ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

TRABAJO FIN DE GRADO

SISTEMA DE MEDICIÓN DE PERÍMETROS LÁSER

TITULACIÓN:

GRADO EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ALUMNO: Isidro Jesús Castro García.

TUTOR: Jonay Toledo Carrillo.

SEPTIEMBRE 2014.

Índice

0. Resumen/Abstract

0.1. Resumen:

En el presente documento del trabajo fin de grado del Grado en Ingeniería electrónica Industrial y Automática, se describe la confección de un sensor de distancias mediante el empleo de sensores ópticos CCD lineales.

En el presente documento se recogen una descripción de todos los fundamentos teóricos que son necesarios para entender el funcionamiento del sensor, así como una descripción detallada de todos los procedimientos que se han realizado en el laboratorio a la hora de confeccionar el prototipo.

Además se incluye una descripción de los códigos empleados en la fabricación de la placa, siendo incluidos estos además en los anexos del presente documento.

Se ha dedicado un capítulo del documento a realizar una demostración matemática de la influencia que dos parámetros tienen en las distancias medibles y la resolución, incluyéndose además dos ejemplos numéricos que ayudan a elegir los parámetros deseados una vez conocidos los requerimientos de medida que se necesitan.

Otro apartado es dedicado a los resultados que se han obtenido de los sensores. En dicho capítulo se han adjuntado gráficas con los datos experimentales recogidos en el laboratorio, además de los cálculos de las distancias, que se han realizado con el software de cálculo Matlab.

También se han descrito los problemas que han surgido en el proceso de experimentación con los sensores así, así como las medidas adoptadas para intentar solventarlos.

Finalmente se ha añadido una conclusión acerca de todo lo trabajo, los resultados obtenidos y los conocimientos y habilidades que se han desarrollado en el transcurso del trabajo.

0.2. Abstract:

This document belongs to the finally grade work of Electronic Engineering, and it describes the making of a distance measuring sensor, using CCD lineal image sensors.

Herein we can find a description of all theorical fundaments necessary to understand the motion of the sensor. It also contains all the processes that were done in the laboratory to make the prototype. The codes used in the experiments are included in the annex of the document.

In one chapter of the document there is a description of the influence of two parameters of the prototype in the measuring and the resolution of the sensor. The distance operation has been made with Matlab.

It also has a description of all the problems that have appear during de experimentation, and the decisions taken to avoid them.

Finally we can find the conclusion of the work and the valuations of the results of the experimentation, the knowledge learned and the skills developed during the work.

1. Introducción

El presente documento constituye la memoria del la asignatura de carácter anual trabajo fin de grado perteneciente al cuarto curso del plan de estudios del Grado en Ingeniería Electrónica Industrial y Automática. Dicho trabajo ha sido tutorizado por el profesor Jonay Toledo, que propuso los contenidos del presente trabajo.

1.1. Aspectos generales del proyecto:

1.1.1. Objetivos

1.1.1.1. Objetivos técnicos

En la presente memoria se describe una propuesta técnica que propone el desarrollo de un sensor láser para robot capaz de medir distancias mediante el empleo Arduino. Para ello se ha haráempleo de un sensor CCD lineal, el TDC1201D que por sus características, las cuales se describirán más adelante, lo hacen apropiado para aplicaciones en las que se desee realizar el cálculo de distancias. El control del sensor se realizará mediante la ayuda de un Arduino que será el elemento que generará los pulsos de entrada necesarios para el correcto funcionamiento del sensor. A su vez será el encargado de generar leer las salidas proporcionadas por el sensor y convertirlas a formato digital para su correcto procesamiento.

1.1.1.2. Objetivos personales

Los objetivos personales perseguidos con el desarrollo del presente trabajo fin de grado son los de adquirir los conocimientos necesarios en el desarrollo de un proyecto acerca de un tema que guarda una estrecha relación con los estudios cursados en el Grado en Ingeniería Electrónica Industrial y Automática. Se espera obtener experiencia en el aprendizaje autodidacta que se prevé un aspecto importante para conseguir desarrollar el trabajo.

También se espera aprender a utilizar una plataforma tan importante en la electrónica moderna como viene a ser Arduino. El desarrollo del presente trabajo permitirá realizar una primera toma de contacto con esta plataforma, además de ir conociendo la infinidad de posibilidades que ofrece en el mundo de la electrónica, la robótica y la automatización.

La programación de diferentes programas de ARDUINO y la ejecución de los mismos en un circuito analógico, suponen una experiencia nueva que no se ha obtenido a lo largo de la

carrera, y que por lo tanto se espera que aporte nuevos conocimientos acerca de este campo tan útil en la industria moderna.

Además se persigue el objetivo académico de superar los 12 créditos de la asignatura Trabajo fin de grado, correspondiente al plan de estudios del ya citado grado.

1.1.2. Alcance:

El alcance del presente proyecto abarcará todos los procedimientos necesarios en la realización del prototipo.

Se expondrán los diferentes conceptos teóricos que hacen rigen el funcionamiento del mismo, así como los modelos matemáticos y físicos que permiten su viabilidad. Se incluye una descripción básica de algunas características comunes entre los diferentes instrumentos de medida que se encuentran en el mundo industrial, características que compartirá el prototipo considerado en el presente documento.

Los diferentes equipos y materiales utilizados en el desarrollo del proyecto aparecerán descritos en la presente memoria. Además se han adjuntado en el capítulo de ANEXOS las hojas de datos de los elementos más importantes; y en la cual aparece información importante sobre el estado de las conexiones o los requerimientos de alimentación y ondas de entrada necesarias.

Posteriormente se realizará una descripción detallada de los pasos seguidos durante la experimentación llevada a cabo en el laboratorio. En ella se expondrán el circuito montado, así como las diferentes pruebas que se han realizado para controlar el sensor mediante ARDUINO. En el mismo se expondrán las dificultades y problemas surgidos, así como las soluciones adoptadas para solventarlos.

También se detallará el código realizado para el procesamiento de los datos obtenidos del sensor, así como el cálculo realizado para la obtención de la distancia.

1.1.3. Peticionario:

El peticionario del presente documento es la asignatura Trabajo Fin de Grado, que en su guía docente lo indica como requisito ineludible para superar la asignatura.

1.1.4. Emplazamiento:

Toda la experimentación llevada a cabo en el presente trabajo se ha realizado en el laboratorio de Computadoras y Control del edifico de la facultad de Física y Matemática, perteneciente al campus de Anchieta de la universidad de la Laguna.

1.2. Introducción al fotodiodo:

El sensor CCD lineal basa su principio de funcionamiento en un conjunto de 2048 fotodiodos colocados consecutivamente ordenados en fila, y que detectan la luz que les incide, transformando ésta en un voltaje de salida proporcional. El ser capaz de transformar una medida de luz en un voltaje eléctrico convierte al fotodiodo un elemento interesante en diversas aplicaciones de robótica e imagen digital.

Estos diodos presentan una excelente linealidad con respecto a la luz incidente, un bajo nivel de ruido (en el caso del TCD1201D el escaso ruido que se pueda producir se corrige mediante el uso de la salida DOS como ya se explicará más adelante) y un amplio ancho espectral. Además son ligeros y compactos, lo cual facilita su inclusión dentro del integrado.

Esta particularidad hace del CCD lineal un elemento ideal para ser utilizado como un dispositivo detector de luz, pues convierte la luz en voltaje, y ésta variación de electricidad es utilizada para informar de cambios en el nivel de iluminación del fotodiodo. Por lo tanto el diodo sobre el cual incida el haz de luz del láser poseerá un valor de corriente diferente al resto de diodos. El procesamiento de esta información será la que permitirá determinar la distancia del objeto respecto al sensor.

Es común que los fotodiodos vengan equipados con una lente que concentre la cantidad de luz que le incide, lo cual hace que su reacción a la luz sea más evidente.

Una gran ventaja que ofrece el fotodiodo respecto a otro elemento medidor de luz como el LDR o fotorresistencia, es que el fotodiodo presenta una mayor velocidad de respuesta frente a cambios de oscuridad a iluminación y viceversa. Por lo tanto se puede utilizar en aplicaciones que requieren un tiempo de respuesta más pequeño.

1.2.1. Funcionamiento de un fotodiodo

La estructura e un fotodiodo es similar a un diodo de propósito general, al igual que estos se encuentra compuesto por una unión PN que posee la particularidad de ser sensible a la incidencia de luz visible o infrarrojos. Por lo tanto el fotodiodo presenta múltiples similitudes con un diodo semiconductor común, pero con la salvedad de que es capaz de conducir una corriente eléctrica proporcional a la cantidad de luz que le incide.

Para entender el principio básico de operación de un fotodiodo es importante saber el comportamiento que presentan los electrones en un material semiconductor.

Un material semiconductor es un elemento con propiedades eléctricas entre las de un conductor y un aislante. Suelen caracterizarse por tener cuatro electrones de valencia.

Cuando los átomos de un material semiconductor como por ejemplo el silicio se combinan para formar un sólido, lo hacen según un patrón ordenado denominado cristal. Cada átomo comparte sus electrones con cuatro átomos vecinos, de tal forma que tiene ocho electrones en su orbital de valencia.

Los fonones producidos por la energía térmica de los átomos del cristal semiconductor hacen que estos vibren. Las vibraciones de los átomos ocasionalmente pueden hacer que se desligue un electrón del orbital de valencia. Cuando esto ocurre, el electrón liberado gana la energía suficiente como para pasar a un orbital de mayor nivel energético, la denominada banda de conducción. Los electrones que se encuentran en esta banda no son atraídos por los núcleos, por lo que un campo eléctrico externo puede moverlos, formando una corriente continua. Cuando un electrón pasa a la banda de conducción en el cristal se queda un hueco que aporta un exceso de carga positiva.

Para aumentar el número de portadores de carga libres (electrones y huecos), se suele dopar el semiconductor, aumentando así su conductividad. Existen dos posibles tipos de semiconductores dopados:

- Tipo p (con huecos como portadores mayoritarios): Para conseguir un material semiconductor un exceso de huecos se utiliza una impureza trivalente, es decir, cuyos átomos tengan sólo tres electrones de valencia. Al realizar el enlace covalente con los átomos vecinos únicamente habrán siete electrones en la banda de valencia, lo que deja un exceso de un hueco libre por cada átomo de impureza trivalente añadido al cristal.

- Tipo n (con electrones como portadores mayoritarios): Para conseguir un material semiconductor un exceso de electrones se utiliza una impureza pentavalente, es decir, cuyos átomos tengan cinco electrones de valencia. Al realizar el enlace covalente con los átomos vecinos se comparten cuatro de los cinco electrones de su banda de valencia, quedando un exceso de un electrón libre por cada átomo de impureza pentavalente añadido al cristal.

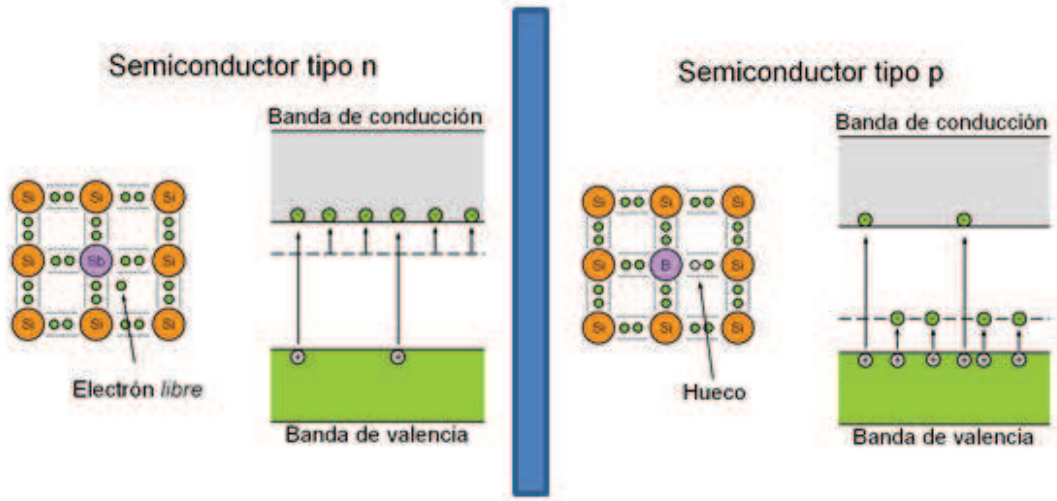

Figura 1.1: Bandas de energía de los semiconductores tipo p y n (Adaptado de https://thetuzaro.wordpress.com/tag/semiconductores-tipo-n/)

Cuando se produce la unión de ambos cristales semiconductores los electrones se ven atraídos eléctricamente hacia la zona p y los huecos a la zona n. Cuando un electrón cruza la unión se recombina con un hueco. Como resultado se forman dipolos en la unión y aparece una barrera de potencial situada en valores próximos a los 0.7V que impide el paso de electrones desde la

banda n hacia la p y huecos desde la zona p a la n. La región entre ambos cristales compuesta por los dipolos que se han formado se denomina zona de deplexión.

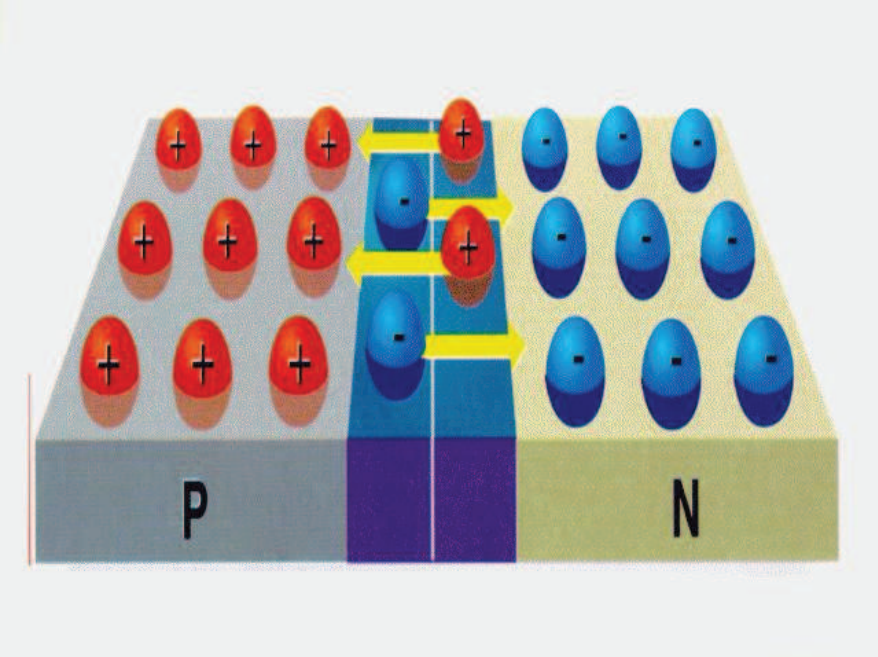

Figura 1.2: Unión pn (http://kerchak.com)

Para que se pueda producir una corriente continua en el elemento conductor es necesario excitar eléctricamente los electrones con la energía suficiente para que puedan atravesar la barrera de potencial surgida en la zona de deplexión. Existen dos formas de polarizar el diodo mediante el uso de una pila: polarización directa y polarización inversa.

Si se polariza un fotodiodo en directa (uniendo el terminal positivo de la pila con la región p y el terminal negativo con la región n) con una tensión superior a la barrera de potencial de la zona de deplexión, los electrones libres de la región n fluyen a través de la unión, creándose así una corriente continua independiente de los efectos de la luz incidente en el fotodiodo. Por lo tanto no se consigue una respuesta proporcional a la cantidad de luz que incide en el fotodiodo. En este caso el fotodiodo se comportaría como un diodo de uso general. En la figura 1.3A se puede observar este fenómeno

La polarización inversa (unir el terminal negativo de la batería a la región p y el terminal positivo a la región n) si permite un correcto funcionamiento del fotodiodo. Cuando un fotodiodo es polarizado en inversa se produce un ensanchamiento en la zona de deplexión, debido a las fuerzas de atracción producidas entre los portadores mayoritarios y los terminales

de la pila. En la zona de deplexión cuando un electrón adquiere la energía suficiente para liberarse de la fuerza de atracción del núcleo atómico se forma una par electrón libre-hueco. Esta energía es denominada como gap y es la energía necesaria para que un electrón pase de la banda de valencia a la banda de conducción. Esta energía puede ser adquirida de diversas formas, sin embargo en los fotodiodos esta energía la adquieren los electrones mediante los fotones que inciden en la unión. Cuando sucede este fenómeno, el electrón es "empujado" por el potencial eléctrico de la unión hacia la región n. Esto hace que un electrón se vea forzado a abandonar el cristal por el cable unido a la región n. Como se forman continuamente pares electrón- hueco, y la formación de estos es proporcional a la cantidad de luz que incide en el cristal, se forma una corriente continua proporcional a la cantidad de luz que incide en el cristal, denominada corriente fotovoltaica. Al ser esta corriente proporcional a la cantidad de luz incidente en el cristal, se consigue el efecto deseado con los fotodiodos, ser capaz de medir la cantidad de luz que incide en el cristal. En la figura 1.3B se puede observar el comportamiento de un fotodiodo polarizado en inversa.

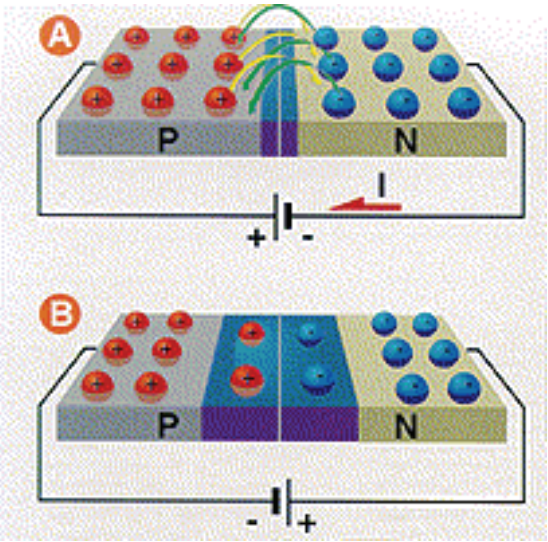

Figura 1.3: Polarización de los semiconductores (http://kerchak.com)

Los CCD lineales utilizados en el presente proyecto se caracterizan por tener 2048 de estos fotodiodos conectados en serie uno al lado de otro. Estos sensores permiten leer la cantidad de luz que recibe cada uno de ellos, propiedad que ha sido utilizada para poder medir las distancias como se explicará a continuación. Estos fotodiodos se encuentran situados dentro de un encapsulado protegidos con un cristal transparente que permite el paso de la luz.

2. Materiales:

En el presente apartado se describen los diferentes componentes electrónicos que han sido necesarios en la experimentación llevada a cabo en el trabajo fin de grado.

Para la puesta en funcionamiento del CCD lineal se ha hecho uso del siguiente conjunto de elementos:

- Placa ARDUINO
- 1 Condensador electrolítico
- 1 Sensor óptico lineal CCD TCD1201D
- 1 Sensor óptico lineal CCD ILX511
- 2 Resistencias 2,2kΩ
- 4 Resistencias 150Ω

2.1. ARDUINO

2.1.1. Introducción e historia

ARDUINO se trata de una plataforma de prototipos de electrónica de código abierto (open – source), creada en el año 2005 por el estudiante Massimo Banzi, con el objetivo de fabricar una placa de microcontroladores económica para los estudiantes de computación y electrónica de su instituto, consiguiendo así economizar la creación de los proyectos escolares del instituto. Está pensada para ser usada por diseñadores, estudiantes de computación o robótica, artistas o simplemente por hobby.

Está compuesta por una placa principal en la cual se encuentra insertado el microcontrolador, así como el resto de controladores y componentes electrónicos que la componen. Esta placa facilita mucho el trabajo con microcontroladores, ya que viene preparada con los componentes necesarios para utilizarla desde que se saca de la caja. Esto facilita su uso, lo cual lo hace una herramienta ideal de aprendizaje en el ámbito de diseño de sistemas electrónicos-automáticos.

Arduino consta de una amplia gama de placas con distinas carácterísticas en los componenetes, módulos, tamaños y señales que puede tratar. La elección de una u otro vendrá dada por los requerimientos del proyecto que se desee llevar a cabo con ella.

El funcionamiento de las diversas placas es muy similar entre ellas, el primer paso para usarla es instalar y configurar el entorno de programación de Arduino. Una vez hecho esto se conecta los componentes electrónicos necesarios para desarrollar el proyecto deseado. Estos irán conectados a los diversos pines de entrada y salida de la placa según convengan. Finalmente se configura el programa o sketch de ARDUINO que controlará el circuito conectado y se carga a la placa.

El lenguaje de programación usado presenta múltiples similitudes con C y C++, lenguajes que son contenido del plan de estudios de dos de la asignaturas de la carrera (INFORMÁTICA e INFORMÁTICA INDUSTRIAL), lo cual ha facilitado el proceso de aprendizaje de programación de la placa.

Se denomina Sketch a una parte de código fuente listo para abrir con el entorno de desarrollo integrado de Arduino y ser cargado sobre nuestro dispositivo. En él se encuentra escrito el comportamiento que tendrá nuestro proyecto: la respuesta ante entradas determinadas, cálculos internos, salidas del sistema, etc… Su extensión es Arduino file (.ino).

En la aplicación propuesta en el presente trabajo, se ha hecho uso de una ARDUINO UNO, que tiene ensamblado el microcontrolador Atmega328.

