ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

Trabajo de Fin de Grado

Transmisión digital con cámara óptica para comunicaciones VLC basada en Raspberry Pi

Titulación: Grado en Ingeniería Electrónica Industrial y Automática
Alumno: Ayoze Dioniz Rodríguez
Tutor: Silvestre Rodríguez Pérez

Septiembre, 2020

ÍNDICE

CAPÍTULO 1: Introducción

1.1 Resumen

Este trabajo tiene como objetivo diseñar e implementar un sistema de comunicación cuyo medio de transmisión es el canal visible o VLC. Por un lado, el envío de la información se realiza modulando la intensidad de una lámpara LED (*light-emitting diode*), mediante la modulación OOK (*On-Off Keying*). Por otro lado, para la recepción de la información se utiliza una cámara como dispositivo receptor, lo que permite obtener la información transmitida mediante la detección de dichas variaciones de intensidad. De esta manera, se establece una comunicación entre el LED (emisor) y la cámara (receptor).

La idea principal del mismo es que las variaciones de intensidad del LED que emiten la información no puedan ser detectadas por el ojo humano, ya que, iluminar es la condición fundamental de una lámpara y por tanto, se pretende que sea un sistema fácilmente integrable y funcional. Para este caso, se ha hecho uso de la misma tarjeta de desarrollo tanto para la emisión de los datos como para la recepción de los mismos (Raspberry Pi 3 B+), previniendo posibles problemas de sincronización entre emisor y receptor.

Como se comentó con anterioridad, la condición de que la transmisión de datos no sea detectada por el ojo humano, no ha sido posible debido a las limitaciones características de los componentes del sistema con el que se ha trabajado. Igualmente, dicho sistema ha sido diseñado pensando en dicha utilidad, siendo compatible en el caso de que se contara con los componentes adecuados para ello.

Por último, el procesamiento de imágenes se ha llevado a cabo mediante el uso de la librería OpenCV, junto al lenguaje de programación PYTHON.

1.2 Abstract

The objective of this project is to design and implement a communication system whose transmission medium is the visible channel or VLC. On the one hand, the information will be sent by modulating the intensity of an LED (*light-emitting diode*) lamp, through OOK (*On-Off Keying*) modulation. On the other hand, to receive the information, a camera will be used as a receiving device, this will allow obtaining the transmitted information by detecting said intensity variations. In this way, we will establish communication between the LED (*emitter*) and the camera (*receiver*).

The same plate has been used for the emission as well as for the reception, thus avoiding synchronization problems. The idea is that intensity variations are not detected by the human eye, since the fundamental condition of a lamp is to illuminate.

As we commented previously, regarding the condition that the transmission is not detected by the human eye, it has not been possible due to the characteristic limitations of the components of the system with which we have worked in this case. Likewise, the system has been designed with said utility in mind, being compatible in the event that the system had the appropriate components for it.

1.3 Introducción

En los últimos años, con la normalización del uso de los dispositivos inteligentes y la gran evolución de las nuevas tecnologías y la domótica, los sistemas de comunicación se han desarrollado significativamente. Debido al auge de la conectividad inalámbrica asociada a todas estas tecnologías, cada vez se registra un mayor aumento del número de usuarios inalámbricos, lo cual en numerosos entornos supone una sobre saturación de dichas redes. Dicho aumento demanda cada vez un mayor ancho de banda que sea capaz de satisfacer las necesidades de conectividad anteriormente citadas, debido a esta demanda se están desarrollando tecnologías alternativas de comunicación inalámbrica que no solo sean capaces de satisfacer esta demanda, sino que también contribuyan al desarrollo de una mayor sostenibilidad y eficiencia energética.

Algunas de estas tecnologías alternativas de comunicación inalámbrica han aprovechado el desarrollo y funcionalidad de las lámparas tipo LED. Este tipo de lámparas se caracteriza por tener una baja emisión de calor ya que están diseñadas con un disipador para evitar sobrecargas, también producen una luz nítida, brillante y fácilmente regulable con un encendido al instante que evita los parpadeos, aumentando la vida útil de esta entre 10 y 30 veces más en comparación con una lampara halógena. Otra característica muy importante sobre el LED es su baja contaminación al no funcionar ni con mercurio ni tungsteno, además, no emiten radiación infrarroja ni ultravioleta y, al ser más eficientes, producen menos CO_2.

La luz LED no solamente es capaz de iluminar o decorar una estancia, sino también de efectuar transmisiones de datos a través de ellas, sistemas que por lo tanto están basados en la comunicación por luz visible (VLC), usando la luz del LED como emisor y un sensor de imagen (IS) o una cámara como receptor. Ello permite integrar junto con los elementos decorativos los sistemas de emisión/recepción de datos y hacerlos más funcionales que otros sistemas inalámbricos.

Este trabajo consiste en el desarrollo de un sistema de comunicación VLC de transmisión con cámara óptica, basado en el sistema IEEE 802.15.7 en el que se emplea una lámpara LED para emitir un mensaje y una cámara para recibirlo. Todo esto controlado por la tarjeta de desarrollo Raspberry Pi 3 *Model* B+.

El funcionamiento del sistema se divide en tres bloques principales: emisión, recepción y conversión, tal como se muestra en la figura 1.1. Como se comentó anteriormente, el LED es el encargado de transmitir el mensaje, cuya recepción se realiza con una cámara, y la implementación del sistema se ha basado en la tarjeta de desarrollo indicada con anterioridad, encargada de sincronizar las etapas de emisión y recepción, adaptar el mensaje para su correcta transmisión, realizar su decodificación en recepción y mostrar el mensaje.

Figura 1.1: Diagrama de bloques del sistema.

Tal y como se aprecia en la figura 1.1, el bloque de emisión se encarga de, a partir del mensaje a transmitir, realizar una codificación según el código de línea RLL 4B6B (ver tabla 1.1 más adelante) y generar la señal a aplicar al diodo LED, es decir, transmitir el mensaje. El bloque de recepción, basado en el

empleo de una cámara óptica, es el encargado de detectar y capturar los cambios de luz que se producen en el LED, mediante la grabación de un video de recepción. En el bloque de conversión, una vez finalizada la recepción del video, éste se fragmenta en imágenes para compararlas entre sí. Mediante dicha comparación se determina si el LED está encendido o apagado. Durante la comparación se guardan los resultados en un fichero con formato txt, un uno lógico si se considera que el LED está encendido y un cero lógico si se detecta que el LED está apagado. Cuando termina la comparación, el programa lee el fichero, realiza la decodificación, y se muestra finalmente el mensaje que inicialmente fue transmitido a través de la pantalla.

1.4 Fundamentos teóricos

1.4.1 Tecnología LiFi

Cuando hablamos de LiFi nos referimos a un sistema de comunicación inalámbrica que hace uso del espectro visible de la luz como medio de transmisión de datos. Este término es usado para etiquetar a sistemas de comunicaciones inalámbricos rápidos y de bajo coste, tratándose del equivalente óptico al Wi-Fi. Este término fue usado por primera vez en un contexto de este tipo por Harald Haas en una conferencia en TED sobre la comunicación mediante luz visible.

Uno de los factores determinantes que ha llevado a la evolución de esta tecnología ha sido la aparición y extensión del uso del LED, que está reemplazando a las lámparas comunes. La capacidad de conmutación de estos dispositivos, junto con su menor demanda de energía y mayor sostenibilidad, han permitido que, aparte de servir como iluminación para diferentes ambientes y/o estancias se pueda enviar información a través de ellos, encendiéndolos y apagándolos con una velocidad que es imperceptible para el ojo humano.

La idea de tener como punto de comunicación una lámpara es muy potente, ya que, se puede obtener información muy rápido y sin la necesidad de dividir el ancho de banda. Además, este tipo de sistema nos permite eliminar la necesidad de hacer grandes cambios en la infraestructura y de hacer uso de dispositivos complejos (Barajas,2013).

1.4.2 Cómo funciona la red LiFi

Como indicamos en el apartado anterior, la tecnología LiFi usa la luz LED para transmitir señales inalámbricas que contienen una carga de datos. Lo hace mediante la codificación de dicha información, basada en la frecuencia de la luz LED, de manera tal, que no es perceptible al ojo humano. La velocidad de

trasmisión está relacionada directamente con el color de las luces LED, ya que, entre más lejana del blanco cálido, más rápida es la transmisión, debido a que puede usarse un mayor espectro de frecuencia.(Barajas,2013).

El funcionamiento de LiFi:

Para poder transmitir información, se necesita conectar una matriz de LED equipada con LiFi, de este modo el transmisor LiFi estará listo para realizar el envio de información, únicamente necesitamos un dispositivo que tenga instalado el receptor de LiFi, completando así el ciclo de enviar y recibir información de un modo correcto.

VLC:

Esta tecnología surge a partir de la unión de la comunicación visual con la comunicación sin cables. Emplea la propagación de la luz en el espacio para transmitir información entre un emisor y un receptor, es decir, utiliza la luz visible para enviar información al mismo tiempo que para iluminar. Este tipo de sistemas son de gran utilidad para situaciones en las cuales la utilización de medios físicos para la transmisión de información resulta imposible, es costosa o presenta algún otro inconveniente. Al mismo tiempo cabe destacar que el buen uso de la misma supone un mayor cuidado del medio ambiente y un ahorro energético considerable.

Representa sólo una fracción de lo que parece ser un movimiento mucho más grande hacia la óptica inalámbrica. El transmisor LED sostendrá un micro-chip que va a hacer el trabajo de procesado de datos. La intensidad de la luz puede ser manipulada para enviar los datos por pequeños cambios en la amplitud. Esta tecnología utiliza el espectro visible de la luz, una parte del espectro electromagnético que todavía no es en gran medida utilizado, de hecho, esta tecnología transfiere miles de flujos de datos de forma simultánea y en paralelo a una velocidad más alta (Sanchez, 2013).

Ventajas:

- Una de las principales ventajas es que el ancho de banda no se divide, independientemente de los usuarios que ocupen servicio.
- No interfiere con otras señales como las de radio frecuencia.
- Es muy rápido, su velocidad de transferencia va desde los 15 Mb/s, hasta los 20 Gb/s.

- La dualidad, es decir que al mismo tiempo ilumina el ambiente y permite la transmisión de datos lo que produce un ahorro de energía.
- Cualquier bombilla o farola puede convertirse en un *hotspot* o *router* luminoso de forma barata y sencilla, poniéndole un emisor LiFi.
- No requiere las cotizadas frecuencias radioeléctricas que requiere el Wi-Fi.
- LiFi es mucho más simple y utiliza métodos de modulación directos, similares a las utilizadas en los dispositivos de comunicaciones de infrarrojos de bajo coste tales como los mandos a distancia.
- Ausencia de cables, mayor sostenibilidad y ahorro energético.

Desventajas:

- No funciona bajo la luz solar directa.
- No puede cruzar obstáculos o paredes.
- Solo funciona con aquellos dispositivos que tengan un receptor para tal tecnología, es decir, que cuenten con un receptor capaz de descodificar la señal luminosa.
- El alcance de haz de luz en los LED no es muy amplio alcanzando hasta los 0.5 -1 m.
- Si interfiere algún objeto entre el emisor o receptor, se corta la transferencia de datos, ya que las ondas de luz visible no traspasan objetos como si lo hacen las ondas de radio frecuencia.

1.4.3 Sistemas VLC

Las comunicaciones por luz visible (VLC), consisten en la transmisión de datos por medio del espectro de luz visible, se trata de una nueva alternativa, que se ha estudiado en profundidad durante los últimos años, que permita desahogar al saturado espectro de radio frecuencia. Además, el gran desarrollo y normalización en el uso de los dispositivos LED ha permitido que las bombillas y sistemas de iluminación convencionales se estén sustituyendo por esta nueva tecnología que nos aporta mayor durabilidad, mayor ahorro energético y por consiguiente, muchísima más eficiencia y coste. La elevada velocidad de conmutación de estos dispositivos ha hecho posible su utilización no sólo como sistema de iluminación sino también como un medio para transmitir la información, lo que se conoce como VLC, especificado en el estándar 802.15.7.

1.4.4 Estándar IEEE 802.15.7

Es el protocolo de luz visible de comunicación, este método se caracteriza por la comunicación inalámbrica de corto alcance por medio de espectros de luz donde su mayor ventaja es soportar altas velocidades de datos hasta 96Mb/s y la luz visible es de 380 – 789 nm, este método proporciona atenuación en mecanismos durante la comunicación de luz para que siempre sea visible sin parpadeos. A finales del año 2011 fue producido el proyecto de norma para IEEE 802.15.7 para VLC. Esta norma cubre tanto la capa (PHY2) área de interfaz física, realmente importante ya que utiliza la luz visible, como el control de acceso al medio (MAC). Se propone el estándar para una variedad de aplicaciones relacionadas con VLC *Wireless Personal Área Networks* (WPAN).

