Visible light communication for Internet of Things

Trabajo final de grado

Escuela Técnica de Ingeniería de Telecomunicación de Barcelona

Universidad Politécnica de Cataluña

Por

Juan Carlos Costa Mari

Tutor: Josep Paradells Aspas

Barcelona, Mayo 2018

Abstract

This project consists of designing and implementing a VLC system on raspberry and arduino platforms, low cost components. The light of the Raspberry Pi sends data to all devices, while the Arduinos can send data to the camera. That is, the camera can receive information from many devices in parallel.

In this work, we have worked on the uplink of a full duplex VLC system. It is a prototype of wireless communication through visible light using a LED as a transmitting device and a digital camera as a receiver. It is designed to work indoors. The communication signal is transmitted as a sequence of ones and zeros that turn on or off the light source. A camera receives the signal when observing the presence or not of the point of light and converting it again in a binary signal. Image processing techniques are applied to calculate the signal received from the images.

Resum

Aquest projecte consisteix en dissenyar i implementar un sistema VLC a les plataformes de raspberry i arduino, components de baix cost. La llum de la Raspberry Pi envia dades a tots els dispositius, mentre que els Arduinos poden enviar dades a la càmera. És a dir, la càmera pot rebre informació de molts dispositius en paral·lel.

En aquest treball, s'ha treballat en el uplink d'un sistema VLC full duplex. És un prototip de comunicació sense fils a través de la llum visible utilitzant un led com a dispositiu transmissor i una càmera digital com a receptor. Està dissenyat per treballar en interiors. El senyal de comunicació es transmet com una seqüència d'uns i zeros que encenen o apaguen la font de llum. Una càmera rep el senyal en observar la presència o no del punt de llum i convertir-la novament en un senyal binari. S'apliquen tècniques de processament d'imatges per calcular el senyal rebut de les imatges.

Resumen

Este proyecto consiste en diseñar e implementar un sistema VLC en las plataformas de raspberry y arduino, componentes de bajo coste. La luz de la Raspberry Pi envía datos a todos los dispositivos, mientras que los Arduinos pueden enviar datos a la cámara. Es decir, la camara puede recibir información de muchos dispositivos en paralelo.

En este trabajo, se ha trabajado en el uplink de un sistema VLC full duplex. Es un prototipo de comunicación inalámbrica a través de la luz visible utilizando un led como dispositivo transmisor y una cámara digital como receptor. Está diseñado para trabajar en interiores. La señal de comunicación se transmite como una secuencia de unos y ceros que encienden o apagan la fuente de luz. Una cámara recibe la señal al observar la presencia o no del punto de luz y convertirla nuevamente en una señal binaria. Se aplican técnicas de procesamiento de imágenes para calcular la señal recibida de las imágenes.

A mi familia, gracias a su esfuerzo y ánimos.

A todos mis amigos y compañeros, con los cuales he ido creciendo año tras año.

Agradecimientos

Principalmente doy las gracias a mi tutor Josep Paradells Aspas que me ha ayudado durante el proyecto cuando lo he necesitado. Además, también me gustaría agradecer a las personas que han pasado alguna vez por el laboratorio y me han resuelto alguna duda puntual.

Historial de revisión

Revisión	Fecha	Objetivo
0	01/04/2018	Creación del documento
1	28/04/2018	Revisión del documento
2	09/05/2018	Finalización documento

Lista de distribución del documento

Nombre	E-mail
Juan Carlos Costa Mari	juan.carlos.costa18@gmail.com
Josep Paradells Aspas	josep.paradells@entel.upc.edu

Escrito por:		Revisado y aprobado por:	
Fecha	09/05/2018	Fecha	10/05/2018
Nombre	Juan Carlos Costa Mari	Nombre	Josep Paradells Aspas
Posición	Autor	Posición	Supervisor

Tabla de contenidos

Lista de figuras

Lista de tablas:

1. Introducción

1.1. Objetivos del proyecto

Este proyecto esta supervisado por el profesor Josep Paradells Aspas, dentro del departamento de Ingeniería Telemática de la ETSETB.

El objetivo principal del trabajo consiste en implementar un sistema de transmisión que opere en el rango de frecuencias (390-700nm de longitud de onda) del espectro electromagnético denominado luz visible. La motivación para usar estas frecuencias es la saturación del espectro electromagnético por tecnologías ya existentes, por tanto, la luz como medio de transmisión es una alternativa que cada vez tiene más importancia.

Este proyecto se ha centrado en el sistema uplink de un sistema full duplex.

Para llevar a cabo la implementación, se usaran distintos dispositivos:

- Raspberry Pi
- Arduino o otro procesador de bajo coste
- Retroflector o fotodiodo
- Módulo de cámara raspberry
- Otros

Los objetivos del proyecto se pueden resumir de la siguiente manera:

- Implementar un sistema de comunicación
- Mejorar la velocidad de transmisión de bits
- Introducir sincronización entre dispositivos para maximizar la velocidad uplink

1.2. Requerimientos y especificaciones

Requerimientos funcionales:

- Transmitir a máxima velocidad posible
- Sincronizar ambos dispositivos

Requerimientos no funcionales:

- Usar arduino u otro procesador de bajo coste
- Usar Raspberry Pi
- Lenguajes de programación: python, opencv
- Conocimientos de procesado de imágenes

1.3. Plan de trabajo

1.3.1. Estructura de desglose del trabajo

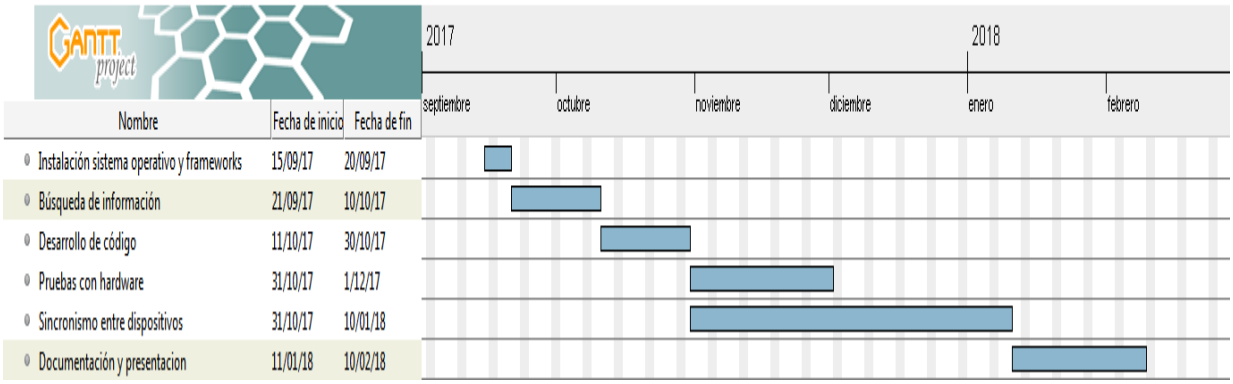
1.3.2. Fases del proyecto, tareas e hitos

Proyecto: VLC for IoT	WP ref: 1
Componente principal: Investigación y diseño	
Descripción: Descripción del problema y búsqueda de información	Fecha inicio:15/09/2017 Fecha finalización:10/10/2017
Tarea T1: Instalación del sistema operativo y frameworks Tarea T2: Búsqueda de información	

Proyecto: VLC for IoT	WP ref: 2
Componente principal: Implementación del software y hardware	
Descripción: Escribir el código necesario para realizar el sistema de comunicación	Fecha inicio:11/10/2017 Fecha finalización:10/01/2018
Tarea T1: Procesamiento de imágenes Tarea T2: Implementar software raspberry Tarea T3: Implementar software arduino Tarea T4: Posibles ajustes de hardware y montaje Tarea T5: Sincronización del sistema	

Proyecto: VLC for IoT	WP ref: 3
Componente principal: Documentación y presentación	
Descripción: Escribir memoria del proyecto y presentarlo.	Fecha inicio:11/01/18 Fecha finalización:10/02/18

1.3.3. Plan de trabajo (Diagrama de gantt)



1.3.4. Incidencias y modificaciones del plan de trabajo

Durante el proyecto han salido varios problemas los cuales han hecho que se tenga que retrasar el plan inicial. Uno de ellos era encontrar la mejor forma de sincronizar ambos dispositivos, que se ha prolongado más tiempo del previsto. Al principio, se buscó una sincronización, se queria dar la señal de sincronización directamente conectando el bus de la camara con el led. Al final se desestimo esta opción por su dificultad.

2. Estado del arte:Comunicación por luz visible

2.1. Un poco de historia

El primer sistema de transmisión inalámbrico fue el fotófono, diseñado por Alexander Graham Bell, que consistía en la transmisión de sonido mediante la emisión de luz. Aunque fue un gran invento, nunca se expandió su uso y los sistemas de comunicación sin cable de frecuencias más baja consiguieron más impulso.

Sin embargo, actualmente, se usa la luz para enviar datos, por ejemplo mediante la utilización de fibras ópticas.

También está creciendo el interés de usar la luz para comunicarse sin cables. Con los avances en las tecnologías, ahora mismo es posible usar este tipo de sistemas.

Con el paso del tiempo, el número de dispositivos móviles ha crecido de forma muy rápida y consecuentemente ha saturado el espectro electromagnético ya que este es limitado. Debido a esto se están investigando otras tecnologías de comunicación sin cable, entre las cuales destaca la comunicación por luz visible.

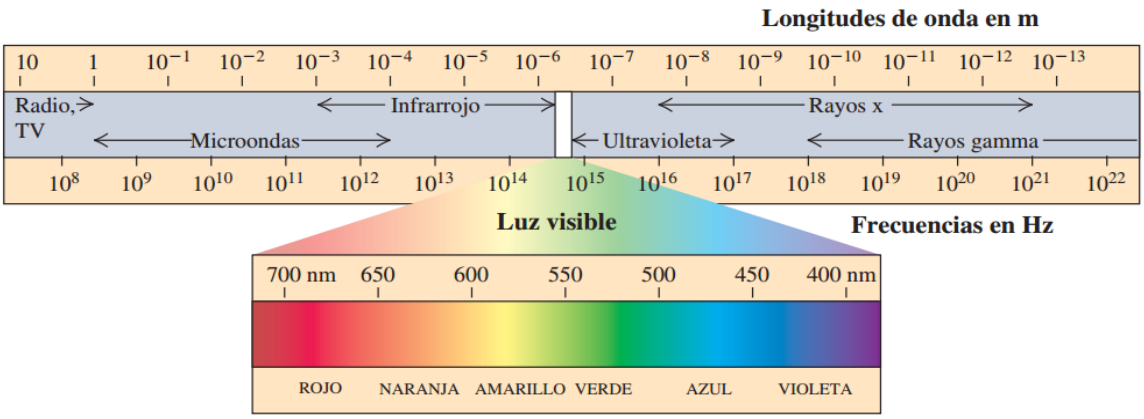


Figura 1: Espectro radioeléctrico

El principal desarrollo tecnológico que ha hecho posible los sistemas VLC han sido los LEDs. Se ha conseguido aumentar la potencia que emiten y además son capaces de cambiar de estado (apagado/encendido) a muy altas frecuencias. Además, con el paso de los años, también se ha mejorado en cuanto a eficiencia y coste.

