PROYECTO FINAL DE CARRERA

SISTEMA DE COMUNICACIÓN ÓPTICO NO GUIADO BASADO EN RASPBERRY PI

(RASPBERRY BASED OPTICAL WIRELESS COMMUNICATION SYSTEM)

Estudios: Ingeniería de Telecomunicaciones

Autor: Rubén Dario López Castillo

Director/a: José A. Lázaro / Adolfo Comerón

Año: 2016

Índice general

Agradecimientos

A mis padres:

Iván y Nancy por el apoyo prestado durante toda mi etapa universitaria.

A los directores del proyecto:

Jose A. Lázaro y Adolfo Comerón, por el invaluable apoyo prestado durante la realización del presente trabajo.

Resum del Projecte

Aquest projecte de final de carrera té com a objectiu dissenyar i construir un enllaç de telecomunicacions òptic no guiat, mitjançant la utilització d'elements de baix cost i una Raspberry pi a cada extrem de l'enllaç, com a font d'informació i receptor final respectivament.

El proces seguit pel enllaç de telecomunicació òptic no guiat consisteix en enviar les dades contingudes en un fitxer a través de llum infraroja, generada per un díode làser semiconductor, i recuperar-les mitjançant un fotodíode, amb l'atmosfera com a canal de transmissió.

El senyal es codificat i modulat mitjançant polsos PPM a la raspberry de transmissió i enviat a través del làser, per ser demodulat i descodificat en el moment de la recepció en la segona raspberry pi. Un cop realitzat aquest procés, el senyal es recuperat per un fotodíode el qual transforma la potència òptica rebuda en un senyal electrònic. Aquest senyal es transmet al pin físic de la raspberry pi per així poder procedir a la seva lectura.

La modulació, codificació, demodulació i descodificació es realitza mitjançant un prototip de protocol de comunicació programari que, programat en llenguatge C, realitza tot el procés necessari per convertir cada caràcter del fitxer de text en la seva forma binària i a partir d'aquí, convertir aquests bits en forma PPM i realitzar el procés d'enviament i recepció.

Resumen del Proyecto

El presente proyecto final de carrera tiene como objetivo diseñar y construir un enlace de telecomunicaciones óptico no guiado con elementos de bajo coste y con una Raspberry pi en cada extremo del enlace, como fuente de información y receptor final respectivamente.

El proceso seguido por el enlace de telecomunicación óptico no guiado consiste en enviar los datos contenidos en un fichero de texto a través de luz infrarroja, generada por un diodo láser semiconductor, y recuperarlos mediante un fotodiodo, teniendo la atmosfera como canal de transmisión.

La señal es codificada y modulada mediante pulsos PPM en la Raspberry de transmisión, enviada por el láser y demodulada y decodificada en recepción por la segunda Raspberry pi. La señal es recuperada por un fotodiodo el cual transforma la potencia óptica recibida en una señal electrónica, esta señal es llevada al pin físico de la Raspberry pi para luego proceder a su lectura.

La modulación, codificación, demodulación y descodificación se realiza mediante un prototipo de protocolo de comunicación software programado en lenguaje C. Este protocolo realiza todo el proceso necesario para convertir cada carácter del fichero de texto en su forma binaria y a partir de ahí convertir estos bits en forma PPM y realizar el proceso de envío y recepción.

Abstract

The aim of this project is to design and construct a low cost elements Raspberry pi based optical wireless communication system.

The process followed by this telecommunication system consists in sending data from a text file through infrared light, generated by a diode laser semiconductor, and recover it with a photodiode, using atmosphere as transmission channel.

The signal is encoded and modulated in PPM pulses in the Raspberry pi that works in transmission, and then the signal is sent through the laser and demodulated and decoded in the reception module by the second Raspberry pi. Signal is recovered by a photodiode which transforms the received optical power into electronic signal. The recovered signal is brought to the Raspberry pi pin and is ready to be read.

Modulation, encode, demodulation and decode is made with a software communication protocol prototype, programmed in C language. This protocol does the needed process to convert every character of the text file into its binary representation to be able to convert them in a PPM form and do the send and receive process.

1. Introducción

1.1 Contexto del proyecto

El proyecto nace de la idea del profesor Jose Lázaro de intentar un enlace óptico no guiado entre dos Raspberry pi. Esto viene a raíz de que en una asignatura de master se les daba la opción a los alumnos de realizar este tipo de enlaces por medio de elementos ópticos o elementos WiFi, y ninguno de los estudiantes eligió la opción de óptica no guiada. Bien es cierto que el mercado ofrece mayor cantidad de posibilidades respecto a módulos de Wireless en emisión y recepción con sus respectivos protocolos de actuación, mientras que por la parte óptica no existe ese mercado de módulos, ni mucho menos protocolos diseñados para tañes fines a nivel educativo al menos.

1.2 Objetivos

En este Proyecto Final de Carrera (PFC) se abordan los siguientes objetivos principales:

El primero está relacionado con la enseñanza y aprendizaje proporcionado por la escuela durante estos años de carrera y consiste en profundizar en conocimientos en el tema de las comunicaciones ópticas. Ello se llevará a cabo mediante el uso de componentes ópticos y optoelectrónicos tales como láseres, fotorreceptores y lentes.

El segundo también de carácter formativo, consiste en adquirir conocimientos relacionados con un controlador de última generación como es la Raspberry Pi. Mediante la aplicación de conocimientos de programación y control de microcontroladores.

El tercero es desarrollar un enlace de telecomunicación óptico controlado por Raspberry Pi el cual permita aplicar los conocimientos anteriormente mencionados.

1.3 Estructura de la memoria

La memoria se constituye de cinco capítulos, cuatro de los cuales desarrollan la temática tratada en este proyecto, tanto teórica cómo práctica el primero de ellos hace una introducción descriptiva del contenido del documento.

Los capítulos dos y tres constituyen una parte más teórica en cuanto a teoría aplicada en el proyecto y los elementos del mismo.

El cuarto capítulo trata la temática más concreta de lo trabajado. Desarrollando una parte principal del proyecto, la implementación del protocolo de comunicación.

El capítulo cinco trata a fondo el enlace óptico hecho para el proyecto, la parte hardware del mismo.

2. Fundamentos para la transmisión de pulsos digitales. Modulación PPM.

2.1 Modulación PPM (Pulse Position Modulation)

La modulación PPM (Pulse Position Modulation) es una modulación digital que se caracteriza por transmitir los datos en pulsos cortos. Estos pulsos tienen todos la misma duración y amplitud, siendo el único parámetro cambiante la posición de cada pulso en un *slot* de tiempo dado.

Una representación de los bits de datos enviados mapeados a los símbolos transmitidos para cada uno de ellos sería por ejemplo [1]:

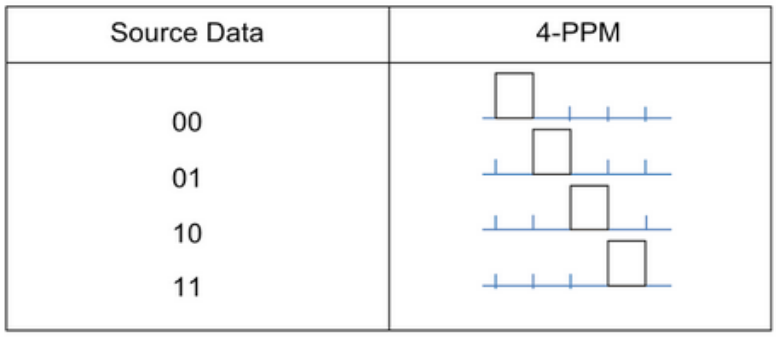


Figura 1: Mapeo de símbolos para una 4-PPM

En una modulación Q-PPM un símbolo corresponde a B=log2 Q bits. LA duración del símbolo Ts es dividida en Q slots de tiempo de duración T=Ts/Q y un pulso óptico es enviado en uno de estos Q slots. [ART 1]. La figura 1 muestra un ejemplo de una 4-PPM. Dado que esta modulación contiene un pulso cada Q slots, esta tiene un *duty cycle*[1] de 1/Q y un *peak-to-average power ratio* (PAPR)[2] de Q. De esta manera es posible variar Q para hacer un compromiso flexible entre la eficiencia de potencia y la eficiencia de ancho de banda.

[1] Duty cycle: relación entre la duración del pulso total y el pulso activo.
[2] PAPR

Así pues en recepción no será necesario un umbral de decisión de potencia para determinar los datos recibidos sino que, teniendo en cuenta un compromiso con el sincronismo, se determina el dato recibido en función de la posición en que es leído. A cambio de evitar las decisiones respecto las potencias recibidas, el uso de PPM requiere una señal que indique el momento en el cual se debe iniciar el proceso de intercambio de datos, de lo contrario se produciría un error en las lecturas de datos.

El hecho de no depender de la potencia recibida para determinar los datos leídos hace que este tipo de modulación sea idónea para transmisiones en las cuales, los dispositivos de emisión usan picos de potencia pequeños de manera que estos no suponen un gran peligro para quienes lo utilizan.

2.2 PPM en FSO (Free Space Optics)

Pulse Position Modulation (PPM) es una técnica de modulación que básicamente sirve para incrementar la eficiencia de transmisión en sistemas FSO (Free Space Optics). PPM es una técnica de modulación ortogonal en banda base que ha sido estudiada extensamente en comunicaciones ópticas por su gran eficiencia de potencia comparada con otras técnicas de modulación en banda base [1]. Pero, a cambio de estas grandes prestaciones en potencia presenta un incremento en el ancho de banda requerido y una mayor complejidad de diseño y manejo.

El factor de superioridad en eficiencia de potencia hace que se ajuste muy bien a dispositivos de mano que requieren un menor consumo. Es por esto que en comunicaciones punto a punto que usan ordenadores portátiles y miniordenadores hacen uso de esta técnica de modulación de manera muy habitual [1].

2.3 Explicación del diseño de PPM para el enlace.

La señal PPM utilizada en este proyecto es una señal binaria de posición de pulsos o 2-PPM, siendo esta la elegida entre las Q-PPM posibles, debido a que se trabajara con señales binarias y solo será necesario enviar dos valores de bit “1” o “0”.

Esto quiere decir que solo hay dos posiciones en las que se puede encontrar el pulso modulado. Para el “1” lógico se activará el primer slot de tiempo y el segundo slot para la modulación del “0” lógico.

Estas señales se han programado mediante código y su periodo es variable en función de las necesidades y/o las prestaciones del enlace.

La modulación de realiza activando el GPIO elegido de la Raspberry pi a su valor alto, de 3.3V, durante un tiempo predeterminado que tendrá que ver con el periodo elegido para la señal.

3. Componentes de un enlace óptico no guiado en un sistema de telecomunicaciones

3.1 Introducción

En este capítulo se explican las características principales de los elementos ópticos y optoelectrónicos que forman parte de un enlace de telecomunicaciones óptico inalámbrico.

