industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Sistema de medida de movimientos oculares basado en reflectometría infrarroja.

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Raquel Buitrago Jiménez**
Director: Joaquín Roca González

Cartagena, 11 de Enero de 2017

Universidad Politécnica de Cartagena

industriales
etsii UPCT

Universidad Politécnica de Cartagena

DECLARACIÓN DE HONESTIDAD ACADÉMICA

El alumno: RAQUEL BUITRAGO con DNI: 48696104-K

como autor del TFG/TFM de título:

SISTEMA DE MEDIDA DE MOVIMIENTOS OCULARES BASADO EN REFLECTOMETRÍA INFRARROJA

correspondiente a la titulación de Grado/Máster :

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

de la Universidad Politécnica de Cartagena,

Declara que:

- El mencionado trabajo ha sido íntegramente elaborado por el abajo firmante.
- Que el trabajo arriba mencionado es inédito y en él no existe plagio de ninguna naturaleza.
- Que en ningún caso se han utilizado como propios resultados ni materiales obtenidos o generados por otros autores.
- Que los resultados utilizados realizados por otros autores han sido debidamente identificados en la memoria.

Y para que así conste, se firma la presente declaración en Cartagena a 11 de 01 de 2017

Fdo.- .....

Este documento, una vez firmado, lo deberá subir el estudiante al portal de servicios cuando solicite la defensa del TFE.

ÍNDICE

1. Introducción

1.1. Objetivo

Se redacta el presente documento de "Sistema de medida de movimientos oculares basado en reflectometría infrarroja" como Trabajo Final de Grado y conclusión de la titulación del Grado en Ingeniería Electrónica Industrial y Automática en la Universidad Politécnica de Cartagena.

El objetivo que se pretende alcanzar es el de definir y desarrollar el firmware necesario para implementar un dispositivo capaz de obtener señales procedentes del posicionamiento ocular mediante reflectometría infrarroja.

Su fin es conseguir, a través de éste, un sistema posterior de seguimiento ocular que nos permita ayudar a facilitar la vida de personas que padecen enfermedades como la esclerosis lateral amiotrófica o cualquier otra que provoque la parálisis total o parcial de su organismo.

Esta tecnología es aún novedosa, pues poder realizar un trabajo simplemente con mover nuestros ojos o poder realizar un control preciso de lo que dicta la mirada son tecnologias que las grandes empresas ya están trabajando.

Además de este objetivo principal se han fijado los siguientes objetivos parciales con el fin de lograr el desarrollo propuesto:

a) Puesta en marcha del entorno de desarrollo para la programación del microcontrolador.

b) Estudio de las capacidades de la plataforma elegida mediante la programación de programas elementales que permitan estudiar las siguientes funcionalidades:
   I. Control de entradas/salidas digitales (Encendido LEDS, lectura pulsadores, etc)
   II. Rutinas de temporización (Retardos / interrupciones)
   III. Conversión analógico/digital (Lectura potenciómetro)
   IV. Comunicaciones USB CDC (Modo texto)
   V. Conversión digital/analógico (generación de señales PWM)

c) Diseño del esquemático y diseño de placas de circuito impreso bajo Orcad Layout.

d) Especificación y selección de componentes.

e) Pero además de la creación del dispositivo comentado también se ha de desarrollar el estudio de otras partes asociadas a él que tienen especial importancia para su correcto desarrollo, como son:
   I. El estudio de costes y presupuesto.
   II. Planos de conexionado eléctrico.
   III. El diagrama de Gantt del proyecto. Incluido a continuación.
   IV. El estudio de mercado.
   V. Normativa y reglamento aplicados.

1.2. Alcance

Este proyecto abarca a todo el mercado interesado en el diseño de dispositivos que usen seguimiento ocular. Esto es debido a que se centra en la obtención de los datos procedentes del ojo, pudiendo así dejar vía libre para la creación de diversos sistemas posteriores.

El ámbito de aplicación de este proyecto se extiende, además de a personas que padecen discapacidad física a todo el mercado en general. Ésto es, que cualquier persona podría utilizar esta tecnología si se utilizara por ejemplo para el diagnóstico clínico o en el campo de la realidad virtual entre muchas otras aplicaciones.

2. Estado del arte

El estudio del estado del arte de todas las ramas que abarca dicho proyecto es indispensable para obtener un conocimiento más amplio de todos los campos a los que se hace referencia en él, aclarando así un poco mejor el posterior procedimiento a seguir para la implementación del mismo.

2.1. Fisiología del sistema de movimientos oculares.

Aunque en el proyecto no se estudian los movimientos oculares, se hace referencia de ellos para la obtención de las señales a recibir.

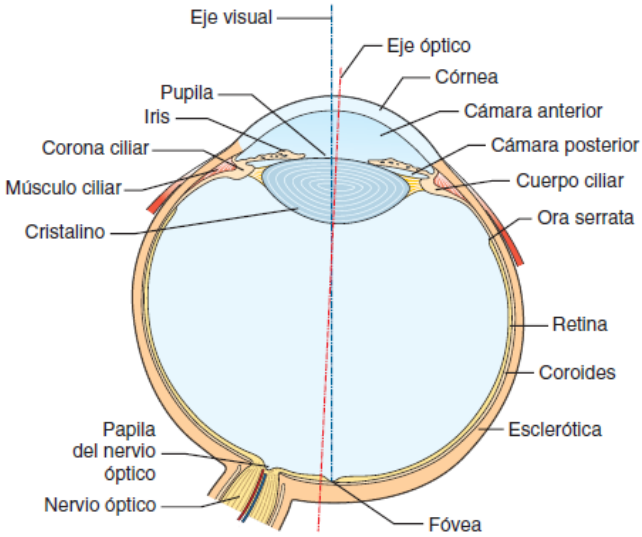


Figura 1.-Bulbo ocular derecho, en sección horizontal, con los ejes óptico y visual. ["Fisiología médica" - McGraw Hill Ed Interamericana].

2.1.1. Cinemática ocular.

Comenzando con la cinemática básica se tiene una serie de definiciones de los ejes y ángulos de los que se compone el globo ocular:

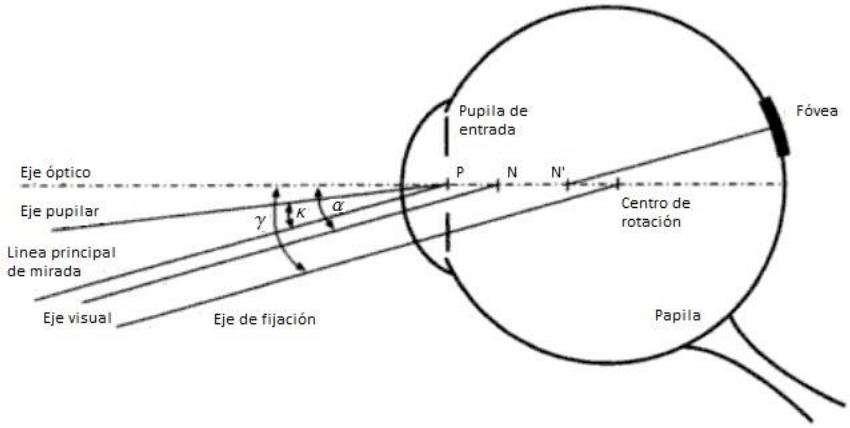


Figura 2.-Cinemática básica del globo ocular [Fisiología de los movimientos oculares - Documents].

Como se puede ver en la figura 2, el sistema cinemático ocular se basa en los siguientes componentes:

- Eje óptico: Recta que pasa por los puntos nodales y une todos los centros de curvatura de los dioptrios oculares.
- Eje visual: par de rectas paralelas que unen el punto de fijación con la fóvea a través de los puntos nodales N y N'.
- Centro de rotación: Punto alrededor del cual se producen los giros del ojo. Éste no es fijo ya que depende de los pequeños movimientos traslacionales.
- Eje de fijación: Es el eje que une el centro de rotación del ojo con el punto de fijación.
- Eje pupilar: Eje normal a la córnea que pasa por el centro de la pupila de entrada.
- Línea principal de mirada: Es la línea que une el punto objeto con el centro de la pupila de entrada.
- Ángulo α: ángulo formado entre el eje visual y el eje óptico. Su valor suele estar entre los 4 y 8º, tomándose un valor estándar de 5º.
- Ángulo γ: Es el ángulo formado entre el eje de fijación y el óptico. Es mayor que α.
- Ángulo κ: es el formado entre el eje pupilar y la línea principal de mirada. Generalmente κ es menor que α.

Además, a partir de la definición de los ejes de Fick, cualquier rotación alrededor de uno de sus ejes se denomina movimiento simple o secundario, mientras que cualquier combinación de movimientos o giro alrededor de un eje que no sea de Fick se denomina movimiento oblicuo o terciario.

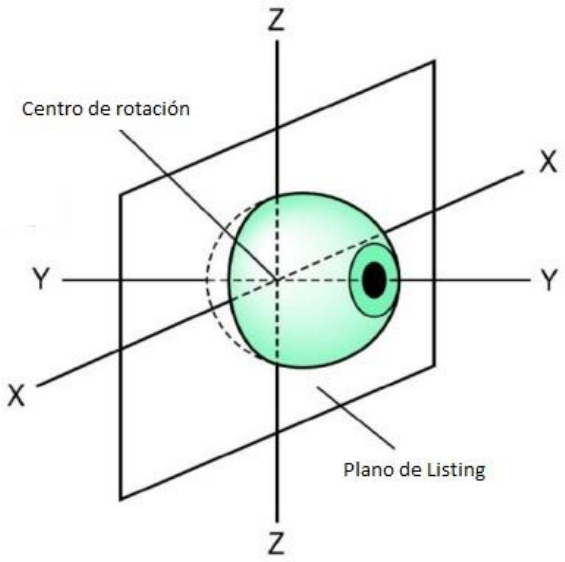


Figura 3.- Ejes de Fick. Ejes principales de rotación del ojo

Plano de Listing: Es un plano frontal que contiene el centro de rotación de los dos ojos (contiene los ejes de Fick YZ)

Las posiciones normales en las que se puede encontrar el ojo se les denomina versiones y se pueden clasificar en:

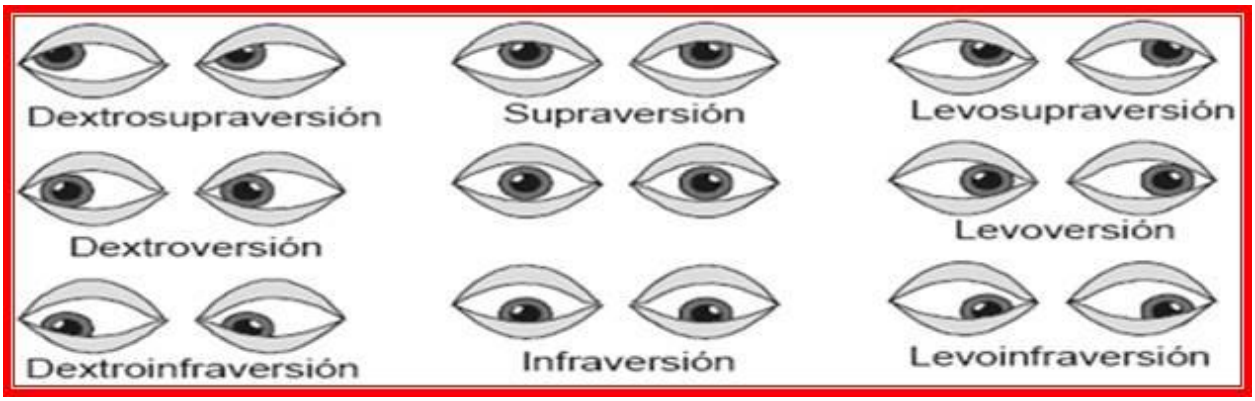


Figura 4.- Versiones. Principales posiciones del ojo.

2.1.2. Dinámica ocular.

Considerando el ojo como una esfera libre en las tres dimensiones del espacio XYZ, es necesario desarrollar al menos 6 fuerzas que generan tres pares de éstas. Y así, en el sistema anatómico del ojo, la naturaleza ha desarrollado exactamente seis músculos que permiten los tres pares de fuerzas comentados. De esta manera los músculos recto externo e interno son los que permiten los giros alrededor del eje Y; los músculos recto superior e inferior son los responsables de poder girar alrededor del eje Z; y los músculos oblicuos superior e inferior, con sus particulares sistemas de inserción ocular, se centran en los giros alrededor del eje X.

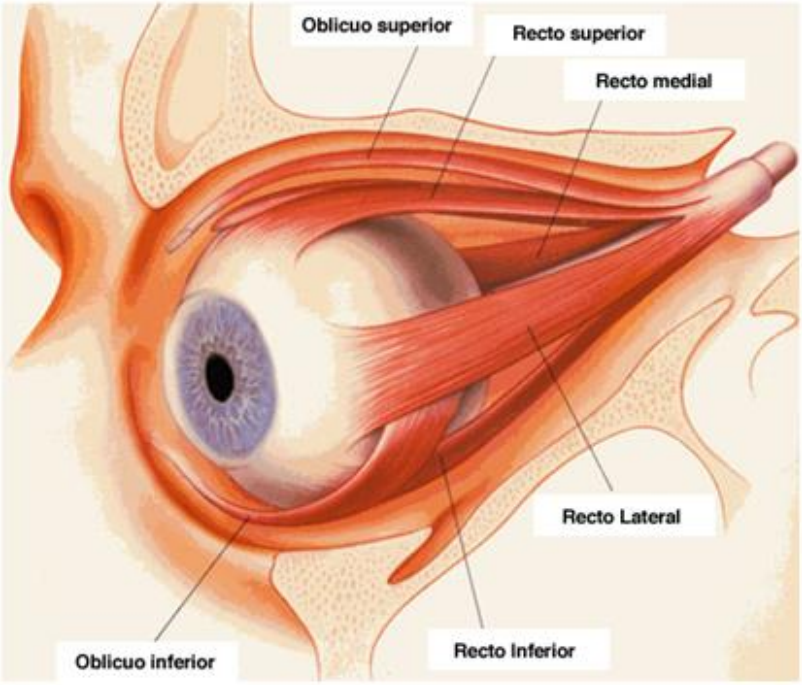


Figura 5.- Músculos del ojo. [Anatomía - AAPOS].

Existen muchas y variadas clasificaciones de los movimientos oculares atendiendo a diferentes aspectos, como por ejemplo, el que sean binoculares o monoculares, reflejos o voluntarios, o según sus rutas neuronales de control. Sin embargo, en estas clasificaciones siempre existen movimientos de difícil clasificación, por lo que es necesario hacer uso de clasificaciones más correctas que sean claras a la hora de determinar las características del movimiento. En esta línea, la clasificación funcional de Carpenter parece la más coherente, ya que se basa en atender a la funcionalidad del movimiento.

Según este criterio, se pueden determinar tres tipos de movimientos:

Movimientos para el mantenimiento de la mirada

Son aquellos que compensan el movimiento de la cabeza o del objeto para que la imagen permanezca sobre la fóvea.

Los movimientos corporales generados van siempre acompañados por un desplazamiento de las imágenes en la retina, por lo que se generaría una visión borrosa si no existieran mecanismos de estabilización de la mirada. Los reflejos optocinéticos y vestíbulo-oculares estabilizan la imagen en la retina durante el movimiento del campo visual.

- El reflejo oculovestibular es el mecanismo que mueve los ojos para compensar movimiento de cabeza, produciendo un movimiento ocular en la dirección opuesta al movimiento de la cabeza y conservando la imagen en el centro del campo visual.
- Mientras que el sistema optocinético compensa el movimiento continuo de todo campo visual, pues grandes desplazamientos del campo visual (en ausencia de movimientos de cabeza), requieren movimientos oculares lentos para compensar. Éste, complementa el VOR, particularmente en el rango de baja frecuencia, donde la ganancia del VOR es baja.

Movimientos para el desplazamiento de la mirada

Permiten pasar la atención de un objeto a otro (incremento del campo visual efectivo). Fundamentalmente se dan tres tipos: sacádicos, persecuciones o de seguimiento y vergencias.

- Movimiento ocular sacádico.

Los ojos de los humanos no miran una escena de forma estática por lo general, sino que se mueven buscando partes interesantes de una escena y construyendo un mapa mental referente a ella. La razón es que sólo la parte central de la retina (la fóvea) tiene una alta concentración de conos, siendo la parte de la retina encargada de la visión en alta resolución.

La visualización de toda la escena en alta resolución requeriría un diámetro del nervio óptico mayor que el del propio globo ocular, además de un cerebro varias veces superior al actual.

La dinámica del movimiento sacádico da cuenta de la complejidad del mecanismo que controla el movimiento del ojo. La velocidad angular máxima que se da durante un movimiento sacádico puede ser de hasta 1000 grados/s. Una sacada típica dura entre 20 y 200 milisegundos.

- Movimientos de seguimiento.

Son movimientos oculares de seguimiento rotacional rápido, mucho más rápidos que los movimientos optocinéticos. Participan en una amplia variedad de circunstancias oculomotoras, como mantener la mirada ante derivas involuntarias de ojo o la regulación de la fijación durante los movimientos vergenciales.

- Movimientos vergenciales.

Son movimientos binoculares en los que se varía el ángulo de cruce de los ejes visuales. Si el ángulo aumenta, el movimiento se denomina convergencia; si el ángulo disminuye, se denomina divergencia. Estos movimientos oculares, como los sacádicos y los de seguimiento, se utilizan para alinear las fóveas de los dos ojos sobre el objeto de interés, cualquiera que sea la distancia del mismo. Para un objeto en el infinito, los ejes visuales de los dos ojos deben estar paralelos (vergencia nula); para objetos más cercanos, los ojos deben converger.

Micromovimientos de fijación.

Son conocidos como micromovimientos por su pequeña amplitud. Su función es mantener la fijación e impedir el fenómeno, aunque es posible que estén involucrados en otros procesos de visión cuya finalidad se desconoce todavía. Parecen estar relacionados también con el mecanismo de la acomodación. Permiten que el sistema visual esté en las mejores condiciones dinámicas para tener una AV máxima. Destacan tres tipos de micromovimientos:

- Trémores.

Son temblores de los ojos, de muy pequeña amplitud, entre 17" y 1', y de altísima frecuencia, entre 30 y 75 ciclos/s.

- Fluctuaciones.

Son de mayor amplitud (5') y frecuencia más baja (unos 5 ciclos/s).

- Microsacádicos.

Son los micromovimietnso de mayor amplitud, entre 5 y 10' de arco, con una velocidad media de 10º/s.

Tras el estudio de los movimientos oculares, en lo referente a la aplicación en el proyecto se puede concluir que se necesita un sistema que trabaje con tiempos lo suficientemente pequeños como para captar todos estos movimientos, pues muchos de ellos son del orden de milisegundos e incluso menores.

2.2. Sistemas basados en seguimiento ocular.

Los sistemas de seguimiento ocular se pueden desarrollar con diferentes tecnologías:

- Mediante potenciales eléctricos.

Utilizando el potencial eléctrico medido con electrodos colocados alrededor de los ojos para detectar el movimiento.

- Mediante sensado invasivo.

Utilizando algo adjunto al ojo como una lente de contacto especial con un espejo incorporado o un sensor de campo magnético.

- Mediante sensado no invasivo.

Sin necesidad que haya contacto, a través de la luz y por lo general luz infrarroja. Ésta se refleja en los ojos y se capta mediante una cámara de video o algún otro sensor óptico. La información recogida se analiza para extraer la rotación de los ojos y los cambios en los reflejos.

2.2.1. Seguimiento ocular por reflectometría infrarroja.

Una noticia impactante sobre esta tecnología es la de un desarrollador británico que logró idear un sistema de reconocimiento ocular llamado OptiKey, extremadamente barato y con software Open Source que rivaliza con los caros sistemas comerciales disponibles para los afectados por la ELA. Julius Sweetland no se dedicaba a este ámbito, sin embargo la muerte de su tía, afectada por este tipo de enfermedad, hizo que este joven se lanzara a un proyecto personal que le ocupó los ratos libres durante cuatro años y que se ha convertido en OptiKey, un desarrollo capaz de funcionar al mismo nivel que otros desarrollos comerciales que pueden llegar a costar 4.000 dólares.

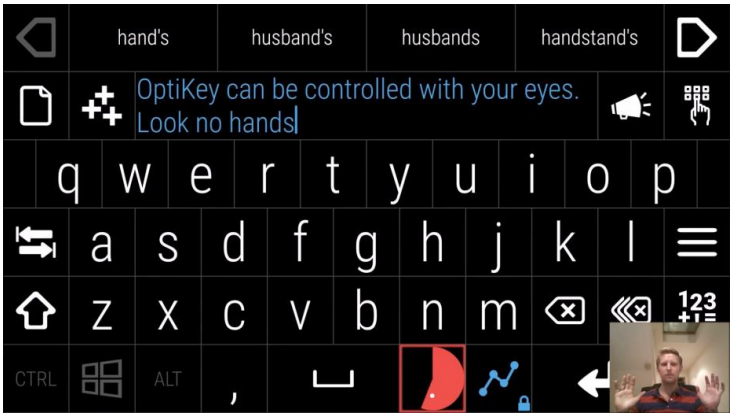


Figura 6.- Teclado digital Optikey.

El software es gratuito y se ha liberado con licencia Open Source GPLv3 en GitHub, donde cualquiera puede seguir su desarrollo, descargarlo, modificarlo y utilizarlo libremente. Solamente es necesario un PC y una cámara de seguimiento de ojos para funcionar, y esta cámara puede ser tanto la de la PlayStation como cualquier cámara especializada como por ejemplo la EyeWriter.

La EyeWriter capitaneada por Graffiti Reseach Lab, openFrameworks y The Ebeling Group consiste en un sistema de bajo coste y software libre escrito en C++ capaz de seguir e interpretar el movimiento de los ojos para que personas con parálisis total, como uno de los

miembros del grupo Graffiti Reseach Lab, el “graffitero” Tony Quan, puedan dibujar usando únicamente sus ojos.

Este dispositivo usa como hardware el Eye de Sony (PS3) modificado enganchado a unas gafas y principalmente se compone de dos partes: la que se comunica con el ojo y la que dibuja en la pantalla.

Figura 7.- Dispositivo con tecnología EyeWriter implementada.

A partir de la EyeWriter, Samsung desarrolló una adaptación que además permite escribir y navegar por la web llamada EyeCAN.

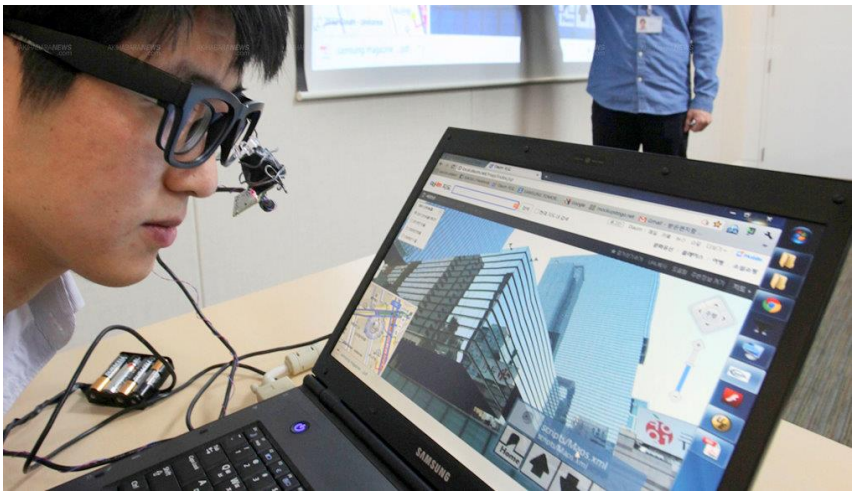


Figura 8.- Dispositivo EyeCAN de Samsung.

Más tarde apareció EyeCan +, donde la necesidad del uso de gafas desaparece, incluyendo la cámara en la pantalla del ordenador con el que se trabaje.

Figura 9.- Dispositivo EyeCAN+.

El seguimiento de los movimientos oculares también son utilizados en la investigación en los sistemas visuales, en psicología, en lingüística cognitiva y en diseño de productos.

Con los progresivos avances, hoy en día la tecnología de seguimiento ocular utiliza cámaras de alta velocidad (por ejemplo 60 imágenes por segundo) para rastrear el movimiento de los globos oculares, la dilatación de la pupila (pupilometría) y el parpadeo del sujeto, entre otros factores. Existen diferentes tecnologías de medición y algunas de ellas, como los monitores de Tobii, están diseñadas de una manera tan poco invasiva que utilizar esa tecnología no difiere de visualizar imágenes en un monitor convencional.

Figura 10.- Dispositivo Tobii [Laptopmag].

Los usuarios que adquieren los monitores Tobii, que pueden ser cualquier persona, evitan el movimiento innecesario de tener que desplazar el ratón o arrastrar un dedo en un panel táctil para mover el cursor. Esto supone una interacción mucho más natural e intuitiva con el dispositivo.

Una de las tantas aplicaciones posibles es la de medir la respuesta de los sujetos de estudio en neuromarketing o psicología. La información que recogen los sistemas de seguimiento visual pueden servir para conocer los recorridos visuales de los sujetos y crear mapas que señalen los puntos “calientes” de la imagen, es decir, los lugares en los que la vista se detiene durante más

tiempo. También pueden indicar las trayectorias que siguen y el orden en el que son examinados los elementos.

Por otro lado, para inferir la implicación emocional con lo que se está observando, otras técnicas de investigación incluso utilizan los datos relativos al parpadeo, velocidad de movimiento y dilatación de la pupila. Este es el caso de la tecnología Emotion Tool de iMotions, una compañía danesa especializada en el desarrollo de software de seguimiento ocular.

Figura 11.- iMotions, plataforma basada en la tecnología enfocada a las emociones.

La mirada puede revelar mucho acerca de las intenciones, pensamientos y acciones de una persona, ya que es un buen indicador de lo que realmente interesa o capta la atención del espectador.

3. Requisitos de diseño

Como dato de partida se solicita como proyecto una pequeña parte de lo que podría ser un sistema de seguimiento ocular basado en la reflectometría de luz infrarroja.

Los datos de los que se parte como requisitos mínimos son los siguientes:

Se intentará controlar una serie de 4 diodos LEDs y un fotodiodo para cada ojo, de manera que con el uso del microcontrolador PIC18F14K50, los diodos inciden luz infrarroja sobre el ojo siendo ésta reflejada y detectada por el fotodiodo.

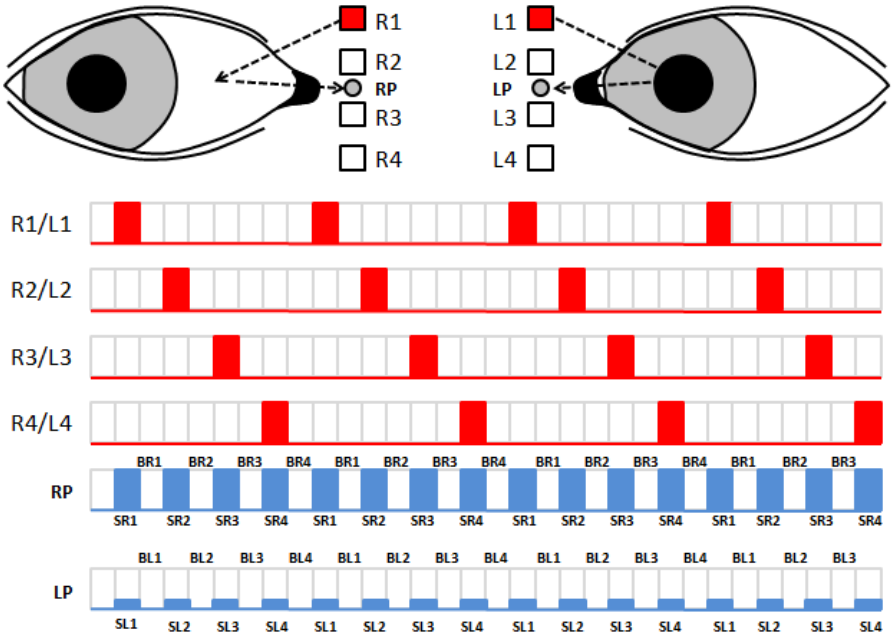


Figura 12.- Esquema de la secuencia de funcionamiento de los leds.

El seguimiento ocular ocurre cuando, en función de la cantidad de luz reflejada por el ojo ya se puede conocer la posición de éste dentro de la órbita ocular. Pues si la señal recibida por el fotodiodo corresponde con poca cantidad de luz, se deberá a que el LED está indicidiendo sobre una superficie que no refleja mucha luz, es decir, sobre una superficie oscura como son la pupila o el iris, y por el contrario, si refleja mucha cantidad de luz será debido a que el LED incide sobre la esclerótica.

La funcionalidad principal del programa es la de realizar barridos de luz por los 4 leds y capturar la señal recibida por los fotodiodos en el instante en el que acaba cada barrido. La rutina de encendido y apagado de los leds es la que se muestra en la Figura 12 trabajando esto en tiempos de microsegundos.

3.1. Seguridad fotobiológica.

El dispositivo expone al ojo a un constante bombardeo de luz infrarroja. Éste hecho deberá de ser estudiado en cuanto a los riesgos para la salud que pueda suponer.

Los LEDs que se usan en este proyecto son los SFH 4651. En su hoja de características se observa que su máxima potencia es de 40mW y se explica que:

"Estos dispositivos emiten luz infrarroja no visible altamente concentrada que puede ser peligrosa para el ojo humano dependiendo de su funcionamiento. Los productos que incorporan estos dispositivos tienen que seguir el precauciones de seguridad que figuran en la norma IEC 60825-1 e IEC 62471."

Por ello se procede en este apartado a estudiar los requisitos necesarios a cumplir por estas dos normas.

En la norma UNE EN 60825-1 se realiza el cálculo de la distancia nominal de riesgo ocular (DNRO). La distancia nominal de riesgo ocular para un láser determinado, conocida normalmente por sus siglas DNRO, es la distancia a la cual la exposición a la radiación igual a la exposición máxima permisible (EMP) apropiada para la córnea. La exposición se puede estimar a partir de la siguiente expresión:

$$E = \frac{4 \cdot P_o \cdot e^{-\mu r}}{\pi \cdot (a + r\emptyset)^2}$$

Donde:

- E es la irradiancia de la fuente en watios/m2.
- P_o es la potencia radiante del láser expresada en watios.
- El término exponencial hace referencia a las pérdidas debidas a la atenuación atmosférica (normalmente despreciable).
- a es el diámetro del haz.
- r es la distancia a la que se encuentra el trabajador.
- Ø es la divergencia del haz.

En nuestro caso:

$P = 40\text{mW};\ a = 0.2\text{mm};\ r = 1\text{cm}\ ;\ \emptyset = 20º.$

Entonces:

$$E = \frac{4 \cdot 0.04\text{W}}{\pi \cdot (0.2 \cdot 10^{-3}\text{m} + 0.01\text{m} \cdot \frac{20 \cdot 2 \cdot \pi}{360})^2} = 0.041323\text{W/m}^2$$

En el caso de que coincidan la exposición o irradiancia con la EMP, la distancia r será igual a la DNRO. El cálculo de dicha distancia se puede hacer mediante la fórmula siguiente:

$$DNRO = \frac{(4 \cdot \frac{P_o}{\pi \cdot E_{EMP}})^{1/2} - a^r}{\emptyset}$$

La DNRO se calcula de forma diferente en función del tipo de exposición: láser continuo, láser de impulsos.

$$\mathrm{DNRO} = \frac{(4 \cdot \frac{0.04}{\pi \cdot 0.0413})^{1/2} - (0.2 \cdot 10^{-3})^{0.01}}{\frac{20 \cdot 2 \cdot \pi}{360}} = 0.55$$

Debido a que se trata de una exposición pulsada, este valor tendrá un determinado porcentaje.

Si se requiere más información sobre dichas normas consulte el siguiente enlace.

4. Desarrollo software

4.1. Entorno de desarrollo

Figura 13.- Esquema de conexión del entorno de desarrollo software.

Para programar el PIC llevamos a cabo los siguiente pasos:

1º. Se crea el código de programa mediante el compilador CCS.

2º. Al compilar, se genera un archivo .HEX.

3º. Desde MPLAB se importa el archivo .HEX y se graba el PIC

4º. Se conecta directamente el PIC al PC para reconocerlo como puerto COM.

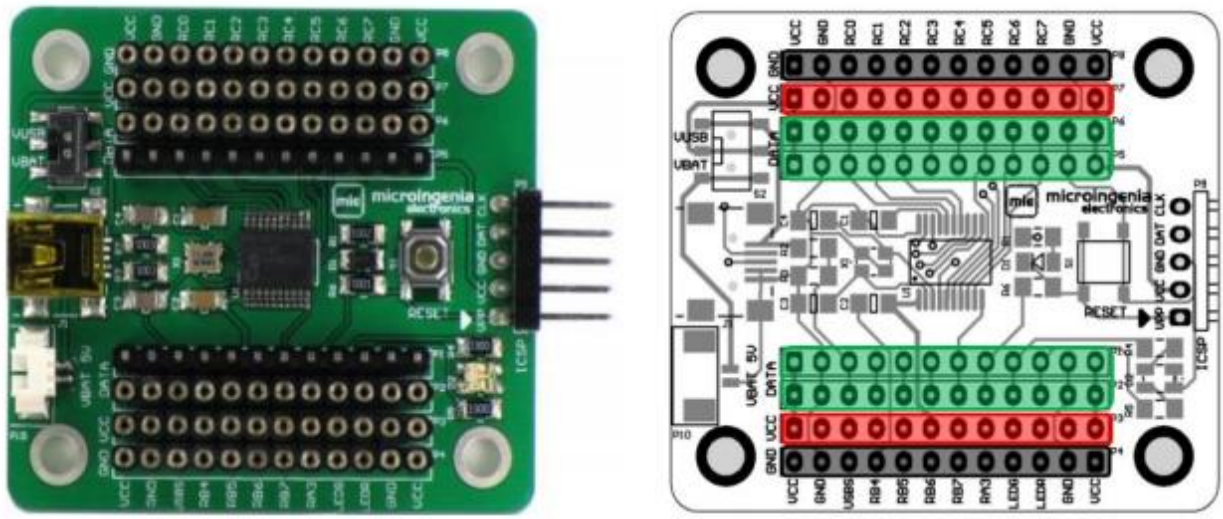


Figura 14.- Placa de desarrollo 18F1XK50 Trainerv1.0

4.2. Código de programa.

4.2.1. Pruebas iniciales. Blink de un led.

Mediante el programa PIC C Compiler se programa en lenguaje C el código necesario para que los LEDs se enciendan de manera deseada.

Como primer paso para la puesta a punto del entorno de desarrollo CCS, se plantea un problema clásico en la programación de microcontroladores, el encendido y apagado temporizado de un LED.

El propio programa incluye a modo de ejemplo el siguiente código para el blink de LEDs

```
#include <exampleblink.h>

void main()
{
   setup_timer_3(T3_DISABLED | T3_DIV_BY_1);
   setup_comparator(NC_NC_NC_NC);// This device COMP currently not supported
by the PICWizard

   //Example blinking LED program
   while(true)
   {
      output_low(LED);
      delay_ms(DELAY);
      output_high(LED);
      delay_ms(DELAY);
   }
}
```

Listado 01.- Ejemplo de blink de un LED.

Con el fin de poder compilar programas para la plataforma elegida (PIC18F14K50) se hace preciso incorporar un fichero de configuración especialmente programado a los fines de este proyecto. Para ello se incluyeron los ficheros de cabecera propios del microcontrolador y las conexiones de E/S.

```
#include <18F14K50.h>
#fuses HS, NOWDT, NOPROTECT, NOLVP, NODEBUG, NOBROWNOUT, USBDIV1, PLLEN,
CPUDIV1, PUT, MCLR
#use delay(clock=48000000)
```

Listado 02.- Cabecera de programa.

La primera línea del código hace referencia al microcontrolador que va a ser utilizado en este proyecto (18F14K50), en este fichero se detallan todas las partes del PIC y su configuración.

La segunda línea se encarga de las directivas de preprocesado del microcontrolador, es decir, activa los fuses con cada uno de esos comandos. De manera particular cada uno de ellos activa:

- HS: Habilita el uso de reloj externo de alta velocidad.
- NOWDT: Deshabilita el uso del perro guardián.
- NOPROTECT: Deshabilita la protección del código.
- NOLVP: Deshabilita Low Voltage ICSP Programming.
- NODEBUG: Deshabilita la depuración en línea.
- NOBROWNOUT: Deshabilita reset del PIC por caída de voltaje.
- USBDIV1: Divisor de frecuencia USB igual a 1.

- PLLEN: Habilita el uso de PLL.
- CPUDIV1: Divisor de frecuencia CPU igual a 1.
- PUT: Power Up Timer.
- MCLR: Habilita pin de reset.

La tercera línea configura la frecuencia a la que trabaja el microcontrolador, en este caso trabaja a 48MHz velocidad establecida para las comunicaciones USB.

Se usa el diodo led verde incluido en la placa de desarrollo del microcontrolador 18F14K50 conectando el puerto RB4 al LEDG.

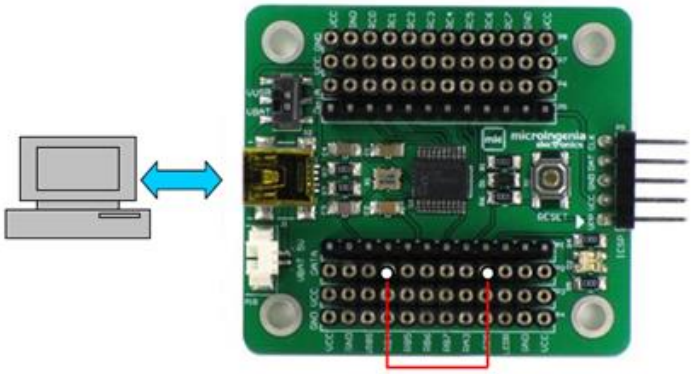


Figura 15.- Conexiones de la placa 4.2.1

El programa a probar será el siguiente:

```
#include <18F14K50.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT
#use delay(clock=48000000)

void main(void)
{
 while(TRUE)
 {
 output_high(PIN_B4);
 delay_ms(500);
 output_low(PIN_B4);
 delay_ms(500);
 }
}
```

Listado 03.- Encendido y apagado de un led con el uso de la función delay.

Cuando se compila y se ensambla un programa, junto con el archivo ".c" creado se generan otros archivos entre los que se incluye el archivo en hexadecimal (.HEX).

Éste es un formato de archivo para la programación de microcontroladores. Su contenido se corresponde exactamente con lo que ha de ser grabado en la memoria de programa (o EEPROM) del PIC. Se trata de una serie de instrucciones que el grabador de PIC's PickIt3 sabe interpretar y grabar.

Éste archivo .HEX se ha de importar en MPLAB.

Un dato importante en la puesta en marcha del entorno MPLAB es la alimentación del PIC, es necesario que el propio PickIt3 lo alimente a un voltaje de 5V para que funcione, de lo contrario

aparecerá el error que podemos observar en la Figura 16 " PK3Err0045". Esto se configura desde el mismo MPLAB.

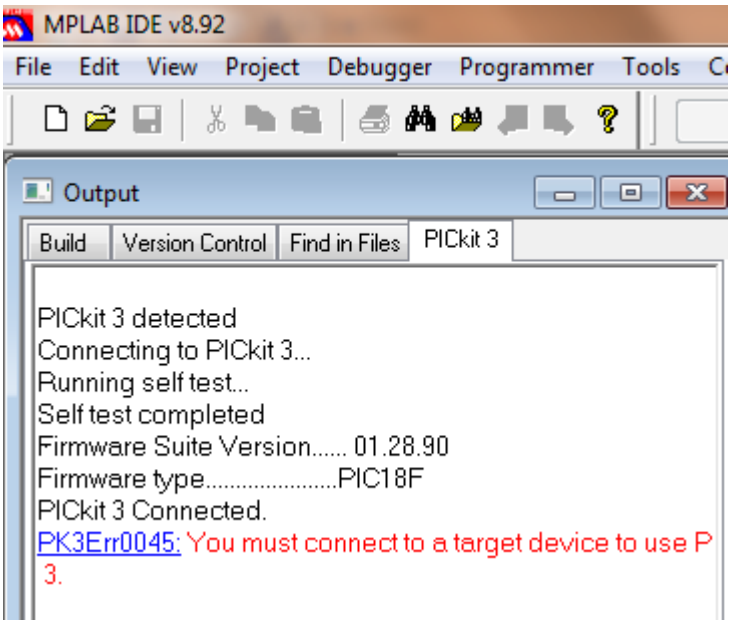


Figura 16.- Error en la alimentación de la placa en el entorno MPLAB.

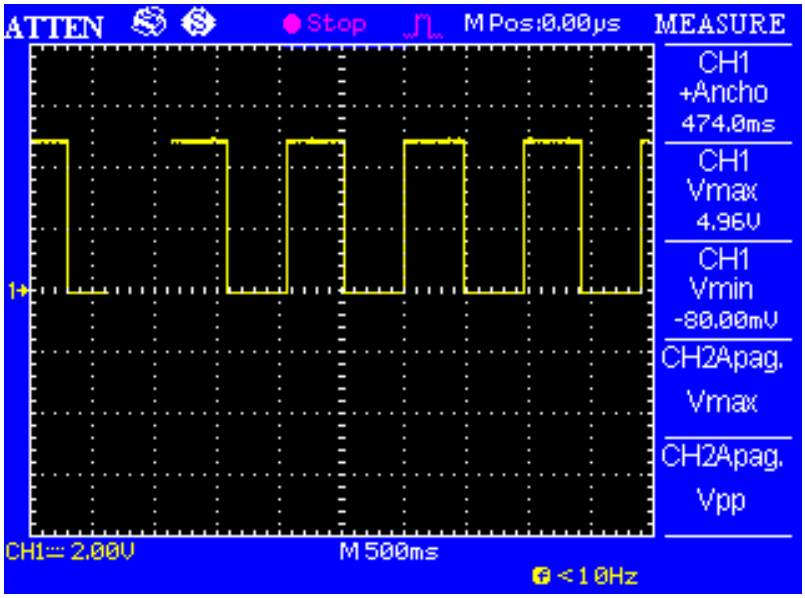


Figura 17.- Captura de la imagen que muestra el osciloscopio.

En el osciloscopio se puede apreciar que de los 500ms a los que debía ir el programa él capta unos 474,0ms. Esto es debido a que la función delay no es muy precisa y por ello se realiza otro programa en el que se sustituye esta temporización por una interrupción realizada por el timer0.

4.2.2. Interrupciones temporizadas (timer0).

El programa anterior con la función delay es demasiado sencillo, por lo que se modificará por otro en el que se le incluyen interrupciones temporizadas configurando el timer0 interno del microcontrolador.

Las conexiones de la placa serán iguales a las del apartado anterior 4.2.1.

```
void main(void)
{
 disable_interrupts(GLOBAL);
 disable_interrupts(INT_TIMER0);
 setup_timer_0(RTCC_DIV_1);
 set_timer0(41726); // Valor de precarga para que el timer0 cuente 0.5 seg
 enable_interrupts(GLOBAL);
 enable_interrupts(INT_TIMER0);
 output_low(PIN_B4);

 while(TRUE){}
}
#int_timer0
void isr_timer0(void)
{
output_toggle(PIN_B4);
 set_timer0(41726);
}
```

Listado 04.- Encendido y apagado de un led con interrupción del timer0.

En este segundo programa algo más complejo, cuando el contador llega a contar 41726 Bits (alrededor de 0,5 segundos)

Se hace uso de las siguientes funciones: se produce la interrupción por desbordamiento del contador timer0 pasando a ejecutarse su rutina de interrupción. En esta interrupción se lleva a cabo un 'toggle' (pasar de nivel bajo a alto y viceversa) del pin al que está conectado el LED.

- disable_interrupts() : Deshabilita las interrupciones que van dentro del parentesis.
- enable_interrupts() : Habilita las interrupciones.
- setup_timer_0(RTCC_DIV_256) : Habilita la interrupción del timer0 y configura el preescaler a 256.
- set_timer0() : Escribe un valor en el registro timer0.
- #int_timer0 : Habilita la función de atención a la interrupción void isr_timer0(void).