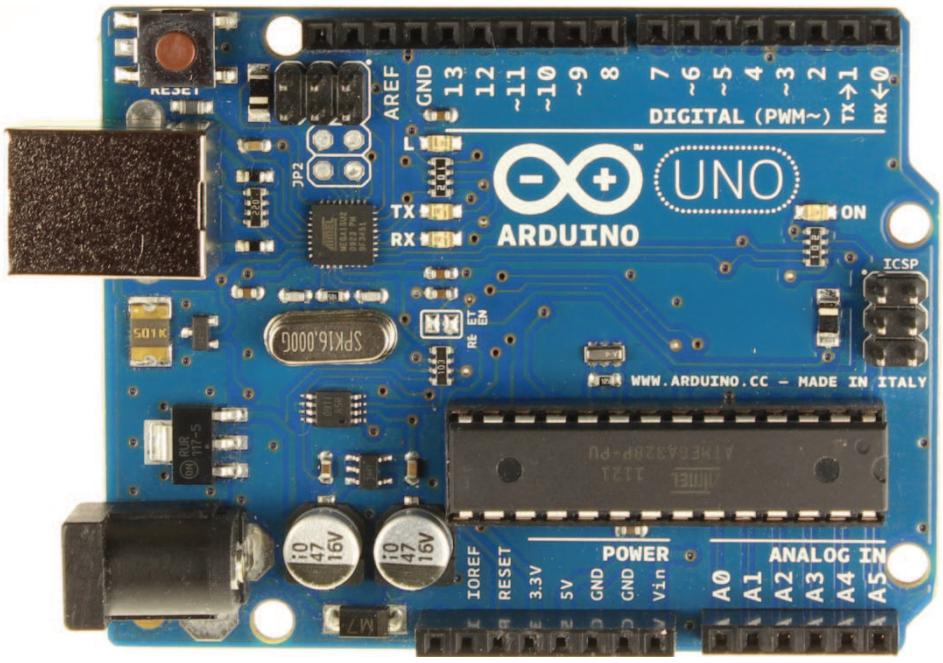

Figura 2.1: Arduino Uno (Imagen de http://www.flacsoandes.org/arteactual/?p=1666)

La gran comunidad que se ha formado en torno a la plataforma ha hecho que se disponga de una gran cantidad de módulos que permiten ampliar sus funcionalidades. Además se disponen de numerosos foros de ayuda donde los diversos usuarios se ayudan unos a otros para conseguir solventar los diversos problemas que se exponen en el mismo, lo cual facilita la aparición de nuevos proyectos con Arduino como elemento principal.

2.1.2. Sketchs de Arduino:

Como se ha comentado, los programas que controlan el comportamiento de la placa ARDUINO reciben el nombre de sketchs, y están escritos en un lenguaje de programación similar al C.

Cuando se observa un código de ARDUINO se pueden diferenciar distintas secciones que se repiten en cualquier sketch:

- Comentarios
- Declaraciones
- Función setup
- Función loop

2.1.2.1. Comentarios:

Es habitual encontrar en las primeras líneas de código una serie de comentarios que aclaren el funcionamiento que tendrá el programa. Los comentarios podrán ocupar una única línea o varias de ellas. En el primer caso deberán estar escritos a continuación de los símbolos // , mientras que en el segundo caso se escribirán los comentarios entre los símbolos /* ….*/.

2.1.2.2. Declaraciones:

Es la primera parte del código que es compilada. En ella se realizan las declaraciones de las variables que almacenan valores que son guardados para ser utilizados posteriormente por el programa. También se realiza la declaración de las librerías necesarias para el programa.

2.1.2.3. Función setup

La función setup se trata de la primera función que aparece en un código de ARDUINO, y es llamada cuando el sketch empieza. Es usada para realizar la inicialización de variables, definir los pines como entradas o salidas, empezar a usar librerías, etc. Es útil para asignar valores y propiedades a la placa que no varían durante toda la ejecución del código. Esta función únicamente se ejecutará una vez, después de encender o resetear la placa ARDUINO.

2.1.2.4. Función loop:

Es la siguiente función que aparece en el skecth de ARDUINO. Como su nombre indica se ejecuta una y otra vez de manera cíclica hasta que se pulsa el botón de reset de la placa o se apaga la alimentación de la placa. En esta función se define el comportamiento que debe seguir la placa en el proyecto desarrollado. En el caso que nos ocupa se han asignado aquí el comportamiento seguido por las señales de entrada del sensor CCD.

2.2. TCD1201D

El TCD1201D es un sensor óptico lineal compuesto por un conjunto de 2048 fotodiodos que proporcionan una salida proporcional a la cantidad de luz que les incide. Por sus características el TCD1201D es ideal para ser usado en aplicaciones de cálculo de distancias como la que se describe en el presente documento.

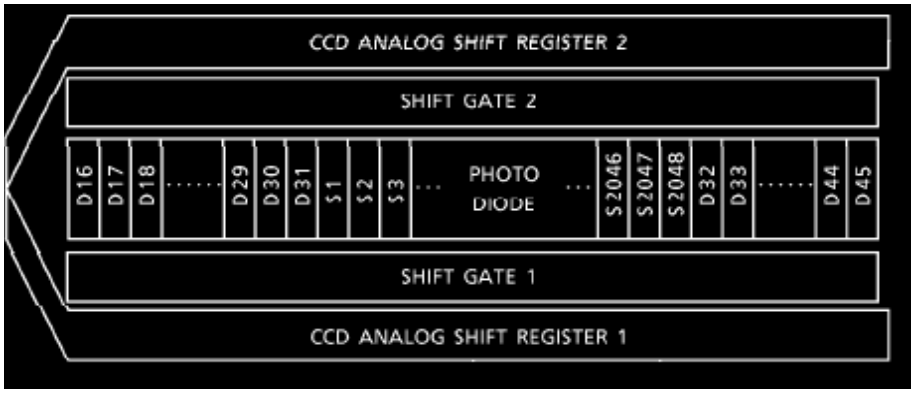

Figura 2.2: Esquema de TCD1201D (Hoja de datos del fabricante).

Como se puede observar en la figura: 2.2, aparte de los 2048 valores útiles de los fotodiodos, el TCD contiene de una serie de valores dummy que deberán ser tenidos en cuenta a la hora de realizar las lecturas.

Este sensor viene dispuesto en un encapsulado de 22 pines de los cuales 7 de ellos presentan funcionalidad (Del pin 7 al 18 no se conectan).

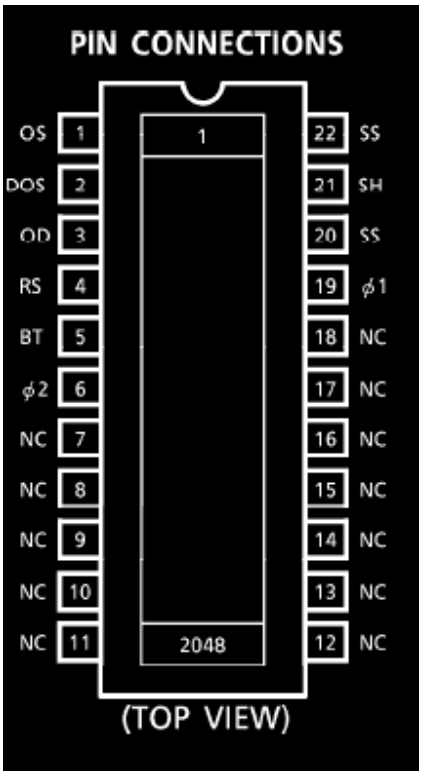

Figura 2.3: Esquema de los pines del TCD1201D del (Hoja de datos del fabricante).

En la figura: 2.3, se puede ver un esquema con las conexiones de cada uno de los pins del integrado. El dispositivo opera con una única fuente de alimentación de 5 voltios que recibe por el pin 3, siendo este el pin de alimentación.

El TCD1201D proporciona dos valores de salida, son las señales OS y DOS, que son proporcionadas por los pines 1 y 2 respectivamente. La inclusión de dos tipos de señales de ha salida tiene como objetivo detectar la cantidad de luz ambiental que incide sobre todo el sensor. La salida DOS asigna a todos los píxeles una cantidad de luz uniforme que representa

la cantidad de luz ambiental que incide sobre ellos. Esta cantidad de luz será igual para todos los fotodiodos del sensor.

La salida OS se corresponde con el valor de luz puntual que incide sobre cada uno de los fotodiodos del sensor.

Una descripción gráfica de estas señales se puede encontrar en la hoja de datos del fabricante, la podremos ver en la figura: 2.4.

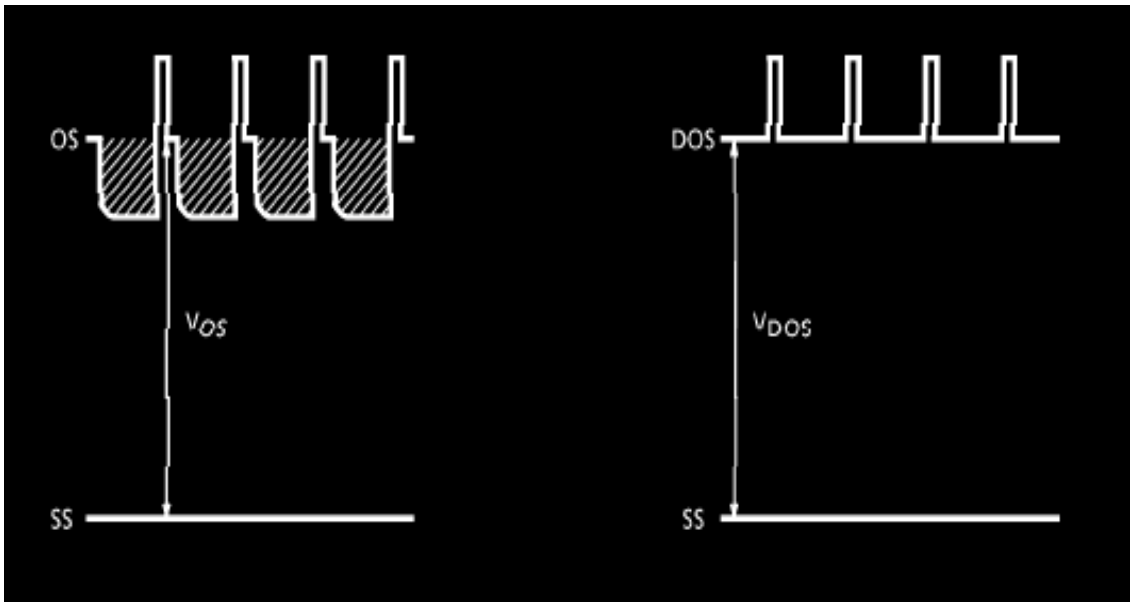

Figura 2.4: Forma de las señales de salida (Hoja de datos del fabricante)

Como se puede observar, el valor de V_{DOS} se mantiene constante, siendo este valor una compensación por el valor de luz natural que incide sobre el sensor. Por otro lado también se puede contemplar que V_{OS} se trata de una señal que cambia a lo largo del tiempo. Para cada valor de lectura toma un valor diferente en función de la luz incidente en el correspondiente fotodiodo. El resultado es una señal con forma de pulso con una amplitud dependiente del tiempo. Es importante tener en cuenta que el valor de voltaje de la señal VOS es menor cuanto mayor sea la cantidad de luz incidente sobre el fotodiodo correspondiente. Este aspecto será tomado en cuenta posteriormente a la hora de tratar los datos.

El diagrama de tiempos con los requerimientos que deben cumplir las señales de control nos viene dado por el fabricante y se puede observar en la figura 2.5:

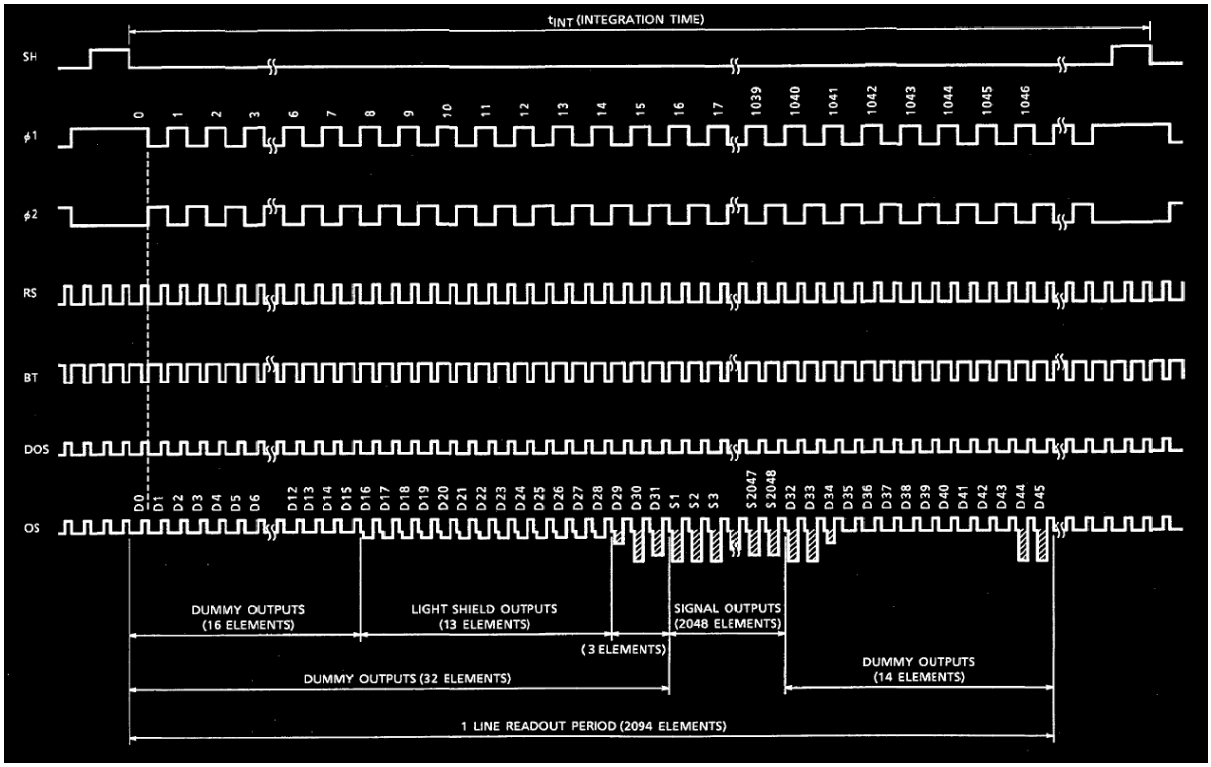

Figura 2.5: Diagrama de tiempos del TCD1201D (Hoja de datos del fabricante).

Como se puede observar en la figura:__ el diagrama de tiempos del TCD1201D se encuentra divido en varias zonas significativamente distintas.

En primer lugares necesario una serie de ciclos iniciales que preparan al sensor `para la lectura. Este tramo finaliza cuando la señal SH pasa de 5V a 0V.

En este momento se inicia el ciclo de lectura. Sin embargo los valores que obtiene como salida no son los correspondientes a los fotodiodos que posee, sino que son valores dummy, o carentes de información. Antes de encontrar el primer valor de lectura útil se realiza la lectura de 32 valores dummys. Como se puede ver aparecen divididos en 3 tramos: un tramo de 16 salidas que proporciona valores sin información; otro de 13 elementos que se denominan como protectores de luz y también son carentes de información; y un último tramo de 3 elementos dummy que no poseen descripción alguna por parte del fabricante. Estos 3 tramos

suman un total de 32 elementos dummys, que corresponderán con las primeras 32 salidas del sensor.

Una vez han concluido estos pulsos, el sensor ofrece como salida los pulsos con la información de los 2048 fotodiodos. Estos pulsos poseerán una amplitud proporcional a la cantidad de luz que incida en cada fotodiodo.

Cuando se han producido todos los pulsos correspondientes al sensor, se encuentran 14 salidas dummy. Estas salidas, al igual de lo que ocurrían al inicio del ciclo de lectura, no aportan información alguna de relevancia, pero es necesario reproducirlas para el correcto funcionamiento del sensor.

Finalmente es necesario realizar una serie de pulsos extras para finalizar el ciclo de lectura, así como volver a poner a 5V la señal SH, para que termine de incrementarse el tiempo integral.

En cuanto a las dimensiones del sensor TCD1201D, cabe destacar que el integrado en el cual se encuentra encapsulado el sensor tiene una longitud de 3cm. Cada fotodiodo posee una dimensión de 14um, habiendo un total de 2048 fotodiodos no sale una longitud útil de:

$$X = 2048 \times 14\ \mu m = 2.8672\ cm$$

Ecuación 2.1.

Dicha longitud se encuentra centrada en el integrando. Estos datos deberán ser tenidos en cuenta a la hora de calcular las distancias máximas y mínimas que puede medir el sensor así como su resolución.

Para obtener medidas más precisas será necesario tener en cuenta el grosor del borde del encapsulado que se encuentra antes del primer diodo.

Para calcularlo deberemos restar la longitud que componen los 2048 fotodiodos a la longitud total del encapsulado del sensor.

La longitud del encapsulado del sensor es de 4.2cm, mientras que la longitud útil se ha visto que es de 2.8672cm, por lo tanto nos queda que el grosor de los bordes es de:

$$Grosor = \frac{4.16 - 2.8672}{2} = 0.646\ cm$$

Ecuación 2.2.

Se ha dividido entre 2 porque únicamente es necesario tener en cuenta el grosor del borde situado antes del primer fotodiodo. Además se ha tenido en cuenta que el sensor posee una disposición simétrica, con el espacio de lectura útil centrado en el encapsulado. En la figura: 2.6 se puede ver el dimensionamiento del sensor, que deberá ser tomado en cuenta además a la hora de pasar el circuito desde la protoboard a una placa más rígida.

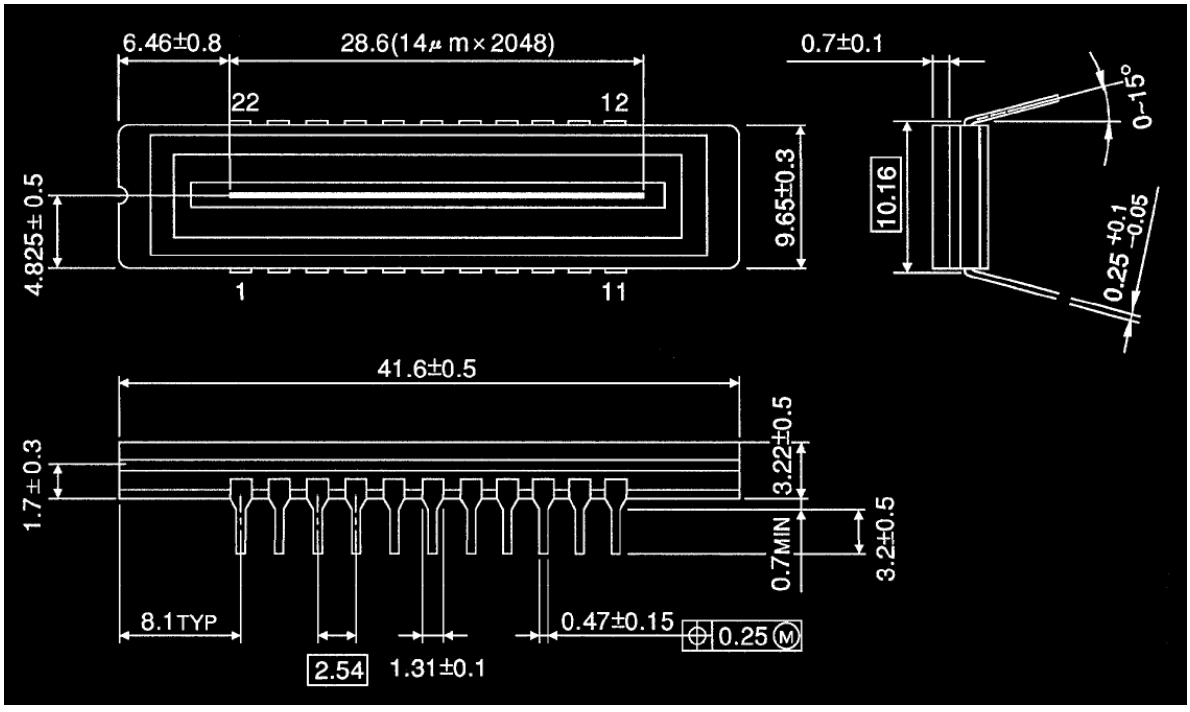

Figura 2.6: Esquema del dimensionamiento del TCD1201D (Hoja de datos del fabricante)

2.3. ILX511

El ILX511 de Sony, se trata de un sensor óptico CCD lineal, al igual que el tCD1201D. Sin embargo el ILX511 presenta una serie de diferencias con el TCD1201D que han hecho que sea considerado también como un sensor alternativo al TCD1201D en el diseño del prototipo.

Posee un total de 2048 píxeles útiles, al igual que el TCD1201D. Además presenta las mismas dimensiones y, al igual que el TCD1201D, se presenta dispuesto en un encapsulado de 22 pines. En la figura: 2.7 se puede ver un esquema del integrado, proporcionada por el fabricante en su correspondiente hoja de datos:

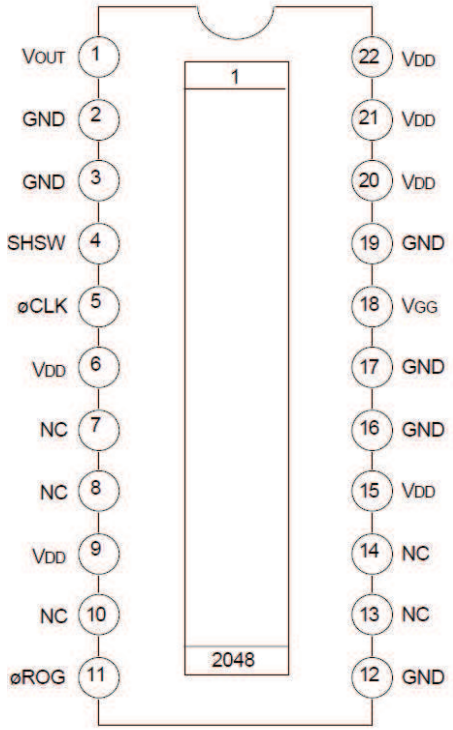

Figura 2.7: Esquema del integrado del ILX511 (Hoja de datos del fabricante).