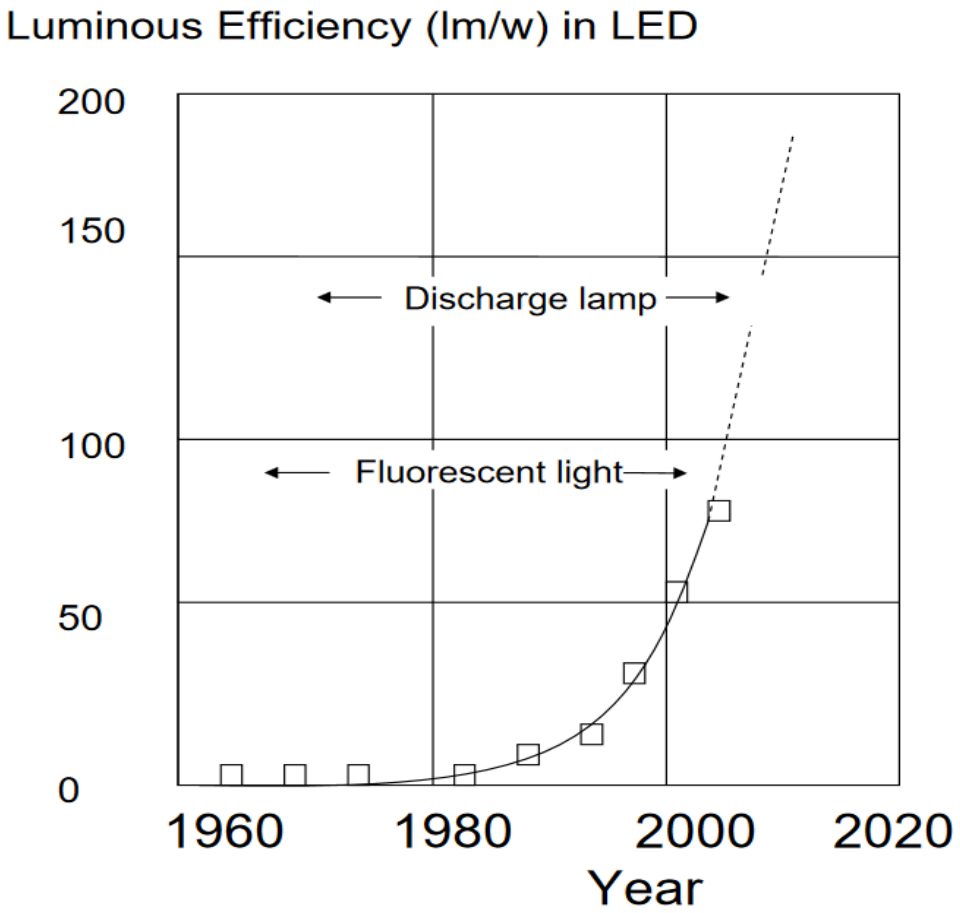


Figura 2: Eficiencia de los leds en el tiempo

Los sistemas VLC usan luz visible para la comunicación, es decir, ondas electromagnéticas entre 380nm y 750nm de longitud de onda que corresponden a 430 THz y 790THz de frecuencia. De esta manera se soluciona el problema de poco ancho de banda que había antes con los sistemas RF. Otra ventaja importante reside en que los receptores de VLC solo reciben señales de la misma habitación que el transmisor con lo cual son más seguros y tienes más privacidad que los sistemas RF. Además, una fuente de luz se puede usar tanto para iluminar como para comunicar y por tanto es un ahorro de energía. Finalmente, para disminuir el consumo de energia, cabe decir que se puede usar la luz para alimentar dispositivos.

2.2. Aplicaciones

Hay diferentes aplicaciones para esta tecnología:

Li-Fi: Es un sistema de comunicación bidireccional y totalmente conectado como el WI-FI que puede complementarlo en espacios cerrados donde haya interferencias con sistemas RF. Por lo tanto, se puede definir el Li-Fi como una particularición de VLC para proporcionar acceso a internet a través de la luz.

Parámetro	Li-Fi	Wi-Fi
Dispositivo transmisión	LED	Router
Velocidad transferencia	>1Gbps	>150Mbps
	Luz	Espectro radio
Rango del espectro	Ancho de banda muy elevado	Poco ancho de banda
Coste	Barato	Caro
Frecuencia operación	600 THz	2,4 GHz

Tabla 1: Comparación entre WiFi y LiFi

Aparte de las comunicaciones broadcast o punto a punto, VLC permite otras aplicaciones. Por ejemplo, es una oportunidad para crear un sistema de posicionamiento que sea flexible y preciso dentro de edificios. Actualmente, los sitemas de satelites tienen problemas en estos ámbitos ya que no son ni baratos ni precisos. En cambio, se pueden aprovechar las señales transmitidas por los LEDs para determinar la localización de objetos o personas dentro de habitaciones o edificios de una manera mas eficiente mediante la triangulación.

Como la luz, no interfiere con la radiación electromagnética de otros dispositivos es una importante alternativa en ámbitos donde la compatibilidad electromagnética es un requisito. Así pues, uno de los posibles ámbitos de uso, es usar estos sistemas en hospitales y aviones

Otro aspecto interesante es la comunicación entre vehículos, y además, se puede aprovechar la infraestructura que ya está instalada en las carreteras.

También se buscan nuevas maneras para comunicarse debajo del agua. El agua presenta una atenuación bastante mayor que el aire y tradicionalmente se han usado técnicas acústicas. VLC ofrece varias ventajas frente a las comunicaciones acústicas, entre ellas destacan por ejemplo más velocidad de transmisión y menos atenuación.

Otra situación en la que la comunicación mediante luz visible presenta ventajas ante los medios habituales de radiofrecuencia es en las comunicaciones seguras de corto alcance. Ya que la luz visible tiene una longitud de onda más corta que la radiofrecuencia, no se propaga más allá de un espacio cerrado, haciendo imposible que una antena externa intercepte las señales, sin necesidad de encriptar la información o usar complejos protocolos de seguridad. Esto también permite la posibilidad de “reutilizar” las frecuencias utilizadas en habitaciones adyacentes.

2.3. Modos de transmisión

Los canales de transmisión se clasifican en función de si hay obstáculos físicos entre el emisor y receptor y de la alineación de estos.

Así, se puede considerar posibles modos de transmisión. El primero es con línea de visión(line-of-sight) dirigida. En esta configuración no existen obstáculos y el emisor y el

receptor están orientados y alineados. Esta configuración se da sobre todo en comunicaciones punto-a-punto de larga distancia con diodos láser y en exteriores. Es la que menos pérdidas tiene, ya que utiliza un haz de luz concentrado y es con el que se pueden alcanzar tasas de transmisión más altas. Sin embargo, como el receptor no se puede mover sus aplicaciones son limitadas.

En la segunda posible configuración se sigue considerando una línea de visión entre el receptor y el emisor, pero sin estar orientados. En este caso la fuente de luz ya no es coherente, ya que tiene que emitir en toda el área de cobertura, y el receptor debe tener un ángulo de visión amplio. Este canal suele emplearse como broadcast, o comunicación punto-multipunto. Este modo presenta una menor eficiencia que el anterior, así como velocidades de transmisión inferiores. Esta es la configuración fundamental para los sistemas de VLC, ya que permite combinar la iluminación y la comunicación, diseñando un sistema broadcast con cobertura en todas las áreas iluminadas, sin necesidad de alinear emisor y receptor.

3. Metodología y desarrollo

3.1. Hardware

3.1.1. Raspberry Pi

El Raspberry Pi es un ordenador de una única placa o SBC (acrónimo en íngles de Single-Board Computer) cuyo objetivo principal es la enseñanza, aunque también se ha extendido de forma bastante amplia en al ámbito profesional en algunos proyectos.

Los primeros modelos de Raspberry Pi se lanzaron a principios del año 2012 los cuales no habrían sido capaces de realizar este proyecto ya que solo contaban con un procesador de un solo núcleo, con lo cual sería imposible procesar las imágenes en tiempo real. Con el paso de los años se han ido mejorando las especificaciones. En este proyecto se ha utilizado el Rapsberry Pi 3 que tiene un procesador mucho mejor con 4 núcleos, conectividad por wifi y 1 GB de memoria RAM.

Entre las especificaciones destacan:

- BCM2837:Microprocesador integrado de 1.2GHz de 64 bits
- GPIO: Es la interfaz entre la placa y componentes del exterior que puedan ser controlados
- 4 puertos USB:
- Ranura para Micro SD: La tarjeta de memoria es donde se instala el sistema operativo y además es donde se guardan los archivos.
- Conector CSI: Conector para conectar la cámara

- HDMI: Salida para conectar la placa a un monitor
- WI-Fi

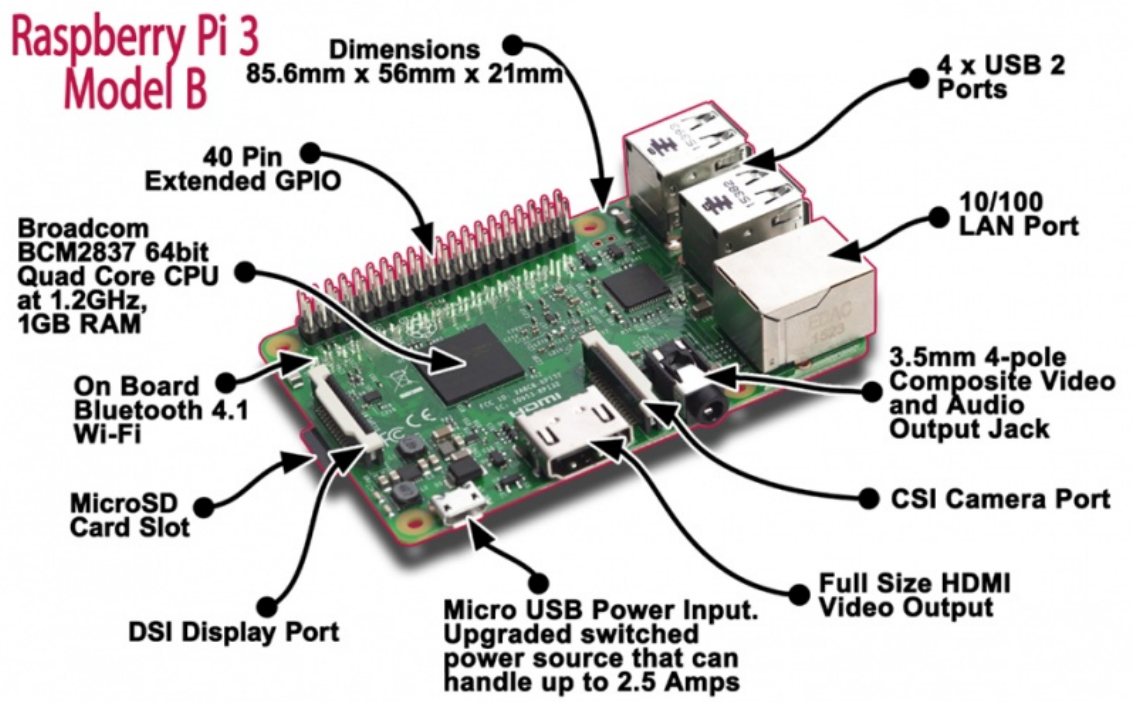


Figura 3: Placa de Raspberry Pi

3.1.2. Arduino

Arduino es un proyecto de hardware y código abierto. Hay diferentes versiones de la placa que incluyen un micro controlador con varias entradas y salidas para señales digitales y análogicas y un puerto de comunicaciones USB para conectarse con un ordenador. Además de los diferentes modelos, también existen extensiones (Shields) para ampliar las funcionalidades básicas de Arduino.

Se ha usado el Arduino UNO (Rev 3) para el proyecto, que es la versión más utilizada y de la que se puede encontrar más información y ayuda para el diseño de aplicaciones. Está basado en el micro controlador ATmega328 y consta de catorce entradas/salidas digitales (seis de ellas con posibilidad de utilizar modulación en anchura de pulso o PWM), seis entrada analógicas (no tiene salidas analógicas), un cristal de cuarzo de 16 MHz, un conector USB (que permite su alimentación y la conexión con un PC) y un conector de alimentación que permite su alimentación entre 7V y 12 V.