El cálculo de la precarga del timer0 se lleva a cabo de la siguiente manera. Primero calculamos el tiempo por cuenta "TC" a partir de la frecuencia del oscilador.

$$\mathrm{TC} = \frac{1}{\mathrm{F_{osc}}/_4} = \frac{1}{48\mathrm{MHz}/_4} = 8{,}33 \cdot 10^{-8}\mathrm{s}$$

Y a partir de éste, se obtienen las cuentas "C", necesarias en función del tiempo que se desea que pase:

$$\mathrm{C} = \frac{\mathrm{Tiempo}}{\mathrm{TC}} = \frac{0{,}5\mathrm{s}}{8{,}33 \cdot 10^{-8}\mathrm{s}} = 6002400{,}96$$

Sin embargo dado que el timer0 es de 16 bits sólo puede contar hasta $2^{16} = 65536$.

Por ello será necesario el uso de un preescaler. La función del preescaler es la de dividir la frecuencia para que el resultado de la cuenta sea menor y se ajuste a un dato válido.

Se utiliza un preescaler de 1:256, quedando la frecuencia final como:

$$F = \frac{F_{osc}}{4} \cdot \frac{1}{256} = 46875\text{Hz}$$

Y volviendo a realizar los cálculos anteriores pero con esta nueva frecuencia:

$$TC = \frac{1}{46875} = 2{,}133 \cdot 10^{-5}\text{s}$$

$$C = \frac{0{,}5\text{s}}{2{,}133 \cdot 10^{-5}\text{s}} = 23437$$

Entonces si con esta cuenta se obtienen 0,5 segundos, la precarga del timer0 debe ser: $((2^{16} - 23437) = 41726$ para que cuente solo 0,5 segundos y desborde.

Se modifica el programa para que trabaje a 125 microsegundos (con 64035 de precarga del timer0) y se observa a continuación en la Figura 18 que el osciloscopio marca una medida de 129us.+

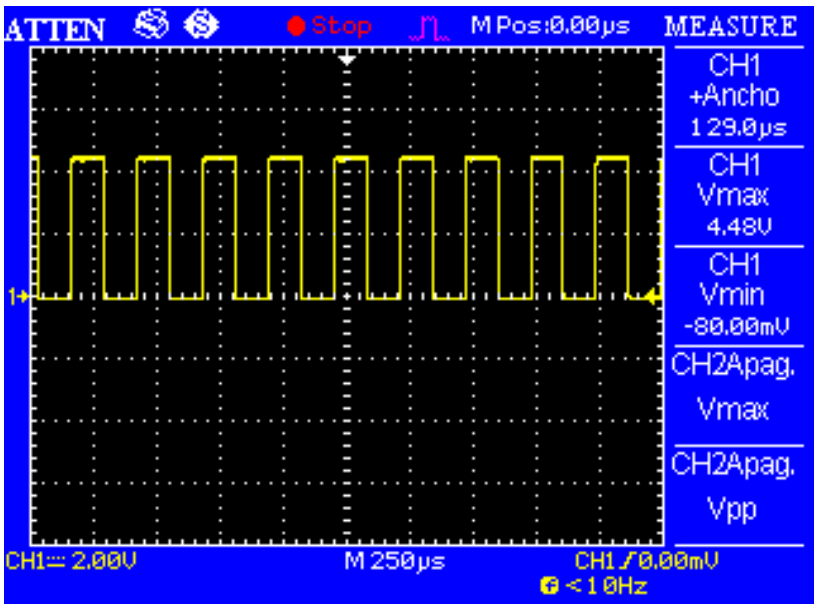


Figura 18.- Señal obtenida en el osciloscopio por el código que usa el timer0 con precarga para 125 microsegundos.

Ésta diferencia de tiempo es debida al tiempo que tarda la función de interrupción en llevarse a cabo, error que solucionaremos en el programa final más adelante.

4.2.3. Comunicación USB

Tomando como base el primer programa implementado que funciona con la función delay, se empieza por un ejemplo fácil de como enviar vía USB diferentes cadenas de caracteres dependiendo de la acción que se lleva a cabo en cada momento.

Para la comunicación USB se utiliza la librería usb_cdc.h. Ésta librería incluye todas las funciones que se necesitan.

```
void main()
{
 usb_cdc_init();
 usb_init();
 usb_wait_for_enumeration();

 output_low(PIN_B4);

 while(TRUE)
 {
 output_high(PIN_B4);
 printf(usb_cdc_putc, "Encendido LED verde\n\r");
 delay_ms(500);
 output_low(PIN_B4);
 printf(usb_cdc_putc, "Apagado LED verde\n\r");
 delay_ms(500);
 }
}
```

Listado 05.- Comunicación USB

Se incluyen las funciones:

- usb_cdc_init(): Configura la comunicación USB.
- usb_init(): Inicializa el hardware del USB.
- usb_wait_for_enumeration(): Espera hasta que el PicUSB sea configurado por el host.
- usb_cdc_putc: Se encarga de escribir por la pantalla del programa virtual.

Para la comunicación USB, una vez programado el microcontrolador para que el PIC pueda enviar datos al ordenador es necesario retirar el PickIt y alimentar la placa directamente desde el ordenador.

La primera vez que se conecta al ordenador, éste alimentará al PIC pero no lo reconocerá como un dispositivo nuevo conectado. Para que el ordenador reconozca al PIC como un puerto COM es necesario un driver que permita reconocer la placa desde el USB como un puerto COM virtual.

Para ello se cuenta con el driver "MiEUSBCDCDriver". Se abre el driver como bloc de notas y se tienen que modificar dos datos, el ID del vendedor (VID) y del producto (PID), éstos son dos números de 16 bits representados en hexadecimal que definen el PIC exacto a utilizar.

Estos datos del dispositivo que se está usando se pueden observar en el "Administrador de dispositivos" del equipo cuando éste se encuentra conectado, se selecciona el dispositivo que hace referencia a la placa conectada y en detalles se selecciona "Id. de hardware" donde se muestra dichos valores:

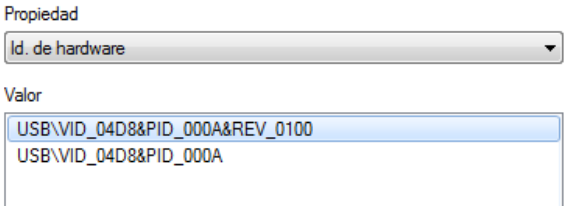


Figura 19.- VID y PID del dispositivo conectado.

Éstos, en caso de que no aparezcan en la información del driver se deben introducir manualmente.

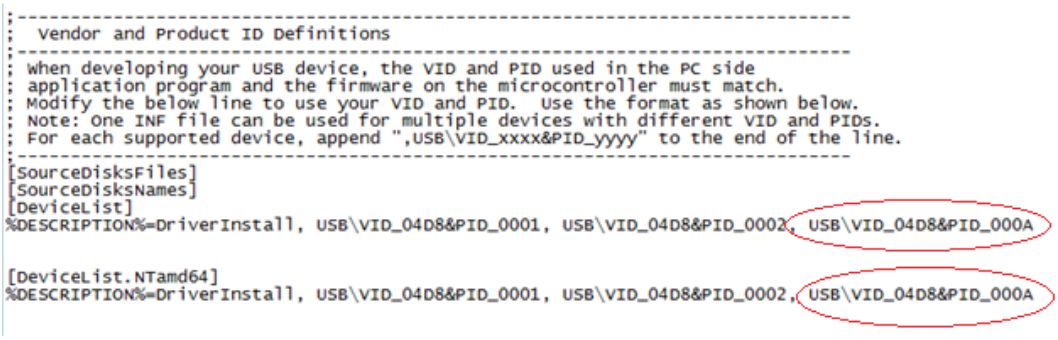


Figura 20.- VID y PID introducidos en el driver.

Una vez hecho esto se tiene que actualizar el controlador del dispositivo que se reconoce como el PIC e introducir aquí el driver. Así reconocerá el microcontrolador como un puerto COM.

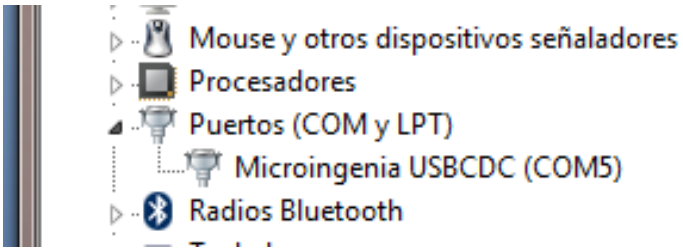


Figura 21.- En el administrador de dispositivos del equipo se observa que se crea un Puerto COM5 al introducir la placa vía USB al PC.

Para visualizar lo enviado por el puerto serie contamos con el programa Realterm.

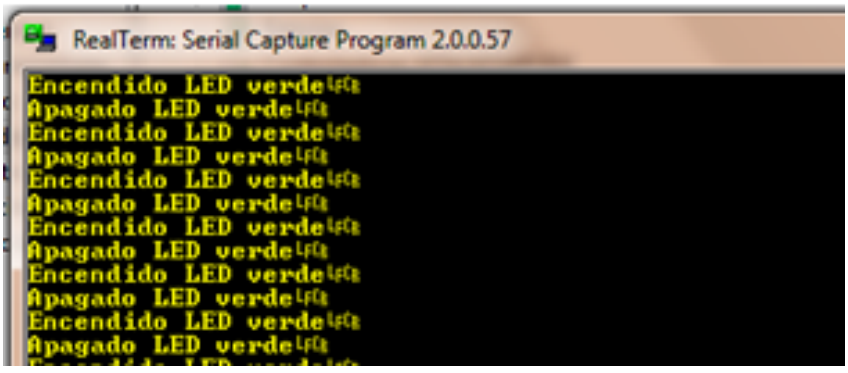


Figura 22.- Comunicación vía USB realizada con éxito.

4.2.4. Convertidor ADC.

Los datos procedentes de los fotodiodos son valores analógicos, por lo que será necesario conocer y trabajar con el convertidor digital-analógico ADC interno del microcontrolador.

Para visualizar el correcto funcionamiento del código de programa se usa un potenciómetro.

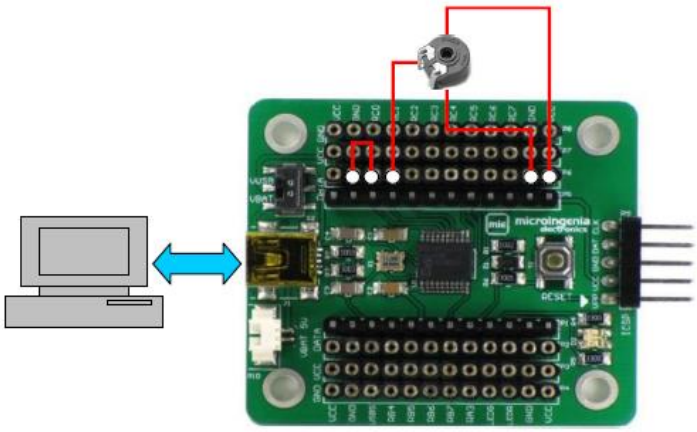


Figura 23.- Conexiones de la placa con el PC y el potenciómetro.

Se utiliza, para empezar a aprender el funcionamiento del convertidor ADC, el siguiente código de programa:

```
void main()
{
signed int16 Res=0;

usb_cdc_init();
usb_init();
usb_wait_for_enumeration();

setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(sAN5|VSS_VDD);
set_adc_channel(5);

 while(true)
 {
 Res = read_adc();
printf (usb_cdc_putc,"%li\r\n", Res);
 delay_ms(110);
 }
}
```

Listado 06.- Lectura de valores analógicos.

Con este código imprimirá por la pantalla del Realterm cada 110 ms el valor procedente de la entrada analógica 5 "Res". Las funciones básicas a utilizar para una conversión A/D son:

- setup_adc(ADC_CLOCK_INTERNAL): Ésta configura el modo de la conversión A/D, en este caso reloj igual a el reloj interno de la conversión.

- setup_adc_ports(sAN5|VSS_VDD): Configura que pines son digitales o analógicos, el valor de referencia de la conversión es VSS o VDD.

- set_adc_channel(5): Se utiliza el canal 5 para la conversión.

- read_adc(): Lee el valor analógico que viene a ser un número entero dependiendo de la directiva "#DEVICE ADC="empleada. En nuestro caso es de 10 bits, es decir, de 0 a 1024.

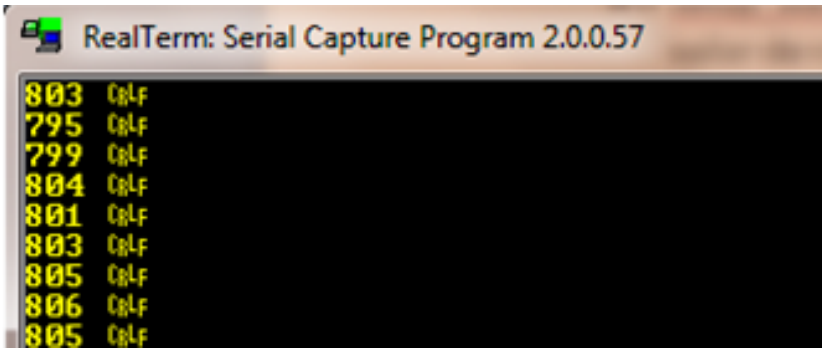


Figura 24.- Captura del programa Realterm en el que se obtienen los valores analógicos del potenciómetro.

4.2.5. Secuencia de los leds.

Básicamente se trata de un programa con 4 leds temporizados de manera que se enciendan y apaguen uno tras otro recorriendo un periodo hasta que el primero vuelva a encenderse y a comenzar la rutina.

Entre el apagado de uno de ellos y el encendido del siguiente también se deja este mismo tiempo, obsérvese la Figura 25 para una mejor comprensión:

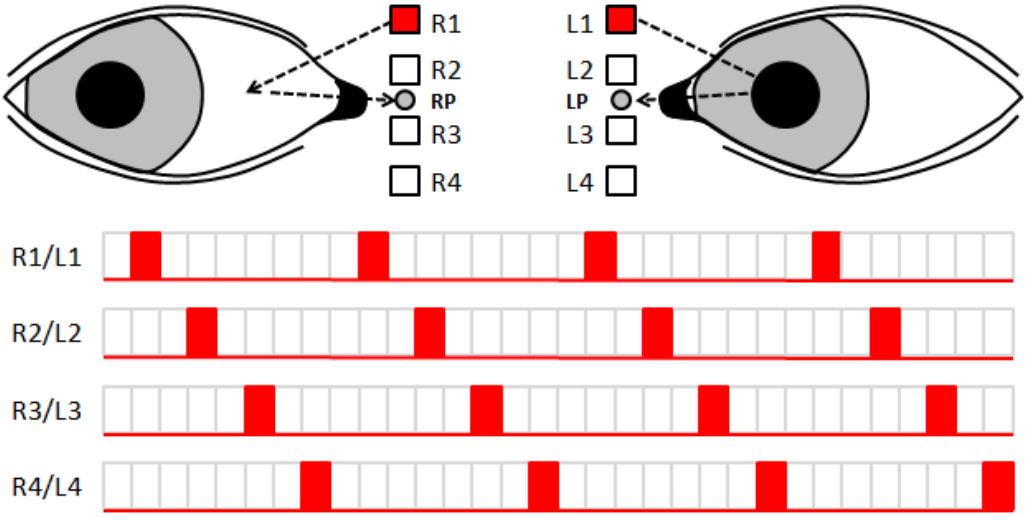


Figura 25.- Secuencia a seguir por los 4 LEDs.

Para comenzar se establece un valor de 1ms de periodo, es decir dividido entre 8 son 125 microsegundos para cada led.

Para una mejor comprensión del código observar el siguiente flujograma de la Figura 26.

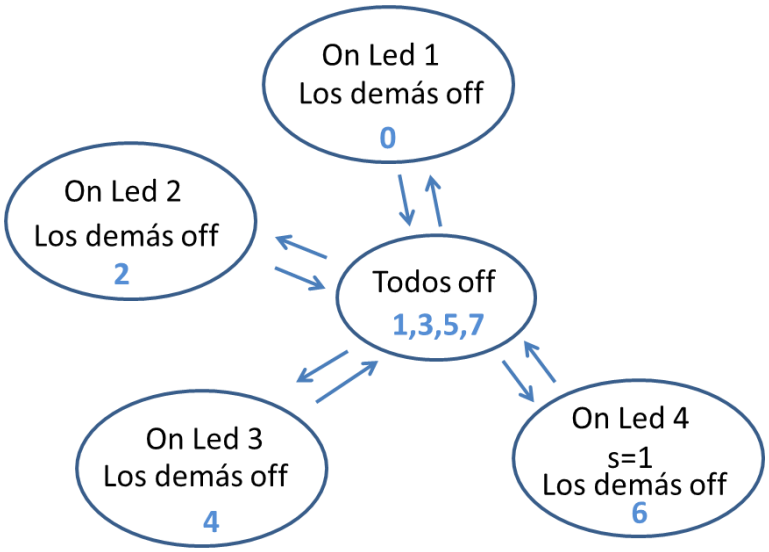


Figura 26.- Flujograma de la secuencia de encendido de los leds.

Se introduce un contador para ir pasando de un estado a otro del flujograma, de forma que al llegar a ser éste mayor que 7 se vuelve a iniciar empezando de nuevo el ciclo.

Esta secuencia se introduce dentro de la función de interrupción del timer0, así cada vez que se cambie de un estado a otro pasará el tiempo deseado.

```
#int_timer0
void isr_timer0()
{
   if(cont==0){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   }
   if(cont==2){
   output_low(Led1);
   output_high(Led2);
   output_low(Led3);
   output_low(Led4);
   }
   if(cont==4){
   output_low(Led1);
   output_low(Led2);
   output_high(Led3);
   output_low(Led4);
   }
   if(cont==6){
   output_low(Led1);
   output_low(Led2);
   output_low(Led3);
   output_high(Led4);
   }
   if((cont==1) || (cont==3) || (cont==5) || (cont==7)){
   output_low(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   }
cont++;

if(cont>7){
cont=0;}

set_timer0(64035);
}
```

Listado 07.- Secuencia de encendido y apagado de los LEDs.

El contador se incrementa al final de la secuencia y se carga el timer0 para que vuelva a entrar en la rutina una y otra vez.

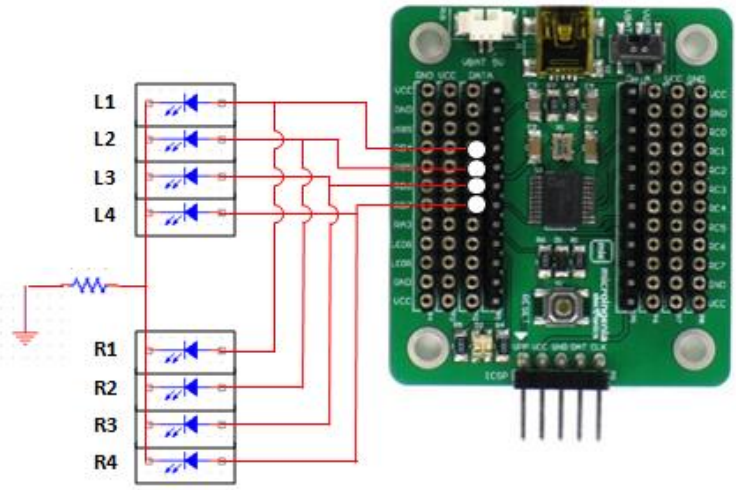


Figura 27.- Esquema de conexiones de los diodos leds.

Se utilizan los mismos pines tanto para los diodos del ojo derecho como para los del izquierdo como se observa en la Figura 27, pues se encenderán y apagarán a la misma vez los pines con el mismo número.

Las conexiones reales para las comprobaciones de funcionamiento en el laboratorio son las que se pueden observar en la Figura 28, como solo se cuenta con dos salidas del osciloscopio solo se puede observar el funcionamiento de dos leds.

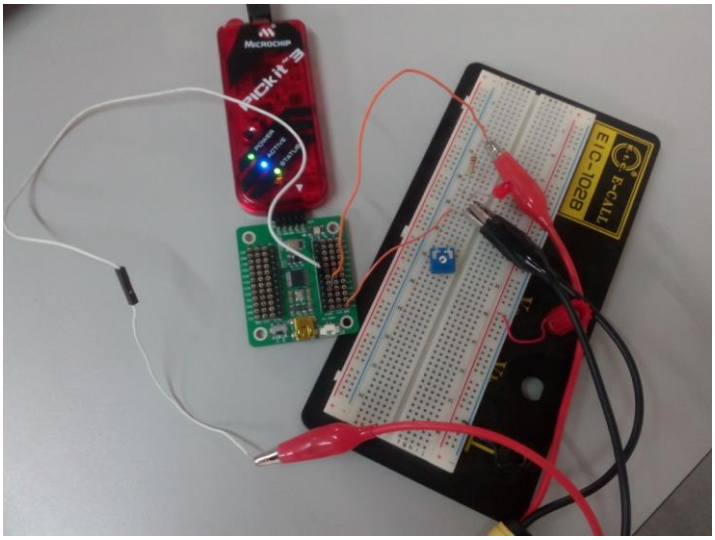


Figura 28.- Conexión de 2 diodos LEDs.

Se quiere observar el tiempo entre parpadeos o el tiempo que está midiendo entre cada señal, (Se introduce de 125us).

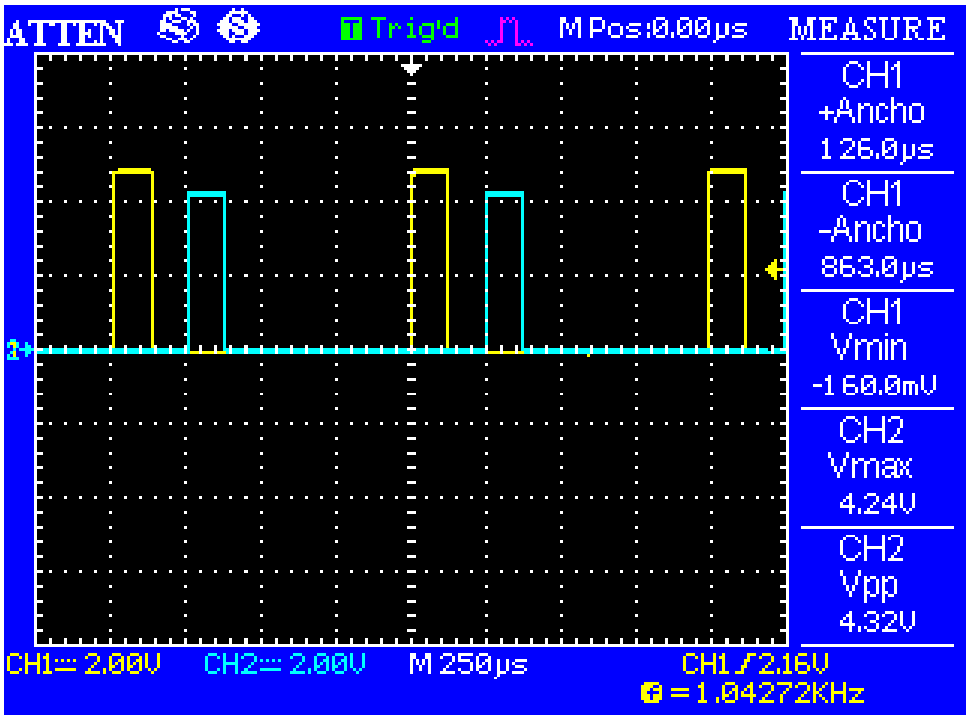


Figura 30.- Osciloscopio (pines B4y B5).

4.2.6. Introducir valor por teclado.

Con el fin de hacer más moldeable el programa se realiza una modificación al anterior, ésta consiste en tener la posibilidad de mandar por teclado el tiempo entre parpadeo de los leds.

Primero se implementó el siguiente programa con la función delay para simplificar las cosas. Sin embargo para no ser redundante se pasa directamente al código implementado con interrupciones del timer0 pues es igual pero más completo. El código con la función delay se adjunta en el anexo.

Para una mejor comprensión del siguiente código observar el flujograma siguiente que rige sus funciones principales de una forma vistosa.

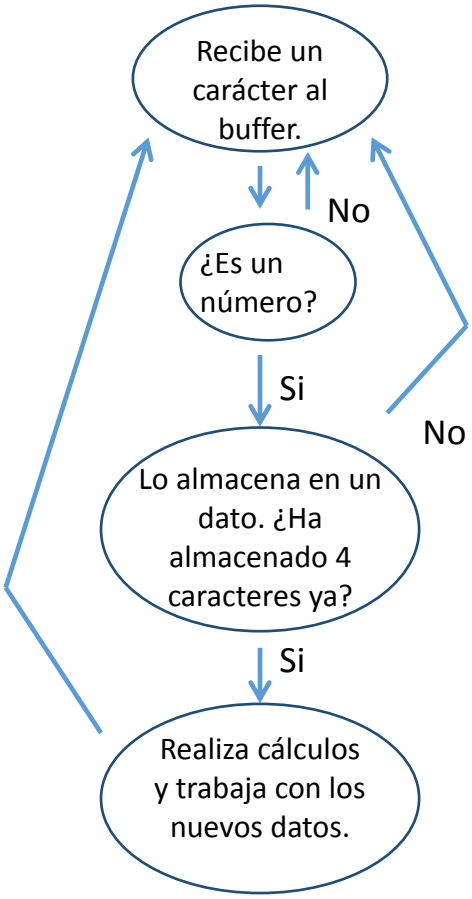


Figura 31.- Flujograma principal del código de programa para introducir datos por teclado.

El hecho de que necesite 4 caracteres es porque se pueden introducir datos de hasta 9999 microsegundos.

El nuevo fragmento del programa es el siguiente:

```
void usb_debug_task(void) //Esta función depura las tareas del USB.
{
   static int8 last_cdc;
   int8 new_cdc;
   new_cdc = usb_cdc_connected ();
   last_cdc = new_cdc;
}

void main(void)
{
```

```
[...]
 while(true)
 {
  usb_task();
  usb_debug_task();
  if (usb_cdc_kbhit())
  {
   for (i=0;i<4;i++)
   {
    do
    {
     in_data[i] =usb_cdc_getc();
    }while (in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51
    &&in_data[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55&&
    in_data[i]!=56&&in_data[i]!=57);

    valores[i] =(in_data[i] -48); //pasamos de ASCII a decimal
    printf (usb_cdc_putc, "%c", in_data[i]);
   }

   tiempo =(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3]);
   cuenta=(tiempo*0.001)/0.00002133333333; // Para us el tiempo a la -6 y
   precarga=(65536-cuenta); // entre 8.3333E-8

   disable_interrupts(GLOBAL);
   disable_interrupts(INT_TIMER0);
   setup_timer_0(RTCC_DIV_256);// Para us se usa 1 en el preescaler
   set_timer0(precarga);
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_TIMER0);

   while (true){}
  }
 }
}
#int_timer0
void isr_timer0()[...] // Secuencia de leds anterior.
```

Listado 08.- Introducir valor por teclado.

- usb_cdc_kbhit( ) : Devuelve TRUE si hay uno o más carácter recibidos y esperando en el búfer de recepción.
- usb_cdc_getc( ) :Obtiene un carácter del búfer de recepción.

Entonces si la función kbhit se hace verdadera empezará a imprimir por pantalla los datos introducidos por teclado.

La función do-while sirve para que sólo se puedan introducir por teclado números, pues en el código ASCII los números, desde el 0 hasta el 9 son del 48 al 57.

Se almacenan cada uno de los números introducidos en un array y después se calcula el tiempo como se observa a continuación.

```
tiempo =(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3]);
```

A continuación de esto se realiza el cálculo de la precarga del timer0 tal y como se explica en su apartado correspondiente 4.2.2.

Este cálculo de la precarga del timer0 sirve para valores de tiempo comprendidos entre un segundo y unos pocos milisegundos introduciendo además el tiempo por teclado en

milisegundos. Pues este código se usa para experimentar en el laboratorio y observar resultados correctos de forma visual.

Para poder introducir los datos en microsegundos habrá que realizar una pequeña modificación del código anterior en cuanto a esta última cuenta.

Para trabajar introduciendo el tiempo en microsegundos y con un rango de 5 milisegundos hasta unos pocos microsegundos se ha de modificar el preescaler a 1 y la cuenta será:

$$\text{Cuenta} = \frac{\text{Tiempo} \cdot 10^{-6}}{8{,}3333 \cdot 10^{-8}} = \frac{\text{Tiempo} \cdot 100}{8{,}3333}$$

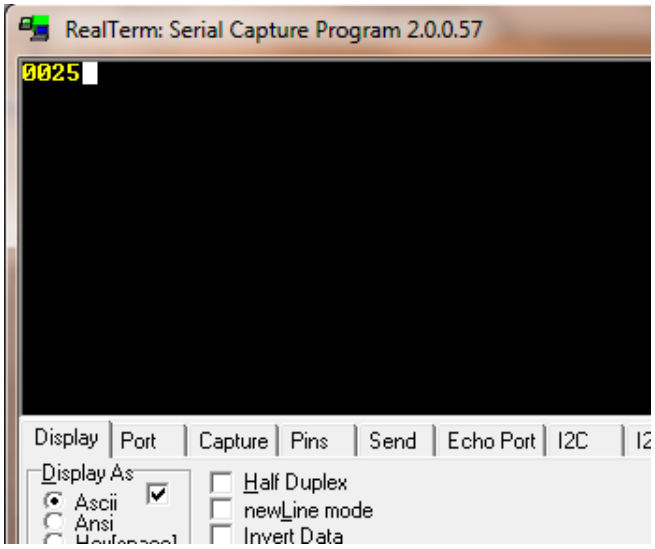


Figura 32.- Ejemplo de introducción del tiempo en formato Ascii por el Realterm.

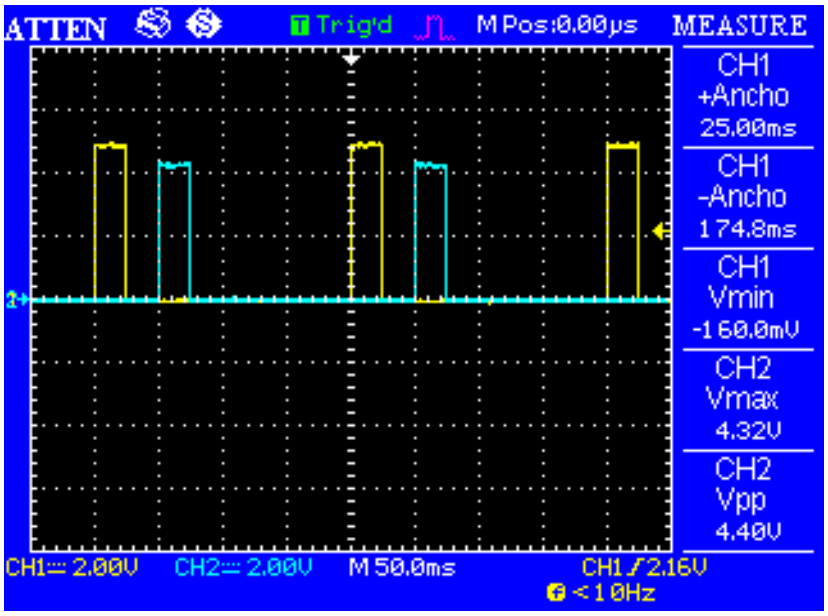


Figura 33.- Señal del osciloscopio con 25ms de tiempo entre parpadeo.

4.2.7. Muestreo entradas analógicas.

La siguiente modificación de código a estudiar será la forma de introducir el muestreo de las dos entradas analógicas (fotodiodos) a la mitad de cada pulso de los leds.

Para ello se aumenta la frecuencia con la que se interrumpe el timer0 al doble de la inicial. Para una mejor comprensión de lo que se va a realizar observar la siguiente imagen:

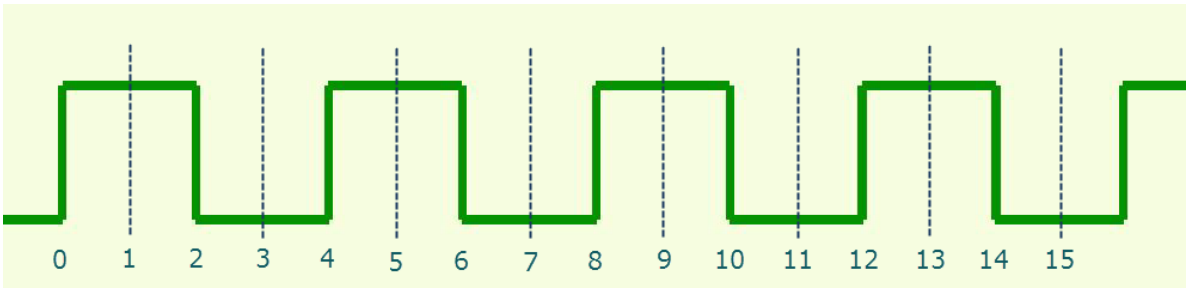


Figura 34.- Ciclo de encendido y apagado de los leds (verde) junto con el ciclo de interrupción del timer0.

El primer pulso corresponde al encendido del LED1, el segundo al LED2 y sucesivos.

De manera que en los pulsos 1, 5, 9 y 13 se realiza la lectura o habilitación del canal analógico (set_adc_channel(5)) y en los pulsos 2, 6, 10 y 14 se almacenan estos valores en un array ($\mathrm{ADC\_up}[4] = \mathrm{read\_adc}()$). Esto es así porque entre la habilitación de la entrada analógica y la lectura del valor se tiene que dejar un tiempo mínimo para que se obtenga el valor, llamado tiempo de adquisición (1,7us - 4us).

Además de este array también se utiliza el mismo procedimiento para crear otro en los valores bajos (ADC_down[4]), es decir en los pulsos 3, 7, 11 y 15. Esto se hace con el fin de conocer nuestro valor mínimo y poder eliminar posteriormente el error que se pueda estar cometiendo.

Este programa con una sola entrada analógica se puede encontrar en el anexo.

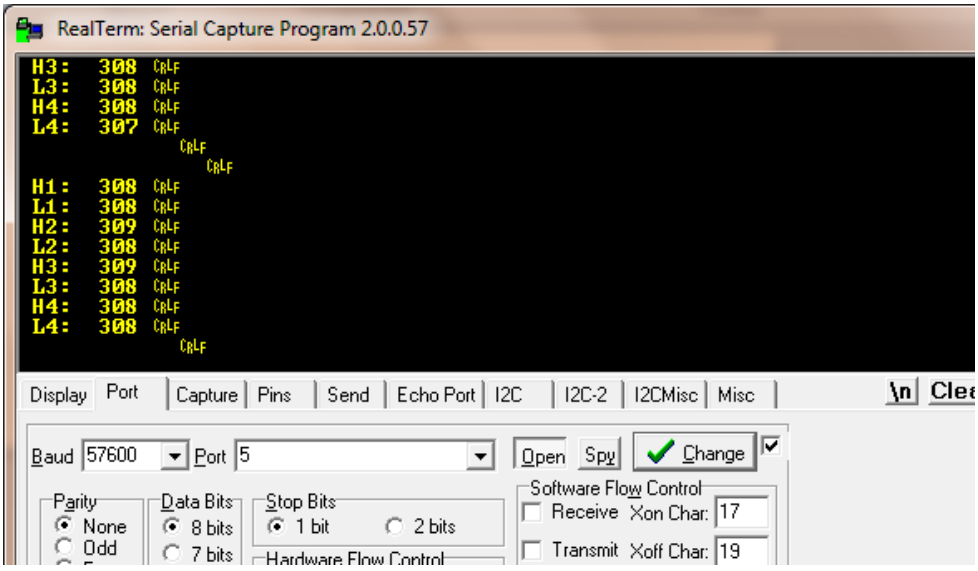


Figura 35.- Salida del código con una sola entrada. Valores en binario de 0 a 1025 bits.

Este procedimiento anterior nos sirve para una sola entrada analógica y debido a que no se pueden leer varias entradas analógicas de forma simultánea porque están multiplexadas se hace otro segundo programa en el cual:

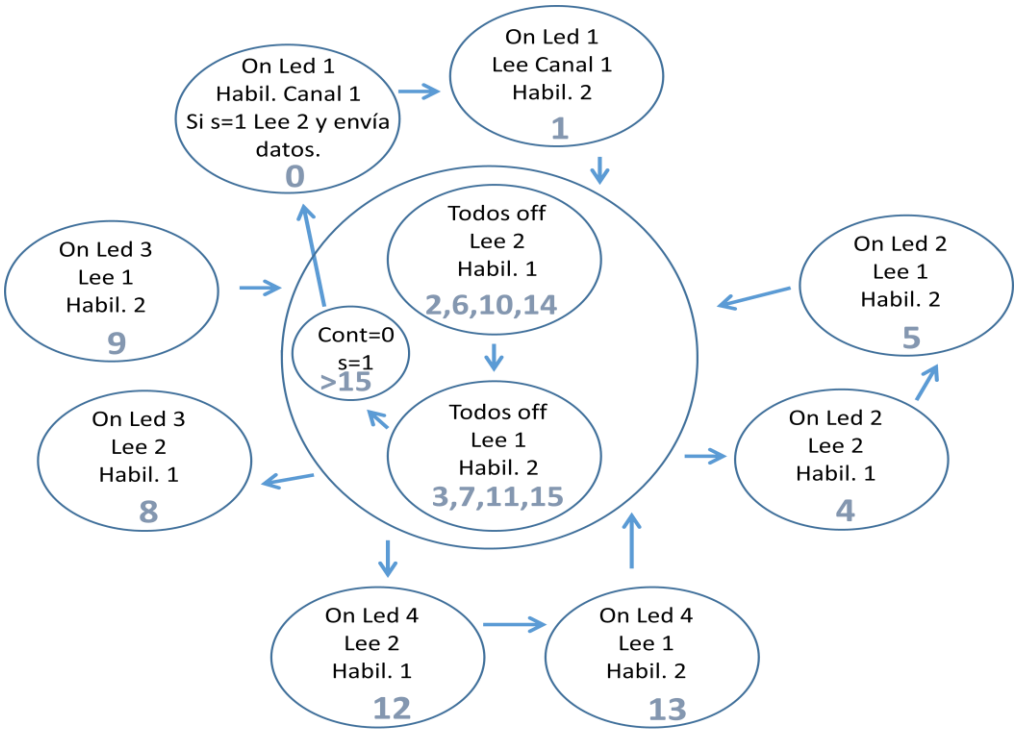


Figura 36.- Flujograma que muestra el ciclo de muestreo.

- En los valores 0, 4, 8 y 12 se pretende habilitar la entrada analógica 1 (canal 5).
- En los valores 1, 5, 9 y 13 se quiere leer la entrada 1 en el valor alto y habilitar la segunda (canal 8).
- En los valores 2, 6, 10 y 14 se lee la entrada 2 en nivel alto y se habilita la primera para proceder a leer el estado bajo.
- En los valores 3, 7 11 y 15 se lee la entrada 1 a nivel bajo y se habilita la 2.

Entonces en los primeros valores 0, 4 8 y 12 además de habilitar la entrada 1 también se quiere leer la entrada 2 en el nivel bajo. Sin embargo, no se quiere leer el nivel bajo de la entrada 2 en el valor 0, sino en el 16, por ello creamos la variable binaria "s". Que será 1 siempre a partir de la primera vuelta.

Cada una de las lecturas se almacena en un dato diferente "ADC_[n]".

Además el envío de los datos se realiza introduciendo la variable binaria "a". El programa envía los valores por el puerto serie cuando esta variable se hace verdadera, justo al acabar una rutina. Obsérvese en el Listado 09 rodeado por un círculo rojo como se introduce ésta en el programa.

```
#int_timer0
void isr_timer0()
{
  if(cont==0){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   if(s==1){
    ADC_down2[3]=read_adc();
   a=1;
   }
```

```
   set_adc_channel(5);
   }
   if(cont==1){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   a=0;
   ADC_up1[0]=read_adc();
   set_adc_channel(8);
   }
 [...]
   if((cont==3) || (cont==7) || (cont==11) || (cont==15)){
   output_low(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
    if(cont==3){
    ADC_down1[0]=read_adc();
    set_adc_channel(8);
    }
    if(cont==7){
    ADC_down1[1]=read_adc();
    set_adc_channel(8);
    }
    if(cont==11){
    ADC_down1[2]=read_adc();
    set_adc_channel(8);
    }
    if(cont==15){
    ADC_down1[3]=read_adc();
    set_adc_channel(8);
    }
   }
cont++;
if(cont>15){
s=1;
cont=0;}
set_timer0(precarga);
}
```

Listado 09.- Fragmento de la secuencia del muestreo de dos entradas analógicas.

AJUSTE DEL TIEMPO ENTRE PARPADEO

El resultado de la señal en el osciloscopio con estos datos cuando se introducen 125us devuelve un valor de 190us de periodo.

Esto es debido a que por tratarse de tiempos tan pequeños, el tiempo que tarda el microcontrolador en realizar todas las operaciones internas en la interrupción del timer0 influye y se suma al resultado.

Este tiempo si se tratara de una interrupción simple se obtiene del datasheet, sin embargo al realizar tantas tareas dentro de la función de interrupción éste aumentará.

Para poder obtener cual es dicho tiempo y poder restárselo al código, se procede a averiguarlo de forma empírica dando valores y observando en el osciloscopio.

El tiempo que tarda la interrupción es de 70us aproximadamente. En el código se tiene que reflejar esto, y se hace de la siguiente forma:

```
tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3])-70;
```

Se carga el código al microcontrolador y se comprueba el correcto funcionamiento de las entradas haciendo diversas pruebas como por ejemplo conectar uno de los pines a tierra como se muestra en la siguiente figura 37.

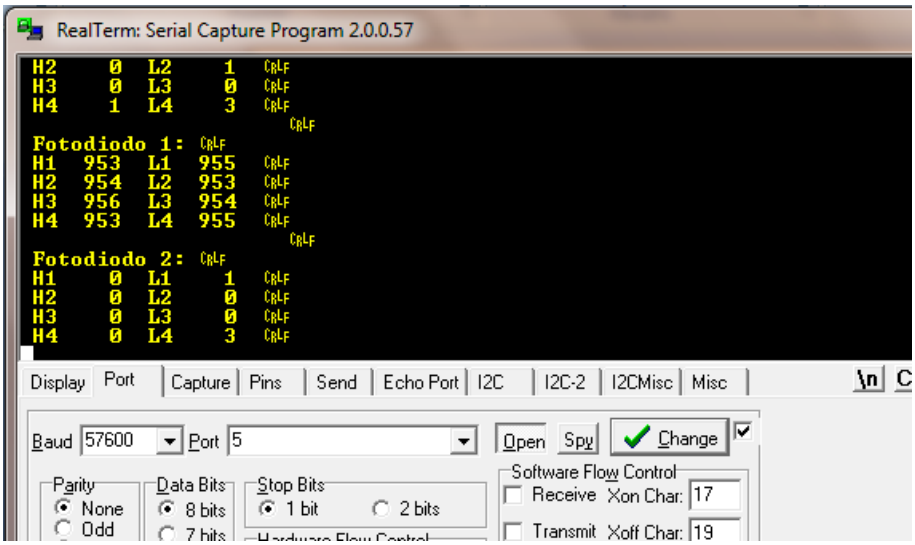


Figura 37.- Salida del código por Realterm I.

O se varia el potenciómetro viendo que se ajustan los valores en tiempo real.