Se definirá la propuesta de enlace diseñado para este trabajo y los elementos que forman parte de dicha propuesta.

3.2 Componentes de un enlace óptico no guiado

Un enlace cualquiera se compone de un emisor, un canal y un receptor. El sistema electrónico genera una señal de información. Esta información se modula a un dispositivo emisor de luz, el cual genera una señal luminosa con longitud de onda λ. La señal se transmite a través de la atmosfera hacia el receptor, en donde, mediante un fotodiodo se transforma en una señal electrónica, que es procesada por un sistema electrónico de recepción que recupera la información.

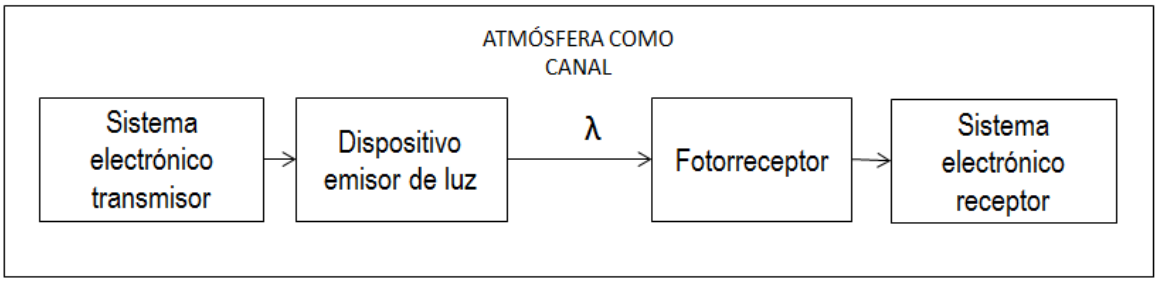


Figura 2. Diagrama de bloques simplificado de un enlace de telecomunicaciones óptico no guiado.

El sistema transmisor se encargará de modular la información de manera que el dispositivo emisor de luz genere una portadora luminosa con longitud de onda λ. Esta señal se transmite por el canal hacia el receptor, donde un fotodiodo la transforma en una señal eléctrica procesada por el sistema controlador en recepción.

3.2.1 Características básicas de rendimiento de los emisores de luz

- Longitud de onda

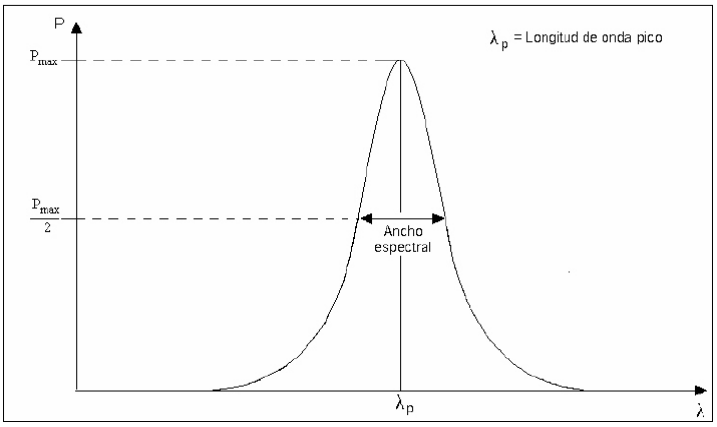


Figura 3. Espectro de un emisor de luz

La longitud de onda de un emisor de luz viene definida por la longitud de onda de pico, ya que el emisor de luz abarca un rango de frecuencias y no sólo una de ellas. La longitud de onda de pico es aquella donde se concentra la mayor parte de potencia como se muestra en la figura.

- Ancho espectral

Un emisor de luz esta idealmente diseñado para emitir en una sola longitud de onda, pero en la práctica se produce la emisión en un rango que se centra en la longitud de onda de pico. Se define el ancho espectral como la diferencia en longitud de onda entre los dos puntos en que la potencia de la señal ha caído hasta un 50%, como se muestra en la figura 2.

- Potencia

La potencia es la energía óptica a la salida del emisor de luz. Éste parámetro es de suma importancia ya que de éste depende la distancia máxima del enlace óptico.

- Tiempo de respuesta

Es el tiempo que tarda el emisor en responder ante un cambio en la señal de entrada eléctrica. Usualmente se toman como puntos de transición el 10% y el 90% de la potencia máxima de la señal. Esta característica es una de las más importantes ya que determina el ancho de banda del emisor de luz y consecuentemente del sistema en general

3.2.2 Fotodetectores

El objetivo de un receptor óptico es convertir la señal óptica nuevamente en señal eléctrica y de esta manera obtener los datos transmitidos por medio de esta onda de luz. El componente básico de un receptor óptico es un fotodetector, el cual convierte la señal óptica en señal eléctrica mediante el efecto fotoeléctrico.

3.2.3.1 Características principales de rendimiento en los fotodetectores

Con el objetivo de caracterizar estos elementos, se introducirán conceptos como responsividad, efecto cuántico, tiempo de subida y ancho de banda, ya que todos estos conceptos son comunes en todos los fotodetectores [LIBRO2]

- Responsividad
- Efecto cuántico
- Tiempo de respuesta
- Respuesta espectral

3.2.3.1.1 Responsividad y efecto cuántico

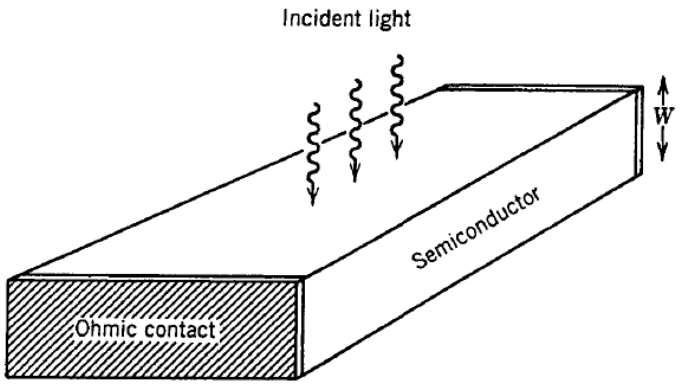


Figura 4: Bloque semiconductor usado como fotodetector. [Fuente: Fiber-Optic Communication systems. Govind P. Agrawal.]

Considerando la figura 2 como referencia. Si la energía *hv* de fotones incidentes en la banda prohibida, cada vez que un fotón es absorbido por el semiconductor se genera un par electron-vacio. Bajo la influencia de un campo eléctrico creado por un voltaje aplicado, los electrones y huecos son movidos de un lado al otro del semiconductor, teniendo como resultado un flujo eléctrico.

La fotocorriente I_p es directamente proporcional a la potencia óptica incidente Pin.

$$Ip = RdPin \tag{1}$$

Donde definimos Rd como la responsividad del fotorreceptor, a mayor responsividad más corriente es producida para una potencia de entrada dada. La responsividad es expresada en términos de (A/W).

La responsividad puede ser expresada en términos de eficiencia cuántica *ŋ*:

$$\eta = \frac{ratio\ de\ generación\ de\ electrones}{ratio\ de\ incidencia\ de\ fotones} = \frac{Ip/q}{Pin/hV} = \frac{hv}{q}Rd \tag{2}$$

Teniendo en cuenta lo obtenido en (1). La responsividad se puede expresar como

$$Rd = \frac{\eta q}{hv} \approx \frac{\eta \lambda}{1.24} \tag{3}$$

Donde λ = c/v es expresada en micrómetros. La responsividad de un fotodetector incrementa con la longitud de onda λ simplemente porque hay más fotones presentes para la misma potencia óptica [2].

3.2.3.1.2 Tiempo de subida, ancho de banda y corriente oscura

El tiempo de subida es definido como el tiempo que tarda la corriente en pasar de un 10 a un 90% de su valor final cuando por algún incidente hay un cambio brusco en la potencia óptica incidente. Este tiempo también dependerá del tiempo que le toma a los electrones y huecos para viajar a los contactos eléctricos, y también a su vez del tiempo de respuesta del circuito eléctrico usado para procesar la fotocorriente [2].

El tiempo respuesta de vendrá dado por

$$Tr = (\ln 9)Trc \tag{4}$$

Donde Trc es el tiempo constante de un circuito RC.

Por lo que el tiempo de respuesta de un fotodetector puede ser escrito de manera completa como

$$Tr = (\ln 9)(Ttr + Trc) \tag{5}$$

Donde Ttr es el tiempo de transito de los electrones y Trc es el tiempo constante de un circuito equivalente RC.

Esta característica está directamente relacionada con el ancho de banda en el que puede operar el detector. El ancho de banda de un fotodetector queda definido como [2]

$$\Delta f = \frac{1}{2\pi(Ttr+Trc)} \tag{6}$$

El tercer parámetro importante en un fotodetector junto con el ancho de banda y la responsividad es la corriente de oscuridad (*Dark current)* $I_{d.}$

I_d es la corriente generada en un fotodetector en la ausencia de ninguna señal óptica. Como buena característica de un fotodetector, esta corriente oscura debería ser despreciable (I_d < 10 nA) [2].

3.2.3.2 Fotodiodo PIN

El fotodiodo PIN es un dispositivo que está formado por tres capas o *'layers',* una capa de material tipo n, una capa intrínseca y una capa de material modo p. Los fotones recibidos inciden en la capa intrínseca generando pares electrón-hueco. El fotodiodo PIN, y en general los fotodiodos, se polarizan de manera inversa para que el campo eléctrico generado acelere las cargas presentes en la zona intrínseca hacia los electrodos, generando una corriente eléctrica.

Algunas de las principales características de los fotodiodos pin de silicio, unos de los más comunes, y los utilizados en este proyecto.

Tabla 1. Características del fotodiodo PIN de silicio [2]

Parámetro	Símbolo	Unidad	Valor común en fotodiodo de silicio
Longitud de onda	λ	µm	0.4 – 1.1
Responsividad	R_d	A/W	0.4 – 0.6
Eficiencia cuántica	ŋ	%	75 – 90
Corriente oscura	I_d	nA	1 – 10
Tiempo de subida	Tr	ns	0.5 – 1
Ancho de banda	Δf	GHz	0.3 – 0.6
Voltaje bias	V_b	V	50 – 100

4. Implementación de un protocolo para comunicación óptica no guiada con Raspberry Pi como controlador.

4.1 Raspberry pi

Raspberry pi es ordenador de tamaño reducido diseñado originalmente para la educación. La idea de su creador Eben Upton era crear un dispositivo de coste bajo con el que se pudiese mejorar las habilidades de programación y el entendimiento

Figura 5. Dispositivo Raspberry pi su reducido tamaño

del hardware en un nivel pre-universitario. Pero debido a su pequeño tamaño y precio accesible, rápidamente hizo que investigadores, estudiantes y entusiastas de la electrónica adoptaran estos dispositivos para la realización de proyectos que requerían algo más que un microcontrolador básico, como puede serlo un arduino[3] [URL2].