Actualmente el MAC es compatible con tres múltiples topologías de acceso; *peer-to-peer*, configuración de estrella y en el modo de difusión. La MAC también se ocupa de los problemas de gestión de la capa física, tales como protocolos de direccionamiento. La capa física se divide en tres tipos; PHY I, II y III, y éstos emplean una combinación de diferentes esquemas de modulación. (Nawaz *et al*, 2019)

1.4.5 Modulación OOK

En su forma más simple, un indicador digital de '1' está representado por la luz de estado "*On*" y un indicador digital de '0' está representado por un estado de la luz en "*Off*". Se trata de un método simple en cuanto a la generación y detección. El estándar 802.15.7 utiliza codificación Manchester para garantizar que el periodo de impulsos positivos es el mismo que los negativos, pero esto también duplica el ancho de banda requerido para la transmisión OOK. Por otra parte, para garantizar que la lámpara no deje de realizar su principal función de iluminar se suelen emplear códigos RLL (*Run Lenght Limited*), cuya función es la de aleatorizar los datos a transmitir. En la figura 1.3 se muestra la estructura del sistema emisor que se ha implementado en este trabajo, donde se puede observar el bloque encargado de la codificación, mencionado anteriormente, y el bloque que lleva a cabo la modulación OOK.

La modulación OOK es un caso particular de la modulación por desplazamiento de amplitud (*Amplitude Shift Keying*, ASK), es una forma de modulación en la cual se representan los datos digitales como variaciones de la amplitud de la onda portadora, manteniendo la frecuencia y la fase constante.

Cómo se ha indicado con anterioridad, la modulación OOK es el formato de modulación más simple utilizado en comunicaciones ópticas, el cual consiste en emitir un pulso de luz cada vez que la información que llega al transmisor es un bit 1 y nada en el caso de que la información que llega al transmisor es un bit 0 como se puede observar en la figura 1.2.

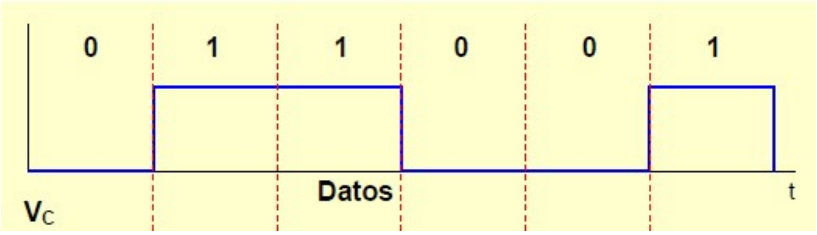


Figura 1.2: Representación de la modulación OOK.

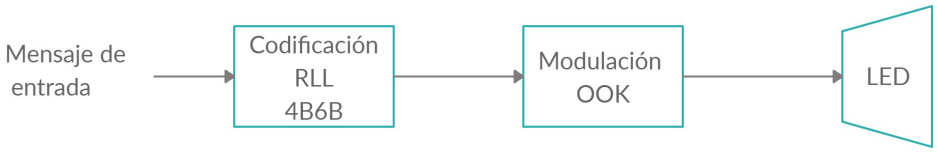


Figura 1.3: Diagrama de bloques de la modulación OOK.

1.4.6 Código RLL

Los códigos de línea se usan para evitar series largas de unos y ceros que podrían causar potencialmente problemas de detección de parpadeo. Estos códigos de línea RLL tienen como función aleatorizar los datos a transmitir garantizando el equilibrio de unos y ceros, es decir, haciéndolos equiprobables. En el estándar IEEE 802.15.7 se especifica la utilización de varios tipos de códigos de línea del tipo RLL como el código Manchester, el 4B6B o el 8B10B.

En este trabajo se ha empleado el código 4B6B, el cual expande símbolos codificados de 4 bits de entrada a 6 bits de salida, que se transmite en números binarios y se caracteriza por tener la misma cantidad de ceros y unos, como podemos observar en la tabla 1. Esta codificación garantiza un mejor funcionamiento del LED al evitar varios ceros consecutivos previniendo el apagado del LED (Ghassemlooy, 2017).

Las características del código 4B6B son las siguientes:

- Siempre 50% de ciclo de trabajo durante un símbolo codificado.
- Código RLL equilibrado en DC.
- Capacidad de detección de errores.
- Permite una recuperación razonable del reloj.

4B (input)	6B (output)	Hex
0000	001110	0
0001	001101	1
0010	010011	2
0011	010110	3
0100	010101	4
0101	100011	5
0110	100110	6
0111	100101	7
1000	011001	8
1001	011010	9
1010	011100	A
1011	110001	B
1100	110010	C
1101	101001	D
1110	101010	E
1111	101100	F

Tabla 1.1: Código RLL 4B6B.

1.4.7 OpenCV

Para el reconocimiento de imágenes se ha utilizado la librería OpenCv, librería que contiene más de 2500 algoritmos, que incluyen algoritmos de visión por computador y de *machine learning*, todo ello en código abierto.

OpenCV es una biblioteca libre desarrollada originalmente por Intel que vio la luz en el año 1999. Basándose originalmente su estructura en C/C++, siendo su mejor virtud que es multiplataforma, se puede ejecutar en diferentes sistemas operativos como Linux, Windows, Mac OS X, Android e iOS. También la podemos utilizar en diferentes lenguajes de programación como C, C++, Java, Objective C, y Python.

Esta librería incluye algoritmos que permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios, etc.

Se usa en aplicaciones como la detección de intrusos en vídeos, monitorización de equipamientos, ayuda a navegación de robots, comparación de imágenes, inspeccionar etiquetas en productos, entre otros. Y por todo esto, es la biblioteca de visión artificial más usada en el mundo.

CAPÍTULO 2: Sistema

2.1 Descripción del sistema.

El sistema se divide en tres bloques: emisión, recepción y conversión. En la etapa de emisión se utiliza un diodo LED para el envío del mensaje. Para la recepción se dispone de una cámara, modelo PI CAMERA V2 descrita posteriormente, encargada de recibir la señal del LED. La Raspberry Pi , Model 3 B+ mostrada también más adelante, será la encargada de sincronizar las etapas de emisión y recepción, luego convertir la información dada para mostrar el mensaje inicial en pantalla.

Figura 2.1: Diagrama de bloques del sistema.

El sistema comienza en el bloque de emisión como se ve la figura 2.1, se encarga de recibir el mensaje, luego lo codifica con el sistema RLL 4B6B, es decir, lo convierte de 4 bits a 6 bit, asignando su correspondiente valor según en la tabla 1.1 mostrada más adelante. La transmisión del mensaje se realiza a través de la modulación OOK, si el bit a transmitir es un "1", el LED no cambiará por lo que continuará encendido, si por el contrario fuera un "0", este se apagará.

Para la recepción se ha utilizado una cámara modelo PI CAMERA V2, detallada más adelante. Por medio del código hemos programado la duración del video, en este caso dos segundos, y modificado sus características, como brillo, contraste, duración, *frames* por segundo (FPS), etc. A través de la cámara grabamos en video la secuencia de parpadeo del LED, esta frecuencia se ha sincronizado con la velocidad de captura de fotogramas de la cámara (FPS). El video esta sincronizado con la emisión, de manera que cuando el LED envía el mensaje la cámara está grabando. Una vez termine de grabar el parpadeo dicho video se fragmenta en imágenes.

Una vez está el video descompuesto en imágenes se pasa a la etapa de conversión, en el que hay un comparador de imágenes, donde se diferencia si el LED está encendido o apagado. Para determinar esto, se compara la primera imagen con el LED encendido con el resto de imágenes que se han fragmentado, la comparación se realiza en escala de grises para evitar coincidencias no deseadas. El programa comparará puntos de similitud entre ellas, si el comparador detecta varios puntos iguales en la imagen dará un alto

porcentaje de similitud, por lo cual el LED estará encendido, de lo contrario dará un bajo porcentaje, lo que indicará que el LED está apagado.

El sistema se ha programado para que los resultados se guarden en un fichero con formato txt, almacenando un uno lógico si se considera que el LED está encendido y un cero lógico si se detecta que el LED está apagado.

Cuando finaliza la comparación, el programa leerá el fichero txt con los resultados almacenados, este realiza la decodificación 4B6B que transforman las cadenas de 6 bit recibidos en 4 bits, de acuerdo con su valor establecido en la tabla 1,1. Y finalmente al acabar esta, muestra el mensaje inicialmente transmitido.

2.2 Componentes.

RASPBERRY Pi

Raspberry Pi es un micro-ordenador SBC (*Single Board Computer*) creado por Eben Upton junto con un grupo de profesores, académicos e informáticos a través de su fundación Raspebrry Pi ubicada en Reino Unido. Este proyecto nace con la intención de Upton de animar especialmente a los jóvenes y al público en general a aprender informática.

Sin duda alguna estamos ante un micro-ordenador que ha sentado un hito, no solamente en lo que al mundo de la informática se refiere, sino que también lo ha hecho en el ámbito de la tecnología en todo su conjunto. La primera aparición de Raspberry Pi, como la conocemos hoy día, data de febrero de 2012 aunque la versión más primitiva de esta la ubicamos en 2006, esta última no era más que un micro-controlador con un circuito impreso, montado en un módulo del tamaño de una memoria USB con un puerto HDMI en un extremo y, un puerto USB en el otro. No será hasta el año 2011 cuando se crearán los primeros prototipos de este micro-ordenador que culminará en producto final en el año 2012.

Actualmente se disponen de siete modelos de Raspberry Pi en el mercando que van desde la versión inicial de 2012 Raspberry Pi 1 hasta Raspberry Pi 4 B, cuyo lanzamiento data de junio de 2019 (Velasco, 2018).

RASPBERRY PI 3 B+

Para este trabajo se utilizará la Raspberry Pi 3 Model B+, la cual se muestra en la Imagen 2.1, modelo que data de marzo 2018. Este nuevo micro-ordenador ha supuesto un rediseño absoluto de la tarjeta de desarrollo, manteniendo el mismo tamaño y la misma posición de los elementos que en el modelo Pi 3, pero se ha cambiado el procesador por otro más potente que funciona a 1,4 GHz, y además elimina el cuello de botella de la conectividad incluyendo Bluetooth 4,2, BLE, Wi-Fi a doble banda 2,4 GHz y 5 GHz y, además, la tarjeta de red, Gigabit Ethernet, ya no está limitada a los 100 Mbps, sino que es capaz de alcanzar los 300 Mbps al funcionar sobre USB 2.0 (Velasco, 2018).

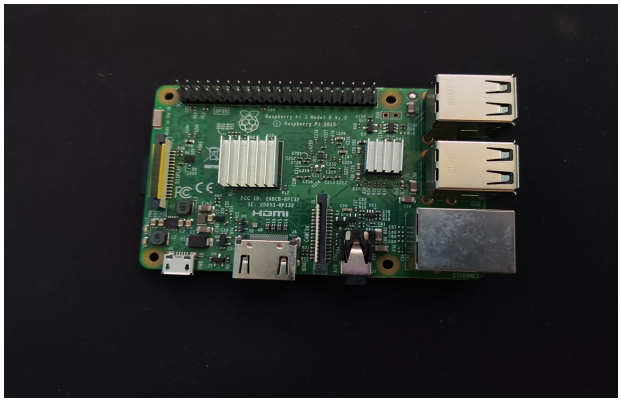


Figura 2.2: Raspberry Pi 3 b+

PI CAMERA V2

Es un módulo oficial de Raspberry lanzado en 2016, cuenta con un sensor IMX219 de Sony con 8 megapíxeles, es capaz de tomar fotografías como de grabar videos a altas resoluciones(1080p) sacrificando fps (30 FPS), sin embargo, si se graba a una resolución más baja ganamos más FPS (720p a 60 FPS, 480p a 60/90 FPS). Parte importante del trabajo ya que grabaremos a 640x480p a 20 fps.

La cámara, que se expone en la figura 2.3, funciona con cualquier modelo de Raspberry Pi y cuenta con numerosas bibliotecas de terceros que han sido creadas para ella, una de las más importantes es la biblioteca PiCamera Python.

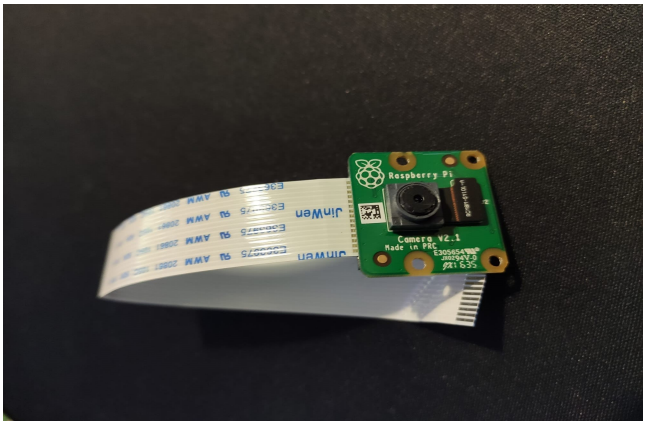


Figura 2.3: Pi Camera V2

DIODO LED BLANCO

La idea principal de este trabajo era utilizar una lámpara LED, pero debido a la situación de alarma por el COVID-19, que no ha permitido acceder al laboratorio, se empleado como dispositivo emisor del mensaje un diodo LED de luz blanca como se puede apreciar en la figura 2.4.