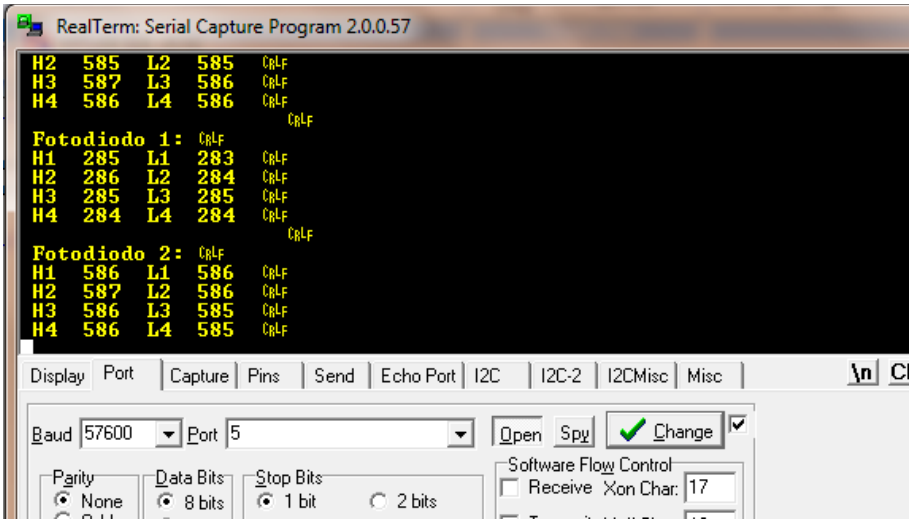


Figura 38.- Salida del código por Realterm II.

Tener en cuenta que estos valores se están imprimiendo en valores reales, es decir de 0 a 1025 bits que se imprimen directamente en pantalla. Esto se modificará más adelante para reducir el espacio que ocupan.

4.2.8. Uso de la señal PWM

La generación de señales PWM es una de las técnicas más utilizadas para realizar control con microcontroladores.

Para poder generar esta señal con el PIC, se hace uso de los módulos CCP (Comparador, Captura y PWM).

La modulación por ancho de pulsos (también conocida como PWM) permite obtener una señal periódica (Es decir, que se repite en el tiempo) la cual se puede modificar su ciclo de trabajo (Duty Cycle en ingles). Dicho PWM puede tener una resolución máxima de 10 bits**.** En otras palabras, como se sabe que el PIC trabaja con voltajes binarios (0V o +5V), se puede configurar el PWM para que trabaje un determinado tiempo en +5V (Ton) frente al tiempo que está en nivel bajo 0V (Toff), tal y como se puede apreciar en la figura 39.

De esta manera la tensión media aplicada a la carga es proporcional al tiempo en que la señal estuvo en +5V (Ton). Y así podemos por ejemplo controlar la luminosidad de lámparas o la velocidad de un motor.

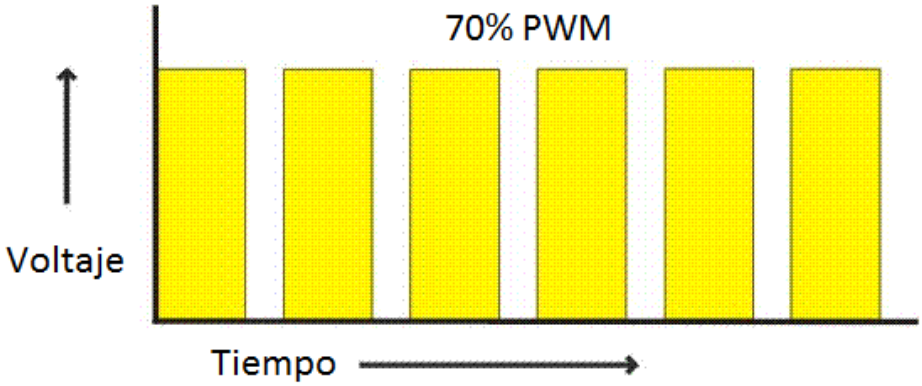


Figura 39.- Gráfica de la señal PWM.

Se hace un programa que únicamente establezca una señal PWM para empezar a entender este módulo. Para el manejo de la señal PWM en C existen dos instrucciones básicas.

- setup_ccpx(modo): Para configurar el módulo CCP.

Setup_CCPx(modo)	Modo	Registro CCPxCON
CCP_OFF	Deshabilitación	00000000
CCP_CAPTURE_FE	Captura por flanco de bajada	00000100
CCP_CAPTURE_RE	Captura por flanco de subida	00000101
CCP_CAPTURE_DIV_4	Captura cada 4 pulsos	00000110
CCP_CAPTURE_DIV_16	Captura cada 16 pulsos	00000111
CCP_COMPARE_SET_ON_MATCH	Salida a 1 en comparación	00001000
CCP_COMPARE_CLR_ON_MATCH	Salida a 0 en comparación	00001001
CCP_COMPARE_INT	Interrupción en comparación	00001010
CCP_COMPARE_RESET_TIMER	Reset TMR1 en comparación	00001011
CCP_PWM	Modo PWM habilitado	00001100

Figura 40.- Diferentes modos de configuración de la función setup_ccp.

De todas las configuraciones que se observan en la figura 40 se utiliza la ultima de todas, CCP_PWM para habilitar el modo PWM.

- set_pwmx_duty(valor): define el ciclo de trabajo y "valor" es un dato de 10 bits.

Para hacer uso de estas instrucciones y generar una señal de PWM con el PIC se tiene que utilizar el timer2. Éste es un contador de 8 Bits el cual usa el PIC internamente como base de tiempos para la modulación del PWM. Por ello la fórmula a usar para obtener el "valor" a introducir en la función anterior es:

$$\text{Ciclo de trabajo (Tanto por uno)} = \frac{\text{valor}}{4 \cdot (\text{Carga del Timer2} + 1)}$$

Para la configuración del Timer2 se cuenta con la función:

- setup_timer_2(Preescaler, Carga, Postscaler)

El preescaler puede ser de 1, 4 o 16 y la carga de timer2 solo puede tomar valores entre 0 y 255 debido a que es un registro de 8 bits.

Con la siguiente formula se pueden calcular los valores de la carga del timer2 en función de la frecuencia de salida deseada.

$$\text{PWM}_{\text{Frecuencia}} = \frac{\text{Frecuencia del cristal}}{4 \cdot (\text{Carga del Timer2} + 1) \cdot (\text{Preescaler del Timer2})}$$

La frecuencia debe ser lo más alta posible porque a la salida de la señal se aplicará un filtro paso bajo y para que haga un seguimiento rápido la constante de tiempo debe ser lo suficiente pequeña. Dado que en el filtro se cumple que:

$$f = \frac{1}{2\pi RC} = \frac{1}{2\pi\tau}$$

Se establece una frecuencia de 100kHz o 10kHz y luego se escoge la de 100kHz por ser más conveniente para el circuito.

Además, en el programa también se introducen diferentes valores de ciclo de trabajo para comprobar cuando funcionan los leds con la luminosidad y tiempo de respuesta deseados.

La señal de salida PWM se produce en el pin C5 (CCP1).

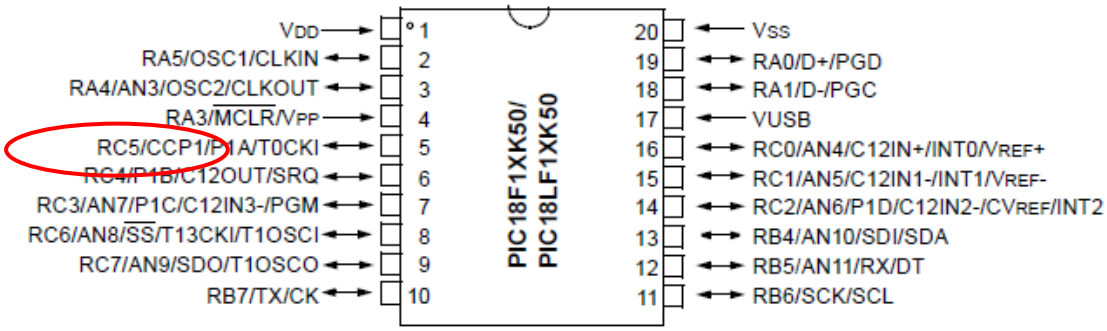


Figura 41.- Correspondencia de pines para el pic18F14K50

Dicho esto, el programa de prueba implementado es el siguiente:

```
void main(void)
{
long duty_PWM1=48; // 10% duty cycle on pin C5
long duty_PWM2=240; // 50% duty cycle on pin C5
long duty_PWM3=336; // 70% duty cycle on pin C5
long duty_PWM4=432; // 90% duty cycle on pin C5

output_low(PIN_C5);    // Set CCP1 output low
output_drive(PIN_C5); // A few older compilers, don't correctly set the TRIS
                          when this is done, but current ones do.
setup_timer_2(T2_DIV_BY_1, 119, 1);  // 100kHz
setup_ccp1(CCP_PWM);  // Configure CCP1 as a PWM

set_pwm1_duty(duty_PWM1);
delay_ms(5000);
set_pwm1_duty(duty_PWM2);
delay_ms(5000);
set_pwm1_duty(duty_PWM3);
delay_ms(5000);
set_pwm1_duty(duty_PWM4);
delay_ms(5000);

while(1); //Prevent PIC from going to sleep  (Important !)
}
```

Listado 10.- Ejemplo de señal PWM.

El procedimiento es el comentado anteriormente. Se introduce que cada 5 segundos cambie el porcentaje de ciclo de trabajo.

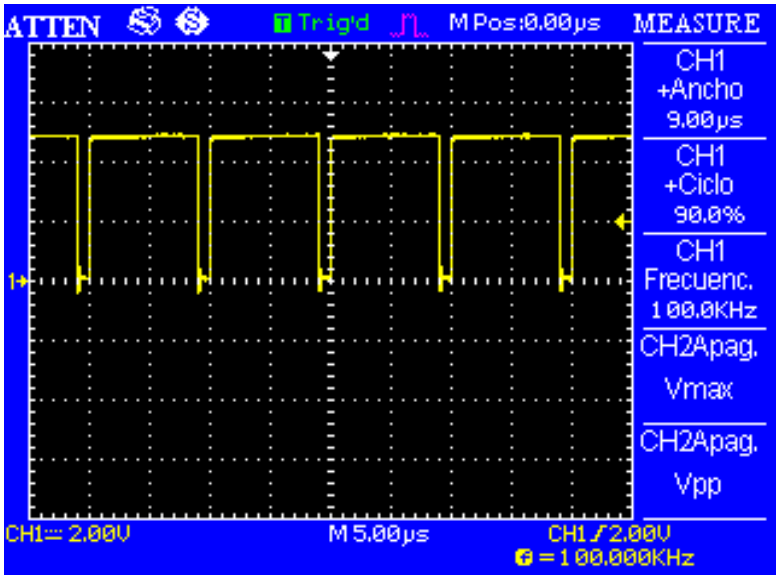


Figura 42.- Resultado en el osciloscopio de la señal PWM al 90% y con una frecuencia de 100kHz.

Ahora hay que introducir esta señal dentro del código de programa además de tener la posibilidad de poder introducir el porcentaje deseado por teclado como se hizo anteriormente con el tiempo entre parpadeo.

Se añade además tanto al introducir los datos como al obtenerlos en el Realterm una cabecera y un fin de mensaje. Esto servirá para detectar cuando empieza y termina cada bloque de texto enviado.

Las modificaciones de código importantes se observan en el siguiente fragmento:

```
while(true)
 {
  usb_task();
  usb_debug_task();

  if(usb_cdc_kbhit())
     {
     printf(usb_cdc_putc,"Cabecera |");

//Valor del PWM
       for(i=0;i<3;i++)
       {
       do{
        pwm[i]=usb_cdc_getc();
         }
while(pwm[i]!=48&&pwm[i]!=49&&pwm[i]!=50&&pwm[i]!=51&&pwm[i]!=52&&pwm[i]!=53
&&pwm[i]!=54&&pwm[i]!=55&&pwm[i]!=56&&pwm[i]!=57);

         pwm1[i]=(pwm[i]-48);
         printf(usb_cdc_putc,"%c" ,pwm[i]);
       }

       printf(usb_cdc_putc,"%% |FinDeMensaje");

      porcent=(pwm1[0]*100+pwm1[1]*10+pwm1[2]);
      duty_PWM=(4.8*porcent);

while(TRUE){

// PWM
output_low(PIN_C5);
output_drive(PIN_C5);
setup_ccp1(CCP_PWM);
setup_timer_2(T2_DIV_BY_1, 119, 1);
set_pwm1_duty(duty_PWM);
    }
   }
  }
}
```

Listado 11.- Incorporación de la señal PWM al programa.

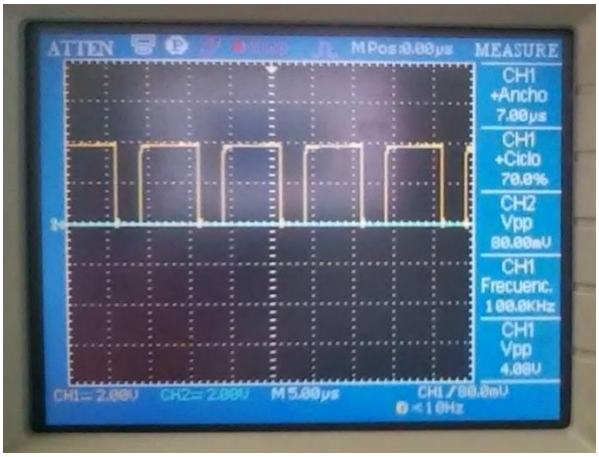


Figura 43.- Señal PWM observada en el osciloscopio obtenida por el pin C5 del microcontrolador.

A la salida del pin C5 por la que se obtiene la señal PWM se conecta el siguiente circuito con el fin de adaptar la señal para los leds.

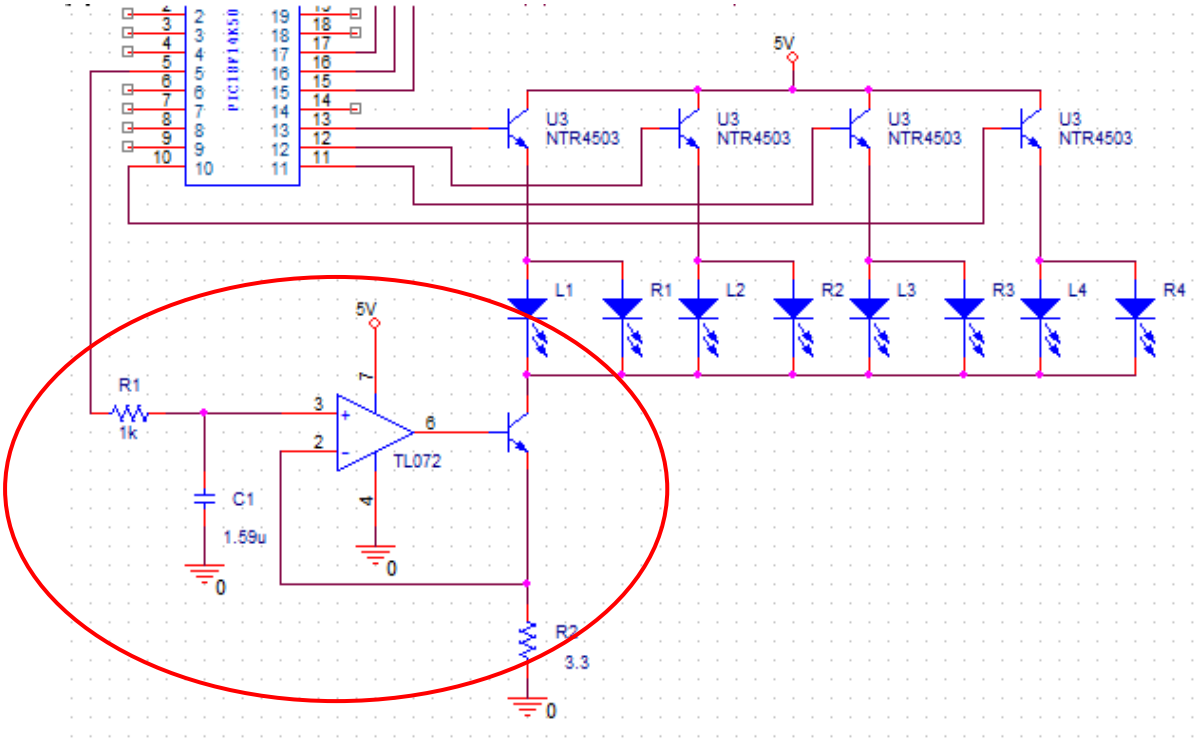


Figura 44.- Esquema de conexiones para adaptar la señal PWM.

DISEÑO Y CÁLCULO DEL FILTRO DE SALIDA PWM

Las especificaciones del esquema anterior referidas a los componentes se estudian más adelante, ahora solo se explican los detalles referidos a la señal PWM.

Se introduce un filtro paso bajo que recorte hasta frecuencias que sean la cuarta parte de la que se tiene, es decir, si tenemos una señal PWM de 100kHz, se aplica un filtro de 25kHz. Para calcular éste se tiene que:

$$f = \frac{1}{2\pi RC}$$

Entonces, introduciendo un valor arbitrario y razonable de uno de los dos componentes se puede calcular el otro con la herramienta Solver de Excel, escogiendo finalmente valores estándares de:

$$C = 680pF \text{ y } R = 10k\Omega$$

DISEÑO Y CÁLCULO DE LA FUENTE DE CORRIENTE

Se quiere una intensidad aproximada de 1A y para ello se introduce la resistencia R2.

Primero se necesita conocer el voltaje, se tienen 5 voltios de alimentación a los que se le tiene que restar las caídas de todos los componentes puestos en serie, los dos transistores y un diodo led.

$$5V - 0.1V - 1.4V - 0.2V = 3.3V$$

Teniendo esto, la resistencia que se tiene que introducir es de:

$$\mathrm{R2} = \frac{3.3\mathrm{V}}{1\mathrm{A}} = 3.3\Omega$$

Esta resistencia será entonces de una potencia de $\mathrm{P} = \mathrm{V} \cdot \mathrm{I} = 3.3\mathrm{W}$. Sin embargo, debido a que se trata de una señal pulsada con $\delta = 10\%$ basta con introducir una resistencia de una potencia de 0.33W.

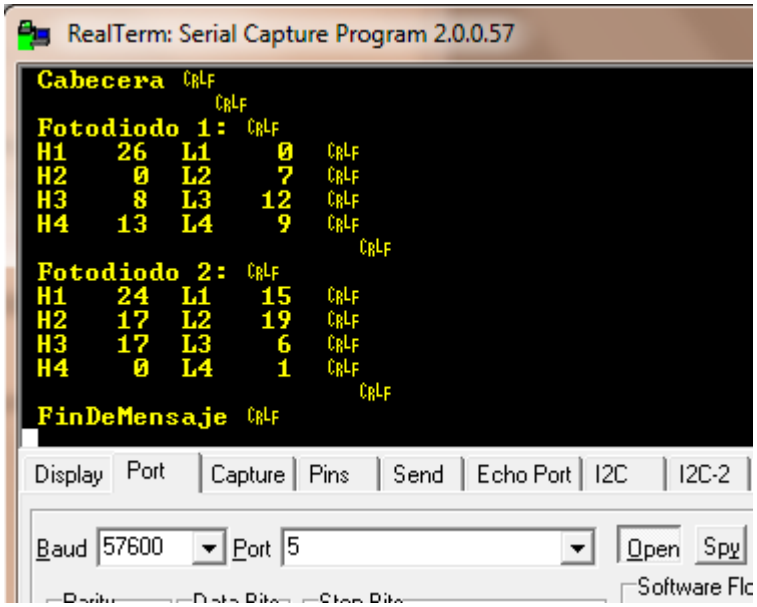


Figura 45.- Salida por Realterm.

Éste es el resultado final del programa. Sin embargo se puede afinar mucho más. Lo siguiente a modificar será que se usan enteros de 16 bits por cada uno de los caracteres y espacios que se envían, esto supone mucho tamaño y se está realizando un envío de información innecesaria. Ya que se quiere un programa que se basa en enviar datos se limita su tamaño procediendo al estudio de la transmisión binaria de datos.

4.2.9. Transmisión binaria.

Se quiere enviar los valores de las señales analógicas recibidas por los fotodiodos usando el mínimo espacio posible. Para ello lo primero que se hace es dividir el dato recibido de 16bits en dos datos de 8 bits. Esto se hará con la función:

```
msb=make8(dato_16bits, 1);
lsb=make8(dato_16bits, 0);
```

Donde "msb" hace referencia al bit más significativo (en inglés) y "lsb" al bit menos significativo:

Entonces simplemente se introduce esto en el programa exactamente a continuación de captar los valores analógicos. Dentro de cada uno de los ifs que se encuentran en la rutina de la función de interrupción, de la siguiente forma:.

```
if(cont==4){
   output_low(Led1);
   output_high(Led2);
   output_low(Led3);
   output_low(Led4);
   ADC_down2[0]=read_adc();
   msb_down20=make8(ADC_down2[0],1);
   lsb_down20=make8(ADC_down2[0],0);
   set_adc_channel(5);
   }
```

Listado 12.- Ejemplo de la aportación al código para la transmisión binaria de datos.

La cabecera y el final del mensaje serán dos valores aleatorios fijos, con la única condición de que sean imposibles de alcanzar por las señales analógicas de entrada, importante para poder reconocerlos y que nunca coincidan con cualquier valor de los datos. Se establecen por ejemplo los siguientes valores:

```
Cabecera=0xf0ff;
FinDeMensaje=0xf000;
```

La transmisión de los valores se hará en forma de caracteres:

```
  printf(usb_cdc_putc,"%c%c",msb_cabecera, lsb_cabecera);
  printf(usb_cdc_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c", msb_up10, lsb_up10,
   msb_down10, lsb_down10, msb_up11, lsb_up11, msb_down11, lsb_down11,
   msb_up12, lsb_up12, msb_down12, lsb_down12, msb_up13, lsb_up13,
   msb_down13, lsb_down13);
  printf(usb_cdc_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c", msb_up20, lsb_up20,
   msb_down20, lsb_down20, msb_up21, lsb_up21, msb_down21, lsb_down21,
   msb_up22, lsb_up22, msb_down22, lsb_down22, msb_up23, lsb_up23,
   msb_down23, lsb_down23);
  printf(usb_cdc_putc,"%c%c",msb_fdm, lsb_fdm);
```

Listado 13.- Fragmento de código donde imprime usando el comando %c.

Se observa la transmisión del mensaje de la cabecera seguido de los dos fotodiodos y el final de mensaje. En cada uno de los fotodiodos se envían los estados en nivel alto y bajo del led 1 (segundo subíndice de las variables a 0) y a continuación todos los demás leds con ese mismo procedimiento. El primer subíndice de las variables informa del fotodiodo.

Lo que se observa por la pantalla del Realterm son una serie de símbolos como se muestra en la figura 46.

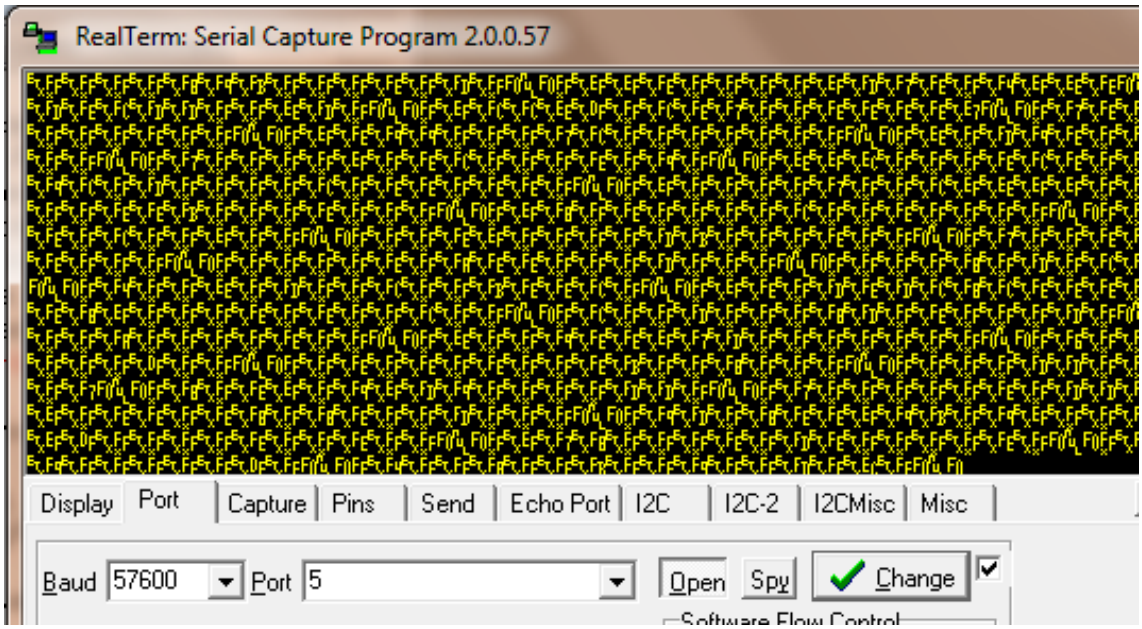


Figura 46.- Salida del Realterm en transmisión binaria.

Se pasa al formato Hex+Ascii (véase en la Figura 47) desde el mismo Realterm se puede comprobar que son correctos los datos. Se observa la cabecera y fin de mensaje además de que ambas salidas marcan el mismo valor estable alrededor de 03FF debido a que hemos conectado las salidas a la tensión de alimentación.

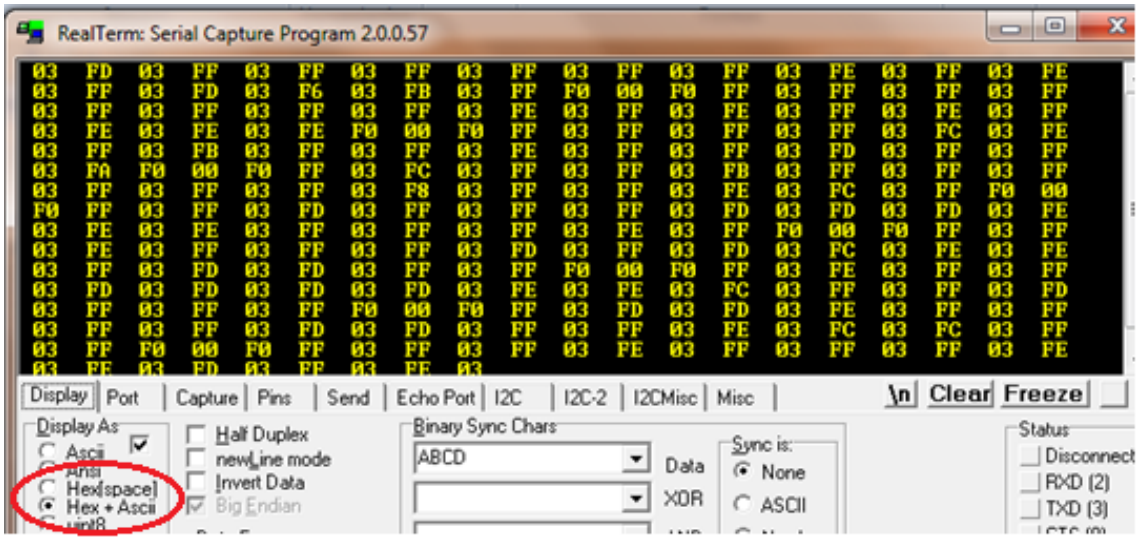


Figura 47.- Valores en Hex+Ascii.

Con esto, se procede a capturar una serie de datos para poder procesarlos posteriormente. Se hace esta captura en la pestaña "Capture" de Realterm creando así un bloc de notas.

El bloc de notas se puede abrir con el programa Frhed, y éste se encarga de leer los datos de forma hexadecimal.

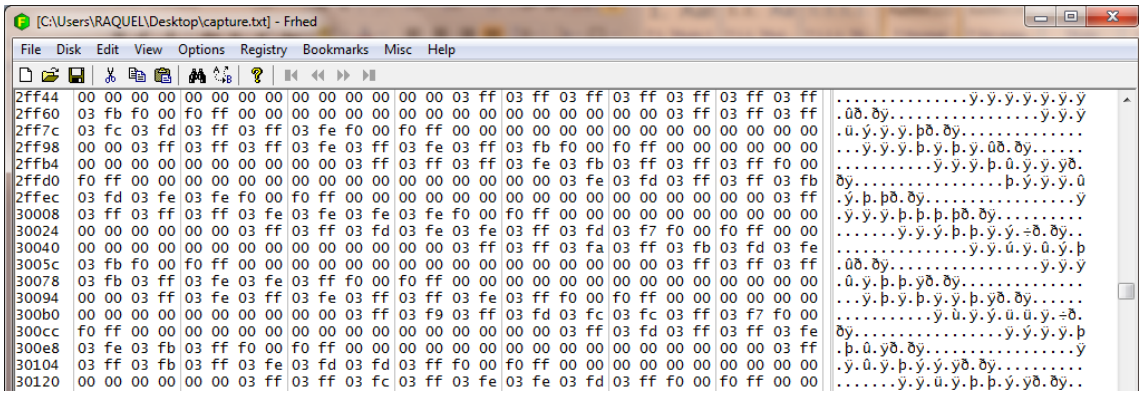


Figura 48.- Programa Frhed.

Obsérvese como dato que tenemos una entrada a cero y la otra marcando un valor casi estable de 03FF.

La captura de los datos se hace con el fin de enviar estos datos al programa Labview, cuyo cometido es la recepción de los datos para su posterior visualización, tanto en una tabla de valores como en una gráfica, para poder comprobar los datos de una manera más descriptiva.

4.2.10. Establecer valores predeterminados.

Se quiere hacer el programa más dinámico de manera que, de forma predeterminada y sin necesidad de introducir nada, funcione con un tiempo entre parpadeo de los leds de 125 microsegundos y con la señal PWM funcionando al 20% de duty. A su vez, también se quiere tener la posibilidad de modificar estos valores un número ilimitado de veces. Con esto el código de programa será más útil a la hora de trabajar con él.

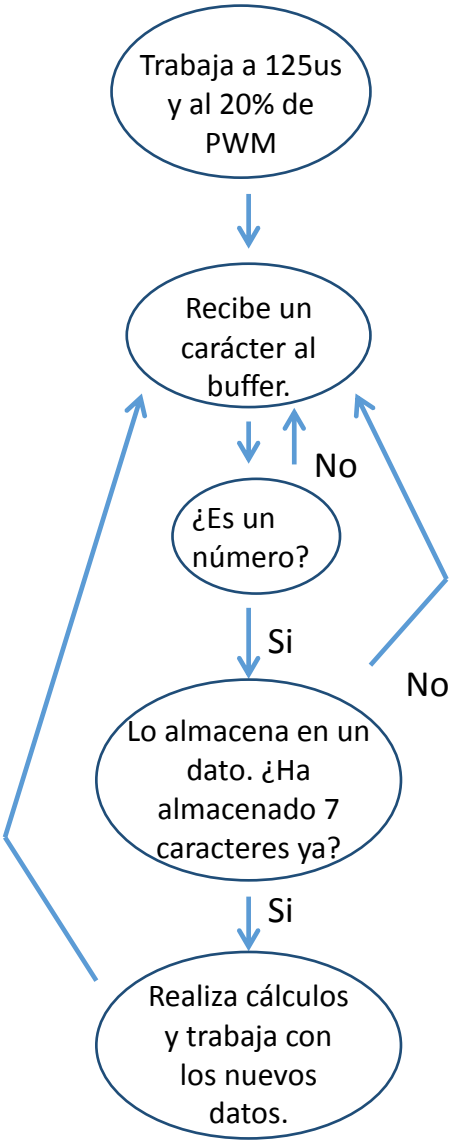


Figura 49.- Flujograma de la tarea principal del programa final.

En el anterior flujograma, el hecho de que necesite 7 caracteres es debido a, los cuatro anteriores correspondientes al tiempo de parpadeo, más tres que corresponden con el porcentaje del PWM.

Las modificaciones introducidas se basan en establecer la configuración de la señal PWM y del timer0 antes y fuera de la rutina while(true) a la que se entra para introducir valores nuevos. Con esto el programa funcionará previamente sin necesidad de introducir nada.

```
// Señal PWM
```

```
output_low(PIN_C5);
output_drive(PIN_C5);
setup_ccp1(CCP_PWM);
setup_timer_2(T2_DIV_BY_1, 119, 1);
set_pwm1_duty(duty_PWM);

// Interrupcion del timer0.
disable_interrupts(GLOBAL);
disable_interrupts(INT_TIMER0);
setup_timer_0(RTCC_DIV_1);
set_timer0(precarga);
enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);

while(true){ //Introducir nuevos valores.
usb_task();
usb_debug_task();

  if(usb_cdc_kbhit())
     {
      printf(usb_cdc_putc,"%c%c", msb_cabecera, lsb_cabecera); //Cabecera

      //Tiempo entre parpadeos de los LEDs
      for(i=0;i<4;i++)[...]

       //Valor del PWM
      for(i=0;i<3;i++)[...]

      printf(usb_cdc_putc,"%c%c",msb_fdm, lsb_fdm); //Fin de mensaje

disable_interrupts(GLOBAL);
disable_interrupts(INT_TIMER0);
setup_timer_0(RTCC_DIV_1);
set_timer0(precarga);
enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);

output_low(PIN_C5);
output_drive(PIN_C5);
setup_ccp1(CCP_PWM);
setup_timer_2(T2_DIV_BY_1, 119, 1);
set_pwm1_duty(duty_PWM);
     }
```

Listado 14.- Modificación para introducir unos valores predeterminados.

También se introduce la configuración de ambos cuando se termina de introducir los valores nuevos para así actualizarse.

Otra nueva aportación será la de introducir un comando justo detrás de la cabecera que sirva de contador (cont1). Éste facilitará las cosas para la posterior lectura y calibración de los datos.

```
[...]
while(usb_cdc_kbhit()==0){

 if(a==1){

  cont1++;
  if(cont1==65535){
  cont1=0;
  }

  printf(usb_cdc_putc,"%c%c",msb_cabecera, lsb_cabecera);
  printf(usb_cdc_putc, "%c",cont1);
[...]
```

Listado 15.- Modificación para introducir un contador de bloques enviados.

El Listado 15 muestra la secuencia en la que se envian los datos, la que se realizará siempre que no exista ningún dato esperando en el buffer, es decir, cuando se haya acabado de introducir todos `while(usb_cdc_kbhit()==0)` y cuando la variable a sea 1 es decir, cada vez que acabe un barrido de los 4 leds.

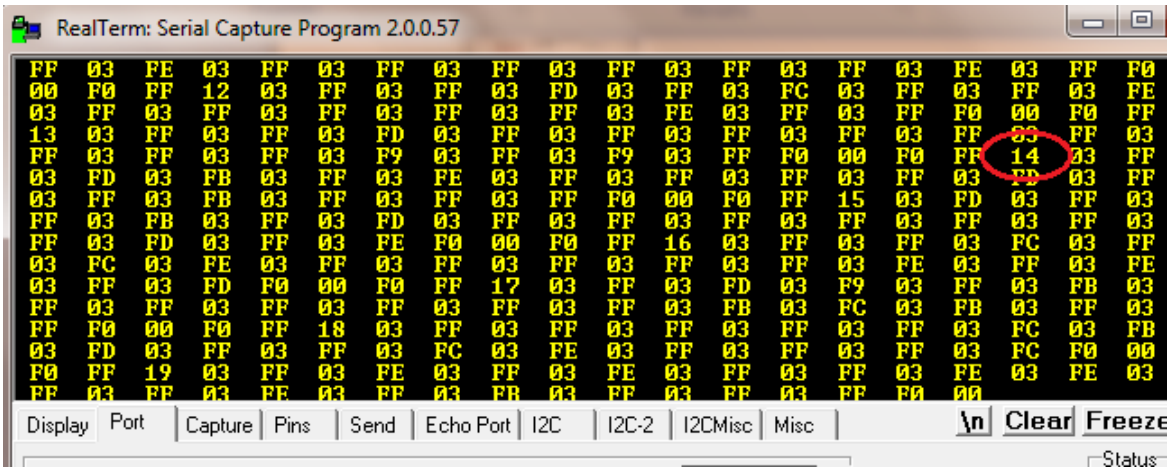


Figura 50.- Salida del Realterm donde se muestra el comando contador.

Se observa justo detrás del encabezado (F0 FF).

4.2.11. Verificación por redundancia cíclica.

La comprobación de redundancia cíclica (CRC) es un código de detección de errores en el que los bloques de datos que entran reciben un corto valor de comprobación adjunto basado en el resto de una división polinómica de su contenido.

Para el cálculo del código CRC se requiere la definición de un polinomio generador. Este polinomio se convierte en el divisor en la división larga polinómica, que toma el mensaje como el dividendo, el cociente se descarta y el resto se convierte en el resultado.

Este resultado es el valor de comprobación o CRC , obtenido para cada bloque de datos a ser enviados o almacenados y anexado a ellos.

Cuando se recibe una palabra de código, el dispositivo compara su valor de comprobación con un recién calculado a partir del bloque de datos, o de manera equivalente, realiza una CRC sobre toda la palabra de código y compara el valor de comprobación resultante con un esperado residuo constante.

Si los valores de control CRC no coinciden, entonces el bloque contiene un error de datos. Entonces el dispositivo puede tomar medidas correctivas, como no enviar el dato erróneo.

Existen diferentes tipos de CRC en función de la aplicación para la que se lo requiera. Se usa el tipo CRC-CCITT.

El polinomio generador para CRC-CCITT es el: $x^{16} + x^{12} + x^5 + 1$

Cuando se trabaja en CCS se tiene la librería crc.c, donde se encuentra la función:

- generate_16bit_crc(data, length, pattern)

Esta función genera un valor CRC para el dato introducido.

Al igual que en todos los apartados anteriores se empieza el estudio de esta sección con un pequeño ejemplo. En éste simplemente se obtiene el valor del CRC de un array fijo y se imprime junto con él para comprobar si funciona bien.

```
int8 i, Buffer[5];
int16 crc;

Buffer[0]=0x0f;
Buffer[1]=0x06;
Buffer[2]=0x0a;

while(true){
  crc = generate_16bit_crc(Buffer, 3, CRC_CCITT);
  Buffer[3]=make8(crc,1);
  Buffer[4]=make8(crc,0);

  for(i=0;i<5;i++){
   printf(usb_cdc_putc,"%c",Buffer[i]);
  }
}
}
```

Listado 16.- Pequeño ejemplo del uso de CRC.

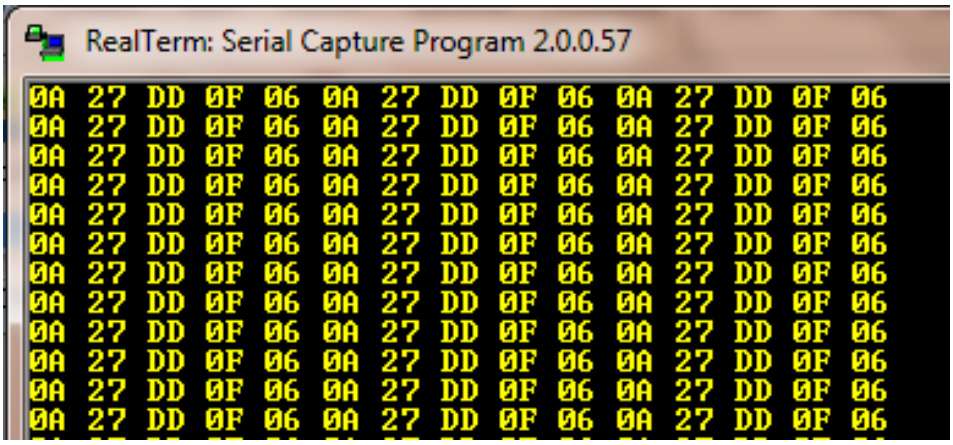


Figura 51.- Recepción del código de programa anterior con CRC.

Se observa que se recibe un valor de CRC de 27 DD en hexadecimal. Para comprobar si este valor es correcto se puede calcular sencillamente a través de una página de internet que lo calcula automáticamente.

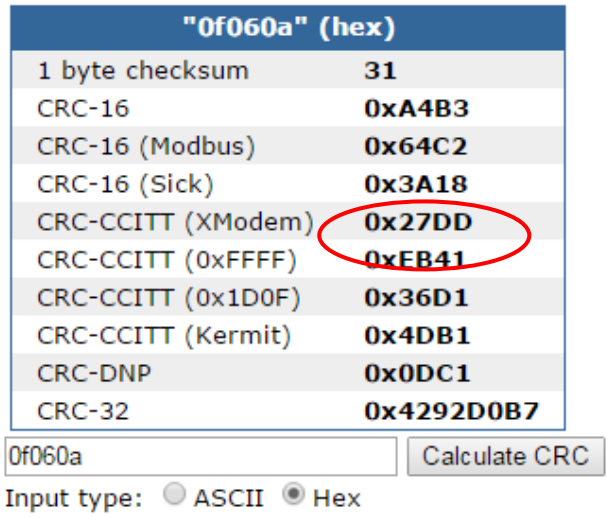


Figura 52.- Calculo a través de la calculadora CRC de la página web.

Para introducir esto en el programa final, el primer paso es transformar los datos que se envían a una sola matriz con el fin de poder verificar su correcta recepción más sencillamente.

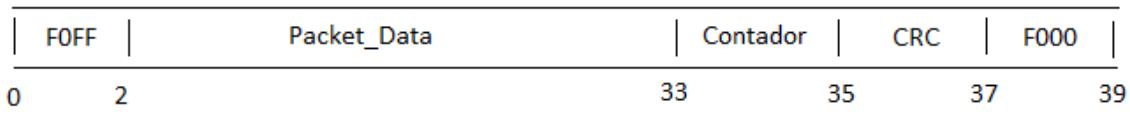


Figura 53.- Orden final del bloque de datos a enviar, matriz Buffer.

Se crea un array de 40 datos llamada Buffer como se observa en la figura 53, donde Packet_Data también se ha creado nueva como el conjunto de todas las señales analógicas que se reciben de ambas entradas.

```
void isr_timer0()
{
  if(cont==0){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   if(s==1){
    ADC_down2[3]=read_adc();
    Packet_Data[30]=make8(ADC_down2[3],1);
    Packet_Data[31]=make8(ADC_down2[3],0);

    CRC=generate_16bit_crc(Packet_Data, 32, CRC_CCITT);
    Buffer[36]=make8(CRC,1);
    Buffer[37]=make8(CRC,0);

      for(i=0;i<32;i++){
      Buffer[i+2]=Packet_Data[i];
       }
    a=1;
    }
   set_adc_channel(5);
   }
```

Listado 17.- Incorporación de CRC en el programa final.

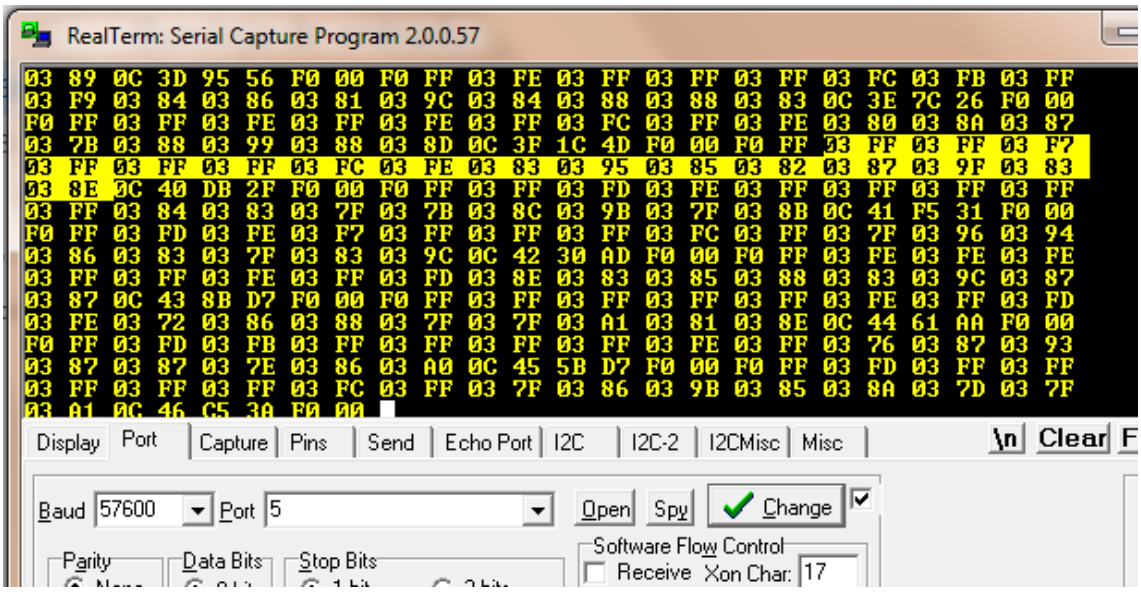


Figura 54.- Secuencia de datos recibidos en hexadecimal.