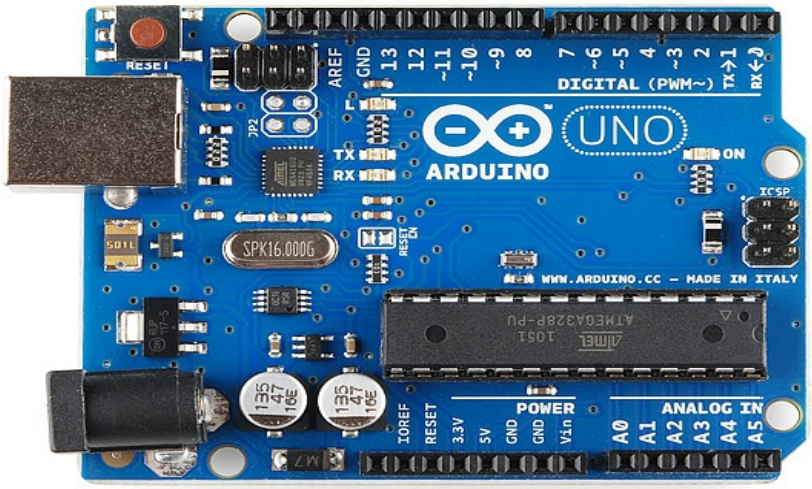


Figura 4: Placa arduino

3.1.3. Sensor de luz

Se han usado dos sensores diferentes:

El Sensor de luz de precisión (1127) mide el nivel de luz perceptible en lux; su rango de medición es de 1 lux a 1000 lux. Es un sensor más preciso comparado con un simple sensor LDR. El sensor se conecta a cualquier dispositivo con una entrada analógica, funciona directamente con dispositivos de 5V y 3.3V. Tiene un tiempo máximo de respuesta de 20ms y el pico de sensibilidad máxima está en los 580nm.

El fotodiodo BPW34 de Vishay Semiconductor es sensible a la radiación por infrarrojos y a la luz visible. Es ideal para aplicaciones de alta velocidad ya que tiene tiempos de respuesta muy rápidos, concretamente tiene un 'Rise time' y 'Fall time' de 100ns aproximadamente.

3.1.4. Driver del led Raspberry Pi

Se hace uso de un led que consumirá una potencia de unos 15W. Como la placa de Raspberry Pi no es capaz de suministrar tanta potencia, se debe construir un driver que se encargue de ello. Las salidas de los pines de la placa proporcionan un voltaje de 3.3V, con lo cual no es posible saturar un mosfet de potencia como el IRFBE30 que necesita un voltaje entre gate y source de unos 4V como mínimo. Consecuentemente se usa otro transistor de más pequeña potencia que sirve para saturar o no al de mayor potencia.

El driver diseñado es el siguiente:

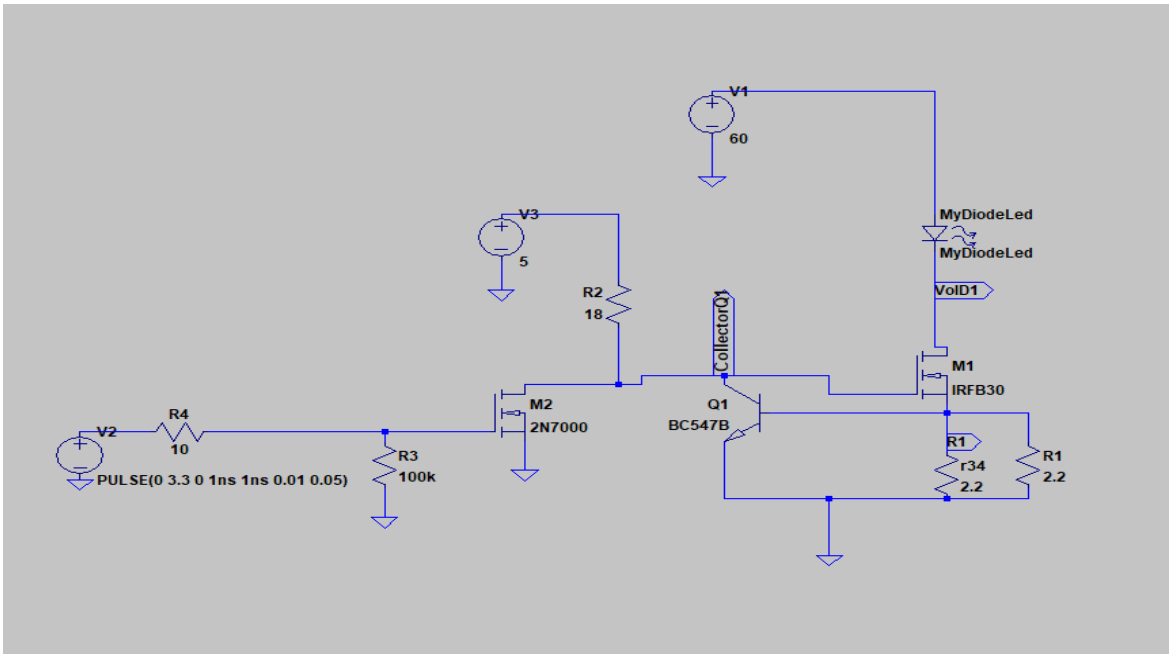


Figura 5: Circuito del driver del led Raspberry

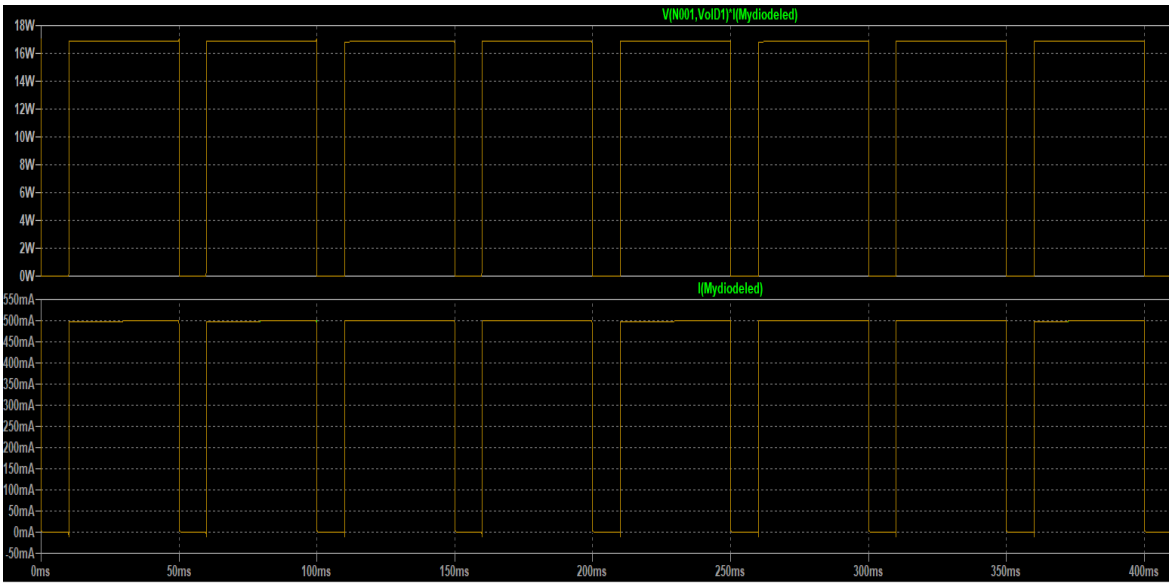


Figura 6: Intensidad y potencia del led Raspberry

Es importante destacar que el circuito es un inversor, cuando el GPIO esta en modo HIGH el led se apaga, en cambio cuando este en modo LOW la luz se enciende. La corriente del led esta limitada a unos 500mA gracias a las resistencias de 2.2 ohms.

De igual forma, el circuito funciona como un regulador de corriente. Cuando a través de las resistencias pasa mucha corriente y la caida de voltaje es igual al voltaje de saturación de base-emisor del NPN, se enciende Q1 y el gate de M1 va a tierra, disminuyendo de esta manera la intensidad que circula por el led. A continuación se reduce la caida de tension en las resistencias, apagando el Q1 y poniendo el gate de M1 a 5V con lo que vuelve a pasar mas corriente por el led.

3.1.5. Driver del led Arduino

También se ha diseñado un driver para el led del Arduino. Al principio se intentó utilizar un led pequeño que emite a unos 66mW pero la luminosidad no era suficiente a cierta distancia para poderlo detectar con la cámara. Por lo tanto se diseñó el siguiente circuito para controlar un led de mayor potencia de 500mW:

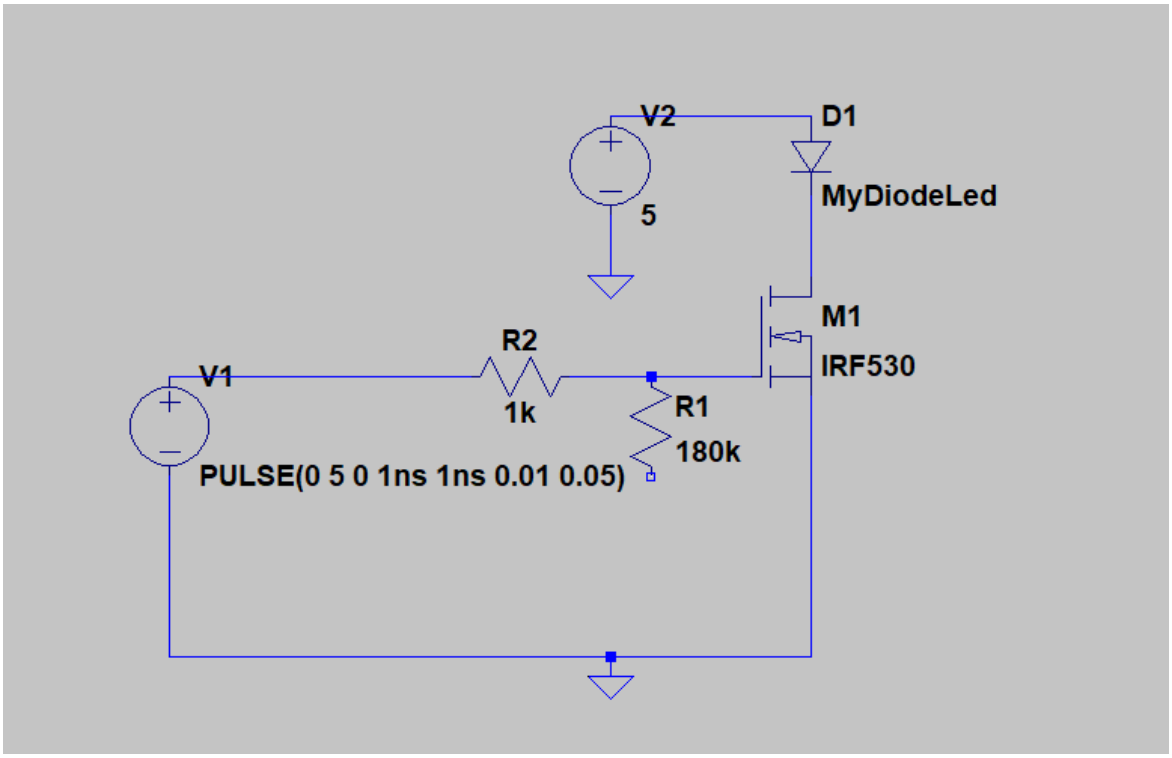


Figura 7: Circuito del driver del led de arduino

Es importante destacar que un pin de GPIO tiene un límite de 40mA y el pin Vcc de 5V de arduino puede ofrecer unos 500mA.

3.2. Sistemas embebidos y sistemas operativos

Un sistema embebido es un sistema electrónico diseñado específicamente para realizar unas determinadas funciones, habitualmente formando parte de un sistema de mayor complejidad. La característica principal es que emplea para ello uno o varios procesadores digitales en formato microprocesador, micro controlador o DSP lo que le permite dar cierta inteligencia a un sistema más grande.

En principio un sistema operativo tiene unos objetivos básicos: por una parte se debe otorgar al usuario que usa la máquina física un entorno fácil de manejar que oculte el diseño del hardware.