La principal ventaja que se ofrece con respecto al TCD1201D es que el ILX511 precisa únicamente de dos señales de control en su entrada: una señal de reloj denominada por el fabricante como CLK, que se suministra al sensor a través del pin 5; y una señal encargada de controlar el tiempo de integración del sensor, que aparece denominada por el fabricante con el nombre de ROG, y que es suministrada al sensor a través del pin 11.

La salida se encuentra en el pin 1, y esta puede presentarse de dos maneras en función de la configuración del pin 4 SHSW, que permite variar entre los dos métodos de operación que permite el sensor ILX511: con S/H o sin S/H.

A continuación se muestran dos gráficas en las cuales se puede ver la forma que presentaría la señal de salida en cada uno de los métodos.

Sin S/H:

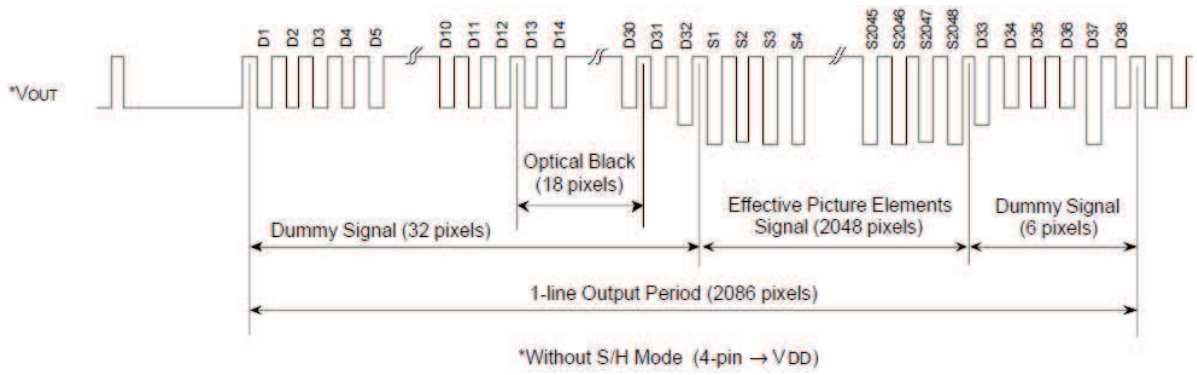

Figura 2.8: Señal de salida sin S/H del ILX511 (Hoja de datos del fabricante).

Como se puede observar en la figura: 2.8 la señal de salida operando sin S/H, se encuentra compuesta por una serie de pulsos de amplitud variable en función de la cantidad de luz que afecte a cada diodo.

Con S/H:

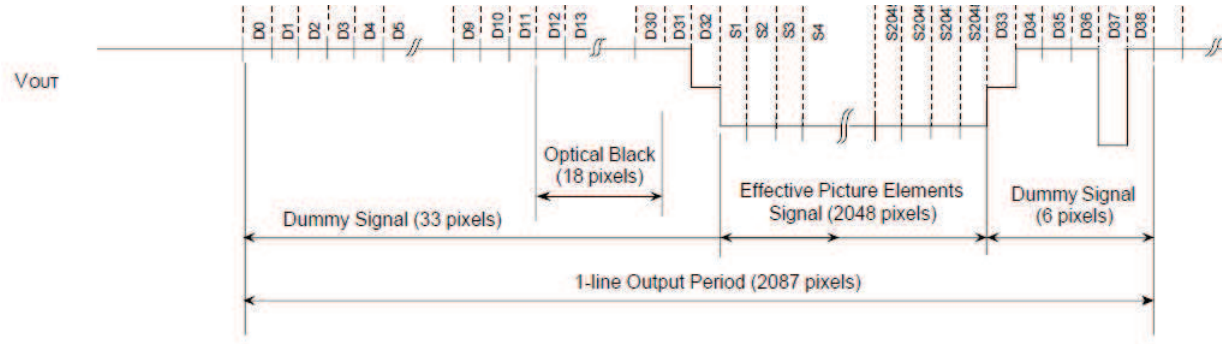

Figura 2.9: Señal de salida con S/H del ILX511 (Hoja de datos del fabricante).

La figura: 2.9 representa la forma que tendría la señal de salida del sensor si se opera con S/H. En este caso la señal de salida es una señal continua que varía su amplitud en función de la luz que le incida a cada fotodiodo. La principal diferencia que se contempla al comparar las figuras: 2.8 y 2.9 es que en el segundo caso la señal carece de pulsos como pasaba cuando se operaba sin S/H.

Para configurar el régimen en el que operara el sensor se ha de configurar el pin 4 como indica el fabricante en la tabla: 2.1.

Modelo	Pin 4 (SHSW)
Con S/H	GND
Sin S/H	V_{DD}

Tabla 2.1: Configuración del Pin SHSW del ILX511 (Hoja de datos del fabricante).

En la experimentación realizada en el laboratorio se ha decidido optar por utilizar el ILX511 con S/H debido a que se ha considerado que la forma de la señal obtenida con este modelo es más sencilla de entender que en el modelo sin S/H.

Mediante el empleo de ARDUINO se pretende generar las señales de entradas necesarias con el fin de controlar el sensor. Para ello se deberá seguir la tabla de tiempos propuesta por el fabricante en la hoja de datos. Dicho diagrama aparece representado en la figura:__

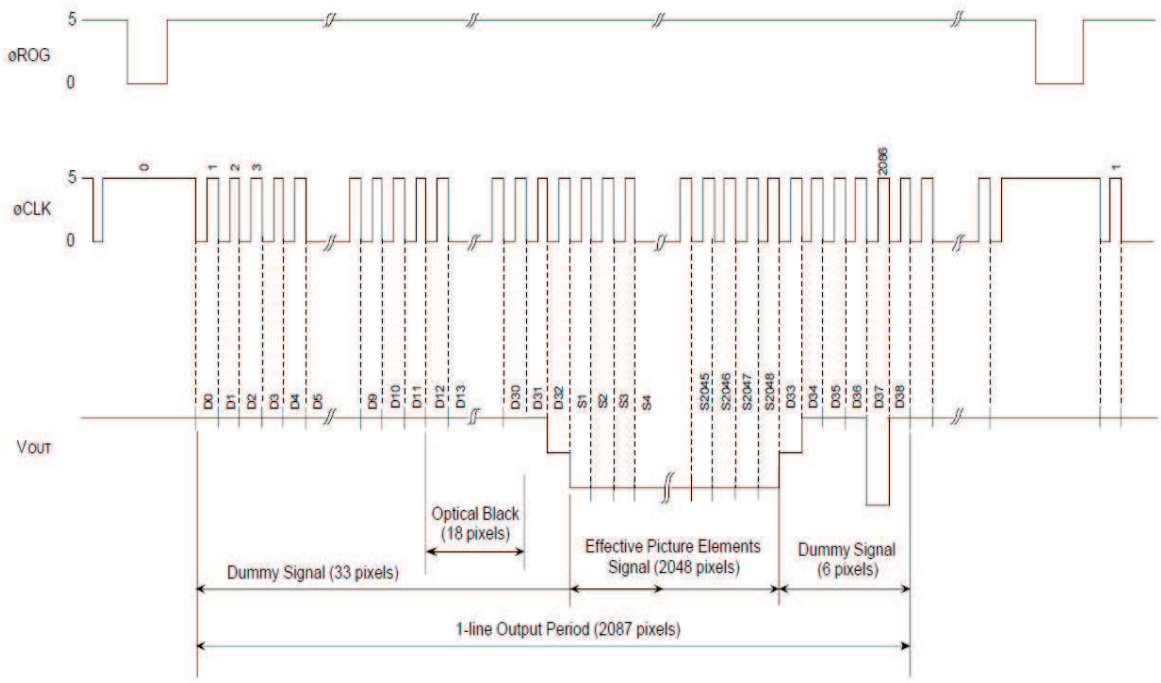

Figura 2.10: Diagrama de tiempos del ILX511 (Hoja de datos del fabricante).

Como se puede observar en la figura: 2.10 se presentan algunas similitudes con el diagrama de tiempos del TCD1201D. En ambos casos son necesarios una serie de pulsos iniciales para configurar el sensor que obtienen como resultado una señal de salida sin información o dummy.

Al principio de un ciclo de lectura es necesario poner a 0 lógico un cierto periodo de tiempo la señal ROG, con el objetivo de reiniciar el tiempo de integración. Una vez esta señal pasa a 5V y el reloj empieza a generar pulsos, se produce la salida de un total de 33 valores dummy. Una vez se han sucedido estos pulsos comienza una lectura de 2048 valores útiles, al igual que pasaba en el caso del TCD1201D. Finalmente es necesario realizar al menos 6 pulsos más para terminar el ciclo de lectura.

En el caso del ILX511 se posee el mismo espacio de lectura útil que en el caso del TCD1201D, así como el borde situado antes del primer fotodiodo, que sigue siendo de 0.646cm. En la figura: 2.11 se puede ver todo el dimensionamiento del sensor ILX511.

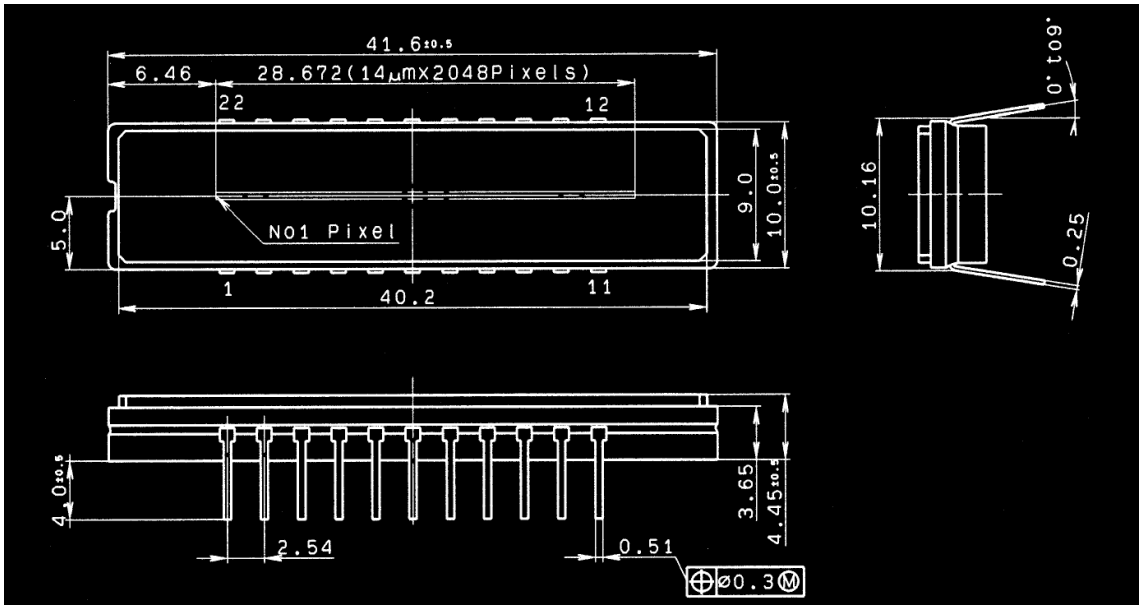

Figura 2.11: Esquema del dimensionamiento del ILX511 (Hoja de datos del fabricante).

El sensor ILX511 presenta una serie de ventajas respecto al TCD1201D que han motivado a realizar la experimentación con él también. A parte de que sean necesarias menos señales de control, también ofrece la ventaja de poder permitir trabajar a frecuencias más bajas que el TCD1201D, lo cual es importante cuando se trabaja con Arduino.

3. Procedimientos

3.1. Circuitos electrónicos

3.1.1. Circuito del TCD1201D

Para que el sensor TCD1201D funcione de manera correcta deberá montarse en un circuito electrónico que viene indicado por el fabricante. Se pueden ver sus características en la hoja de datos del sensor. En la figura: 3.1 aparece ilustrado el esquema eléctrico que aparece en la hoja de datos con detalles de todas las conexiones:

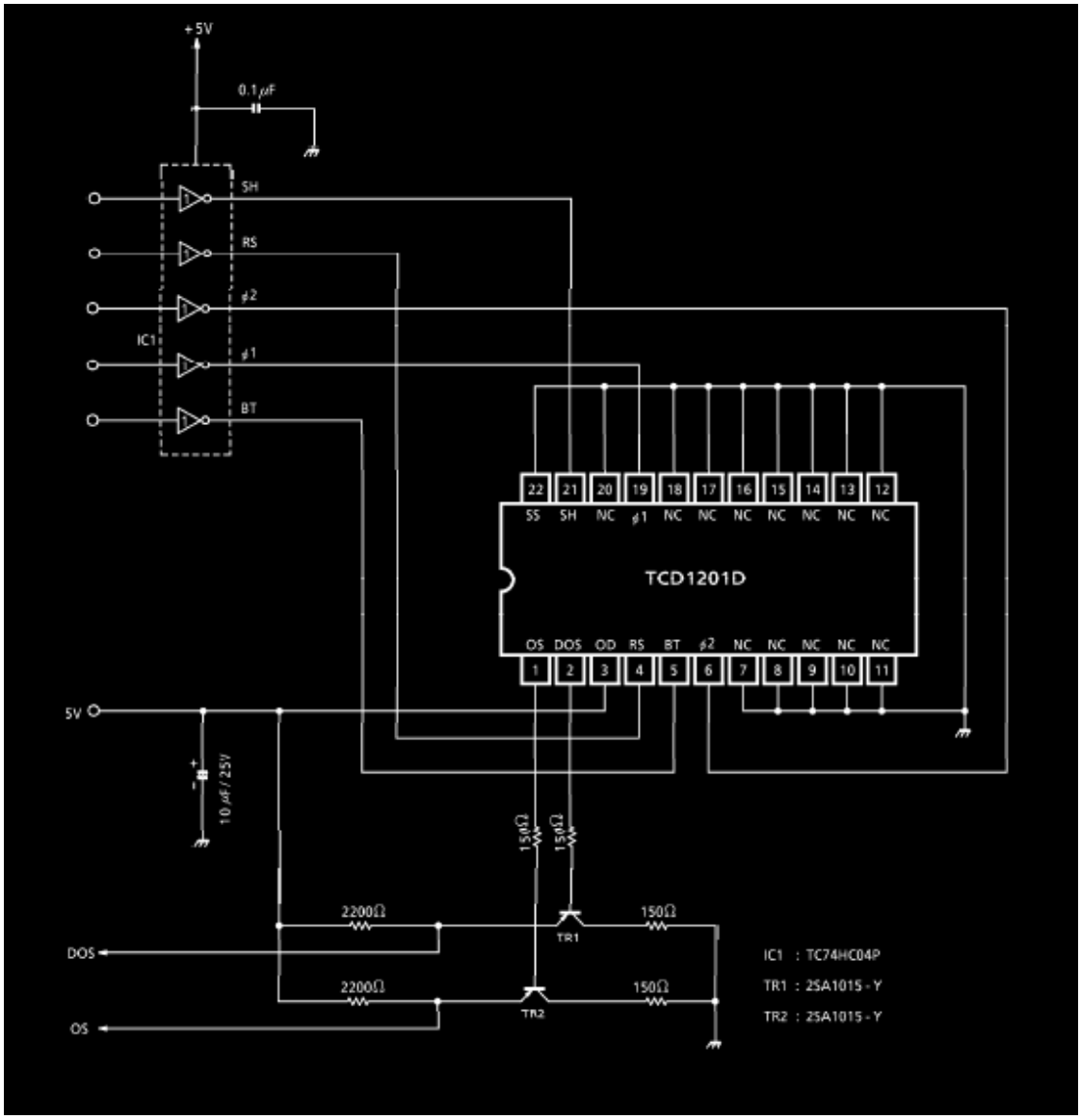

Figura 3.1: Esquema del circuito de funcionamiento del TCD1201D (Hoja de datos del fabricante)

Como se puede observar en la figura: 3.1, el circuito propuesto por el fabricante consta de dos transistores bipolares de unión PNP conectados a los pines 1 y 2 del integrado del TCD1201D. La conexión entre el transistor y el pin correspondiente se realiza a través de la base y con la inclusión de una resistencia de 150Ω que tiene la función de limitar la corriente. El colector se encuentra unido a tierra a través de otra resistencia de 150Ω mientras que el emisor se une a la alimentación de 5V mediante una resistencia de 2k2Ω. Estos transistores en la configuración en la que se encuentran conectados tienen la función de amplificar la corriente asociada a las salidas del sensor.Para medir la salida amplificada del sensor habría que medir en el nodo que une la resistencia de 2k2Ω con el emisor del transistor.

Al esquema del circuito proporcionado por el fabricante hay que añadirle la placa ARDUINO, teniendo en cuenta que las conexiones se establecen entre los pines:

Pin PORTD	3	4	5	6	7
Pin TCD1201D	21	19	6	4	5

Tabla 3.1: Conexiones entre Arduino y el TCD1201D

Conectando la ARDUINO al circuito mediante el conexionado descrito en la tabla: 3.1, obtenemos un circuito como el de la figura: 3.2.

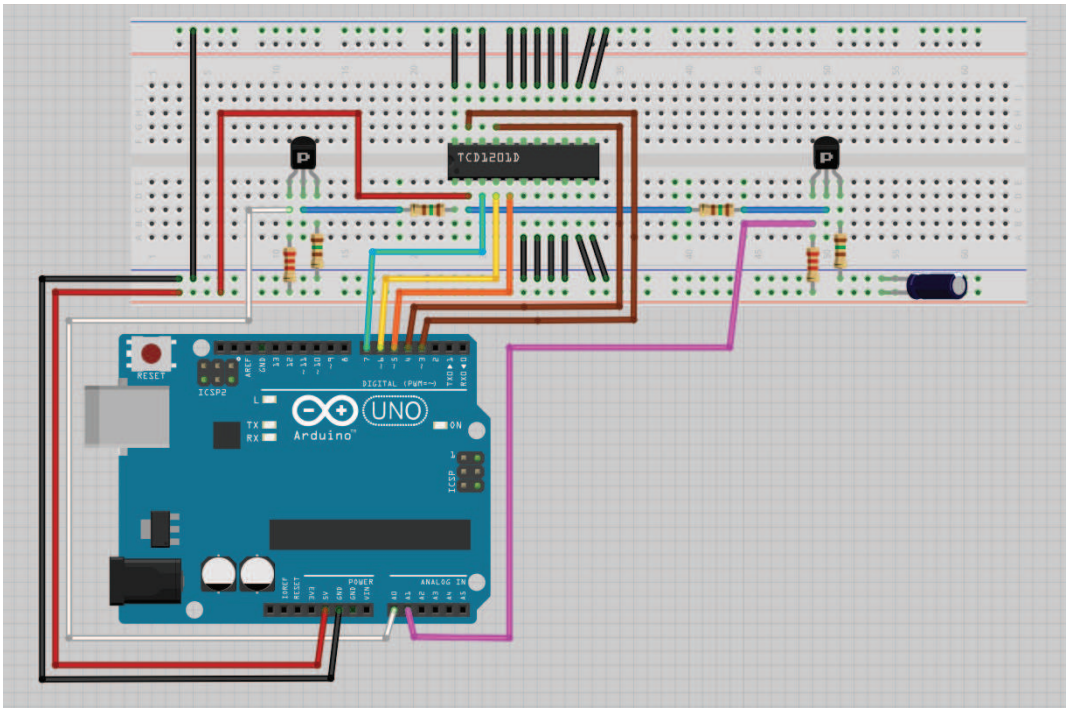

Figura 3.2: Esquema del circuito montado en el laboratorio.

El montaje de este circuito se ha realizado en la protoboard. Posteriormente con el objetivo de ganar en firmeza y rigidez se ha transferido el circuito a una placa física. Esto se ha realizado soldando los distintos componentes a la placa mediante soldadura de estaño. Además se ha requerido del uso de un taladro para perforar las pistas que unían zonas del circuito con voltajes distintos. Es importante tener a mano un multímetro durante toda la operación de fabricación del circuito, pues es necesario medir continuidades entre los puntos que generen dudas para evitar posibles cortocircuitos.

En la figura: 3.3 se puede ver el circuito una vez soldados todos los elementos

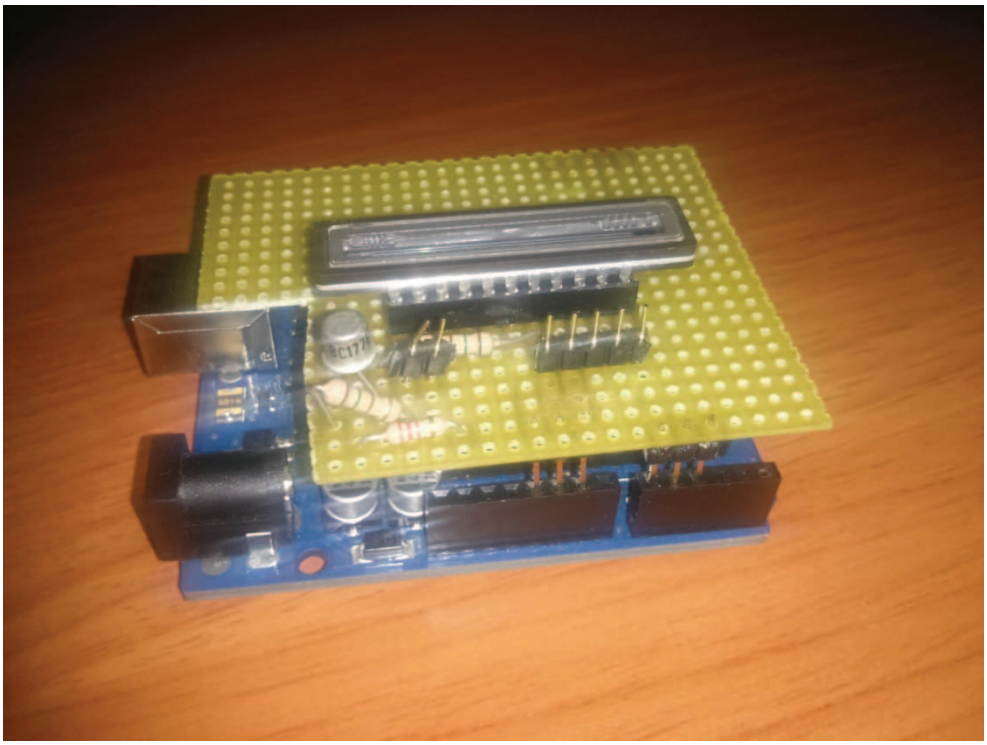

Figura 3.3: Circuito de funcionamiento del TCD1201D soldado a la placa y conectado a Arduino

3.1.2. Circuito del ILX511:

Al igual que pasaba con el TCD1201D, el ILX511 también precisa de un circuito electrónico, indicado por el fabricante en la hoja de datos, para poder funcionar de manera correcta. Dicho circuito puede observarse en la figura: 3.4.