El entorno de trabajo que se puede encontrar en estos dispositivos es linux. Y dentro de este entorno existen varios lenguajes de programación por medio de los cuales se puede definir el protocolo deseado para la realización de la tarea requerida por el proyecto.

[3] Arduino: Consiste en un circuito impreso con un microcontrolador y puertos digitales y analógicos de entrada/salida.[URL10]

En éste proyecto final de carrera, se usa la Raspberry pi como controlador en emisión y recepción del enlace óptico. Por medio del control de los pines o gpio[4] se generan tramas de bits para la transmisión de datos. El control de los gpio se hace mediante un control software de los mismos. Este control software ha sido creado de cero para este proyecto.

4.2.1 Características principales

Uno de los principales componentes utilizados en este proyecto final de carrera es el miniordenador Raspberry Pi. Algunas de las principales características se describen a continuación:

La Raspberry pi está basada en el microchip de Broadcom BCM2835 en los primeros modelos y BCM28365 para los segundo modelos de Raspberry. En este proyecto se trabaja con el segundo y el tercer modelo de Raspberry.

La Raspberry pi 2 usa el Broadcom2836 con el procesador ARM Cortex-A7, de 32 bits y cuatro núcleos trabajando a 900 MHz, con 256KB de cache compartida. Mientras que las Raspberry pi 3 usa el BCM2837 con el procesador ARM Cortex-A53, de 64 bits y cuatro nucleos trabajando a 1.2 Ghz, con 512 KB de cache compartida.

Ademas de ello, éste miniordenador esta equipado con elementos de hardware que hacen que la interacción con este aparato sea fácil y cómoda. Dispone de una salida de video HDMI, así como 4 puertos USB y un puerto ethernet. Mediante estos componentes se puede realizar un uso directo conectando el dispositivo a una pantalla y un teclado, o bien, realizando una conexión remota y trabajando desde otro ordenador. Además de ellos, la Raspberry pi 3 esta equipada con WIFI 802.11n a 2.4 GHz además del puerto ethernet. [URL3]

[4] Gpio: General-purpose input/output [URL1]
[5] Información del chip de Raspberry BCM2836: [URL4]

4.2.2 Gpio (General-purpose input/output)

Los Gpio pins son herramientas físicas integrades a los micro ordenadores con el objetivo de tenir una comunicación fácil con elementos externos a su circuito integrado.

Dentro de estas funciones que pueden hacer estos pines estan, en su función de outputs digitales, encender o apagar luces, motores u otros dispositivos digitales. En su función de input digital, se puede usar, por ejemplo para leer el estado de sensores o botones o de manera simultanea usando uno o más de ellos para realitzar comunicación directa on otros chips o modulos usando protocolos de nivel bajo, como es el caso de este proyecto.

BCM	WiringPi	Name	Physical		Name	WiringPi	BCM
		3.3v	1	2	5v		
2	8	SDA.1	3	4	5V		
3	9	SCL.1	5	6	0v		
4	7	1-Wire	7	8	TxD	15	14
		0v	9	10	RxD	16	15
17	0	GPIO. 0	11	12	GPIO. 1	1	18
27	2	GPIO. 2	13	14	0v		
22	3	GPIO. 3	15	16	GPIO. 4	4	23
		3.3v	17	18	GPIO. 5	5	24
10	12	MOSI	19	20	0v		
9	13	MISO	21	22	GPIO. 6	6	25
11	14	SCLK	23	24	CE0	10	8
		0v	25	26	CE1	11	7
0	30	SDA.0	27	28	SCL.0	31	1
5	21	GPIO.21	29	30	0v		
6	22	GPIO.22	31	32	GPIO.26	26	12
13	23	GPIO.23	33	34	0v		
19	24	GPIO.24	35	36	GPIO.27	27	16
26	25	GPIO.25	37	38	GPIO.28	28	20
		0v	39	40	GPIO.29	29	21
BCM	WiringPi	Name	Physical		Name	WiringPi	BCM

Figura 6. Disposición física y numeración de los GPIO Raspberry pi [URL5]

En la figura se encuentra la numeración y disposición física de los pines en la Raspberry pi.

Se puede diferenciar unos pines que tienen valores fijos de salida, como lo son los de 3,3V, 5V, y 0V. También unos pines numerados que no tienen una función fija y pueden ser utilizados por el usuario de manera libre. Otros de los pines están configurados previamente para que, mediante el uso de unas librerías específicas,

se ponga en práctica el uso de protocolos extendidos de bajo nivel como SPI6, "I2C"7, o Serial UART8.

Dado que no existe ningún tipo de protección física, es importante tener en cuenta las limitaciones electrónicas de los pines para evitar malfuncionamiento y/o daño.

4.2.3 Lenguajes de programación aplicados en Raspberry pi. Benchmarking.

Existen diferentes herramientas de programación por medio de las cuales hacer que el controlador haga lo que el usuario quiere que haga. Cada uno de ellos tiene unas características diferentes y aporta ventajas y desventajas dependiendo el punto de vista desde el cual se mire.

Entre los lenguajes más utilizados se encuentran: Shell, Python, Ruby, C, C++, C#, java. Todos estos tienen su propia forma de programar, lenguaje, enfoque de la programación y entorno en el cual se desarrolla.

La siguiente tabla mostrará diferentes lenguajes mediante los cuales se puede generar una onda cuadrada por medio de un pin de Raspberry pi, además de las librerías utilizadas en cada uno de los casos, y la máxima frecuencia de onda generada con cada uno de estos lenguajes.

Tabla 2. Frecuencia de onda cuadrada según el lenguaje

Lenguaje	Librería	Frecuencia onda cuadrada
Shell	/proc/mem acces	2.8 kHz
Python	RPi.GPIO	70 kHz
Python	wiringpi2 bindings	28 kHz

[6]SPI: Serial Peripherial Interface [URL7]
[7] I2C: Protocolo serie para conectar dispositivos de baja velocidad como microcontroladores [URL8]
[8] Serial UART: Serial Universal Asynchronous Receiver/Transmitter [URL9]

Ruby	wiringPi bindings	21 kHz
C	Librería nativa	22 MHz
C	BCM 2835	5.4 MHz
C	wiringPi	4.1 – 4.6 MHz

Se puede ver que el lenguaje de programación que da un mejor resultado en términos de frecuencia para una onda cuadrada es el C desde su librería nativa. Ahora bien, para poder usarlo sin ningún tipo de librería como la BCM 2835 o wiringPi, se debería crear todo tipo de funciones de escritura, lectura, generación y control, así que en principio se podría generar una onda de unos 5MHz.

En éste proyecto se trabaja con lenguaje C usando la librería wiringPi y su numeración y funciones de control para realizar el envío y la recepción por medio de los pines de la Raspberry pi.

4.2 Contexto del protocolo

Se puede crear un sistema de telecomunicación basado en algún protocolo dado previamente, como lo puede ser el utilizar la herramienta de wifi proporcionada por un dispositivo concreto como una Raspberry pi, y por medio de esa herramienta crear una comunicación mutua entre dos de estos elementos. Pero también existe la posibilidad crear un sistema de comunicaciones sin estar basado en ningún protocolo ya establecido, basándose simplemente en la creación de unas reglas adecuadas para el tipo de sistema utilizado y la información a ser tratada.

Para el enlace de comunicaciones, basado en instrumentos ópticos de bajo coste y controlado por dos terminales Raspberry pi, que se trata en este trabajo, se encontró la dificultad de no encontrar ningún protocolo ya existente que se adecuara o bien a las características del enlace o bien a la forma en que se ha tratado la información para conseguir éxito en el envío, creando esto problemas en la detección de la señal incluso en un prototipo inicial de comunicación por cable.

Esta dificultad radicaba en la imposibilidad de generar un sincronismo entre el emisor y el receptor. No había una herramienta ya diseñada que permitiera realizar el envío de datos modulados por medio del láser y a la vez que fuese capaz de establecer unas premisas de comunicación para garantizar la correcta recepción de los datos. Los datos se enviaban de manera correcta pero no había manera de encontrar una recepción con éxito.

Es por esto que se tomó la decisión de crear un protocolo que ayudará a la realización de la comunicación. Este prototipo de protocolo se basa en la creación de una trama específica y unas funciones programadas para ser realizadas en los momentos oportunos con el fin de garantizar el envío de la información de un extremo al otro del enlace

4.3 Explicación del protocolo

El protocolo se basa en el esquema de transmisión completa a nivel de tratamiento de la información, trabajado sobre lenguaje C:

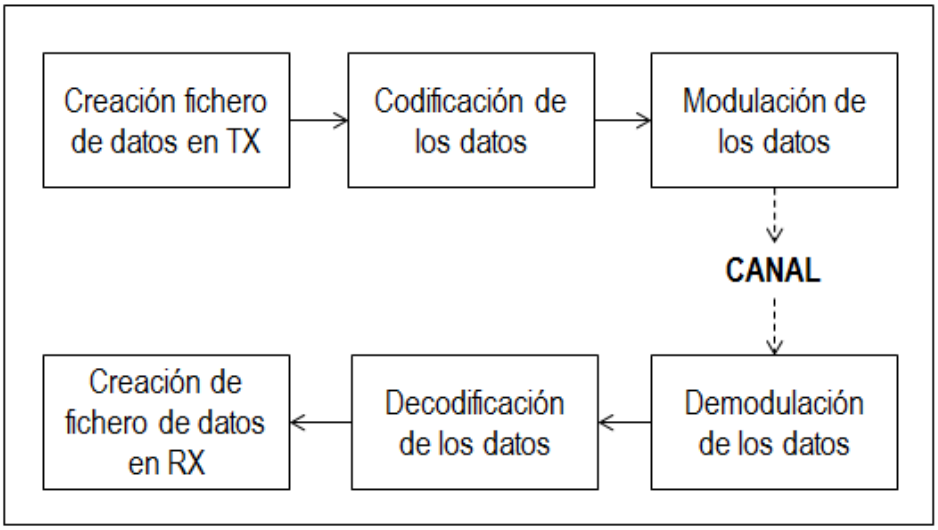


Figura 7. Diagrama de bloques del tratamiento de los datos

Lo primero a tomar en cuenta es la forma de la trama a ser enviada para cada dato. Cada envío de un carácter de datos se compondrá de la siguiente forma de trama:

Bit de inicio	Datos	Bit de paridad	Bit de parada	Tiempo de espera

Figura 8. Forma de trama del protocolo

El bit de inicio da comienzo a la trama, siempre es un bit “1”, se utiliza para que el receptor sepa que se da inicio a la comunicación y debe empezar a leer datos.

