que se pretende calcular

Una vez que contamos con los parámetros intrínsecos y extrínsecos de ambas cámaras, se realizará una captura fotográfica con cada cámara.

A continuación, se identificará el objeto (del cual sabemos su posición real) en la foto capturada por la Cámara 1, y se procede a determinar la posición *x* e *y* en píxeles en la que se encuentra el punto conocido.

En la imagen siguiente, se puede comprobar cómo el vértice superior izquierdo de la caja se encuentra en la posición *(322, 294)*.

Figura 37: Posición en píxeles del punto que se desea calcular en la fotografía capturada por la cámara 1

A continuación, se procederá a identificar el objeto (del cual sabemos su posición real) en la foto capturada por la Cámara 2, y de la misma forma que se ha realizado anteriormente, se procede a determinar la posición *x* e *y* en píxeles en la que se encuentra el punto conocido.

En la imagen siguiente, se puede comprobar cómo en este caso el vértice superior izquierdo de la caja se encuentra en la posición *(410, 268)*.

Figura 38: Posición en píxeles del punto que se desea calcular en la fotografía capturada por la cámara 2

Una vez aplicado el algoritmo implementado, el resultado obtenido es el siguiente:

$$x = 1196.31165353000,\ y = 600.31718330700,\ z = 179.25004214100$$

En la tabla siguiente, se muestra la diferencia entre las coordenadas reales y las calculadas y la distancia euclídea entre ambas posiciones. Se puede observar cómo la distancia entre ambos puntos es de aproximadamente de 11 milímetros.

Resultado reconstrucción 3D													
Xcam1	**Ycam1**	**Xcam2**	**Ycam2**	**Xreal**	**Yreal**	**Zreal**	***Xcal***	***Ycal***	***Zcal***	***DifX***	***DifY***	***DifZ***	***dist3D***
322	294	410	268	1200	600	190	*1196.31*	*600.31*	*179.25*	*-3.68*	*0.31*	*-10.74*	**11.36**

Tabla 4: Resultado reconstrucción 3D del punto seleccionado

Para confirmar que el procedimiento de reconstrucción es correcto, se tomaron 19

puntos más, con posiciones x,y,z conocidas (localizadas en nuestro sistema de referencia) y se repitió el mismo procedimiento descrito anteriormente, calculando la distancia euclídea entre la posición 3D real y la calculada para cada uno de ellos.

En la tabla siguiente, se muestran los resultados.

Xcam1	*Ycam1*	*Xcam2*	*Ycam2*	*Xreal*	*Yreal*	*Zreal*	**Xcalc**	**Ycalc**	**Zcalc**	**DifX**	**DifY**	**DifZ**	***dist3D***
149	246	295	337	400	300	162	399,25	297,85	161,70	-0,74	-2,14	-0,29	***2,28***
262	337	457	312	1200	300	78	1197,00	301,57	71,10	-2,99	1,57	-6,89	***7,67***
260	260	339	283	800	600	190	804,47	602,87	184,92	4,47	2,87	-5,07	***7,35***
322	**294**	**410**	**268**	1200	600	190	1196,31	600,31	179,25	-3,68	0,31	-10,74	***11,36***
411	290	334	259	1200	1200	67	1203,16	1208,17	59,86	3,16	8,17	-7,13	***11,30***
395	206	240	209	800	1500	332	807,81	1510,78	325,38	7,81	10,78	-6,61	***14,87***
43	261	229	462	0	0	0	-8,30	-6,05	-0,11	-8,30	-6,05	-0,11	***10,27***
83	293	345	418	400	0	0	396,84	-4,70	0,08	-3,15	-4,70	0,08	***5,66***
190	380	514	356	1200	0	0	1196,27	-5,61	-4,04	-3,72	-5,61	-4,04	***7,86***
160	242	155	376	0	600	0	-3,88	601,17	-3,27	-3,88	1,17	-3,27	***5,20***
250	227	107	320	0	1200	0	-0,58	1205,62	-4,14	-0,58	5,62	-4,14	***7,01***
336	406	520	308	1600	300	0	1598,55	296,56	-0,09	-1,44	-3,43	-0,09	***3,72***
304	285	300	303	800	900	0	798,18	902,42	-2,88	-1,81	2,42	-2,88	***4,18***
365	318	368	286	1200	900	0	1197,12	901,22	-3,05	-2,87	1,22	-3,05	***4,37***
347	273	268	284	800	1200	0	799,41	1206,89	-1,80	-0,58	6,89	-1,80	***7,15***
444	289	305	256	1200	1500	0	1204,89	1499,25	-0,06	4,89	-0,74	-0,06	***4,95***
516	320	361	244	1600	1500	0	1610,66	1503,05	2,81	10,66	3,05	2,81	***11,44***
366	233	151	266	400	1800	0	407,01	1800,79	0,98	7,01	0,79	0,98	***7,13***
476	278	278	243	1200	1800	0	1205,51	1805,81	-0,38	5,51	5,81	-0,38	***8,02***
545	306	332	233	1600	1800	0	1608,69	1807,68	0,48	8,69	7,68	0,48	***11,61***