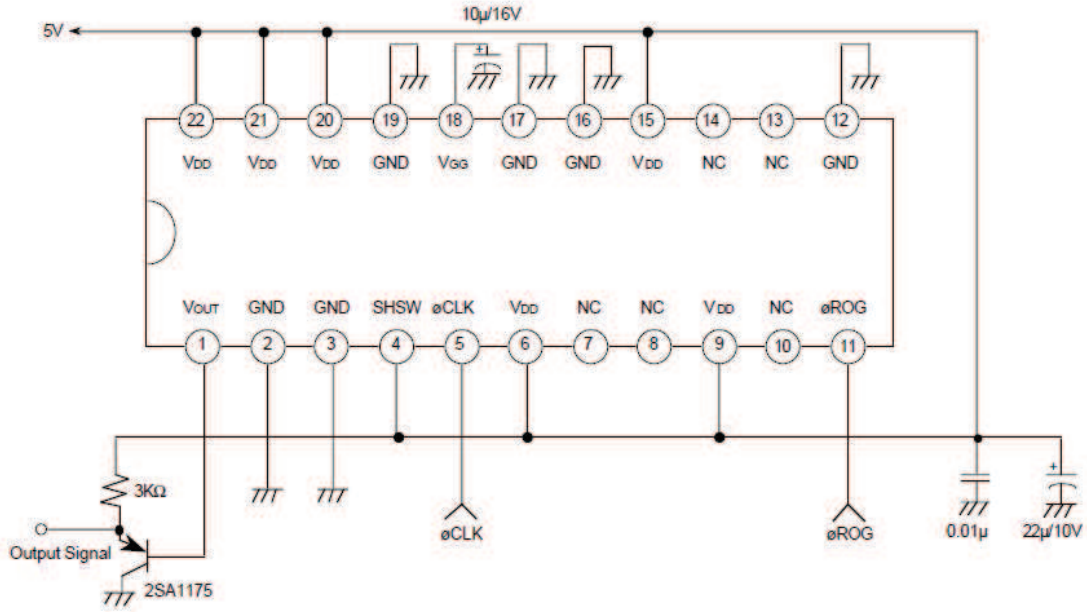

Figura 3.4: Esquema del circuito de funcionamiento del ILX511 (Hoja de datos del fabricante).

Nótese que con el circuito propuesto por el fabricante se opera sin S/H. Como la experimentación que se ha realizado con el ILX511 ha sido con S/H es importante cambiar la conexión propuesta del pin 4 SHSW y conectarlo a tierra en vez de a la alimentación.

Al igual que en el circuito del TCD1201D se ha colocado un transistor PNP en la salida del sensor con el fin de amplificar la corriente de salida.

A su vez se han dispuesto de una serie de condensadores que se encargan de mantener estables las señales de alimentación.

A este esquema se le añadirá la placa ARDUINO, que es el dispositivo encargado de generar las señales de control del sensor. A la hora de conetarlo es importante tener en cuenta el conexionado que se hará:

Señal	Pin ARDUINO	Pin ILX511
ROG	12	11
CLK	13	5

Tabla 3.2: Conexiones entre Arduino y el ILX511.

En la figura: 3.5 se puede ver el esquema del circuito con la ARDUINO conectada.

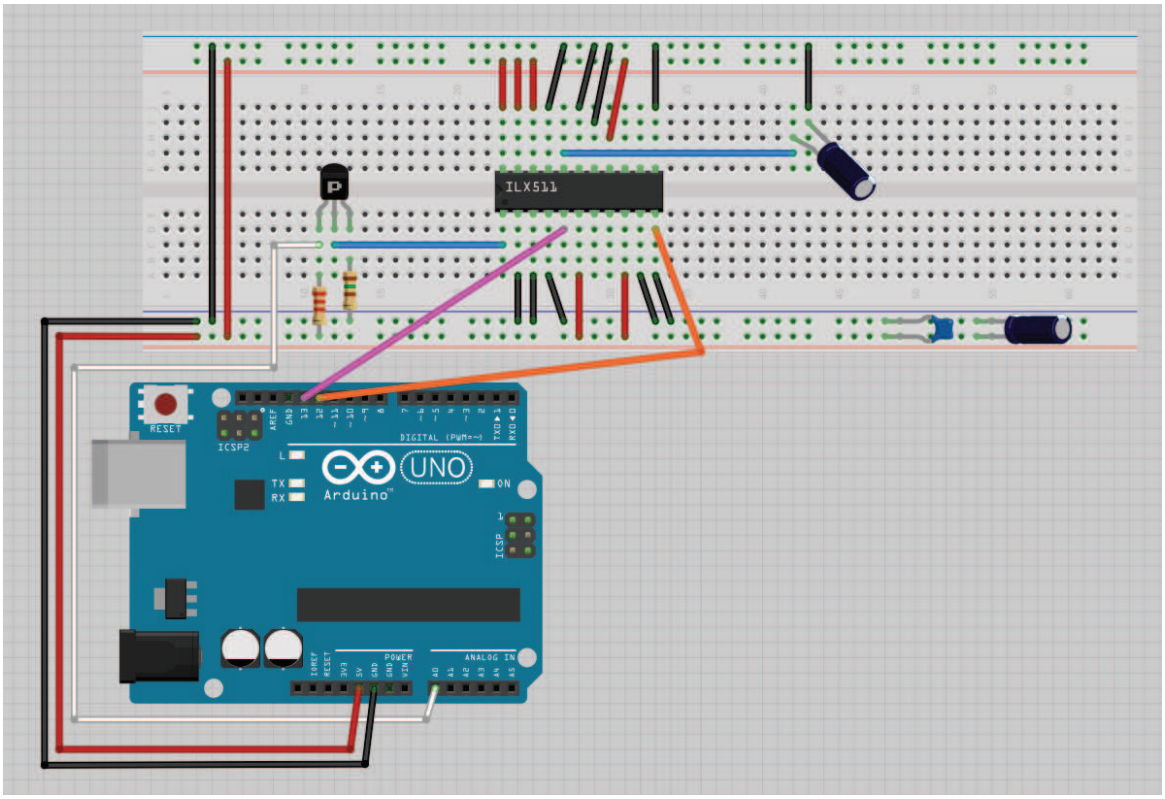

Figura 3.5: Esquema del circuito montado en el laboratorio

3.2. Control del TCD1201D

Como se ha comentado anteriormente, el sensor TCD1201D presenta unas características técnicas particulares que lo convierten en un elemento ideal para el cálculo de distancias.

En el presente apartado se exponen los pasos seguidos en la experimentación con el sensor, las dificultades que han surgido en el proceso y una breve exposición de los resultados obtenidos.

El primer paso en la experimentación con el TCD1201D ha sido una lectura detallada de la hoja de datos con el fin de conocer las características técnicas del sensor y los requerimientos que precisa el mismo para un correcto funcionamiento.

Para el correcto uso del sensor son requeridas señales de control a la entrada con unas características técnicas muy particulares. Como se ha comentado anteriormente este sensor requiere el uso de2 pares de relojes a la entrada, son los pares ϕ_1 - ϕ_2y RS – BT; siendo el par RS-BT el doble de rápido que el par ϕ_1 - ϕ_2.

Los requisitos de estos sensores se pueden ver en la figura 3.6.

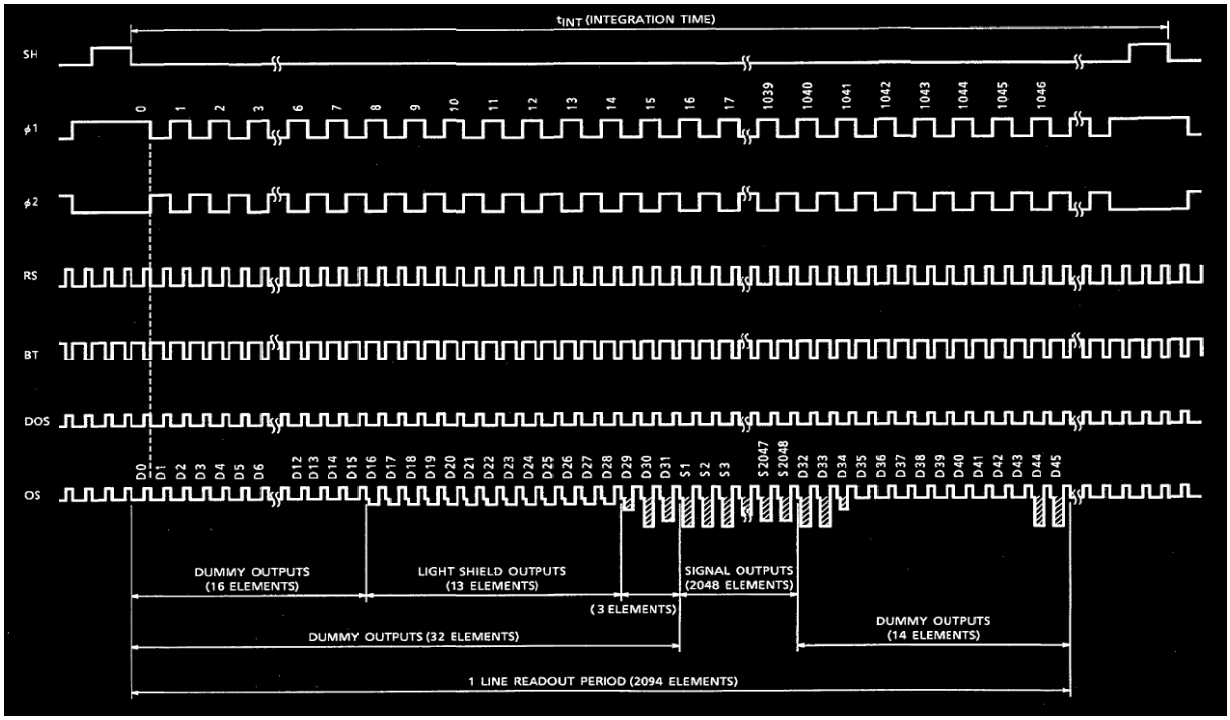

Figura: 3.6: Diagrama de tiempos del TCD1201D (Hoja de datos del fabricante).

Para controlar el TCD1201D con ARDUINO es necesario replicar estas señales mediante un sketch que las reproduzca de manera precisa. Para ello es necesario utilizar un registro de salida que cambie su estado en función del tiempo, reproduciendo cada uno de los bits del mismo cada una de las cinco señales de entrada del TCD1201D.

El registro utilizado para reproducir las señales de entrada necesarias del TCD1201D es el PORTD. Son necesarias 5 señales salida para el correcto funcionamiento del sensor, por lo tanto se han hecho uso de los pines 3 a 7 del PORTD, dejando los pines 0, 1 y 2 con un valor de 0 lógico durante toda la ejecución del programa.

3.2.1. Comentarios y definiciones:

En primer lugar se deben asignar los pines del PORTD a cada una de las entradas del integrado. Esto se realiza a, principio del código mediante el comando #define.

```
/*
  Linear CCD readout
  lCCD: Toshiba TCD 1201D, 2048 pixel
  Pinmap: 22 DIL
  Chip                                Arduino Pin
    1  :  OS (Output Signal)            A0
    2  :  DOS (Compensation Output)     A1
    3  :  OD (Power +5V)                +5V
    4  :  RS (Reset Gate)               6
    5  :  BT (Boost Pulse)              7
    6  :  P2 (clock Phase 2)            5
    19 :  P1 (clock Phase 1)            4
    21 :  SH (Shift Gate)               3
    Resto de pines :                    GND
*/
#define OS_PIN A0
#define DOS_PIN A1
#define SH_PIN 3
#define P1_PIN 4
#define P2_PIN 5
#define RS_PIN 6
#define BT_PIN 7
```

Figura 3.7: Comentarios iniciales y definición de pines.

Como se puede observar en la figura: 3.7, por comodidad a la hora de entender el código se ha incluido un comentario inicial en el cual se describe el mapeado del CCD. Aparecen detalladas las conexiones existentes entre el Arduino y el CCD.

Debajo de este comentario se encuentra la asignación de pines a las entradas del CCD. Como se puede observar se han asignado las entradas SH, P1, P2, RS, y BT a los pines 3, 4, 5, 6, y 7 respectivamente.

3.2.2. Función setup:

La definición detallada de la función setup la podemos ver en la figura: 3.8:

```
void setup() {
  Serial.begin(9600);  //velocidad de datos en bits por segundo (baudios) para la transmisión
                       //de datos en serie para comunicarse con el computador.

  Serial.println("Setup start");

  //Asignamos cada pin al bit del portD que queramos
  for (int i=3;i<8;i++) pinMode(i,OUTPUT);

  PORTD = B00000000;
  Serial.println("Setup Complete");
}
```

Figura 3.8: Función setup.

Como podemos observar se realizan varios procesos. En primer lugar se asigna la velocidad de transmisión de datos en serie, que será de 9600 baudios. Luego se imprime por pantalla el mensaje “Setup Start en el cual se indica que comenzarán las asignaciones requeridas para el correcto funcionamiento del programa.

Las asignaciones realizadas en el setup serán 2:

- Definimos los pines 3 a 7 del PORTD como salidas. Para ellos se realiza un bucle for que irá asignando consecutivamente cada pin como salida.
- Se les asigna un 0 lógico a todos los pines del PORTD para comenzar el programa.

Una vez realizadas estas asignaciones se imprime por pantalla el mensaje “Setup Complete”, indicando así que se han terminado de realizar las asignaciones del setup.

3.2.3. Función loop():

La función loop() se encarga de llamar cíclicamente a la función readCCD(), donde se encuentra el código encargado de realizar las ondas de entrada necesarias para el TCD1201D. Entre cada ciclo de lectura se ha incorporado un delay de 100 ms. Además se imprime por pantalla el mensaje “Start loop”, que nos indica cuando se vuelve a ejecutar un nuevo ciclo.

```
void loop() {

 Serial.println("start loop");

 //Instanciampos una funcion encargada de realizar
 //las lecturas del sensor
   readCCD();
   delay(100);
}
```

Figura 3.9: Función loop.

3.2.4. Función readCCD

La función readCCD() se trata de una función sin argumento de entrada y sin valor de retorno, que se encarga de generar todas las ondas de entrada requeridas por el TCD1201D. Para ellos se van haciendo sucesivas asignaciones al PORTD que van reproduciendo las ondas requeridas en las sucesivas entradas del TCD1201D.

3.2.4.1. Preparación del sensor:

Antes de realizar la lectura, es necesario realizar una serie de ciclos previos que sirven para preparar el sensor para las posteriores lecturas.

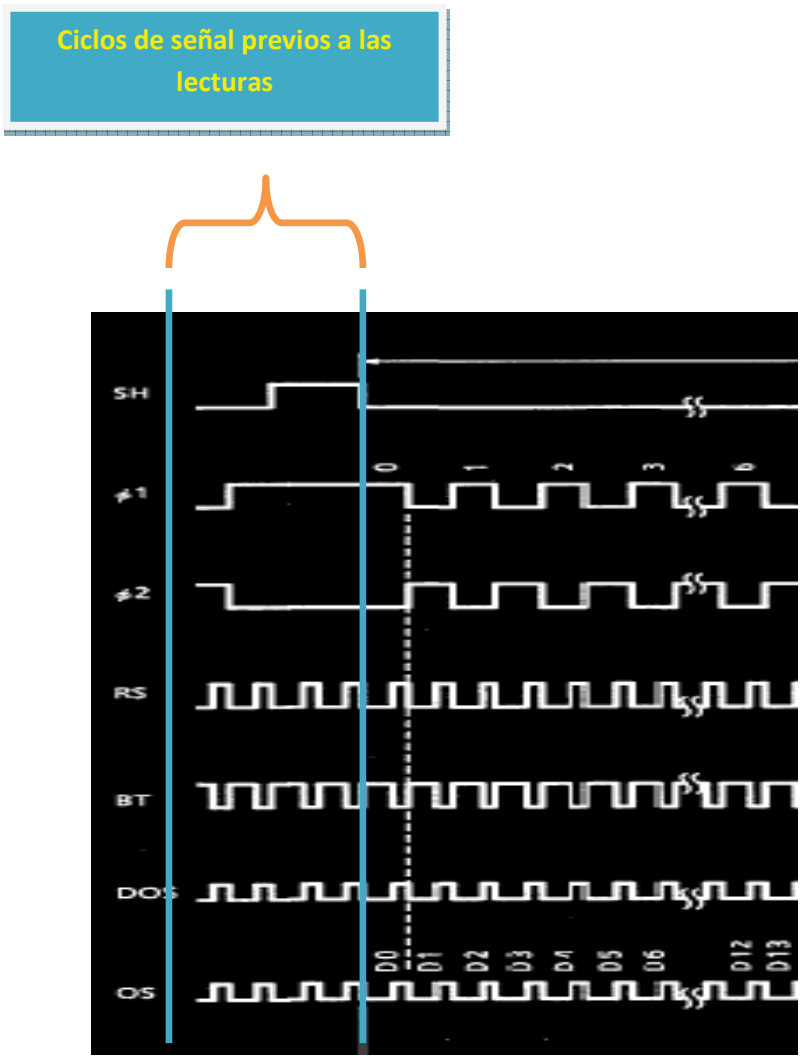

Figura 3.10: Ciclos de señal previos a la lectura (Adaptado de la hoja de datos del fabricante).

La secuencia se puede observar en la figura 3.10. En primer lugar serealizarán un par de ciclos con la señal SH a 0. Posteriormente se pasa esta señal a 1, mientras se siguen realizando ciclos de reloj de las otras 4 señales. Cuando la señal SH pasa de un 1 lógico a un 0 lógico, el tiempo de integración se activa y el sensor comienza su ciclo de lectura.

En el fragmento de código detallado en la figura: 3.11 podemos ver como se asignan estos ciclos iniciales. Además se han añadido unos delays de 2 microsegundos en el momento que pasa la señal SH de 1 a 0 (momento en el que empieza la lectura).

```
void readCCD() {
  Serial.println("Start read cycle");
 // Lo ponemos todo a 0
 PORTD = B00000000;
 delay(1);

 PORTD= B10100000;
 PORTD= B10010000;
 PORTD= B10011000;
 delayMicroseconds(2);
 PORTD= B10010000;
 delayMicroseconds(2);
```

Figura 3.11: Ciclos de preparación del sensor.

3.2.4.2. Lectura de los primeros 32 valores dummy;

Cuando la señal SH pasa de 1 a 0 comienza el ciclo de lectura del sensor y se realiza la lectura de los primeros 32 valores dummy. En la figura: 3.12 se ven representados estos valores.

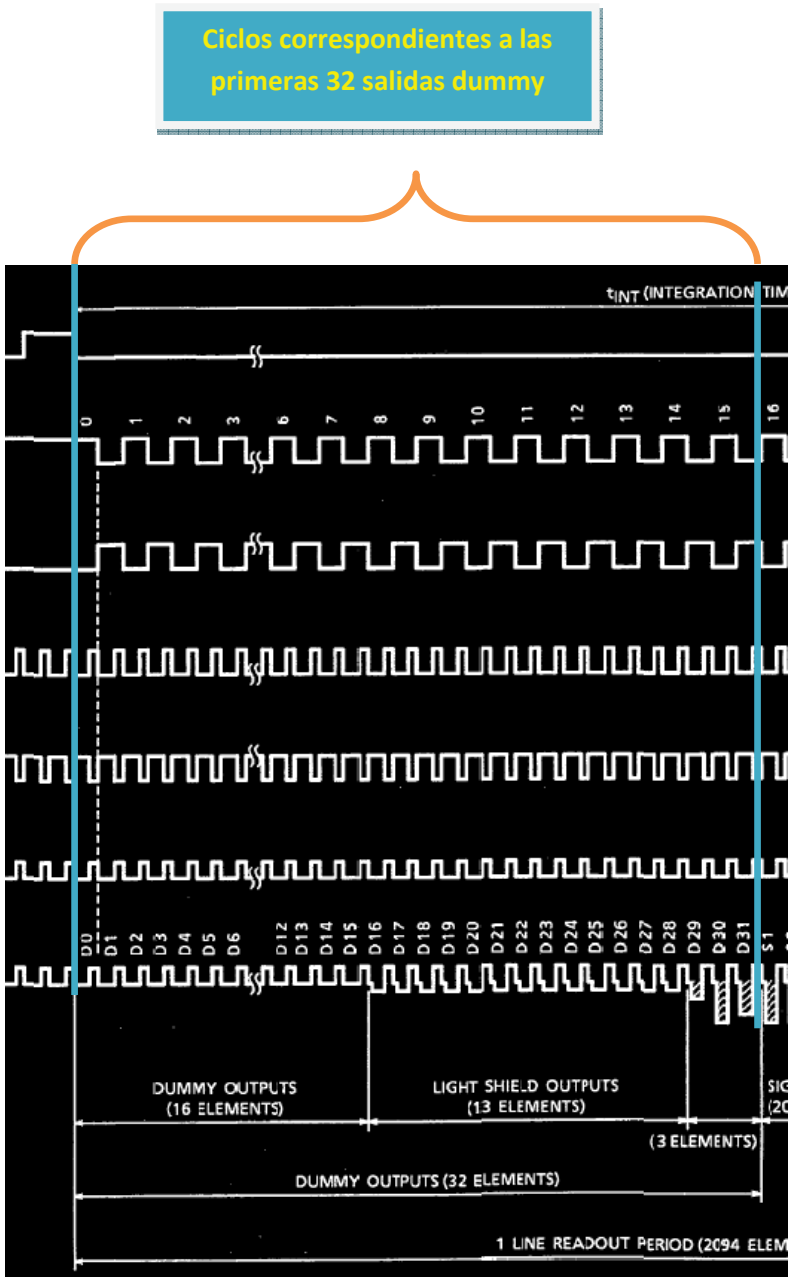

Figura 3.12: Primeras 32 salidas dummy. (Adaptado de la hoja de datos del fabricante).