Los bits de datos son 8 bits, que consisten en el carácter que se está enviando usando el código ASCII, forma que utiliza por defecto el lenguaje C al usar una variable del tipo ‘char’.

El bit de paridad es una suma binaria de todos los bits enviados, para confirmar en recepción que los datos enviados son correctos y coinciden con lo que pretendía enviarse. Es un bit de control opcional en la trama.

El bit de parada es un bit que siempre es “0”. Es un bit opcional usado en este tipo de tramas para dar tiempo al receptor a procesar los datos.

Por último, el protocolo fue definido dejando un tiempo de espera en “low” siendo éste un tiempo prudente para que el receptor pueda comenzar a recibir otro dato.

A partir de la forma de trama definida anteriormente se realizara por medio de unas funciones programadas para el envío (ver código en el apéndice). Estas funciones siguen una lógica simple para el envío de cada uno de los caracteres del fichero de datos.

Cada carácter leído del fichero de texto previamente creado que se va a enviar se convierte en un dato de 8 bits por medio de la forma ASCII de cada uno de ellos. Esto se podría definir como la codificación dada para los datos que se van a enviar. Después de esto, se realiza la modulación de los datos convirtiendo cada uno de los 8 bits obtenidos en la forma PPM definida para los “1” y los “0” lógicos, estos bits son escritos como unos y ceros lógicos a través del pin de la Raspberry pi, consiguiendo que el diodo laser realice la misma modulación hecha en los pines por medio del programa de envío.

Una vez los datos han pasado por la parte física del enlace y han sido adecuados para poder ser tratados por la Raspberry pi en recepción se procede a la lectura de los mismos.

Este proceso de lectura lo realiza una función que comienza a actuar una vez detecta un cambio de flanco generado por el primer bit, el bit de inicio. Una vez detectado este cambio se espera el tiempo necesario para completar un periodo de la señal, y después de esto un tiempo de desfase, para poder leer las señales en un punto estable.

Se lee cada uno de los bits por el pin de la Raspberry pi esperando los tiempos necesarios entre cada lectura para el procesamiento de los datos por parte de la Raspberry pi

Cada carácter que se va recibiendo se almacena en un buffer, y el contenido de este buffer se escribe en el archivo '.txt' que se crea en recepción.

5. Montaje del enlace óptico

5.1 Elementos utilizados en el montaje y función de los mismos.

5.1.1 Raspberry pi en emisión y recepción

Como controladores de envío y recepción de la información están dos dispositivos rapsberry pi, colocados en los extremos del enlace. Cada uno de ellos contiene un software sobre el cual se ha trabajado.

La configuración inicial de los dispositivos empieza en la carga de la imagen de software en la tarjeta de memoria y continúa hasta las eventuales actualizaciones o mejoras que pueda ir recibiendo por parte de los creadores del software.

En las dos Raspberry pi se ha instalado una imagen de 'Raspbian Jessie'9 además de un paquete con la librería 'WiringPi'. Esta librería permite un fácil acceso al control de los GPIO desde el lenguaje C, básicos para el control de la modulación de datos.

Los dispositivos son uno del modelo 2B y el otro del modelo 3.

Figura 9. Raspberry pi 2B usada en emisión

[9] Raspbian jessie: sistema operativo proporcionado por Raspberry pi para usar en sus terminales

Figura 10. Raspberry pi 3 usada en recepción

5.1.2 Diodo Laser

El diodo laser utilizado para realizar el envío de datos es el L850P010 de Thorlabs10. La elección de éste es basada simplemente en tratar con bajar potencias para reducir al máximo posible el peligro de daño que se pueda producir al usarlo, además de por su bajo coste y sencillo modo de implementación.

Las características básicas de éste diodo láser son:

Una potencia máxima de salida Po=10mW

Una longitud de onda típica de 850nm

L850P010

Figura 11: Diodo láser L850P010

Una descripción más detallada de las características y prestaciones del láser se pueden encontrar en el apéndice “Especificaciones de los elementos del enlace”

[10] Thorlabs: Empresa especializada en la creación de bloques láser y sistemas de fibra óptica.

5.1.3 Fotodiodo receptor

El fotodiodo receptor utilizado en el montaje es el fotodiodo de silicio FDS1010 de Thorlabs. Se ha elegido éste componente dado que es un receptor que se comporta bien para fuentes de luz pulsadas, así como su por gran superficie comparada con otras opciones en las que se requería un trabajo extra en la focalización de la luz enviada hacia una superficie de recepción mucho menor.

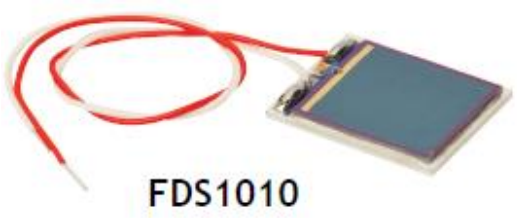


Figura 12. Fotodiodo de silicio FDS1010

El FDS1010 tiene un ancho espectral por el cual responde a potencias incidentes con longitudes de onda desde los 400 hasta los 1100nm con la siguiente curva de responsividad:

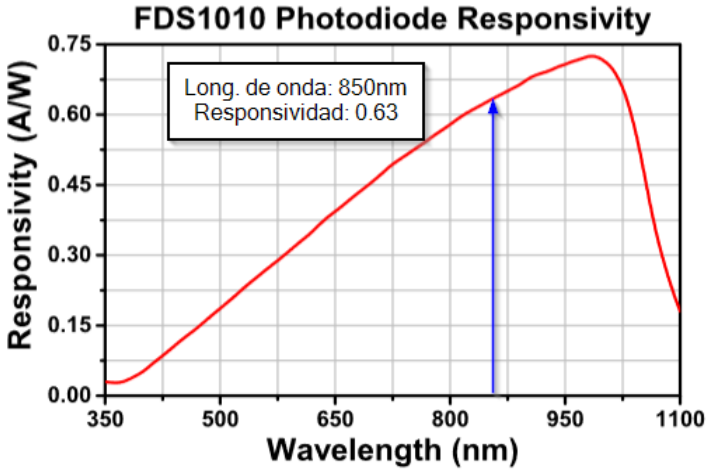


Figura 13. Responsividad del fotodiodo FDS1010 (Fuente Thorlabs)

De la curva se puede extraer el comportamiento que tiene el receptor ante una potencia incidente con una longitud de onda de 850nm como es la del láser incidente usado en el montaje.

Teniendo en cuenta el circuito de la figura 13, el cual contiene una resistencia de carga, Rload, que es escogida de tal manera que no se sobrepasen las corrientes máximas del fotodiodo para evitar un malfuncionamiento o daño.

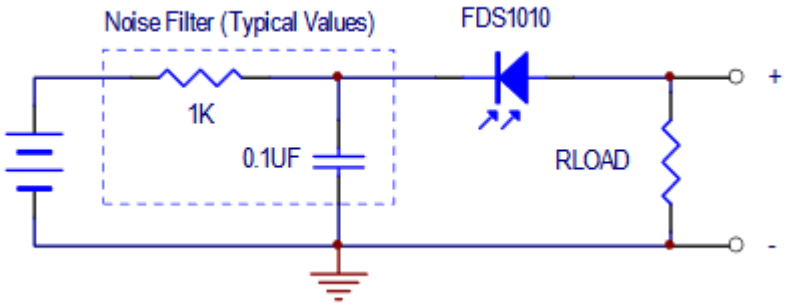


Figura 14. Circuito típico para la recepción. (Fuente Thorlabs)

A partir de esta configuración el voltaje de salida viene dado por:

$$Vo = P * \Re(\lambda) * Rload \quad (7)$$

Siendo R(λ)=0.63 según la gráfica de la responsividad del fotodiodo y P la potencia incidente recibida en la superficie del mismo.

Una descripción más detallada de las características y prestaciones del fotodiodo se pueden encontrar en el apéndice “Especificaciones de los elementos del enlace”

5.1.4 Circuito adaptativo recepción fotodiodo

El circuito usado para la adaptación de la señal recibida por el receptor se basa en el proporcionado por el fotorreceptor en sus especificaciones, un demodulador optoelectrónico.

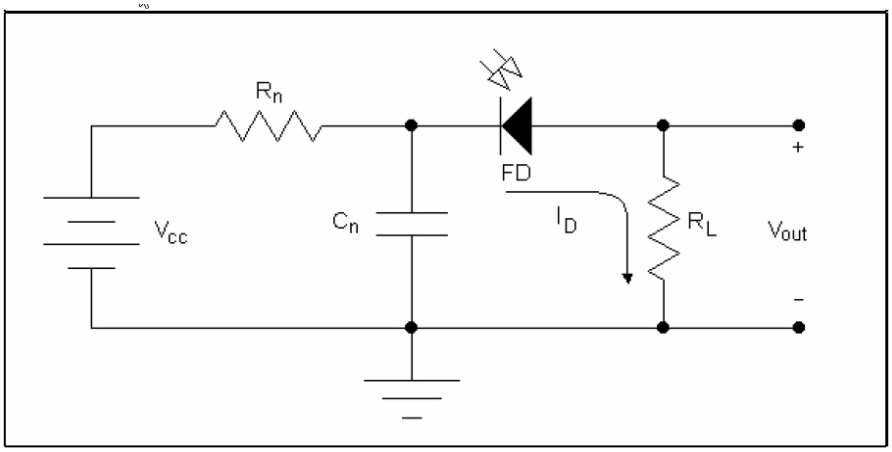


Figura 15. Diagrama del demodulador optoelectrónico

Éste convierte la señal óptica en señal eléctrica mediante el fotodiodo. La resistencia Rn y el capacitor Cn se usan como filtro de ruido. El fotodiodo FD convierte la señal óptica de información en una corriente eléctrica ID. Esta corriente circula a través de la resistencia de carga RL obteniendo un voltaje proporcional. EL voltaje de salida se puede determinar con la expresión (7).