Por otra parte, y no menos importante, debe garantizar una serie de recursos tanto de software como de hardware que se comparten por los procesos que se están ejecutando.

El núcleo de un sistema operativo siempre ofrece una serie de recursos para una determinada aplicación que en la mayoría de los casos pueden no ser necesarios pero que al estar incluidos, hacen el sistema operativo más robusto. El problema es que se aumenta el tiempo requerido para ciertas aplicaciones.

Como se ha mencionado, una función importante del sistema operativo es asignar recursos de la máquina a actividades que se deban realizar. En un RTOS (siglas en ingles de un sistema operativo en tiempo real) este hecho es crítico porque hay actividades que se deben repetir en el tiempo de manera muy precisa. Por lo tanto se deben definir muy bien las actividades a realizar mediante un scheduler.

Parámetros a destacar de un RTOS:

- Tiempo de ejecución de una tarea: tiempo que tarda una tarea en completarse.
- Período de una tarea: la frecuencia con la que se llama a ejecutar una tarea. En general, se puede asumir que las tareas son periódicas.
- Utilización de CPU: el porcentaje de tiempo que el procesador se está utilizando para ejecutar un tarea programada
- Uso total de la CPU: la suma de la utilización de todas las tareas que se ejecutaran
- Preemption: esto ocurre cuando una tarea de mayor prioridad está configurada para ejecutarse mientras que una tarea de menor prioridad está ya en proceso.
- Fecha límite: una hora de finalización para la cual un sistema debe finalizar la ejecución de todas sus tareas programadas. La pérdida de un plazo tiene diferentes consecuencias, dependiendo del tipo de fecha límite que fue perdido.
    - fecha límite: una fecha límite para la cual fallar sería una falla completa del sistema.
    - fecha límite ‘soft’: una fecha límite para la cual hay una penalización. Se va arrastrando durante la ejecución

- Prioridad: importancia asignada a cada tarea; determina el orden en que las tareas se han de ejecutar.

- Instante crítico: un instante durante el cual todas las tareas del sistema están listas para comenzar a ejecutarse simultáneamente

Por lo tanto, cabe destacar que, un RTOS debe ser determinista, es decir, ha de realizar ciertas operaciones en unos instantes fijos con el menor retardo posible. Consecuentemente, para poder procesar las solicitudes de manera determinista la velocidad de la CPU es muy importante. Para saber si un sistema es determinista, se debe medir el retardo máximo desde la llegada de una interrupción hasta el inicio de ese proceso. En un RTOS va de microsegundos hasta un milisegundo, y en un OS puede llegar a los cientos de milisegundos.

También es muy importante el tiempo durante el cual se ejecuta el proceso que debe ser lo más constante posible al ejecutar el proceso repetidamente. Por tanto comentar que debe ser muy fiable ya que está controlando sucesos que llegan del entorno ES.

En principio la Raspberry Pi no fue diseñada para aplicaciones industriales o comerciales sino para la enseñanza, por lo tanto se han de tener diferentes aspectos en cuenta a la hora de usarlo como una aplicación en tiempo real.

- Disponibilidad de hardware
- Estabilidad
- Acceso a documentación y referencias
- Soporte para hardware y software

El núcleo ARM es parte del Broadcom BCM2835/36/37. Hay alguna documentación para los periféricos del chip pero es limitada y contiene algunos errores y omisión de información

Dentro del chip hay una pequeña ROM de arranque que lee archivos de tarjetas SD y los ejecuta. Cuando la Raspberry Pi se enciende o se reinicia, se intenta conectar con la tarjeta SD y busca un archivo llamado bootcode.bin; si lo encuentra, lo carga en la memoria y salta hacia él. Este fragmento de código continúa cargando el resto del sistema, como el firmware y el kernel ARM.

Ejecutar un sistema operativo en una nueva plataforma de hardware generalmente implica desarrollar un paquete de soporte de placa (BSP). El BSP es una colección de

rutinas dependientes del hardware y controladores de dispositivos que ofrecen soporte para los bloques de hardware fundamentales en el procesador, lo que permite que el kernel brinde sus servicios al software de la aplicación.

También se necesitan controladores de dispositivo para otros dispositivos e interfaces que la aplicación quiera usar, por ejemplo, USB, redes, SDHC, I2C, SPI, E/S, etc.

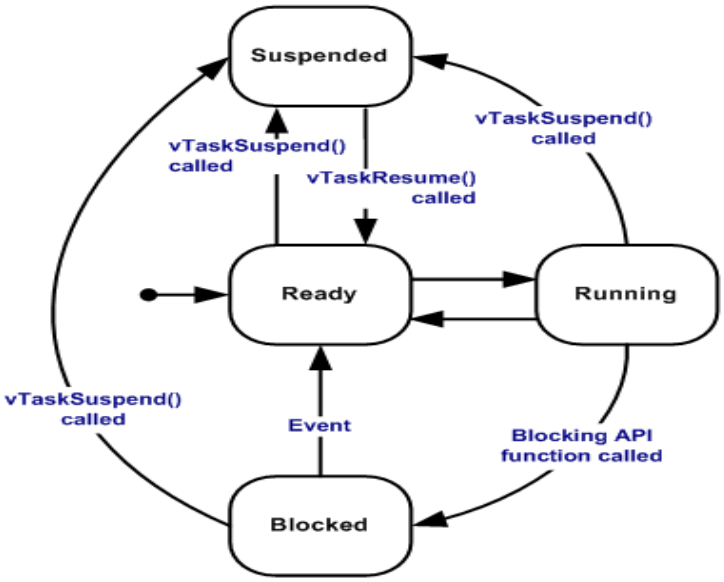
3.2.1. Sistemas operativos Raspberry Pi

A continuación se presentan algunos sistemas operativos en tiempo real que se les ha vinculado, como mínimo, alguna vez con la Raspberry:

FreeRTOS es un RTOS bastante reducido en comparación con grandes RTOS comerciales (como VxWorks, Windows CE, QNX, etc.). Es relativamente fácil de entender y trabajar, es gratuito para uso comercial y es bastante utilizado . La desventaja principal de trabajar con este es la falta de soporte para funciones complejas.

Se ha implementado para muchos micro controladores, alrededor de unos 35. Es un sistema operativo pequeño y simple escrito mayoritariamente en C (tiene un conjunto de bibliotecas y un programador de tareas). Tiene métodos para múltiples tareas, semáforos y timers. No hay una distribución oficial para Raspberry Pi, aunque si que se ha implementado por terceras personas de manera no oficial y sin documentación. Esto incluye una serie de problemas así como también, se debe programar y compilar en un ordenador de escritorio, e ir insertando los datos en la tarjeta SD.

En cada ciclo de reloj (configurado en 1 ms en la Raspberry Pi) el planificador lanza una interrupción y considera todas las tareas para ejecutar. Ejecuta la tarea listada de mayor prioridad. Si hay un empate de tareas que tienen mayor prioridad, utiliza la planificación de turnos para alternar entre las tareas de esa prioridad. También se puede "Bloquear" tareas y hacer que se eliminen de la lista hasta que ocurra un evento (ejemplos de eventos incluyen un semáforo que se establece y una cierta cantidad de tiempo que pasa). Mientras que por ejemplo FreeRTOS en Arduino conserva las funciones para controlar GPIO, una API genérica ha de ser portada para que se pueda controlar los pines GPIO específicos cuando se usa FreeRTOS en el Pi.



Running:
La tarea se está ejecutando
Ready:
La tarea esta lista para ejecutarse pero hay otras con más prioridad que se ejecutan.
Blocked:
Tareas que esperan algún evento.
Suspended:
Tareas que solo entran o salen de este estado si se les da la orden explícitamente

Figura 8: Estados FreeRTOS

Xenomai es un framework para desarrollo en tiempo real que coopera con el núcleo de Linux.

Hay dos opciones para usarlo:

La primera opción es complementar a Linux con un co-kernel en tiempo real que se ejecuta junto a él. Esto es una pequeña extensión llamada Cobalt que está integrada en el kernel de Linux, y se ocupa de todas las actividades donde el tiempo debe ser muy preciso, como el manejo de interrupciones y la programación de subprocesos en tiempo real. El núcleo Cobalt tiene una mayor prioridad sobre las actividades nativas del kernel.

En esta configuración dual del núcleo, todas las API de Xenomai proporcionan una interfaz con el núcleo de Cobalt, y solo esas API se consideran aptas en tiempo real.

La segunda opción es confiar en las capacidades en tiempo real del núcleo Linux nativo, formando el núcleo de Mercury. A menudo, las aplicaciones requerirán que la extensión PREEMPT-RT se habilite en el kernel para la entrega de servicios en tiempo real. Sin embargo, esto no es obligatorio, depende de los requisitos de la aplicación con respecto a la capacidad de respuesta y la máxima inestabilidad.

La limitación está en que no tiene soporte para el estándar de camera CSI-2, si lo tiene para USB aunque no para tiempo real

RTEM (Real Time Executive for multiprocessor Systems) soporta interfaces de programación como POSIX. Se usa en ámbitos como los vuelos de aviones, medicina, redes y más dispositivos encastados como:ARM, PowerPC, Intel, Blackfin, MIPS, Microblaze. Ha habido algún intento de traspasarlo a la Raspberry.

ChibiOS/RT es un sistema operativo en tiempo real que soporta múltiples arquitecturas. Es para sistemas incrustados de 8, 16 y 32 bits. También se ha llevado a cabo una implementación del sistema para la Raspberry Pi implementando los siguientes controladores de dispositivos:

- ADC
- CAN
- DAC
- EXT
- GPT
- HAL
- I2C
- I2S
- ICU
- MAC
- PAL
- PWM
- QSPI
- RTC
- SDC
- Serial
- SPI
- ST
- UART
- USB
- WDG

Además de los drivers mencionados también hay otros que se han implementado gracias a terceras personas, entre los ejemplos se encuentran por ejemplo GPIO, GPT.

La arquitectura se compone de:

ChibiOS / RT, es el scheduler

ChibiOS / HAL, la capa de abstracción de hardware que contiene controladores para la mayoría de los periféricos comunes.

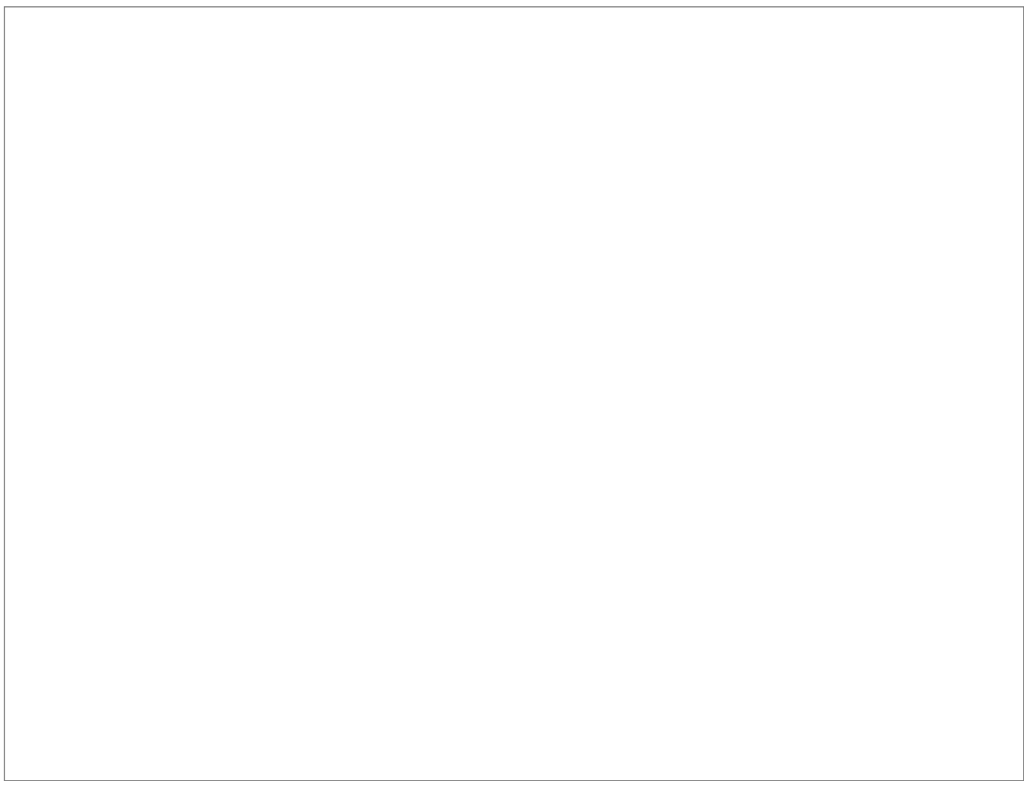


Figura 9: Arquitectura ChibiOS

Con este sistema se han realizado proyectos de Raspberry Pi básicos como por ejemplo encender y apagar un led. El principal problema de los sistemas operativos descritos es la integración con la cámara Raspberry Pi mediante la interfaz de comunicaciones CSI-2 además del framework OpenCV. Es importante destacar la poca documentación existente de estos sistemas operativos en la placa de desarrollo Raspberry ya que no son los oficiales y consecuentemente añade mucha dificultad.