La lectura de dichos valores corresponde con el primer tramo de lectura de la función readCCD(), y su definición puede contemplarse en el código de la figura 3.13.

```
for (int i=0; i<16;i++){

   PORTD= B00010000;
   PORTD= B01010000;
   PORTD= B11010000;
   PORTD= B10010000;
   PORTD= B10100000;

   PORTD= B00100000;
   PORTD= B01100000;
   PORTD= B11100000;
   PORTD= B10100000;
   PORTD= B10010000;

   }
```

Figura 3.13: Fragmento de código que genera las 32 primeras salidas dummy

El código se ejecuta dentro de un bucle for de 16 iteraciones con un total de 2 lecturas por cada iteración. Cada 5 líneas de código correspondientes a asignaciones del PORTD se realiza una lectura del sensor, realizándose un total de 2x16= 32 lecturas de valores dummy.

3.2.4.3. Lectura de los 2048 valores útiles:

Una vez leídos los primeros 32 dummys empieza la lectura de los 2048 fotodiodos de los que se encuentra compuesto el TCD1201D. Para ello se procede de manera similar a la lectura de los 32 dummys, pues las características de los relojes no cambian.

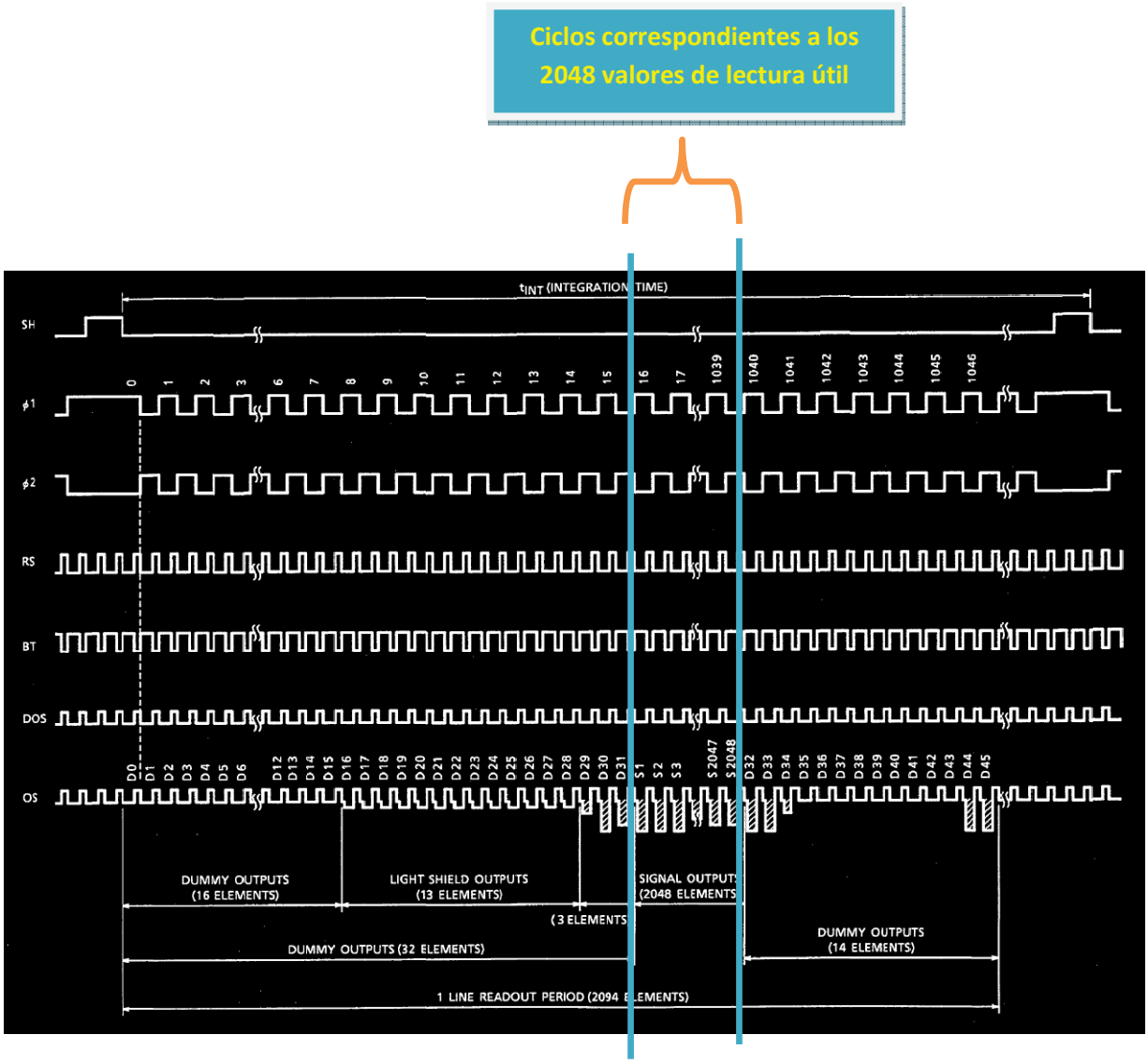

Figura 3.14: 2048 valores de lectura útiles (Adaptado de la hoja de datos del fabricante).

En la figura 3.14 aparece escalado el periodo de lectura útil, sin embargo comprende prácticamente la totalidad del ciclo de lectura total del sensor.

En el programa que se expone en el presente apartado las salidas han sido captadas mediante el osciloscopio digital Hameg, adicionalmente y para dotar de mayor funcionalidad al prototipo se puede insertar a la hora de realizar las lecturas una línea de código que se encargue de realizar las lecturas de los valores analógicos de salida del sensor. Esta línea es el analogRead().

En la experimentación llevada a cabo no se pudo introducir satisfactoriamente esta línea de código, pues el procesador del ARDUINO UNO no era lo suficientemente rápido como para ejecutar dicha línea sin insertar un retraso considerable en la ejecución, lo cual descuadraba las señales. Sin embargo haciendo uso de un procesador más rápido el tiempo de ejecución de dicha línea de código podría ser lo suficientemente baja como para que no afecte a la forma de las señales.

El fragmento de código quedaría como se muestra en la figura 3.15.

```
for (int i=0; i<1024; i++){

  PORTD= B00010000;
  PORTD= B01010000;
  PORTD= B11010000;
  PORTD= B10010000;
  PORTD= B10100000;
  valor = analogRead(-A0);

  PORTD= B00100000;
  PORTD= B01100000;
  PORTD= B11100000;
  PORTD= B10100000;
  PORTD= B10010000;
  valor = analogRead(-A0);
}
```

Figura 3.15: Fragmento de código que genera las 2048 salidas útiles y lee el valor analógico para procesarlo por ordenador. (No funcionó correctamente debido a los comentados problemas con la velocidad).

La línea de código introducida realiza la lectura del valor que se encuentra en el puerto analógico A0. El signo negativo tiene como objetivo cambiar de signo la señal para obtener una señal que crezca en el sentido positivo de voltaje, pues como se ha visto con anterioridad los fotodiodos ejercen un voltaje de salida negativo. Los diferentes valores de salida pueden ser guardados en una variable cualquiera como viene a ser “valor”.Los datos almacenados en esa variable han sido convertidos a formato digital mediante el empleo de un conversor

analógico-digital, lo que hace que estos datos pueden ser procesados posteriormente como cualquier otro

3.2.4.4. Lectura de los 14 valores dummy finales:

Una vez acabados de leer los valores de los 2048 fotodiodos, es necesario realizar 14 ciclos más para leer las 14 salidas dummy del final.

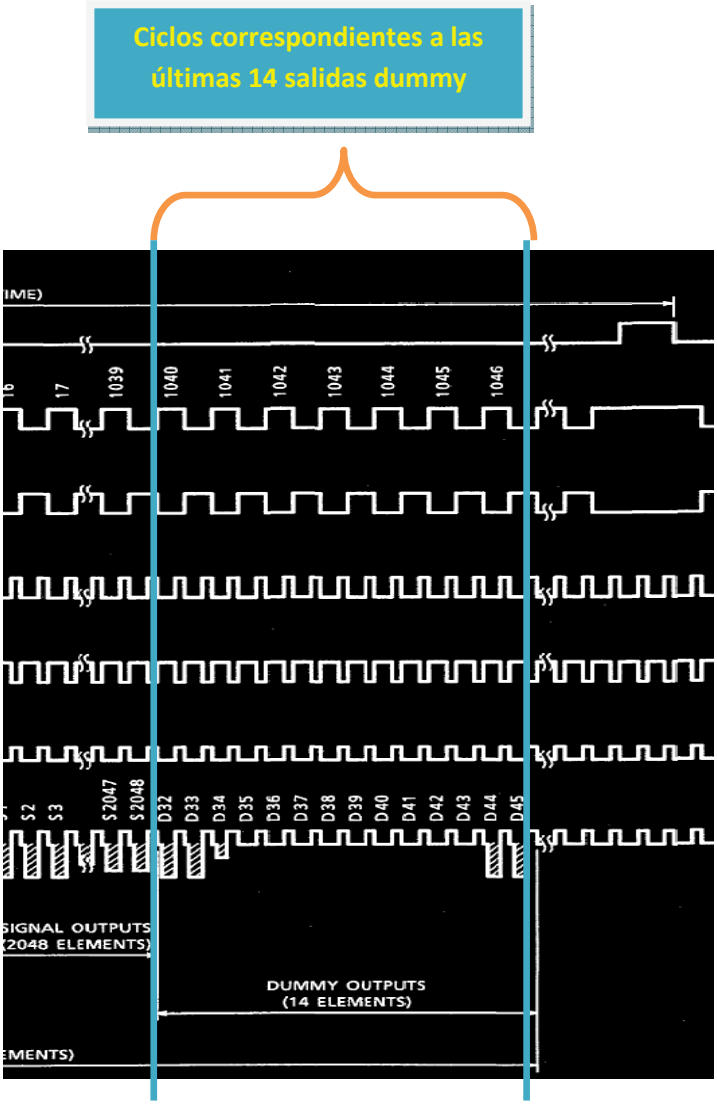

Figura 3.16: 14 valores dummy finales (Adaptado de la hoja de datos del fabricante).

Para simular estas salidas, se ha procedido de igual manera que para los ciclos anteriores. En la figura 3.17 se puede observar el fragmento de código que ha permitido simular estos ciclos de lectura dummys.

```
for(int i=0; i<7; i++){

  PORTD= B00010000;
  PORTD= B01010000;
  PORTD= B11010000;
  PORTD= B10010000;
  PORTD= B10100000;

  PORTD= B00100000;
  PORTD= B01100000;
  PORTD= B11100000;
  PORTD= B10100000;
  PORTD= B10010000;

  }
 Serial.println("Read cycle complete");
}
```

Figura 3.17: Fragmento de código encargado de generar los 14 pulsos dummy finales.

Una vez realizada la lectura de estos valores dummy, se puede dar por concluido el ciclo de lectura. Dicho evento podrá apreciarse en la consola mandando a imprimir por pantalla el mensaje “Read cicle complete”.

Los valores de salida del sensor nunca fueron leídos por la función analogRead() de Arduino, ya que el tiempo que se dedicaba a realizar esta sentencia descuadraba la forma que debían presentar las señales de control del sensor. Por eso se decidió realizar las lecturas mediante el software de cálculo Matlab.

3.3. Control del ILX511:

El procedimiento seguido con el ILX511 es similar al que se realizó con el TCD1201d. Se realiza un sketch de ARDUINO que reproduzca las señales de control que vienen detalladas en la hoja de datos del fabricante y que se pueden contemplar en la figura 3.18.

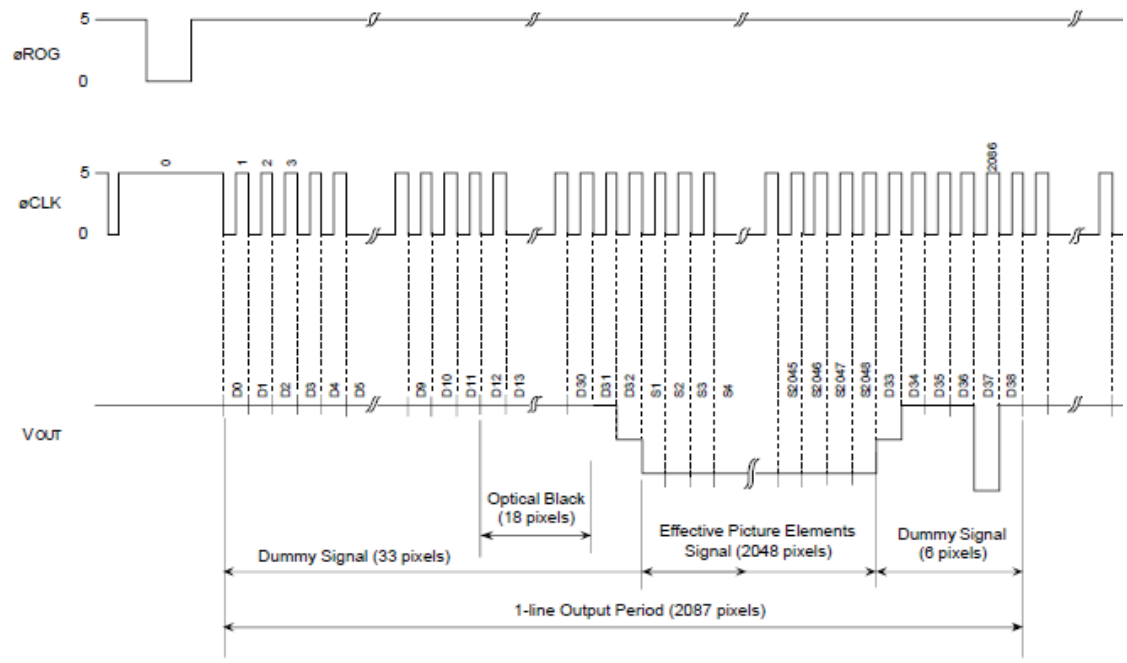

Figura 3.18: Diagrama de tiempos del ILX511 con S/H (Hoja de datos del fabricante).

3.3.1. Comentarios y definiciones:

En primer lugar se han declarado las variables de las señales de control y se les ha asignado el pin de Arduino al que irán asociadas:

```
// Se declaran las señales de control que se utilizarán y
// se les asigna el entero del pin al que estaran asociadas
int rog = 12;
int clk = 13;
```

Figura 3.19: Declaración de la señales de control y asignación del pin al que irán asociadas

3.3.2. Función setup:

Lo primero que se ha establecido en el setup es la velocidad de las comunicaciones seriales entre la placa y el computador, estableciéndose estas a una velocidad de 9600

Además se han configurado los pines asignados a la señales de control como salidas de la placa. Además se les ha asignado a estos un valor inicial de 5V.

```
void setup() {
  //Establecemos una velocidad de
  //comunicación serial de 9600Mbps
    Serial.begin(9600);
  // Confiuramos os pines de las señales de control como salidas
  pinMode(clk, OUTPUT);
  pinMode(rog, OUTPUT);
  //Para empezar el ciclo las establecemos en alta
  digitalWrite(clk, HIGH);
  digitalWrite(rog,HIGH);
}
```

Figura 3.20: Fragmento de código correspondiente a la función setup.

3.3.3. Función loop():

Observando el diagrama de tiempos del ILX511 se aprecian dos regiones bien diferenciadas: una antes de empezar el ciclo de lectura que tiene como objetivo preparar el sensor para que lea correctamente y en la cual no se proporciona salida alguna; y una segunda que abarca todo el ciclo de lectura en la cual se producen la salida de todos los valores, incluyendo tanto dummys como valores de lectura útiles.

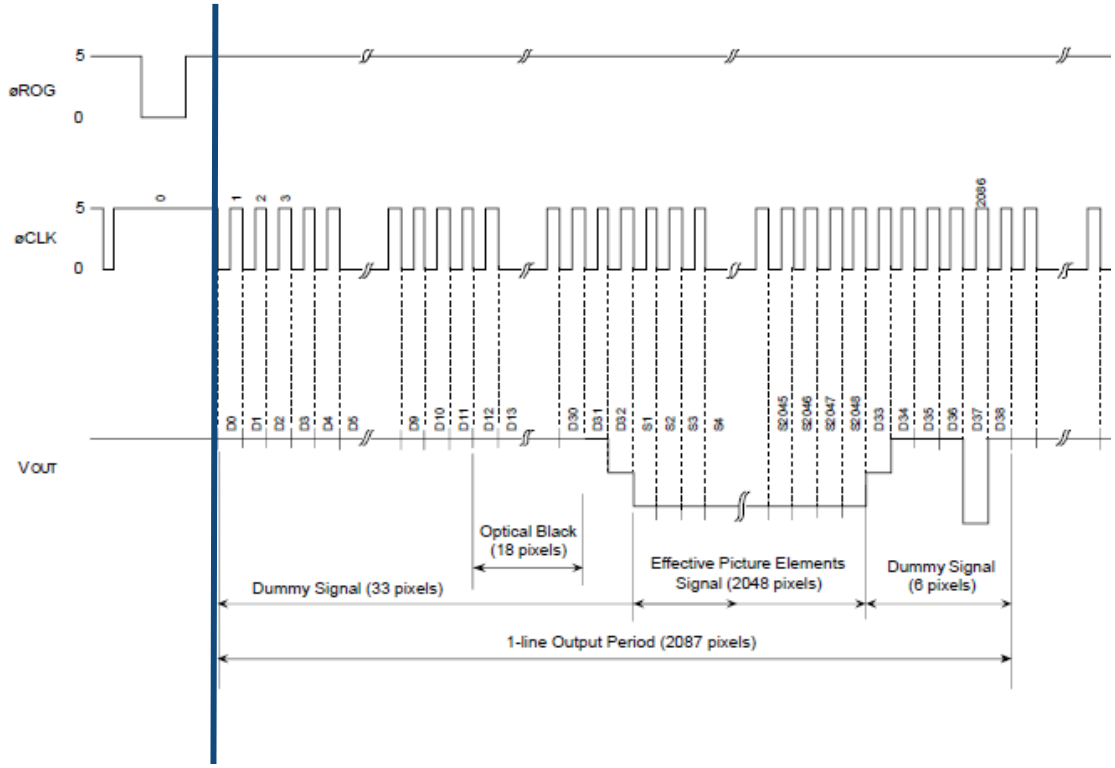

Figura 3.21: Diagrama de tiempos del ILX511 (Hoja de datos del fabricante).

Lo primero que se ha configurado en la función loop es el primer tramo de preparación del sensor, tramo que se puede ver delimitado a la izquierda de la línea vertical de la figura 3.21.

```
// Funcion loop que se ejecuta de manera cíclica:
void loop() {
  //Configuramos el pulso inicial
  digitalWrite(clk, HIGH);
  digitalWrite(rog,HIGH);
  delayMicroseconds(1);

  digitalWrite(clk, LOW);
  delayMicroseconds(1);

  digitalWrite(clk, HIGH);
  delayMicroseconds(3);

  digitalWrite(rog,LOW);
  delayMicroseconds(5);

  digitalWrite(rog,HIGH);
  delayMicroseconds(3);
```

Figura 3.22: Fragmento de código correspondiente al tramo de preparación del sensor.

Como se puede observar en la figura 3.22, en este tramo se colocan los pines a alta o baja según especifique el diagrama de tiempos, y se espera una serie de tiempo mediante el empleo de delays, que incorporan el ancho de los pulsos.

Una vez ejecutados estos pulsos iniciales, comienza el ciclo de lectura del sensor. Como dicho ciclo es uniforme en su totalidad se ha realizado mediante el empleo de un bucle for de 2100 iteraciones. Aunque un periodo de lectura consta de 2087 pulsos, el fabricante indica que ese es el valor mínimo de pulsos que debe haber, por lo tanto se han realizado 13 más por seguridad.

```
//Realizamos el ciclo de lectura
for(int i=0;i<2100;i++){
digitalWrite(clk,LOW);
delayMicroseconds(1);
digitalWrite(clk, HIGH);
delayMicroseconds(1);
}
```

Figura 3.23: Generación de los pulsos correspondientes a los 2100 valores de lectura (útiles + dummy + adicionales)

Una vez se han analizado ambos códigos se puede observar que el código de control del ILX511 es más sencillo de implementar debido a que posee menos señales de control que se deben configurar, lo cual hace que no sea necesario hacer uso de un registro de salida como el PORTD, sino que se puedan simular las señales mediante el empleo de pines de manera individual.

Los resultados obtenidos de este sensor fueron mejores que los del TCD1201D, debido a que el ILX511 permite realizar lecturas más lentas. Sin embargo en el momento de entrega de la memoria no se pudo comprobar al 100% la veracidad total de los datos debido a problemas de saturación del sensor que serán comentados más adelante.