Tabla 5: Resultados de reconstrucción de 20 puntos

Se puede comprobar como la precisión de la reconstrucción de los 20 puntos conocidos es inferior a los 12 mm, llegando a ser en en el mejor de los casos inferior a los 3 mm.

6.5. Análisis de errores.

En este apartado se calcularán los errores en las direcciones X, Y, Z y en la distancia entre el punto x,y,z real y el calculado. Por último, se obtendrá el error total.

6.5.1 El error en la dirección X.

En la figura 6.12 se muestra el histograma del error de posicionamiento en la dirección X para los 20 puntos reconstruidos. Se puede observar que sus valores se encuentran en el intervalo [-12,15] medido en milímetros.

Además, suponiendo una distribución normal, la media es de 0.9221 mm. y la desviación estándar es 5.1932 mm., por lo que, si la distribución fuese normal, el 99.87% de los errores se encontrarán en el intervalo [m-3*s,m+3*s] = [0.9921-3*5.1932,0.9921+3*5.1932] = [-14.5875, 16.5717]

Una vez analizados los errores obtenidos, podemos concluir que el error en el posicionamiento en la coordenada x es inferior a 1.7 cm.

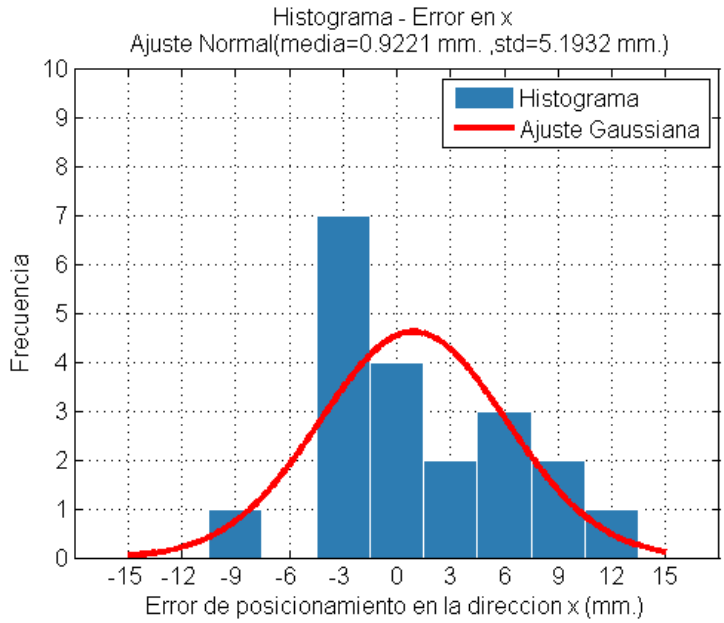


Figura 39: Histograma del error de posicionamiento en la dirección X

6.5.2 El error en la dirección Y.

En la figura 6.13 se muestra el histograma del error de posicionamiento en la dirección Y para los 20 puntos reconstruidos. Se puede observar que sus valores se encuentran en el intervalo [-9,15] medido en milímetros.