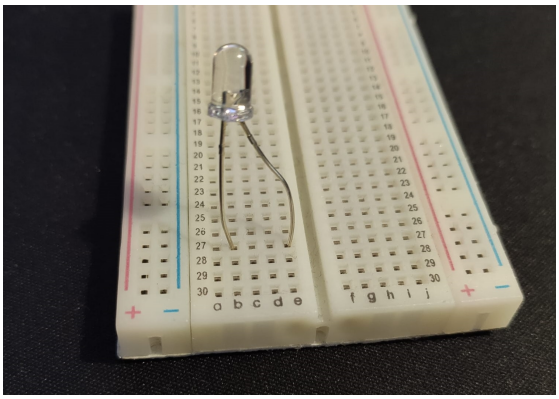


Figura 2.4: Diodo LED

2.3 Implementación del sistema.

Como ya se ha comentado, para implementar el sistema se ha utilizado una Raspberry Pi 3 Model B+. El sistema se ha dividido en cuatro partes:

- Conexionado: conexión de la Raspberry al circuito.
- Emisión: esta parte se encarga de codificar y enviar el mensaje.
- Recepción: se encarga de recibir el mensaje a través de la cámara.
- Conversión: realiza la decodificación y conversión 4B6B del mensaje recibido.

Conexionado:

La Raspberry cuenta con 40 pin de conexión, como se puede observar en la figura 2.5 la conexión con la placa de pruebas o prototipo (*protoboard*), se realiza a través de los pines 6 y 12 de la misma. El pin 6 es la conexión a tierra y el pin 12 es el GPIO 18 un puerto de entrada/salida, en este caso se utiliza como puerto de salida, que se utiliza para generar el pulso banda base que controlará la emisión del LED, es decir, su encendido y apagado, según se establece en la modulación OOK.

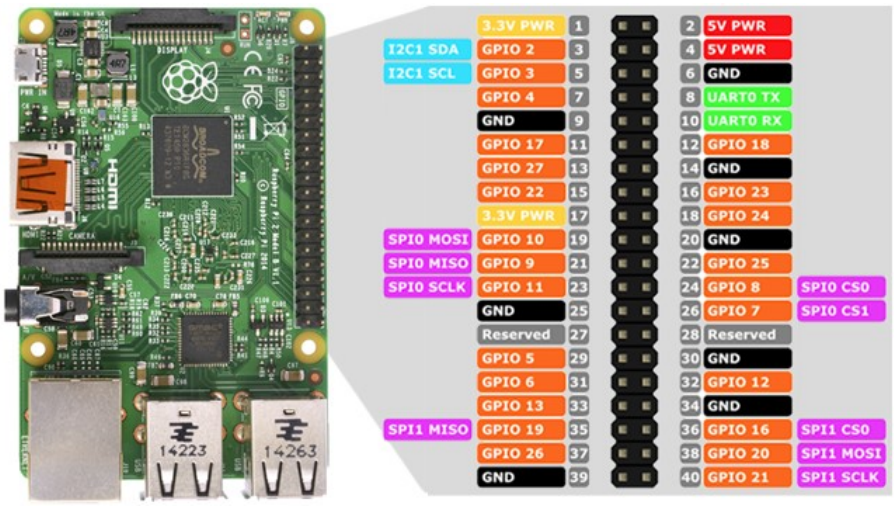


Figura 2.5: Estructura de pines de la Raspberry Pi 3 B+

El pin 6 (GND) se conecta a la placa *protoboard*, como se ve en la figura 2.2, que a su vez se conecta a un extremo de la resistencia de 330 Ω, cuyo otro extremo irá conectado al ánodo del diodo LED. Para que

el diodo LED se encienda o apague en función de la señal generada a través del pin 12 (GPIO 18), dicho pin se ha conectado al cátodo del LED.

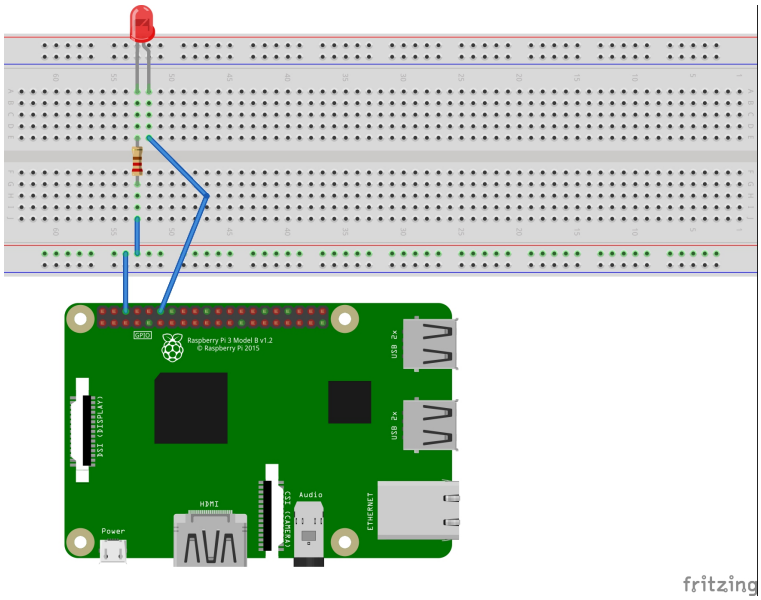


Figura 2.2: Circuito de conexiones.

Una vez realizado el conexionado de la placa *protoboard*, solo queda conectar la cámara en la ranura correspondiente en la Raspberry pi, la cual se muestra en la figura 2.6.

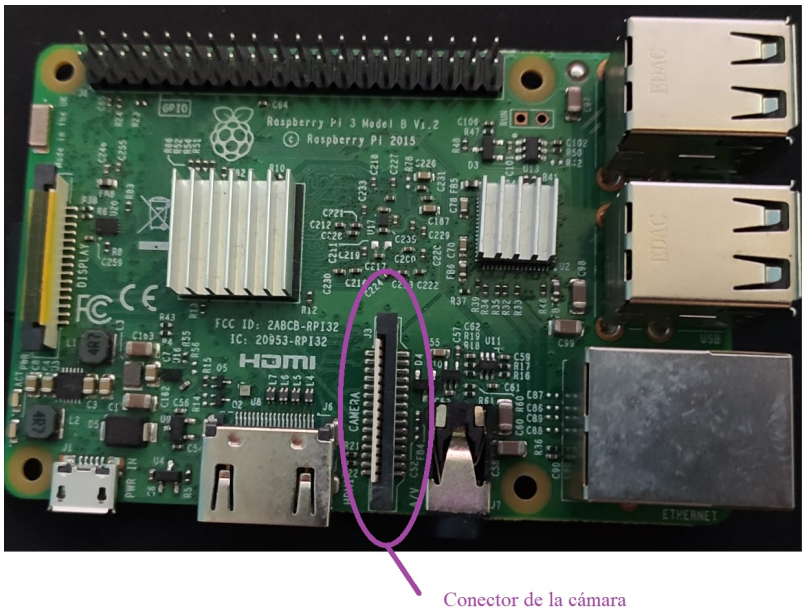


Figura 2.6: Conector para la Pi Camera V2

Emisión:

El bloque de emisión es el encargado de escanear la secuencia de bits a transmitir, y realizar la codificación 4B6B, es decir, según detecta la secuencia la convierte de 4 bits a 6 bit según lo estipulado en la tabla 1.1, que se recoge en el estándar IEEE 802.15.7 VLC. Como se comentó con anterioridad, dicha conversión se realiza con la finalidad de aleatorizar los datos para eliminar cadenas largas de unos y ceros lógicos, evitando que se pueda observar el parpadeo del LED.

La secuencia de datos generada se muestra en pantalla junto con la secuencia codificada, para comprobar si el mensaje emitido y el recibido son similares, es decir, para comprobar el correcto funcionamiento de la transmisión.

La transmisión del mensaje se efectúa a través de la modulación elegida anteriormente, la modulación OOK. El LED estará inicialmente encendido, tal como se ve en la figura 2.7, si el bit a transmitir es un "1", el LED continuará encendido, si por lo contrario fuera un "0", se apagará. La velocidad de encendido/apagado se ha fijado a 0,05 s (20 Hz), una velocidad que en este caso es detectable para el ojo humano, pero ajustada para que la cámara pueda detectarla sin problemas. Una vez enviada la secuencia, el LED seguirá encendido continuando con la función principal que debería de tener una lámpara LED, que es la de iluminar.

Recepción:

Previamente a este paso, mediante código hemos ajustado las características de la cámara, como se aprecia en la figura 2.7, de manera que se pueda usar en sitios con distinta luminosidad, modificando brillo, contraste, exposición, ISO, etc. De la misma manera, también se ha ajustado los *frames* por segundo a los que graba la cámara, sincronizándolo con la velocidad de transmisión del LED.

```
with picamera.PiCamera() as picam:   ##Abrimos la cámara y se configuran sus características
    picam.rotation = 180
    picam.resolution = (640, 480)
    picam.awb_mode= 'auto'
    picam.exposure_mode = 'auto'
    picam.contrast= -10
    picam.brightness= 30
    picam.ISO = 700
    picam.framerate=21
    picam.start_preview()
    time.sleep(1)
```

Figura 2.7 Características de la cámara.

La recepción está sincronizada con la emisión, de modo que antes de que el LED envíe el mensaje la cámara empezará a grabar. La cámara graba videos de dos segundos, tiempo en el que se produce la transmisión del mensaje.

Conversión:

Esta parte es la encargada de realizar la decodificación y conversión 4B6B del mensaje recibido por la cámara. Es decir, una vez finalizada la recepción el video, éste se fragmenta en imágenes, aproximadamente en unas 40. Una vez se dispone de todas las imágenes comienza el proceso de comparación, el cual consiste en comparar cada una de las imágenes con la primera de ellas, donde el LED estaba emitiendo continuamente un único lógico.

La comparación se realiza en escala de grises, esta se basa en analizar si las imágenes son iguales o no a la primera de ellas, analizando la coincidencia o no de sus características. Para ello se usa el comando SIFT (*Scale Invariant Feature Transform*) que se muestra en la figura 2.8, en la primera línea solo cargamos el comando, este se encarga de localizar los puntos clave importantes de cada imagen.

La detección de estos puntos de interés tiene como objetivo encontrar regiones interesantes o áreas especiales en la imagen, estos se consideran especiales porque no importa como cambie la imagen, que debe poder encontrar los mismos puntos clave en una imagen modificada cuando se compara con la primera de ellas. En las siguientes dos líneas encontramos los puntos clave y los descriptores de la imagen original y de la imagen para comparar. Por último, solo se imprime el tamaño de los *keypoints* de cada imagen.

```
149
150     shift = cv2.xfeatures2d.SIFT_create()
151     kp_1, desc_1 = shift.detectAndCompute(img1, None)    ##Se obtienen los puntos clave y los descriptores
152     kp_2, desc_2 = shift.detectAndCompute(img2, None)
```

Figura 2.8 Comando de obtención de puntos clave y descriptores.

Con el comando FlannBasedMatcher se encuentran las coincidencias entre los descriptores de las dos imágenes, después de encontrar las coincidencias, estas se almacenan en una matriz *matches*. La matriz contendrá todas las coincidencias posibles, pero también muchas coincidencias falsas. Para evitar esto, el programa selecciona los *match* o coincidencias de mayor calidad. La calidad de un *match* se determina por la distancia entre descriptores. Por lo que el programa tomará siempre los *match* con menor distancia, limitando así también el número de coincidencias, pero asegurando que estas sean buenas. Si se compara la figura 2.9 y la figura 2.10 se puede apreciar como en la primera las coincidencias entre LED encendido / LED apagado

son muy bajas por lo cual se ha detectado que el LED estaba apagado, mientras que en la segunda se aprecia una mayor cantidad de coincidencias por lo que se detecta que el LED está encendido.