3.4. Cálculo del máximo con ARDUINO:

Para determinar la distancia a la que se encuentra situado el objeto reflejado por el láser es imprescindible determinar cuál es el diodo que recibe mayor cantidad de luz. Es útil guardar también una serie de valores próximos al máximo para mejorar la resolución del sensor mediante un simple cálculo matemático que resuelva el triángulo formado por el sensor, el láser y el objeto.

Arduino carece de memoria suficiente como para almacenar todos los datos, motivo por el cual es necesario realizar el cálculo del máximo mientras se realizan las lecturas, analizando los valores de lectura mientras estas se van produciendo.

Este segmento de código no se puedo ejecutar, pues se empleaba un tiempo de procesamiento demasiado alto que afectaba a la forma de las señales de control que debían generarse de manera casi simultánea a este cálculo. La velocidad de procesamiento requerida debía ser superior a la del ATMEGA328 implementado en ARDUINO UNO, la placa utilizada en la experimentación, por lo tanto es necesario utilizar otro modelo ARDUINO que incorpore un microcontrolador más rápido que permitarealizar la lectura y el tratamiento de esta información sin afectar al tiempo de lectura del sensor.

Otra posible solución es realizar el prototipo con un sensor que permita frecuencias de trabajo algo menores, motivo por el cual se procedió a experimentar con el ILX511 después de haberlo hecho con el TCD1201D.

En el presente apartado se expondrán los algoritmos necesarios para calcular la posición del diodo sobre la que incide una mayor cantidad de luz, y por lo tanto será el diodo sobre el que incida la luz del láser reflejado.

El código desarrollado tiene como objetivo almacenar el valor del diodo sobre el que incide una mayor cantidad de luz, y los valores de los 5 diodos anteriores y los 5 posteriores. Como resultado se obtiene un array de 11 valores, las 5 primeras posiciones del vector se corresponden con los valores de los 5 diodos previos al máximo. El valor 6 se corresponde con el valor del diodo máximo. Finalmente las 5 últimas posiciones del vector almacenan el valor de los 5 diodos siguientes al diodo que posee el valor máximo.

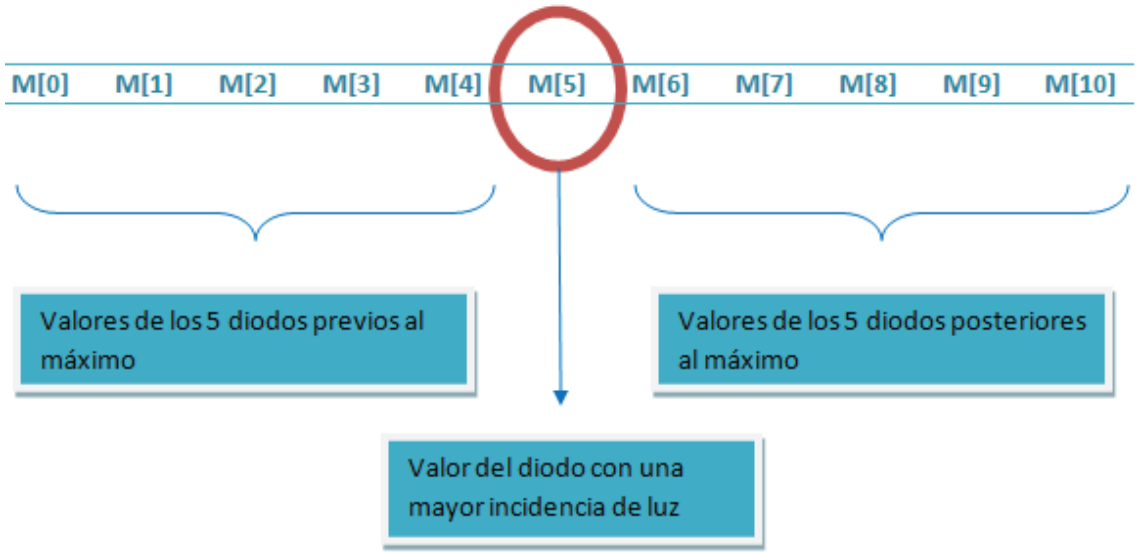

Figura 3.24: Estructura del vector máximo

Aunque el Buffer posee una extensión de 11 posiciones en la aplicación que nos ocupa, se ha decidido utilizar en su nomenclatura una extensión de [2N +1], como se puede observar en la figura:__.

```
int N = 5;
int BuffMaximo[2*N+1];
int Buffer[N];
```

Figura 3.25: Definiciones iniciales

Esta nomenclatura permite variar la extensión del BuffMaximo variando únicamente el valor de la N, variable que se emplea numerosamente a lo largo del código. Como se puede observar en la figura 3.25 es importante declarar antes la variable N y asignarle el valor 5. En la ejecución del código todos los valores que hagan referencia a valores de extensiones estarán referenciados a esta variable. Por lo tanto si se deseara obtener una mayor precisión que la obtenida pesando un Buffer de 11 valores, sólo habría que aumentar el valor asignado a la N.

3.4.1. Buffer cíclico

Para poder realizar las asignaciones de este vector es necesario apoyarse en un buffer cíclico que vaya almacenando los 5 últimos valores leídos.

Este buffer debe guardar los valores de luz únicamente de los últimos 5 fotodiodos leídos. Para liberar espacio se irán sobrescribiendo los datos más antiguos, permitiendo así disponer de las últimas 5 lecturas únicamente.

Los valores que se van almacenando en el vector BuffMax provienen de la lectura analógica de los valores de salida de cada uno de los fotodiodos del sensor. Para realizar la lectura de un valor analógico con ARDUINO hay que hacer uso de la función analogRead(). Dicha función recibe como argumento el nombre del pin analógico sobre el que se desea realizar la lectura. En el caso que nos ocupa la salida del sensor TCD1201D se encuentra conectada con el pin A0, que ha sido definido en la estructura del sketch como OS_PIN, ya que se corresponde con la salida del TCD1201D denominada OS.

Para conseguir que el buffer cíclico funcione correctamente su índice debe incrementarse de manera cíclica en el rango 0 a 4 a medida se van realizando sucesivas lecturas de los fotodiodos. Esto se consigue colocando dentro de los corchetes que definen el índice la operación j%N. Esta operación devuelve como resultado el resto de la división de j entre N. Esta operación permite que funcione correctamente el buffer cíclico ya que su resultado es el índice que debe ocupar el siguiente valor leído.

El fragmento de código correspondiente al buffer cíclico se puede ver a continuación:

```
Buffer[j%N]=  - analogRead(OS_PIN);
j++;
```

Figura 3.26:Fragmento de código que implementa un buffer cíclico

La variable contadora j hace referencia al número de lectura útil que se está realizando. Esta variable es inicializada a 0 fuera del bucle for que realiza la lectura de los valores

Como se puede comprobar en la tabla () el buffer cíclico presenta el comportamiento esperado:

j	0	1	2	3	4	5	6	7	8
j%N	0	1	2	3	4	0	1	2	3

Tabla 3.3: Desarrollo de los índices del buffer cíclico

3.4.2. Cálculo del BuffMaximo

3.4.2.1. Cálculo del valor máximo del diodo

A medida se van leyendo los valores de salida de los diodos vamos comparando este valor con una variable máximo donde se ha guardado el máximo de los anteriores valores leídos. Para esto hacemos uso del siguiente fragmento de código:

```
if (valor > maximo){
 maximo = valor;
 posmaximo = j;
 BuffMaximo[N]= valor;
 for(int k=0; k<N; k++){
 BuffMaximo[k] = Buffer[(j+k)%N];
 }
Maxpost = N;
}
if (Maxpost > 0){
  BuffMaximo[2*N+1-Maxpost] = valor;
  Maxpost--;
}
Buffer[j%N]=  - analogRead(OS_PIN);
j++;
```

Figura 3.27: Código del cálculo del máximo

Como se puede ver en la figura 3.27, para localizar el máximo se ha implementado un bloque de código if que tiene como argumento la condición de que el siguiente valor leído (asignado previamente en la variable valor) sea mayor que el valor guardado en máximo. Cuando se cumpla esta condición:

- La variable valor pasa a ser el nuevo máximo
- La posición del diodo que tiene este valor se guarda en la variable posmaximo
- El nuevo máximo es guardado en la posición N del vector BuffMaximo (La 5 en el caso que nos ocupa).

3.4.2.2. Asignación de los 5 valores previos al máximo

Una vez detectado un máximo, se procede a rellenar los valores 0 a 5 del BuffMaximo. Estas asignaciones se hacen con el fragmento de código de la figura 3.28.

```
if (valor > maximo){
 maximo = valor;
 posmaximo = j;
 BuffMaximo[N]= valor;
 for(int k=0; k<N; k++){
 BuffMaximo[k] = Buffer[(j+k)%N];
 }
Maxpost = N;
}
if (Maxpost > 0){
  BuffMaximo[2*N+1-Maxpost] = valor;
  Maxpost--;
}
Buffer[j%N]=  - analogRead(OS_PIN);
j++;
```

Figura 3.28: Fragmento de código que asigna los 5 valores previos al máximo.

Los valores de los 5 diodos justo anteriores al máximo se encuentran guardados en el Buffer cíclico. Para asignarlos a las 5 primeras posiciones del BuffMaximo se ha hecho uso de un bucle for que va realizando de manera iterativa este proceso en 5 pasos.

Cuando se detecta el máximo, el Buffer cíclico tiene como valor más reciente el correspondiente al fotodiodo justo anterior al máximo; este valor estará guardada en la posición j%N del Buffer como se explicó con anterioridad, pudiendo ser esta posición cualquiera de las 5 que posee, lo que dificulta la asignación de estos valores al BuffMaximo. Para poder resolver este problema y hacer corresponder los índices correctamente, se ha hecho uso de la operación (j+k) % N en los índices del Buffer cíclico.

Un ejemplo del funcionamiento de esta operación puede verse en la tabla 3.4. En ella se observa el comportamiento que tiene el algoritmo suponiendo que el valor máximo se da en la posición 7.

K	j	(j-k)%N
0	2	2
1	3	3
2	4	4
3	5	0
4	6	1

Tabla 3.4: Iteraciones que demuestran el funcionamiento del algoritmo implementado.

Como se puede observar en este ejemplo que el máximo se encuentra en 7, el último valor del Buffer cíclico fue guardado en la posición [(j-1)%N]= [6%5]= 1. Este significa que el valor más antiguo se encuentra en la posición 2 del Buffer cíclico, y éste es el que debe ser guardado en la posición 0 del BuffMaximo. Como se puede ver en la tabla:__, a las siguientes posiciones de memoria se le van asignando los siguientes valores del Buffer cíclico en orden descendente de antigüedad.

3.4.2.3. Asignación de los 5 valores posteriores al máximo

Una vez asignados los 6 primeros valores del BuffMaximo, se procede a rellenar con los 5 posteriores al máximo. Es importante que esta parte del código se ejecute únicamente los 5 valores posteriores al máximo, sin embargo dicho fragmento de código no puede ir incluido dentro del bloque if que identifica el máximo como se hizo con los 5 valores previos, pues los 5 valores siguientes no tienen por qué ser máximos. La solución adoptada para lograr este objetivo se puede observar en el fragmento de código descrito en la figura 3.29.

```
if (valor > maximo){
 maximo = valor;
 posmaximo = j;
 BuffMaximo[N]= valor;
 for(int k=0; k<N; k++){
 BuffMaximo[k] = Buffer[(j+k)%N];
 }
Maxpost = N;
}
if (Maxpost > 0){
  BuffMaximo[2*N+1-Maxpost] = valor;
  Maxpost--;
}
Buffer[j%N]=  - analogRead(OS_PIN);
j++;
```

Figura 3.29: Fragmento de código encargado de asignar los 5 valores posteriores al máximo.

Para realizar las asignaciones de los 5 valores posteriores al máximo se ha hecho uso de una variable auxiliar llamada Maxpost. Esta variable se reinicia a 5 cada vez que se encuentra un nievo máximo. Cuando esto ocurre el programa sigue ejecutando el siguiente bloque if, que posee como condición que se cumpla que esta variable contadora sea mayor que 0. Este bloque if contiene la asignación de estos valores posteriores. Para conseguir que las asignaciones se realicen en el orden deseado se ha hecho uso de la operación [2N+1-Maxpost] en el índice del BuffMaximo. En la tabla:__ se expone el comportamiento seguido por los índices del buffer máximo.

N	5	4	3	2	1
2N+1-Maxpost	6	7	8	9	10

Tabla 3.5: Iteraciones que demuestran el correcto funcionamiento del algoritmo encargado de asignar los 5 valores posteriores al máximo.

Como se puede observar en la tabla3.5 cada iteración en la cual el valor leído no sea superior al máximo, se decrementará en 1 el valor de Maxpost, lo que hace que el incide de

BuffMaximo se vaya incrementando en 1 con cada iteración. Esta operación termina de rellenar los espacios de memoria del BuffMaximo.

Después de realizar todo el procedimiento descrito en este código, se puede saber dónde incide la máxima intensidad de luz, y los píxeles de su alrededor, por lo que sería posible hacer un cálculo de resolución subpixel utilizando esta información.

Para ellos se realiza una interpolación considerando que los valores de voltaje próximos al máximo siguen una distribución normal.

4. Elección de parámetros.

Como ya se ha comentado anteriormente, el sensor de distancia propuesto en el presente documento realiza el cálculo de la distancia mediante un algoritmo geométrico que relaciona la posición del diodo sobre el que impacta una mayor cantidad de luz con la distancia del objeto detectado. Este cálculo resuelve el triángulo formado entre el láser, el sensor y el objeto del cual se quiere conocer la distancia a la que se encuentra.

Existen dos parámetros del triangulo que son elegidos en función de los requerimientos de medición de la aplicación para la cual se utiliza el prototipo. Dichos parámetros son el ángulo que forma el láser con la recta que contiene al sensor CCD y la distancia que existe entre el láser y el sensor CCD. La elección de estos parámetros afectara a los valores de las distancias máximas y mínimas que se pueden medir, así como a la resolución del sensor.

- Modificar la distancia entre el laser y el sensor CCD afecta a las distancias máximas y mínima medibles, sin embargo la resolución no se ve afectada.
- Modificar el ángulo que forma el láser con la recta que contiene al sensor CCD afecta tanto a la resolución como a las distancias máxima y mínima medibles.

Estos parámetros serán escogidos según los requerimientos que exijan las aplicaciones para las que se use el sensor, siendo unos valores convenientes para unas aplicaciones y otros muy diferentes los óptimos para otras aplicaciones.

En el presente apartado se estudiará la incidencia de estos parámetros a los valores de distancias medibles y resolución.

4.1.Valor mínimo medible:

El valor mínimo medible se obtiene cuando el fotodiodo con mayor incidencia de luz es el que se encuentra en la posición 1. Este caso es representado en la figura 4.1.

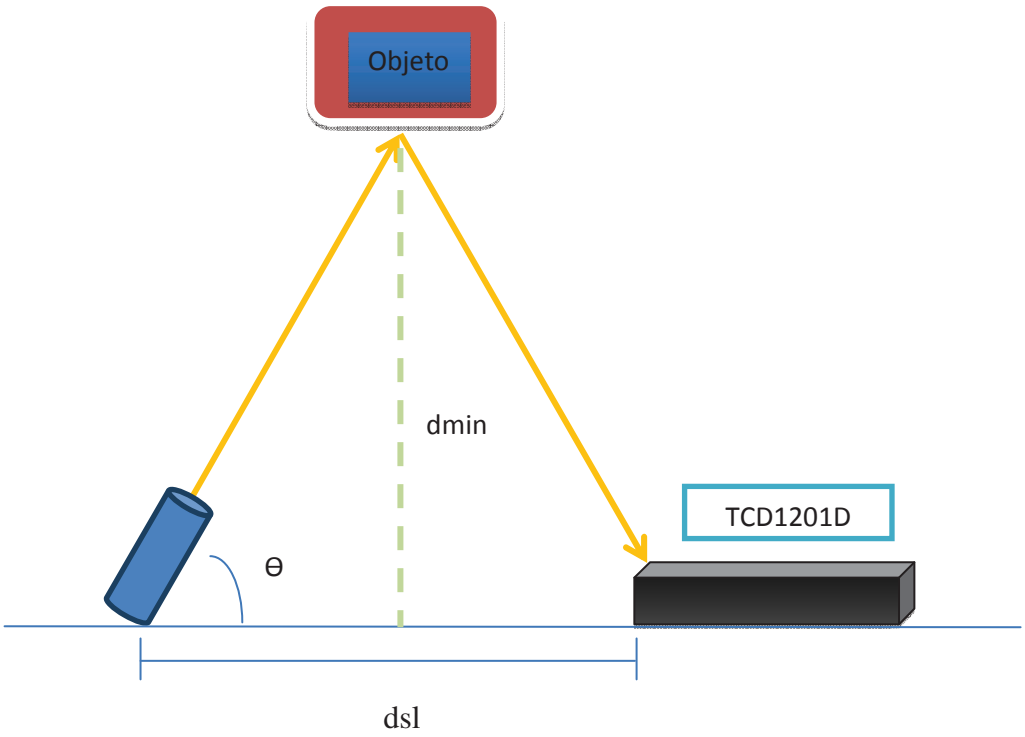

Figura 4.1: Representación gráfica del menor valor medible.

DóndeΘ es el ángulo de inclinación del sensor respecto al sensor y dsl es la distancia existente entre el láser y el sensor.

Haciendo uso de la relación trigonométrica de la tangente se obtiene la expresión:

$$tg\Theta = \frac{2dmin}{(dsl + 0.646x10^{-2})}$$

Ecuación 4.1.

Nótese que en la ecuación 4.1 se le ha añadido el valor $0.646\,x10^{-2}m$ a la distancia láser sensor. Este valor es el correspondiente a la longitud del borde del encapsulado situado antes del primer diodo, y es necesario tenerlo en cuenta en los cálculos para obtener mayor precisión en las medidas.

Despejando dmin:

$$dmin = \frac{tg\Theta\ (dsl + 0.646x10^{-2})}{2}$$

Ecuación 4.2.

Como se puede ver en la ecuación 4.2, la distancia mínima medible depende directamente de la distancia sensor láser y del ángulo de inclinación del sensor.

4.2. Valor máximo medible:

La máxima distancia medible se obtiene cuando el fotodiodo con mayor incidencia de luz es el 2048. Este caso aparece contemplado en la figura .

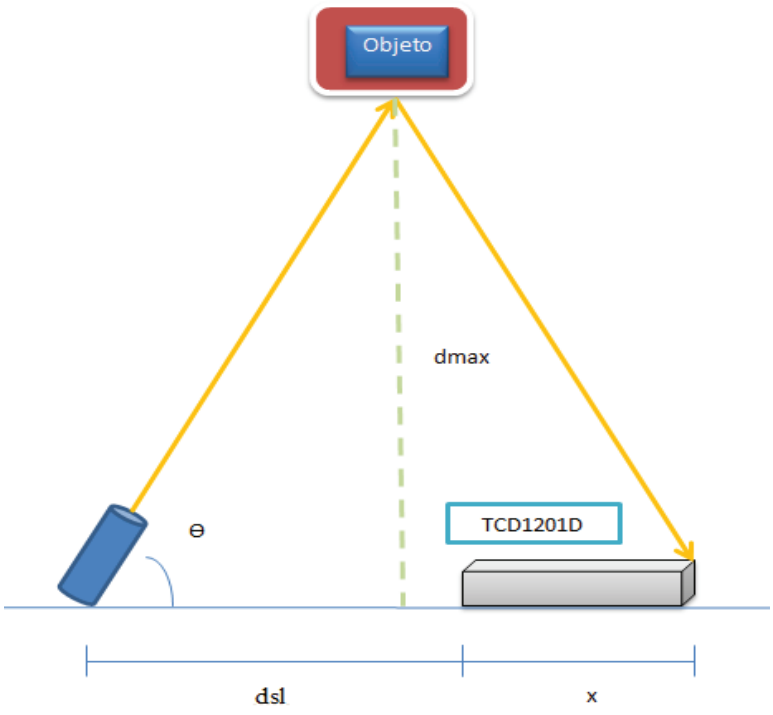

Figura 4.2: Representación gráfica del mayor valor medible.

La variable x en este caso contempla toda la extensión del sensor:

$$x= 2048 \text{ x } 14 \text{ x} 10^{-6} + 0.646x10^{-2}\text{cm}$$

Ecuación 4.3.

El procedimiento de resolución es el mismo que en el caso anterior:

$$tg\Theta = \frac{2\text{dmax}}{dsl + 2048x14x10^{-6} + 0.646x10^{-2}}$$

Ecuación 4.4.

Reordenando la ecuación: 4.4 se obtiene la expresión:

$$dmáx = \frac{tg\Theta(dsl + 2048x14x10^{-6} + 0.646x10^{-2})}{2}$$

Ecuación 4.5.

Como se puede observar en la ecuación: 4.5, la distancia máxima también depende de la distancia sensor láser y del ángulo de inclinación del láser.

4.3. Efecto de los parámetros en la resolución del sensor:

La resolución del sensor se define como el mínimo cambio necesario en la distancia del objeto para apreciar un cambio en la salida. En el caso del sensor contemplado en el presente documento ésta vendrá definida como el mínimo valor de distancia que provocará que se ilumine el fotodiodo de la posición siguiente.

Para determinar este valor es importante saber el rango de valores que se permite medir. El rango de valores medibles es el que se encuentra en el intervalo entre dmin y dmáx.

$$dmáx - dmin = \frac{tg\Theta(dsl + 0.646x10^{-2} + 2048x14x10^{-6}) - tg\Theta(dsl + 0.646x10^{-2})}{2}$$

Ecuación 4.6

Operando:

$$dmáx - dmin = \frac{tg\Theta\,[2048x14x10^{-6} + \cancel{0.646x10^{-2}} - \cancel{0.646x10^{-2}} + \cancel{dsl} - \cancel{dsl}]}{2}$$

Se obtiene la ecuación 4.7

$$dmáx - dmin = \frac{tg\Theta\,(2048x14x10^{-6})}{2}$$

Ecuación 4.7.