Además, suponiendo una distribución normal, la media es de 1.7868 mm. y la desviación estándar es 4.7477 mm., por lo que, si la distribución fuese normal, el 99.87% de los errores se encontrarán en el intervalo [m-3*s,m+3*s] = [1.7868-3*4.7477,1.7868+3*4.7477] = [-12.4563, 16.0299]

Una vez analizados los errores obtenidos, podemos concluir que el error en el posicionamiento en la coordenada y es inferior a 1.7 cm.

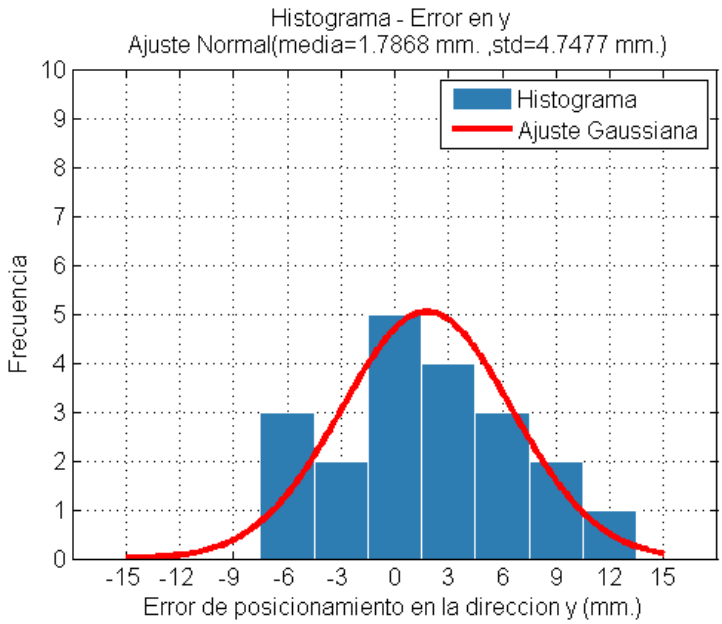


Figura 40: Histograma del error de posicionamiento en la dirección Y

6.5.3 El error en la dirección Z.

En la figura 6.14 se muestra el histograma del error de posicionamiento en la dirección Y para los 20 puntos reconstruidos. Se puede observar que sus valores se encuentran en el intervalo [-15,6] medido en milímetros.

Además, suponiendo una distribución normal, la media es de 2.6139 mm. y la desviación estándar es 3.4156 mm., por lo que, si la distribución fuese normal, el 99.87% de los errores se encontrarán en el intervalo [m-3*s,m+3*s] = [2.6139-3*3.4156,2.6139+3*3.4156] = [-7.6329, 12.8607]

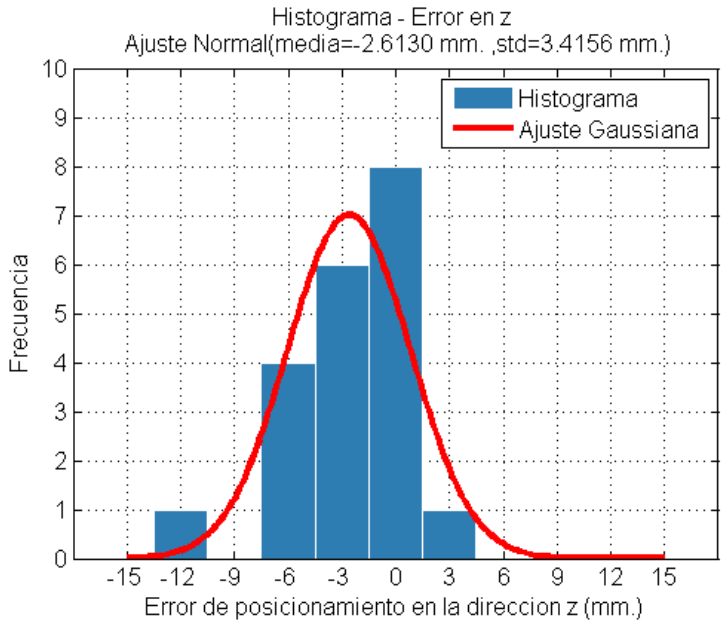


Figura 41: Histograma del error de posicionamiento en la dirección Z

Una vez analizados los errores obtenidos, podemos concluir que el error en el posicionamiento en la coordenada z es inferior a 1.3 cm.