Finalmente el sistema operativo usado es el Raspbian que es de descarga gratuita y ademas es el oficial para la Raspberry Pi y con lo cual existe soporte.

La Raspberry Pi viene por defecto sin disco para almacenamiento ni sistema operativo incorporado. Debido a esto se necesita una microSD para cargar el sistema operativo y guardar los archivos necesarios.

Los pasos que hay que seguir son los siguientes:

1. Descargar de la página oficial el sistema operativo Raspbian

2. Usando Win32DiskImager se carga la imagen del sistema operativo en la SD.

3. La primera vez que se arranca la Raspberry Pi con la tarjeta incorporada se necesita como mínimo un teclado, un ratón y una pantalla. Una vez dentro, se debe ajustar la configuración de la siguiente forma:

    - Expand file system
    - Enable camera
    - Enable VNC

También se puede configurar para acceder a la Raspberry desde un ordenador remoto por ejemplo usando PuTTy (por comandos) o mediante VNC Viewer (por interfaz gráfica)

3.3. Camara de la placa VS USB camara

La cámara de la raspberry está diseñada específicamente para conectarse mediante un cable plano con el conector CSI-2 de la placa. El CSI-2 (Camera Serial interface 2) es una especificación de la interfaz que define el dispositivo periférico y el procesador en este caso, la cámara y la Raspberry Pi.

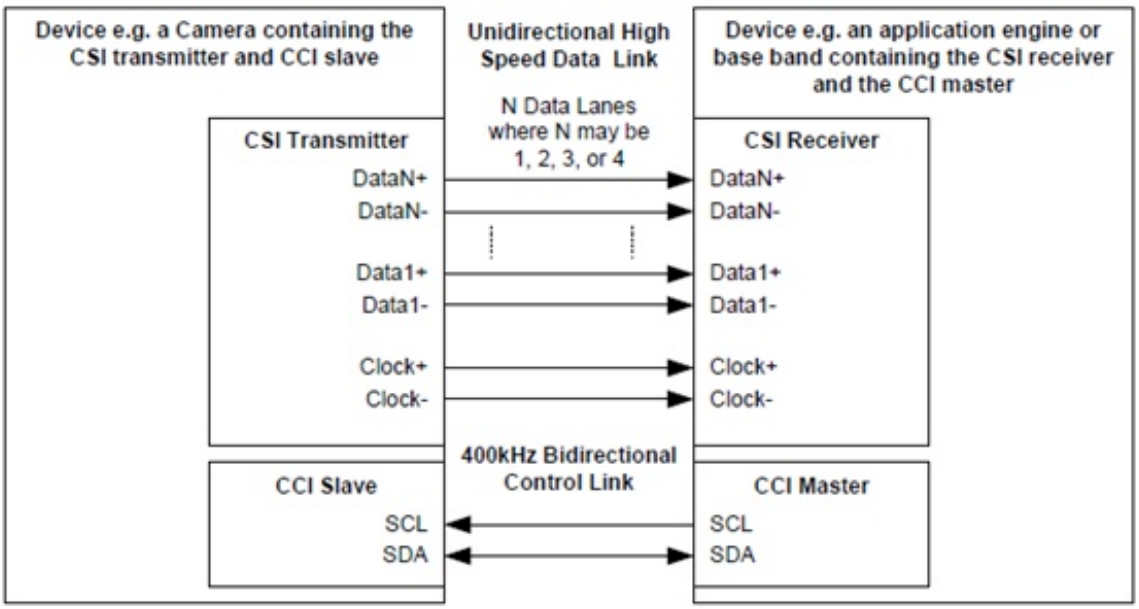


Figura 10: Esquema de comunicación de interfaz CSI-2

En este caso hay dos ‘data lanes’. El protocolo de la interfaz decide que paquetes de datos se van a enviar por cada ‘data lane’, es decir, la información se envía de forma paralela. Por el lado contrario, el receptor une los paquetes recibidos para volver a tener la misma secuencia que se envía. Además, como el CSI-2 esta conectado directamente con el procesador, se dispone de una velocidad de transmisión mayor que con cámaras USB. Por lo tanto, se espera más velocidad en fotogramas por segundo y un menor consumo de los recursos hardware.

3.4. Sincronización

Si se usa un led como trasmisor y una camara como receptor en un sistema asíncrono el factor limitante en la velocidad de transmisión es el frame rate de la camara. Como postula el teorema de Nyquist, la frecuencia de muestreo del receptor ha de ser como mínimo el doble de la frecuencia de la señal transmitida. Por lo tanto la máxima velocidad posible de transmisión viene determinada por la mitad del frame rate especificada por la camara.

Para mejorar el sistema, se busca la sincronización.

Si se quiere interpretar correctamente las señales recibidas del transmisor, los intervalos de los bits del receptor y el transmisor deben ser exactamente iguales. Si por ejemplo, el reloj del receptor es un poco más rápido o lento, los intervalos de tiempo no se corresponden y consecuentemente hay errores en la comunicación.

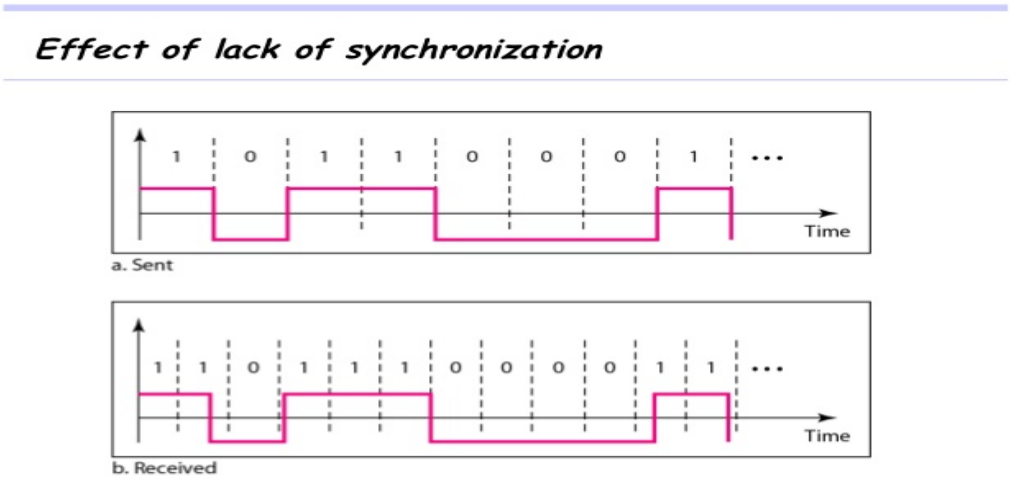


Figura 11: Ejemplo de falta de sincronización

El módulo de cámara de raspberry se debe sincronizar con el otro dispositivo, en este caso un arduino. Como se trata de comunicación por luz visible, cada vez que el led del transmisor realice un parpadeo o no (dependiendo de si se envía un 0 o 1), en el mismo instante se debe capturar un fotograma con la cámara. Como se ha dicho anteriormente, se debe evitar por ejemplo que haya dos bits de información (dos parpadeos) en la ventana de tiempo de un fotograma.

El led debe ir transmitiendo bits y la camara debe ir muestreando el estado del led. La camara no muestrea exactamente a intervalos regulares y por lo tanto se pueden perder bits. Debido a esto, se le debe indicar al transmisor en que momento se quiere recibir la información mediante una señal de flag de bit.

Por lo tanto, para sincronizar los dispositivos se usa una señal de flag (apagar led). Para ello se han probado diversas opciones:

La primera idea era conseguir apagar el led del flag conectándolo directamente al bus CSI-2 de la cámara Raspberry Pi. El principal problema era que entre las diversas señales del bus no había ninguna que indicara el instante de inicio de captura de un fotograma o el final de este, o como mínimo una combinación de señales con este propósito.

Figura 12: Camara de la raspberry

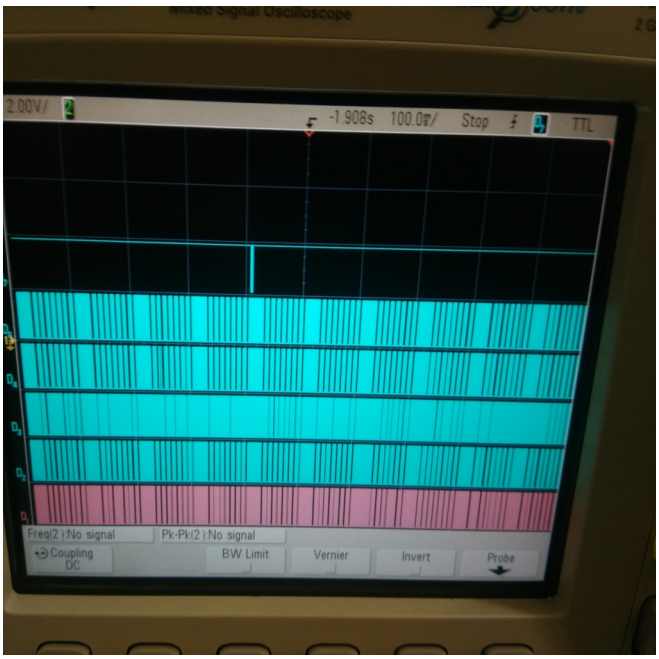


Figura 13: Señales de la camara de la raspberry

La segunda idea era usar el propio led de la camara Raspberry Pi como flag. Internamente pero, esta señal del bus está conectada a un pin GPIO y tampoco se mejoraba la velocidad de fotogramas.

Finalmente se ha decidido usar un pin GPIO para controlar el led.