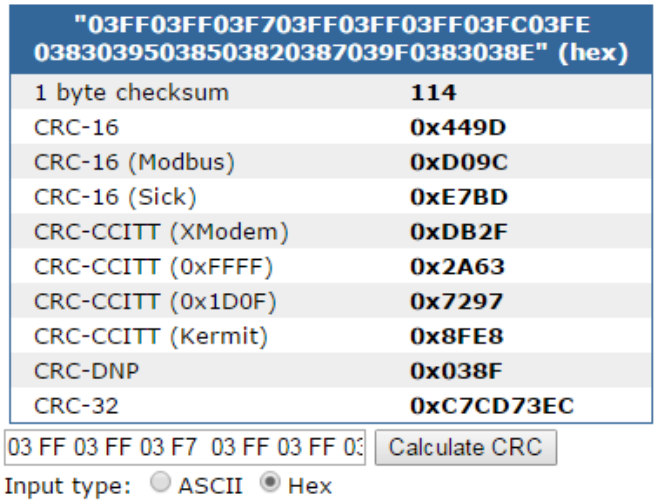


Figura 55.- Comprobación del valor CRC.

Sin embargo, al comprobar de nuevo el funcionamiento del código comprobamos que, al haber introducido la generación del CRC dentro de la rutina de interrupción, nos influye en los tiempos de parpadeo del Led1, haciendo éste del nivel de microsegundos.

Por ello, procedemos a medir cuanto tiempo tarda en generarse el valor del CRC con el uso de la función generate_16bits_CRC. Y lo haremos con el siguiente programa que pone a valor alto el pinB5 cada vez que se genera un valor de CRC.

```
void main(void)
{
Buffer[32]={0x03, 0xF7, 0x03, 0xFB, 0x03, 0xFF, 0x03, 0xFF, 0x03, 0xFF, 0x03,
0xFE, 0x03, 0xFF, 0x03, 0xFF, 0x03, 0x73, 0x03, 0xA0, 0x03, 0x89, 0x03, 0x8F,
0x03, 0x79, 0x03, 0x9C, 0x03, 0x86, 0x03, 0x98};

int16 crc;
output_low(PIN_B5);

while(true){
  output_high(PIN_B5);
  crc = generate_16bit_crc(Buffer, 32, CRC_CCITT);
  output_low(PIN_B5);
 }
}
```

Listado 18.-Comprobación del tiempo para generar el CRC.

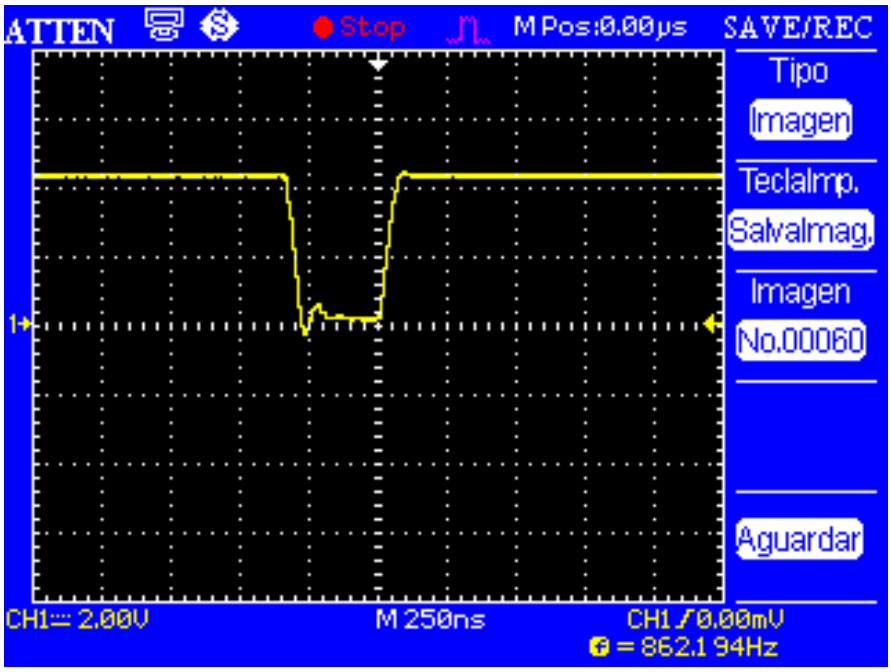


Figura 56: En el osciloscópio se observa que la frecuencia es de 862,194Hz.

Esto nos dice que el periodo de la señal es de la inversa de éste, 1,1598 ms. Restándole los 300ns aproximadamente en los que el pinb5 se encuentra en estado off, se queda mas o menos igual. También se midió el tiempo para un CRC de 8 bits simplemente modificando en el código la función generate_8bits_CRC.

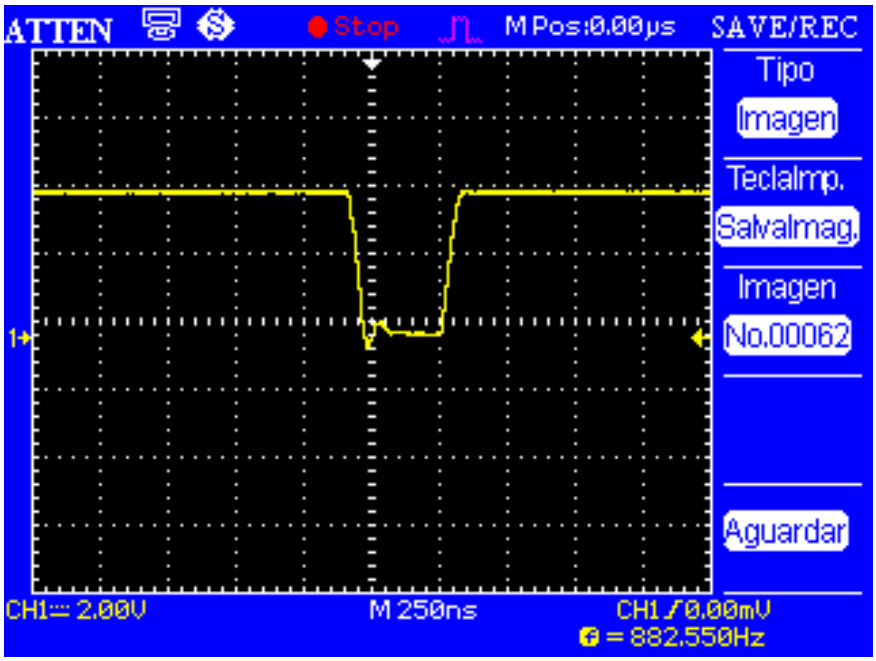


Figura 57:Frecuencia para un CRC de 8 bits.

Pero aún así el tiempo de compilación es demasiado elevado de 1,133ms.

Tal vez, para realizar la verificación de los datos en el programa habría que utilizar el lenguaje ensamblador o utilizar alguna otra función que tardase menos.

Al menos queda comprobada en esta sección que no se puede introducir el valor del CRC de los datos debido a que los tiempos con los que estamos trabajando son demasiado pequeños.

4.2.12. Código de programa final.

```
#include <18F14K50.h>
#device ADC=12
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>
#fuses  HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,
MCLR
#use delay(clock=48000000)

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int16 Cabecera=0xf0ff;
int16 FinDeMensaje=0xf000;

int32 precarga=65206;
int16 ADC_up1[4],ADC_down1[4],ADC_up2[4],ADC_down2[4];
unsigned int16 cont1;
int8 i, cont=0, Packet_Data[32], Buffer[38];
int1 s=0, a=0;

void usb_debug_task(void) //Esta función depura las tareas del USB.
{
 static int8 last_cdc;
 int8 new_cdc;

 new_cdc=usb_cdc_connected();
 last_cdc=new_cdc;
}

#int_timer0
void isr_timer0() // Secuencia de encendido/apagado y muestreo de entradas.
{
 if(cont==0){
  output_high(Led1);
  output_low(Led2);
  output_low(Led3);
  output_low(Led4);
  if(s==1){
   ADC_down2[3]=read_adc();
   Packet_Data[30]=make8(ADC_down2[3],1);
   Packet_Data[31]=make8(ADC_down2[3],0);
   a=1;
   }
  set_adc_channel(5);
  }
 if(cont==1){
  output_high(Led1);
  output_low(Led2);
  output_low(Led3);
  output_low(Led4);
  a=0;
  ADC_up1[0]=read_adc();
  Packet_Data[0]=make8(ADC_up1[0],1);
  Packet_Data[1]=make8(ADC_up1[0],0);
  set_adc_channel(8);
  }
```

```
 if((cont==2) || (cont==6) || (cont==10) || (cont==14)){
  output_low(Led1);
  output_low(Led2);
  output_low(Led3);
  output_low(Led4);
  if(cont==2){
   ADC_up2[0]=read_adc();
   Packet_Data[16]=make8(ADC_up2[0],1);
   Packet_Data[17]=make8(ADC_up2[0],0);
   }
  if(cont==6){
   ADC_up2[1]=read_adc();
   Packet_Data[20]=make8(ADC_up2[1],1);
   Packet_Data[21]=make8(ADC_up2[1],0);
   }
  if(cont==10){
   ADC_up2[2]=read_adc();
   Packet_Data[24]=make8(ADC_up2[2],1);
   Packet_Data[25]=make8(ADC_up2[2],0);
   }
  if(cont==14){
   ADC_up2[3]=read_adc();
   Packet_Data[28]=make8(ADC_up2[3],1);
   Packet_Data[29]=make8(ADC_up2[3],0);
   }
  set_adc_channel(5);
 }
 if((cont==3) || (cont==7) || (cont==11) || (cont==15)){
  output_low(Led1);
  output_low(Led2);
  output_low(Led3);
  output_low(Led4);
  if(cont==3){
   ADC_down1[0]=read_adc();
   Packet_Data[2]=make8(ADC_down1[0],1);
   Packet_Data[3]=make8(ADC_down1[0],0);
  }
  if(cont==7){
   ADC_down1[1]=read_adc();
   Packet_Data[6]=make8(ADC_down1[1],1);
   Packet_Data[7]=make8(ADC_down1[1],0);
  }
  if(cont==11){
   ADC_down1[2]=read_adc();
   Packet_Data[10]=make8(ADC_down1[2],1);
   Packet_Data[11]=make8(ADC_down1[2],0);
  }
  if(cont==15){
   ADC_down1[3]=read_adc();
   Packet_Data[14]=make8(ADC_down1[3],1);
   Packet_Data[15]=make8(ADC_down1[3],0);
  }
  set_adc_channel(8);
 }
 if(cont==4){
  output_low(Led1);
  output_high(Led2);
  output_low(Led3);
  output_low(Led4);
  ADC_down2[0]=read_adc();
  Packet_Data[18]=make8(ADC_down2[0],1);
```

```
  Packet_Data[19]=make8(ADC_down2[0],0);
  set_adc_channel(5);
 }
 if(cont==5){
  output_low(Led1);
  output_high(Led2);
  output_low(Led3);
  output_low(Led4);
  ADC_up1[1]=read_adc();
  Packet_Data[4]=make8(ADC_up1[1],1);
  Packet_Data[5]=make8(ADC_up1[1],0);
  set_adc_channel(8);
 }
 if(cont==8){
  output_low(Led1);
  output_low(Led2);
  output_high(Led3);
  output_low(Led4);
  ADC_down2[1]=read_adc();
  Packet_Data[22]=make8(ADC_down2[1],1);
  Packet_Data[23]=make8(ADC_down2[1],0);
  set_adc_channel(5);
 }
 if(cont==9){
  output_low(Led1);
  output_low(Led2);
  output_high(Led3);
  output_low(Led4);
  ADC_up1[2]=read_adc();
  Packet_Data[8]=make8(ADC_up1[2],1);
  Packet_Data[9]=make8(ADC_up1[2],0);
  set_adc_channel(8);
 }
 if(cont==12){
  output_low(Led1);
  output_low(Led2);
  output_low(Led3);
  output_high(Led4);
  ADC_down2[2]=read_adc();
  Packet_Data[26]=make8(ADC_down2[2],1);
  Packet_Data[27]=make8(ADC_down2[2],0);
  set_adc_channel(5);
 }
 if(cont==13){
  output_low(Led1);
  output_low(Led2);
  output_low(Led3);
  output_high(Led4);
  ADC_up1[3]=read_adc();
  Packet_Data[12]=make8(ADC_up1[3],1);
  Packet_Data[13]=make8(ADC_up1[3],0);
  set_adc_channel(8);
 }
 cont++;
 if(cont>15){
  s=1;
  cont=0;
 }
 set_timer0(precarga);
}
```

```
void main(void)
{
 usb_cdc_init();
 usb_init();
 usb_wait_for_enumeration();

 setup_adc(ADC_CLOCK_INTERNAL);
 setup_adc_ports(sAN5||sAN8); //Puertas C1 y C6

 int8 i, in_data[4], valores[4], pwm1[3], pwm[3];
 int16 tiempo, t;
 int32 cuenta;
 long duty_PWM=96, porcent;

 Buffer[0]=make8(Cabecera,1);
 Buffer[1]=make8(Cabecera,0);
 Buffer[36]=make8(FinDeMensaje,1);
 Buffer[37]=make8(FinDeMensaje,0);

 //PREDEFINIDO 125us Y 20% DE PWM

 // Señal PWM
 output_low(PIN_C5);
 output_drive(PIN_C5);
 setup_ccp1(CCP_PWM);
 setup_timer_2(T2_DIV_BY_1, 119, 1);
 set_pwm1_duty(duty_PWM);

 // Interrupcion del timer0.
 disable_interrupts(GLOBAL);
 disable_interrupts(INT_TIMER0);
 setup_timer_0(RTCC_DIV_1);
 set_timer0(precarga);
 enable_interrupts(GLOBAL);
 enable_interrupts(INT_TIMER0);

 while(true)
 {
  usb_task();
  usb_debug_task();

  //Introducir nuevos valores.
  if(usb_cdc_kbhit())
  {
   //Cabecera
   printf(usb_cdc_putc,"%c%c", Buffer[0], Buffer[1]);

   //Tiempo entre parpadeos de los LEDs
   for(i=0;i<4;i++)
   {
    do
    {
     in_data[i]=usb_cdc_getc();
    }
    while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51
    &&in_data[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55
    &&in_data[i]!=56&&in_data[i]!=57);
    valores[i]=(in_data[i]-48); //pasamos de ASCII a decimal
   }
   tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3])-70;
   printf(usb_cdc_putc,"%c",tiempo);
```

```
   //Valor del PWM
   for(i=0;i<3;i++)
   {
    do
    {
     pwm[i]=usb_cdc_getc();
    }
    while(pwm[i]!=48&&pwm[i]!=49&&pwm[i]!=50&&pwm[i]!=51&&pwm[i]!=52
    &&pwm[i]!=53&&pwm[i]!=54&&pwm[i]!=55&&pwm[i]!=56&&pwm[i]!=57);
    pwm1[i]=(pwm[i]-48);//pasamos de ASCII a decimal
   }
   porcent=(pwm1[0]*100+pwm1[1]*10+pwm1[2]);
   printf(usb_cdc_putc,"%c",porcent);

   //Fin de mensaje
   printf(usb_cdc_putc,"%c%c",Buffer[38], Buffer[39]);

   //Calculos previos
   t=tiempo/2;
   cuenta=(t/0.08333333);
   precarga=(65536-cuenta);
   duty_PWM=(4.8*porcent);

   //Configuración de los nuevos valores
   disable_interrupts(GLOBAL);
   disable_interrupts(INT_TIMER0);
   setup_timer_0(RTCC_DIV_1);
   set_timer0(precarga);
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_TIMER0);

   output_low(PIN_C5);
   output_drive(PIN_C5);
   setup_ccp1(CCP_PWM);
   setup_timer_2(T2_DIV_BY_1, 119, 1);
   set_pwm1_duty(duty_PWM);
  }
 // Imprimir datos.
  while(usb_cdc_kbhit()==0)
  {
   if(a==1)
   {
    for(i=0;i<32;i++)
    {
     Buffer[i+2]=Packet_Data[i];
    }
    cont1++;
    Buffer[34]=make8(cont1,1);
    Buffer[35]=make8(cont1,0);
    if(cont1==65535)
    {
     cont1=0;
    }
    for(i=0;i<38;i++)
    {
     printf(usb_cdc_putc,"%c",Buffer[i]);
    }
   }
  }
 }
}
```

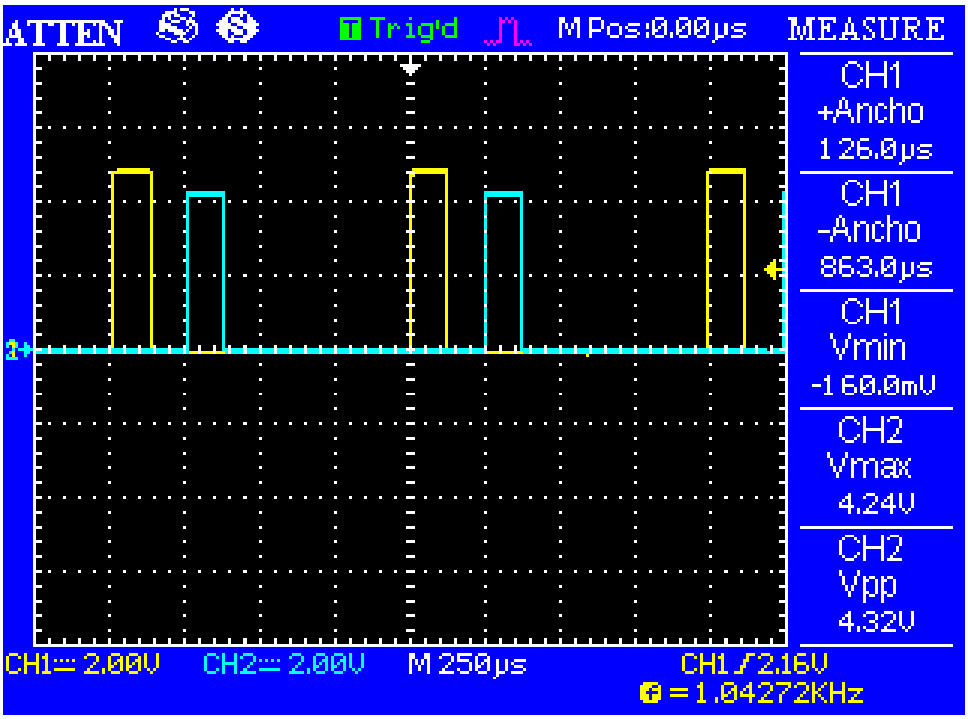


Figura 58: Resultado final leds.

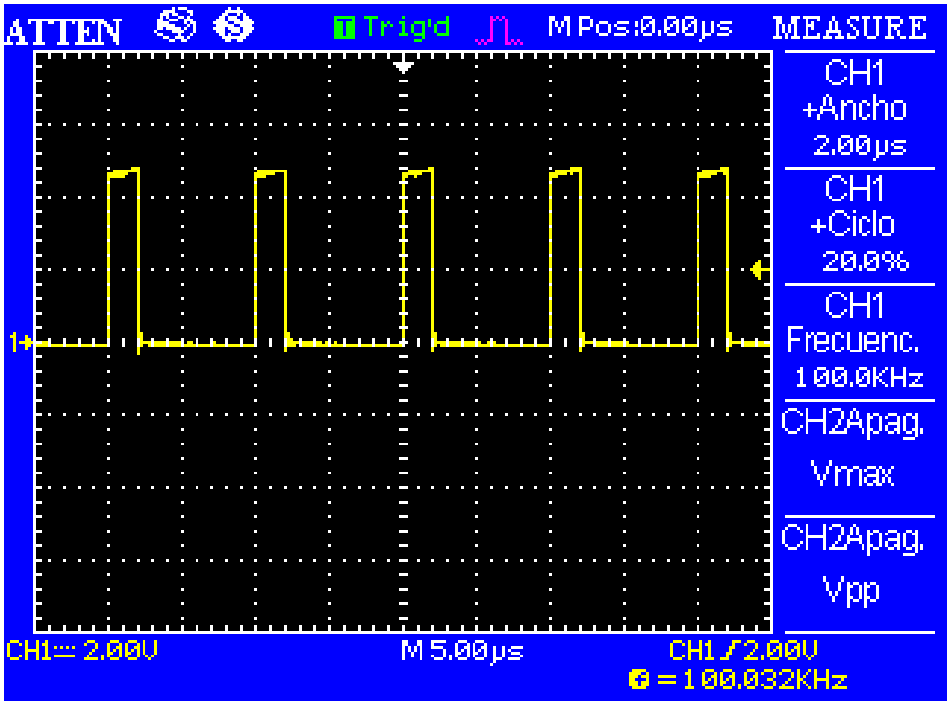


Figura 59: Resultado final PWM.

5. Desarrollo hardware.

5.1. Esquema.

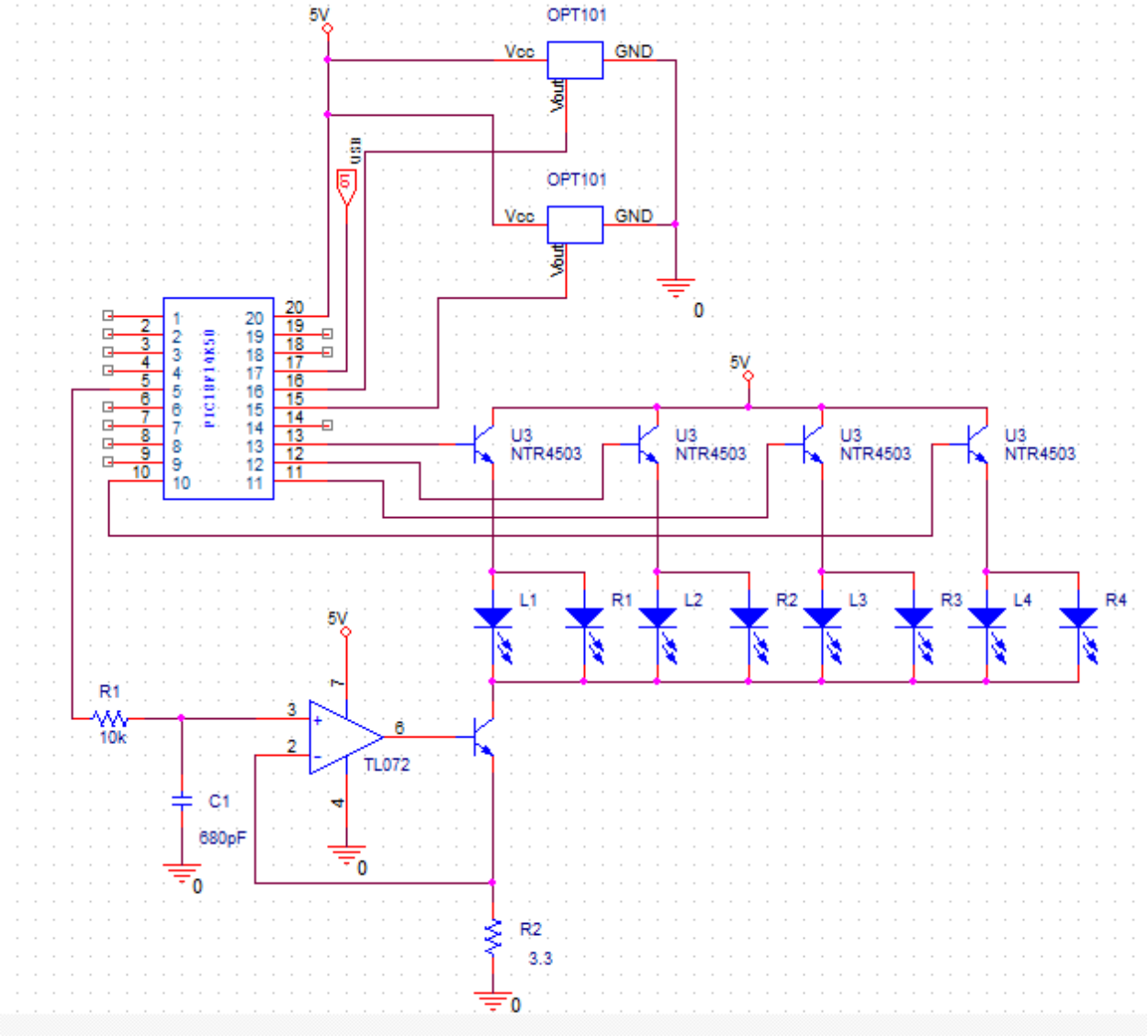


Figura 59.- Esquema simplificado bajo Orcad Capture.

5.2. Elección de componentes

5.2.1. Fotodiodos.

Se incorpora el encapsulado OPT101 de Texas Instruments. Se trata de fotodiodos monolíticos con amplificadores de transimpedancia. La combinación integrada de fotodiodos y amplificadores de transimpedancia en un solo chip elimina los problemas comúnmente encontrados en diseños discretos, tales como errores de fuga de corriente, levantamiento de ruido y un pico de ganancia como resultado de la capacitancia perdida. El voltaje de salida aumenta linealmente con la intensidad de la luz.

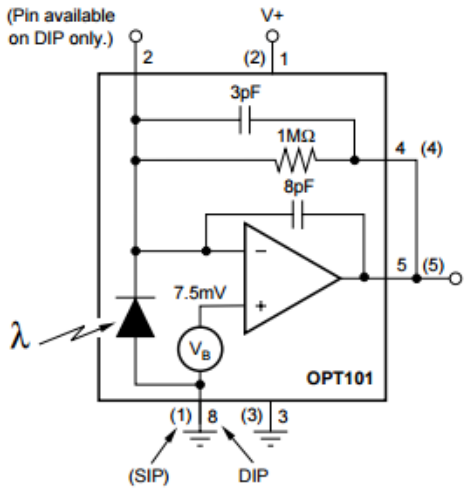


Figura 60.- Encapsulado del OPT101.

5.2.2. Diodos LEDs

Se escogen los emisores de luz infrarroja (IR) de la marca OSRAM Opto Semiconductors. Esta familia de LED IR de recepción superior se compone de emisores de infrarrojos de alta potencia. Estos emisores de recepción superior se suministran con encapsulado MIDLED de OSRAM.

Figura 61.- Emisor de luz infrarroja OSRAM Opto Semiconductors

Características de los LED IR de recepción superior MIDLED:

- Dimensiones: 3,3 x 2,35 x 1,7 mm
- Encapsulado MIDLED de perfil bajo
- Recepción superior
- Alta potencia
- Ángulos de haz estrechos

5.2.3. Amplificador

Se escoge el TL072.

5.2.4. Transistor.

En los 4 pines del pic correspondientes con las salidas a los leds se debe colocar un transistor en cada uno de ellos.

El transistor que se escogerá, debido a que es el que más se ajusta a los parámetros necesarios, será el NTR4503.

MAXIMUM RATINGS (T_J = 25°C unless otherwise noted)

Parameter			Symbol	Value	Unit
Drain−to−Source Voltage			V_{DSS}	30	V
Gate−to−Source Voltage			V_{GS}	±20	V
Continuous Drain Current (Note 1)	Steady State	T_A = 25°C	I_D	2.0	A
		T_A = 85°C		1.5	
	t ≤ 10 s	T_A = 25°C		2.5	
Power Dissipation (Note 1)	Steady State	T_A = 25°C	P_D	0.73	W
Continuous Drain Current (Note 2)	Steady State	T_A = 25°C	I_D	1.5	A
		T_A = 85°C		1.1	
Power Dissipation (Note 2)		T_A = 25°C	P_D	0.42	W
Pulsed Drain Current	t_p = 10 µs		I_{DM}	6.0	A
ESD Capability (Note 3)	C = 100 pF, RS = 1500 Ω		ESD	125	V
Operating Junction and Storage Temperature			T_J, T_{stg}	−55 to 150	°C

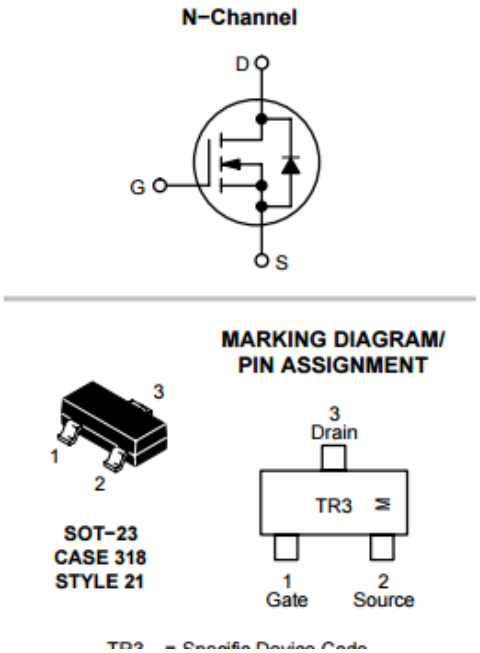


Figura 62.- Datos importantes del transistor a utilizar.

5.3. Detector todo/nada por reflectometría infrarroja.

Anteriormente a la programación del microcontrolador se empezó a estudiar en el laboratorio el funcionamiento de los leds y fotodiodos infrarrojos de forma analógica sin el uso de un micro.

Aunque esta opción finalmente se descartó, queda constancia de su implementación.

Se observa el siguiente circuito donde se obtiene un comportamiento de los LEDs y fotodiodo como detector todo/nada ante el acercamiento del paciente.

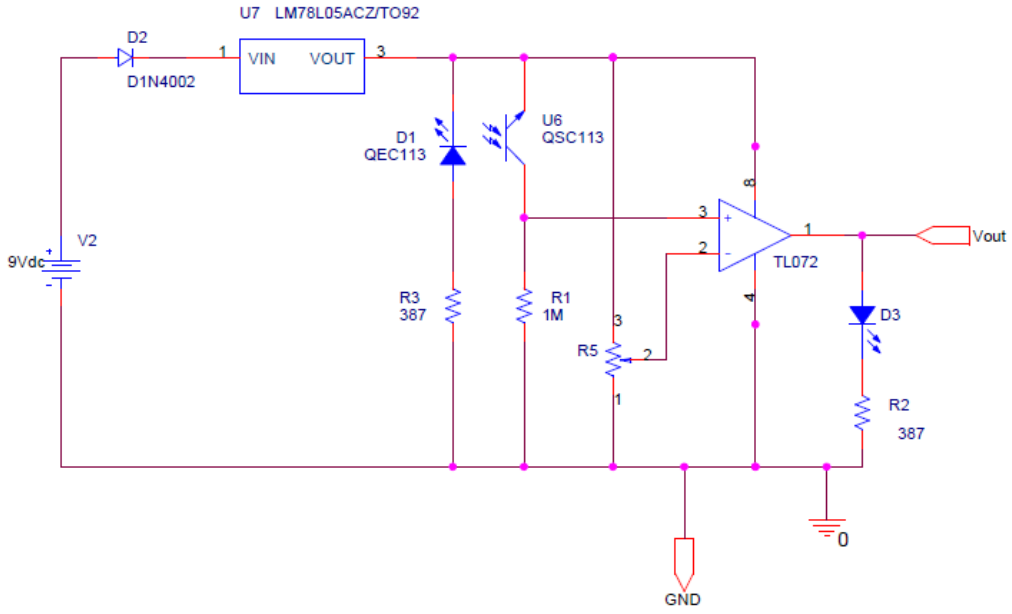


Figura 63.- Esquemático del circuito realizado con Orcad Capture.

Este circuito es una base para el estudio y ayuda a comprender el funcionamiento de los posteriores circuitos.

Se trata de un comparador, es decir un amplificador operacional en lazo abierto (sin realimentación entre su salida y su entrada) que se usa para comparar una tensión variable con otra tensión fija que se utiliza como referencia. En este caso esta tensión fija se ajusta con un potenciómetro (R5) hasta obtener el valor fijo deseado que haga que se cumpla lo siguiente.

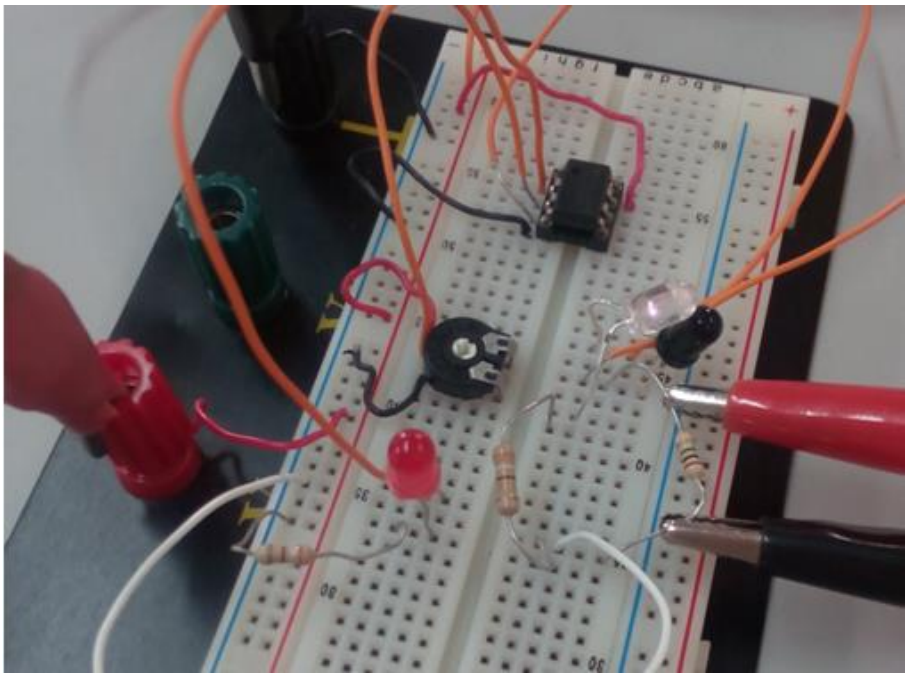


Figura 64: Placa de pruebas inicial del circuito.

El funcionamiento de este circuito consiste en que al aproximar un objeto sobre el LED infrarrojo se produce un reflejo de su luz sobre el fotodiodo, haciendo que reciba mayor cantidad. Este fotodiodo se encuentra conectado en la entrada positiva del amplificador de manera que cuando esto ocurre, la tensión en la patilla positiva supera la tensión de referencia impuesta por el potenciómetro, haciendo que se aumente la tensión a la salida y se encienda el LED rojo como respuesta.

Las pinzas que se pueden observar a la derecha de la fotografía corresponden a la entrada del osciloscopio. Se quiere observar la tensión que presenta la resistencia de 1MΩ cuando el LED rojo está encendido y cuando está apagado para comprobar si la diferencia es significativa o no. Estas pinzas producen en el circuito un efecto que hace reducir la sensibilidad del mismo cuando se produce un reflejo y esto es debido a que se le suma a la resistencia una impedancia en paralelo al osciloscopio que varía el valor de voltaje teniendo que modificar la referencia.

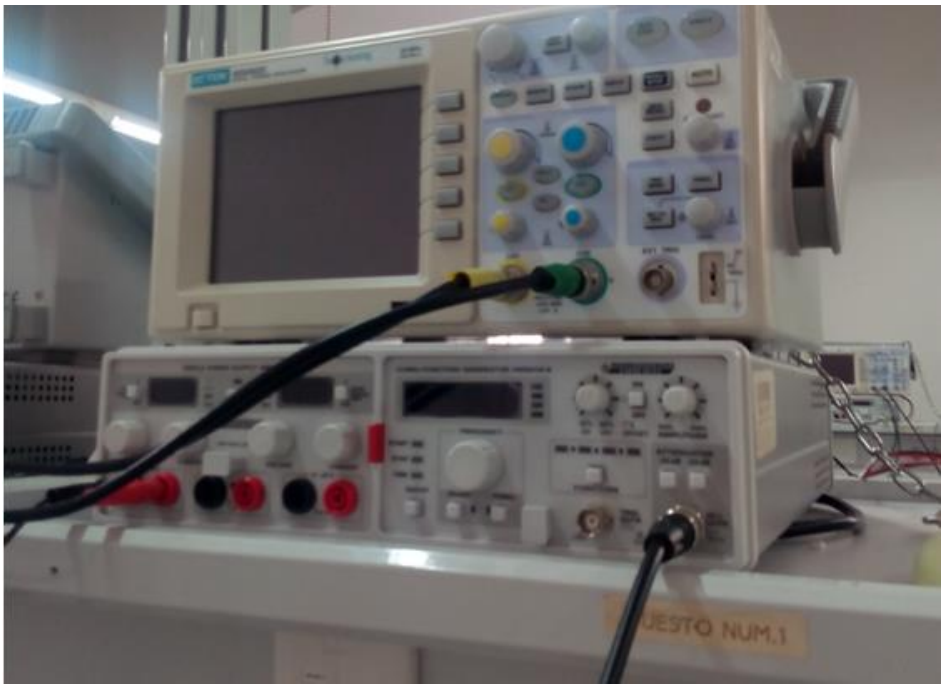


Figura 65.- Material de laboratorio utilizado. Osciloscopio y fuente de alimentación.

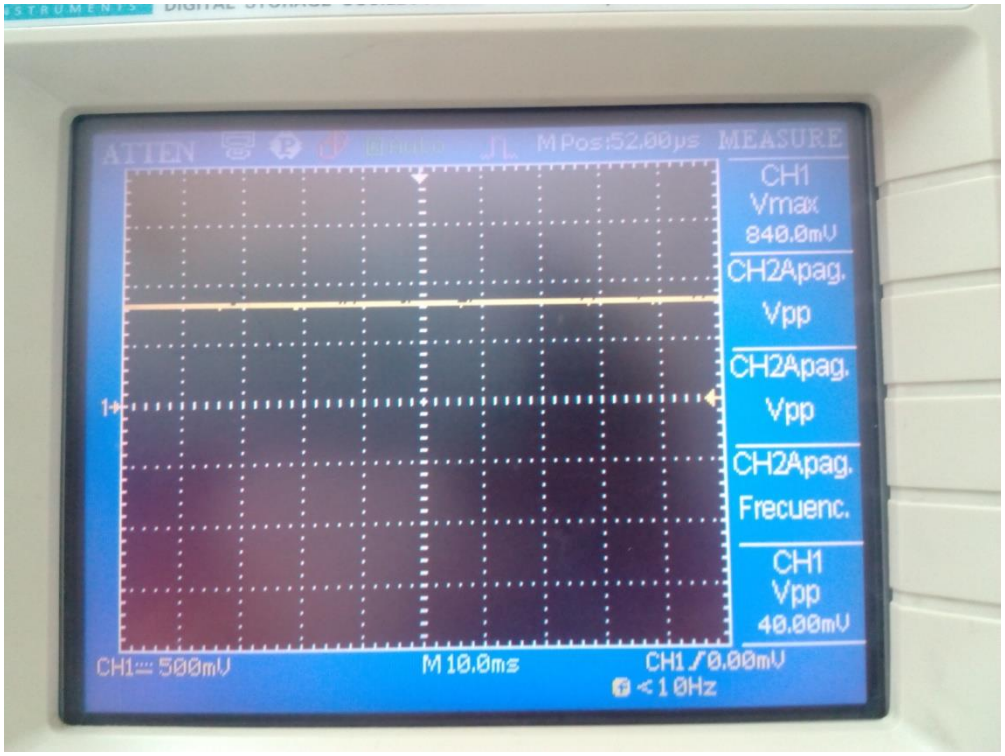


Figura 66.- Señal obtenida por el osciloscopio en la resistencia de 1MΩ cuando no se refleja la luz infrarroja.

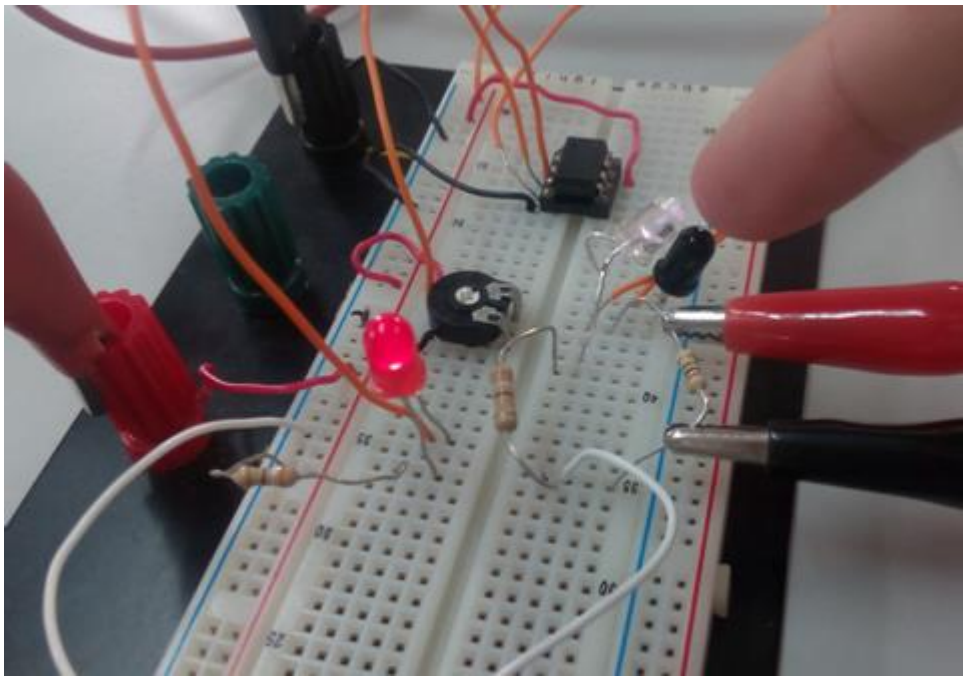


Figura 67.- Demostración del LED rojo encendido al reflejarse la luz infrarroja.

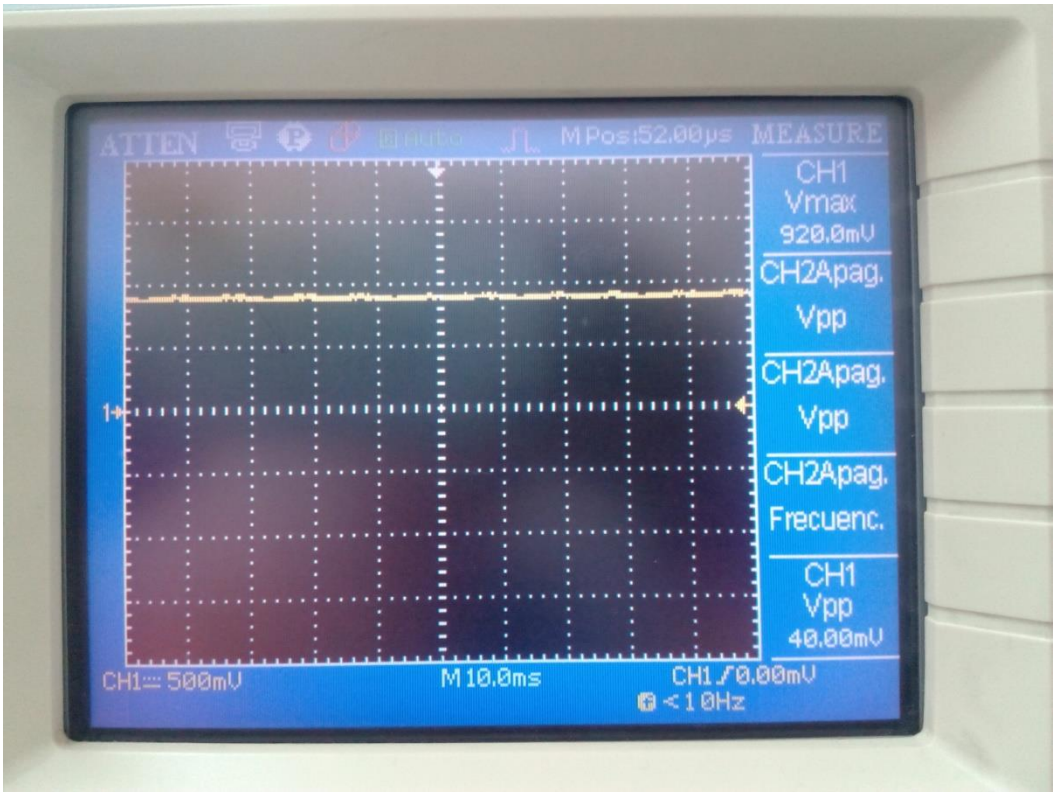


Figura 68.- Señal obtenida por el osciloscopio en la resistencia de 1MΩ cuando se refleja luz infrarroja.