5.2 Montaje inicial

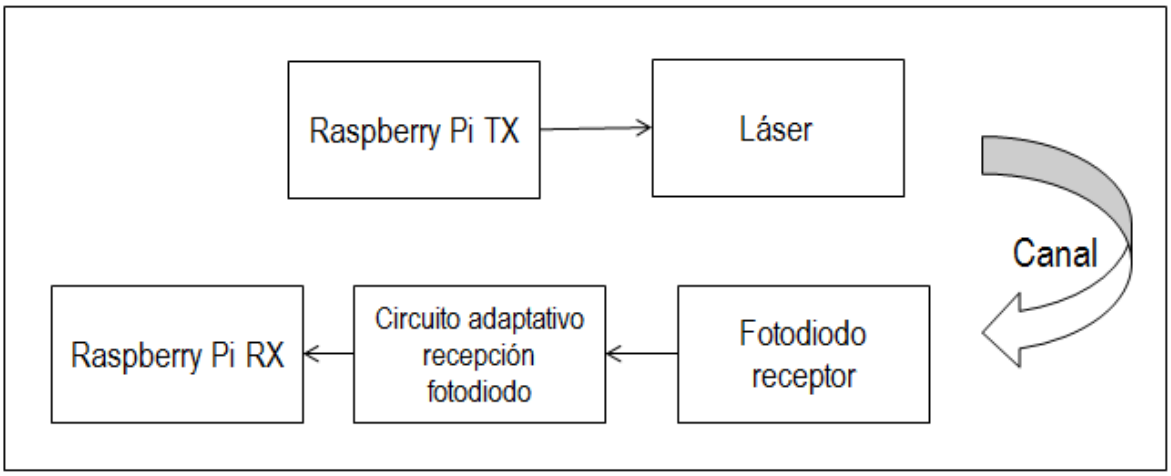


Figura 16. Diagrama de bloques el montaje

Después de escoger los elementos que formarían parte del enlace presentado en la figura se da paso a conocer por medio de las características de los elementos, una previsión del escenario que se encontrará en el momento de hacer la revisión práctica en el laboratorio.

El proceso de envío de datos se empieza desde la primera Raspberry pi, en la que se ha generado un fichero .txt con la información que se quiere transmitir. Este fichero será enviado bit a bit por medio del láser, pero antes de ello cada uno de estos bits pasará el protocolo de transmisión mencionado anteriormente.

Se ha programado que el pin haciendo on-off module el láser de manera que se pueda enviar la información por vía óptica no guiada. El láser proporciona una potencia óptica de salida máxima de unos 10mW a los 850nm, de los cuales y por las características de las pruebas que se harán se usará la mitad, ya que 10mW de potencia en un láser no visible podría generar un escenario de peligro innecesario.

La potencia emitida por el láser atraviesa el canal, que en éste caso no es otro que la atmósfera, donde hay que tener muy en cuenta las condiciones de realización del enlace y el entorno en el que se encuentra. Determinaremos que se realiza el enlace en un espacio cerrado en el cual hay una menor cantidad de contaminación lumínica que en un espacio abierto, ya que esto dificultaría el trato del espectro recibido y la detección de los datos podría ser más complicada debido al ruido añadido y el efecto que éste ruido pueda dar en la sincronía de los datos enviados respecto a los recibidos. El estudio del impacto generado por la contaminación lumínica en un enlace de datos óptico no guiado se podría tomar como una ampliación del presente proyecto.

Esos aproximadamente 5mW de potencia incidirán sobre un fotodetector de silicio que trabaja con diversas longitudes de onda entre las que se encuentran los 850nm del diodo láser. A los 850nm el fotodetector aplica una responsividad de valor 0.63 como se muestra en la figura 8, este dato nos deja la siguiente ecuación asumiendo que los 5mW inciden sobre el fotodetector siendo la distancia muy pequeña y la ecuación (7)

$$Vo = 5mW * 0.63 * Rload \tag{8}$$

Para poder realizar una detección en la Raspberry pi de debe obtener una señal con valores cercanos a los 3,3V teniendo en cuenta que por seguridad es mejor usar valores bajos antes que incidir con tensiones que puedan dañar la Raspberry pi ya que esta no tienen ningún control físico para evitar estas situaciones.

Con este cálculo y con el objetivo de encontrar una tensión de alrededor de 3.3V, y usando (8) encontramos una Rload=1047.6 Ω, que llevándola al valor comercial más cercano sería Rload=1kΩ, para el que el Valor de V_0=3,15V.

Con este valor de V_0 la Raspberry pi está preparada para realizar la lectura de los bits recibidos y así poder realizar la demodulación y decodificación de los datos.

5.3 Pruebas de laboratorio

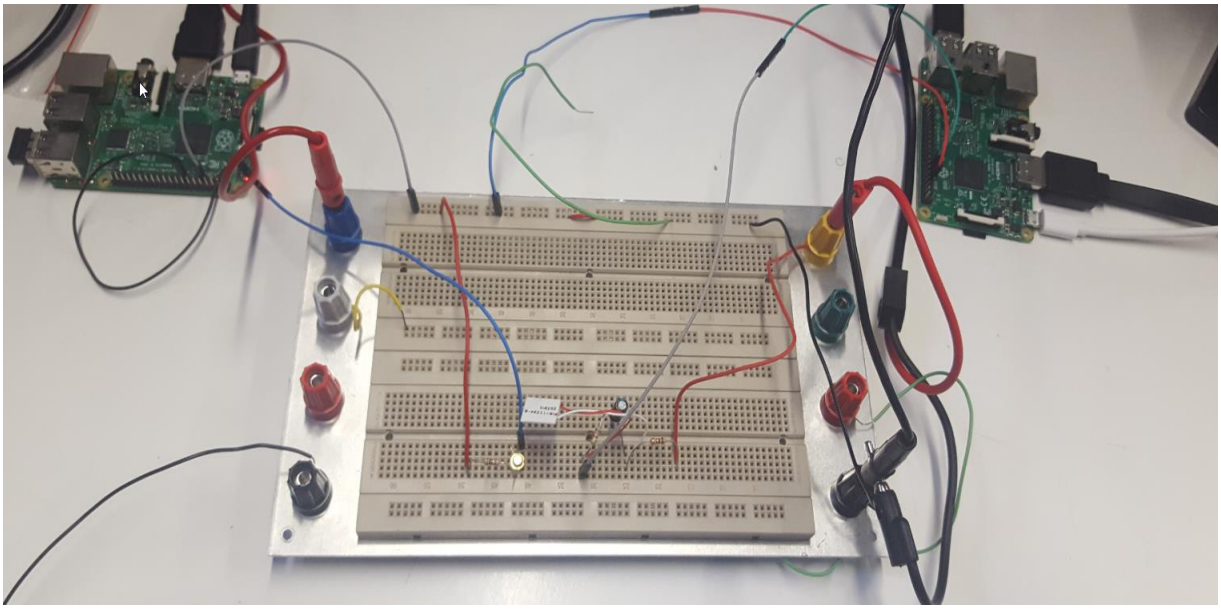


Figura 17. Montaje circuital completo del enlace

Se realiza un montaje completo con todas las partes operantes para el envío de datos de un extremo al otro del enlace. Por un lado, está la Raspberry pi de transmisión que modulará el láser incidente y por el otro lado estará el sistema receptor que le proporcionara los datos a la segunda Raspberry pi para que esta pueda reconstruir el fichero inicial de datos.

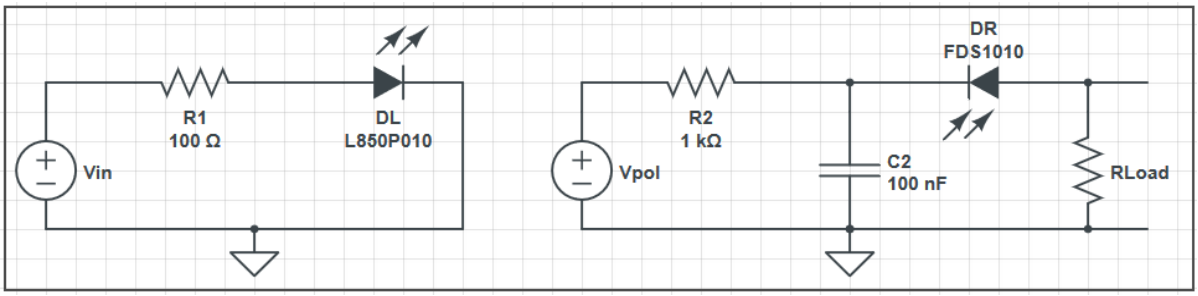


Figura 18. Configuración electrónica de los elementos del enlace

En este diagrama se pueden distinguir la parte de emisión y recepción del enlace. En la primera de ellas se encuentra la tensión dada por el pin gpio de la Raspberry pi y que por medio de una resistencia de control modula el diodo láser.

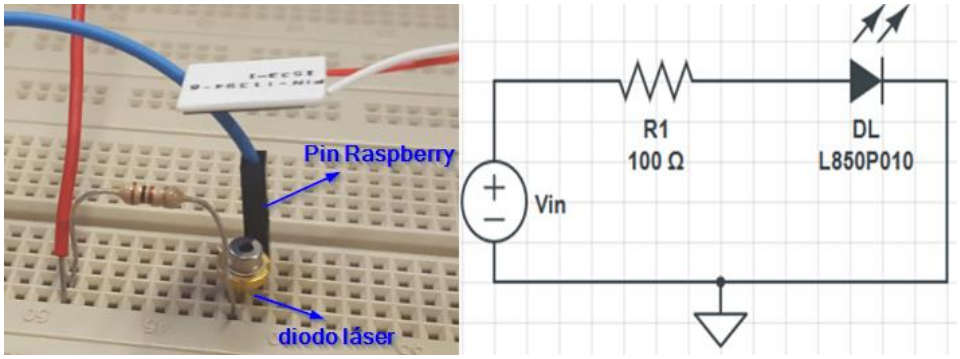


Figura 19. Circuito de polarización del láser

Esta tensión Vin llevada al circuito por el pin gpio de la Raspberry pi, por especificaciones del aparato debería rondar los 3.3 voltios, pero se puede comprobar que eso no ocurre, o no al menos es todos los momentos. Al modular el pin con una cierta velocidad se puede comprobar que esta tensión baja hasta los 2.7 voltios aproximadamente.

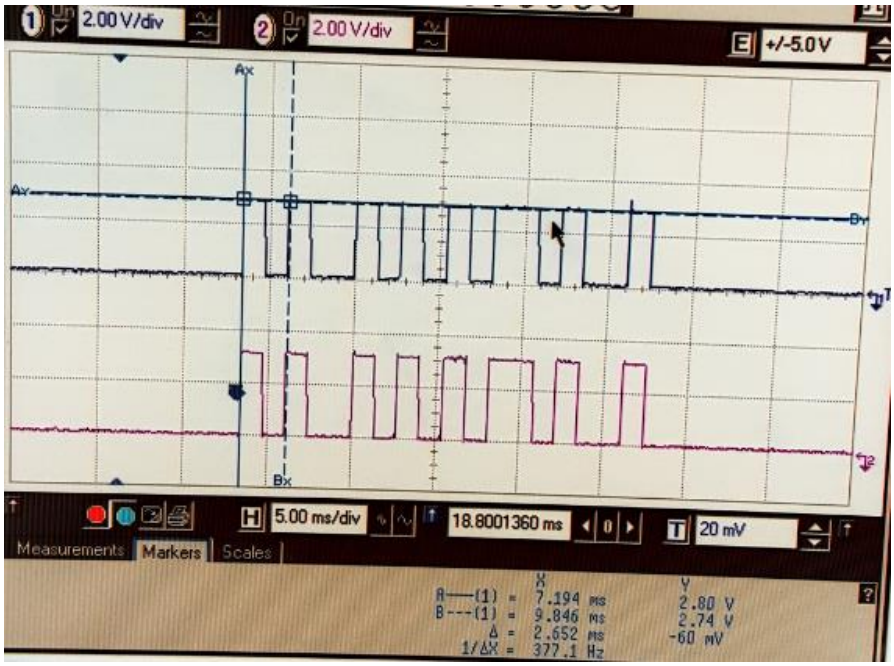


Figura 20. Tensión en emisión y recepción del sistema

En la figura 19 se puede ver la señal enviada por el pin de la Raspberry pi de transmisión y la señal recibida a la salida del fotodiodo receptor. Se puede ver que el valor de la tensión en el módulo de transmisión no alcanza el valor especificado en los datos de la Raspberry pi, esto puede ocurrir porque los efectos de carga del circuito y la impedancia intrínseca de los pines.