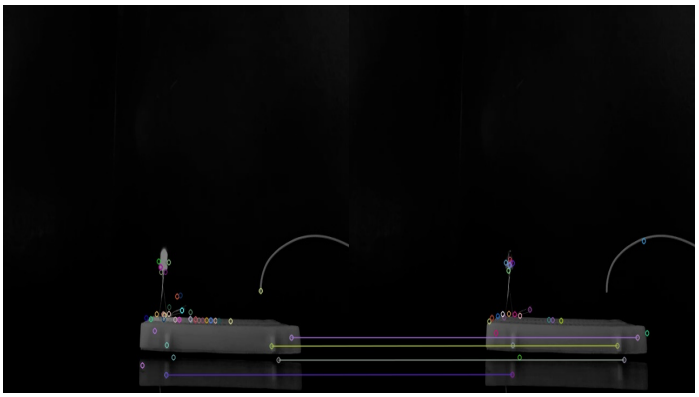


Figura 2.9: Comparación LED encendido / LED apagado

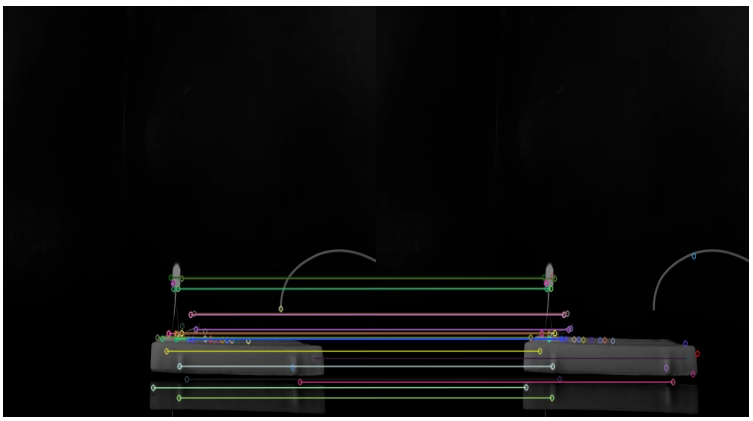


Figura 2.10: Comparación LED encendido / LED encendido

Tanto estas coincidencias como los *matches* buenos saldrán en pantalla, como se ve en la figura 2.11, en cada comparación de imágenes.

```
frame0.jpg
frame0.jpg
('Keypoints 1st imagen', '86')
('Keypoints 2st imagen', '86')
('GOOD matches', 86.0)
('Porcentaje de similitud', 100.0, '%')
```

Figura 2.11 Inicio de comparación.

Por último, se calcula el porcentaje de similitud (16%) con el que se decide si una imagen es igual a otra o por el contrario son distintas. Se verifica el número de puntos clave de ambas imágenes utilizando len(kp_1) y len(kp_2) y se toma el número de las imágenes que tiene menos puntos clave. Luego se dividen las buenas coincidencias por el número de puntos clave. Obteniendo un número entre 0 (si no hubo coincidencias) y luego se multiplica por 100 para obtener un resultado porcentual, dicha operación se ve reflejada en la figura 2.12.

```
170
171     number_keypoints = 0                ##Calcula el porcentaje de similitud
172     if (len(kp_1) <= len(kp_2)):
173         number_keypoints = float(len(kp_1))
174     else:
175         number_keypoints = float(len(kp_2))
176     porce = 0
177     x = float(len(good_points))
178     y = float(number_keypoints)
179     porce = float(((x / y) * 100.0))
180     print("GOOD matches",x)
181     print("Porcentaje de similitud",float(porce),'%')
182
183
```

Figura 2.12 Comando del cálculo de porcentaje de similitud.

Si una imagen es similar a otra es porque en ambas el LED está encendido por lo que el resultado será que se ha recibido un “1”, por el contrario, si no fueran iguales, indicaría que lo que se ha recibido es un “0”. Como se comentó con anterioridad, los resultados se guardan en un fichero en formato txt. Una vez acabada la comparación se pasa el fichero con los resultados por el decodificador 4B6B, cuya función es realizar el proceso inverso que se muestra en la tabla 1.1, es decir, se realiza una conversión de palabras de 6 bits a 4 bits. Una vez realizada la conversión, se dispone del mensaje recibido, que también se muestra en la pantalla con la finalidad de compararlo con el transmitido.

CAPÍTULO 3: Resultados

En este apartado se muestra la aplicación del algoritmo en el que se basa el sistema desarrollado, el cual se muestra en la figura 3.1, así como distintos tipos de resultados de la comparación de imágenes en el proceso de recepción, concretamente en el proceso de decisión de si se ha recibido un uno o un cero lógico.

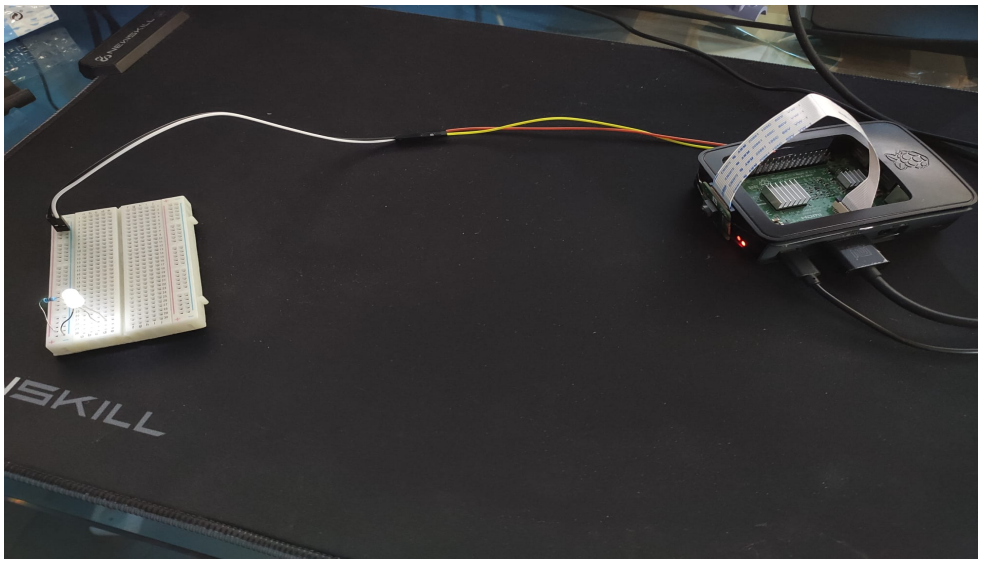


Figura 3.1 Sistema en funcionamiento.

Una vez el programa escanea el mensaje, este se muestra en pantalla como en la figura 3.2, tanto el mensaje recibido como el codificado. Una vez mostrado en pantalla el mensaje se envía a través del LED.

```
pi@raspberrypi:~/Desktop/proces $ python procesos.py
El codigo enviado es 0001
El codigo codificado es 001101
AVC-H264 import - frame size 640 x 480 at 25.000 FPS
AVC Import results: 41 samples - Slices: 1 I 40 P 0 B - 0 SEI - 1 IDR
Saving video.mp4: 0.500 secs Interleaving
vid convertido
```

Figura 3.2 Inicio del Programa.

Una vez el mensaje es emitido, recibido y transformado en imágenes, éstas se comparan con la primera imagen correspondiente al LED encendido (uno lógico), mostrado en la figura 3.3. La primera imagen siempre nos dirá si el sistema está bien calibrado. Ya que siempre deberá darnos el 100% de similitud puesto que se está comparando la misma imagen (en este caso el *frame*) como se muestra en la figura 3.4, de lo contrario ha habido un error.

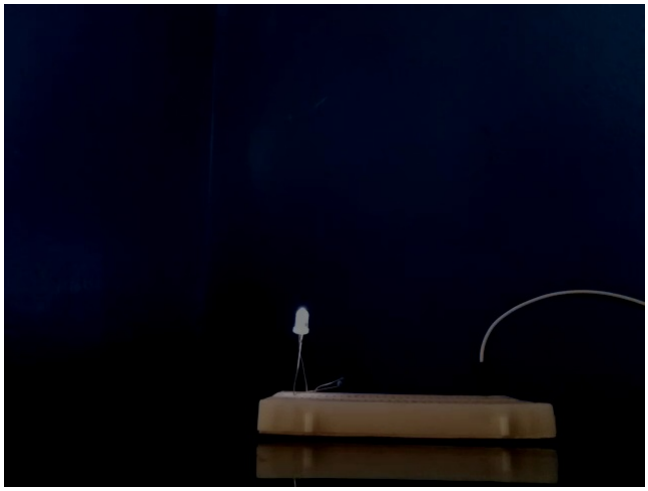


Figura 3.3: Diodo LED encendido.

```
frame0.jpg
frame0.jpg
('Keypoints 1st imagen', '86')
('Keypoints 2st imagen', '86')
('GOOD matches', 86.0)
('Porcentaje de similitud', 100.0, '%')
```

Figura 3.4. Resultado de un LED encendido.

Durante la comparación siempre existirán dos opciones de decisión, que el LED esté apagado o encendido, es decir, que se haya recibido un cero o un uno lógico. Se considera que una imagen es similar a la primera (LED encendido) cuando su similitud ronde entre el 22% y el 40%, tal y como se aprecia en la figura 3.5 y la figura 3.6, puesto que las imágenes nunca son exactamente iguales. Dicha comprobación aparecerá en pantalla durante el momento de procesado, así que se puede ir viendo cual es la similitud imagen a imagen.

```
('Keypoints 1st imagen', '86')
('Keypoints 2st imagen', '81')
('GOOD matches', 18.0)
('Porcentaje de similitud', 22.22222222222222, '%')
```

Figura 3.5 Resultado de un LED encendido a baja similitud.

```
('Keypoints 1st imagen', '86')
('Keypoints 2st imagen', '80')
('GOOD matches', 32.0)
('Porcentaje de similitud', 40.0, '%')
```

Figura 3.6 Resultado LED encendido a alta similitud.

Cuando la similitud baja a un porcentaje entre el 8 y el 16 % significará que el LED está apagado tal como se ve en la figura 3.8, ya que es un porcentaje muy bajo de similitud entre imágenes. El programa detecta menos puntos de interés en el segundo fotograma comparado con el primero como aparece en la figura 3.7.

```
('Keypoints 1st imagen', '86')
('Keypoints 2st imagen', '90')
('GOOD matches', 7.0)
('Porcentaje de similitud', 8.13953488372093, '%')
```

Figura 3.7 Resultado de un LED apagado.

Figura 3.8 Diodo LED apagado.

Como se mencionó anteriormente, el porcentaje de similitud máximo que admitimos para considerar que el LED está apagado es del 16%. Este umbral de similitud se ha extraído durante el proceso de calibración del sistema. Esto es debido a que el diodo LED tiene la cubierta del mismo color que la luz que emite y a veces nos daba falsos negativos al quedar residuos de luz en la imagen. Si la imagen supera el 16% de similitud se entiende que el LED está encendido. Esto se puede apreciar en la figura 3.9, donde se ve el límite entre encendido y apagado.

```
('Keypoints 1st imagen', '86')
('Keypoints 2st imagen', '65')
('GOOD matches', 10.0)
('Porcentaje de similitud', 15.384615384615385, '%')
```

Figura 3.9 Resultado de un LED apagado en su límite de similitud.

Para finalizar, una vez acabada la comparación, se mostrará en pantalla el mensaje codificado que ha detectado el programa y su conversión hecha, para su comprobación con el mensaje inicial, esto es enviado junto con un mensaje de cierre del sistema tal y como se muestra en la figura 3.10.

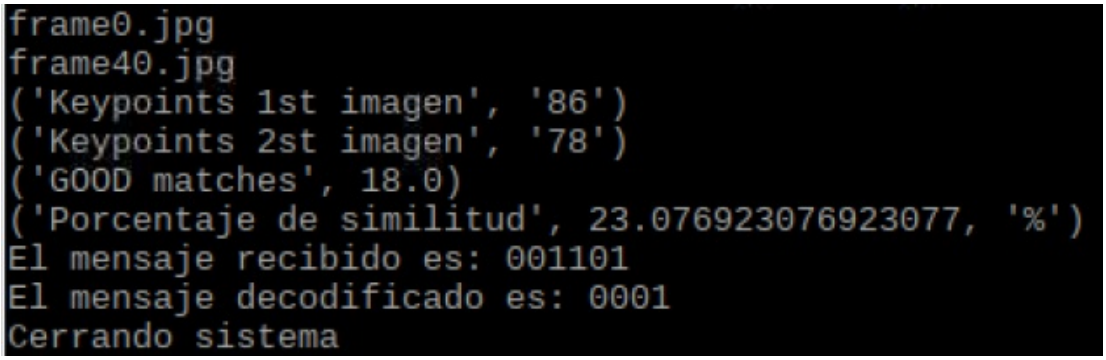


Figura 3.10 Cierre del sistema.