Una vez obtenida una variación de señal significativa deseada se tiene que dejar constancia del circuito implementándolo en una placa. Para ello se hace uso del programa Orcad Layout, mediante el cual obtenemos el siguiente circuito.

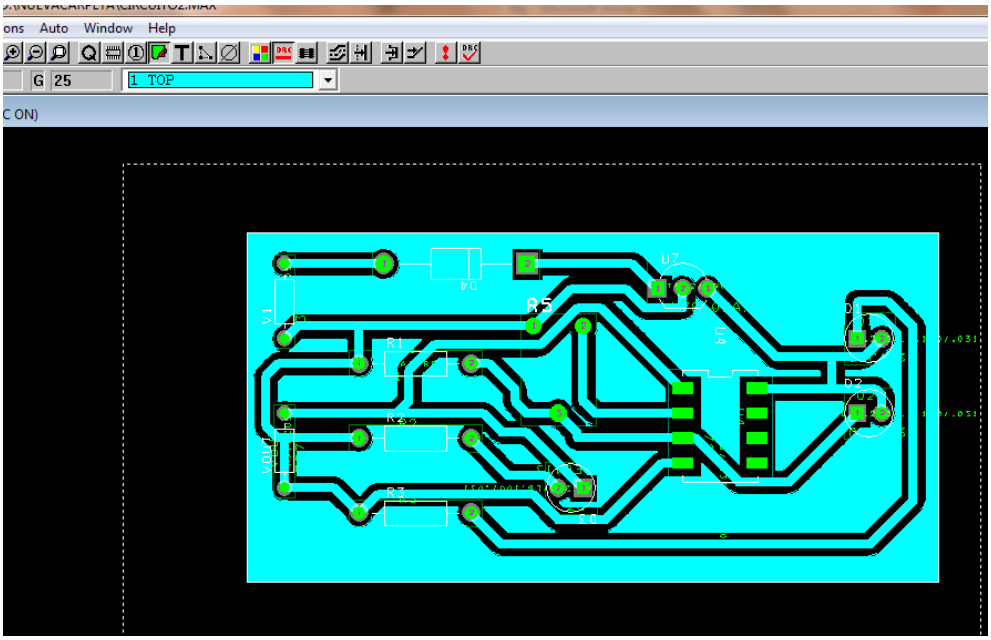


Figura 69.- Layout del PCB.

El ruteado de la placa se ha realizado a una sola cara, con pistas de 0.04 pulgadas de ancho y con un plano de masa con 0.03 pulgadas de clearance, es decir, de separación entre pistas.

Hay que tener especial cuidado con las islas de masa y tener unas separaciones lo bastante grandes entre las pistas y el cobre de alrededor para que no hayan problemas posteriores a la hora de soldar. Los pines también han de ser lo suficiente grandes para que no se rompan al taladrar.

El revelado de una placa se puede llevar a cabo de diferentes formas. Se opta por una placa de fibra de vidrio insolada con una máquina insoladora.

Para realizar el proceso es necesario imprimir la capa del Layout en la que quedan impresas solo las pistas y los pines y a ser posible en blanco y negro y en papel vegetal o por el estilo.

La insoladora hacía su efecto correcto en un tiempo estimado de unos 35 segundos, dando problemas en el posterior revelado si se excede o se queda corto de tiempo en un margen de unos 10 segundos.

Seguidamente se procede a revelar la placa con cuidado de no tocar la superficie insolada y sin que le de la luz directa (es recomendable hacer todo este proceso con la menor luz posible) se mete la placa insolada en la cubeta del revelador dejando la parte insolada hacia arriba para que no toque con el fondo del recipiente y se eche a perder la placa.

El líquido revelador se trata de una mezcla entre agua y sosa cáustica previamente preparada.

Una vez bien limpia de los restos del revelador, se debería poder ver el cobre brillante en las zonas donde no tiene que ir cobre y en las zonas de cobre se debería ver la película fotosensible oscura.

Ahora se procede al atacado de la placa con una solución de ácido e ir moviéndolo suavemente para que el líquido no se quede parado. En un tiempo aproximado de 5 minutos se podrá observar que ya se ha comido el cobre sobrante.

6. Conclusiones

Finalizado el desarrollo del trabajo fin de grado se ha considerado alcanzado el objetivo principal del TFG, el desarrollo y definición del firmware necesario para implementar un dispositivo capaz de obtener señales procedentes del posicionamiento ocular mediante reflectometría infrarroja.

De igual forma, se han logrado gran parte de los objetivos parciales planteados al inicio del desarrollo:

a) Puesta en marcha del entorno de desarrollo para la programación del microcontrolador

Se instaló y configuró con éxito el editor/compilador de C para microcontroladores PIC CCS v4.13. De igual forma se puso a punto el entorno de programación IDE MPLAB v8.92 de Microchip, configurando con éxito el programador finalmente utilizado (Microchip PICkit 3).

b) Estudio de las capacidades de la plataforma elegida mediante la programación de programas elementales que permitan estudiar siguientes funcionalidades

Se programaron distintos bloques de código para la evaluación del control de entradas/salidas digitales, las rutinas de temporización, conversión analógico/digital, las comunicaciones USB CDC y la conversión digital/analógico.

De forma adicional se implementaron distintas rutinas para implementar funcionalidades no contempladas inicialmente tales como las comunicaciones USB CDC en modo binario y la verificación de la integridad de paquetes mediante el cálculo del CRC (Verificación por redundancia cíclica).

c) Especificación y selección de componentes.

Aunque el objetivo principal del TFG se centra en el el desarrollo y definición del firmware necesario, se ha considerado adecuado incluir la selección y especificación de algunos componentes hardware imprescindibles para la consecución del mismo tales como el microcontrolador, los diodos LEDS, los fotodiodos autoamplificados y los componentes del sistema de excitación y control de los LEDs, cuyas características pueden consultarse en las hojas de datos adjuntas en el anexo 04.

d) Documentación del proyecto

Se ha incluido un estudio de costes y presupuesto del proyecto, su planificación (incluyendo diagrama de Gantt) y se ha llevado a cabo una revisión de la normativa inicialmente considerada.

De forma adicional, se ha llevado a cabo una revisión de la normativa de seguridad fotobiológica, planteado el estudio y caracterización del dispositivo propuesto. A pesar de no haber sido no contemplada inicialmente, se ha considerado interesante de cara a facilitar la homologación del dispositivo final.

Limitaciones del desarrollo

Finalmente han quedado fuera del desarrollo del proyecto el diseño del esquemático final, las placas de circuito impreso y el estudio de mercado puesto que finalmente han sido llevados a cabo por otros miembros del equipo investigador. Sin embargo, se ha incluido el diseño y definición del aquellas secciones del esquemático directamente relacionadas con el funcionamiento del firmware propuesto (subsistema de disparo, fuente de corriente y control PWM de los iluminadores LEDs y los fotodiodos amplificados).

La implementación del algoritmo de cálculo de CRC16 no está lo suficientemente optimizada para su uso en tiempo real con la frecuencia de muestreo impuesta como requisito de diseño. Sin embargo, las pruebas realizadas en el laboratorio parecen indicar que si sería posible su uso de forma satisfactoria si el muestreo se realizase a 500 Hz en lugar de los 1000 Hz considerados. Otra solución podría ser la migración a otra plataforma de mayor potencia computacional (Raspberry Pi, dsPic, etc.)

Conclusiones y futuros trabajos

La solución propuesta cumple con los requisitos impuestos en la propuesta original.

Se han incluido algunas funcionalidades no contempladas inicialmente como las comunicaciones en modo binario y el control de la intensidad de la iluminación mediante el diseño de una fuente de corriente controlada en PWM.

Las comuniaciones en modo texto limitan la velocidad de muestreo al aumentar de forma muy significativa el ancho de banda utilizado frente a las comunicaciones en modo binario.

El control por redundancia cíclica supone una gran carga computacional, lo que limita la funcionalidad sobre la plataforma elegida (PIC 18F14K50 @ 48MHz).

Como futuros trabajos se plantea el uso de cuatro fotodiodos y la inclusión de sensores adicionales para la aquisición de otras modalidades (acelerómetro/IMU, sensor temperatura, pulsioximetría, etc.). En este caso sería interesante estudiar la migración a una plataforma computacional más rápida.

Anexos.

Anexo 01. Códigos de programa.

Introducción del tiempo entre parpadeo con la función delay.

```
#include <18F14K50.h>
#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int32 precarga;
int8 cont=0;

void usb_debug_task(void)
{
   static int8 last_cdc;
   int8 new_cdc;
   new_cdc = usb_cdc_connected ();
   last_cdc = new_cdc;
}

void main(void)
{
 int8 in_data[4];
 int8 i;
 int16 tiempo, valores[4];
 int32 cuenta;

 usb_cdc_init ();
 usb_init ();
 usb_wait_for_enumeration ();

 while(true)
 {
  usb_task();
  usb_debug_task();
  if (usb_cdc_kbhit())
  {
   for (i=0;i<4;i++)
   {
    do
    {
     in_data[i] =usb_cdc_getc();
    }
    while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51
    &&in_data[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55
    &&in_data[i]!=56&&in_data[i]!=57);

    valores[i] =(in_data[i] -48); //pasamos de ASCII a decimal
```

```
    printf (usb_cdc_putc, "%c", in_data[i]);
   }

   tiempo =(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3]);

   while(TRUE)
   {
   output_high(PIN_B4);
   delay_ms(tiempo);
   output_low(PIN_B4);
   delay_ms(tiempo);
   }
  }
 }
}
```

Introducción del tiempo entre parpadeos con interrupciones del timer0.

```
#include <18F14K50.h>
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>

#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)
#bit SBOREN = 0xfd0.1

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int32 precarga;
int8 cont=0;

void usb_debug_task(void)
{
 static int8 last_cdc;
 int8 new_cdc;
 new_cdc=usb_cdc_connected();
 last_cdc=new_cdc;
}

void main(void)
{
 usb_cdc_init(); //Configura la comunicación USB
 usb_init(); //Inicializa el hardware del USB
 usb_wait_for_enumeration(); //Espera hasta que el PicUSB sea configurado por
el host

 int16 in_data[4],i;
 int32 tiempo, cuenta, valores[4];

 while(true)
 {
  usb_task();
  usb_debug_task();

  if(usb_cdc_kbhit())
  {
   for(i=0;i<4;i++)
   {
    do
    {
     in_data[i]=usb_cdc_getc();
    }
    while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51
    &&in_data[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55
    &&in_data[i]!=56&&in_data[i]!=57);

    valores[i]=(in_data[i]-48);//pasamos de ASCII a decimal
    printf(usb_cdc_putc,"%c",in_data[i]);
   }

   tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3]);
```

```
   cuenta=(tiempo*0.001)/0.0000213333333;
   precarga=(65536-cuenta);

   disable_interrupts(GLOBAL);
   disable_interrupts(INT_TIMER0);
   setup_timer_0(RTCC_DIV_256);
   set_timer0(precarga);
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_TIMER0);

   while(TRUE){}
  }
 }
}

#int_timer0
void isr_timer0()
{
 if(cont==0){
  output_high(Led1);
  output_low(Led2);
  output_low(Led3);
  output_low(Led4);
  }
 if(cont==2){
  output_low(Led1);
  output_high(Led2);
  output_low(Led3);
  output_low(Led4);
  }
 if(cont==4){
  output_low(Led1);
  output_low(Led2);
  output_high(Led3);
  output_low(Led4);
  }
 if(cont==6){
  output_low(Led1);
  output_low(Led2);
  output_low(Led3);
  output_high(Led4);
  }
 if((cont==1) || (cont==3) || (cont==5) || (cont==7)){
  output_low(Led1);
  output_low(Led2);
  output_low(Led3);
  output_low(Led4);
  }
 cont++;
 if(cont>7){
  cont=0;
  }
 set_timer0(precarga);
}
```

Muestreo de una sola entrada analógica.

```
#include <18F14K50.h>
#device ADC=10
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>

#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int32 precarga;
int16 ADC_up[4],ADC_down[4];
int8 cont=0;
int1 s=0;

void usb_debug_task(void)
{
 static int8 last_cdc;
 int8 new_cdc;
 new_cdc=usb_cdc_connected();
 last_cdc=new_cdc;
}

#int_timer0
void isr_timer0()
{
 if(cont==0){
  output_high(Led1);
  output_low(Led2);
  output_low(Led3);
  output_low(Led4);
  s=0;
  }
 if(cont==4){
  output_low(Led1);
  output_high(Led2);
  output_low(Led3);
  output_low(Led4);
  }
 if(cont==8){
  output_low(Led1);
  output_low(Led2);
  output_high(Led3);
  output_low(Led4);
  }
 if(cont==12){
  output_low(Led1);
  output_low(Led2);
  output_low(Led3);
  output_high(Led4);
  }
 if(cont==1){
  output_high(Led1);
  output_low(Led2);
```

```
 output_low(Led3);
 output_low(Led4);
 set_adc_channel(5);
 }
if(cont==5){
 output_low(Led1);
 output_high(Led2);
 output_low(Led3);
 output_low(Led4);
 set_adc_channel(5);
 }
if(cont==9){
 output_low(Led1);
 output_low(Led2);
 output_high(Led3);
 output_low(Led4);
 set_adc_channel(5);
 }
if(cont==13){
 output_low(Led1);
 output_low(Led2);
 output_low(Led3);
 output_high(Led4);
 set_adc_channel(5);
 }
if((cont==2) || (cont==6) || (cont==10) || (cont==14)){
 output_low(Led1);
 output_low(Led2);
 output_low(Led3);
 output_low(Led4);
 if(cont==2){
  ADC_up[0]=read_adc();
  set_adc_channel(5);
  }
 if(cont==6){
  ADC_up[1]=read_adc();
  set_adc_channel(5);
  }
 if(cont==10){
  ADC_up[2]=read_adc();
  set_adc_channel(5);
  }
 if(cont==14){
  ADC_up[3]=read_adc();
  set_adc_channel(5);
  }
 }
if((cont==3)||(cont==7)||(cont==11)||(cont==15)){
 output_low(Led1);
 output_low(Led2);
 output_low(Led3);
 output_low(Led4);
 if(cont==3){
  ADC_down[0]=read_adc();
  }
 if(cont==7){
  ADC_down[1]=read_adc();
  }
 if(cont==11){
  ADC_down[2]=read_adc();
  }
```

```
  if(cont==15){
   ADC_down[3]=read_adc();
   s=1;
   }
  }
 cont++;
 if(cont>15){
  cont=0;
  }
 set_timer0(precarga);
}

void main(void)
{
 usb_cdc_init(); //Configura la comunicación USB
 usb_init(); //Inicializa el hardware del USB
 usb_wait_for_enumeration(); //Espera hasta que el PicUSB sea configurado

 setup_adc(ADC_CLOCK_INTERNAL);
 setup_adc_ports(sAN5|VSS_VDD); //Puerta C1

 int16 i,in_data[4];
 int32 tiempo, t,cuenta, valores[4];

 while(true)
 {
  usb_task();
  usb_debug_task();

  if(usb_cdc_kbhit())
  {
   for(i=0;i<4;i++)
   {
   //La función do-while hace que sea imposible introducir letras.
    do{
     in_data[i]=usb_cdc_getc();
     }
     while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51
         &&in_data[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55
         &&in_data[i]!=56&&in_data[i]!=57);

    valores[i]=(in_data[i]-48);//pasamos de ASCII a decimal
    printf(usb_cdc_putc,"%c",in_data[i]);
   }

   tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3]);
   t=tiempo/2;
   cuenta=(t*0.001)/0.0000213333333;
   precarga=(65536-cuenta);

   disable_interrupts(GLOBAL);
   disable_interrupts(INT_TIMER0);

   setup_timer_0(RTCC_DIV_256);
   set_timer0(precarga);

   enable_interrupts(GLOBAL);
   enable_interrupts(INT_TIMER0);

   while(TRUE){
    if(s==1){
```

```
      printf(usb_cdc_putc,"\r\n H1: %4ld \r\n L1: %4ld \r\n H2: %4ld \r\n L2:
      %4ld \r\n H3: %4ld \r\n L3: %4ld \r\n H4: %4ld \r\n L4: %4ld \r\n\r\n",
      ADC_up[0], ADC_down[0], ADC_up[1], ADC_down[1], ADC_up[2], ADC_down[2],
      ADC_up[3], ADC_down[3]);
     }
    }
   }
  }
}
```

Muestreo de las dos entradas analógicas.

```
#include <18F14K50.h>
#device ADC=12
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>

#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int32 precarga;
int16 ADC_up1[4],ADC_down1[4],ADC_up2[4],ADC_down2[4];
int8 cont=0;
int1 s=0, a=0;

void usb_debug_task(void)
{
 static int8 last_cdc;
 int8 new_cdc;
 new_cdc=usb_cdc_connected();
 last_cdc=new_cdc;
}

#int_timer0
void isr_timer0()
{

  if(cont==0){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   if(s==1){
    ADC_down2[3]=read_adc();
    a=1;
    }
   set_adc_channel(5);
   }
   if(cont==4){
   output_low(Led1);
   output_high(Led2);
   output_low(Led3);
   output_low(Led4);
   ADC_down2[0]=read_adc();
   set_adc_channel(5);
   }
   if(cont==8){
   output_low(Led1);
   output_low(Led2);
   output_high(Led3);
   output_low(Led4);
   ADC_down2[1]=read_adc();
   set_adc_channel(5);
   }
```

```
if(cont==12){
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_down2[2]=read_adc();
set_adc_channel(5);
}
if(cont==1){
output_high(Led1);
output_low(Led2);
output_low(Led3);
output_low(Led4);
a=0;
ADC_up1[0]=read_adc();
set_adc_channel(8);
}
if(cont==5){
output_low(Led1);
output_high(Led2);
output_low(Led3);
output_low(Led4);
ADC_up1[1]=read_adc();
set_adc_channel(8);
}
if(cont==9){
output_low(Led1);
output_low(Led2);
output_high(Led3);
output_low(Led4);
ADC_up1[2]=read_adc();
set_adc_channel(8);
}
if(cont==13){
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_up1[3]=read_adc();
set_adc_channel(8);
}

if((cont==2) || (cont==6) || (cont==10) || (cont==14)){
 output_low(Led1);
 output_low(Led2);
 output_low(Led3);
 output_low(Led4);
 if(cont==2){
  ADC_up2[0]=read_adc();
  }
 if(cont==6){
  ADC_up2[1]=read_adc();
  }
 if(cont==10){
  ADC_up2[2]=read_adc();
  }
 if(cont==14){
  ADC_up2[3]=read_adc();
  }
 set_adc_channel(5);
}
```

```
   if((cont==3) || (cont==7) || (cont==11) || (cont==15)){
   output_low(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);

    if(cont==3){
    ADC_down1[0]=read_adc();
    set_adc_channel(8);
    }
    if(cont==7){
    ADC_down1[1]=read_adc();
    set_adc_channel(8);
    }
    if(cont==11){
    ADC_down1[2]=read_adc();
    set_adc_channel(8);
    }
    if(cont==15){
    ADC_down1[3]=read_adc();
    set_adc_channel(8);
    }
   }

cont++;

if(cont>15){
s=1;
cont=0;}

set_timer0(precarga);
}

void main(void)
{
usb_cdc_init();//Configura la comunicación USB
usb_init();//Inicializa el hardware del USB
usb_wait_for_enumeration();//Espera hasta que el PicUSB sea configurado.

setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(sAN5||sAN8);//Puerta C1 y C6

int16 i,in_data[4];
int32 tiempo, t,cuenta, valores[4];
float voltaje;

while(true)
 {
  usb_task();
  usb_debug_task();

  if(usb_cdc_kbhit())
     {
      for(i=0;i<4;i++)
       {
        //La función do-while hace que sea imposible introducir letras.
        do{
        in_data[i]=usb_cdc_getc();
         }
while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51
         &&in_data[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55
```

```
          &&in_data[i]!=56&&in_data[i]!=57);

          valores[i]=(in_data[i]-48);//pasamos de ASCII a decimal
          printf(usb_cdc_putc,"%c",in_data[i]);
        }

      tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3]);
      t=tiempo/2;
      cuenta=(t*0.001)/0.0000213333333;
      precarga=(65536-cuenta);

disable_interrupts(GLOBAL);
disable_interrupts(INT_TIMER0);

setup_timer_0(RTCC_DIV_256);
set_timer0(precarga);

enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);

while(TRUE){

 if(a==1){

  printf(usb_cdc_putc,"\r\n Fotodiodo 1: \r\n");
  printf(usb_cdc_putc," H1 %4ld  L1 %4ld  \r\n", ADC_up1[0], ADC_down1[0]);
  printf(usb_cdc_putc," H2 %4ld  L2 %4ld  \r\n", ADC_up1[1], ADC_down1[1]);
  printf(usb_cdc_putc," H3 %4ld  L3 %4ld  \r\n", ADC_up1[2], ADC_down1[2]);
  printf(usb_cdc_putc," H4 %4ld  L4 %4ld  \r\n", ADC_up1[3], ADC_down1[3]);

  printf(usb_cdc_putc,"\r\n Fotodiodo 2: \r\n");
  printf(usb_cdc_putc," H1 %4ld  L1 %4ld  \r\n", ADC_up2[0], ADC_down2[0]);
  printf(usb_cdc_putc," H2 %4ld  L2 %4ld  \r\n", ADC_up2[1], ADC_down2[1]);
  printf(usb_cdc_putc," H3 %4ld  L3 %4ld  \r\n", ADC_up2[2], ADC_down2[2]);
  printf(usb_cdc_putc," H4 %4ld  L4 %4ld  \r\n", ADC_up2[3], ADC_down2[3]);

    }
   }
  }
 }
}
```

```
#include <18F14K50.h>
#device ADC=12
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>

#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int32 precarga;
int16 ADC_up1[4],ADC_down1[4],ADC_up2[4],ADC_down2[4];
int8 cont=0;
int1 s=0, a=0;

void usb_debug_task(void)
{
 static int8 last_cdc;
 int8 new_cdc;
 new_cdc=usb_cdc_connected();
 last_cdc=new_cdc;
}

#int_timer0
void isr_timer0()
{
  if(cont==0){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   if(s==1){
    ADC_down2[3]=read_adc();
    a=1;
    }
   set_adc_channel(5);
   }
   if(cont==4){
   output_low(Led1);
   output_high(Led2);
   output_low(Led3);
   output_low(Led4);
   ADC_down2[0]=read_adc();
   set_adc_channel(5);
   }
   if(cont==8){
   output_low(Led1);
   output_low(Led2);
   output_high(Led3);
   output_low(Led4);
   ADC_down2[1]=read_adc();
   set_adc_channel(5);
   }
   if(cont==12){
```

```
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_down2[2]=read_adc();
set_adc_channel(5);
}

if(cont==1){
output_high(Led1);
output_low(Led2);
output_low(Led3);
output_low(Led4);
a=0;
ADC_up1[0]=read_adc();
set_adc_channel(8);
}
if(cont==5){
output_low(Led1);
output_high(Led2);
output_low(Led3);
output_low(Led4);
ADC_up1[1]=read_adc();
set_adc_channel(8);
}
if(cont==9){
output_low(Led1);
output_low(Led2);
output_high(Led3);
output_low(Led4);
ADC_up1[2]=read_adc();
set_adc_channel(8);
}
if(cont==13){
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_up1[3]=read_adc();
set_adc_channel(8);
}

if((cont==2) || (cont==6) || (cont==10) || (cont==14)){
 output_low(Led1);
 output_low(Led2);
 output_low(Led3);
 output_low(Led4);
 if(cont==2){
  ADC_up2[0]=read_adc();
  }
 if(cont==6){
  ADC_up2[1]=read_adc();
  }
 if(cont==10){
  ADC_up2[2]=read_adc();
  }
 if(cont==14){
  ADC_up2[3]=read_adc();
  }
 set_adc_channel(5);
}
```

```
   if((cont==3) || (cont==7) || (cont==11) || (cont==15)){
   output_low(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);

    if(cont==3){
    ADC_down1[0]=read_adc();
    }
    if(cont==7){
    ADC_down1[1]=read_adc();
    }
    if(cont==11){
    ADC_down1[2]=read_adc();
    }
    if(cont==15){
    ADC_down1[3]=read_adc();
    }
    set_adc_channel(8);
   }

cont++;

if(cont>15){
s=1;
cont=0;}

set_timer0(precarga);
}

void main(void)
{
usb_cdc_init();//Configura la comunicación USB
usb_init();//Inicializa el hardware del USB
usb_wait_for_enumeration();//Espera hasta que el PicUSB sea configurado por
el host

setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(sAN5||sAN8);//Puerta C1 y C6

int16 i,in_data[4], pwm[3];
int32 tiempo, t, cuenta, valores[4], pwm1[3];
float voltaje;
long duty_PWM, porcent;

while(true)
 {
  usb_task();
  usb_debug_task();

  if(usb_cdc_kbhit())
     {
     printf(usb_cdc_putc,"Cabecera |");

//Tiempo entre parpadeos de los LEDs

      for(i=0;i<4;i++)
       {
        //La función do-while hace que sea imposible introducir letras. Sólo
números.
        do{
```

```
        in_data[i]=usb_cdc_getc();
         }
while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51&&in_data
[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55&&in_data[i]!=56&&in_d
ata[i]!=57);

         valores[i]=(in_data[i]-48);//pasamos de ASCII a decimal
         printf(usb_cdc_putc,"%c",in_data[i]);
       }

       printf(usb_cdc_putc,"us |");

//Valor del PWM

       for(i=0;i<3;i++)
       {
       do{
        pwm[i]=usb_cdc_getc();
         }
while(pwm[i]!=48&&pwm[i]!=49&&pwm[i]!=50&&pwm[i]!=51&&pwm[i]!=52&&pwm[i]!=53&
&pwm[i]!=54&&pwm[i]!=55&&pwm[i]!=56&&pwm[i]!=57);

         pwm1[i]=(pwm[i]-48);//pasamos de ASCII a decimal
         printf(usb_cdc_putc,"%c",pwm[i]);
       }

       printf(usb_cdc_putc,"%% |FinDeMensaje");

      tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3])-70;
      t=tiempo/2;
      cuenta=(t/0.08333333);
      precarga=(65536-cuenta);

      porcent=(pwm1[0]*100+pwm1[1]*10+pwm1[2]);
      duty_PWM=(4.8*porcent);


disable_interrupts(GLOBAL);
disable_interrupts(INT_TIMER0);

setup_timer_0(RTCC_DIV_1);
set_timer0(precarga);

enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);

while(TRUE){

// PWM
output_low(PIN_C5);   // Set CCP1 output low
output_drive(PIN_C5); // A few older compilers, don't correctly set the TRIS
when this is done, but current ones do.

setup_ccp1(CCP_PWM);  // Configure CCP1 as a PWM

 //   The cycle time will be (1/clock)*4*t2div*(period+1)
 //   In this program clock=48000000
 //     (4/48000000)*1*X = 100 kHz

setup_timer_2(T2_DIV_BY_1, 119, 1);  // 100kHz
```

```
set_pwm1_duty(duty_PWM); // % duty cycle on pin C5


 if(a==1){
  printf(usb_cdc_putc,"\r\n Cabecera \r\n");

 printf(usb_cdc_putc,"\r\n Fotodiodo 1: \r\n");
  printf(usb_cdc_putc," H1 %4ld  L1 %4ld  \r\n", ADC_up1[0], ADC_down1[0]);
  printf(usb_cdc_putc," H2 %4ld  L2 %4ld  \r\n", ADC_up1[1], ADC_down1[1]);
  printf(usb_cdc_putc," H3 %4ld  L3 %4ld  \r\n", ADC_up1[2], ADC_down1[2]);
  printf(usb_cdc_putc," H4 %4ld  L4 %4ld  \r\n", ADC_up1[3], ADC_down1[3]);

   printf(usb_cdc_putc,"\r\n Fotodiodo 2: \r\n");
  printf(usb_cdc_putc," H1 %4ld  L1 %4ld  \r\n", ADC_up2[0], ADC_down2[0]);
  printf(usb_cdc_putc," H2 %4ld  L2 %4ld  \r\n", ADC_up2[1], ADC_down2[1]);
  printf(usb_cdc_putc," H3 %4ld  L3 %4ld  \r\n", ADC_up2[2], ADC_down2[2]);
  printf(usb_cdc_putc," H4 %4ld  L4 %4ld  \r\n", ADC_up2[3], ADC_down2[3]);

   printf(usb_cdc_putc,"\r\n FinDeMensaje \r\n");

  delay_ms(tiempo);//para que solo lo haga una vez
    }
   }
  }
 }
}
```

Transmisión en binario de los datos.

```
#include <18F14K50.h>
#device ADC=12
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>

#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int16 Cabecera=0xf0ff;
int16 FinDeMensaje=0xf000;

int32 precarga;
int16 ADC_up1[4],ADC_down1[4],ADC_up2[4],ADC_down2[4];
int8 cont=0, msb_cabecera, lsb_cabecera, msb_fdm, lsb_fdm;
int8  msb_up10,  lsb_up10,  msb_up11,  lsb_up11,  msb_up12,  lsb_up12,  msb_up13,
lsb_up13;
int8  msb_up20,  lsb_up20,  msb_up21,  lsb_up21,  msb_up22,  lsb_up22,  msb_up23,
lsb_up23;
int8 msb_down10, lsb_down10, msb_down11, lsb_down11, msb_down12, lsb_down12,
msb_down13, lsb_down13;
int8 msb_down20, lsb_down20, msb_down21, lsb_down21, msb_down22, lsb_down22,
msb_down23, lsb_down23;

int1 s=0, a=0;

void usb_debug_task(void)
{
 static int8 last_cdc;
 int8 new_cdc;

 new_cdc=usb_cdc_connected();
 last_cdc=new_cdc;
}

#int_timer0
void isr_timer0()
{
  if(cont==0){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   if(s==1){
    ADC_down2[3]=read_adc();
    msb_down23=make8(ADC_down2[3],1);
    lsb_down23=make8(ADC_down2[3],0);
    a=1;
    }
   set_adc_channel(5);
   }
   if(cont==4){
   output_low(Led1);
```

```
output_high(Led2);
output_low(Led3);
output_low(Led4);
ADC_down2[0]=read_adc();
msb_down20=make8(ADC_down2[0],1);
lsb_down20=make8(ADC_down2[0],0);
set_adc_channel(5);
}
if(cont==8){
output_low(Led1);
output_low(Led2);
output_high(Led3);
output_low(Led4);
ADC_down2[1]=read_adc();
msb_down21=make8(ADC_down2[1],1);
lsb_down21=make8(ADC_down2[1],0);
set_adc_channel(5);
}
if(cont==12){
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_down2[2]=read_adc();
msb_down22=make8(ADC_down2[2],1);
lsb_down22=make8(ADC_down2[2],0);
set_adc_channel(5);
}

if(cont==1){
output_high(Led1);
output_low(Led2);
output_low(Led3);
output_low(Led4);
a=0;
ADC_up1[0]=read_adc();
msb_up10=make8(ADC_up1[0],1);
lsb_up10=make8(ADC_up1[0],0);
set_adc_channel(8);
}
if(cont==5){
output_low(Led1);
output_high(Led2);
output_low(Led3);
output_low(Led4);
ADC_up1[1]=read_adc();
msb_up11=make8(ADC_up1[1],1);
lsb_up11=make8(ADC_up1[1],0);
set_adc_channel(8);
}
if(cont==9){
output_low(Led1);
output_low(Led2);
output_high(Led3);
output_low(Led4);
ADC_up1[2]=read_adc();
msb_up12=make8(ADC_up1[2],1);
lsb_up12=make8(ADC_up1[2],0);
set_adc_channel(8);
}
if(cont==13){
```

```
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_up1[3]=read_adc();
msb_up13=make8(ADC_up1[3],1);
lsb_up13=make8(ADC_up1[3],0);
set_adc_channel(8);
}

if((cont==2) || (cont==6) || (cont==10) || (cont==14)){
 output_low(Led1);
 output_low(Led2);
 output_low(Led3);
 output_low(Led4);
 if(cont==2){
  ADC_up2[0]=read_adc();
  msb_up20=make8(ADC_up2[0],1);
  lsb_up20=make8(ADC_up2[0],0);
  }
 if(cont==6){
  ADC_up2[1]=read_adc();
  msb_up21=make8(ADC_up2[1],1);
  lsb_up21=make8(ADC_up2[1],0);
  }
 if(cont==10){
  ADC_up2[2]=read_adc();
  msb_up22=make8(ADC_up2[2],1);
  lsb_up22=make8(ADC_up2[2],0);
  }
 if(cont==14){
  ADC_up2[3]=read_adc();
  msb_up23=make8(ADC_up2[3],1);
  lsb_up23=make8(ADC_up2[3],0);
  }
 set_adc_channel(5);
}
if((cont==3) || (cont==7) || (cont==11) || (cont==15)){
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_low(Led4);

 if(cont==3){
 ADC_down1[0]=read_adc();
 msb_down10=make8(ADC_down1[0],1);
 lsb_down10=make8(ADC_down1[0],0);
 }
 if(cont==7){
 ADC_down1[1]=read_adc();
 msb_down11=make8(ADC_down1[1],1);
 lsb_down11=make8(ADC_down1[1],0);
 }
 if(cont==11){
 ADC_down1[2]=read_adc();
 msb_down12=make8(ADC_down1[2],1);
 lsb_down12=make8(ADC_down1[2],0);
 }
 if(cont==15){
 ADC_down1[3]=read_adc();
 msb_down13=make8(ADC_down1[3],1);
```

```
    lsb_down13=make8(ADC_down1[3],0);
    }
    set_adc_channel(8);
   }

cont++;

if(cont>15){
s=1;
cont=0;}

set_timer0(precarga);
}

void main(void)
{
usb_cdc_init();//Configura la comunicación USB
usb_init();//Inicializa el hardware del USB
usb_wait_for_enumeration();//Espera hasta que el PicUSB sea configurado por
el host

setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(sAN5||sAN8);//Puerta C1 y C6

int8 i, in_data[4], valores[4], pwm1[3], pwm[3];
int16 tiempo, t;
int32 cuenta;
float voltaje;
long duty_PWM, porcent;

msb_cabecera=make8(Cabecera,1);
lsb_cabecera=make8(Cabecera,0);
msb_fdm=make8(FinDeMensaje,1);
lsb_fdm=make8(FinDeMensaje,0);

while(true)
 {
  usb_task();
  usb_debug_task();

  if(usb_cdc_kbhit())
     {
      printf(usb_cdc_putc,"%c%c", msb_cabecera, lsb_cabecera); //Cabecera

//Tiempo entre parpadeos de los LEDs

      for(i=0;i<4;i++)
       {
        //La función do-while hace que sea imposible introducir letras. Sólo
números.
        do{
        in_data[i]=usb_cdc_getc();
         }
while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51&&in_data
[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55&&in_data[i]!=56&&in_d
ata[i]!=57);

         valores[i]=(in_data[i]-48); //pasamos de ASCII a decimal
       }
       tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3])-70;
       printf(usb_cdc_putc,"%c",tiempo);
```

```
//Valor del PWM
       for(i=0;i<3;i++)
       {
       do{
        pwm[i]=usb_cdc_getc();
         }
while(pwm[i]!=48&&pwm[i]!=49&&pwm[i]!=50&&pwm[i]!=51&&pwm[i]!=52&&pwm[i]!=53&
&pwm[i]!=54&&pwm[i]!=55&&pwm[i]!=56&&pwm[i]!=57);

         pwm1[i]=(pwm[i]-48);//pasamos de ASCII a decimal
       }
      porcent=(pwm1[0]*100+pwm1[1]*10+pwm1[2]);
      printf(usb_cdc_putc,"%c",porcent);

      printf(usb_cdc_putc,"%c%c",msb_fdm, lsb_fdm); //Fin de mensaje

      t=tiempo/2;
      cuenta=(t/0.08333333);
      precarga=(65536-cuenta);
      duty_PWM=(4.8*porcent);

disable_interrupts(GLOBAL);
disable_interrupts(INT_TIMER0);
setup_timer_0(RTCC_DIV_1);
set_timer0(precarga);
enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);

// PWM
output_low(PIN_C5);   // Set CCP1 output low
output_drive(PIN_C5); // A few older compilers, don't correctly set the TRIS
when this is done, but current ones do.
setup_ccp1(CCP_PWM);  // Configure CCP1 as a PWM
setup_timer_2(T2_DIV_BY_1, 119, 1);  // 100kHz
set_pwm1_duty(duty_PWM);  // % duty cycle on pin C5

while(TRUE){

 if(a==1){
  printf(usb_cdc_putc,"%c%c",msb_cabecera, lsb_cabecera);
  printf(usb_cdc_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",  msb_up10,  lsb_up10,
msb_down10, lsb_down10, msb_up11, lsb_up11, msb_down11, lsb_down11, msb_up12,
lsb_up12,    msb_down12,    lsb_down12,    msb_up13,    lsb_up13,    msb_down13,
lsb_down13);
  printf(usb_cdc_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",  msb_up20,  lsb_up20,
msb_down20, lsb_down20, msb_up21, lsb_up21, msb_down21, lsb_down21, msb_up22,
lsb_up22,    msb_down22,    lsb_down22,    msb_up23,    lsb_up23,    msb_down23,
lsb_down23);
  printf(usb_cdc_putc,"%c%c",msb_fdm, lsb_fdm);
    }
   }
  }
 }
}
```

Valores predeterminados de 125us y 20% de PWM.

```
#include <18F14K50.h>
#device ADC=12
#include <usb_cdc.h>
#include <pic18_usb.h>
#include <usb.c>

#fuses
HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)

#define  Led1  PIN_B4
#define  Led2  PIN_B5
#define  Led3  PIN_B6
#define  Led4  PIN_B7

int16 Cabecera=0xf0ff;
int16 FinDeMensaje=0xf000;

int32 precarga=64035;
int16 ADC_up1[4],ADC_down1[4],ADC_up2[4],ADC_down2[4];
unsigned int16 cont1;
int8 cont=0, msb_cabecera, lsb_cabecera, msb_fdm, lsb_fdm;
int8  msb_up10,  lsb_up10,  msb_up11,  lsb_up11,  msb_up12,  lsb_up12,  msb_up13,
lsb_up13;
int8  msb_up20,  lsb_up20,  msb_up21,  lsb_up21,  msb_up22,  lsb_up22,  msb_up23,
lsb_up23;
int8  msb_down10,  lsb_down10,  msb_down11,  lsb_down11,  msb_down12,  lsb_down12,
msb_down13, lsb_down13;
int8  msb_down20,  lsb_down20,  msb_down21,  lsb_down21,  msb_down22,  lsb_down22,
msb_down23, lsb_down23;
int1 s=0, a=0;

void usb_debug_task(void) //Esta función depura las tareas del USB.
{
 static int8 last_cdc;
 int8 new_cdc;

 new_cdc=usb_cdc_connected();
 last_cdc=new_cdc;
}

#int_timer0
void isr_timer0() // Secuencia de encendido y apagado de los leds.
{
  if(cont==0){
   output_high(Led1);
   output_low(Led2);
   output_low(Led3);
   output_low(Led4);
   if(s==1){
    ADC_down2[3]=read_adc();
    msb_down23=make8(ADC_down2[3],1);
    lsb_down23=make8(ADC_down2[3],0);
    a=1;
    }
   set_adc_channel(5);
   }
   if(cont==4){
   output_low(Led1);
```

```
output_high(Led2);
output_low(Led3);
output_low(Led4);
ADC_down2[0]=read_adc();
msb_down20=make8(ADC_down2[0],1);
lsb_down20=make8(ADC_down2[0],0);
set_adc_channel(5);
}
if(cont==8){
output_low(Led1);
output_low(Led2);
output_high(Led3);
output_low(Led4);
ADC_down2[1]=read_adc();
msb_down21=make8(ADC_down2[1],1);
lsb_down21=make8(ADC_down2[1],0);
set_adc_channel(5);
}
if(cont==12){
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_down2[2]=read_adc();
msb_down22=make8(ADC_down2[2],1);
lsb_down22=make8(ADC_down2[2],0);
set_adc_channel(5);
}

if(cont==1){
output_high(Led1);
output_low(Led2);
output_low(Led3);
output_low(Led4);
a=0;
ADC_up1[0]=read_adc();
msb_up10=make8(ADC_up1[0],1);
lsb_up10=make8(ADC_up1[0],0);
set_adc_channel(8);
}
if(cont==5){
output_low(Led1);
output_high(Led2);
output_low(Led3);
output_low(Led4);
ADC_up1[1]=read_adc();
msb_up11=make8(ADC_up1[1],1);
lsb_up11=make8(ADC_up1[1],0);
set_adc_channel(8);
}
if(cont==9){
output_low(Led1);
output_low(Led2);
output_high(Led3);
output_low(Led4);
ADC_up1[2]=read_adc();
msb_up12=make8(ADC_up1[2],1);
lsb_up12=make8(ADC_up1[2],0);
set_adc_channel(8);
}
if(cont==13){
```

```
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_high(Led4);
ADC_up1[3]=read_adc();
msb_up13=make8(ADC_up1[3],1);
lsb_up13=make8(ADC_up1[3],0);
set_adc_channel(8);
}

if((cont==2) || (cont==6) || (cont==10) || (cont==14)){
 output_low(Led1);
 output_low(Led2);
 output_low(Led3);
 output_low(Led4);
 if(cont==2){
  ADC_up2[0]=read_adc();
  msb_up20=make8(ADC_up2[0],1);
  lsb_up20=make8(ADC_up2[0],0);
  }
 if(cont==6){
  ADC_up2[1]=read_adc();
  msb_up21=make8(ADC_up2[1],1);
  lsb_up21=make8(ADC_up2[1],0);
  }
 if(cont==10){
  ADC_up2[2]=read_adc();
  msb_up22=make8(ADC_up2[2],1);
  lsb_up22=make8(ADC_up2[2],0);
  }
 if(cont==14){
  ADC_up2[3]=read_adc();
  msb_up23=make8(ADC_up2[3],1);
  lsb_up23=make8(ADC_up2[3],0);
  }
 set_adc_channel(5);
}
if((cont==3) || (cont==7) || (cont==11) || (cont==15)){
output_low(Led1);
output_low(Led2);
output_low(Led3);
output_low(Led4);

 if(cont==3){
 ADC_down1[0]=read_adc();
 msb_down10=make8(ADC_down1[0],1);
 lsb_down10=make8(ADC_down1[0],0);
 }
 if(cont==7){
 ADC_down1[1]=read_adc();
 msb_down11=make8(ADC_down1[1],1);
 lsb_down11=make8(ADC_down1[1],0);
 }
 if(cont==11){
 ADC_down1[2]=read_adc();
 msb_down12=make8(ADC_down1[2],1);
 lsb_down12=make8(ADC_down1[2],0);
 }
 if(cont==15){
 ADC_down1[3]=read_adc();
 msb_down13=make8(ADC_down1[3],1);
```

```
    lsb_down13=make8(ADC_down1[3],0);
    }
    set_adc_channel(8);
   }

cont++;

if(cont>15){
s=1;
cont=0;}

set_timer0(precarga);
}

void main(void)
{
usb_cdc_init();//Configura la comunicación USB
usb_init();//Inicializa el hardware del USB
usb_wait_for_enumeration();//Espera hasta que el PicUSB sea configurado por
el host

setup_adc(ADC_CLOCK_INTERNAL); //Entradas analógicas
setup_adc_ports(sAN5||sAN8); //Puerta C1 y C6

int8 i, in_data[4], valores[4], pwm1[3], pwm[3];
int16 tiempo, t;
int32 cuenta;
long duty_PWM=96, porcent;

msb_cabecera=make8(Cabecera,1);
lsb_cabecera=make8(Cabecera,0);
msb_fdm=make8(FinDeMensaje,1);
lsb_fdm=make8(FinDeMensaje,0);

//PREDEFINIDO 125us Y 20% DE PWM

 // Señal PWM
output_low(PIN_C5);
output_drive(PIN_C5);
setup_ccp1(CCP_PWM);
setup_timer_2(T2_DIV_BY_1, 119, 1);
set_pwm1_duty(duty_PWM);

// Interrupcion del timer0.
disable_interrupts(GLOBAL);
disable_interrupts(INT_TIMER0);
setup_timer_0(RTCC_DIV_1);
set_timer0(precarga);
enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);

while(true){ //Introducir nuevos valores.
usb_task();
usb_debug_task();

  if(usb_cdc_kbhit())
     {
      printf(usb_cdc_putc,"%c%c", msb_cabecera, lsb_cabecera); //Cabecera

      //Tiempo entre parpadeos de los LEDs
      for(i=0;i<4;i++)
```

```
       {
        do{
        in_data[i]=usb_cdc_getc();
         }
while(in_data[i]!=48&&in_data[i]!=49&&in_data[i]!=50&&in_data[i]!=51
         &&in_data[i]!=52&&in_data[i]!=53&&in_data[i]!=54&&in_data[i]!=55
         &&in_data[i]!=56&&in_data[i]!=57);

         valores[i]=(in_data[i]-48); //pasamos de ASCII a decimal

       }
       tiempo=(valores[0]*1000+valores[1]*100+valores[2]*10+valores[3])-70;
       printf(usb_cdc_putc,"%c",tiempo);

       //Valor del PWM
       for(i=0;i<3;i++)
       {
       do{
        pwm[i]=usb_cdc_getc();
         } while(pwm[i]!=48&&pwm[i]!=49&&pwm[i]!=50&&pwm[i]!=51&&pwm[i]!=52
         &&pwm[i]!=53&&pwm[i]!=54&&pwm[i]!=55&&pwm[i]!=56&&pwm[i]!=57);

         pwm1[i]=(pwm[i]-48);//pasamos de ASCII a decimal

       }
      porcent=(pwm1[0]*100+pwm1[1]*10+pwm1[2]);
      printf(usb_cdc_putc,"%c",porcent);

      printf(usb_cdc_putc,"%c%c",msb_fdm, lsb_fdm); //Fin de mensaje

      //Calculos
      t=tiempo/2;
      cuenta=(t/0.08333333);
      precarga=(65536-cuenta);
      duty_PWM=(4.8*porcent);

disable_interrupts(GLOBAL);
disable_interrupts(INT_TIMER0);
setup_timer_0(RTCC_DIV_1);
set_timer0(precarga);
enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER0);

output_low(PIN_C5);
output_drive(PIN_C5);
setup_ccp1(CCP_PWM);
setup_timer_2(T2_DIV_BY_1, 119, 1);
set_pwm1_duty(duty_PWM);
     }

while(usb_cdc_kbhit()==0){

 if(a==1){
  cont1++;

  if(cont1==65535){
  cont1=0;
  }
  printf(usb_cdc_putc,"%c%c",msb_cabecera, lsb_cabecera);
  printf(usb_cdc_putc, "%c",cont1);
```

```
  printf(usb_cdc_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c", msb_up10, lsb_up10,
msb_down10, lsb_down10, msb_up11, lsb_up11, msb_down11, lsb_down11, msb_up12,
lsb_up12,   msb_down12,   lsb_down12,   msb_up13,   lsb_up13,   msb_down13,
lsb_down13);
  printf(usb_cdc_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c", msb_up20, lsb_up20,
msb_down20, lsb_down20, msb_up21, lsb_up21, msb_down21, lsb_down21, msb_up22,
lsb_up22,   msb_down22,   lsb_down22,   msb_up23,   lsb_up23,   msb_down23,
lsb_down23);
  printf(usb_cdc_putc,"%c%c",msb_fdm, lsb_fdm);
   }
  }
 }
}
```

Anexo 02. Presupuesto

En el presupuesto del proyecto, además de incluir el precio de todos los materiales necesarios para implementarlo, también y sobre todo hay que tener en cuenta el coste y horas de la mano de obra necesaria para llevar a cabo cada parte del proyecto.