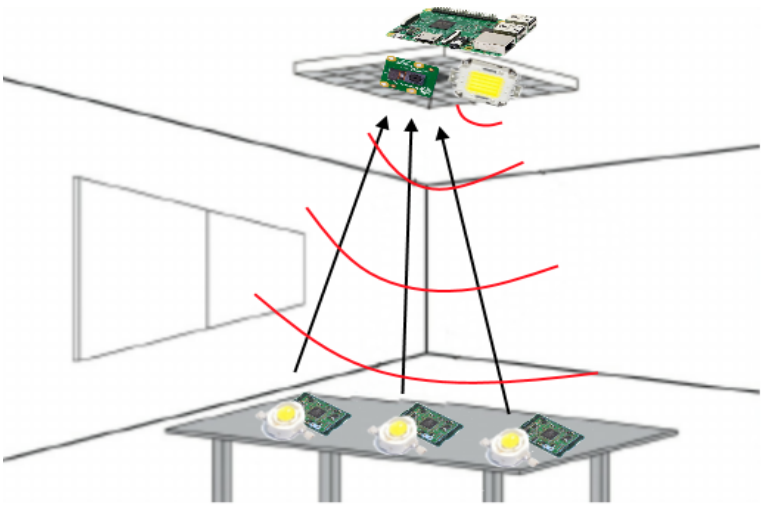


Figura 14: Esquema del sistema completo

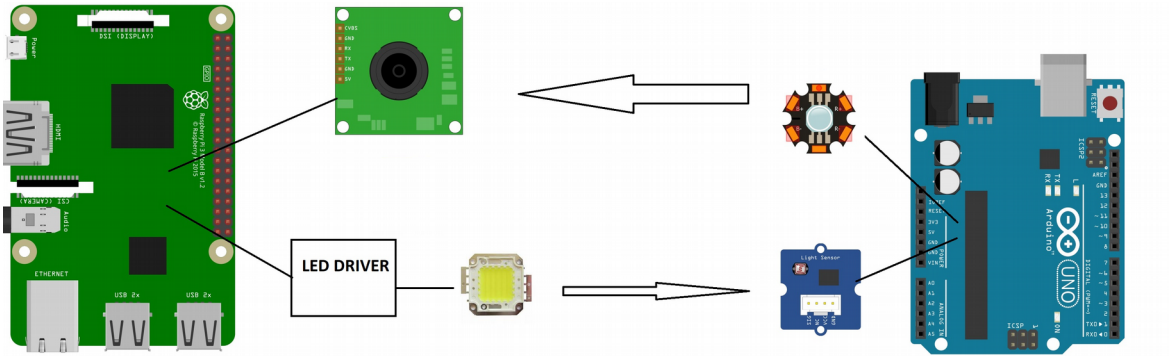


Figura 15: Esquema del sistema elemento a elemento

3.5. Software

3.5.1. Algoritmo

En estado 'idle' los leds tanto del arduino como la raspberry están encendidos. A continuación el dispositivo que inicia la comunicación es la Raspberry Pi realizando una foto. Analizando la imágen, se detecta el circulo de luz del led mediante el método de gradiente de Hough y también se calcula el valor umbral para después binarizar la imagen. En el siguiente paso, la raspberry apaga el led y es la señal para que el arduino empiece a transmitir información. Cada vez que la Raspberry Pi apague el led(flag) se transmite un único bit del otro dispositivo.

Cuando se esta transmitiendo, el micro controlador captura imágenes continuamente de la cámara. A continuación cada imagen es analizada usando técnicas de procesado para detectar la señal transmitida. Las imágenes con color no son necesarias para distinguir si

el led está o no encendido, por lo tanto, el primer paso es convertir la imagen a escala de grises, con lo cual hay muchos menos puntos de información y la manipulación de imágenes es mucho más rápida. Con la transformación cada pixel es representado por un valor entre 0 y 255.

A continuación se aplica un desenfoque gaussiano para reducir ruido (posible luz ambiente) y finalmente se aplica un umbral binario para convertir la imagen en blanco y negro. Finalmente se decide si se ha detectado o no la luz del led

Finalmente se acaba la comunicación cuando se termina el tiempo establecido de los test.

Los dos dispositivos vuelven al estado 'idle' para iniciar la comunicación en otro momento.

3.6. Simulación de procesamiento teórico

A continuación se ha hecho un estudio de la capacidad de procesamiento de las imágenes de la Raspberry Pi cuando se graba en vídeo segun el número de threads:

Usando solo un hilo, la Raspberry Pi tiene problemas para poder procesar 90 fps, que es el máximo que permite la cámara. Aproximadamente, con este método, se puede procesar el 10% de 90 de fotogramas que llegan en un segundo.

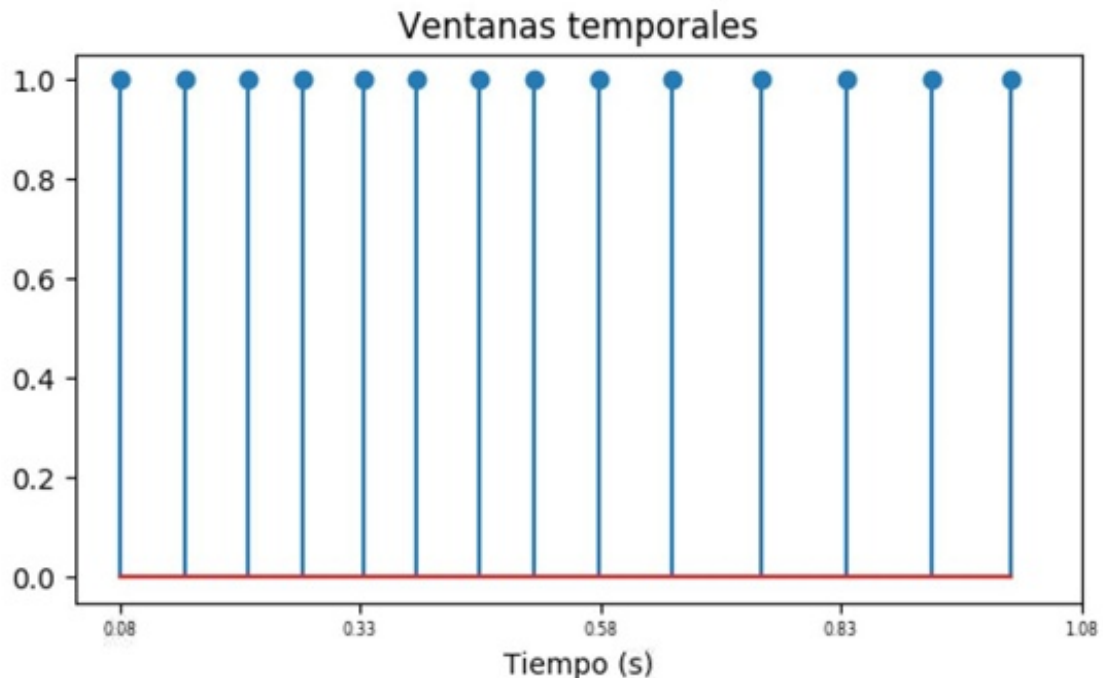


Figura 16: Espaciado en el tiempo de capacidad de procesamiento

FPS 12.9

Máxima duración tiempo fotograma 0.09238 s

Mínima duración tiempo fotograma 0.05511

Duración promedio tiempo fotograma 0.07128

Desviación estándar tiempo fotograma 0.01262

Cuando el controlador accede a la cámara para obtener fotogramas se produce una operación de bloqueo (función read()), esto significa que no puede continuar con la ejecución de instrucciones hasta recibir una señal y como consecuencia la velocidad se reduce.

Usando 2 hilos se pueden lograr unos resultados mucho mejores. El thread principal es el encargado de realizar el procesamiento de los fotogramas mientras que el otro es el encargado de manejar las tareas de entrada, es decir, leer las imágenes del sensor de la cámara.

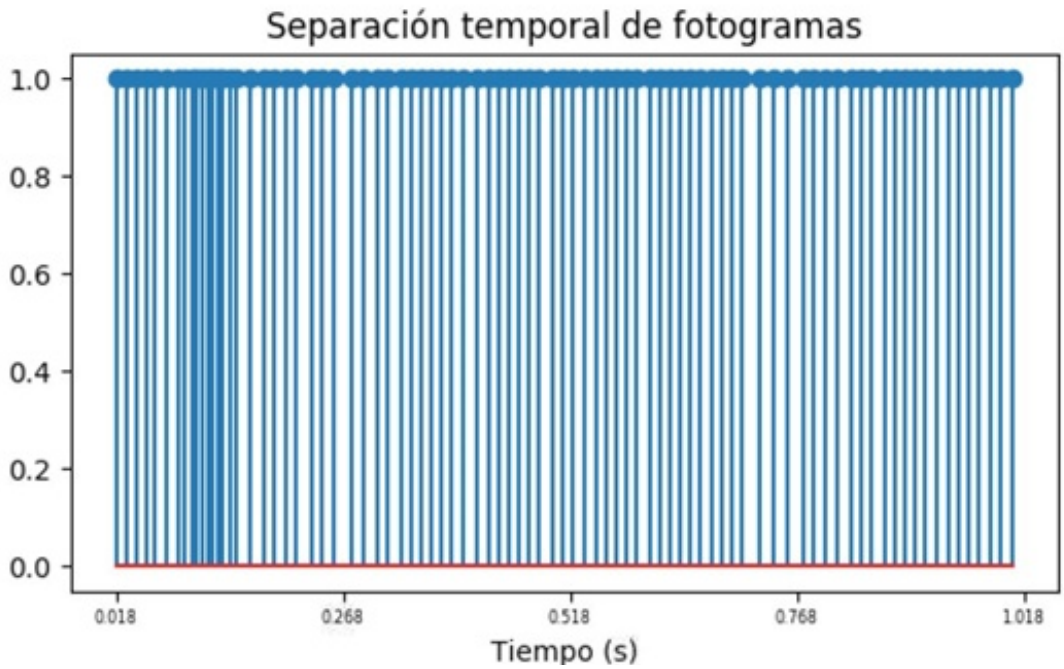


Figura 17: Espaciado en el tiempo de capacidad de procesamiento 2

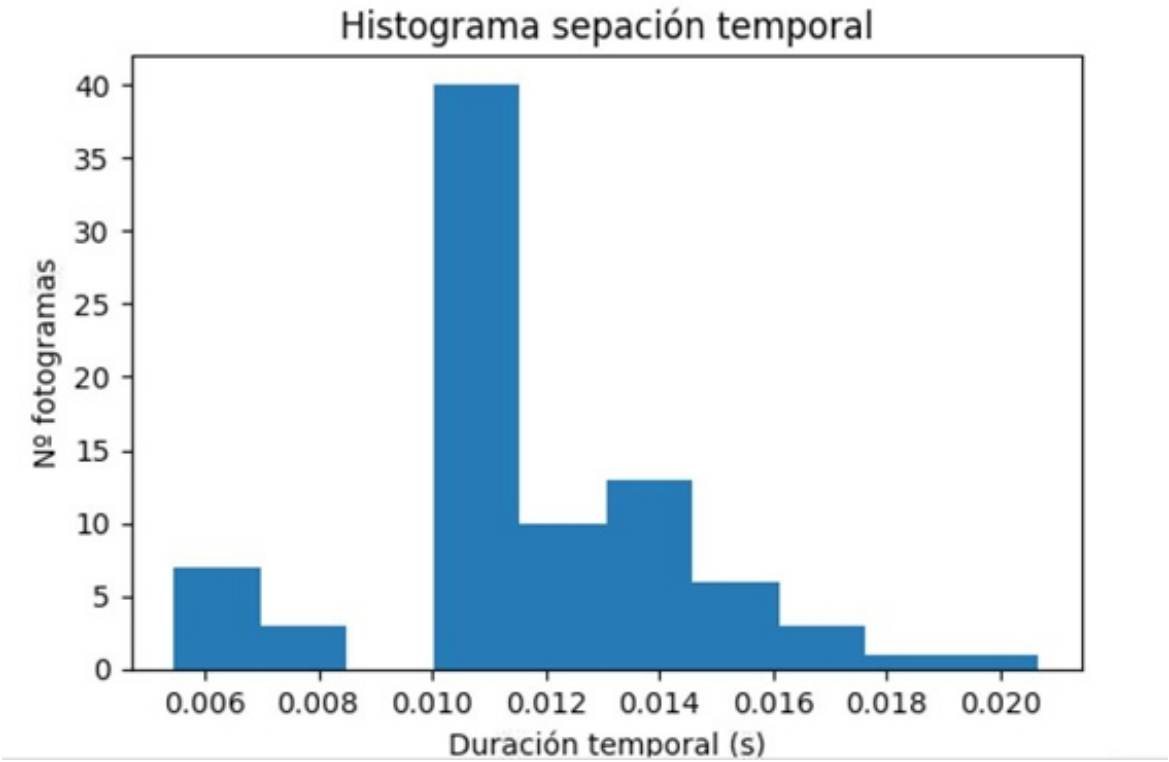


Figura 18: Histograma de la duración temporal de procesamiento de los fotogramas

FPS 83.44

Máxima duración tiempo fotograma 0.0208 s

Mínima duración tiempo fotograma 0.00546 s

Duración promedio tiempo fotograma 0.01177 s

Desviación estándar tiempo fotograma 0.0028 s

Con dos hilos se pueden llegar a los 320 fps (sin usar la función resize):

Máxima duración tiempo fotograma 0.00611 s

Mínima duración tiempo fotograma 0.00173 s

Duración promedio tiempo fotograma 0.00291 s

Desviación estándar tiempo fotograma 0.00100 s

Aunque estos fotogramas procesados no son reales, ya que la cámara solo llega a 90 fps, lo que significa que el thread principal está procesando más de una vez el mismo fotograma.