CAPÍTULO 4: Presupuesto

4.1 Coste de Materiales.

Concepto	Coste (€/u)	Cantidad (u)	Total (€)
Raspberry Pi 3 Modelo Bplus	38,95	1	38,95
Pi Camera Module V2	25,90	1	25,90
LED Kingbriht Blanco	1,20	1	1,20
Resistencia RS PRO, 330Ω	0,12	1	0,12
Placa de Prueba, 80x60x10 mm	11,83	1	11,83
		Total:	78,00

4.2 Coste de mano de obra.

Concepto	Coste (€/u)	Cantidad (h)	Total (€)
Análisis	50,00	120	6.000,00
Codificación	60,00	250	15.000,00
Implementación	30,00	50	1.500,00
Documentación	20,00	50	1.000,00
		Total:	23.500,00

4.3 Coste total del proyecto.

Concepto	Coste (€)
Coste de materiales	78,00
Coste de mano de obra	23.500,00
Gastos generales: 5% (CMT + CMO)	1.178,90
Beneficio: 15% (CMT + CMO)	3.536,70
Total del proyecto	28.293,60

CAPÍTULO 5: Conclusiones

El objetivo principal de este Trabajo de Fin de Grado (TFG) ha sido, teniendo en cuenta la gran demanda y evolución de las comunicaciones inalámbricas, diseñar un sistema de transmisión digital de baja velocidad que utiliza como medio de transmisión la luz visible (VLC), caracterizado por emplear una lámpara LED y una cámara óptica como dispositivo emisor y receptor, respectivamente. Siendo una de sus principales características que la variación de intensidad de la luz no sea perceptible al ojo humano, lo cual no se ha podido cumplir por la limitación de las características de la cámara empleada. Se ha adaptado la velocidad de transmisión para que la cámara sea capaz de detectar la variación de intensidad del LED, pero el programa ha sido diseñado de forma que sea compatible con componentes que posean características que permitan una transmisión a mayor velocidad. Este hecho es bastante importante ya que la adaptabilidad del programa implica una gran versatilidad a la hora de aplicarlo en diferentes entornos y mayores velocidades.

Se ha diseñado el sistema de transmisión VLC, pudiendo realizar perfectamente la conversión de los datos según el estándar IEEE 802.15.7 VLC, a través de la modulación OOK y el sistema de codificación RLL 4B6B. Para su implementación se ha empleado la tarjeta de desarrollo Raspberry Pi 3 B+ tanto realizar el bloque emisor como el receptor. Como emisor se ha empleado un diodo LED de luz blanca y como receptor se ha empleado una Pi Camera.

Se ha podido comprobar que el sistema funciona a una velocidad de transmisión baja debido a las limitaciones de los componentes utilizados, como se mencionó con anterioridad. El porcentaje de efectividad es de un 20% aproximadamente, debido a los falsos positivos a la hora de identificar las imágenes. Esto se debe a las características de la cámara y a la baja luminosidad del diodo LED. Si se utilizara una cámara con mejores características, no tanto en resolución, sino con un índice de *frames* por segundo bastante mayor, se podría aumentar considerablemente la velocidad de transmisión del LED hasta hacerla imperceptible al ojo humano. Y una lámpara LED en lugar del mencionado diodo LED, al ser su intensidad superior a la del diodo LED, el programa distinguiría las variaciones de intensidad con mayor facilidad. En cuanto al color del LED nos sería indiferente, puesto que el programa hace la comparación de imágenes en escala de grises.

Conclusion

The main objective of this Final Degree Project (TFG) has been, taking into account the great demand and evolution of wireless communications, to design a low-speed digital transmission system that uses visible light (VLC) as a transmission medium, characterized by using an LED lamp and an optical camera as a transmitter and receiver device, respectively. One of its main characteristics is that the variation in light intensity is not perceptible to the human eye, which has not been possible due to the limitation of the characteristics of the camera used. The transmission speed has been adapted so that the camera is able to detect the variation in intensity of the LED, but the program has been designed in such a way that it is compatible with components that have characteristics that allow a transmission at a higher speed. This fact is quite important since the adaptability of the program implies great versatility when applying it in different environments and at higher speeds.

The VLC transmission system has been designed, being able to perfectly convert the data according to the IEEE 802.15.7 VLC standard, through OOK modulation and the RLL 4B6B coding system. For its implementation, the Raspberry Pi 3 B + development card has been used to make both the emitter and the receiver block. A white light LED diode has been used as the emitter and a Pi Camera has been used as the receiver.

The system has been found to operate at a low transmission speed due to the limitations of the components used, as mentioned above. The percentage of effectiveness is approximately 20%, due to the false positives when identifying the images. This is due to the characteristics of the camera and the low luminosity of the LED diode. If a camera with better characteristics were used, not so much in resolution, but with a much higher frame rate per second, the transmission speed of the LED could be considerably increased until it is imperceptible to the human eye. And an LED lamp instead of the aforementioned LED diode, since its intensity is higher than that of the LED diode, the program would distinguish the intensity variations more easily. As for the color of the LED, it would be indifferent to us, since the program compares images in grayscale.

BIBLIOGRAFÍA

(2020). Retrieved 6 September 2020, from https://catedras.facet.unt.edu.ar/ft/wp-content/uploads/sites/123/2017/03/10-MODULACIONES-ESPECIALES.pdf

(2020). Retrieved 7 September 2020, from https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Análisis: Raspberry Pi 3 Modelo B+. (2020). Retrieved 7 September 2020, from https://hardzone.es/reviews/perifericos/analisis-raspberry-pi-3-modelo-b/

Ghassemlooy, Z., Alves, L., Zvanovec, S., & Khalighi, M. (2017). *Visible light communications*.

Ghassemlooy, Z., Luo, P., & Zvanovec, S. (2016). *Optical Camera Communications*.

IEEE 802.15.7-Compliant Ultra-Low Latency Relaying VLC System for Safety-Critical ITS - IEEE Journals & Magazine. (2020). Retrieved 6 September 2020, from https://ieeexplore.ieee.org/abstract/document/8873657

LiFi – ElectronicosOnline.com Magazine. (2020). Retrieved 6 September 2020, from https://www.electronicosonline.com/etiqueta/lifi/

Quintana Sánchez, C. (2013) Transmisión de datos por medio del sistema VLC.

Raspberry Pi Camera Module - Raspberry Pi Documentation. (2020). Retrieved 7 September 2020, from https://www.raspberrypi.org/documentation/raspbian/applications/camera.md

Rodriguez, P. (2020). Comunicaciones por luz visible: Cuando los bits nos lleguen de las bombillas. Retrieved 6 September 2020, from https://www.xatakahome.com/la-red-local/comunicaciones-por-luz-visible-cuando-los-bits-nos-lleguen-de-las-bombillas

Saha, N., Ifthekhar, M., Le, N., & Jang, Y. (2015). *Survey on optical camera communications: challenges and opportunities*.

ANEXO 1

Código del sistema

1. Código de sincronización.

```python
#!/usr/bin/env python
# coding=utf-8

import subprocess


# Inicia las rutas de los scripts
scripts_paths = ("/home/pi/Desktop/proces/parpadeo.py", "/home/pi/Desktop/proces/view.py")

# Se crea el proceso
procesos = [subprocess.Popen(["python", script]) for script in scripts_paths]

# Se espera a que todos los subprocesos terminen.
for proceso in procesos:
    proceso.wait()
```

2. Código de emisión.

```
1
2
3  import RPi.GPIO as GPIO
4  import time
5  GPIO.setwarnings(False)
6  GPIO.setmode(GPIO.BCM)
7  GPIO.setup(18, GPIO.OUT)
8  GPIO.output(18, True)
9  from subprocess  import call
10 import cv2
11 import sys
12 import numpy as np
13
14 codigo = 0
15
16 mensaje = open ("mensaje.txt","r")        ##Creamos un fichero con formato txt
17 with open("/home/pi/Desktop/proces/mensaje.txt", 'r') as ObjFichero:  ##Abrimos el fichero
18     for line in ObjFichero:         ##Comienza la codificación
19
20         Pos0 = line.find('0000')
21         if Pos0 >= 0:
22             print ("El codigo enviado es 0000")
23             print ("El codigo codificado es 001110")
24             codigo = "001110"
25             break
26
27         Pos1 = line.find('0001')
28         if Pos1 >= 0:
29             print ("El codigo enviado es 0001")
30             print ("El codigo codificado es 001101")
31             codigo = "001101"
32             break
33
34         Pos2 = line.find('0010')
35         if Pos2 >= 0:
36             print ("El codigo enviado es 0010")
37             print ("El codigo codificado es 010011")
38             codigo = "010011"
39             break
40
```

```
40
41          Pos3 = line.find('0011')
42          if Pos3 >= 0:
43              print ("El codigo enviado es 0011")
44              print ("El codigo codificado es 010110")
45              codigo = "010110"
46              break
47
48          Pos4 = line.find('0100')
49          if Pos4 >= 0:
50              print ("El codigo enviado es 0100")
51              print ("El codigo codificado es 010101")
52              codigo = "010101"
53              break
54
55          Pos5 = line.find('0101')
56          if Pos5 >= 0:
57              print ("El codigo enviado es 0101")
58              print ("El codigo codificado es 100011")
59              codigo = "100011"
60              break
61
62          Pos6 = line.find('0110')
63          if Pos6 >= 0:
64              print ("El codigo enviado es 0110")
65              print ("El codigo codificado es 100110")
66              codigo = "100110"
67              break
68
69          Pos7 = line.find('0111')
70          if Pos7 >= 0:
71              print ("El codigo enviado es 0111")
72              print ("El codigo codificado es 100101")
73              codigo = "100101"
74              break
75
```

```
        Pos8 = line.find('1000')
        if Pos8 >= 0:
            print ("El codigo enviado es 1000")
            print ("El codigo codificado es 011001")
            codigo = "011001"
            break

        Pos9 = line.find('1001')
        if Pos9 >= 0:
            print ("El codigo enviado es 1001")
            print ("El codigo codificado es 011010")
            codigo = "011010"
            break

        Pos10 = line.find('1010')
        if Pos10 >= 0:
            print ("El codigo enviado es 1010")
            print ("El codigo codificado es 011100")
            codigo = "011100"
            break

        Pos11 = line.find('1011')
        if Pos11 >= 0:
            print ("El codigo enviado es 1011")
            print ("El codigo codificado es 110001")
            codigo = "110001"
            break

        Pos12 = line.find('1100')
        if Pos12 >= 0:
            print ("El codigo enviado es 1100")
            print ("El codigo codificado es 110010")
            codigo = "110010"
            break
```

```
        Pos13 = line.find('1101')
        if Pos13 >= 0:
            print ("El codigo enviado es 1101")
            print ("El codigo codificado es 101001")
            codigo = "101001"
            break

        Pos14 = line.find('1110')
        if Pos14 >= 0:
            print ("El codigo enviado es 1110")
            print ("El codigo codificado es 101010")
            codigo = "101010"
            break

        Pos15 = line.find('1111')
        if Pos15 >= 0:
            print ("El codigo enviado es 1111")
            print ("El codigo codificado es 101100")
            codigo = "101100"
            break


time.sleep(4)

for i in codigo:        ##Comienza el parpadeo
    if (i == "1"):
       GPIO.output(18, True)
       time.sleep(0.05)
       GPIO.output(18, False)
       time.sleep(0.05)

    else:
        GPIO.output(18, False)
        time.sleep(0.05)

GPIO.output(18, True)
```

3. Código de recepción y conversión.

```python
import time
import picamera
from subprocess  import call
import cv2
import sys
import numpy as np


with picamera.PiCamera() as picam:   ##Abrimos la cámara y se configuran sus características
    picam.rotation = 180
    picam.resolution = (640, 480)
    picam.awb_mode= 'auto'
    picam.exposure_mode = 'auto'
    picam.contrast= -10
    picam.brightness= 30
    picam.ISO = 700
    picam.framerate=21
    picam.start_preview()
    time.sleep(1)

    picam.start_recording('video.h264')   ##Se pone nombre y formato al video
    picam.wait_recording(2)
    picam.stop_recording()
    picam.stop_preview()
    picam.close()

    command = "MP4Box -add video.h264 video.mp4"   ##Se cambia el formato del video de .h264 a .mp4
    call([command], shell=True)
    print("video convertido")


vidcap = cv2.VideoCapture('video.mp4')
success,image = vidcap.read()
count = 0
success = True
```

```
while success:
  cv2.imwrite("frame%d.jpg" % count, image)     ##Fragmenta el video en imágenes
  success,image = vidcap.read()
  print ('Read a new frame: ', success)
  count += 1

count = 0
success = True
binar = open ('binario.txt','w')
while success:

    img1 = cv2.imread("frame0.jpg",0)            ##Comienza la comparación de imágenes
    img2 = cv2.imread("frame%d.jpg" % count,0)
    if img2 is None:

        binar.close()
        binar = open ("binario.txt","r")
        with open("//home/pi/Desktop/proces/binario.txt", 'r') as ObjFichero:  ##Abre el fichero .txt
            for line in ObjFichero:             ##Comienza la decodificación
                Pos0 = line.find('001110')
                if Pos0 >= 0:
                    print ("El codigo codificado es 001110")
                    print ("El codigo enviado es 0000")
                    break
                Pos1 = line.find('001101')
                if Pos1 >= 0:
                    print ("El codigo codificado es 001101")
                    print ("El codigo enviado es 0001")
                    break
```

```
65                  Pos2 = line.find('010011')
66                  if Pos2 >= 0:
67                      print ("El codigo codificado es 010011")
68                      print ("El codigo enviado es 0010")
69                      break
70                  Pos3 = line.find('010110')
71                  if Pos3 >= 0:
72                      print ("El codigo codificado es 010110")
73                      print ("El codigo enviado es 0011")
74                      break
75                  Pos4 = line.find('010101')
76                  if Pos4 >= 0:
77                      print ("El codigo codificado es 010101")
78                      print ("El codigo enviado es 0100")
79                      print ("0100")
80                      break
81                  Pos5 = line.find('100011')
82                  if Pos5 >= 0:
83                      print ("El codigo codificado es 100011")
84                      print ("El codigo enviado es 0101")
85                      break
86                  Pos6 = line.find('100110')
87                  if Pos6 >= 0:
88                      print ("El codigo codificado es 100110")
89                      print ("El codigo enviado es 0110")
90                      break
```

```
91                  Pos7 = line.find('100101')
92                  if Pos7 >= 0:
93                      print ("El codigo codificado es 100101")
94                      print ("El codigo enviado es 0111")
95                      break
96                  Pos8 = line.find('011001')
97                  if Pos8 >= 0:
98                      print ("El codigo codificado es 011001")
99                      print ("El codigo enviado es 1000")
100                     break
101                 Pos9 = line.find('011010')
102                 if Pos9 >= 0:
103                     print ("El codigo codificado es 011010")
104                     print ("El codigo enviado es 1001")
105                     break
106                 Pos10 = line.find('011100')
107                 if Pos10 >= 0:
108                     print ("El codigo codificado es 011100")
109                     print ("El codigo enviado es 1010")
110                     break
111                 Pos11 = line.find('110001')
112                 if Pos11 >= 0:
113                     print ("El codigo codificado es 110001")
114                     print ("El codigo enviado es 1011")
115                     break
116                 Pos12 = line.find('110010')
117                 if Pos12 >= 0:
118                     print ("El codigo codificado es 110010")
119                     print ("El codigo enviado es 1100")
120                     break
```