Se ha podido comprobar que el valor de la potencia se ve afectado por el valor bajo de tensión aplicado por el pin. Se ha realizado una prueba con un pin no modulable que proporciona una tensión constante de 3.3v obteniendo como resultado una medición de potencia de 5.5 mW, pero esta medición no es posible realizarla cuando se está modulando el láser ya que estos instrumentos calculan la potencia media medida y la edición es por tanto ineficaz.

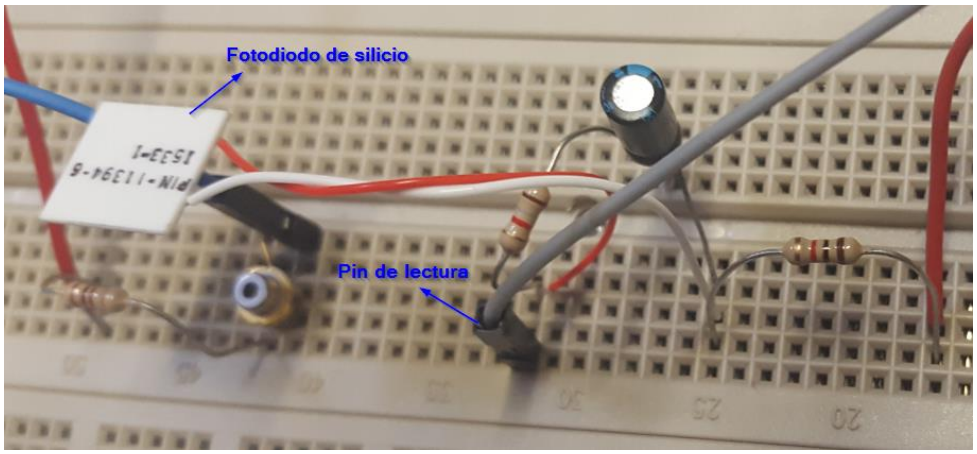


Figura 21. Circuito del demodulador optoelectrónico

Para poder ver esta potencia recibida se usa la expresión (7) con la tensión recibida, pero teniendo en cuenta el efecto de carga creado por el pin de la Raspberry pi en recepción. Para realizar éste cálculo se toman dos medidas de tensión con dos resistencias de carga diferentes.

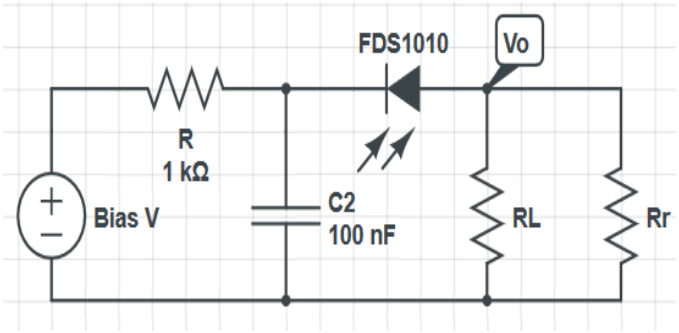


Figura 22. Demodulador optoelectrónico con el efecto de carga del pin

Para los siguientes valores obtenidos de tensiones en recepción con unas resistencias de carga dadas:

V_{01}= 2.77V V_{02}= 4.58V

R_{a1}= 1.8kΩ R_{a2}= 3,3kΩ

Haciendo la relación V_{01}/V_{02} siendo cada una de estas tensiones las descritas en (7) y siendo RL el paralelo de esta con Rr (RL//Rr)

$$\frac{V01}{V02} = \frac{Ra1\,(Ra2+Rr)}{Ra2(Ra1+Rr)} \tag{9}$$

Y con los valores anteriormente dados se obtiene Rr= 12kΩ

Aplicando (7) y sabiendo que la resistencia de carga será en este caso RL//Rr se obtiene que la potencia que está incidiendo en el fotorreceptor para un valor de aproximadamente 2,8V en el pin de transmisión es de:

$$P = \frac{Vo}{Ri\frac{Ra1*Rr}{ra1}} \approx 2.8mW \tag{10}$$

Para probar hasta donde se podía llegar en el enlace en términos de rapidez, se empezó en un periodo de 3ms de símbolo PPM, y se fue disminuyendo el periodo de la misma en saltos de 0.2ms

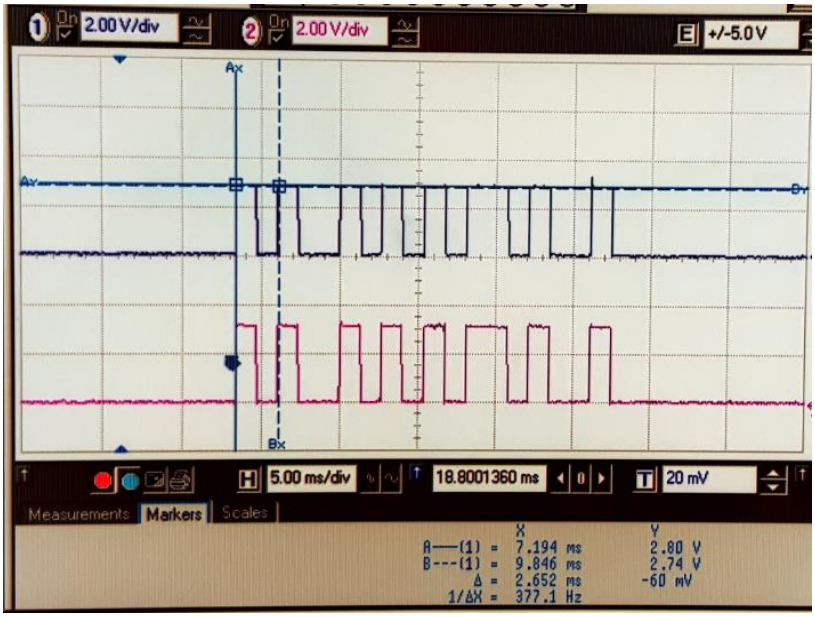


Figura 23. Señales enviada y recibida a 2.6ms de periodo de símbolo

Se puede ver que se mantiene la forma y periodo de las señales por lo que la lectura en recepción se realiza correctamente por parte del protocolo que utiliza la Raspberry pi en recepción.

Se continua bajando el periodo de los símbolos y continua habiendo recepciones correctas hasta la creación del fichero de recepción con los datos correctamente enviados. Los fallos se producen cuando se llega a un periodo de símbolo de 600 microsegundos.

Figura 24. Señal enviada y recibida para un periodo de 600 microsegundos

En la figura se puede apreciar que la señal presenta deformaciones en los flancos aunque se considera que esto no debería general un problema en la lectura de los datos ya que el protocolo espera un tiempo suficiente como para encontrar una señal estable. También se puede ver una variación en los flancos altos de algunos bits de la señal, aunque esto tampoco debería presentar un problema en la lectura de los datos ya que se ha podido comprobar que el pin de la Raspberry pi da un margen lo suficientemente amplio como para poder soportar estas variaciones y leer cualquier parte de este flanco como señal alta o “1” lógico.

Es por esto que determina que el problema que genera que no se realice una recepción correcta y por lo tanto una creación correcta del fichero de texto es intrínseco de la Raspberry pi. Es probable que el tiempo de proceso sea en alguno de los casos mayor que el tiempo que tarda en recibir los siguientes datos y por eso, aunque llega a crear un fichero de salida, este no lleva los caracteres correctos, es decir se producen errores en la lectura de los mismos mas no un colapso del programa.

Haciendo un estudio más a fondo se pusieron trazas en la función de lectura, para indicar si se producían errores en la lectura de uno o más bits. Se comprobó que existían errores de lectura en bits puntuales, aunque la función podía continuar y leer los siguientes bits. Es decir, la forma de onda y la velocidad eran aun soportables para el hardware de la Raspberry pero el software intentaba ir a mayor velocidad y no era capaz de conseguir lecturas correctas.

5.4 Líneas futuras

En este apartado quisiera comentar unes líneas de trabajo que podrían seguirse en éste proyecto a modo de ampliación del mismo.

- Como primera de ellas y para enlaces de mayor distancia en los cuales la potencia de que llegará al receptor fuese mucho menos previsible, usar un esquema circuital que use un amplificador con control automático de ganancia. Esto significa que para una tensión dada en un margen definido por las especificaciones del mismo, se encontrará la misma tensión en la resistencia de carga del fotorreceptor. De esta manera no habría problemas en detección relativos a falta de nivel de señal.

- Una segunda consideración sería en recepción, y para evitar los efectos de carga que pueda generarse en los pines de la Raspberry pi. Esto se podría conseguir poniendo una etapa intermedia entre la recepción y el pin de la Raspberry. El efecto que genera el amplificador operacional, “engaña” al fotodiodo haciéndole creer que no hay ninguna impedancia de carga.

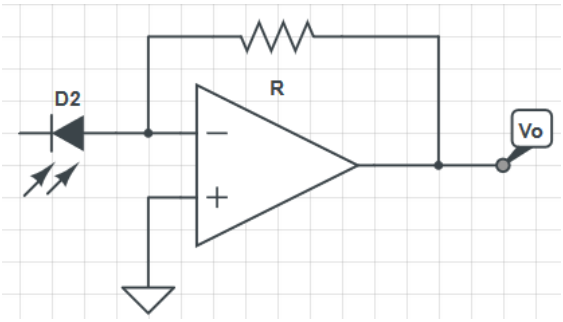


Figura 25. Etapa con amplificador de transimpedancia

$$Vo = -Responsividad * Popt * R \tag{11}$$

La manera de aumentar la señal sería aumentando la resistencia, aunque esto tendría un límite ya que para resistencias muy altas se genera un compromiso con la velocidad del sistema.