3.7. Implementación

Existe una herramienta por línea de comandos(Raspivid) que se usa para video con el módulo de cámara. Este programa es de código abierto y por tanto se puede ver su implementación. También la librería Picamera es una interfaz de python de código abierto.

La capa MMAL que está por debajo de picamera y de la herramienta raspivid proporciona una interfaz para el firmware de la cámara que se ejecuta en la GPU. Conceptualmente, presenta la cámara con tres "puertos": el puerto fijo, el puerto de video y el puerto de vista previa.

Finalmente se ha decidido usar python por su simplicidad a costa de aumentar la latencia. Se ha descargado la librería Picamera, diseñada para controlar la cámara mediante el conector CSI-2. Dentro de ella, se ha modificado el método **`capture_continuous`** para controlar el led en los instantes precisos. El programa principal consta de 2 threads, el principal es el encargado de procesar las imágenes repetidamente mientras que el otro se encarga de ir capturando fotogramas.

Entre las funciones de procesamiento, destacan aquellas que proceden de la librería openCV:

`cv2.HoughCircles:`Para detectar el circulo de luz del led.

`cv2.GaussianBlur:` Mezcla ligeramente los colores de los píxeles que son vecinos el uno al otro.

`cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)` : Para cambiar de BGR(formato de imágenes por defecto de la cámara) a escala de grises.

`cv2.threshold(img,thr,255,cv2.THRESH_BINARY):` Para binarizar la imagen.

El umbral se decide al principio de la ejecución con la primera imagen. Se detecta la luz emitada por el led y se coge el píxel del centro del círculo. A continuación se convierte a escala de grises. Se hace el promedio entre el negro absoluto (0) con el valor de luminosidad del píxel escogido del led. Además, se le resta un valor(10) que se ha determinado empíricamente después de varias pruebas.

En cuanto al arduino, es importante destacar lo siguiente:

La función digitalWrite, que cambia los estados de los pines de salida en Arduino, tarda un tiempo (cada digitalWrite son unos 4us).Por lo tanto, manipulando directamente los registros del chip ATmega328, el proceso se reduce. Se usa el PORTB que mapea los pines digitales del 8 al 13 para encender el led del Arduino. Tiene los siguientes registros:

- DDRB - El registro de configuración del modo de los pines del puerto B - lectura/escritura
- PORTB - Registro de datos del puerto B - lectura/escritura
- PINB - Registro de pines de entrada del puerto B - solo lectura

Cada vez que se ejecuta la función Delay() el micro controlador de Arduino se bloquea, espera a que transcurra el tiempo que se ha establecido y ejecuta la siguiente instrucción. Esto provoca el problema de que no se pueden leer eventos (no se pueden leer señales de sensores porque los ignora). Además, cuando se usan estas funciones, los problemas que presenta esta función se van arrastrando en cada iteración del bucle principal y consecuentemente hay desajustes en los intervalos de tiempo. Por lo tanto se ha optado por usar la función millis() e implementar un timer.

4. Resultados

4.1. Velocidad de transmisión

Las primeras pruebas se realizaron con el propósito de determinar la máxima velocidad de transmisión posible acorde con los componentes usados. En principio, como ya se ha comentado anteriormente, la velocidad está muy ligada con el flag de sincronización entre los dispositivos. Este flag, debe ser muy preciso en el tiempo, con lo cual el sistema operativo influye.

El test se realizó transmitiendo bits durante 30 segundos con una distancia de 145 cm en una habitación totalmente oscura, solo los leds la iluminaban. Los bits que se enviaban corresponden a los siguientes ‘110100’. Como se puede observar en la figura de abajo, cada vez que el led de la raspberry se apagaba(D3 en estado alto) se capturaba un fotograma que bien podía ser 1 o 0(D6).

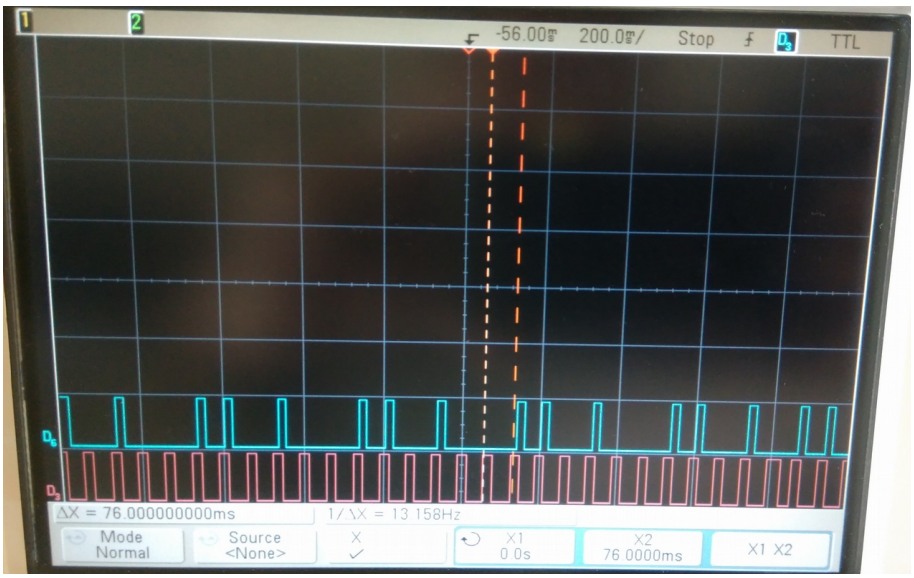


Figura 19: Ejemplo de secuencia de bits '110100'

El ADC del arduino tiene una resolución de 10 bits, con lo que devuelve valores enteros entre 0 y 1023 siendo la sensibilidad de 4.8 mV.

En este primer test se ha usado el sensor de luz (1127) que tiene un tiempo de respuesta de 20 ms. El valor umbral del arduino está en 10.

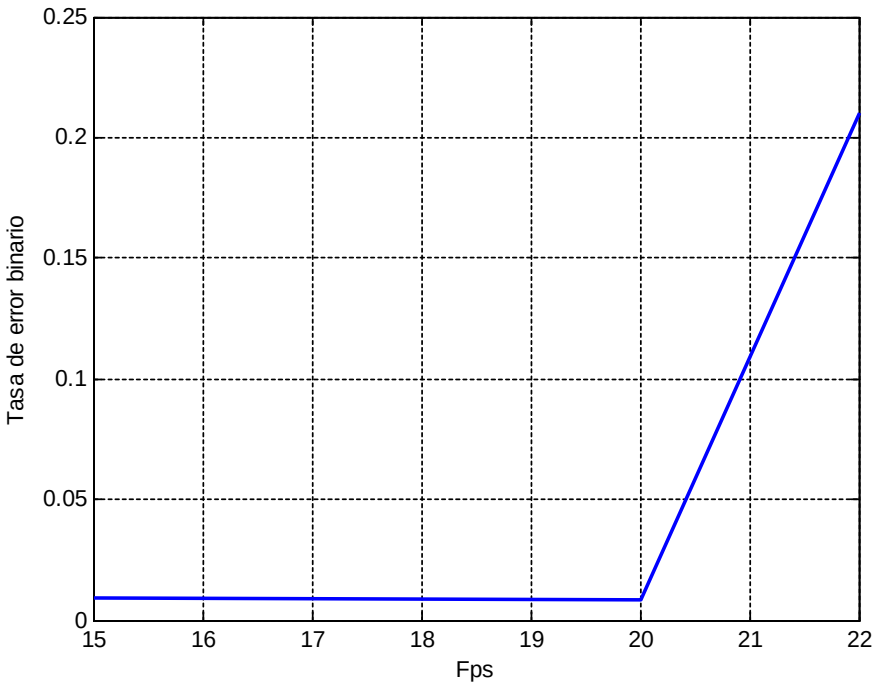


Figura 20: Errores de bits sensor 1127

Como se puede ver en la gráfica, hasta 20 fps, aproximadamente el error(0.008) es contante. Cabe destacar que el test se realizó durante 30 segundos, es decir, con mas tiempo de transmisión el error hubiese disminuido. A partir de 20 fps, empiezan a haber desajustes. Esto es debido, a que se debe incorporar un retardo de 20 ms para que al poner el estado del pin de la Raspberry Pi en HIGH (significa que se apaga el led) el sensor de luz tenga tiempo a leerlo. Durante este tiempo cogen prioridad otras tareas y por tanto aumentan los errores cuando se incrementan los bits por segundo.

De vez en cuando, se producen retardos en los bits que se cuentan como un error. Por ejemplo, puede pasar lo siguiente:

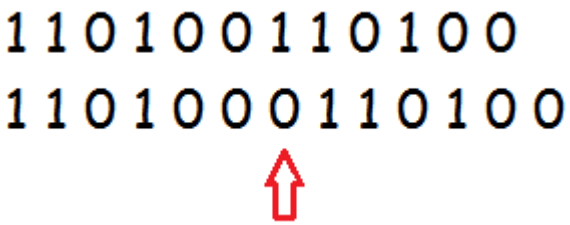


Figura 21: Ejemplo de retardo

En el segundo test se ha utilizado el fotodiodo BPW34 que tiene un tiempo de respuesta mucho menor. Como consecuencia se ha podido disminuir el tiempo de flag en unos 5 ms.

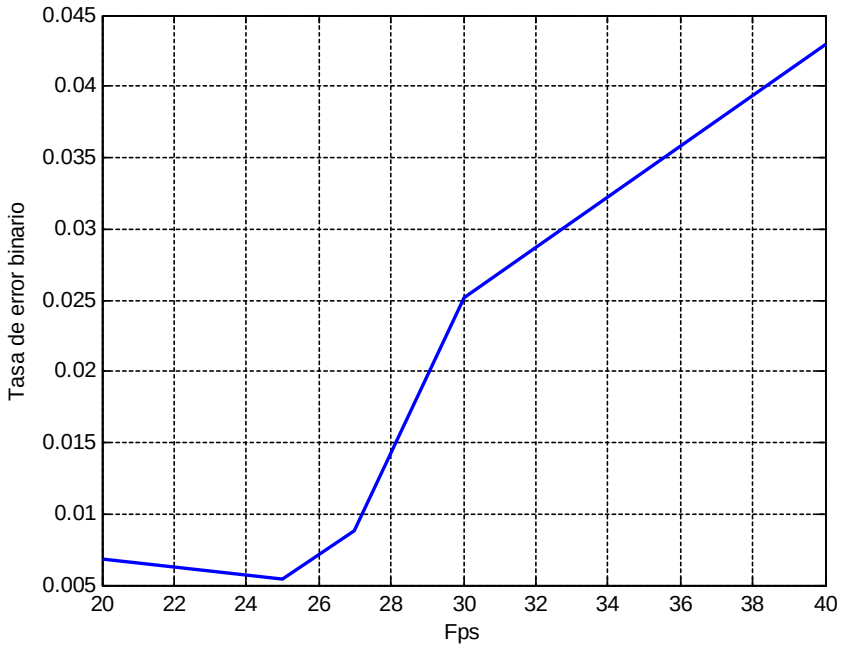


Figura 22: Errores de bits fotodiodo BPW34

En la gráfica se puede observar que el error entre 20 y 25 fps es aproximadamente constante (entre 0.005 y 0.008) como sucedia con el otro sensor.