6.5.4 El error total.

En la figura 6.15 se muestra el histograma del error total para los 20 puntos reconstruidos. Se puede observar que sus valores se encuentran en el intervalo [0,16] medido en milímetros.

El ajuste de los errores en la distancia entre el punto x,y,z real y el calculado a una curva queda fuera del ámbito del proyecto debido a su complejidad, al tratarse de la raíz cuadrada de la suma de tres distribuciones Gaussianas de diferentes medias y varianzas [6].

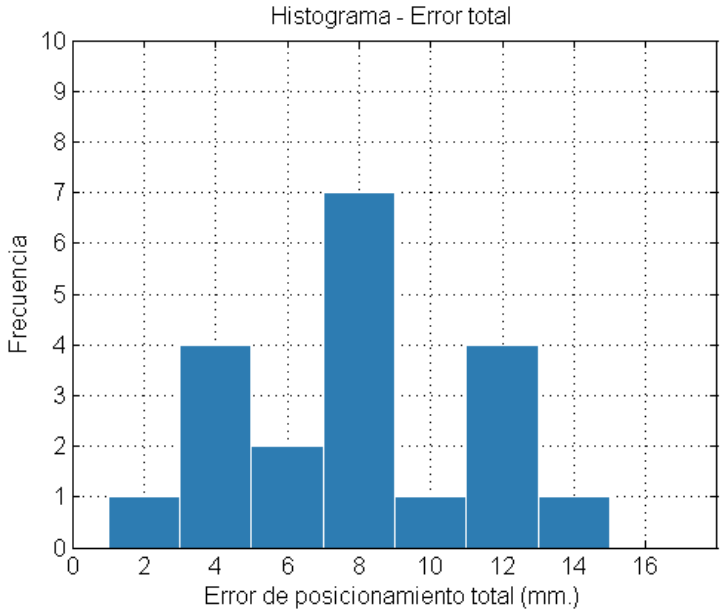


Figura 42: Histograma del error total

6.6 Referencias bibliográficas incluidas en este capítulo

VISIÓN ARTIFICIAL - EN GENERAL
La referencia fundamental para Visión Artificial es:
[1] Hartley, R.~I. and Zisserman, A. (2004) Multiple View Geometry in Computer Vision. Second ed. 2004: Cambridge University Press.

[2] Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE *Transactions on pattern analysis and machine intelligence, 22*(11), 1330-1334.

[3] Zhang, Z (2004) Camera Calibration, Chapter 2, pages 4-43, in G. Medioni and S.B. Kang, eds., Emergin Topics in Computer Vision, Prentice Hall Professional Technical Reference.

[4] Abdel-Aziz, Karara., Direct linear transformation into object space coordinates in closerange photogrametry. In Proc. Symp. Close-Range Photogrametry, 1971: p. 1-18.

[5] Bardsley, D., & Li, B. 3D Reconstruction Using the Direct Linear Transform with a Gabor Wavelet Based Correspondence Measure Technical Report.

[6] Bausch J. (2013) On the efficient calculation of a linear combination of chi-square random variables with an application in counting string vacua. Journal of Physics A: Mathematical and Theoretical, Volume 46, Number 50, pp. 505202(1-23)

[7] Weng J., Cohen P., Herniou M. (1992) Camera Calibration with Distortion Models and Accuracy Evaluation. IEEE Ttransactions on pattern analysis and machine intelligence. Volume 14, Number 14

7. Conclusiones.

Se puede afirmar que se ha logrado con éxito abordar el posicionamiento de un dron en un espacio cerrado y utilizando únicamente tecnología visual, en nuestro caso, basado en dos cámaras conectadas cada una a un ordenador de placa única.