Como se disponen de 2048 posibles posiciones de salida, el mínimo valor medible será:

$$Resolución = \frac{dmáx - dmin}{Nº\ posiciones} = \frac{tg\Theta\ (\cancel{2048}x14x10^{-6})}{2} /\cancel{2048}$$

$$Resolución = \frac{tg\Theta\ (14x10^{-6})}{2}$$

Ecuación 4.8.

Como se puede observar en la ecuación 4.8 la resolución depende únicamente de la orientación del láser. Cuanto mayor sea el ángulo de orientación del láser mayor será el cambio que deberá producirse en la entrada para apreciar un cambio en la salida, esto significa que a mayor ángulo menor resolución se dispondrá.

4.4. Ejemplos:

A continuación se muestran una serie de ejemplos de cómo afecta la elección de los parámetros a las distancias medibles y a la resolución.

4.4.1. Caso 1: Orientación para medir hasta 10 cm con dsl= 5cm.

En primero lugar afrontaremos el estudio de un caso en el cual se medirán distancias relativamente cortas (distancia máxima de 10cm). En la figura 4.3 se puede observar una descripción gráfica del problema que se plantea:

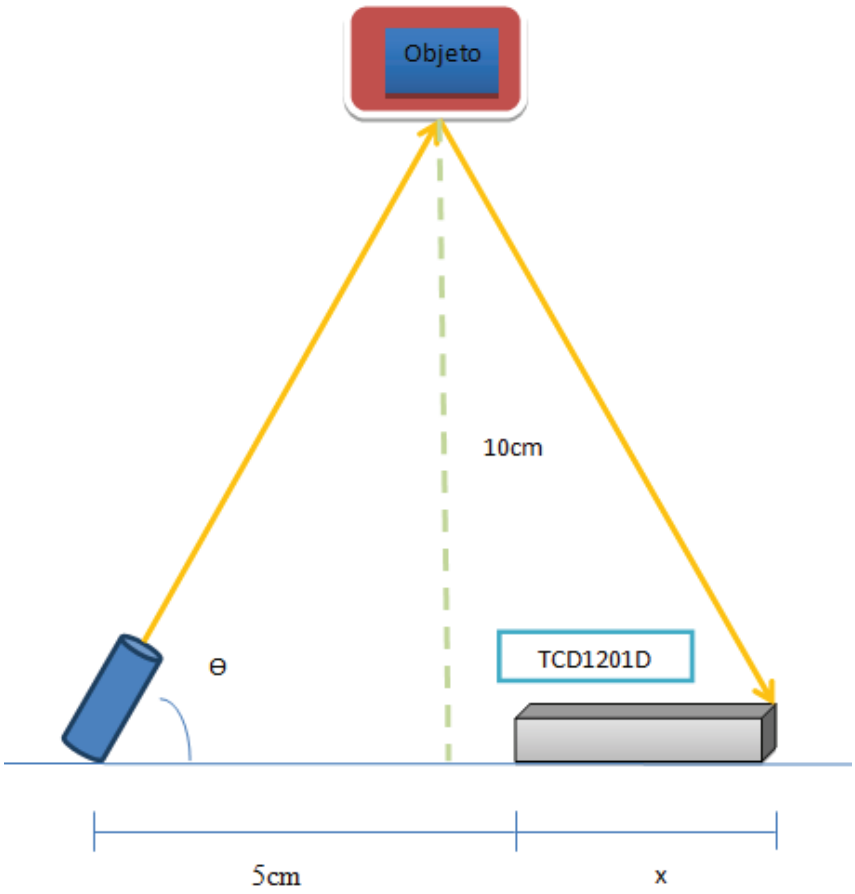

Figura 4.3: Representación gráfica del ejemplo propuesto.

Aplicando la ecuación 4.4, con dmax = 10cm y dsl = 5cm, se obtiene:

$$tg\Theta = \frac{2x10x10^{-2}}{5x10^{-2} + (2048x14x10^{-6}) + 0.646x10^{-2}}$$

Despejando el ánguloΘ:

$$\Theta = arctg(\frac{2x10x10^{-2}}{5x10^{-2} + (2048x14x10^{-6}) + 0.646x10^{-2}})$$

$$\Theta = 66.97^{o}$$

Las distancia mínima medible se puede calcular haciendo uso de la expresión, donde el ángulo $\Theta = 66.97^{o}$

$$dmin = \frac{tg66.97\ (5x10^{-2} + 0.646x10^{-2})}{2} = 0.0664\ \text{m} = 6.64\ \text{cm}$$

La resolución del prototipo con estos parámetros será de:

$$Resolución = \frac{dmáx - dmin}{Nº\ posiciones} = \frac{10 - 6.64}{2048} = 1.641\ x\ 10^{-3} cm = 16.41\ \mu m$$

Con los resultados obtenidos se observa que se puede medir en el rango de los 6.64cm y los 10cm con una resolución de16.41 μm , haciendo uso de una inclinación del láser de 66.97^{o} y una distancia láser sensor de 5cm.

Ahora expondremos un caso con unos parámetros distintos para observar su influencia en las distancias medibles y en la resolución del aparato.

4.4.2. Caso2: Inclinación del láser de 70º, y distancia sensor láser de 7cm.

En este caso las mediciones que realizarán el sensor serán distancias mucho mayores que en el caso anterior. Al contrario que en el ejemplo anterior, se han fijado ya los valores de inclinación del láser y la distancia sensor láser, y lo que se hará es ver cómo afectan estas elecciones a los valores de distancias medibles y resolución, y se compararán los resultados con el caso anterior.

En la figura: 4.4 se puede observar una descripción gráfica del problema:

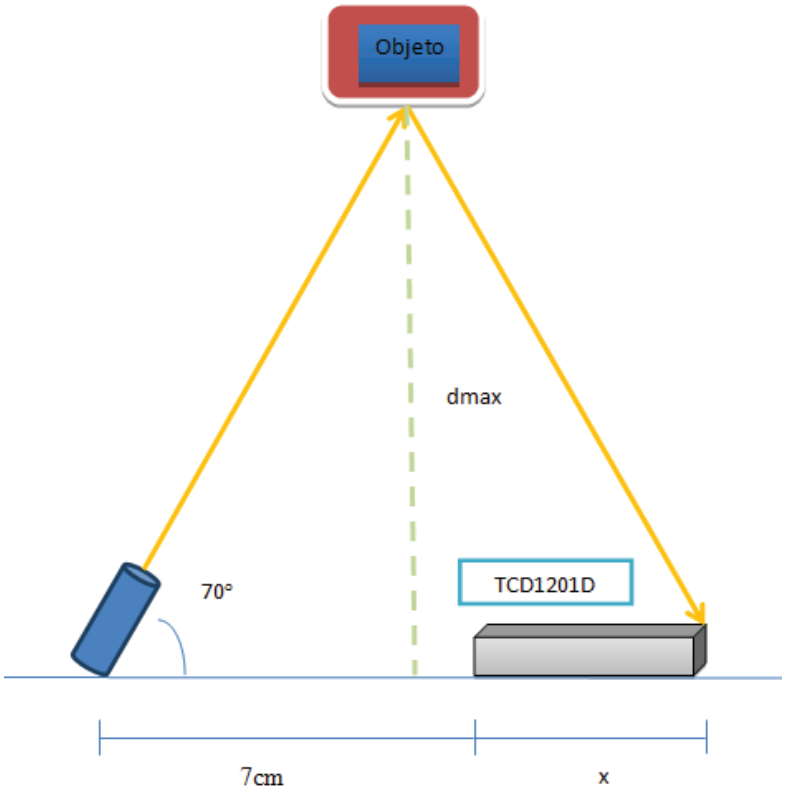

Figura 4.4: Representación gráfica del ejemplo propuesto.

En primer lugar se hallará la distancia máxima medible haciendo uso de la expresión 4.5

$$dmáx = \frac{tg70(7x10^{-2} + 2048x14x10^{-6} + 0.646x10^{-2})}{2} = 0.144m = 14.4cm$$

Se utiliza la expresión 4.2 para hallar la distancia mínima medible:

$$dmin = \frac{tg70(7x10^{-2} + 0.646x10^{-2})}{2} = 0.105 \text{ m} = 10.5 \text{ cm}$$

La resolución del prototipo con estos parámetros será de:

$$Resolución = \frac{dmáx - dmin}{Nº\ posiciones} = \frac{14.4 - 10.5}{2048} = 1.904\ x\ 10^{-3}cm = 19.04\ \mu m$$

Por lo tanto un prototipo con 70º de inclinación del láser y una distancia láser sensor de 7 cm se pueden medir distancias comprendidas entre los 10.5cm y los 14.4cm con una resolución de 19.04 μm.

Comparando ambos ejemplos se observa que en el primer caso se podía medir un rango de medidas más pequeño que en el segundo, siendo además estas distancias más próximas al sensor. En el segundo caso se mide un rango de medidas más amplio en distancias más lejanas al sensor, pero se observa que se pierde resolución respecto al primer caso.

5. Exposición y tratamiento de resultados

Como se ha comentado con anterioridad se han producido dificultades a la hora de leer los valores de salida del sensor con Arduino ya que se ralentizabamucho el programa, lo cual hacia que las señales de entrada del TCD1201D sean excesivamente lentas y que además los anchos de pulso no fueran simétricos durante todo el proceso. Para poder realizar las lecturas con Arduino se requiere una velocidad de procesamiento bastante alta que no se pudo conseguir con el procesador del Arduino Uno. Para poder solventar el problema del tratamiento de datos se realizó un código en el software de cálculo Matlab, que será descrito en el presente apartado de la memoria.

Una solución que se ha adoptado para leer las salidas del sensor ha sido conectarlas al osciloscopio digital HAMEG. Este osciloscopio permite exportar los datos reflejados en pantalla para tratarlos mediante un PC. En la experimentación llevada a cabo el tratamiento de los datos se ha realizado haciendo uso del software de cálculo Matlab. Mediante este software se puede implementar un programa que realice el tratamiento de los datos del sensor, esto es, calcular el diodo con una mayor incidencia de luz, y a partir de este dato y la distancia entre el láser y el TCD1201D y el ángulo entre el láser y el sensor, calcular la distancia del objeto sobre el que impacta el láser.

Todo el código mostrado a continuación ha sido diseñado para el tratamiento de datos del TCD1201D, sin embargo también podría ser utilizado un código idéntico para el tratamiento de datos del ILX511

El tratamiento de datos en Matlab consta de 3 partes:

- Importar el archivo en formato .CSV.
- Hallar la posición del diodo con mayor incidencia de luz.
- Calcular la distancia a la que se encuentra el objeto.

Para conseguir estos tres apartados mencionados se han escrito 2 funciones en scripts de Matlab:

1. Función que calcula el diodo con mayor cantidad de luz
2. Función que calcula la distancia con el objeto.

Sin embargo cabe destacar que la función encargada de realizar el cálculo trigonométrico es llamada desde dentro del script de una función principal que será la encargada de realizar todo el proceso desde que se importan los datos hasta que se entrega el valor de la distancia a la cual se encuentra el objeto. Estas funciones serán descritas con mayor profundidad en los apartados siguientes de este capítulo, sin embargo se describen a continuación aspectos comunes a todas las funciones de Matlab.

Una definición para función de Matlab sería Las funciones en MATLAB son programas que toman las variables que se les pasan (variables de entrada), realiza unos cálculos y manipulaciones con ellas y devuelve unos resultados (variables de salida). La estructura general es:

funtion [variables de salida] = nombrefuncion(variables de entrada)

Las órdenes evaluadas por la función, así como las variables intermedias creadas por estas órdenes, están escondidas, sólo son visibles las variables de entrada y salida. Esto hace que las funciones sean muy adecuadas para encapsular funciones matemáticas útiles o secuencias de órdenes que aparezcan a menudo.

MATLAB nos permite crear funciones propias en forma de archivos .m. Un archivo .m de función es similar a un archivo script, al igual que ellos son archivos de texto creados en un editor de texto. La diferencia entre ambos es que la función sólo se comunica con el espacio de trabajo a través de las variables de entrada y salida, las variables intermedias dentro de la función no aparecen ni interactúan con el espacio de trabajo de MATLAB.

Se ha creado una función de Matlab llamada valordistancia que recibe como argumento el nombre del fichero que contiene los datos que se desean tratar, la distancia existenteentre el láser y el sensor (dsl) y el ángulo de orientación del láser(theta); y que proporciona como

valor de retorno la distancia a la cual se encuentra el objeto del sensor. Mediante una serie de comandos que serán descritos a continuación se hallará la posición del fotodiodo que recibe mayor cantidad de luz. La posición de este diodo es el argumento de entrada de la función distancia, que realiza un cálculo trigonométrico para calcular la distancia entre el sensor y el objeto. La función distancia se encuentra contenida dentro de la función valordistancia

5.1. Importar el archivo en formato .CSV

Como se ha comentado con anterioridad, el osciloscopio HAMEG exporta por defecto archivos de datos separados por coma o .CSV. La primera parte del código de Matlab tiene como objetivo importar los datos del fichero .CSV para su posterior tratamiento.

Hay que tener la precaución de trabajar con el directorio en el que estén guardados los ficheros .CSV con los que desea trabajar, en caso contrario no podrán importarse

Para importar los datos de Matlab se utiliza la función propia de Matlab csvread( ). Esta función recibe como argumento el nombre del fichero de datos que se desea importar, y proporciona a la salida una variable con la extensión de esos datos:

```
function [d]= valordistancia(filename,dsl,theta)

%Se importan los datos mediante la funcion csvread
datos= csvread(filename);
```

Figura 5.1: Código encargado de importar los datos del .CSV.

Como se puede ver en la figura 5.1 esta función recibe como argumento el fichero que recibe como argumento a su vez la función principal valordistancia.

La función csvread() importa todos los datos contenidos en el archivo .CSV que se le pasa como argumento y crea una variable llamada datos donde se encuentran guardados estos valores.

Una vez importados todos los datos en una matriz de Matlab pueden ser operados como una variable cualquiera de Matlab, pudiéndose usar como argumento de cualquier función de Matlab.

El osciloscopio HAMEG realiza capturas de hasta 6000 valores de manera simultánea, por lo tanto la variable datos será un vector de dimensión 6000x2. Una columna se corresponde con el valor de voltaje de la salida OS. La otra columna se corresponde con los valores temporales en los que han sido recogidas esas muestras.

5.2. Hallar la posición del diodo con mayor incidencia de luz.

Los datos recogidos en la variable "datos" no solo se corresponden con los valores útiles de los fotodiodos, sino que también se han recogido voltajes cuando el sensor se encuentra sin leer. Para poder determinar la posición del diodo con mayor cantidad de luz recibida será necesario quedarse únicamente con los valores de voltaje de las 2048 lecturas útiles. Para esto será necesario seguir una serie de pasos que tendrán como objetivo eliminar los datos tomados cuando el sensor no se encuentra activo.

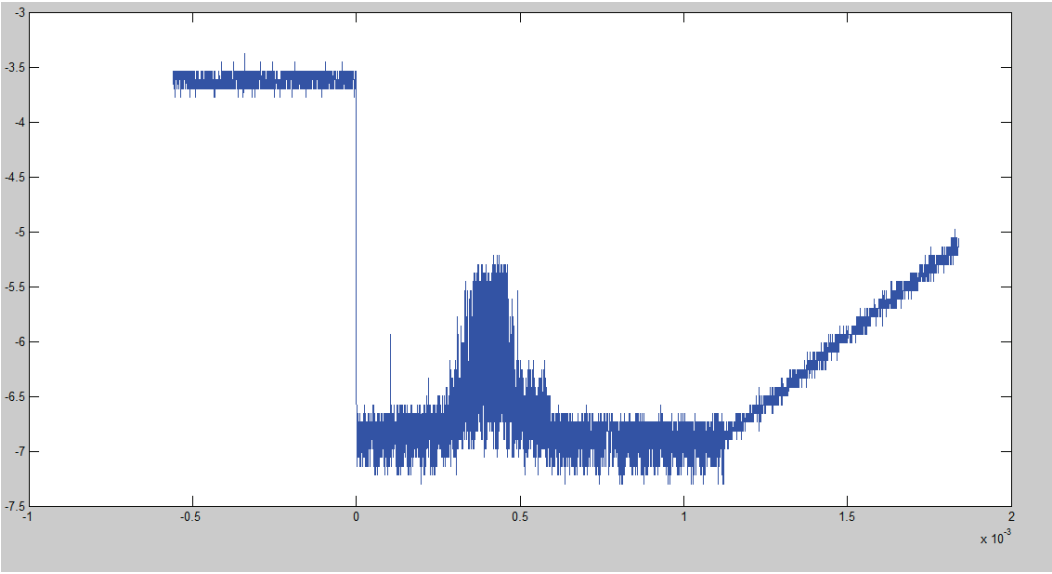

Figura 5.2: Toma de datos del osciloscopio digital HAMMEG

En primer lugar se habrá de detectar el extremo inicial del ciclo de lectura. Este extremo es fácil de detectar de manera visual en el osciloscopio HAMMEG, pues se sabe que el ciclo de

lectura se inicia cuando la señal SH pasa de un 1 lógico a un 0 lógico. Sin embargo en el procesado automático de la información presentará algunas complejidades.

Una solución para detectar el inicio de lectura puede ser cuando la señal desciende de los -5V, como se puede ver en los ciclos de lectura. Esta idea se implementa en la siguiente línea de código mostrada en la figura 5.3.

```
a= min(find(datos(:,2) < -5)); %Cogemos el primer dato del ciclo de lectura
```

Figura 5.3: Línea de código encargada de calcular el primer valor del sensor activo.

Esta línea de código le asigna a una variable a el mínimo valor de voltaje que cumpla la condición de ser menor que -5V. Esto se consigue anidando la función find con la función min.

Una vez hallado el índice del primer valor de lectura útil es necesario conocer el último para poder acotar el espacio de lectura útil, para ello se procede de una forma similar (Figura 5.4).

```
b= max(find(datos(:,2) < -7)); %Cogemos el ultimo dato del ciclo de lectura
```

Figura 5.4: Línea de código encargada de calcular el último valor del sensor activo.

Se procede de manera similar al cálculo del primer valor de lectura útil pero con la salvedad de que se ha optado por coger el máximo valor que cumpla que es menor que -7V. A partir de ahí la señal crece hasta los -5V donde se mantiene estable durante el periodo que el sensor no proporciona valores de salida útiles.

Esta forma de proceder es válida para cualquier fichero de salida que exponga los datos en la configuración matemática seguida durante la experimentación.

Los datos son guardados en una variable aparte para tratarlos por separado (Figura 5.5).

```
%Separamos los datos en un nuevo vector
datosnuevos = datos(a:b);
```

Figura 5.5: Línea de código que guarda el rango de datos a, b en una nueva variable.

Este conjunto de datos engloba todas las lecturas que realiza el osciloscopio en un ciclo de lectura del sensor, sin embargo no se poseen un total de 2048 valores útiles de voltaje, por lo que estos datos aún no pueden dar información sobre el índice del fotodiodo con mayor recepción de luz. Para conseguir adecuar esta información a los valores deseados es necesario realizar un tratamiento matemático previo. En el caso que nos ocupa se ha decidido realizar uso de la interpolación lineal. Para ello se hará uso de la función de Matlab interp1(). Esta función permite realizar múltiples tipos de tratamiento de datos.

En la figura 5.6se puede ver una descripción general que realiza Matlab de esta función.

```
Vq = interp1(X,V,Xq,METHOD) specifies alternate methods.
The default is linear interpolation. Use an empty matrix [] to specify
the default. Available methods are:

  'nearest'  - nearest neighbor interpolation
  'linear'   - linear interpolation
  'spline'   - piecewise cubic spline interpolation (SPLINE)
  'pchip'    - shape-preserving piecewise cubic interpolation
  'cubic'    - same as 'pchip'
  'v5cubic'  - the cubic interpolation from MATLAB 5, which does not
               extrapolate and uses 'spline' if X is not equally
               spaced.
```

Figura 5.6: Explicación de la función interp1 (Obtenida de la ayuda de Matlab).

Como se puede contemplar esta función permite realizar múltiples tipos de interpolaciones en función de las necesidades que se requieran en cada problema. En el caso que nos ocupa se ha dejado el argumento del método en blanco, por lo que la función operará con su definición por defecto, la interpolación lineal.

A la hora de realizar la interpolación es importante tener en cuenta que un ciclo de lectura del TCD1201D no solo engloba las 2048 lectura útiles, sino que también se produce la lectura de 32 valores dummys al inicio del ciclo y 14 valores dummys al final del ciclo. En la figura 5.7 se puede ver la línea de código correspondiente a la interpolación:

```
Puntos=1:2048+32+14;
Nuevos = interp1(1:b-a+1,datos(a:b,2),Puntos);
```

Figura 5.7: Uso de la función interp1.

Se ha decidido representar estos valores interpolados para poder observar bien los valores de un ciclo de lectura.

```
plot (Puntos,Nuevos);
```

Figura 5.8: Línea de código encargada de realizar la representación gráfica.

Una vez obtenida la matriz de datos con valores de 2098 valores de lectura se puede calcular el máximo de una manera relativamente sencilla con Matlab. Para hallarlo se hace uso de la siguiente línea de código:

```
%Hallamos el valor del máximo
max= find(Nuevos == max(Nuevos));
```

Figura 5.9: Instrucción encargada de hallar el índice máximo.

Como se puede ver en la figura 5.9, el procedimiento se basa en realizar la función find que asigna a una variable e valor de retorno de la función find. Este valor será el que cumpla la condición expresada entre corchetes, que es la posición del máximo valor de “Nuevos”.