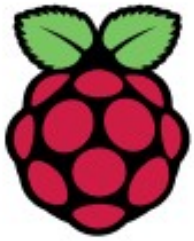
```
                Pos13 = line.find('101001')
                if Pos13 >= 0:
                    print ("El codigo codificado es 101001")
                    print ("El codigo enviado es 1101")
                    break
                Pos14 = line.find('101010')
                if Pos14 >= 0:
                    print ("El codigo codificado es 101010")
                    print ("El codigo enviado es 1110")
                    break
                Pos15 = line.find('101100')
                if Pos15 >= 0:
                    print ("El codigo codificado es 101100")
                    print ("El codigo enviado es 1111")
                    break

        success = False
        sys.exit("Cerrando el sistema")

    def compara (img1,img2):
        difference = cv2.subtract(img1, img2) ##Comienza la comparación de pixeles
        b, g, r = cv2.split(difference)
        print(cv2.countNonZero(b))
        if (cv2.countNonZero(b) == 0 and cv2.countNonZero(g) == 0 and cv2.countNonZero(r) == 0):
            print('Las imagenes son completamente iguales')
        else:
            print('Las imagenes no son iguales')


    shift = cv2.xfeatures2d.SIFT_create()
    kp_1, desc_1 = shift.detectAndCompute(img1, None)    ##Se obtienen los puntos clave y los descriptores
    kp_2, desc_2 = shift.detectAndCompute(img2, None)

```

```
154
155     print('frame0.jpg ')                          ##Muestra las fotos que se están comparando
156     print('frame%d.jpg ' % count)
157     print("Keypoints 1st image", str(len(kp_1)))  ##Muestra los puntos claves y los descriptores
158     print("Keypoints 2st image", str(len(kp_2)))
159
160     index_params = dict(algorithm=0, trees=5)
161     search_params = dict()
162
163     flann = cv2.FlannBasedMatcher(index_params, search_params)   ##Busca coincidencias entre descriptores
164     matches = flann.knnMatch(desc_1, desc_2, k=2)         ## Se almacenan las coincidencias en una matriz
165
166     good_points = []
167     for m, n in matches:                    ##Se seleccionan solo las buenas coincidencias
168         if m.distance < 0.2*n.distance:
169             good_points.append(m)
170
171     number_keypoints = 0                   ##Calcula el porcentaje de similitud
172     if (len(kp_1) <= len(kp_2)):
173         number_keypoints = float(len(kp_1))
174     else:
175         number_keypoints = float(len(kp_2))
176     porce = 0
177     x = float(len(good_points))
178     y = float(number_keypoints)
179     porce = float(((x / y) * 100.0))
180     print("GOOD matches",x)
181     print("Porcentaje de similitud",float(porce),'%')
182
183
184     if (porce <= 16):
185         binar.write("0")
186     else:
187         binar.write("1")
188
189     count += 1
190
191
192 cv2.destroyAllWindows()
```

ANEXO 2

Hoja de características
Raspberry Pi 3B+

DATASHEET



Raspberry Pi Compute Module 3+

Raspberry Pi Compute Module 3+ Lite

Release 1, January 2019

Table 1: Release History

Release	Date	Description
1	28/01/2019	First release

The latest release of this document can be found at `https://www.raspberrypi.org`

Contents

List of Figures

List of Tables

1 Introduction

The Raspberry Pi Compute Module 3+ (CM3+) is a range of DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (on non-Lite variants) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these modules have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards, opening up more options for the designer.

The CM3+ contains a BCM2837B0 processor (as used on the Raspberry Pi 3B+), 1Gbyte LPDDR2 RAM and eMMC Flash. The CM3+ is currently available in 4 variants, CM3+/8GB, CM3+/16GB, CM3+/32GB and CM3+ Lite, which have 8, 16 and 32 Gigabytes of eMMC Flash, or no eMMC Flash, respectively.

The CM3+ Lite product is the same as CM3+ except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device.

Note that the CM3+ is electrically identical and, with the exception of higher CPU z-height, physically identical to the legacy CM3 products.

CM3+ modules require a software/firmware image dated November 2018 or newer to function correctly.

2 Features

2.1 Hardware

- Low cost
- Low power
- High availability
- High reliability
  - Tested over millions of Raspberry Pis Produced to date
  - Module IO pins have 15 micro-inch hard gold plating over 2.5 micron Nickel

2.2 Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

2.3 Software

- ARMv8 Instruction Set
- Mature and stable Linux software stack
  - Latest Linux Kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Full availability of GPU functions using standard APIs

3 Block Diagram

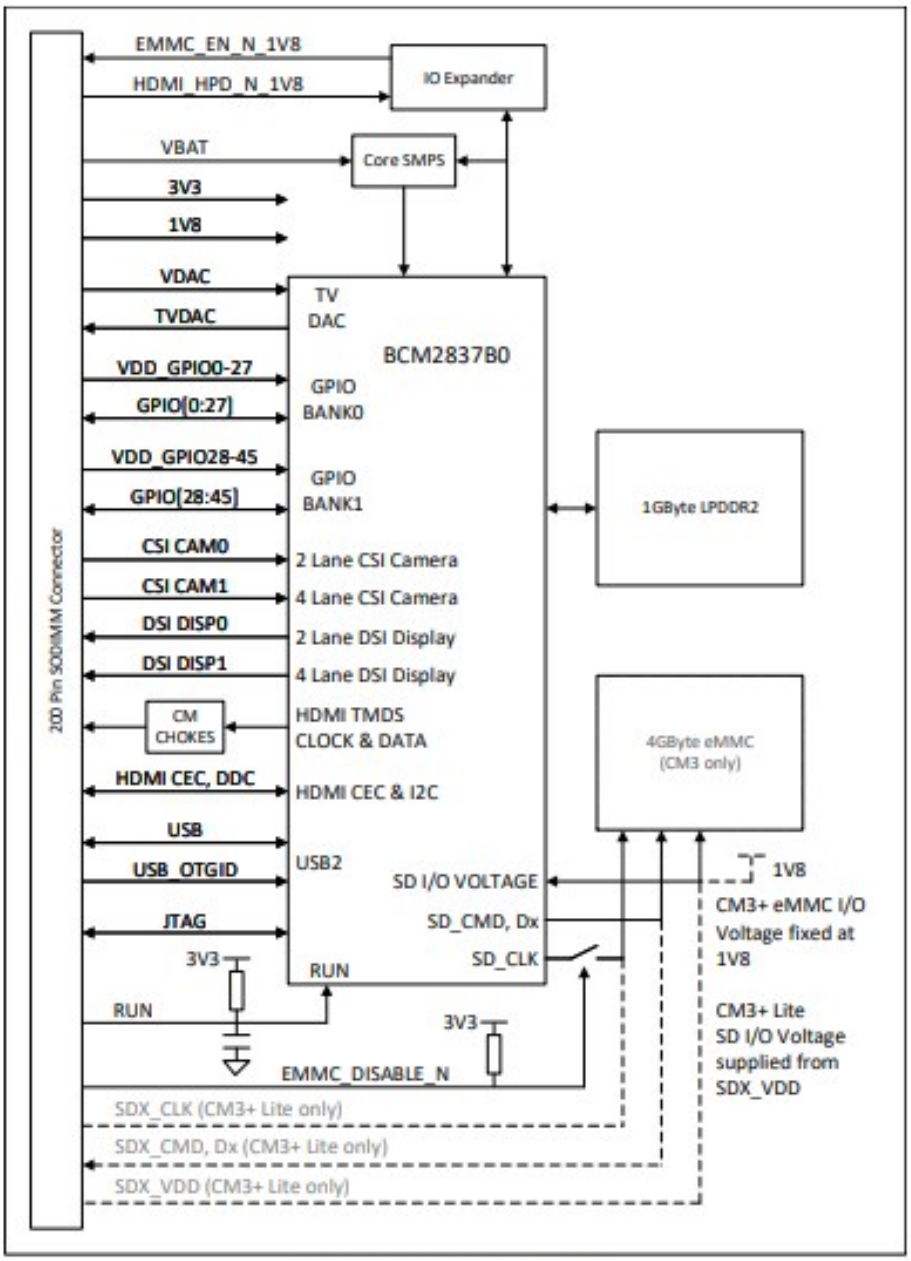


Figure 1: CM3+ Block Diagram

4 Mechanical Specification

The CM3+ modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules and therefore should work with the many DDR2 SODIMM sockets available on the market. **(Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.)**

The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture.

The maximum component height on the underside of the Compute Module is 1.2mm.

The maximum component height on the top side of the Compute Module is 2.5mm.

The Compute Module PCB thickness is 1.0mm +/- 0.1mm.

Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified.

Figure 2 gives the CM3+ mechanical dimensions.