También hay que resaltar que la solución aportada está basada completamente en Software Libre.

Para la realización de este trabajo se ha necesitado estudiar en profundidad los siguientes aspectos:

- Proceso de calibración de una cámara.
- Visión Artificial.
- Sincronización de capturas fotográficas.
- Reconstrucción 3D.
- Análisis de errores.

También fue necesario crear un dispositivo óptico de medición del tiempo con precisión suficiente (1 milésima de segundo) para poder medir el grado de sincronización en la captura simultánea de imágenes desde dos cámaras diferentes.

Para lograr el posicionamiento de un dron en movimiento en un espacio cerrado, se realizó un proceso de calibrado de las cámaras en el que se determinaron las características ópticas, geométricas y digitales de cada cámara.

Además, para la realización del proceso de reconstrucción 3D del dron, se le añadió un led encendido en su parte superior y se implementó un sistema de reconocimiento basado en la identificación del punto de máxima luminosidad. La posición 3D se obtuvo a partir de dos imágenes capturadas con cámaras diferentes en el mismo instante.

Para comprobar que el proceso de reconstrucción fue correcto, se creó un sistema de referencia propio y se tomaron 20 puntos con posiciones (x,y,z) conocidas y se calculo la distancia euclídea entre la posición 3D real y la calculada.

Una característica importante de esta solución es que se han usado dispositivos de bajo coste (SBC, cámaras, cableado, leds, etc) comprobándose que han sido suficiente para alcanzar el posicionamiento de un dron en un espacio interior con un elevado nivel de

precisión.

Por último, mencionar que tras la implementación de esta solución tecnológica se ha realizado un tratamiento estadístico de los errores cometidos, cuyos resultados permiten concluir que el error de posicionamiento en la coordenada X e Y fue inferior a 1.7 cm y en el eje Z inferior a 1.3 cm. Con estos datos, el error de posicionamiento global (distancia euclidea entre la posición real y la posición calculada) para un elemento móvil (dron) es inferior a 16 mm.

8. Recomendaciones para trabajos futuros.

Durante la realización de este trabajo han surgido nuevas ideas y opciones que añadir, además de mejorar las realizadas. Algunas de ellas se indican a continuación:

- Extender la solución a más de dos cámaras y en un espacio interior de mayores dimensiones.

- Desarrollo de una solución tecnológica bajo un sistema operativo de tiempo real.

- Posibilidad de reconocer varios drones usando leds de colores diferentes.

- Aunque el programa permite mostrar la posición 3D del dron en tiempo real por pantalla en modo texto (y en modo gráfico en postproceso). Una mejora podría consistir en extender la representación gráfica del aparato en tiempo real.

- La solución presentada en este trabajo utiliza un sistema basado en los interfaces series para la interconexión entre las *SBC* con cámara y el *PC*, esto supone una limitación a la hora de conectar un elevado número de cámaras. Una mejora consistiría en utilizar otro interfaz, como por ejemplo es *I2C,* el cual permitiría al SBC Maestro gestionar un número más elevado de cámaras conectadas al tratarse de un bus diseñado para interactuar en modo maestro-esclavo, incluso añadiría la posibilidad de trabajar en modo multimaestro.

- El equipamiento utilizado no deja de ser un equipamiento de bajo coste y sin prestaciones profesionales. Una posible mejora podría consistir en utilizar cámaras con mayor resolución y capaz de manejar un mayor número de fotogramas por segundo.

- El dispositivo óptico creado para la medición del tiempo actualmente alcanza

una precisión de 1 milésima de segundo, una posible mejora podría consistir en mejorarlo para alcanzar precisiones inferiores a la milésima de segundo.

- Una última recomendación, consistiría en aprovechar esta solución tecnológica para desplazar un dron dentro de un espacio cerrado de forma autónoma.

Anexo I. Hardware Empleado.