El presupuesto es el plan financiero estimado para un proyecto, para el cual se requiere administrar fondos. Este documento debe incluir los gastos en los que se prevé incurrir en un período de tiempo determinado, como también el ingreso que se generará durante el transcurso del proyecto. Un presupuesto bien concebido puede contribuir en gran medida a la comprensión del proyecto.

Según las normas, el presupuesto puede consistir en una declaración de una sola hoja en la que consten los gastos estimados. También podría reflejarse en una hoja de cálculo completa que incluya los ingresos y las donaciones proyectadas, así como una descripción detallada que explique los diversos rubros de gastos e ingresos.

El presupuesto de este proyecto no será muy grande ya que básicamente consiste en la programación de un microcontrolador y de diferentes comprobaciones para la adaptación de señales para conseguir el funcionamiento de nuestro seguimiento ocular. Pero aunque solo se base en esto, el estudio financiero constará de los esfuerzos realizados y el tiempo invertido por la mano de obra.

Coste de la mano de obra:

Tareas implicadas	Duración (dias)	Duración (Horas)	Precio(€)
Diseño de circuitos	5	30	450
Programación del microcontrolador	15	90	1350
Montajes y pruebas de laboratorio	26	130	1950
Creación de PCBs con Orcad	2	14	210
Revelado y soldado de placas	2	10	150
Redacción del informe	20	120	1800
		Total:	6000

Coste hora ingeniero	15
Número de horas del proyecto	400
Total coste mano de obra	6000
21% IVA	1260
Total coste mano de obra con IVA	7260

Coste de los componentes:

Componente	Coste(€/Ud.)	Cantidad	Precio(€)
Transistor NTR4503	0,92	5	4,59
Transistor Canal N	0,38	1	0,38
Amplificador TL072	0,72	1	0,72
Microcontrolador PIC18F14K50-I	2,39	1	2,39
Fotodiodos OPT101	4,04	2	8,08
Diodos LEDs	0,753	10	7,53
		Total:	23,69

Anexo 03. Planificación

El diagrama de GANTT es una herramienta que permite modelar la planificación de las tareas necesarias para la realización del proyecto. Éste es imprescindible en cualquier proyecto que se desee presentar.

Se trata de una representación gráfica del progreso del proyecto, donde cada tarea es representada por una línea, y las columnas representan los días, semanas, o meses del programa, dependiendo de la duración del proyecto. El tiempo estimado para cada tarea se muestra a través de una barra horizontal cuyo extremo izquierdo determina la fecha de inicio prevista y el extremo derecho determina la fecha de finalización estimada. Las tareas se pueden colocar en cadenas secuenciales o se pueden realizar simultáneamente.

Éste diagrama se llevará a cabo mediante el programa Projectlibre. Pues ésta es la principal alternativa de código abierto a Microsoft Project.

Si las tareas son secuenciales, las prioridades se pueden confeccionar utilizando una flecha que desciende de las tareas más importantes hacia las tareas menos importantes. La tarea menos importante no puede llevarse a cabo hasta que no se haya completado la más importante.

A medida que progresa una tarea, se completa proporcionalmente la barra que la representa hasta llegar al grado de finalización. Así, es posible obtener una visión general del progreso del proyecto rastreando una línea vertical a través de las tareas en el nivel de la fecha actual. Las tareas ya finalizadas se colocan a la izquierda de esta línea; las tareas que aún no se han iniciado se colocan a la derecha, mientras que las tareas que se están llevando a cabo atraviesan la línea.

Adicionalmente, es posible que los eventos más importantes, que no sean las tareas mismas, se muestren en la planificación como puntos de conexión del proyecto: estos se denominan acontecimientos.

Los acontecimientos permiten que el proyecto se realice en fases claramente identificables, evitando que se prolongue la finalización del mismo. Un acontecimiento podría ser la producción de un documento, la realización de una reunión o el producto final de un proyecto. Los acontecimientos son tareas de duración cero, representadas en el diagrama por un símbolo específico, frecuentemente un triángulo invertido o un diamante.

En este proyecto se cuenta con el siguiente desglose de tareas para el diagrama:

	WBS	Nombre	Duracion	Inicio	Terminado
1	**1**	**Introducción**	**3 days**	**28/04/16 9:00**	**2/05/16 16:00**
2	1.1	Identificación de los requisitos del usuario	1 day	28/04/16 9:00	28/04/16 18:00
3	1.2	Identificación de los requisitos de rendimiento y de HW/SW	2 days	28/04/16 18:00	2/05/16 16:00
4	**2**	**Diseño del Proyecto**	**80,5 days**	**2/05/16 16:00**	**5/08/16 16:00**
5	2.1	Obtención de la arquitectura de la aplicación	3 days	2/05/16 16:00	5/05/16 10:00
6	**2.2**	**Diseño de los circuitos eléctricos**	**11 days**	**5/05/16 10:00**	**18/05/16 16:00**
7	2.2.1	Circuitos de potencia	4 days	13/05/16 10:00	18/05/16 16:00
8	2.2.2	Esquematico del circuito	7 days	5/05/16 10:00	13/05/16 10:00
9	**2.3**	**Programacion del microcontrolador**	**17 days**	**18/07/16 9:00**	**5/08/16 16:00**
10	2.3.1	Simulación y testeo	4 days	22/07/16 16:00	27/07/16 19:00
11	2.3.2	Pruebas y ensayos	11,167 days	23/07/16 9:00	5/08/16 16:00
12	2.3.3	Programacion inicial	5,333 days	18/07/16 9:00	22/07/16 16:00
13	**2.4**	**Montajes en el laboratorio**	**26,833 days**	**18/05/16 16:00**	**15/06/16 17:00**
14	2.4.1	Pruebas iniciales	5 days	18/05/16 16:00	24/05/16 18:00
15	2.4.2	Implementación de los diferentes circuitos	20 days	26/05/16 8:00	15/06/16 17:00
16	**2.5**	**Creación de las placas PCB**	**28,667 days**	**20/06/16 9:00**	**22/07/16 16:00**
17	2.5.1	Creación librería personal	7 days	20/06/16 9:00	27/06/16 19:00
18	2.5.2	Montaje con Orcad Layout	13,5 days	28/06/16 9:00	13/07/16 16:00
19	2.5.3	Revelado de las placas y soldado.	8,167 days	13/07/16 16:00	22/07/16 16:00
20	**2.2**	**Diseño de los circuitos eléctricos**	**11 days**	**5/05/16 10:00**	**18/05/16 16:00**
21	2.2.1	Circuitos de potencia	4 days	13/05/16 10:00	18/05/16 16:00
22	2.2.2	Esquematico del circuito	7 days	5/05/16 10:00	13/05/16 10:00
23	**3**	**Documentación**	**67,667 days**	**6/06/16 9:00**	**24/08/16 19:00**
24	3.1	Esquema de Conexiones	2,667 days	1/08/16 9:00	3/08/16 11:00
25	3.2	Planos Finales	2,667 days	22/08/16 9:00	24/08/16 11:00
26	3.3	Código de Programación	1,333 days	28/07/16 9:00	29/07/16 10:00
27	3.4	Análisis de Mercado	9 days	6/06/16 9:00	15/06/16 17:00
28	3.5	Estudio de costes	2,333 days	22/08/16 9:00	23/08/16 19:00
29	3.6	Redacción del informe del proyecto	38,333 days	11/07/16 10:00	24/08/16 19:00
30	**4**	**Entrega del Proyecto**	**2,667 days**	**25/08/16 9:00**	**29/08/16 11:00**
31	4.1	Imprimir Documentación	1,333 days	25/08/16 9:00	26/08/16 10:00
32	4.2	Presentación	1,333 days	26/08/16 10:00	29/08/16 11:00

Figura 70: Desglose de tareas del diagrama de Gantt.

En la tabla anterior se puede observar los pasos a seguir para desarrollar el proyecto completamente y de manera detallada. Y se observa en las imágenes siguientes el diagrama del proyecto.

WBS	Nombre	Duracion
1	**Introducción**	**3 days**
1.1	Identificación de los requisitos del usuario	1 day
1.2	Identificación de los requisitos de rendimie...	2 days
2	**Diseño del Proyecto**	**80,5 days**
2.1	Obtención de la arquitectura de la aplicación	3 days
2.2	**Diseño de los circuitos eléctricos**	**11 days**
2.2.1	Circuitos de potencia	4 days
2.2.2	Esquematico del circuito	7 days
2.3	**Programacion del microcontrolador**	**17 days**
2.4	**Montajes en el laboratorio**	**26,833 days**
2.4.1	Pruebas iniciales	5 days
2.4.2	Implementación de los diferentes circuitos	20 days
2.5	**Creación de las placas PCB**	**28,667 days**
3	**Documentación**	**67,667 days**
3.1	Esquema de Conexiones	2,667 days
3.2	Planos Finales	2,667 days
3.3	Código de Programación	1,333 days
3.4	Análisis de Mercado	9 days
3.5	Estudio de costes	2,333 days
3.6	Redacción del informe del proyecto	38,333 days
4	**Entrega del Proyecto**	**2,667 days**

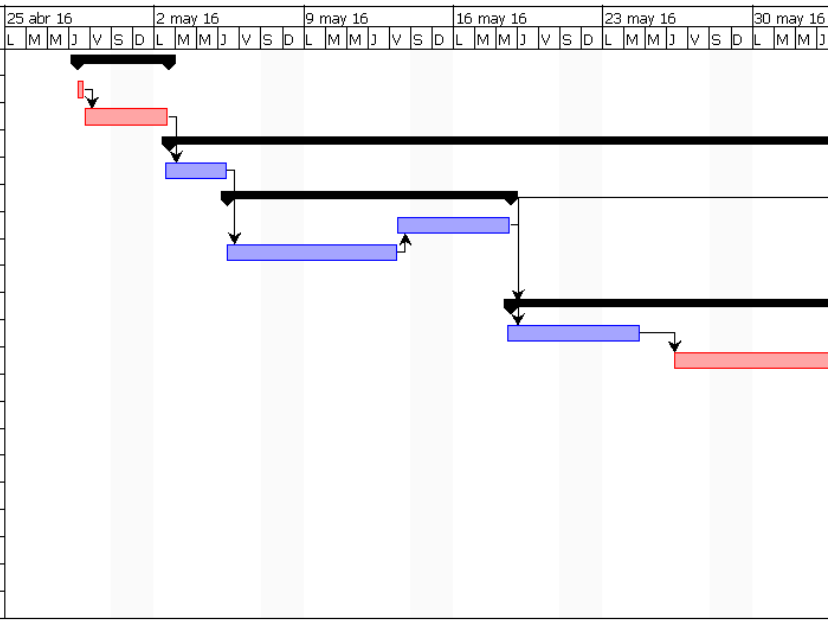


Figura 71: Diagrama de Gantt del proyecto (I)

WBS	Nombre	Duracion
1	**Introducción**	3 days
2	**Diseño del Proyecto**	80,5 days
2.1	Obtención de la arquitectura de la aplicación	3 days
2.2	**Diseño de los circuitos eléctricos**	11 days
2.3	**Programacion del microcontrolador**	17 days
2.3.1	Simulación y testeo	4 days
2.3.2	Pruebas y ensayos	11,167 days
2.3.3	Programacion inicial	5,333 days
2.4	**Montajes en el laboratorio**	26,833 days
2.4.1	Pruebas iniciales	5 days
2.4.2	Implementación de los diferentes circuitos	20 days
2.5	**Creación de las placas PCB**	28,667 days
2.5.1	Creación librería personal	7 days
2.5.2	Montaje con Orcad Layout	13,5 days
2.5.3	Revelado de las placas y soldado.	8,167 days
3	**Documentación**	67,667 days
3.1	Esquema de Conexiones	2,667 days
3.2	Planos Finales	2,667 days
3.3	Código de Programación	1,333 days
3.4	Análisis de Mercado	9 days
3.5	Estudio de costes	2,333 days
3.6	Redacción del informe del proyecto	38,333 days
4	**Entrega del Proyecto**	2,667 days

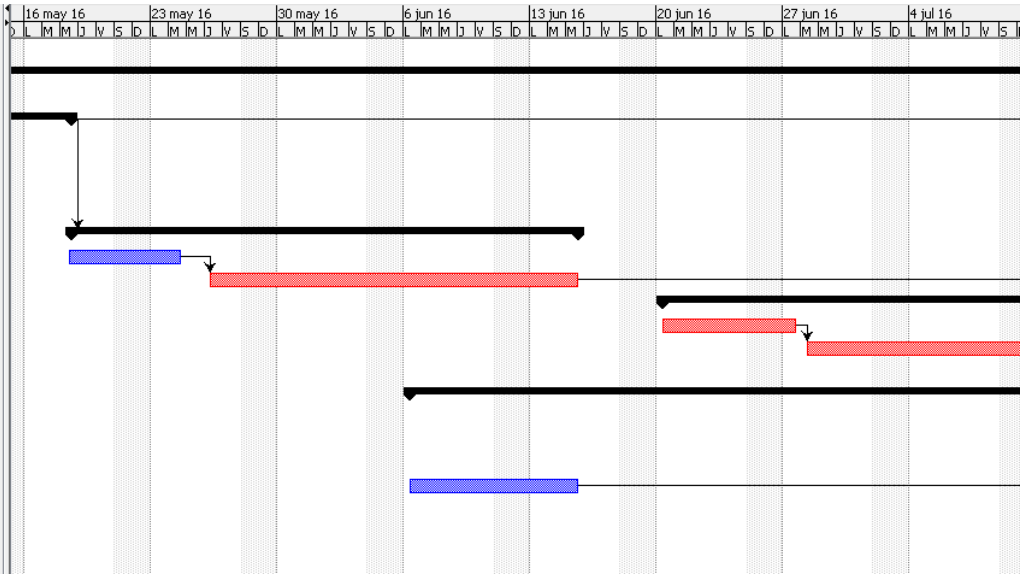


Figura 72: Diagrama de Gantt del proyecto (II)

WBS	Nombre	Duracion
1	**Introducción**	3 days
2	**Diseño del Proyecto**	80,5 days
2.1	Obtención de la arquitectura de la aplicación	3 days
2.2	**Diseño de los circuitos eléctricos**	11 days
2.3	**Programacion del microcontrolador**	17 days
2.3.1	Simulación y testeo	4 days
2.3.2	Pruebas y ensayos	11,167 days
2.3.3	Programacion inicial	5,333 days
2.4	**Montajes en el laboratorio**	26,833 days
2.4.1	Pruebas iniciales	5 days
2.4.2	Implementación de los diferentes circuitos	20 days
2.5	**Creación de las placas PCB**	28,667 days
2.5.1	Creación librería personal	7 days
2.5.2	Montaje con Orcad Layout	13,5 days
2.5.3	Revelado de las placas y soldado.	8,167 days
3	**Documentación**	67,667 days
3.1	Esquema de Conexiones	2,667 days
3.2	Planos Finales	2,667 days
3.3	Código de Programación	1,333 days
3.4	Análisis de Mercado	9 days
3.5	Estudio de costes	2,333 days
3.6	Redacción del informe del proyecto	38,333 days
4	**Entrega del Proyecto**	2,667 days
4.1	Imprimir Documentación	1,333 days
4.2	Presentación	1,333 days

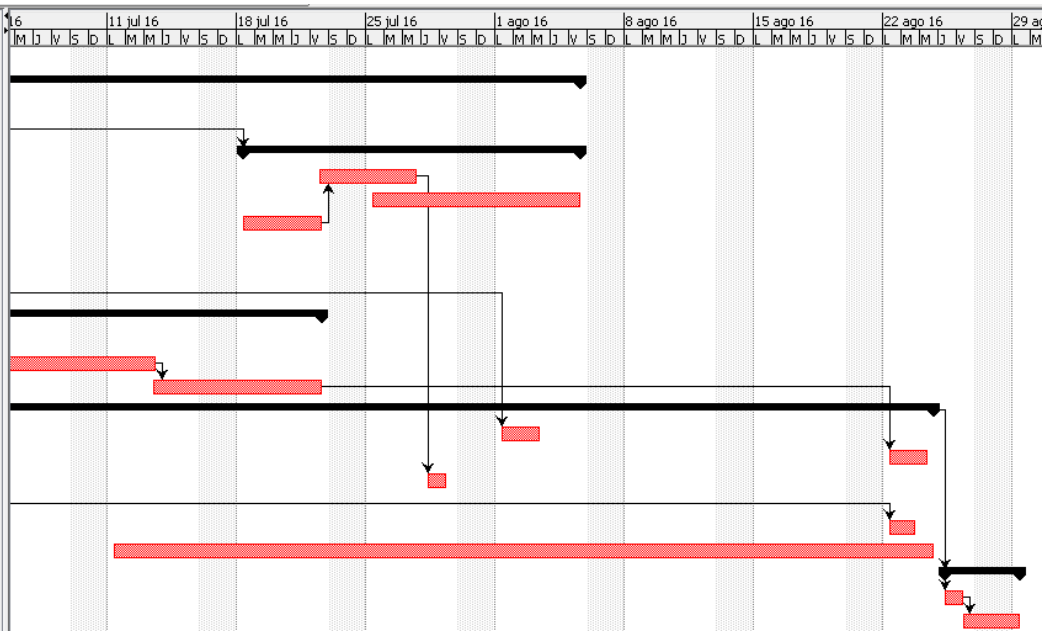


Figura 73: Diagrama de Gantt del proyecto (III).

Anexo 04. Hojas de datos.

Para simplificar el trabajo se incluyen en el proyecto las hojas de datos o direcciones directas hacia las mismas (debido a que son demasiado extensas).

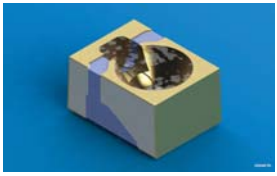
I. Microcontrolador PIC18F14K50 de Microchips.
   Pinche aquí.
II. Amplificador TL072CP
   Pinche aquí.

A continuación se muestran en el siguiente orden los datasheets de los siguientes componentes:

III. Diodo LED SFH 4651
IV. Fotodiodo mololítico con amplificador de transimpedancia.
V. Transistor NTR4503N-D
VI. Módulo Entrenador basado en el microcontrolador PIC 18F14K50 18F14K50Trainer.

Narrow beam LED in MIDLED package (850 nm)
Engwinklige LED im MIDLED-Gehäuse (850 nm)
Version 1.2

SFH 4651



Features:
- High Power Infrared LED (40 mW)
- Short switching times
- Narrow halfangle (± 10°)
- Low profile component
- Taping as Toplooker
- Also available as Sidelooker (SFH4656)

Besondere Merkmale:
- Infrarot LED mit hoher Ausgangsleistung (40 mW)
- Kurze Schaltzeiten
- Enger Abstrahlwinkel (± 10°)
- Geringe Bauhöhe
- Gurtung als Toplooker
- Auch als Sidelooker erhältlich (SFH4656)

Applications
- Infrared Illumination for cameras
- IR data transmission
- Remote control
- Automotive sensors

Anwendungen
- Infrarotbeleuchtung für Kameras
- IR Datenübertragung
- Gerätefernsteuerung
- Sensorik in der Automobiltechnik

Notes

Depending on the mode of operation, these devices emit highly concentrated non visible infrared light which can be hazardous to the human eye. Products which incorporate these devices have to follow the safety precautions given in IEC 60825-1 and IEC 62471.

Hinweise

Je nach Betriebsart emittieren diese Bauteile hochkonzentrierte, nicht sichtbare Infrarot-Strahlung, die gefährlich für das menschliche Auge sein kann. Produkte, die diese Bauteile enthalten, müssen gemäß den Sicherheitsrichtlinien der IEC-Normen 60825-1 und 62471 behandelt werden.

OSRAM Opto Semiconductors

Ordering Information
Bestellinformation

Type: Typ:	Radiant Intensity Strahlstärke I_F = 70 mA, t_p = 20 ms I_e [mW/sr]	Ordering Code Bestellnummer
SFH 4651	60 (≥ 25)	Q65110A8396

Note: Measured at a solid angle of Ω = 0.01 sr

Anm.: Gemessen bei einem Raumwinkel Ω = 0.01 sr

Maximum Ratings (T_A = 25 °C)
Grenzwerte

Parameter Bezeichnung	Symbol Symbol	Values Werte	Unit Einheit
Operation and storage temperature range Betriebs- und Lagertemperatur	T_{op}; T_{stg}	-40 ... 100	°C
Reverse voltage Sperrspannung	V_R	5	V
Forward current Durchlassstrom	I_F	70	mA
Surge current Stoßstrom (t_p = 20 µs, D = 0)	I_{FSM}	0.7	A
Total power dissipation Verlustleistung	P_{tot}	140	mW
ESD withstand voltage ESD Festigkeit (acc. to ANSI/ ESDA/ JEDEC JS-001 - HBM)	V_{ESD}	2	kV
Thermal resistance junction - ambient [1) page 12] Wärmewiderstand Sperrschicht - Umgebung 1) *Seite 12*	R_{thJA}	380	K / W
Thermal resistance junction - soldering point [2) page 12] Wämewiderstand Sperrschicht - Lötstelle [2) Seite 12]	R_{thJS}	220	K / W

Characteristics (T_A = 25 °C)
Kennwerte

Parameter **Bezeichnung**		**Symbol** **Symbol**	**Values** **Werte**	**Unit** **Einheit**
Emission wavelength Zentrale Emissionswellenlänge (I_F = 70 mA, t_p = 20 ms)	(typ)	λ_{peak}	860	nm
Centroid Wavelength Schwerpunktwellenlänge der Strahlung (I_F = 70 mA, t_p = 20 ms)	(typ)	$\lambda_{centroid}$	850	nm
Spectral bandwidth at 50% of I_{max} Spektrale Bandbreite bei 50% von I_{max} (I_F = 70 mA, t_p = 20 ms)	(typ)	$\Delta\lambda$	30	nm
Half angle Halbwinkel	(typ)	φ	± 10	°
Active chip area Aktive Chipfläche	(typ)	A	0.04	mm^2
Dimensions of active chip area Abmessungen der aktiven Chipfläche	(typ)	L x W	0.2 x 0.2	mm x mm
Rise and fall time of I_e (10% and 90% of $I_{e\,max}$) Schaltzeit von I_e (10% und 90% von $I_{e\,max}$) (I_F = 70 mA, R_L = 50 Ω)	(typ)	t_r, t_f	12	ns
Forward voltage Durchlassspannung (I_F = 70 mA, t_p = 20 ms)	(typ (max))	V_F	1.6 (≤ 2)	V
Forward voltage Durchlassspannung (I_F = 500 mA, t_p = 100 µs)	(typ (max))	V_F	2.4 (≤ 3)	V
Reverse current Sperrstrom (V_R = 5 V)	(typ (max))	I_R	not designed for reverse operation	µA
Total radiant flux Gesamtstrahlungsfluss (I_F = 70 mA, t_p = 20 ms)	(typ)	Φ_e	40	mW

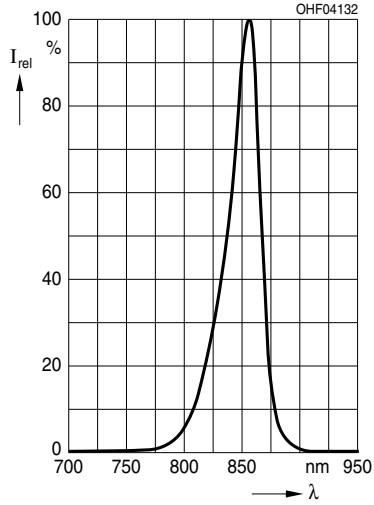
Parameter Bezeichnung		Symbol Symbol	Values Werte	Unit Einheit
Temperature coefficient of I_e or Φ_e Temperaturkoeffizient von I_e bzw. Φ_e (I_F = 70 mA, t_p = 20 ms)	(typ)	TC_I	-0.5	% / K
Temperature coefficient of V_F Temperaturkoeffizient von V_F (I_F = 70 mA, t_p = 20 ms)	(typ)	TC_V	-0.7	mV / K
Temperature coefficient of wavelength Temperaturkoeffizient der Wellenlänge (I_F = 70 mA, t_p = 20 ms)	(typ)	TC_λ	0.3	nm / K

Grouping (T_A = 25 °C)
Gruppierung

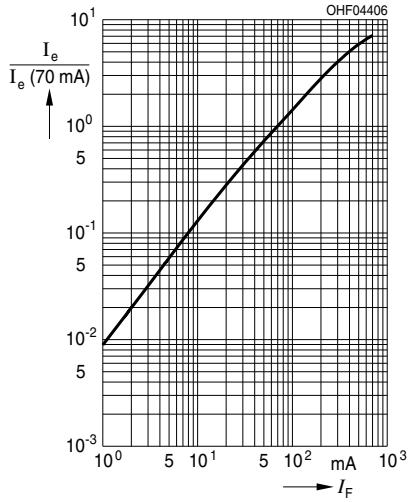
Group Gruppe	Min Radiant Intensity Min Strahlstärke I_F = 70 mA, t_p = 20 ms $I_{e, min}$ [mW / sr]	Max Radiant Intensity Max Strahlstärke I_F = 70 mA, t_p = 20 ms $I_{e, max}$ [mW / sr]	Typ Radiant Intensity Typ Strahlstärke I_F = 500 mA, t_p = 25 µs $I_{e, typ}$ [mW / sr]
SFH 4651-T	25	50	220
SFH 4651-U	40	80	360
SFH 4651-V	63	125	560
SFH 4651-AW	100	200	900

Note: measured at a solid angle of Ω = 0.01 sr
Only one group in one packing unit (variation lower 2:1).
Anm.: gemessen bei einem Raumwinkel Ω = 0.01 sr
Nur eine Gruppe in einer Verpackungseinheit (Streuung kleiner 2:1).

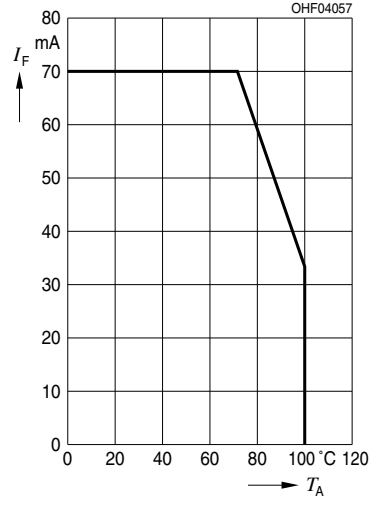
Relative Spectral Emission 3) *page 12*
Relative spektrale Emission 3) *Seite 12*
$I_{rel} = f(\lambda)$, $T_A = 25°C$



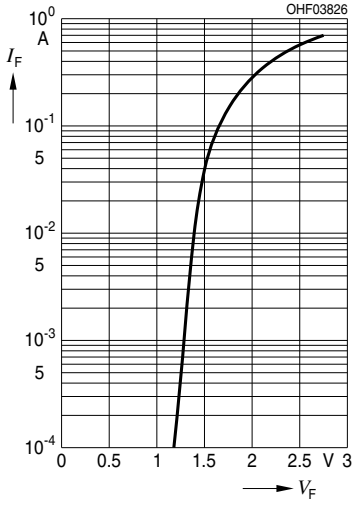
Radiant Intensity 3) *page 12*
Strahlstärke 3) *Seite 12*
$I_e / I_e(70\ mA) = f(I_F)$, single pulse, $t_p = 25\ \mu s$, $T_A = 25°C$



Max. Permissible Forward Current
Max. zulässiger Durchlassstrom
$I_{F,\ max} = f(T_A)$, $R_{thJA} = 380$ K/W

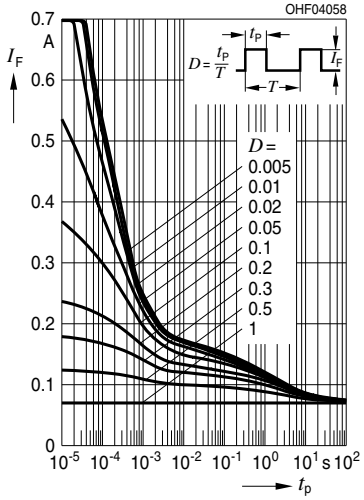


Forward Current 3) *page 12*
Durchlassstrom 3) *Seite 12*
$I_F = f(V_F)$, single pulse, $t_p = 100\ \mu s$, $T_A = 25°C$

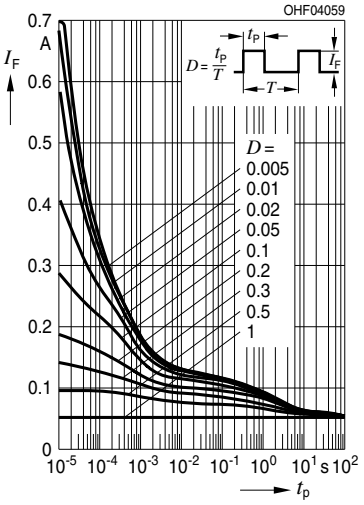


OSRAM Opto Semiconductors

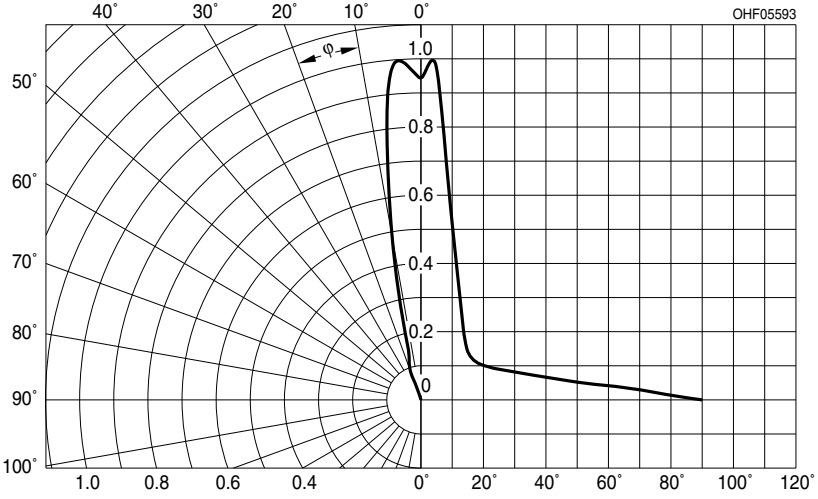
Permissible Pulse Handling Capability
Zulässige Pulsbelastbarkeit
$I_F = f(t_p)$, T_A = 25 °C, duty cycle D = parameter



Permissible Pulse Handling Capability
Zulässige Pulsbelastbarkeit
$I_F = f(t_p)$, T_A = 85 °C, duty cycle D = parameter

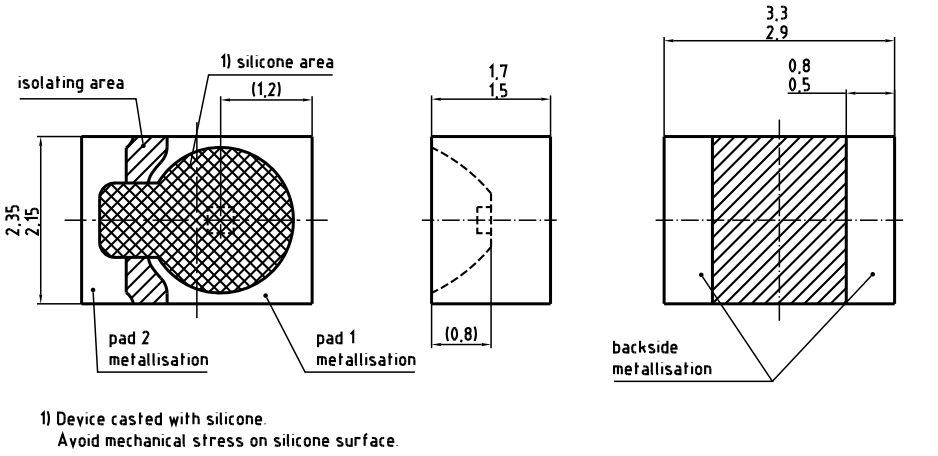


Radiation Characteristics 3) *page 12*
Abstrahlcharakteristik 3) *Seite 12*
$I_{rel} = f(\varphi)$



OSRAM
Opto Semiconductors

Package Outline
Maßzeichnung
(Schematic view only)

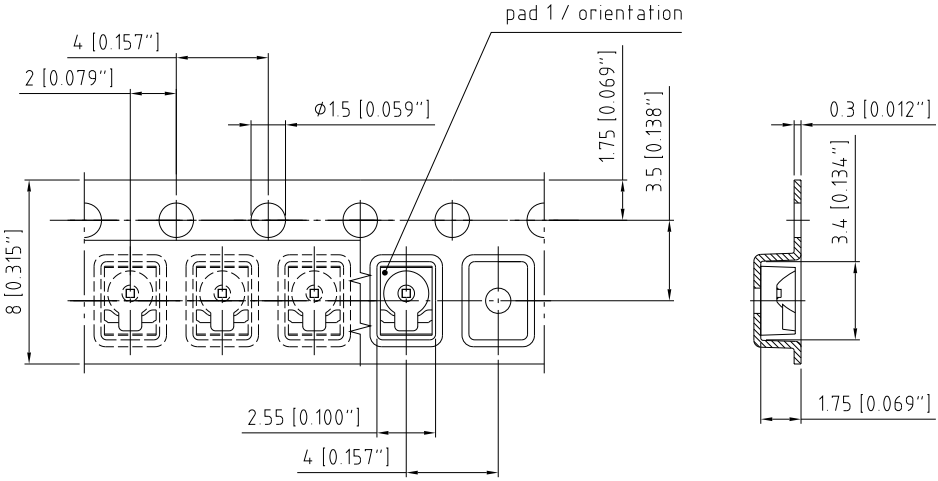


Dimensions in mm (inch). / Maße in mm (inch).

Pad **Pad**	**Description** **Beschreibung**
1	Anode / Anode
2	Cathode / Kathode

Package	MIDLED, Silicone, colourless, clear
Gehäuse	MIDLED, Silikon, farblos, klar
Note:	Schematic view only
Anm.:	Schematische Darstellung

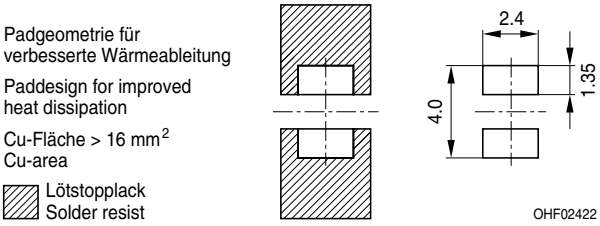
Method of Taping
Gurtung



Dimensions in mm (inch). / Maße in mm (inch).

Note:	Packing unit 2000/reel, ø180 mm or 9000/reel, ø330 mm
Anm.:	Verpackungseinheit 2000/Rolle, ø180 mm oder 9000/Rolle, ø330 mm

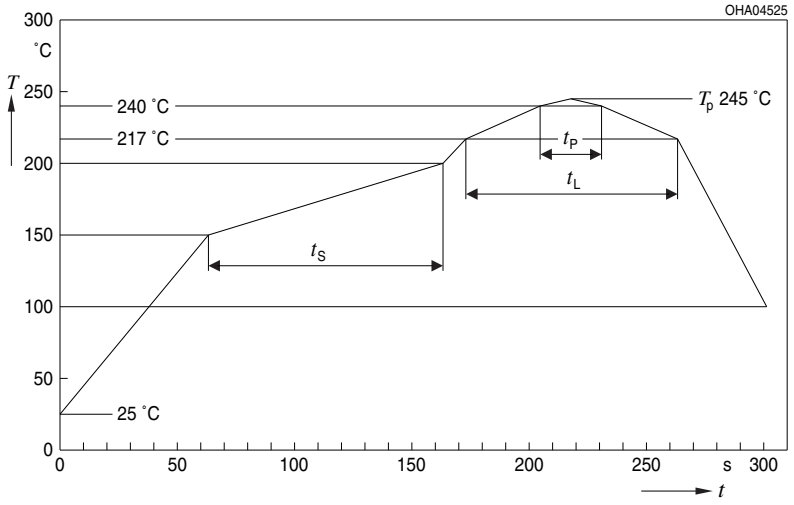
Recommended Solder Pad
Empfohlenes Lötpaddesign



Handling Indication: The package is casted with silicone. Mechanical stress at the surface of the unit should be as low as possible. /

Verarbeitungshinweis: Das Gehäuse ist mit Silikon vergossen. Mechanischer Stress auf der Bauteiloberfläche sollte so gering wie möglich gehalten werden.

Reflow Soldering Profile
Reflow-Lötprofil
Preconditioning: JEDEC Level 2 acc. to JEDEC J-STD-020D.01



OHA04612

Profile Feature Profil-Charakteristik	Symbol Symbol	Pb-Free (SnAgCu) Assembly			Unit Einheit
		Minimum	Recommendation	Maximum	
Ramp-up rate to preheat*) 25 °C to 150 °C			2	3	K/s
Time t_S T_{Smin} to T_{Smax}	t_S	60	100	120	s
Ramp-up rate to peak*) T_{Smax} to T_P			2	3	K/s
Liquidus temperature	T_L		217		°C
Time above liquidus temperature	t_L		80	100	s
Peak temperature	T_P		245	260	°C
Time within 5 °C of the specified peak temperature T_P - 5 K	t_P	10	20	30	s
Ramp-down rate* T_P to 100 °C			3	6	K/s
Time 25 °C to T_P				480	s

All temperatures refer to the center of the package, measured on the top of the component
* slope calculation DT/Dt: Dt max. 5 s; fulfillment for the whole T-range

Disclaimer

Attention please!
The information describes the type of component and shall not be considered as assured characteristics.
Terms of delivery and rights to change design reserved.
Due to technical requirements components may contain dangerous substances.
For information on the types in question please contact our Sales Organization.
If printed or downloaded, please find the latest version in the Internet.
Packing
Please use the recycling operators known to you. We can also help you – get in touch with your nearest sales office.
By agreement we will take packing material back, if it is sorted. You must bear the costs of transport. For packing material that is returned to us unsorted or which we are not obliged to accept, we shall have to invoice you for any costs incurred.
Components used in life-support devices or systems must be expressly authorized for such purpose!
Critical components* may only be used in life-support devices** or systems with the express written approval of OSRAM OS.

*) A critical component is a component used in a life-support device or system whose failure can reasonably be expected to cause the failure of that life-support device or system, or to affect its safety or the effectiveness of that device or system.
**) Life support devices or systems are intended (a) to be implanted in the human body, or (b) to support and/or maintain and sustain human life. If they fail, it is reasonable to assume that the health and the life of the user may be endangered.

Disclaimer

Bitte beachten!
Lieferbedingungen und Änderungen im Design vorbehalten. Aufgrund technischer Anforderungen können die Bauteile Gefahrstoffe enthalten. Für weitere Informationen zu gewünschten Bauteilen, wenden Sie sich bitte an unseren Vertrieb. Falls Sie dieses Datenblatt ausgedruckt oder heruntergeladen haben, finden Sie die aktuellste Version im Internet.
Verpackung
Benutzen Sie bitte die Ihnen bekannten Recyclingwege. Wenn diese nicht bekannt sein sollten, wenden Sie sich bitte an das nächstgelegene Vertriebsbüro. Wir nehmen das Verpackungsmaterial zurück, falls dies vereinbart wurde und das Material sortiert ist. Sie tragen die Transportkosten. Für Verpackungsmaterial, das unsortiert an uns zurückgeschickt wird oder das wir nicht annehmen müssen, stellen wir Ihnen die anfallenden Kosten in Rechnung.
Bauteile, die in lebenserhaltenden Apparaten und Systemen eingesetzt werden, müssen für diese Zwecke ausdrücklich zugelassen sein!
Kritische Bauteile* dürfen in lebenserhaltenden Apparaten und Systemen** nur dann eingesetzt werden, wenn ein schriftliches Einverständnis von OSRAM OS vorliegt.

*) Ein kritisches Bauteil ist ein Bauteil, das in lebenserhaltenden Apparaten oder Systemen eingesetzt wird und dessen Defekt voraussichtlich zu einer Fehlfunktion dieses lebenserhaltenden Apparates oder Systems führen wird oder die Sicherheit oder Effektivität dieses Apparates oder Systems beeinträchtigt.
**) Lebenserhaltende Apparate oder Systeme sind für (a) die Implantierung in den menschlichen Körper oder (b) für die Lebenserhaltung bestimmt. Falls Sie versagen, kann davon ausgegangen werden, dass die Gesundheit und das Leben des Patienten in Gefahr ist.