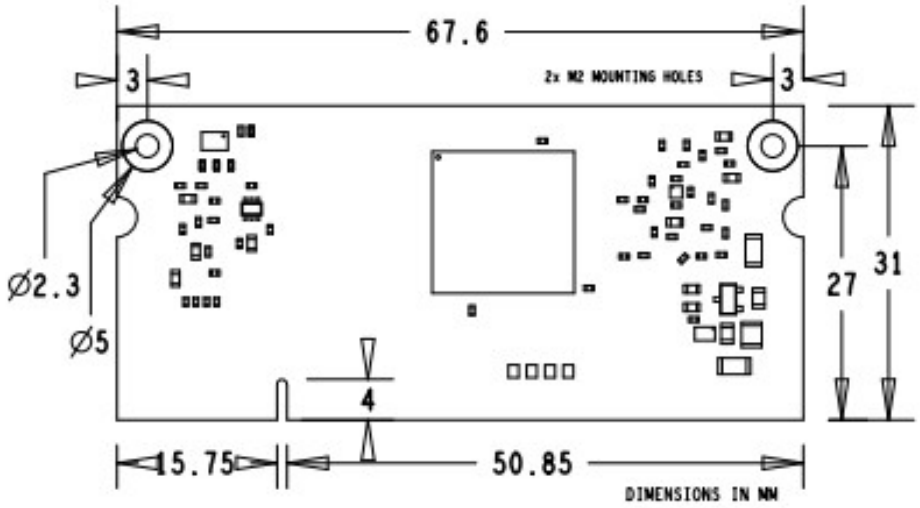


Figure 2: CM3+ Mechanical Dimensions

5 Pin Assignments

Dell+	Dell+ Use	Pin	Pin	Dell+	Dell+ Use
GND		1	2	EMMC_DISABLE_N	
GPIO0		3	4	NC	SDX_VDD
GPIO1		5	6	NC	SDX_VDD
GND		7	8	GND	
GPIO2		9	10	NC	SDX_CLK
GPIO3		11	12	NC	SDX_CMD
GND		13	14	GND	
GPIO4		15	16	NC	SDX_D0
GPIO5		17	18	NC	SDX_D1
GND		19	20	GND	
GPIO6		21	22	NC	SDX_D2
GPIO7		23	24	NC	SDX_D3
GND		25	26	GND	
GPIO8		27	28	GPIO28	
GPIO9		29	30	GPIO29	
GND		31	32	GND	
GPIO10		33	34	GPIO30	
GPIO11		35	36	GPIO31	
GND		37	38	GND	
GPIO0_1T_VDD		39	40	GPIO0_1T_VDD	
GPIO					
GPIO28_IO_VDD		41	42	GPIO28_IO_VDD	
GND		43	44	GND	
GPIO12		45	46	GPIO32	
GPIO13		47	48	GPIO33	
GND		49	50	GND	
GPIO14		51	52	GPIO34	
GPIO15		53	54	GPIO35	
GND		55	56	GND	
GPIO16		57	58	GPIO36	
GPIO17		59	60	GPIO37	
GND		61	62	GND	
GPIO18		63	64	GPIO38	
GPIO19		65	66	GPIO39	
GND		67	68	GND	
GPIO20		69	70	GPIO40	
GPIO21		71	72	GPIO41	
GND		73	74	GND	
GPIO22		75	76	GPIO42	
GPIO23		77	78	GPIO43	
GND		79	80	GND	
GPIO24		81	82	GPIO44	
GPIO25		83	84	GPIO45	
GND		85	86	GND	
GPIO26		87	88	HDMI_HPD_N_1V8	
GPIO27		89	90	EMMC_EN_N_EVB	
GND		91	92	GND	
DSI0_DN1		93	94	DSI1_DP0	
DSI0_DP1		95	96	DSI1_DN0	
GND		97	98	GND	
DSI0_DN0		99	100	DSI1_CP	
DSI0_DP0		101	102	DSI1_CN	
GND		103	104	GND	
DSI0_CN		105	106	DSI1_DP3	
DSI0_CP		107	108	DSI1_DN3	
GND		109	110	GND	
HDMI_CAR_N		111	112	DSI1_DP2	
HDMI_CAR_P		113	114	DSI1_DN2	
GND		115	116	GND	
HDMI_D0_N		117	118	DSI1_DP1	
HDMI_D0_P		119	120	DSI1_DN1	
GND		121	122	GND	
HDMI_D1_N		123	124	NC	
HDMI_D1_P		125	126	NC	
GND		127	128	NC	
HDMI_D2_N		129	130	NC	
HDMI_D2_P		131	132	NC	
GND		133	134	GND	
CAM0_DP3		135	136	CAM0_DP0	
CAM0_DN3		137	138	CAM0_DN0	
GND		139	140	GND	
CAM0_DP2		141	142	CAM0_CP	
CAM0_DN2		143	144	CAM0_CN	
GND		145	146	GND	
CAM0_CP		147	148	CAM0_DP1	
CAM0_CN		149	150	CAM0_DN1	
GND		151	152	GND	
CAM1_DP3		153	154	NC	
CAM1_DN3		155	156	NC	
GND		157	158	NC	
CAM1_DP0		159	160	NC	
CAM1_DN0		161	162	NC	
GND		163	164	GND	
USB_DP		165	166	TVDAC	
USB_DM		167	168	USB_OTGID	
GND		169	170	GND	
HDMI_CEC		171	172	VC_TRST_N	
HDMI_SCL		173	174	VC_TDI	
HDMI_SDA		175	176	VC_TMS	
Rsvd		177	178	VC_TDO	
DO_CORE1DO=NOT CONNECT		179	180	VC_TCK	
GND		181	182	GND	
VSB		183	184	VSB	
VSB		185	186	VSB	
GND		187	188	GND	
VBAT		189	190	VBAT	
3V3		191	192	3V3	
3V3		193	194	3V3	
GND		195	196	GND	
VBAT		197	198	VBAT	
VBAT		199	200	VBAT	

Table 2: Compute Module 3+ SODIMM Connector Pinout

Table 2 gives the Compute Module 3+ pinout and Table 3 gives the pin functions.

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,e]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

[a] The PDN column indicates power-down state (when RUN pin LOW)
[b] Must be driven by an open-collector driver
[c] GPIO have software enabled pulls which keep state over power-down
[d] Only available on Lite variants
[e] The CM will always try to boot from this interface first
[f] Requires external pull-up resistor to 5V as per HDMI spec

Table 3: Pin Functions

6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
		VDD_IO = 3.3V	-	-	0.9	V
V_{IH}	Input high voltage[a]	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
		VDD_IO = 3.3V	1.6	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	µA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
		VDD_IO = 3.3V, IOL = -2mA	-	-	0.14	V
V_{OH}	Output high voltage[b]	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
		VDD_IO = 3.3V, IOH = 2mA	3.0	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
		VDD_IO = 3.3V, VO = 0.4V	18	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
		VDD_IO = 3.3V, VO = 2.3V	17	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 5: DC Characteristics

AC Characteristics are defined in Table 6 and Fig. 3.

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	1.6	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	1.7	-	ns
GPCLK	t_{JOSC}	Oscillator-derived GPCLK cycle-cycle jitter (RMS)	-	-	20	ps
GPCLK	t_{JPLL}	PLL-derived GPCLK cycle-cycle jitter (RMS)	-	-	48	ps

[a] Default drive strength, CL = 5pF, VDD_IOx = 3.3V

Table 6: Digital I/O Pin AC Characteristics

Figure 3: Digital IO Characteristics

7 Power Supplies

The Compute Module 3+ has six separate supplies that must be present and powered at all times; you cannot leave any of them unpowered, even if a specific interface or GPIO bank is unused. The six supplies are as follows:

1. VBAT is used to power the BCM2837 processor core. It feeds the SMPS that generates the chip core voltage.
2. 3V3 powers various BCM2837 PHYs, IO and the eMMC Flash.
3. 1V8 powers various BCM2837 PHYs, IO and SDRAM.
4. VDAC powers the composite (TV-out) DAC.
5. GPIO0-27_VREF powers the GPIO 0-27 IO bank.
6. GPIO28-45_VREF powers the GPIO 28-45 IO bank.

Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

[a] Requires a clean 2.5-2.8V supply if TV DAC is used, else connect to 3V3

Table 7: Power Supply Operating Ranges

7.1 Supply Sequencing

Supplies should be staggered so that the highest voltage comes up first, then the remaining voltages in descending order. This is to avoid forward biasing internal (on-chip) diodes between supplies, and causing latch-up. Alternatively supplies can be synchronised to come up at exactly the same time as long as at no point a lower voltage supply rail voltage exceeds a higher voltage supply rail voltage.

7.2 Power Requirements

Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused, it is usually in a low power state or completely turned off. For instance, if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. This is also the case for camera and display interfaces, HDMI, USB interfaces, video encoders and decoders, and so on.

Powerchain design is critical for stable and reliable operation of the Compute Module 3+. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

Table 8 specifies the recommended minimum power supply outputs required to power the Compute Module 3+.

Supply	Minimum Requirement	Unit
VBAT (CM1)	2000[a]	mW
VBAT (CM3,3L)	3500[a]	mW
3V3	250	mA
1V8	250	mA
VDAC	25	mA
GPIO0-27_VDD	50[b]	mA
GPIO28-45_VDD	50[b]	mA
SDX_VDD	50[b]	mA

[a] Recommended minimum. Actual power drawn is very dependent on use-case
[b] Each GPIO can supply up to 16mA, aggregate current per bank must not exceed 50mA

Table 8: Mimimum Power Supply Requirements

8 Booting

The eMMC Flash device on CM3+ is directly connected to the primary BCM2837 SD/eMMC interface. These connections are not accessible on the module pins. On CM3+ Lite this SD interface is available on the SDX_ pins.

When initially powered on, or after the RUN pin has been held low and then released, the BCM2837 will try to access the primary SD/eMMC interface. It will then look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the SD/eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB; in other words, its USB port is in slave mode waiting to accept boot code from a suitable host.

A USB boot tool is available on Github which allows a host PC running Linux to write the BCM2837 boot code over USB to the module. That boot code then runs and provides access to the SD/eMMC as a USB mass storage device, which can then be read and written using the host PC. Note that a Raspberry Pi can be used as the host machine. For those using Windows a precompiled and packeged tool is available. For more information see here.

The Compute Module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the SD/eMMC interface (by physically disconnecting the SD_CMD pin), forcing BCM2837 to boot from USB. Note that when the eMMC is disabled in this way, it takes a couple of seconds from powering up for the processor to stop attempting to talk to the SD/eMMC device and fall back to booting from USB.

Note that once booted over USB, BCM2837 needs to re-enable the SD/eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the EMMC_EN_N_1V8 pin LOW, which at boot is initially an input with a pull up to 1V8. If an end user wishes to add the ability to access the SD/eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and SD/eMMC must be used; that is, EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET, with the gate controlled by EMMC_EN_N_1V8. **Ensure you use MOSFETs suitable for switching at 1.8V (i.e. use a device with gate threshold voltage, Vt, suitable for 1.8V switching).**

9 Peripherals

9.1 GPIO

BCM2837 has in total 54 GPIO lines in 3 separate voltage banks. All GPIO pins have at least two alternative functions within the SoC. When not used for the alternate peripheral function, each GPIO pin may be set as an input (optionally as an interrupt) or an output. The alternate functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage.

GPIO bank2 is used on the module to connect to the eMMC device and for an on-board I2C bus (to talk to the core SMPS and control the special function pins). On CM3+ Lite most of bank2 is exposed to allow a user to connect their choice of SD card or eMMC device (if required).

Bank0 and 1 GPIOs are available for general use. GPIO0 to GPIO27 are bank0 and GPIO28-45 make up bank1. GPIO0-27_VDD is the power supply for bank0 and GPIO28-45_VDD is the power supply for bank1. SDX_VDD is the supply for bank2 on CM3+ Lite. These supplies can be in the range 1.8V-3.3V (see Table 7) and are not optional; each bank must be powered, even when none of the GPIOs for that bank are used.

Note that the HDMI_HPD_N_1V8 and EMMC_EN_N_1V8 pins are 1.8V IO and are used for special functions (HDMI hot plug detect and boot control respectively). Please do not use these pins for any other purpose, as the software for the module will always expect these pins to have these special functions. If they are unused please leave them unconnected.

All GPIOs except GPIO28, 29, 44 and 45 have weak in-pad pull-ups or pull-downs enabled when the device is powered on. It is recommended to add off-chip pulls to GPIO28, 29, 44 and 45 to make sure they never float during power on and initial boot.

9.1.1 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	-	-	-
1	High	SCL0	SA4	DE	-	-	-
2	High	SDA1	SA3	LCD_VSYNC	-	-	-
3	High	SCL1	SA2	LCD_HSYNC	-	-	-
4	High	GPCLK0	SA1	DPI_D0	-	-	ARM_TDI
5	High	GPCLK1	SA0	DPI_D1	-	-	ARM_TDO
6	High	GPCLK2	SOE_N	DPI_D2	-	-	ARM_RTCK
7	High	SPI0_CE1_N	SWE_N	DPI_D3	-	-	-
8	High	SPI0_CE0_N	SD0	DPI_D4	-	-	-
9	Low	SPI0_MISO	SD1	DPI_D5	-	-	-
10	Low	SPI0_MOSI	SD2	DPI_D6	-	-	-
11	Low	SPI0_SCLK	SD3	DPI_D7	-	-	-
12	Low	PWM0	SD4	DPI_D8	-	-	ARM_TMS
13	Low	PWM1	SD5	DPI_D9	-	-	ARM_TCK
14	Low	TXD0	SD6	DPI_D10	-	-	TXD1
15	Low	RXD0	SD7	DPI_D11	-	-	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	-	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	-	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	-	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	-	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	-
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	-
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	-
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	-
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	-
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	-

Table 9: GPIO Bank0 Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
28	None	SDA0	SA5	PCM_CLK	FL0	-	-
29	None	SCL0	SA4	PCM_FS	FL1	-	-
30	Low	TE0	SA3	PCM_DIN	CTS0	-	CTS1
31	Low	FL0	SA2	PCM_DOUT	RTS0	-	RTS1
32	Low	GPCLK0	SA1	RING_OCLK	TXD0	-	TXD1
33	Low	FL1	SA0	TE1	RXD0	-	RXD1
34	High	GPCLK0	SOE_N	TE2	SD1_CLK	-	-
35	High	SPI0_CE1_N	SWE_N	-	SD1_CMD	-	-
36	High	SPI0_CE0_N	SD0	TXD0	SD1_DAT0	-	-
37	Low	SPI0_MISO	SD1	RXD0	SD1_DAT1	-	-
38	Low	SPI0_MOSI	SD2	RTS0	SD1_DAT2	-	-
39	Low	SPI0_SCLK	SD3	CTS0	SD1_DAT3	-	-
40	Low	PWM0	SD4	-	SD1_DAT4	SPI2_MISO	TXD1
41	Low	PWM1	SD5	TE0	SD1_DAT5	SPI2_MOSI	RXD1
42	Low	GPCLK1	SD6	TE1	SD1_DAT6	SPI2_SCLK	RTS1
43	Low	GPCLK2	SD7	TE2	SD1_DAT7	SPI2_CE0_N	CTS1
44	None	GPCLK1	SDA0	SDA1	TE0	SPI2_CE1_N	-
45	None	PWM1	SCL0	SCL1	TE1	SPI2_CE2_N	-

Table 10: GPIO Bank1 Alternate Functions

Table 9 and Table 10 detail the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the Broadcom Peripherals Specification document and have Linux drivers available.