Para el desarrollo de este proyecto se ha empleado el siguiente equipamiento hardware:

Ordenador de placa única (SBC)

Raspberry Pi es un pequeño ordenador de bajo coste, abierto en hardware y software. Para la elaboración de este trabajo se han utilizado cuatro. En la figura siguiente, se puede ver el aspecto del dispositivo.

Figura 43: Raspberry Pi 3

Sus principales características son:

- Procesador: A 1.2GHz quad-core ARMv8 CPU 64-bit
- Interfaz Bluetooth 4.1
- Memoria RAM: 1GB RAM
- Puertos USB: 4
- Puerto GPIO (General Purpose Input/Output) con 40 pins
- Conexión HDMI
- Interfaz de red Ethernet y 802.11n Wireless LAN
- Conector de audio
- Interfaz para cámara(CSI)
- Interfaz para pantalla (DSI)
- Slot de tarjetas Micro SD.
- Procesador MultimediaVideoCore IV 3D.
- Tamaño: 85.6 x 56.5 mm
- Peso: 45gr

En la imagen siguiente se puede apreciar el aspecto del Puerto *GPIO* que incluye cada

SBC:

Figura 44: Interfaz GPIO Raspberry Pi 3

Cámara

Se han usado dos cámaras *Picamera*. Se trata de una placa de cámara diseñada para las *SBC Raspberrys* que se conectan directamente a su conector *CSI*. La placa cuenta con un sensor *Omnivision 5647* de 5MP (2592 × 1944 píxeles) en un módulo de enfoque fijo y es capaz de capturar imágenes con resolución de 5MP o grabar vídeo HD de 1080p a 30fps. Este tipo de cámaras se conectan a la *SBC* mediante un cable plano de 15 pines a la interfaz *CSI* de la Raspberry Pi. En la imagen siguiente, se puede comprobar el aspecto de la cámara y del cable plano que la une a la *SBC*, a través de la interfaz *CSI*.

Figura 46: Pi camera

Las principales características de este tipo de cámaras son:

- Módulo de cámara Omnivision 5647 de 5MP
- Resolución de captura de foto: de 2592 x 1944
- Admite grabación de vídeo: 1080p a 30fps, 720p a 60fps y 640x480p 60/90
- Interfaz serie para conexión directa a la placa Raspberry Pi
- Compatibilidad con todos los modelos de Raspberry
- Dimensiones: 20x 25x 9 mm
- Peso: 3 g

Ordenador portátil

Para realizar la reconstrucción 3D se ha usado un portátil con las siguientes características:

- Procesador: I5-5200U a 2.2GHz turbo 2.7GHz, 3 Mb cache, 2 núcleos
- Interfaz de red 802.11n Wireless LAN y Ethernet
- Interfaz Bluetooth 4.1
- Memoria RAM: 8GB RAM
- Puertos USB: 4
- Conexión: HDMI y VGA
- Conector de audio
- Lector Tarjetas SD
- Dimensiones: 32x 22x 2 mm
- Peso: 2100 g

Unidad Óptica para la medición del tiempo

Se ha fabricado un dispositivo óptico de medición del tiempo, basado en un ordenador de placa única (*SBC*) y un conjunto de leds (*light emitting diode*). En nuestro caso, 24 leds han sido suficientes para alcanzar una precisión de 1 milisegundo.