Glossary

1) **Thermal resistance:** junction -ambient, mounted on PC-board (FR4), padsize 16 mm^2 each

2) **Thermal resistance:** junction -soldering point, mounted on metal block

3) **Typical Values:** Due to the special conditions of the manufacturing processes of LED, the typical data or calculated correlations of technical parameters can only reflect statistical figures. These do not necessarily correspond to the actual parameters of each single product, which could differ from the typical data and calculated correlations or the typical characteristic line. If requested, e.g. because of technical improvements, these typ. data will be changed without any further notice.

Glossar

1) **Wärmewiderstand:** Sperrschicht -Umgebung, bei Montage auf FR4 Platine, Padgröße je 16 mm^2

2) **Wärmewiderstand:** Sperrschicht -Lötstelle, bei Montage auf Metall-Block

3) **Typische Werte:** Wegen der besonderen Prozessbedingungen bei der Herstellung von LED können typische oder abgeleitete technische Parameter nur aufgrund statistischer Werte wiedergegeben werden. Diese stimmen nicht notwendigerweise mit den Werten jedes einzelnen Produktes überein, dessen Werte sich von typischen und abgeleiteten Werten oder typischen Kennlinien unterscheiden können. Falls erforderlich, z.B. aufgrund technischer Verbesserungen, werden diese typischen Werte ohne weitere Ankündigung geändert.

Published by OSRAM Opto Semiconductors GmbH
Leibnizstraße 4, D-93055 Regensburg

EU RoHS and China RoHS compliant product

此产品符合欧盟 RoHS 指令的要求；

按照中国的相关法规和标准，不含有毒有害物质或元素。

OSRAM
Opto Semiconductors

BURR - BROWN®

OPT101

MONOLITHIC PHOTODIODE AND SINGLE-SUPPLY TRANSIMPEDANCE AMPLIFIER

FEATURES

- SINGLE SUPPLY: +2.7 to +36V
- PHOTODIODE SIZE: 0.090 x 0.090 inch
- INTERNAL 1MΩ FEEDBACK RESISTOR
- HIGH RESPONSIVITY: 0.45A/W (650nm)
- BANDWIDTH: 14kHz at R_F = 1MΩ
- LOW QUIESCENT CURRENT: 120μA
- AVAILABLE IN 8-PIN DIP, 5-PIN SIP, AND 8-LEAD SURFACE MOUNT PACKAGES

APPLICATIONS

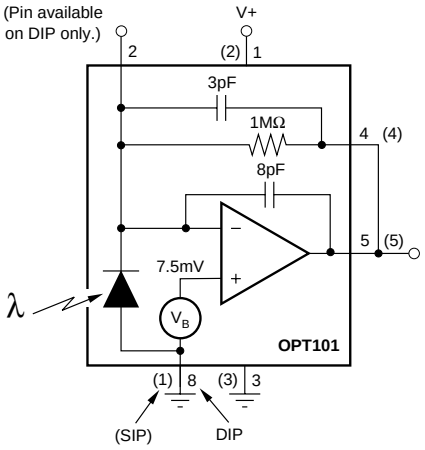
- MEDICAL INSTRUMENTATION
- LABORATORY INSTRUMENTATION
- POSITION AND PROXIMITY SENSORS
- PHOTOGRAPHIC ANALYZERS
- BARCODE SCANNERS
- SMOKE DETECTORS
- CURRENCY CHANGERS

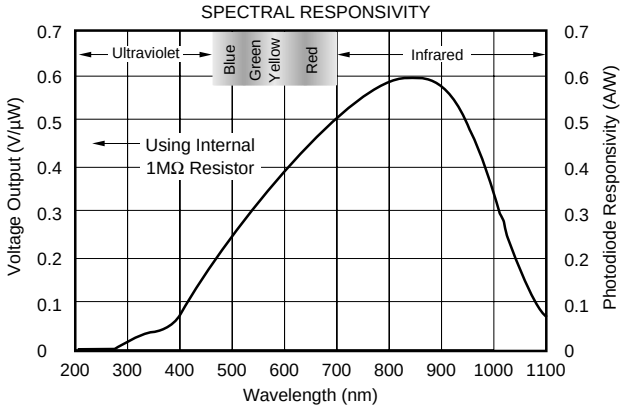
DESCRIPTION

The OPT101 is a monolithic photodiode with on-chip transimpedance amplifier. Output voltage increases linearly with light intensity. The amplifier is designed for single or dual power supply operation, making it ideal for battery operated equipment.

The integrated combination of photodiode and transimpedance amplifier on a single chip eliminates the problems commonly encountered in discrete designs such as leakage current errors, noise pick-up and gain peaking due to stray capacitance. The 0.09 x 0.09 inch photodiode is operated in the photoconductive mode for excellent linearity and low dark current.

The OPT101 operates from +2.7V to +36V supplies and quiescent current is only 120μA. It is available in clear plastic 8-pin DIP, 5-pin SIP and J-formed DIP for surface mounting. Temperature range is 0°C to 70°C.





International Airport Industrial Park • Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 • Tel: (520) 746-1111 • Twx: 910-952-1111
Internet: http://www.burr-brown.com/ • FAXLine: (800) 548-6133 (US/Canada Only) • Cable: BBRCORP • Telex: 066-6491 • FAX: (520) 889-1510 • Immediate Product Info: (800) 548-6132

SPECIFICATIONS

At T_A = +25°C, V_S = +2.7V to +36V, λ = 650nm, internal 1MΩ feedback resistor, and R_L = 10kΩ, unless otherwise noted.

PARAMETER	CONDITIONS	OPT101P, W			UNITS
		MIN	TYP	MAX	
RESPONSIVITY					
Photodiode Current	650nm		0.45		A/W
Voltage Output	650nm		0.45		V/µW
vs Temperature			100		ppm/°C
Unit to Unit Variation	650nm		±5		%
Nonlinearity(1)	FS Output = 24V		±0.01		% of FS
Photodiode Area	(0.090 x 0.090in)		0.008		in^2
	(2.29 x 2.29mm)		5.2		mm^2
DARK ERRORS, RTO(2)					
Offset Voltage, Output		+5	+7.5	+10	mV
vs Temperature			±2.5		µV/°C
vs Power Supply	V_S = +2.7V to +36V		10	100	µV/V
Voltage Noise, Dark, f_B = 0.1Hz to 20kHz	V_S = +15V, V_{PIN3} = –15V		300		µVrms
TRANSIMPEDANCE GAIN					
Resistor			1		MΩ
Tolerance, P			±0.5	±2	%
W			±0.5		%
vs Temperature			±50		ppm/°C
FREQUENCY RESPONSE					
Bandwidth	V_{OUT} = 10Vp-p		14		kHz
Rise Fall Time, 10% to 90%	V_{OUT} = 10V Step		28		µs
Settling Time, 0.05%	V_{OUT} = 10V Step		160		µs
0.1%			80		µs
1%			70		µs
Overload Recovery	100%, Return to Linear Operation		50		µs
OUTPUT					
Voltage Output, High		(V_S) – 1.3	(V_S) – 1.15		V
Capacitive Load, Stable Operation			10		nF
Short-Circuit Current	V_S = 36V		15		mA
POWER SUPPLY					
Operating Voltage Range		+2.7		+36	V
Quiescent Current	Dark, V_{PIN3} = 0V		120	240	µA
	R_L = ∞, V_{OUT} = 10V		220		µA
TEMPERATURE RANGE					
Specification		0		+70	°C
Operating		0		+70	°C
Storage		–25		+85	°C
Thermal Resistance, θ_{JA}			100		°C/W

NOTES: (1) Deviation in percent of full scale from best-fit straight line. (2) Referred to Output. Includes all error sources.

PHOTODIODE SPECIFICATIONS

T_A = +25°C, V_S = +2.7V to +36V unless otherwise noted.

PARAMETER	CONDITIONS	Photodiode of OPT101P			UNITS
		MIN	TYP	MAX	
Photodiode Area	(0.090 x 0.090in)		0.008		in^2
	(2.29 x 2.29mm)		5.2		mm^2
Current Responsivity	650nm		0.45		A/W
	650nm		865		µA/W/cm^2
Dark Current	V_{DIODE} = 7.5mV		2.5		pA
vs Temperature			doubles every 7°C		
Capacitance			1200		pF

OP AMP SPECIFICATIONS

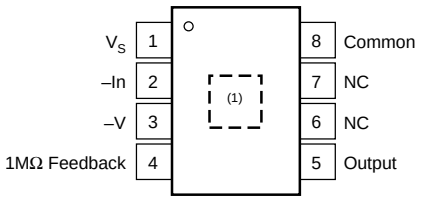
At T_A = +25°C, V_S = +2.7V to +36V, λ = 650nm, internal 1MΩ feedback resistor, and R_L = 10kΩ, unless otherwise noted.

PARAMETER	CONDITIONS	OPT101 Op Amp(1) MIN	TYP	MAX	UNITS
INPUT					
Offset Voltage			±0.5		mV
vs Temperature			±2.5		µV/°C
vs Power Supply			10		µV/V
Input Bias Current	(–) Input		165		pA
vs Temperature	(–) Input		1		pA/°C
Input Impedance					
Differential			400 || 5		MΩ || pF
Common-Mode			250 || 35		GΩ || pF
Common-Mode Input Voltage Range	Linear Operation		0 to [(V_S) – 1]		V
Common-Mode Rejection			90		dB
OPEN-LOOP GAIN					
Open-loop Voltage Gain			90		dB
FREQUENCY RESPONSE					
Gain-Bandwidth Product(2)			2		MHz
Slew Rate			1		V/µs
Settling Time 1%			5.8		µs
0.1%			7.7		µs
0.05%			8.0		µs
OUTPUT					
Voltage Output, High		(V_S) – 1.3	(V_S) – 1.15		V
Short-Circuit Current	V_S = +36V		15		mA
POWER SUPPLY					
Operating Voltage Range		+2.7		+36	V
Quiescent Current	Dark, V_{PIN3} = 0V		120	240	µA
	R_L ∞, V_{OUT} = 10V		220		µA

NOTES: (1) Op amp specifications provided for information and comparison only. (2) Stable gains ≥ 10V/V.

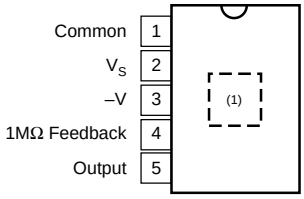
PIN CONFIGURATIONS

Top View **DIP**



NOTE: (1) Photodiode location.

Top View **SIP**



ABSOLUTE MAXIMUM RATINGS

Supply Voltage (V_S to *"Common"* or pin 3)	0 to +36V
Output Short-Circuit (to ground)	Continuous
Operating Temperature	–25°C to +85°C
Storage Temperature	–25°C to +85°C
Junction Temperature	+85°C
Lead Temperature (soldering, 10s)	+300°C
(Vapor-Phase Soldering Not Recommended)	

PACKAGE INFORMATION

PRODUCT	COLOR	PACKAGE	PACKAGE DRAWING NUMBER[1]
OPT101P	Clear	8-Pin Plastic DIP	006-1
OPT101P-J	Clear	8-Lead Surface Mount [2]	006-4
OPT101W	Clear	5-Pin Plastic SIP	321

NOTE: (1) For detailed drawing and dimension table, please see end of data sheet, or Appendix C of Burr-Brown IC Data Book. (2) 8-pin DIP with J-formed leads for surface mounting.

ELECTROSTATIC DISCHARGE SENSITIVITY

This integrated circuit can be damaged by ESD. Burr-Brown recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

MOISTURE SENSITIVITY AND SOLDERING

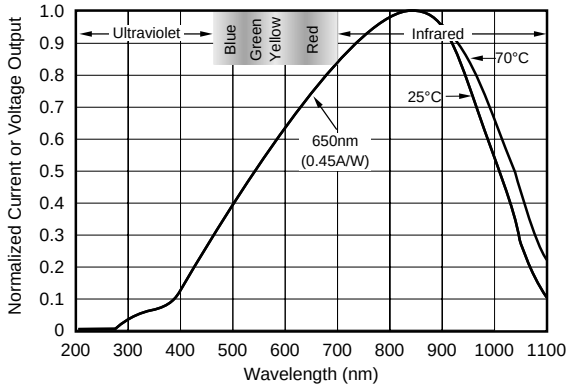
Clear plastic does not contain the structural-enhancing fillers used in black plastic molding compound. As a result, clear plastic is more sensitive to environmental stress than black plastic. This can cause difficulties if devices have been stored in high humidity prior to soldering. The rapid heating during soldering can stress wire bonds and cause failures. Prior to soldering, it is recommended that plastic devices be baked-out at +85°C for 24 hours.

The fire-retardant fillers used in black plastic are not compatible with clear molding compound. The OPT101 plastic packages cannot meet flammability test, UL-94.

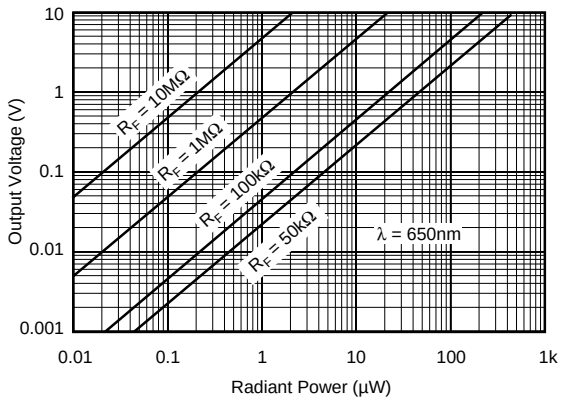
TYPICAL PERFORMANCE CURVES

At T_A = +25°C, V_S = +2.7V to +36V, λ = 650nm, internal 1MΩ feedback resistor, and R_L = 10kΩ, unless otherwise noted.

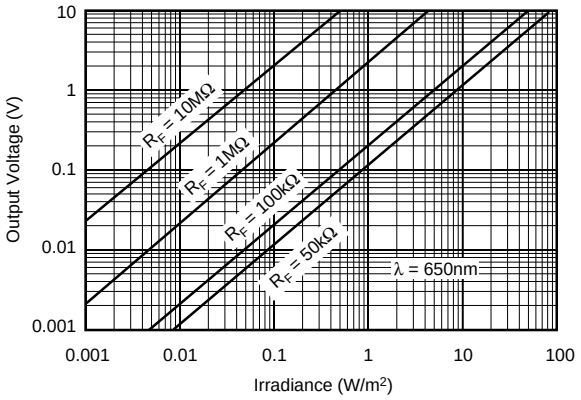
NORMALIZED SPECTRAL RESPONSIVITY



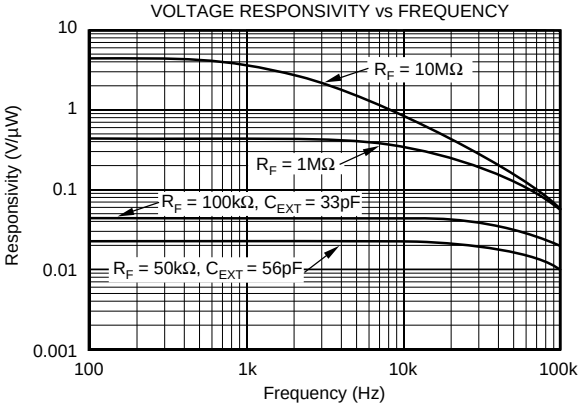
VOLTAGE RESPONSIVITY vs RADIANT POWER



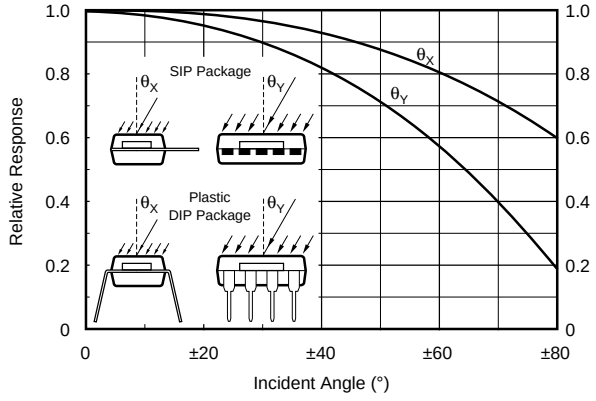
VOLTAGE RESPONSIVITY vs IRRADIANCE



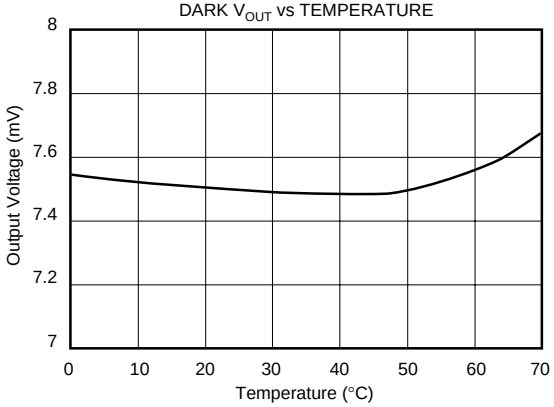
VOLTAGE RESPONSIVITY vs FREQUENCY



RESPONSE vs INCIDENT ANGLE



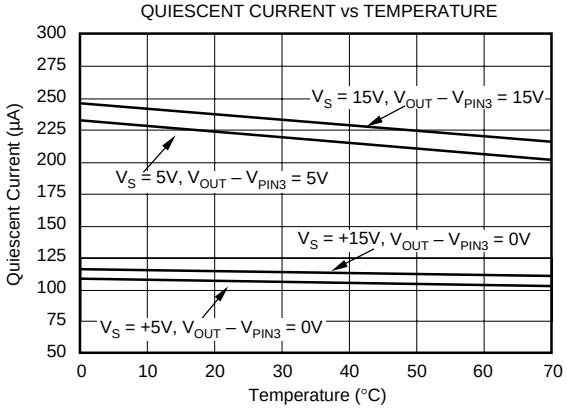
DARK V_{OUT} vs TEMPERATURE



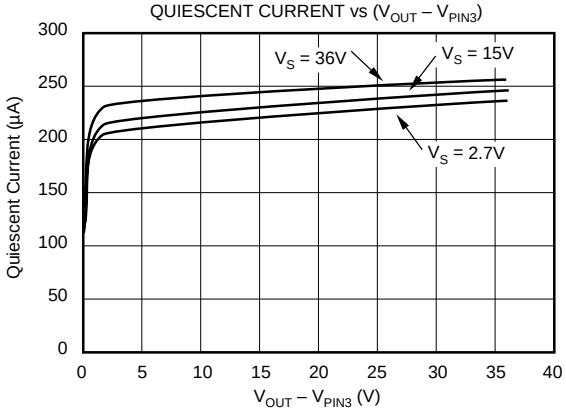
TYPICAL PERFORMANCE CURVES (CONT)

At T_A = +25°C, V_S = +2.7V to +36V, λ = 650nm, internal 1MΩ feedback resistor, and R_L = 10kΩ, unless otherwise noted.

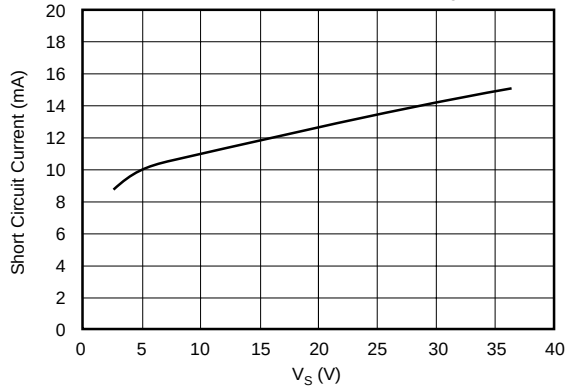
QUIESCENT CURRENT vs TEMPERATURE



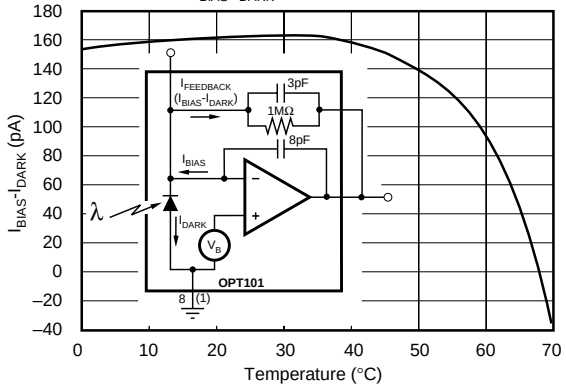
QUIESCENT CURRENT vs (V_{OUT} – V_{PIN3})



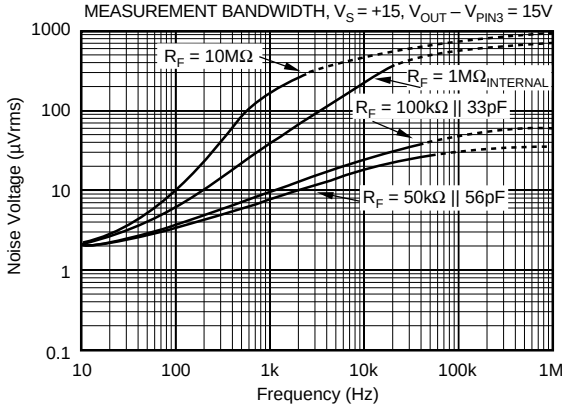
SHORT CIRCUIT CURRENT vs V_S



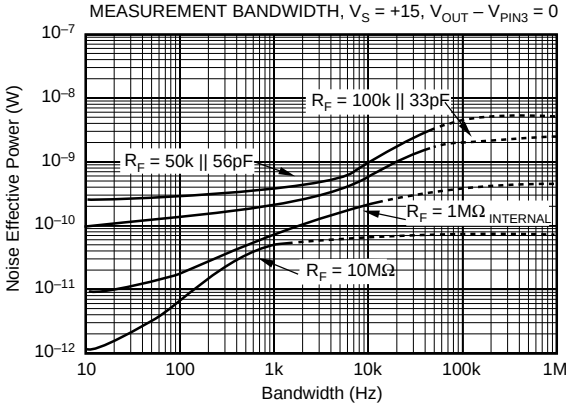
(I_{BIAS}-I_{DARK}) vs TEMPERATURE



OUTPUT NOISE VOLTAGE vs
MEASUREMENT BANDWIDTH, V_S = +15, V_{OUT} – V_{PIN3} = 15V



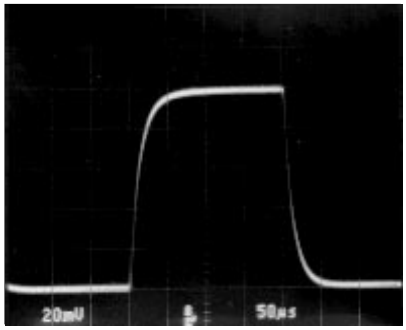
NOISE EFFECTIVE POWER vs
MEASUREMENT BANDWIDTH, V_S = +15, V_{OUT} – V_{PIN3} = 0



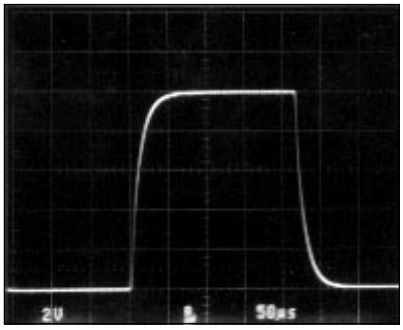
TYPICAL PERFORMANCE CURVES (CONT)

At T_A = +25°C, V_S = +2.7V to +36V, λ = 650nm, internal 1MΩ feedback resistor, and R_L = 10kΩ, unless otherwise noted.

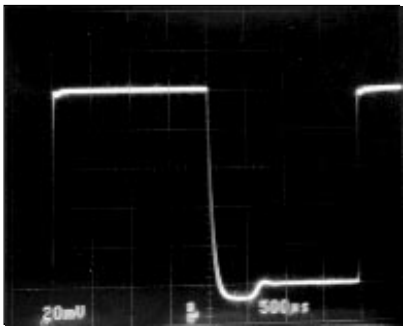
SMALL SIGNAL RESPONSE



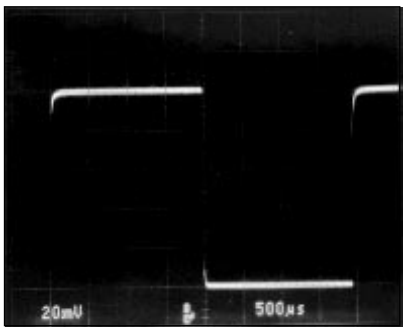
LARGE SIGNAL RESPONSE



SMALL SIGNAL RESPONSE (C_{LOAD} = 10,000 pF)
(Pin 3 = 0V)



SMALL SIGNAL RESPONSE (C_{LOAD} = 10,000 pF)
(Pin 3 = –15V)



APPLICATIONS INFORMATION

Figure 1 shows the basic connections required to operate the OPT101. Applications with high-impedance power supplies may require decoupling capacitors located close to the device pins as shown. Output is 7.5mV dc with no light and increases with increasing illumination.

Photodiode current, I_D, is proportional to the radiant power, or flux, (in watts) falling on the photodiode. At a wavelength of 650nm (visible red) the photodiode Responsivity, R_I, is approximately 0.45A/W. Responsivity at other wavelengths is shown in the typical performance curve “Responsivity vs Wavelength.”

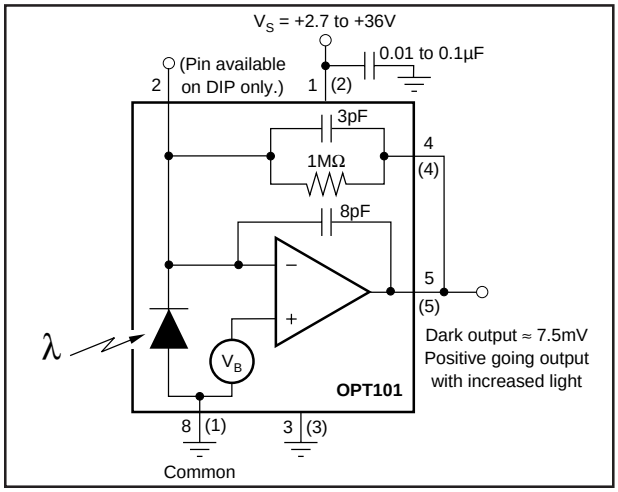


FIGURE 1. Basic Circuit Connections.

The typical performance curve “Output Voltage vs Radiant Power” shows the response throughout a wide range of radiant power. The response curve “Output Voltage vs Irradiance” is based on the photodiode area of 5.2mm^2.

The OPT101’s voltage output is the product of the photodiode current times the feedback resistor, ($I_D R_F$), plus a pedestal voltage, V_B, of approximately 7.5mV introduced for single supply operation. The internal feedback resistor is laser trimmed to 1MΩ. Using this resistor, the output voltage responsivity, R_V, is approximately 0.45V/µW at 650nm wavelength. Figure 1 shows the basic circuit connections for the OPT101 operating with a single power supply and using the internal 1MΩ feedback resistor for a response of 0.45V/µW at 650nm. Pin 3 is connected to common in this configuration.

CAPACITIVE LOADING

The OPT101 is capable of driving load capacitances of 10nF without instability. However, dynamic performance with capacitive loads can be improved by applying a negative bias voltage to Pin 3 (shown in Figure 2). This negative power supply voltage allows the output to go negative in response to the reactive effect of a capacitive load. An internal JFET connected between pin 5 (output) and pin 3 allows the output to sink current. This current sink capability can also be useful when driving the capacitive inputs of some analog-to-digital converters which require the signal source to sink currents up to approximately 100µA. The benefits of this current sink are shown in the typical performance curves “Small Signal Response (C_{LOAD} = 10,000pF)” which compare operation with pin 3 grounded and connected to –15V.

Due to the architecture of this output stage current sink, there is a slight increase in operating current when there is a voltage between pin 3 and the output. Depending on the magnitude of this voltage, the quiescent current will increase by approximately 100µA as shown in the typical performance curve "Quiescent Current vs ($V_{OUT} - V_{PIN3}$)".

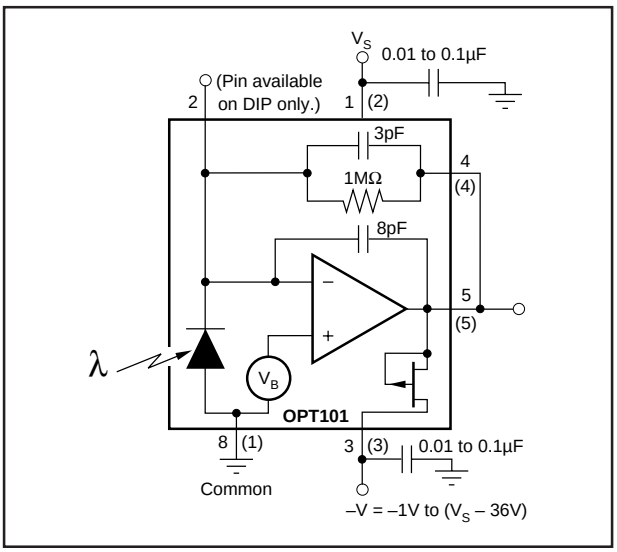


FIGURE 2. Bipolar Power Supply Circuit Connections.

NOISE PERFORMANCE

Noise performance of the OPT101 is determined by the op amp characteristics, feedback components and photodiode capacitance. The typical performance curve “Output Noise Voltage vs Measurement Bandwidth” shows how the noise varies with R_F and measured bandwidth (0.1Hz to the indicated frequency), when the output voltage minus the voltage on pin 3 is greater than approximately 50mV. Below this level, the output stage is powered down, and the effective bandwidth is decreased. This reduces the noise to approximately 1/3 the nominal noise value of 300µVrms, or 100µVrms. This enables a low level signal to be resolved.

Noise can be reduced by filtering the output with a cutoff frequency equal to the signal bandwidth. This will improve signal-to-noise ratio. Also, output noise increases in proportion to the square root of the feedback resistance, while responsivity increases linearly with feedback resistance. Best signal-to-noise ratio is achieved with large feedback resistance. This comes with the trade-off of decreased bandwidth.

The noise performance of the photodetector is sometimes characterized by *Noise Effective Power* (NEP). This is the radiant power that would produce an output signal equal to the noise level. NEP has the units of radiant power (watts), or Watts/$\sqrt{\text{Hz}}$ to convey spectral information about the noise. The typical performance curve “Noise Effective Power” vs Measurement Bandwidth" illustrates the NEP for the OPT101.

DARK ERRORS

The dark errors in the specification table include all sources. The dominant source of dark output voltage is the "pedestal" voltage applied to the non-inverting input of the op amp. This voltage is introduced to provide linear operation in the absence of light falling on the photodiode. Photodiode dark current is approximately 2.5pA and contributes virtually no offset error at room temperature. The bias current of the op amp's summing junction (– input) is approximately 165pA. The dark current will be subtracted from the amplifier's bias current, and this residual current will flow through the feedback resistor creating an offset. The effects of temperature on this difference current can be seen in the typical performance curve "($I_{BIAS} - I_{DARK}$) vs Temperature." The dark output voltage can be trimmed to zero with the optional circuit shown in Figure 3. A low impedance offset driver (op amp) should be used to drive pin 8 (DIP) because this node has signal-dependent currents.

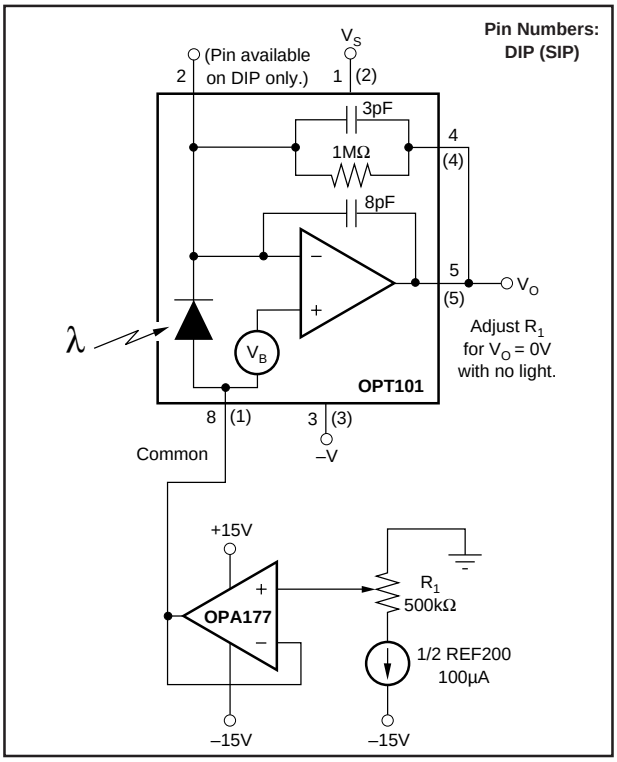
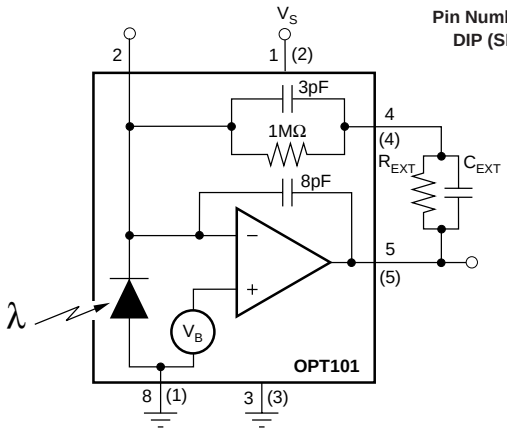


FIGURE 3. Dark Error (Offset) Adjustment Circuit.

CHANGING RESPONSIVITY

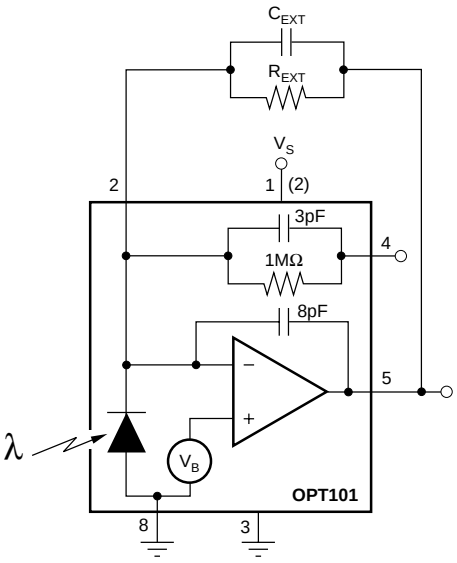
An external resistor, R_{EXT}, can be connected to set a different voltage responsivity. To increase the responsivity, this resistor can be placed in series with the internal 1MΩ (Figure 4a), or with the DIP package, the external resistor can replace the internal resistor by not connecting pin 4 (Figure 4b). The second configuration also allows the circuit gain to be reduced below 10^6V/A by using external resistors of less than 1MΩ.

Figure 4 includes tables showing the responsivity and bandwidth. For values of R_F less than 1MΩ, an external capacitor, C_{EXT} should be connected in parallel with R_F. This capacitor eliminates gain peaking and prevents instability. The value of C_{EXT} can be determined from the table in Figure 4. Values of R_F, other than shown in the table, can be interpolated.



(a)-Series R_{EXT} (for SIP package).

R_{EXT} (MΩ)	C_{EXT} (pF)	DC Gain (x10^6V/A)	Bandwidth (kHz)
1	50	2	8
2	25	3	6
5	10	6	2.5
10	5	11	1.3
50	—	51	0.33



(b)-External Feedback (for DIP package).

R_{EXT} (MΩ)	C_{EXT} (pF)	DC Gain (x10^6V/A)	Bandwidth (kHz)
0.05(1)	56	0.05	58
0.1(1)	33	0.1	44
1	—	1	23
2	—	2	9.4
5	—	5	3.6
10	—	10	1.8
50	—	50	0.34

Note: (1) May require 1kΩ in series with pin 5 when driving large capacitances.

FIGURE 4. Changing Responsivity with External Resistor.

LIGHT SOURCE POSITIONING

The OPT101 is tested with a light source that uniformly illuminates the full area of the integrated circuit, including the op amp. Although IC amplifiers are light-sensitive to some degree, the OPT101 op amp circuitry is designed to minimize this effect. Sensitive junctions are shielded with metal, and the photodiode area is very large relative to the op amp input circuitry.

If your light source is focused to a small area, be sure that it is properly aimed to fall on the photodiode. A narrowly focused beam falling on only the photodiode will provide improved settling times compared to a source that uniformly illuminates the full area of the die. If a narrowly focused light source were to miss the photodiode area and fall only on the op amp circuitry, the OPT101 would not perform properly. The large 0.09" x 0.09" (2.29mm x 2.29mm) photodiode area allows easy positioning of narrowly focused light sources. The photodiode area is easily visible, as it appears very dark compared to the surrounding active circuitry.

The incident angle of the light source also effects the apparent sensitivity in uniform irradiance. For small incident angles, the loss in sensitivity is simply due to the smaller effective light gathering area of the photodiode (proportional to the cosine of the angle). At a greater incident angle, light is diffracted and scattered by the package. These effects are shown in the typical performance curve "Responsivity vs Incident Angle."

DYNAMIC RESPONSE

Using the internal 1MΩ resistor, the dynamic response of the photodiode/op amp combination can be modeled as a simple R • C circuit with a –3dB cutoff frequency of approximately 14kHz. The R and C values are 1MΩ and 11pF respectively. By using external resistors, with less than 3pF parasitic capacitance, the frequency response can be improved. An external 1MΩ resistor used in the configuration shown in Figure 4b will create a 23kHz bandwidth with the same 10^6V/A dc transimpedance gain. This yields a rise time of approximately 15µs (10% to 90%). Dynamic response is not limited by op amp slew rate. This is demonstrated by the dynamic response oscilloscope photographs showing virtually identical large-signal and small-signal response.

Dynamic response will vary with feedback resistor value as shown in the typical performance curve "Responsivity vs Frequency." Rise time (10% to 90%) will vary according to the –3dB bandwidth produced by a given feedback resistor value:

$$t_r = \frac{0.35}{f_C}$$

where:

t_r is the rise time (10% to 90%)
f_C is the –3dB bandwidth

LINEARITY PERFORMANCE

The photodiode is operated in the photoconductive mode so the current output of the photodiode is very linear with radiant power throughout a wide range. Nonlinearity remains below approximately 0.05% up to 100µA photodiode current. The photodiode can produce output currents of 1mA or greater with high radiant power, but nonlinearity increases to several percent in this region.

This very linear performance at high radiant power assumes that the full photodiode area is uniformly illuminated. If the light source is focused to a small area of the photodiode, nonlinearity will occur at lower radiant power.

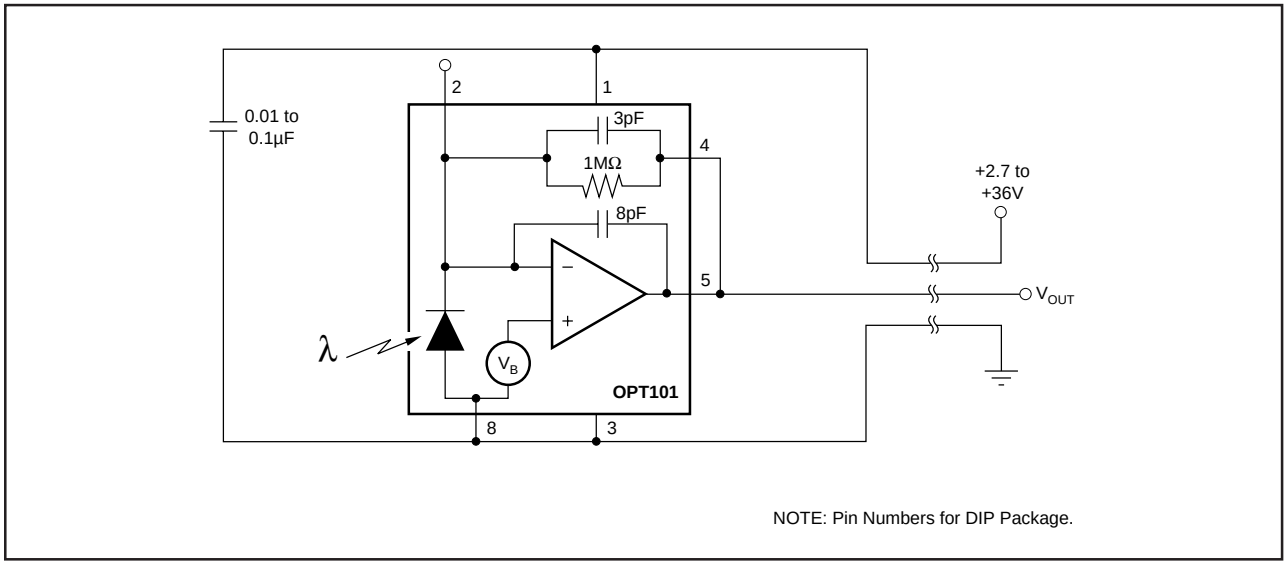


FIGURE 5. Three-Wire Remote Light Measurement.

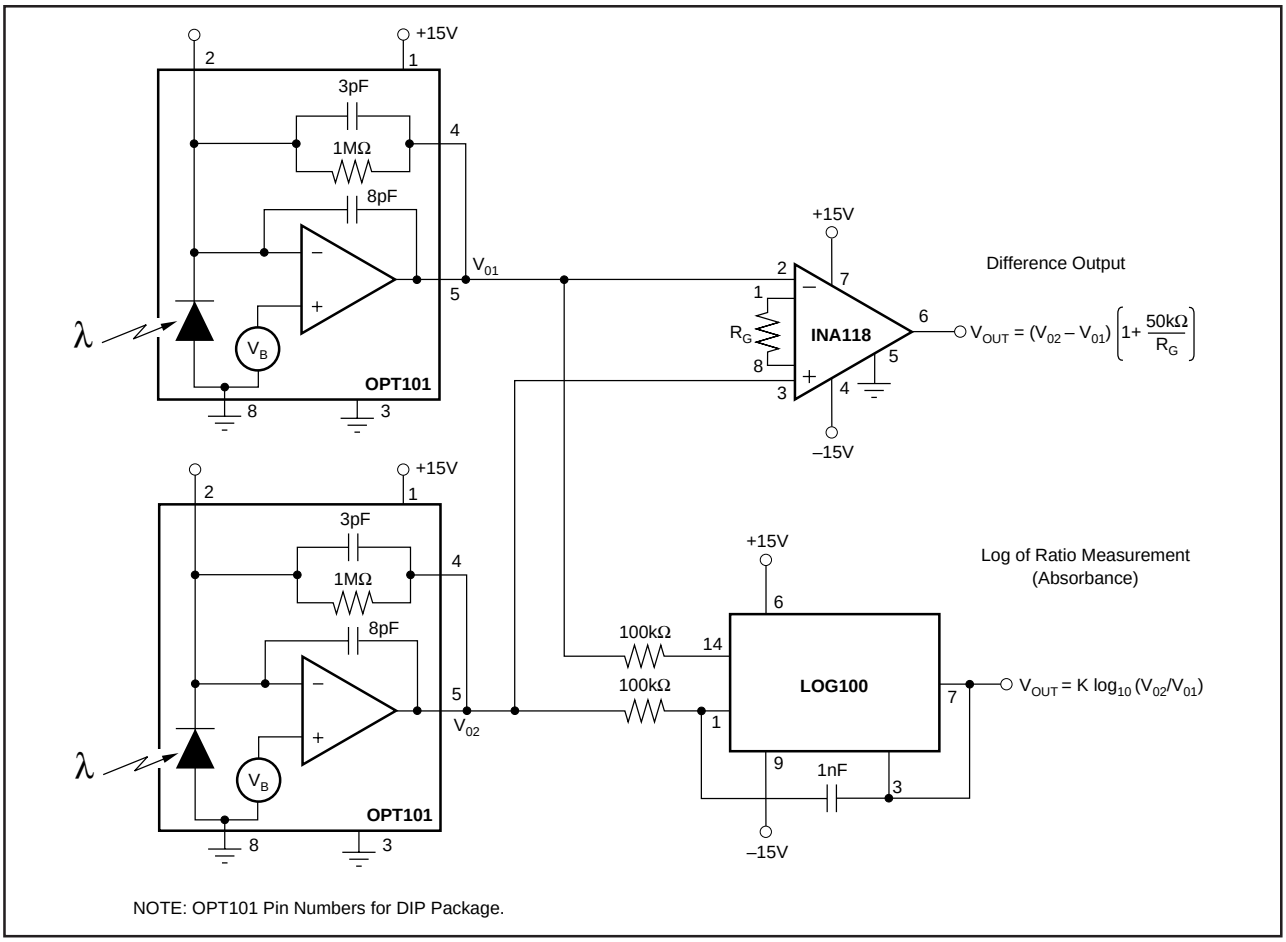


FIGURE 6. Differential Light Measurement.