9.1.2 Secondary Memory Interface (SMI)

The SMI peripheral is an asynchronous NAND type bus supporting Intel mode80 type transfers at 8 or 16 bit widths and available in the ALT1 positions on GPIO banks 0 and 1 (see Table 9 and Table 10). It is not publicly documented in the Broadcom Peripherals Specification but a Linux driver is available in the Raspberry Pi Github Linux repository (bcm2835_smi.c in linux/drivers/misc).

9.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available on bank 0 GPIOs. This up-to-24-bit parallel interface can support a secondary display. Again this interface is not documented in the Broadcom Peripherals Specification but documentation can be found here.

9.1.4 SD/SDIO Interface

The BCM283x supports two SD card interfaces, SD0 and SD1.

The first (SD0) is a proprietary Broadcom controller that does not support SDIO and is the primary interface used to boot and talk to the eMMC or SDX_x signals.

The second interface (SD1) is standards compliant and can interface to SD, SDIO and eMMC devices; for example on a Raspberry Pi 3 B+ it is used to talk to the on-board CYW43455 WiFi device in SDIO mode.

Both interfaces can support speeds up to 50MHz single ended (SD High Speed Mode).

9.2 CSI (MIPI Serial Camera)

Currently the CSI interface is not openly documented and only CSI camera sensors supported by the official Raspberry Pi firmware will work with this interface. Supported sensors are the OmniVision OV5647 and Sony IMX219.

It is recommended to attach other cameras via USB.

9.3 DSI (MIPI Serial Display)

Currently the DSI interface is not openly documented and only DSI displays supported by the official Raspberry Pi firmware will work with this interface.

Displays can also be added via the parallel DPI interface which is available as a GPIO alternate function - see Table 9 and Section 9.1.3

9.4 USB

The BCM2837 USB port is On-The-Go (OTG) capable. If using either as a fixed slave or fixed master, please tie the USB_OTGID pin to ground.

The USB port (Pins USB_DP and USB_DM) must be routed as 90 ohm differential PCB traces.

Note that the port is capable of being used as a true OTG port however there is no official documentation. Some users have had success making this work.

9.5 HDMI

BCM283x supports HDMI V1.3a.

It is recommended that users follow a similar arrangement to the Compute Module IO Board circuitry for HDMI output.

The HDMI CK_P/N (clock) and D0-D2_P/N (data) pins must each be routed as matched length 100 ohm differential PCB traces. It is also important to make sure that each differential pair is closely phase matched. Finally, keep HDMI traces well away from other noise sources and as short as possible.

Failure to observe these design rules is likely to result in EMC failure.

9.6 Composite (TV Out)

The TVDAC pin can be used to output composite video (PAL or NTSC). Please route this signal away from noise sources and use a 75 ohm PCB trace.

Note that the TV DAC is powered from the VDAC supply which must be a clean supply of 2.5-2.8V. It is recommended users generate this supply from 3V3 using a low noise LDO.

If the TVDAC output is not used VDAC can be connected to 3V3, but it must be powered even if the TV-out functionality is unused.

10 Thermals

The BCM2837 SoC employs DVFS (Dynamic Voltage and Frequency Scaling) on the core voltage. When the processor is idle (low CPU utilisation), it will reduce the core frequency and voltage to reduce current draw and heat output. When the core utilisation exceeds a certain threshold the core votlage is increased and the core frequency is boosted to the maximum working frerquency of 1.2GHz. The voltage and frequency are throttled back when the CPU load reduces back to an 'idle' level OR when the silicon temperature as mesured by the on-chip temperature sensor exceeds 80C (thermal throttling).

A designer must pay careful attention to the thermal design of products using the CM3+ so that performance is not artificially curtailed due to the processor thermal throttling, as the Quad ARM complex in the BCM2837 can generate significant heat output under load.

10.1 Temperature Range

The operating temperature range of the module is set by the lowest maximum and highest minimum of any of the components used.

The eMMC and LPDDR2 have the narrowest range, these are rated for -25 to +80 degrees Celsius. Therefore the nominal range for the CM3+ and CM3+ Lite is -25C to +80C.

However, this range is the maximum for the silicon die; therefore, users would have to take into account the heat generated when in use and make sure this does not cause the temperature to exceed 80 degrees Celsius.

11 Availability

Raspberry Pi guarantee availability of CM3+ and CM3+ Lite until at least January 2026.

12 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

ANEXO 3

Hoja de características
Pi Camera V2

The Raspberry Pi Camera Modules are official products from the Raspberry Pi Foundation. The original 5-megapixel model was released in 2013, and an 8-megapixel Camera Module v2 was released in 2016. For both iterations, there are visible light and infrared versions. A 12-megapixel High Quality Camera was released in 2020. There is no infrared version of the HQ Camera, however the IR Filter can be removed if required.

Hardware specification.

	Camera Module v1	Camera Module v2	HQ Camera
Net price	$25	$25	$50
Size	Around 25 × 24 × 9 mm		38 x 38 x 18.4mm (excluding lens)
Weight	3g	3g	
Still resolution	5 Megapixels	8 Megapixels	12.3 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90	1080p30, 720p60 and 640 × 480p60/90	1080p30, 720p60 and 640 × 480p60/90
Linux integration	V4L2 driver available	V4L2 driver available	V4L2 driver available
C programming API	OpenMAX IL and others available	OpenMAX IL and others available	
Sensor	OmniVision OV5647	Sony IMX219	Sony IMX477
Sensor resolution	2592 × 1944 pixels	3280 × 2464 pixels	4056 x 3040 pixels
Sensor image area	3.76 × 2.74 mm	3.68 x 2.76 mm (4.6 mm diagonal)	6.287mm x 4.712 mm (7.9mm diagonal)
Pixel size	1.4 µm × 1.4 µm	1.12 µm x 1.12 µm	1.55 µm x 1.55 µm
Optical size	1/4"	1/4"	
Full-frame SLR lens equivalent	35 mm		
S/N ratio	36 dB		
Dynamic range	67 dB @ 8x gain		
Sensitivity	680 mV/lux-sec		
Dark current	16 mV/sec @ 60 C		
Well capacity	4.3 Ke-		
Fixed focus	1 m to infinity		N/A
Focal length	3.60 mm +/- 0.01	3.04 mm	Depends on lens
Horizontal field of view	53.50 +/- 0.13 degrees	62.2 degrees	Depends on lens
Vertical field of view	41.41 +/- 0.11 degrees	48.8 degrees	Depends on lens
Focal ratio (F-	2.9	2.0	Depends on lens

	Camera Module v1	Camera Module v2	HQ Camera
Stop)			

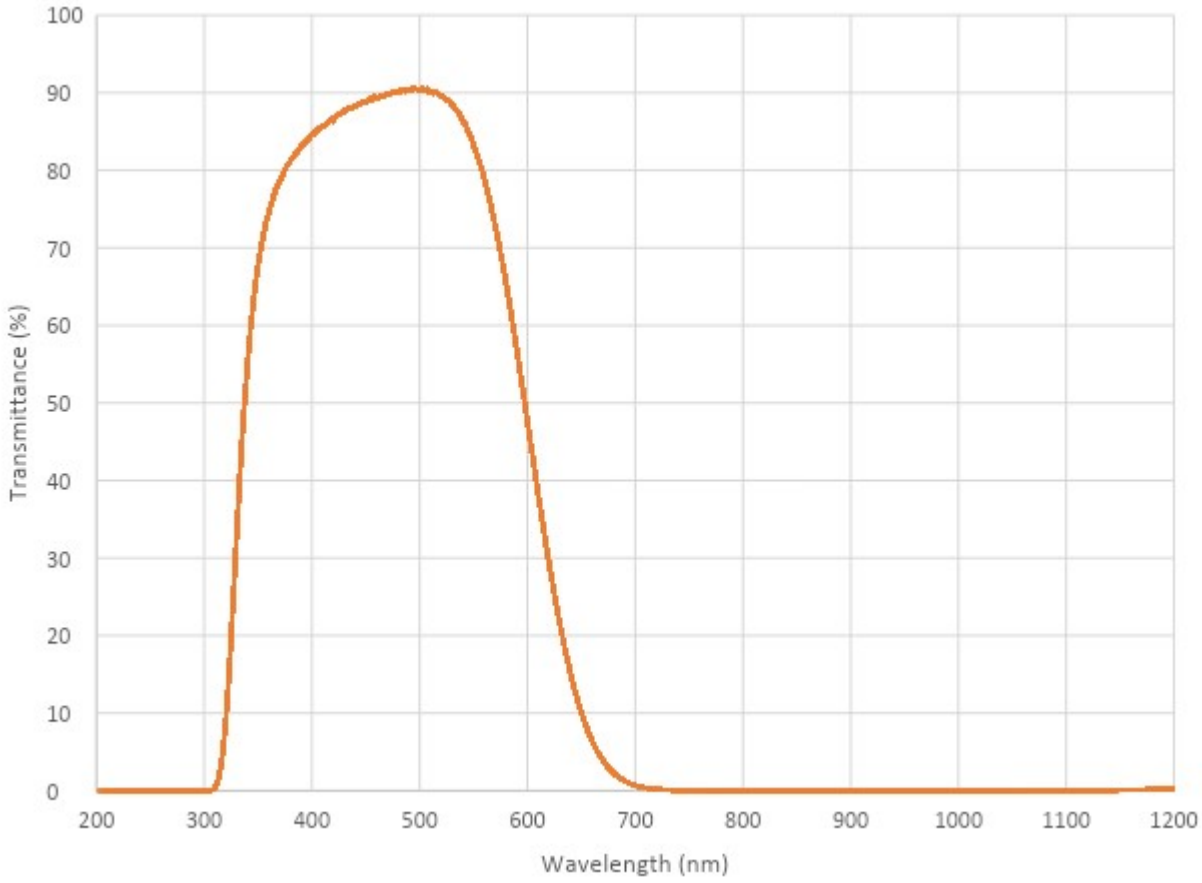
Hardware features.

Available	Implemented
Chief ray angle correction	Yes
Global and rolling shutter	Rolling shutter
Automatic exposure control (AEC)	No - done by ISP instead
Automatic white balance (AWB)	No - done by ISP instead
Automatic black level calibration (ABLC)	No - done by ISP instead
Automatic 50/60 Hz luminance detection	No - done by ISP instead
Frame rate up to 120 fps	Max 90fps. Limitations on frame size for the higher frame rates (VGA only for above 47fps)
AEC/AGC 16-zone size/position/weight control	No - done by ISP instead
Mirror and flip	Yes
Cropping	No - done by ISP instead (except 1080p mode)
Lens correction	No - done by ISP instead
Defective pixel cancelling	No - done by ISP instead
10-bit RAW RGB data	Yes - format conversions available via GPU
Support for LED and flash strobe mode	LED flash
Support for internal and external frame synchronisation for frame exposure mode	No
Support for 2 × 2 binning for better SNR in low light conditions	Anything output res below 1296 x 976 will use the 2 x 2 binned mode
Support for horizontal and vertical sub-sampling	Yes, via binning and skipping
On-chip phase lock loop (PLL)	Yes
Standard serial SCCB interface	Yes
Digital video port (DVP) parallel output interface	No
MIPI interface (two lanes)	Yes
32 bytes of embedded one-time programmable (OTP) memory	No
Embedded 1.5V regulator for core power	Yes

Software features.

Picture formats	JPEG (accelerated), JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888
Video formats	raw h.264 (accelerated)
Effects	negative, solarise, posterize, whiteboard, blackboard, sketch, denoise, emboss, oilpaint, hatch, gpen, pastel, watercolour, film, blur, saturation
Exposure modes	auto, night, nightpreview, backlight, spotlight, sports, snow, beach, verylong, fixedfps, antishake, fireworks
Metering modes	average, spot, backlit, matrix
Automatic white balance modes	off, auto, sun, cloud, shade, tungsten, fluorescent, incandescent, flash, horizon
Triggers	Keypress, UNIX signal, timeout
Extra modes	demo, burst/timelapse, circular buffer, video with motion vectors, segmented video, live preview on 3D models

HQ Camera IR Filter transmission information.
The HQ Camera uses a Hoya CM500 infrared filter. Its transmission characteristics are as represented in the following graph.


100
90
80
70
60
50
40
30
20
10
0
Transmittance (%)
200
300
400
500
600
700
800
900
1000
1100
1200
Wavelength (nm)