El tercer test también ha sido con el fotodiodo BPW34 pero esta vez con más tiempo(130 segundos) de transmisión para ver que sucede con la tasa de error binario.

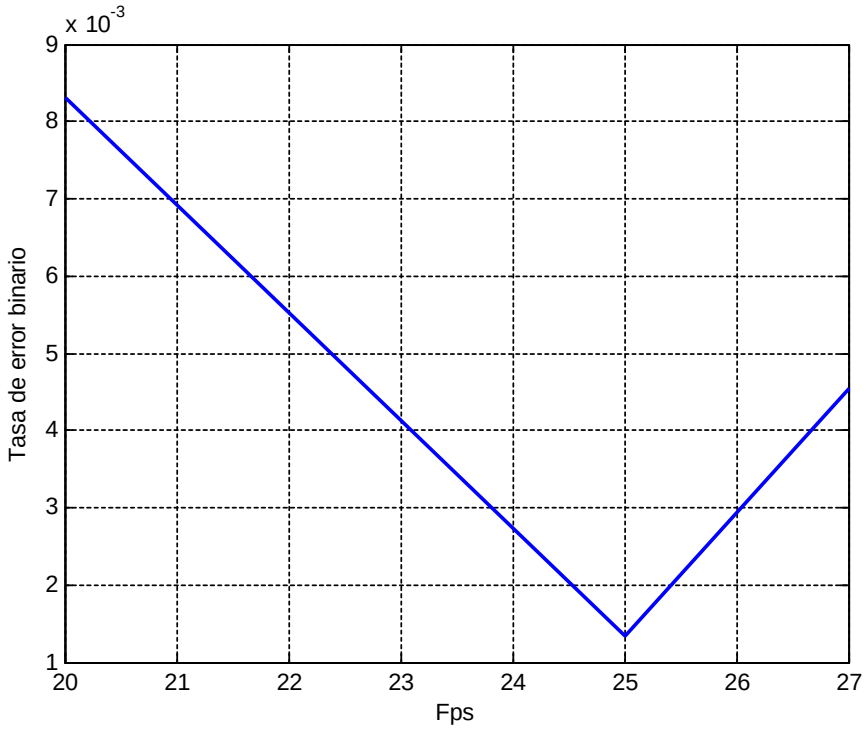


Figura 23: Errores de bits fotodiodo BPW34 durante 130 segundos

Con más tiempo para enviar datos, la tasa de error en 25 fps baja ya que se envía un número mayor de bits. A partir de 30 fps, es ya muy complicado determinar la BER.

El principal problema por el cual no se puede mejorar la velocidad de transmisión es cuando hay mensajes entre el sistema operativo y la cámara. Hay desajustes cuando se tiene que acceder primero al led mediante la interfaz GPIO y a continuación también a la camara. Dentro de la función Capture_Continuos(que captura fotogramas continuamente) se han añadido las siguientes funciones GPIO.output(11,GPIO.HIGH) y GPIO.output(11,GPIO.LOW) así como también un time.sleep(). Esto conlleva un tiempo. Durante esta pausa el firmware de la camara(VCOS) sigue recibiendo filas de las imagenes y las va descartando hasta que realmente se le pide que entregue un fotograma. Por lo tanto, es durante este tiempo cuando aparecen los desajustes.

A continuación se presenta un esquema de la ejecución de la camara:

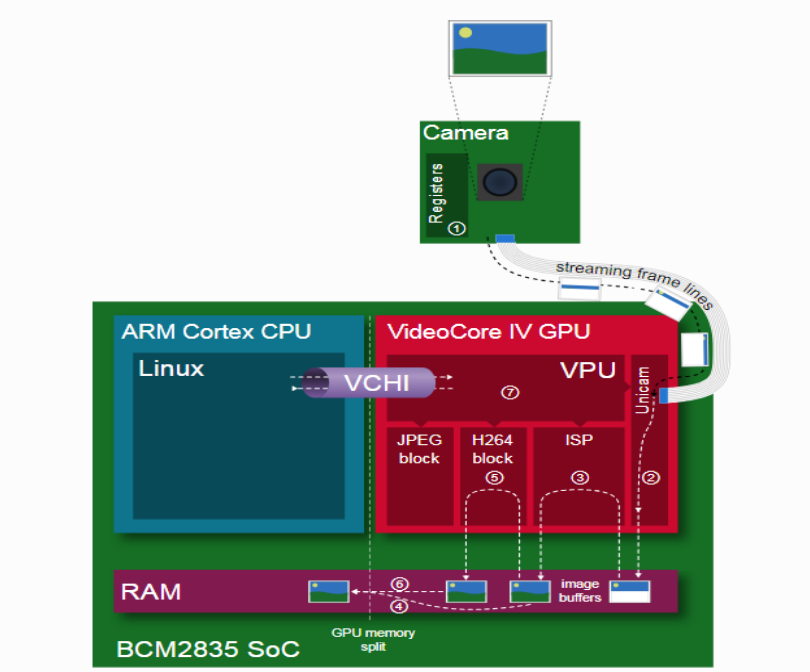


Figura 24: Esquema de ejecución de la cámara

4.2. Distancia de transmisión

El siguiente test consiste en ir separando el transmisor y el receptor para comprobar la máxima distancia. Puede haber varias causas para que al aumentar la distancia no se consiga transmitir:

En primer lugar, fijado un umbral en el arduino, existe una distancia para la cual el fotodiodo no es capaz de distinguir entre el flag apagado o no ya que la señal es muy débil y se mezcla con el ruido(luz ambiente).

Otra razón, es la luminosidad del led del arduino. A cierta distancia, la cámara no puede detectar el círculo de luz del led ya que ocupa muy pocos píxeles en la imagen.

En la habitación totalmente oscura se ha medido el nivel de iluminación que produce el led en función de la distancia.

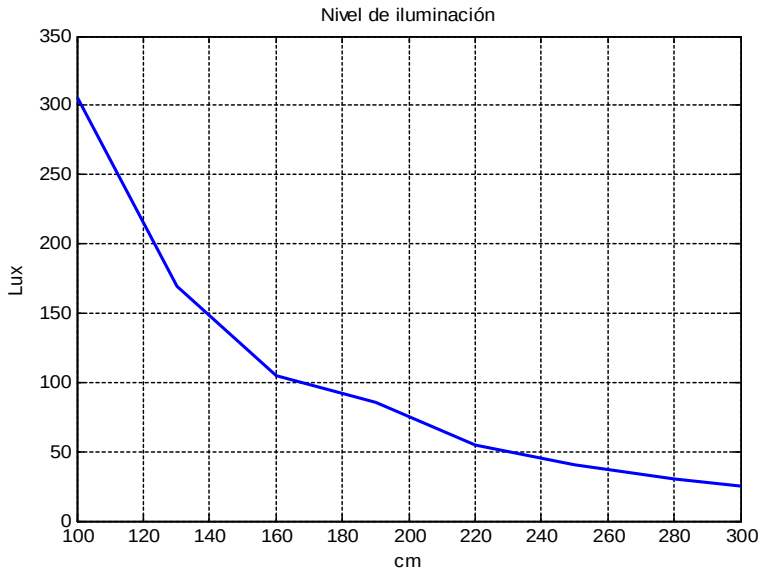


Figura 25: Nivel de iluminación

Por un lado la distancia es independiente de la velocidad, pero por otro lado el transmisor y receptor deben de estar dentre del rango de alcance. Cabe destacar que se debe poner el umbral del arduino con los niveles de iluminacion adecuados para funcionar en diferentes ambientes. Por lo tanto, si se requiere más distancia, se ha de usar un led con mayor potencia.

5. Presupuesto

El coste aproximado de los principales componentes del sistema es el siguiente:

Componente	Precio
Arduino Uno	20€
Raspberry pi	35€
Precision light sensor 1127	5€
Cámara raspberry	20€
Otros	5€
Total	85€

Tabla 2: Presupuesto

6. Conclusiones y futuro desarrollo

Para concluir, este proyecto demuestra que se puede diseñar un enlace uplink VLC mediante una cámara. Con esta sincronización se ha llegado a 25 bps pero con una tasa de error que se debe mejorar, por ejemplo con algún código corrector.

Usar un sistema operativo en tiempo real debe ser mucho mejor para sincronizar los dispositivos. Además, como se ha mencionado anteriormente, una cámara con una velocidad de fotogramas mayor implica más velocidad en transmisión de información.

Un factor importante a considerar cuando se elige una camara es su estabilidad y consistencia. Esto quiere decir que se necesita un 'frame rate' consistente debido a la importancia de los tiempos requeridos para la comunicación.

Para acabar, no hay que olvidar que las plataformas de Raspberry Pi y arduino son componentes de bajo coste diseñados para la educación. Son capaces de diseñar un enlace VLC pero no pueden alcanzar velocidades como otras tecnologías como por ejemplo el Wi-Fi.

Por lo tanto, el proyecto abre una puerta a futuros desarrollos para mejorar sus prestaciones. Se ha de poder sincronizar mejor la cámara con el transmisor así como se queria hacer originalmente. Ejemplos de acciones que se pueden realizar para mejorar el proyecto incluyen:

- Usar plataformas de hardware y software más optimizadas
- Usar más Leds para incrementar la velocidad
- Usar Leds de distintos colores.
- Usar más de una cámara

Bibliografia

[1] Raspberry Pi Organization, https://www.raspberry.org. Recuperado en 2017 y 2018

[2] Python Organization, https://www.python.org. Recuperado en 2017 y 2018

[3] Arduino, https://www.arduino.cc Recuperado en 2017 y 2018

[4] Schmid, S., Corbellini. G., Mangold, S. & Gross, T. R. Continuous synchronization for LED-to-LED visible light communication networks.3rd International Workshop in Optical Wireless Communications,Funchal,Isla Madeira,Portugal,doi:10.1109/IWOW.2014.69507, (2014, Sept. 17–17).

[5] Green, Jonathan, et al. "Camping in the Digital Wilderness: Tents and Flashlights as Interfaces to Virtual Worlds." CHr02 extended abstracts on Human factors in computing systems. ACM, 2002.

[6] 22 de enero de 2012, Simple linear MOSFET Dimming Circuit +PWM, https://laserpointerforums.com/f51/simple-linear-mosfet-dimming-circuit-pwm-70812.html

[7] Libreria Picamera, https://picamera.readthedocs.io Recuperado en 2017 y 2018

[8] T. Cevik, S. Yilmaz, "An overview of visible light communication systems", *ArXiv Prepr. ArXiv151203568*, 2015 .

[9] J. Li, A. Liu, G. Shen, L. Li, C. Sun, and F. Zhao. Retro-VLC: Enabling Battery-free Duplex Visible Light Communication for Mobile and IoT Applications. In Proc. of the ACM HotMobile, páginas 21–26, 2015.

[10] MIPI_Alliance_Specification_for_Camera_Serial_Interface_2__CSI_2_. Recuperado en https://es.scribd.com/document/325741962/MIPI-Alliance-Specification-for-Camera-Serial-Interface-2-CSI-2-pdf

[11] FreeRTOS, https://www.freertos.org/ Recuperado en 2017 y 2018

[12] ChibiOS, http://www.chibios.org/dokuwiki/doku.php Recuperado en 2017 y 2018

[13] OpenCV, https://opencv.org/ Recuperado en 2017 y 2018

[14] Xenomai, https://xenomai.org/ Recuperado en 2017 y 2018

[15] Sistemas operativos en tiempo real. Recuperado en http://isa.uniovi.es/docencia/TiempoReal/Recursos/temas/sotr.pdf

Anexos:

Se ha proporcionado un zip con el código utilizado.