- También sería interesante modificar el protocolo de manera que este fuera más eficiente, ya que este prototipo, que sólo se trataba de una prueba de la viabilidad del mismo no lo es del todo, ya hace las funciones de controlador, modulador, demodulador y demodulador quizá se comportase de una manera más útil separando las tareas por módulos.
- Por último sería modificar los componentes o configuración del enlace para conseguir un sistema que pueda conseguir un control de errores directo. Un enlace full dúplex con la misma configuración actual en paralelo, añadiendo un sistema de control de errores en la función de recepción.

6. Conclusiones

Se ha diseñado y construido un enlace de telecomunicaciones óptico no guiado para la transmisión de un fichero de texto y se ha comprobado la viabilidad de la propuesta de protocolo de telecomunicación que se ha implementado para su realización.

Se comprobó que aunque no existan módulos completos de bajo coste en el mercado es viable realizar un enlace con elementos ópticos controlados con lógica de programación.

Se adquirieron conocimientos relacionados con el dispositivo Raspberry cómo las funciones de su sistema operativo, lenguajes de programación comúnmente usados en esta plataforma com ‘Python’. Sus puntos fuertes como portabilidad, sencillez de acceso y manejo del sistema operativo, similar al Linux, contrastan con las limitaciones electrónicas de sus elementos de comunicación de entrada y salida que son los pines.

Se consiguió realizar una transferencia de un archivo .txt de un extremo al otro del enlace usando los elementos ópticos.

7. Apéndices

7.1 Especificaciones de los elementos del enlace

Product Specification Sheet

THORLABS

Laser Diode

L850P010

Description

Thorlabs Ø5.6 mm, TO-18 can package discreet laser diode is a compact light source suited to many application. Our lasers are fully compatible with our entire line of Laser Diode and TEC Controllers as well as our selection of Laser Diode Mounts and Collimation Solutions.

Specifications

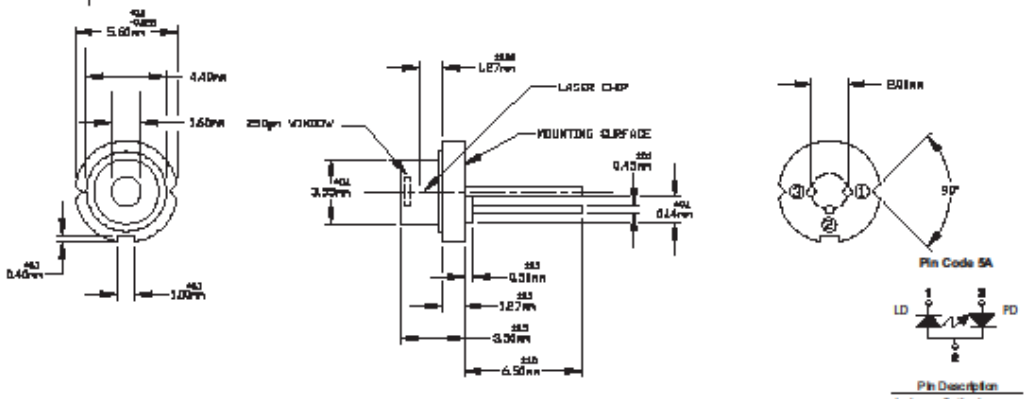
(P_o = 10 mW, T_c = 25 °C)

Specification	Symbol	Max
Optical Output Power, mW	P_o	10
LD Reverse Voltage, V	$V_{R(LD)}$	2
PD Reverse Voltage, V	$V_{R(PD)}$	30
Operation Case Temperature, °C	T_{op}	-10 to 50
Storage Temperature, °C	T_{STG}	-40 to 85

Specification	Symbol	Min	Typ	Max
Lasing Wavelength, nm	λ_c	835	850	865
Threshold Current, mA	I_{th}	10	25	40
Operation Current, mA	I_{op}	25	50	70
Operating Voltage, V	V_{op}	1.8	2.0	2.5
Monitor Current*, mA	I_m	0.05	0.3	1.0
Slope Efficiency, mW/mA	η	0.3	0.5	0.7
Beam Divergence, °	$\theta_{\parallel}$	8	10	12
Beam Divergence, °	$\theta_{\perp}$	25	30	40
Astigmatism**, µm	A_s		11	

*) $V_R = 5\ V$
**) $NA = 0.4$

Drawings



USA, Canada, & S. America
Thorlabs, Inc.
435 Route 206
Newton, NJ 07860, USA
Tel: 973-579-7227
Fax: 973-300-3600
www.thorlabs.com
email: feedback@thorlabs.com

Europe
Thorlabs GmbH
Hans-Böckler-Str. 6
85221 Dachau, Germany
Tel: +49-(0)8131-5956-0
Fax: +49-(0)8131-5956-99
www.thorlabs.com
email: Europe@thorlabs.com

UK and Ireland
Thorlabs LTD.
1 Saint Thomas Place, Ely
Cambridgeshire CB7 4EX, GB
Tel: +44 (0)1353-654440
Fax: +44 (0)1353-654444
www.thorlabs.com
email: sales.uk@thorlabs.com

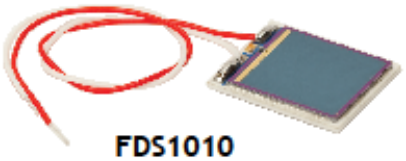
Scandinavia
Thorlabs Sweden AB
Box 141 94
400 20 Göteborg, Sweden
Tel: +46-31-733-30-00
Fax: +46-31-703-40-45
www.thorlabs.com
email: scandinavia@thorlabs.com

Japan and Asia
Thorlabs Japan Inc.
5-17-1, Ohtsuka
Bunkyo-ku, Tokyo 112-0012, Japan
Tel: +81-3-5977-8401
Fax: +81-3-5977-8402
www.thorlabs.jp
email: sales@thorlabs.jp

11910-S01 Rev C – October 7, 2010

THORLABS

Si Photodiode



FDS1010

Description

The Thorlabs FDS1010 photodiode is ideal for measuring both pulsed and CW fiber light sources, by converting the optical power to an electrical current. The detector is an un-housed ceramic wafer with an anode and cathode lead wires. The photodiode anode produces a current, which is a function of the incident light power and the wavelength. The responsivity $\Re(\lambda)$, can be read from the plot on the following page to estimate the amount of photocurrent to expect. This can be converted to a voltage by placing a load resistor (R_L) from the photodiode anode to the circuit ground. The output voltage is derived as:

$$V_o = P \times \Re \times R_L$$

The bandwidth, f_{BW}, and the rise time response, t_R, are determined from the diode capacitance, C_J, and the load resistance, R_L, as shown below. The diode capacitance can be lowered by placing a bias voltage from the photodiode cathode to the circuit ground.

$$f_{BW} = \frac{1}{(2\pi)R_L C_J},\ t_R = \frac{0.35}{f_{BW}}$$

Specifications

Specification		Value
Wavelength Range	λ	350 - 1100 nm
Peak Wavelength	λ_P	970 nm
Responsivity	$\Re(\lambda)$	0.725 A/W
Active Area		100 mm^2
Rise/Fall Time (R_L=50 Ω, 5 V)	t_r/t_f	65 ns
NEP, Typical (970 nm, 5 V)	W//Hz	2.07 x 10^{-13}
Dark Current (5 V)	I_d	600 nA (Max.)
Capacitance (5 V)	C_J	375 pF (Typ.)
Package		Ceramic
Sensor Material		Si

Maximum Rating	
Max Bias (Reverse) Voltage	25 V
Reverse Current	10 mA
Operating Temperature	-10 to 60 °C
Storage Temperature	-20 to 70 °C

Specifications Subject to Change Without Notice

June 22, 2015
2739-S01, Rev G

7.2 Códigos usados en el protocolo

SEND.C

```
// SEND.C

// Autor: Rubén López

// Valor Lógico "0" = GPIO bajo x T/2 + GPIO alto x T/2
// __--
// Valor Lógico "1" = GPIO alto x T/2 + GPIO bajo x T/2
// --__

//Booleanos
#define TRUE 1
#define FALSE 0


#define T 3000              // Definido como microsegundos
#define PARITY  TRUE      // Valor por defecto para bit de paridad
#define STOP    TRUE       // Valor por defecto para bit de parada

// El protocolo define un tiempo de espera de un T después de enviar un
paquete

// Pines utilizados para los GPIO
#define RX_PIN 2          // Pin para recepción
#define TX_PIN 3          // Pin para envío

//Librerías estándar
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
//#include <sys/types>

// Threads
#include <pthread.h>

//Librería de manejo GPIO
#include <wiringPi.h>

// Macros
#define GET_BIT(var, bit) ((var & (1 << bit))>>bit) // Devuelve el valor
de bit de una variable

// Prototipo de funciones

// Convierte un bit a su representación PPM
void bit_to_ppm(int gpio, int periodo, unsigned char bit);

// Calcula la paridad de un char
char get_parity(char letra);

// Escribe un caracter de 8 bits en formato PPM
void wppm(int gpio, int periodo, char letra);

// Lee un caracter de 8 bits en formato PPM y lo devuelve
char rppm(int gpio, int periodo);
```

```
// Funcion thread para leer datos usando ppm
void* reading_function_thread(void *arg);

FILE *fr;

int main (){

    // Caracter para enviar
    int caracter = ' ';

    // Configurar wiringpi
    wiringPiSetup();
    pinMode(TX_PIN, OUTPUT);
    pinMode(RX_PIN, INPUT);

   // Abrir texto
   fr = fopen("texto1.txt","rt");

   //Alta prioridad de ejecución
    piHiPri(99);



    //Leer cada caracter del texto y enviarlo
   while (( caracter = fgetc(fr)) != EOF){
      wppm(TX_PIN, T, (char)caracter);
      delay(500);
   }

    // Enviar un caracter que termina el proceso de recepcion
    wppm(TX_PIN, T, 0xff);

    printf("Texto completo enviado\n");
    return 0;
}

// Convierte un bit a su representación PPM
void bit_to_ppm(int gpio, int periodo, unsigned char bit){

     if(!bit){      //Para el caso bit == "0"
           digitalWrite(gpio, LOW);
        delayMicroseconds(T/2);
        digitalWrite(gpio, HIGH);
        delayMicroseconds(T/2);
     }
     else{          //Para el caso bit == "1"
           digitalWrite(gpio, HIGH);
        delayMicroseconds(T/2);
        digitalWrite(gpio, LOW);
        delayMicroseconds(T/2);
     }
}

// Calcula la paridad de un char
char get_parity(char letra){
     unsigned char paridad = 0;
     int i;
     for(i=0; i<8; i++){
           paridad += GET_BIT(letra, i);
```

```
        }
        return (paridad&1);
}

// Escribe un caracter de 8 bits en formato PPM
void wppm(int gpio, int periodo, char letra){
        int i;
        unsigned char paridad;

        // Comienza utilizando un bit de encendido
        bit_to_ppm(gpio, periodo, 1);

        // Escribe la palabra en PPM
        for (i=0; i<8; i++){ // Para cada bit de la letra
                bit_to_ppm(gpio, periodo, GET_BIT(letra,i) );
        }

        // Envía el valor de paridad, si es definido
        if (PARITY){
                paridad = get_parity(letra);
                bit_to_ppm(gpio, periodo, paridad);
        }

        // Envía un bit de parada, si es definido
        if(STOP){
                bit_to_ppm(gpio, periodo, 0);
        }

        // Realice espera de tiempo T, definido por el protocolo
        digitalWrite(gpio, LOW);
       delayMicroseconds(T);
}

// Lee un caracter de 8 bits en formato PPM y lo devuelve
char rppm(int gpio, int periodo){
        unsigned char letra = 0, error = 0;
        unsigned char bit_aux = 0;
        int i;
        unsigned char paridad;

        // Se avanza en la recepción del mensaje al detectar el cambio de
flanco
        // generado por el primer bit (start bit)
        do { delayMicroseconds(1); } while(digitalRead(gpio)==LOW);
        // Se espera el cambio de flanco generado por el pulso PPM
        do { delayMicroseconds(1); } while(digitalRead(gpio)==HIGH);
        // Se espera el tiempo necesario para completar el periodo de la
señal
        delayMicroseconds(periodo/2);

        // Se espera un tiempo de defase para leer las señales en un punto
estable
        delayMicroseconds(periodo/7);

        // Se comienza a leer el caracter
        for (i=0; i<8; i++){ // Para cada bit de la letra
                bit_aux = digitalRead(gpio);
                //printf("Linea %d, bit leido: %x\n",i, bit_aux);
                letra |= (bit_aux<<i);
```

```
            delayMicroseconds(periodo/2);

            // Revisar el segundo pulso para ver si hay algun error en la
señal
            if(digitalRead(gpio) == bit_aux) {
                  //error |= (1<<i);
                  error++;
                  //printf("Bit leido incorrectamente\n");
            }
            delayMicroseconds(periodo/2);
      }
      // Recibe el valor de paridad y lo confirma
      if (PARITY){
            paridad = get_parity(letra);
            bit_aux = digitalRead(gpio);

            // Confirmar el valor de la paridad
            if (bit_aux != paridad) printf("Error de paridad en la letra
recibida!\n");
            delayMicroseconds(periodo/2);

            if(digitalRead(gpio) == bit_aux) printf("Error ppm en el bit de
paridad\n");
            delayMicroseconds(periodo/2);
      }

      // Confirma el bit de parada
      if(STOP){
            if (digitalRead(gpio) != 0) printf("Error ppm en el bit de
stop\n");
            delayMicroseconds(periodo/2);
            if (digitalRead(gpio) != 0) printf("Error ppm en el bit de
stop\n");
            delayMicroseconds(periodo/2);
      }
      //printf("Errores en la trama: %x\n", error);
      return letra;
}

// Funcion thread para leer datos usando ppm
void* reading_function_thread(void *arg){
    piHiPri(99);
     char ppm_recibido = 'a';
     do{
            ppm_recibido = rppm(RX_PIN, T);
            printf("Caracter recibido = %x \n", ppm_recibido);
     }while(1);
     //return NULL;
}
```