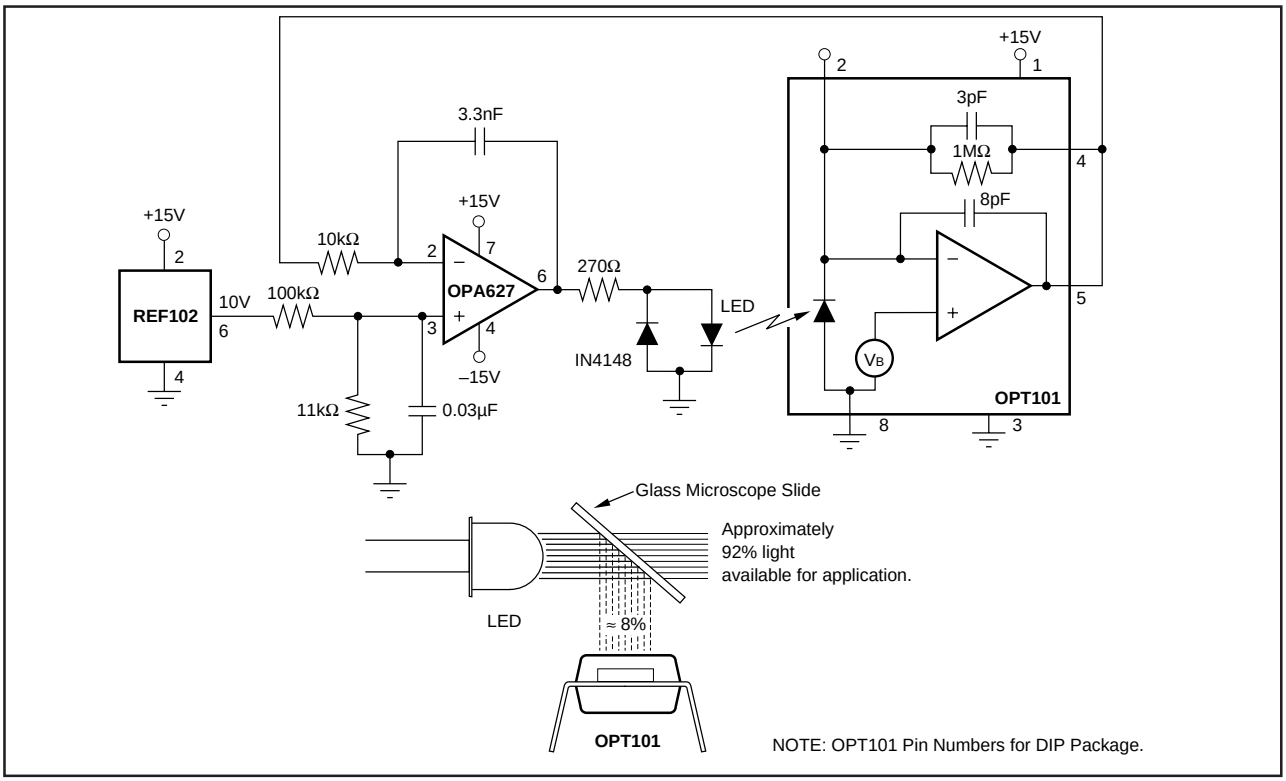


FIGURE 7. LED Output Regulation Circuit.

NTR4503N, NVTR4503N

Power MOSFET

30 V, 2.5 A, Single N−Channel, SOT−23

Features

- Leading Planar Technology for Low Gate Charge / Fast Switching
- 4.5 V Rated for Low Voltage Gate Drive
- SOT−23 Surface Mount for Small Footprint (3 x 3 mm)
- NV Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC−Q101 Qualified and PPAP Capable
- These Devices are Pb−Free and are RoHS Compliant

Applications

- DC−DC Conversion
- Load/Power Switch for Portables
- Load/Power Switch for Computing

MAXIMUM RATINGS (T_J = 25°C unless otherwise noted)

Parameter			Symbol	Value	Unit
Drain−to−Source Voltage			V_{DSS}	30	V
Gate−to−Source Voltage			V_{GS}	±20	V
Continuous Drain Current (Note 1)	Steady State	T_A = 25°C	I_D	2.0	A
		T_A = 85°C		1.5	
	t ≤ 10 s	T_A = 25°C		2.5	
Power Dissipation (Note 1)	Steady State	T_A = 25°C	P_D	0.73	W
Continuous Drain Current (Note 2)	Steady State	T_A = 25°C	I_D	1.5	A
		T_A = 85°C		1.1	
Power Dissipation (Note 2)		T_A = 25°C	P_D	0.42	W
Pulsed Drain Current	t_p = 10 μs		I_{DM}	10	A
Operating Junction and Storage Temperature			T_J, T_{stg}	−55 to 150	°C
Source Current (Body Diode)			I_S	2.0	A
Peak Source Current (Diode Forward)		t_p = 10 μs	I_{SM}	4.0	A
Lead Temperature for Soldering Purposes (1/8" from case for 10 s)			T_L	260	°C

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

THERMAL RESISTANCE RATINGS

Parameter	Symbol	Max	Unit
Junction−to−Ambient − Steady State (Note 1)	$R_{\theta JA}$	170	°C/W
Junction−to−Ambient − t < 10 s (Note 1)	$R_{\theta JA}$	100	
Junction−to−Ambient − Steady State (Note 2)	$R_{\theta JA}$	300	

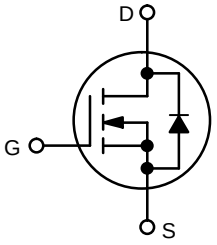
1. Surface−mounted on FR4 board using 1 in sq pad size.
2. Surface−mounted on FR4 board using the minimum recommended pad size.

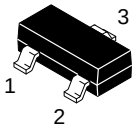
ON Semiconductor®

www.onsemi.com

$V_{(BR)DSS}$	$R_{DS(on)}$ TYP	I_D MAX
30 V	85 mΩ @ 10 V	2.5 A
	105 mΩ @ 4.5 V	

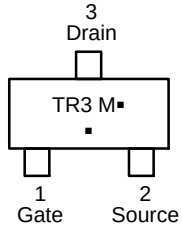
N−Channel





SOT−23
CASE 318
STYLE 21

MARKING DIAGRAM/ PIN ASSIGNMENT



TR3 = Specific Device Code
M = Date Code
▪ = Pb−Free Package

(Note: Microdot may be in either location)

ORDERING INFORMATION

Device	Package	Shipping†
NTR4503NT1G	SOT−23 (Pb−Free)	3000 / Tape & Reel
NVTR4503NT1G	SOT−23 (Pb−Free)	3000 / Tape & Reel

†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specification Brochure, BRD8011/D.

ELECTRICAL CHARACTERISTICS (T_J = 25°C unless otherwise noted)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
OFF CHARACTERISTICS						
Drain−to−Source Breakdown Voltage	$V_{(BR)DSS}$	V_{GS} = 0 V, I_D = 250 μA	30	36		V
Zero Gate Voltage Drain Current	I_{DSS}	V_{GS} = 0 V, V_{DS} = 24 V			1.0	μA
		V_{GS} = 0 V, V_{DS} = 24 V, T_J = 125°C			10	
Gate−to−Source Leakage Current	I_{GSS}	V_{DS} = 0 V, V_{GS} = ±20 V			±100	nA
ON CHARACTERISTICS (Note 3)						
Gate Threshold Voltage	$V_{GS(TH)}$	V_{GS} = V_{DS}, I_D = 250 μA	1.0	1.75	3.0	V
Drain−to−Source On−Resistance	$R_{DS(on)}$	V_{GS} = 10 V, I_D = 2.5 A		85	110	mΩ
		V_{GS} = 4.5 V, I_D = 2.0 A		105	140	
Forward Transconductance	g_{FS}	V_{DS} = 4.5 V, I_D = 2.5 A		5.3		S
CHARGES AND CAPACITANCES						
Input Capacitance	C_{iss}	V_{GS} = 0 V, f = 1.0 MHz, V_{DS} = 15 V		135		pF
Output Capacitance	C_{oss}			52		
Reverse Transfer Capacitance	C_{rss}			15		
Input Capacitance	C_{iss}	V_{GS} = 0 V, f = 1.0 MHz, V_{DS} = 24 V		130	250	pF
Output Capacitance	C_{oss}			42	75	
Reverse Transfer Capacitance	C_{rss}			13	25	
Total Gate Charge	$Q_{G(TOT)}$	V_{GS} = 10 V, V_{DS} = 15 V, I_D = 2.5 A		3.6	7.0	nC
Threshold Gate Charge	$Q_{G(TH)}$			0.3		
Gate−to−Source Charge	Q_{GS}			0.6		
Gate−to−Drain Charge	Q_{GD}			0.7		
Total Gate Charge	$Q_{G(TOT)}$	V_{GS} = 4.5 V, V_{DS} = 24 V, I_D = 2.5 A		1.9		nC
Threshold Gate Charge	$Q_{G(TH)}$			0.3		
Gate−to−Source Charge	Q_{GS}			0.6		
Gate−to−Drain Charge	Q_{GD}			0.9		
SWITCHING CHARACTERISTICS (Note 4)						
Turn−On Delay Time	$t_{d(on)}$	V_{GS} = 10 V, V_{DD} = 15 V, I_D = 1 A, R_G = 6 Ω		5.8	12	ns
Rise Time	t_r			5.8	10	
Turn−Off Delay Time	$t_{d(off)}$			14	25	
Fall Time	t_f			1.6	5.0	
Turn−On Delay Time	$t_{d(on)}$	V_{GS} = 10 V, V_{DD} = 24 V, I_D = 2.5 A, R_G = 2.5 Ω		4.8		ns
Rise Time	t_r			6.7		
Turn−Off Delay Time	$t_{d(off)}$			13.6		
Fall Time	t_f			1.8		
DRAIN−SOURCE DIODE CHARACTERISTICS						
Forward Diode Voltage	V_{SD}	V_{GS} = 0 V, I_S = 2.0 A		0.85	1.2	V
Reverse Recovery Time	t_{RR}	V_{GS} = 0 V, I_S = 2.0 A, dI_S/dt = 100 A/μs		9.2		ns
Reverse Recovery Charge	Q_{RR}			4.0		nC

Product parametric performance is indicated in the Electrical Characteristics for the listed test conditions, unless otherwise noted. Product performance may not be indicated by the Electrical Characteristics if operated under different conditions.

3. Pulse Test: Pulse Width ≤ 300 μs, Duty Cycle ≤ 2%.
4. Switching characteristics are independent of operating junction temperatures.

TYPICAL PERFORMANCE CURVES

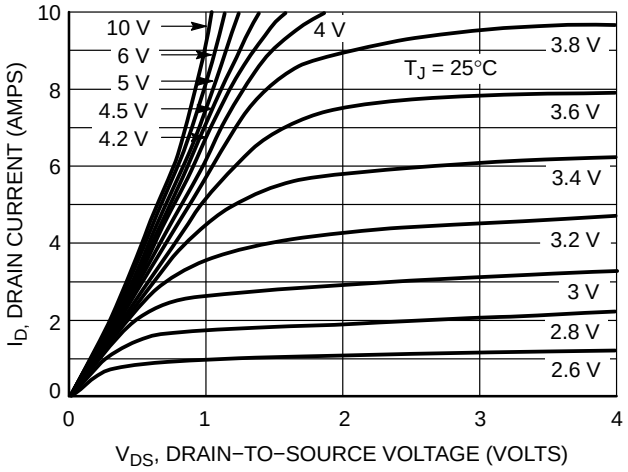


Figure 1. On−Region Characteristics

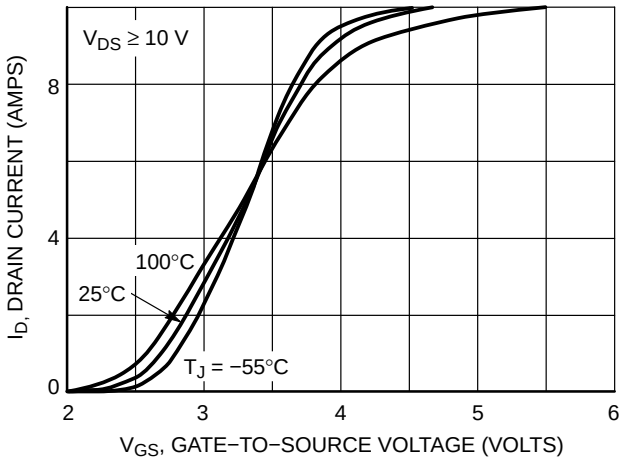


Figure 2. Transfer Characteristics

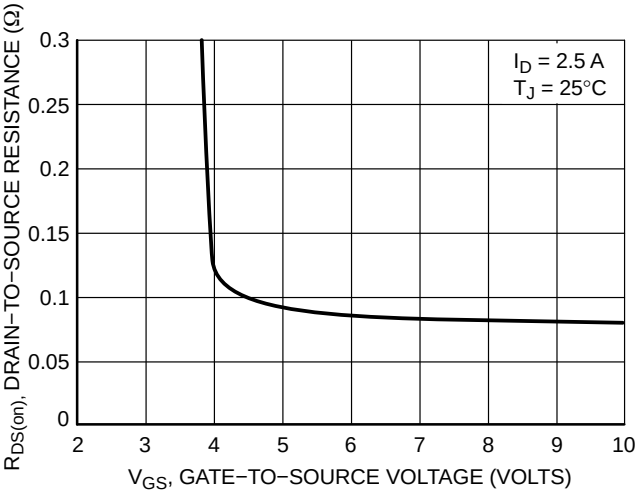


Figure 3. On−Resistance vs. Gate−to−Source Voltage

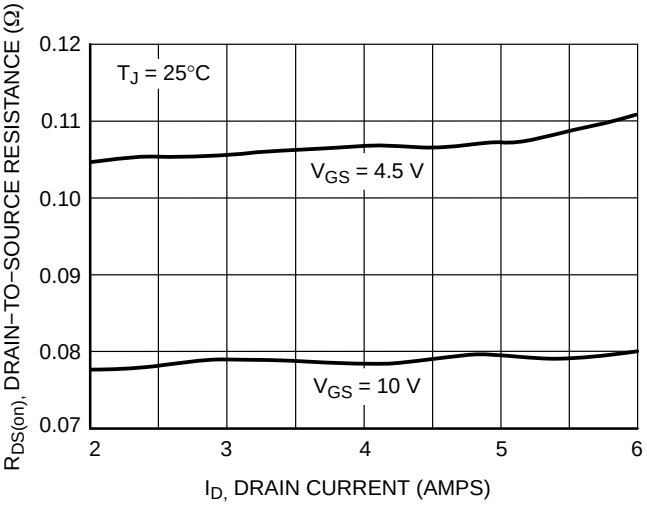


Figure 4. On−Resistance vs. Drain Current and Gate Voltage

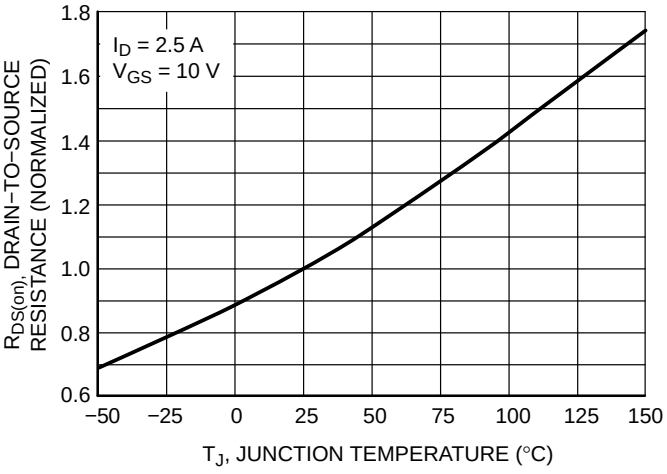


Figure 5. On−Resistance Variation with Temperature

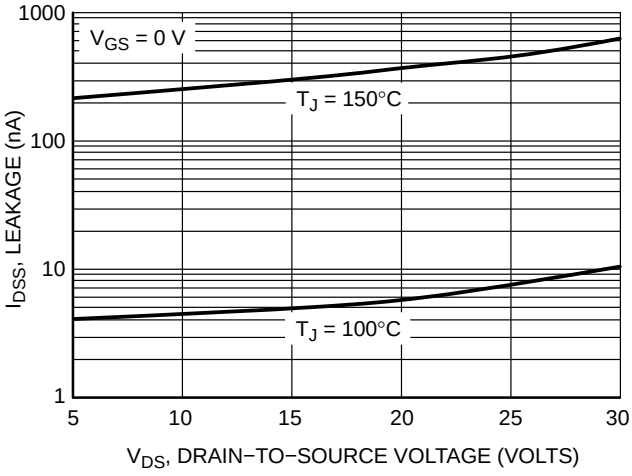


Figure 6. Drain−to−Source Leakage Current vs. Voltage

TYPICAL PERFORMANCE CURVES

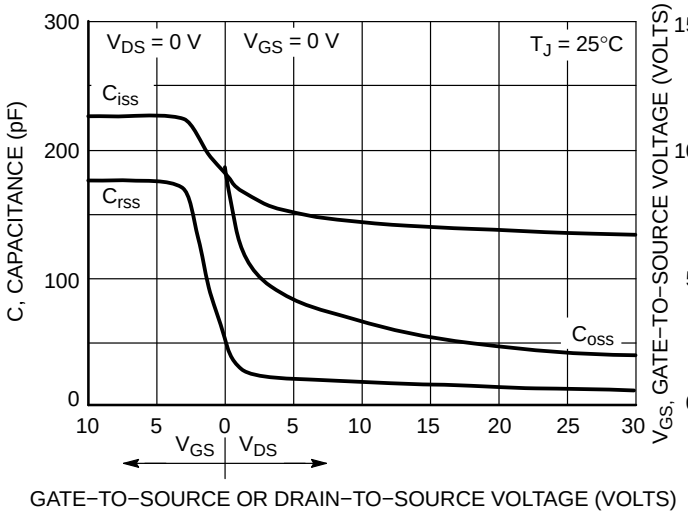


Figure 7. Capacitance Variation

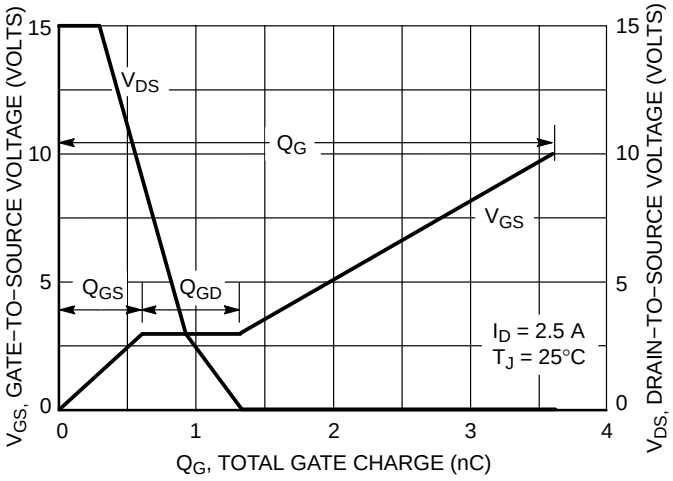


Figure 8. Gate−to−Source and Drain−to−Source Voltage vs. Total Charge

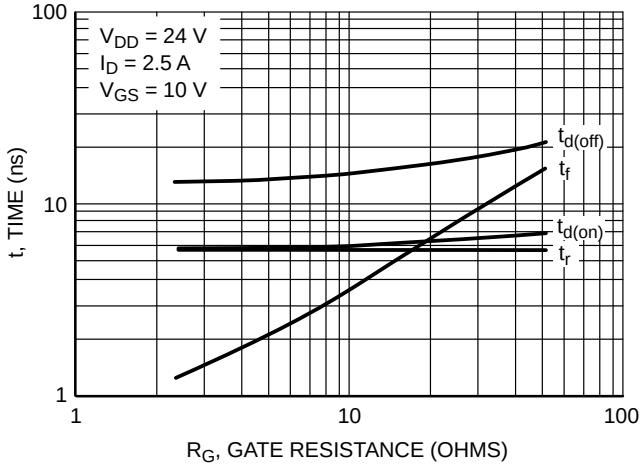


Figure 9. Resistive Switching Time Variation vs. Gate Resistance

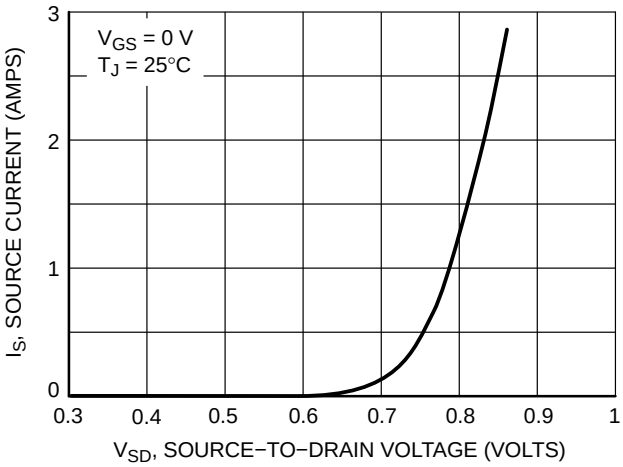
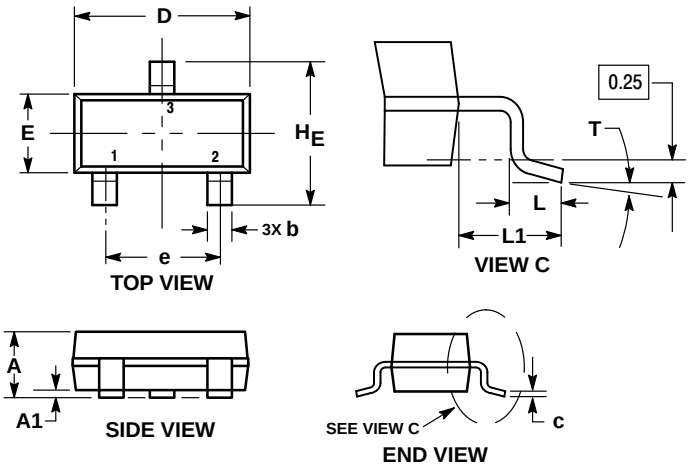


Figure 10. Diode Forward Voltage vs. Current

PACKAGE DIMENSIONS

SOT−23 (TO−236)
CASE 318−08
ISSUE AR

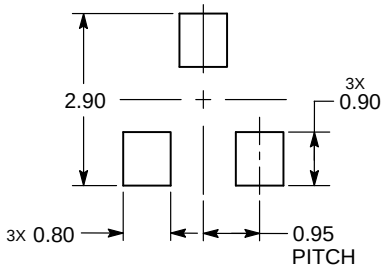


NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. CONTROLLING DIMENSION: MILLIMETERS.
3. MAXIMUM LEAD THICKNESS INCLUDES LEAD FINISH. MINIMUM LEAD THICKNESS IS THE MINIMUM THICKNESS OF THE BASE MATERIAL.
4. DIMENSIONS D AND E DO NOT INCLUDE MOLD FLASH, PROTRUSIONS, OR GATE BURRS.

	MILLIMETERS			INCHES		
DIM	MIN	NOM	MAX	MIN	NOM	MAX
A	0.89	1.00	1.11	0.035	0.039	0.044
A1	0.01	0.06	0.10	0.000	0.002	0.004
b	0.37	0.44	0.50	0.015	0.017	0.020
c	0.08	0.14	0.20	0.003	0.006	0.008
D	2.80	2.90	3.04	0.110	0.114	0.120
E	1.20	1.30	1.40	0.047	0.051	0.055
e	1.78	1.90	2.04	0.070	0.075	0.080
L	0.30	0.43	0.55	0.012	0.017	0.022
L1	0.35	0.54	0.69	0.014	0.021	0.027
H_E	2.10	2.40	2.64	0.083	0.094	0.104
T	0°	−−−	10°	0°	−−−	10°

STYLE 21:
PIN 1. GATE
2. SOURCE
3. DRAIN

RECOMMENDED SOLDERING FOOTPRINT*



DIMENSIONS: MILLIMETERS

*For additional information on our Pb−Free strategy and soldering details, please download the ON Semiconductor Soldering and Mounting Techniques Reference Manual, SOLDERRM/D.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910
Japan Customer Focus Center
Phone: 81−3−5817−1050

ON Semiconductor Website: **www.onsemi.com**

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative

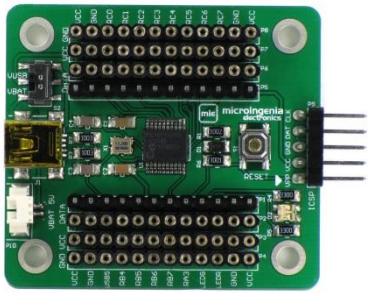
18F14K50Trainer: Módulo Entrenador basado en el microcontrolador PIC 18F14K50 de Microchip

1. Descripción del producto

Este entrenador (*Trainer*) está basado en el microcontrolador (*Microcontroller*) PIC 18F14K50 de Microchip y se ha diseñado para facilitar la implementación rápida de prototipos (*Rapid proptotyping*) de sistemas de control y comunicaciones.
Sus principales características son:

- Posee todos los componentes electrónicos adecuados para alimentarlo a través del bus USB. Además dispone de un conector ICSP para poder conectarlo al grabador/depurador Pickit 2/3 de Microchip.
- Dispone de 9 entradas analógicas y 17 entradas/salidas digitales.
- Posee un terminal (*pin*) de detección de conexión USB, un pulsador de inicialización (*Reset*) y un diodo luminiscente (*LED*) bicolor configurable. Los terminales macho verticales facilitan la conexión del entrenador a los módulos de desarrollo con los cables de prototipo, y los terminales hembra hacen posible la conexión de resistencias de drenador (*pull-up*) y de surtidor (*pull-down*) fácilmente.
- Tiene instalado un programa cargador (*Bootloader*) USB CDC lo que hace innecesaria la utilización de un grabador.

Puedes descargar nuestro Bootloader USB CDC CCS y programas fuente de ejemplo desde su página de producto.



2. Características generales

- PIC 18F14K50
- 48MHz CPU
- 16KB de FlashROM
- 256Bytes de EEPROM
- 768B de RAM
- 17 Entradas/Salidas Digitales
- 9 Entradas Analógicas
- Conector ICSP para conexión con el grabador/depurador Pickit 2/3

3. Especificaciones

- Alimentación: 5 VCC (USB o VBat)
- Dimensiones: 50,8 x 43,18mm (2'' x 1.7'')
- Peso: 15,9g

NOTA: El esquema (*Schematic*), y el programa del ejemplo se pueden descargar desde la página del producto (18F14K50Trainer) a través de www.microingenia.com

4. Descripción de los terminales

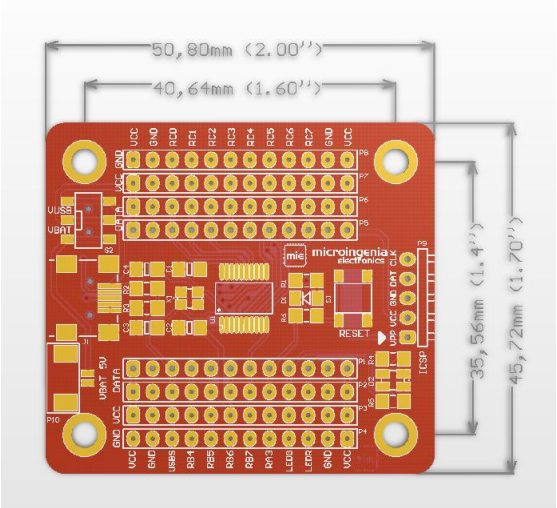


Figura 1.- Dimensiones 18F14K50Trainer

Terminal	Descripción
VCC	Terminal de alimentación
GND	Terminal de masa
USBS	Terminal de detección de conexión USB
RB0-RB7	Terminales de E/S del puerto B
RA3	Terminal de E/S del puerto A
LEDG	Terminal de activación led verde
LEDR	Terminal de activación led rojo
VPP	Terminal de reset
DAT	Terminal de entrada de datos de programación
CLK	Terminal de entrada de reloj para programación
RC0-RC7	Terminales E/S del puerto C

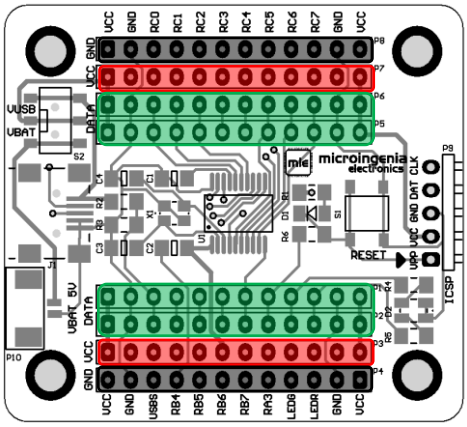


Figura 2.- Entradas / Salidas 18F14K50Trainer

Tira	Descripción
GND	Conectores Hembra conectados a masa
VCC	Conectores Hembra conectados a VCC
DATA	Conectores Hembra/Macho asociados a las Entradas/Salidas

5. Esquema

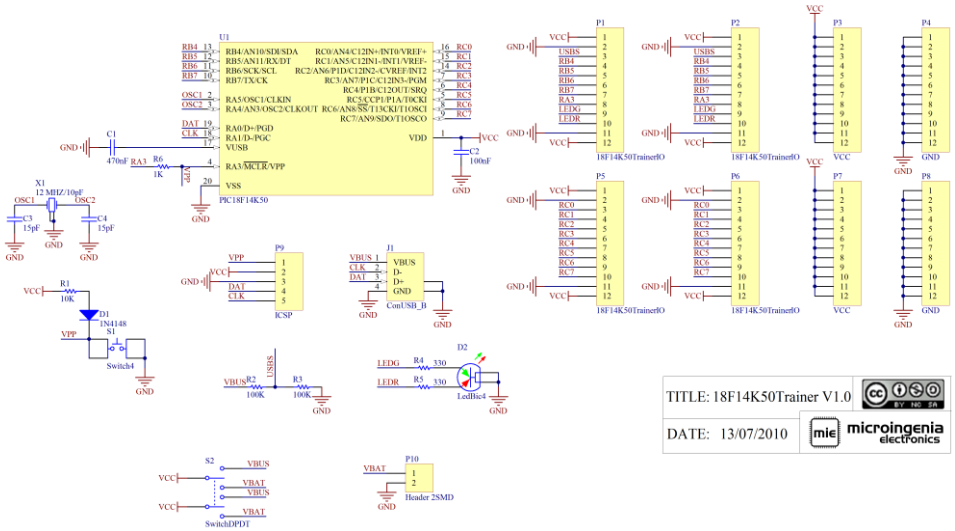


Figura 3.- Esquema del entrenador 18F14K50Trainer

- **Componentes principales:**
  - **U1:** PIC18F14K50, Microcontrolador PIC 18F14K50 de Microchip
  - **J1:** ConUSB, Conector mini USB tipo B
  - **D2:** LedBic4, Diodo luminiscente (*LED*) bicolor (rojo/verde)
  - **S1:** Switch4, Pulsador de inicialización
  - **S2:** Switch4, Selector de tipo alimentación, USB o Batería
  - **P10:** Header 2SMD, Conector de batería

6. Grabación de un programa (.HEX)

El entrenador se suministra con el programa cargador (*bootloader*) preinstalado, para no tener que utilizar un grabador externo y facilitar así la carga de los programas desarrollados en un computador.

Para cargar los programas mediante un grabador externo se utiliza el conector ICSP, utilizado por los grabadores PICkit 3 y PICkit 2 de Microchip.

Para utilizar el programa cargador, o bien, para usar el grabador PICkit 2/3, se recomienda consultar el manual de referencia “Manual Bootloader USB CDC CCS 18F14K50Trainer V1.0”. Descargable desde la página web de producto de la entrenadora (18F14K50Trainer), en la pestaña documentos, sección Bootloader.

7. Ejemplo 1

- **Objetivo**

El ejemplo consiste en un simple oscilador (*Toggle*) de un período de un segundo en los terminales del entrenador. Se debe conectar uno de los extremos del cable de prototipo (hembra-hembra) al terminal “LedG” y el otro, a uno de los terminales de entrada/salida del entrenador para que cambie de estado cada segundo.

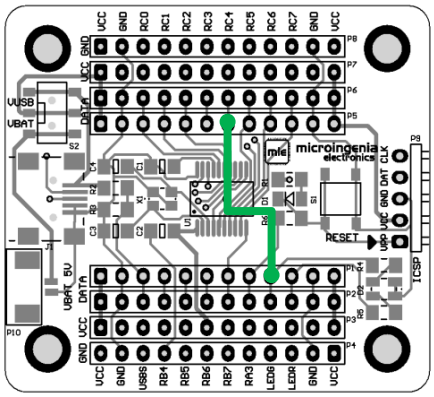


Figura 4.- Ejemplo de conexión 18F14K50Trainer

- **Programa de Aplicación en C para el compilador CCS PCWHD Versión 4.108**

Ejemplo "Ejemplo1 V1.0, CCS C (E1_18F14K50Trainer_CCS_V1.0.zip)" con la función oscilación de salida (*output_toggle*) de CCS. Se utiliza el temporizador 0 (*timer 0*) mediante la función inicialización del temporizador (*setup_timer0*), programado con el valor adecuado para que se produzca el desbordamiento (*Overflow*) de su contenido cada segundo.

Se inicializa mediante la función "set_timer0" con el valor de 1 segundo y se ejecuta la función "isr_timer0", asociada a la interrupción de desbordamiento del temporizador 0 (#int_timer0).

El fichero "config18F14K50Trainer.c" contiene la configuración del entrenador, donde se incluye el microcontrolador utilizado, los fuses necesarios para su funcionamiento (configurables por el usuario según sus necesidades) y el reloj.

"Config18F14K50Trainer.c"

```
#include <18F14K50.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,NODEBUG,NOBROWNOUT,USBDIV1,PLLEN,CPUDIV1,PUT,MCLR
#use delay(clock=48000000)
```

"Main.c"

```
#include "config18F14K50Trainer.c"

#include "usb/usb_bootloader.h"
#include "usb/usb_cdc.h"

void main(void)
{
   disable_interrupts(GLOBAL);
   disable_interrupts(INT_TIMER0);

   setup_timer_0(RTCC_DIV_128);
   set_timer0(18661);

   enable_interrupts(GLOBAL);
   enable_interrupts(INT_TIMER0);

   while(TRUE);
}

#int_timer0
void isr_timer0(void)
{
   // Toggle each PIN every second
   set_timer0(18661);


   output_toggle(PIN_A3);
   output_toggle(PIN_B4);
   output_toggle(PIN_B5);
   output_toggle(PIN_B6);
   output_toggle(PIN_B7);
  // output_toggle(PIN_C0);
   output_toggle(PIN_C1);
```

```
   output_toggle(PIN_C2);
   output_toggle(PIN_C3);
   output_toggle(PIN_C4);
   output_toggle(PIN_C5);
   output_toggle(PIN_C6);
   output_toggle(PIN_C7);
}
```

7. Referencias

7.1. Programas de cálculo

- PIC C Compiler Versión CCS PCWHD v4.13: Programa utilizado para la compilación en lenguaje C de todo el proyecto.
- Project Libre: Programa usado para gestionar y planificar el proyecto.
- Microsoft Office Excel: Para el estudio de los presupuestos y demás cuentas.
- Microsoft Office: Desarrollo de la memoria.
- Orcad 9.2: Es un programa para el diseño de placas y esquemáticos electrónicos mediante el cual se crea el circuito eléctrico del producto y los PCBs necesarios.
- Visio 2013: Programa para diseñar estructuras de desglose de tareas.
- MPLAB X IDE v3.26: Programación del PIC de Microchip.
- Realterm: Programa de terminal serie para la ingeniería. Ideal para el desarrollo , la ingeniería inversa , depuración, registro de datos y la captura , y la prueba automática.
- Zotero: Programa para la recopilación de referencias y citas para el desarrollo de la bibliografía.
- LabView: Captura y visualización de datos.
- Frhed: Visualización de datos en formato hexadecimal.

7.2. Bibliografía

Estado del arte

Fisiología ocular

- «Alcmeón - Revista Argentina de Clínica Neuropsiquiátrica - Los movimientos oculares en la práctica neuropsiquiátrica». Accedido 27 de febrero de 2016. http://www.alcmeon.com.ar/7/28/alc28_01_02.htm.

- «Anatomía — AAPOS». Accedido 27 de febrero de 2016. http://www.aapos.org/es/terms/conditions/22.

- «conti_fm_1e_cap_muestra.pdf». Accedido 27 de febrero de 2016. http://highered.mheducation.com/sites/dl/free/9701073414/817952/conti_fm_1e_cap_muestra.pdf.

- «Córnea - Wikipedia, la enciclopedia libre». Accedido 27 de febrero de 2016. https://es.wikipedia.org/wiki/C%C3%B3rnea.

- «Cristalino - Wikipedia, la enciclopedia libre». Accedido 27 de febrero de 2016. https://es.wikipedia.org/wiki/Cristalino.

- «El Blog de Neuro-Estrabismo y Optometria Comportamental: La Fisiología de los MovimientosOculares». Accedido 26 de febrero de 2016. http://miestrabismo.blogspot.com.es/2015/04/la-fisiologia-de-los-movimientos.html.

- «El ojo: MedlinePlus enciclopedia médica illustración». Accedido 26 de febrero de 2016. https://www.nlm.nih.gov/medlineplus/spanish/ency/esp_imagepages/1094.htm.

- «Fisiologia de Los Movimientos Oculares - Documents». Accedido 26 de febrero de 2016. http://documents.tips/documents/fisiologia-de-los-movimientos-oculares.html.

- «Fundamentos de visión binocular - Francisco M. Martínez Verdú, Álvaro M. Pons Moreno - GoogleLibros». Accedido 8 de marzo de 2016. https://books.google.es/books?id=tP_bvpI47DwC&pg=PA23&lpg=PA23&dq=cinematica+del+ojo&source=bl&ots=I4Dsc6zdir&sig=fdUC3DQUlQ9kPA_1xR09YTedk10&hl=es&sa=X&ved=0ahUKEwiv0Jf7yrHLAhUK0xoKHZ46CdkQ6AEIHDAA

Sistemas de seguimiento ocular.

- «B-EyeCan- Adaptación de EyeWriter para Escribir y Navegar - Plataforma Afectados de ELA-Compendio de Información». Accedido 1 de marzo de 2016. https://sites.google.com/site/plataformadeafectadosela/dispositivos/eyecan.

- «Bioinstrumentacion: Electrooculograma». Accedido 2 de marzo de 2016. http://bioingenieria5.blogspot.com.es/2013/12/electrooculograma.html.

- «Camino de Tierra, Oscar Godoy: Proyecto Eye Writer». Accedido 1 de marzo de 2016. http://oscargozum.blogspot.com.es/2015/04/proyecto-eye-writer.html.

- «EyeCAN : une souris oculaire par Samsung». Accedido 6 de octubre de 2016. http://www.journaldugeek.com/2012/03/03/eyecan-une-souris-oculaire-par-samsung/.

- «Eye Tracking – Seguimiento ocular». Accedido 1 de marzo de 2016. http://neuromarca.com/neuromarketing/eye-tracking/.

- «EyeWriter». Accedido 6 de octubre de 2016. http://www.eyewriter.org/.

- «GitHub - OptiKey/OptiKey: OptiKey - Full computer control and speech with your eyes». Accedido 6 de octubre de 2016. https://github.com/OptiKey/OptiKey.

- «iMotions: Advanced Human Behavior Research Software, Simplified». Accedido 6 de octubre de 2016. https://imotions.com/.

- «OptiKey, la aplicación de seguimiento de los ojos que plantea un futuro mejor para los enfermos de ELA». Accedido 1 de marzo de 2016. http://www.xataka.com/medicina-y-salud/optikeyla-aplicacion-de-seguimiento-de-los-ojos-que-plantea-un-futuro-mejor-para-los-enfermos-de-ela.

- «Stephen Hawking: "La medicina no me curó, así que me apoyo en la tecnología para comunicarme y vivir"». Accedido 1 de marzo de 2016. http://www.xataka.com/n/stephen-hawking-la-medicinano-me-curo-asi-que-me-apoyo-en-la-tecnologia-para-comunicarme-y-vivir.

- Accedido 1 de marzo de 2016. http://hipertextual.com/2009/08/eyewriter-proyecto-de-codigo-librepara-dibujar-con-los-ojos.

Desarrollo software

- «2_Converdidor_AD_PIC16F87X.pdf». Accedido 27 de julio de 2016. http://profesores.sanvalero.net/~arnadillo/Documentos/Apuntes/Prototipo/UD4_PIC16F87X/2_Converdidor_AD_PIC16F87X.pdf.

- «15. PWM - (Ancho por Modulación de Pulso) | Control Automatico Educacion». Accedido 28 de septiembre de 2016. http://controlautomaticoeducacion.com/15-pwm-ancho-por-modulacionde-pulso/.

- «Apuntes Informática / Electrónica - Comunicación USB PIC18F4550 utilizando la clase CDC». Accedido 28 de junio de 2016. http://www.aquihayapuntes.com/indice-practicas-pic-enc/comunicacion-usb-pic18f4550-utilizando-la-clase-cdc.html.

- «El Fichero .HEX explicado». Accedido 28 de junio de 2016. http://www.todopic.com.ar/foros/index.php?topic=23342.0.

- «Intel HEX - Wikipedia, la enciclopedia libre». Accedido 28 de junio de 2016. https://es.wikipedia.org/wiki/Intel_HEX.

- «Modulación por ancho de pulsos - Wikipedia, la enciclopedia libre». Accedido 26 de septiembre de 2016. https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos.

- Accedido 28 de junio de 2016. http://picmania.garcia-cuervo.net/usb_3_cdctransfers.php.

Otros

- «Analisis de Riesgos en operaciones de reparacion de PCB`S.pdf». Accedido 10 de mayo de 2016. https://www.dspace.espol.edu.ec/bitstream/123456789/11795/1/Analisis%20de%20Riesgos%20en%20operaciones%20de%20reparacion%20de%20PCB%60S.pdf.

- «Beatriz Mayoral: Radiación infrarroja en el ojo». Accedido 19 de mayo de 2016. http://beatrizmayoral.blogspot.com.es/2013/12/radiacion-infrarroja-en-el-ojo.html.

- «La radiación infrarroja: un viejo enemigo desconocido del ojo humano (Sponsor) - Longitud de Onda». Accedido 19 de mayo de 2016. http://www.longitudeonda.com/index.php/la-radiacion-infrarrojaenemigo-desconocido-del-ojo/.

- «Normativa.pdf». Accedido 10 de mayo de 2016. http://riubu.ubu.es/bitstream/10259/3589/1/Normativa.pdf.

- «pfc5764.pdf». Accedido 24 de junio de 2016. http://repositorio.upct.es/bitstream/handle/10317/4054/pfc5764.pdf?sequence=1.

- «resumennorma157001.pdf». Accedido 3 de mayo de 2016. http://www.ehu.eus/asignaturasKO/PM/General/resumennorma157001.pdf.

- «Revelado y atacado de nuestros circuitos impresos PCB - Carlitosspedia». Accedido 28 de junio de 2016. http://www.carlitosspedia.info/revelado-y-atacado-de-nuestros-circuitos-impresos-pcb/.

- «T2_caa.pdf». Accedido 26 de mayo de 2016. http://www2.uca.es/grupinvest/instrument_electro/ppjjgdr/Cir_An_Apl/Cir_An_Apl_arch/temas/T2_caa.pdf.