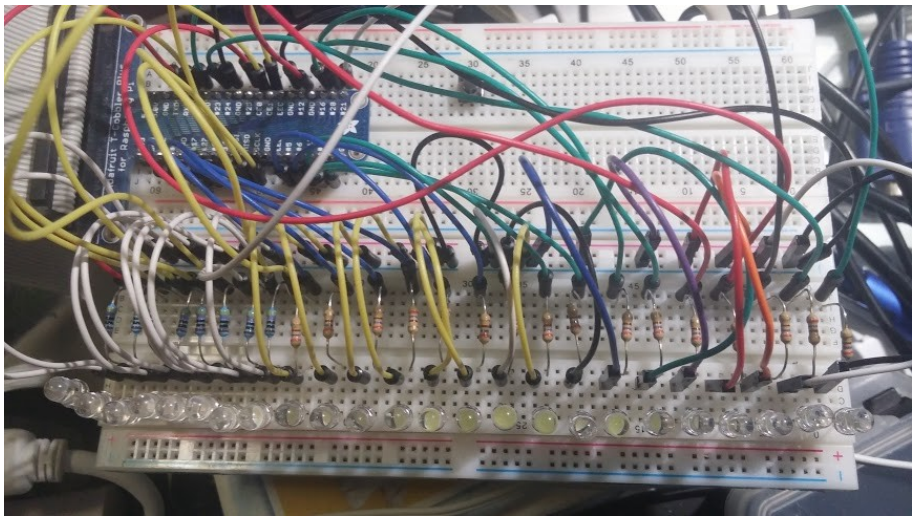


Figura 47: Aspecto del dispositivo óptico creado para la medición del tiempo

Los leds se dividen en dos grupos: uno con 8 leds para el contador binario de ciclos de reloj y otro de 16 leds para definir un ciclo.

Cableado de interconexión

Para la interconexión de los dispositivos implicados en este trabajo se ha fabricado el cableado necesario a partir de cables de pares trenzados.

Se pueden distinguir dos tipos de conexionados:

- Por un lado, el cableado que relaciona la SBC-Maestra con las SBC-Esclavas
- Por otro lado, el cableado que relaciona el PC con las SBC-Esclavas.

En la figura 5, se puede comprobar cómo están interconectados todos los dispositivos indicados anteriormente.

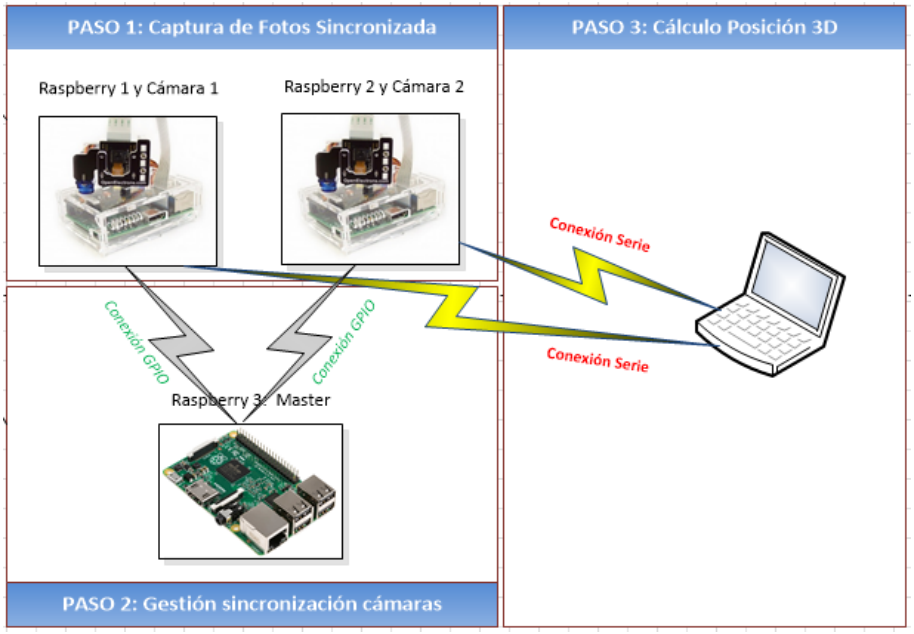


Figura 48: Esquema básico de conexión de los dispositivos implicados en este trabajo

Dron

El dron utilizado para la realización de este trabajo ha sido *Parrot AR. Drone 2.0.* En la imagen siguiente se puede comprobar la apariencia del aparato.