RECIEVE.C

```
// RECIEVE.C
// Autor: Ruben Lopez


// Valor Lógico "0" = GPIO bajo x T/2 + GPIO alto x T/2
```

```
// __--
// Valor Lógico "1" = GPIO alto x T/2 + GPIO bajo x T/2
// --__

//Booleanos
#define TRUE 1
#define FALSE 0


#define T 3000              // Definido como microsegundos
#define PARITY  TRUE       // Valor por defecto para bit de paridad
#define STOP    TRUE        // Valor por defecto para bit de parada

// El protocolo define un tiempo de espera de un T después de enviar un
paquete

// Pines utilizados para los GPIO
#define RX_PIN 2          // Pin para recepción
#define TX_PIN 3          // Pin para envío

//Tamaño del buffer de recepcion
#define BUFFER_SIZE 1000

//Librerías estándar
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
//#include <sys/types>

// Threads
#include <pthread.h>

//Librería de manejo GPIO
#include <wiringPi.h>

// Macros
#define GET_BIT(var, bit) ((var & (1 << bit))>>bit) // Devuelve el valor
de bit de una variable

// Prototipo de funciones

// Convierte un bit a su representación PPM
void bit_to_ppm(int gpio, int periodo, unsigned char bit);

// Calcula la paridad de un char
char get_parity(char letra);

// Escribe un caracter de 8 bits en formato PPM
void wppm(int gpio, int periodo, char letra);

// Lee un caracter de 8 bits en formato PPM y lo devuelve
char rppm(int gpio, int periodo);

// Funcion thread para leer datos usando ppm
void* reading_function_thread(void *arg);

FILE *file;
```

```
int main (){

    // Buffer de recepcion
    char buffer[BUFFER_SIZE];
    int i=0, j;

    // Configurar wiringpi
    wiringPiSetup();
    pinMode(TX_PIN, OUTPUT);
    pinMode(RX_PIN, INPUT);

    // Alta prioridad de ejecucion
    piHiPri(98);

    // Recibir caracteres hasta recibir '0xff' o hasta que se exceda el
    // Tamaño del buffer


    do{
       buffer[i] = rppm(RX_PIN, T);
       i++;
       //printf("%x\n", buffer[i-1]);
    }while( (buffer[i-1]!=0xff) );

    // Abrir el archivo en el que se almacenarán los datos
    file = fopen("recibido.txt","w");
    if(file == NULL){
       printf("Error el abrir archivo!\n");
       exit(1);
    }

    // Guardar los datos en el archivo
    for(j=0; j<i; j++){
       fprintf(file,"%c",buffer[j]);
    }

    fclose(file);
    return 0;
}

// Convierte un bit a su representación PPM
void bit_to_ppm(int gpio, int periodo, unsigned char bit){

     if(!bit){      //Para el caso bit == "0"
           digitalWrite(gpio, LOW);
        delayMicroseconds(T/2);
        digitalWrite(gpio, HIGH);
        delayMicroseconds(T/2);
     }
     else{         //Para el caso bit == "1"
           digitalWrite(gpio, HIGH);
        delayMicroseconds(T/2);
        digitalWrite(gpio, LOW);
        delayMicroseconds(T/2);
     }
}

// Calcula la paridad de un char
char get_parity(char letra){
```

```
      unsigned char paridad = 0;
      int i;
      for(i=0; i<8; i++){
            paridad += GET_BIT(letra, i);
      }
      return (paridad&1);
}

// Escribe un caracter de 8 bits en formato PPM
void wppm(int gpio, int periodo, char letra){
      int i;
      unsigned char paridad;

      // Comienza utilizando un bit de encendido
      bit_to_ppm(gpio, periodo, 1);

      // Escribe la palabra en PPM
      for (i=0; i<8; i++){ // Para cada bit de la letra
            bit_to_ppm(gpio, periodo, GET_BIT(letra,i) );
      }

      // Envía el valor de paridad, si es definido
      if (PARITY){
            paridad = get_parity(letra);
            bit_to_ppm(gpio, periodo, paridad);
      }

      // Envía un bit de parada, si es definido
      if(STOP){
            bit_to_ppm(gpio, periodo, 0);
      }

      // Realice espera de tiempo T, definido por el protocolo
      digitalWrite(gpio, LOW);
     delayMicroseconds(T);
}

// Lee un caracter de 8 bits en formato PPM y lo devuelve
char rppm(int gpio, int periodo){
      unsigned char letra = 0, error = 0;
      unsigned char bit_aux = 0;
      int i;
      unsigned char paridad;

      // Se avanza en la recepción del mensaje al detectar el cambio de
flanco
      // generado por el primer bit (start bit)
      do { delayMicroseconds(1); } while(digitalRead(gpio)==LOW);
      // Se espera el cambio de flanco generado por el pulso PPM
      do { delayMicroseconds(1); } while(digitalRead(gpio)==HIGH);
      // Se espera el tiempo necesario para completar el periodo de la
señal
      delayMicroseconds(periodo/2);

      // Se espera un tiempo de defase para leer las señales en un punto
estable
      delayMicroseconds(periodo/5);

      // Se comienza a leer el caracter
```

```
      for (i=0; i<8; i++){ // Para cada bit de la letra
            bit_aux = digitalRead(gpio);
            //printf("Linea %d, bit leido: %x\n",i, bit_aux);
            letra |= (bit_aux<<i);
                  //printf("%x %x %x \n", letra, bit_aux, i);
            delayMicroseconds(periodo/2);

            // Revisar el segundo pulso para ver si hay algun error en la
señal
            if(digitalRead(gpio) == bit_aux) {
                  error |= (1<<i);
                  error++;
                  //printf("Bit leido incorrectamente\n");
            }
            delayMicroseconds(periodo/2);
      }
      // Recibe el valor de paridad y lo confirma
      if (PARITY){
            paridad = get_parity(letra);
            bit_aux = digitalRead(gpio);

            // Confirmar el valor de la paridad
            if (bit_aux != paridad) printf("Error de paridad en la letra
recibida!\n");
            delayMicroseconds(periodo/2);

            if(digitalRead(gpio) == bit_aux) printf("Error ppm en el bit de
paridad\n");
            delayMicroseconds(periodo/2);
      }

      // Confirma el bit de parada
      if(STOP){
            if (digitalRead(gpio) != 0) printf("Error ppm en el bit de
stop\n");
            delayMicroseconds(periodo/2);
            if (digitalRead(gpio) != 0) printf("Error ppm en el bit de
stop\n");
            delayMicroseconds(periodo/2);
      }
      //printf("Errores en la trama: %x\n", error);
      return letra;
  }

  // Funcion thread para leer datos usando ppm
  void* reading_function_thread(void *arg){
      piHiPri(99);
      char ppm_recibido = 'a';
      do{
            ppm_recibido = rppm(RX_PIN, T);
            printf("Caracter recibido = %x \n", ppm_recibido);
         }while(TRUE);
  }
```

8. Referencias

Libros:

[1] Advanced free space optics (FSO). Arun K. Majumdar. Springer

[2] Fiber-Optic Communication systems. Govind P. Agrawal. Fourth Edition.

Sitios de internet:

[URL1] https://en.wikipedia.org/wiki/General-purpose_input/output

[URL2] https://opensource.com/resources/what-raspberry-pi

[URL3] https://en.wikipedia.org/wiki/Raspberry_Pi#Model_B

[URL4] https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/README.md

[URL5] http://raspberrypihobbyist.blogspot.com.es

[URL6] http://www.corelis.com/education/SPI_Tutorial.htm (SPI)

[URL7] http://i2c.info/

[URL8] http://www.es.freebsd.org/doc/en/articles/serial-uart/index.html

[URL9] http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/

[URL10] https://es.wikipedia.org/wiki/Arduino