Figura 49: Parrot AR. Drone 2.0

El dispositivo cuenta con las siguientes especificaciones técnicas:

- Procesador: Procesador 1 GHz 32 bits ARM Cortex A8 con DSP vídeo 800 MHz TMS320DMC64x
- Sistema operativo: Linux 2.6.32
- RAM: DDR2 1 GB a 200 MHz
- USB: USB 2.0 de alta velocidad para las extensiones
- Wi-Fi: Wi-Fi b g n
- Giroscopio: 3 ejes, precisión de 2000°/segundo
- Acelerómetro: 3 ejes, precisión de ± 50 mg
- Magnetómetro: 3 ejes, precisión de 6°
- Sensor de presión: Precisión de ± 10 Pa
- Sensores de ultrasonidos para medir la altitud: Medición de la altitud
- Cámara vertical: QVGA 60 FPS para medir la velocidad en vuelo
- 4 motores sin escobillas: 14,5 vatios y 28 500 rpm
- Tamaño con carcasa: 451 x 517 mm
- Peso con carcasa: 420 g.

Anexo II. Software Empleado.

A continuación, se describirá el software empleado para la elaboración de este trabajo:

Lenguaje de programación.

El lenguaje de programación empleado para la implementación de este proyecto ha sido *Python*. Se trata de un lenguaje de programación que tiene las siguientes características:

- Lenguaje de Programación Orientado a Objeto.
- Lenguaje de propósito general.
- Interpretado.
- Dinámicamente tipado pero fuertemente tipado.
- Independiente de la plataforma.
- Sintaxis clara, simple, concisa y elegante.
- Lenguaje indentado, no hay separadores de bloques.
- Librerías de todo tipo.
- Alta productividad.

La versión que se ha usado ha sido la ***3.2***

Además se han utilizado las siguientes librerías:

- ***OpenCV***: Se trata de una librería con Licencia *BSD* y por lo tanto es libre tanto para uso académico como para uso comercial. Tiene interfaces *C ++*, *C*, *Python* y *Java* y soporta *Windows*, *Linux*, *Mac OS*, *iOS* y *Android*. *OpenCV* fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. Escrito en *C / C ++* optimizado, la biblioteca puede aprovechar el procesamiento multi-core.
- ***Picamera***: Es una librería específicamente creada para interacturar con la cámara *Raspberry Pi Camera,* modelo de cámara usado en este trabajo. Este módulo también se encuentra bajo Licencia *BSD*.
- ***RPi.GPIO***: Se trata de una librería que permite gestionar los 40 pines del puerto *GPIO (General Purpose Input Output)* de la las *Raspberry Pi.* Este módulo se encuentra licenciado bajo Licencia *MIT.*
- ***Numpy***: Es una librería orientada a la computación científica con Python. También se encuentra bajo Licencia *BSD*, permitiendo la reutilización con pocas restricciones.

- ***Matplotlib***: Se trata de una librería para la representación gráfica en 2D desde Python. Permite crear figuras de gran calidad y en una gran variedad de formatos. La licencia de este módulo también es libre y está basada en la Licencia *PSF* (*Python Software Foundation*).
- ***Serial:*** Es un módulo para *Python* creado para la gestión del puerto serie. Este módulo también se encuentra bajo Licencia *BSD*.

Sistema Operativo.

En todos los *SBC* utilizados en el proyecto, se instaló la distribución *Raspbian Os*, una distribución *Linux* basada en *Debian Wheezy* especialmente pensada para la arquitectura *ARM.* Esta distribución integra por defecto, las herramientas de desarrollo necesarias para el lenguaje de programación *Python*.

En cuanto al PC usado se optó por instalar la distribución *Ubuntu Linux 16.04 para arquitecturas 64 bits*, también basada en *Debian GNU/Linux.* Se ha utilizado una distribución Linux orientada a equipos de escritorio aunque proporciona también soporte de servidores.

Otros programas utilizados.

Para la gestión del proyecto y organización de tareas se utilizó *OpenProj* que cuenta con licencia *Common Public Attribution License.*

Para la generación de los diagramas *UML* se ha usado *ArgoUML* que se encuentra bajo Licencia *BSD*.

Por último, para la elaboración de la documentación del proyecto, realización de cálculos (análisis de errores) y presentación final, se ha utilizado el paquete ofimático *LibreOffice*. Este paquete es Software Libre y está licenciado bajo *LGPLv3*.