5.3. Calcular la distancia a la que se encuentra el objeto.

Como se ha comentado con anterioridad para el cálculo de la distancia se hace uso de un cálculo trigonométrico que resuelve el triángulo formado por el sensor, el láser y el punto donde el láser incide sobre el objeto al cual se le quiere calcular la distancia en la que se encuentra.

Para hacerlo resuelve el triángulo de la figura 5.10.

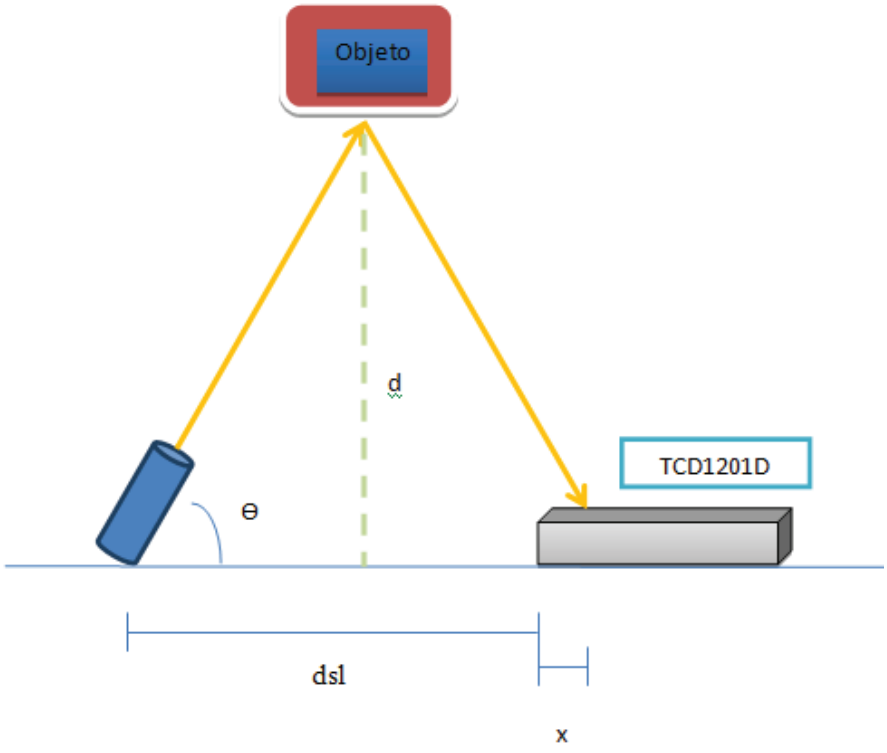

Figura 5.10 Triángulo formado por el sensor el láser y el objeto.

Una descripción detallada de la influencia de estos parámetros en las distancias medibles admisibles y la resolución se encuentra en el capítulo 4: Elección de parámetros.

Para resolver este cálculo se ha creado una función de Matlab resuelve el cálculo trigonométrico y que devuelve como valor de retorno la distancia entre el objeto y el sensor. Dicha función será llamada desde la función principal valordistancia().

El código implementado por esta función se puede observar en la figura 5.11

```
function [d]= distancia(posmax, dsl, theta)

%Se calcula la distancia resolviendo el triángulo

d= tan(theta)*(dsl+posmax*14*10^-6+0.646*10^-2)/2;
```

Figura 5.11: Función encargada de realizar el cálculo de la distancia.

Esta función recibe un único parámetro de entrada: la posición del fotodiodo con mayor incidencia de luz. Conociendo este valor, la distancia entre el sensor y láser, y el ángulo de orientación del láser se puede resolver el algoritmo, obteniendo como resultado la distancia a la cual se encuentra localizado el objeto.

Esta función es llamada desde el código de la función valordistancia mediante la línea de código que aparece en la figura 5.12

```
d= distancia(posmax, dsl, theta);
```

Figura 5.12: Llamada de la función distancia().

5.4. Caso experimental:

Para probar si el código funciona de manera correcta se ha decidido realizar una prueba experimental en la cual se han importando con Matlab los datos de salida del TCD1201D cuando se iluminaba con el láser en uno de sus puntos.

La muestra experimental utilizada se puede observar en la figura 5.13

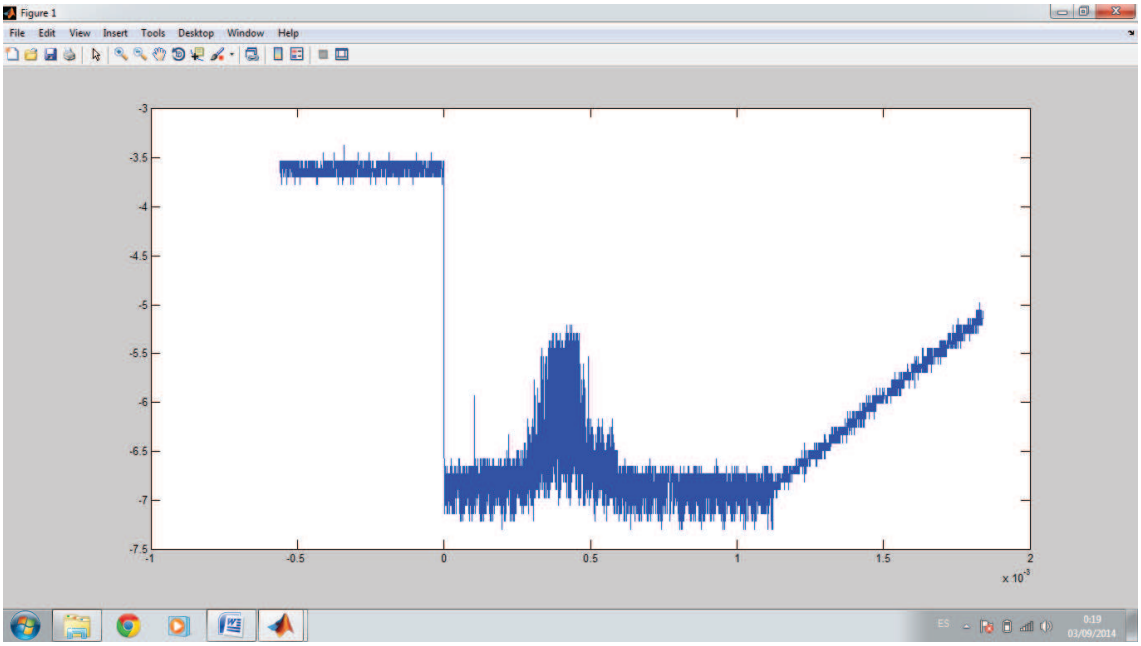

Figura 5.13: Valor de voltaje de salida del sensor durante un periodo de lectura (Más valores de salida con el sensor sin operar).

Como se puede observar en la gráfica de la figura 5.13, los datos fueron invertidos por el hecho de que el sensor TCD1201D funciona aumentando el ancho de los pulsos hacia abajo, lo cual hace visualmente menos evidente los puntos máximos del sensor. Por lo tanto se trata de una operación que tiene como objetivo obtener un resultado gráfico visualmente más estético de los datos obtenidos en la experimentación.

En la figura 5.14 se puede observar los resultados obtenidos de ejecutar esta función:

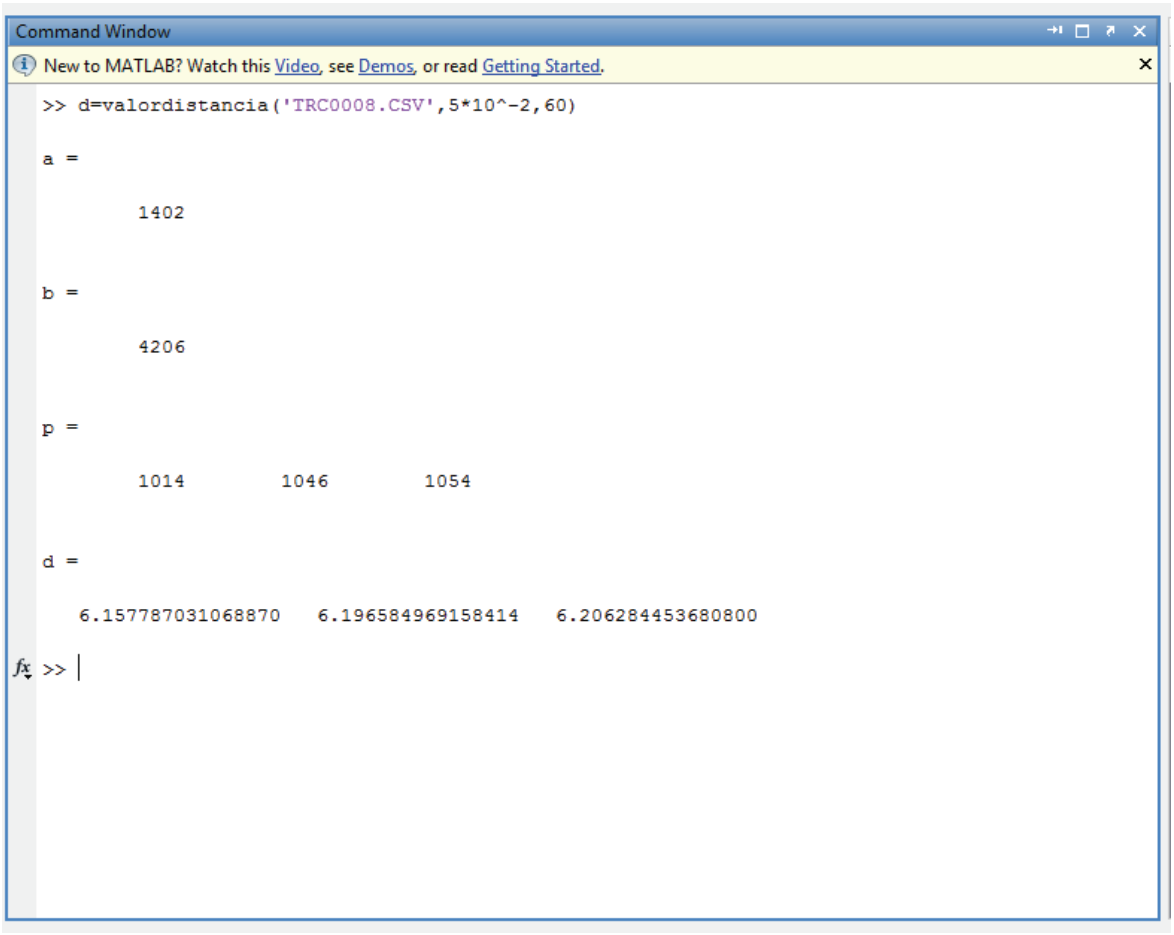

Figura 5.14: Valores de retorno obtenidos al ejecutar la función valordistancia().

Como se puede ver, se le han pasado a la función como argumentos el nombre del fichero en el cual se encontraban los datos, la distancia entre el láser y el sensor en metros, y el ángulo de inclinación del láser en grados.

Como valor de retorno se ha proporcionado la distancia del objeto. Nótese que han sido devueltos tres valores de distancia, esto es debido a que se ha dado el caso de que hay 3 fotodiodos sobre los cuales incide exactamente la misma cantidad de luz. Como la incidencia de luz en el sensor se puede aproximar a una distribución normal, una buena consideración puede ser la de tomar el valor de en medio de estos 3 como el valor real de distancia del objeto. Esta consideración puede considerarse lo suficientemente precisa para muchas aplicaciones.

Además se han devuelto como parámetros complementarios con el objetivo de aportar más información, el valor de la posición de las muestras consideradas como los extremos del ciclo de lectura (a y b); y la posición de los fotodiodos con una incidencia de luz máxima. Si estos valores carecieran de valor alguno se podría omitir su retorno desde la función añadiendo un “ ; “ al final de la sentencia que les asigna un valor.

Otra información que también se ha considerado de interés mostrar por pantalla es una gráfica con los valores de voltaje del ciclo de lectura del sensor, (dummys + valores de fotodiodos), Estos valores son obtenidos con la función interp1() como se explicó con anterioridad.

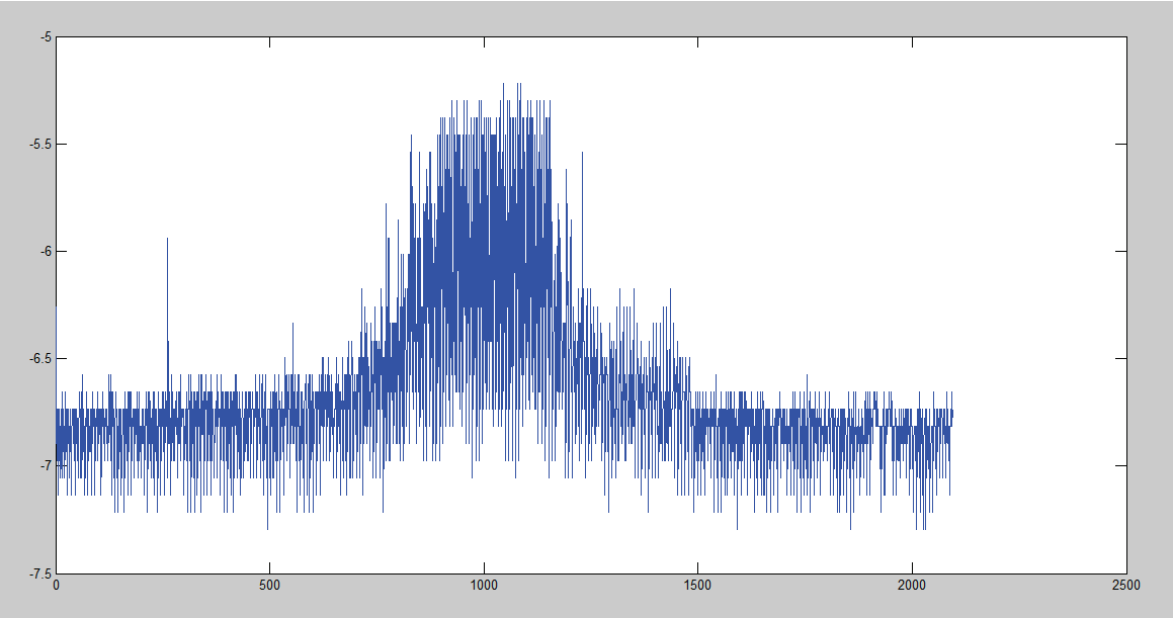

Figura 5.15: Valores de voltaje interpolados con 2094 posiciones.

6. Conclusiones:

6.1. Conclusión:

A la vista de los resultados obtenidos queda claro la dificultad que implica controlar un CCD lineal mediante el empleo de Arduino, debido a la gran sensibilidad que presentan este tipo de sensores.

En el caso del TCD1201D se pudo controlar correctamente pero únicamente a la máxima velocidad, lo cual dificulta el procesado de datos mediante la misma placa. Se precisa de un microcontrolador más rápido que permita realizar la lectura analógica, convierta dicha lectura a valor digital, realice el cálculo del máximo y el de la distancia a una velocidad lo suficientemente rápida como para que no se alteren las señales de control generadas por Arduino. La solución adoptada en el presente documento es la de tratar la información mediante el uso del software de cálculo Matlab, que permite realizar los cálculos de distancia de manera precisa.

Debido a las dificultades comentadas se ha decidido trabajar con otro sensor, el ILX511, que ha parecido ofrecer mejores expectativas a la hora de intentar realizar todos los cálculos con Arduino, implementando los códigos descritos en el apartado 3.4 del presente documento. Sin embargo esto no se ha comprobado al 100% en el momento de entrega de la memoria.

En conclusión aunque se ha conseguido medir correctamente los valores del TCD1201D mediante Matlab, queda abierta todavía una vía de experimentación con el sensor ILX511 que permita realizar los cálculos de manera íntegra con Arduino, aspecto que dotaría al sensor de distancia de mayor practicidad.

Como conclusión personal quisiera destacar la satisfacción que ha sido el trabajar en un tema tan interesante como el que es tratado en el presente trabajo de robótica y automatización; y haber aprendido entre otros conocimientos el uso de una plataforma tan útil como Arduino, la cual me era desconocida al comienzo del trabajo.

6.2. Conclusion (English):

In conclusion the control of a CCD lineal sensor with Arduino is very difficult because CCD lineal sensors are very sensitive.

In the TCD1201D case we could control it correctly, but only in the highest speed, and this difficult the reads of the data. We need a faster microcontroller that allows us to control and read all the data without changing the form of the controller signals. With a faster controller we could read the values, calculate the maximum, and calculate the distance correctly. For that reason we use the oscilloscope to read the values, and the software Matlab to calculate all the values.

Because of this difficulties we also worked with the ILX511 sensor, and it has better expectative to work correctly and I think we could made all the operations with Arduino, because this sensors allows a lower speed. In the moment that the document was upload we hadn´t check that at 100%.

In conclusion although the sensor have measured correctly with Matlab it also exist a way to improve the prototype using ILX511 to realize all the operations with Arduino.

In my opinion I'm very satisfied of work in a Project of robotic and automation, and have learned to use Arduino, that is a component that I didn´t know at the beginning of the work.

7. Bibliografía:

7.1. Libros

[1] Malvino, Albert Paul. Principios de electrónica 4ª edición. Ed. McGraw - Hill, D.L 1994

[2] Nussey, John. Arduino for Dummies 2013.

[3] Pallàs Areny, Ramón. Sensores y Acondicionadores de Señal 4ª edición. Ed. Marcombo 2003

[4] Pérez, César. Matlab y sus Aplicaciones en la Ciencia y en la Ingeniería. Prentice Hall

7.2. Artículos

[5] Battiston Federico. Medición con láser por triangulación. Universidad Tecnológica Nacional, Facultad Regional San Nicolás, 2011.

[6] Bolaños. Sensores ópticos: http://www.bolanosdj.com.ar/index2.htm

[7] Pavón Mera, Esteban. Modelización, simulación y caracterización de fotodiodos p-n y p-i-n. Universitat Rovira i Virgili, Escuela Técnica Superior de Ingeniería 2002.

ANEXOS

```
//Comprobacion de todo un ciclo de lectura
//haciendo uso de la frecuencia máxima
#define OS_PIN A0
#define DOS_PIN A1
#define SH_PIN 3
#define P1_PIN 4
#define P2_PIN 5
#define RS_PIN 6
#define BT_PIN 7


void setup() {
  Serial.begin(9600);  //velocidad de datos en bits por segundo (baudios) par
                       //de datos en serie para comunicarse con el computador

  Serial.println("Setup start");

  //Asignamos cada pin al bit del portD que queramos
  for (int  i=3;i<8;i++)pinMode(i,OUTPUT);

 PORTD = B00000000;
  Serial.println("Setup Complete");
}
 void loop() {

  Serial.println("start loop");

  //Instanciampos una funcion encargada de realizar
  //las lecturas del sensor
    readCCD();
    delay(100);
 }

 void readCCD() {
   Serial.println("Start read cycle");
  // Lo ponemos todo a 0
  PORTD = B00000000;
  delay(1);

  PORTD= B10100000;
  PORTD= B10010000;
  PORTD= B10011000;
  delayMicroseconds(2);
   PORTD= B10010000;
  delayMicroseconds(2);
 for (int i=0; i<16;i++){

    PORTD= B00010000;
    PORTD= B01010000;
```

```
    PORTD= B11010000;
    PORTD= B10010000;
    PORTD= B10100000;

    PORTD= B00100000;
    PORTD= B01100000;
    PORTD= B11100000;
    PORTD= B10100000;
    PORTD= B10010000;

     }

    //Leemos los 2048 valores utiles

  for (int i=0; i<1024; i++){

    PORTD= B00010000;
    PORTD= B01010000;
    PORTD= B11010000;
    PORTD= B10010000;
    PORTD= B10100000;

    PORTD= B00100000;
    PORTD= B01100000;
    PORTD= B11100000;
    PORTD= B10100000;
    PORTD= B10010000;

 }

  for(int i=0; i<7; i++){

    PORTD= B00010000;
    PORTD= B01010000;
    PORTD= B11010000;
    PORTD= B10010000;
    PORTD= B10100000;

    PORTD= B00100000;
    PORTD= B01100000;
    PORTD= B11100000;
    PORTD= B10100000;
    PORTD= B10010000;

     }
  Serial.println("Read cycle complete");
}
```

```
/*
Control del sesnor CCD lineal ILX511
 */

// Se declaran las señales de control que se utilizarán y
// se les asigna el entero del pin al que estaran asociadas
int rog = 12;
int clk = 13;

// Funcion setup
void setup()          {
  //Establecemos una velocidad de
  //comunicación serial de 9600Mbps
    Serial.begin(9600);
  // Confiuramos os pines de las señales de control como salidas
  pinMode(clk, OUTPUT);
  pinMode(rog, OUTPUT);
  //Para empezar el ciclo las establecemos en alta
  digitalWrite(clk, HIGH);
  digitalWrite(rog,HIGH);
}

// Funcion loop que se ejecuta de manera cíclica:
void loop() {
  //Configuramos el pulso inicial
  digitalWrite(clk, HIGH);
  digitalWrite(rog,HIGH);
  delayMicroseconds(1);

  digitalWrite(clk, LOW);
  delayMicroseconds(1);

  digitalWrite(clk, HIGH);
  delayMicroseconds(3);

  digitalWrite(rog,LOW);
  delayMicroseconds(5);

  digitalWrite(rog,HIGH);
  delayMicroseconds(3);

  //Realizamos el ciclo de lectura
  for(int i=0;i<2100;i++){
  digitalWrite(clk,LOW);
  delayMicroseconds(1);
  digitalWrite(clk, HIGH);
  delayMicroseconds(1);
    }
```

```
}
```

```
function [d]= valordistancia(filename,dsl,theta)

%Se importan los datos mediante la funcion csvread
datos= csvread(filename);

%Debemos hacer una detección del inicio del ciclo de lectura
a= min(find(datos(:,2) < -5)) %Cogemos el primer dato del ciclo de lectura
b= max(find(datos(:,2) < -7)) %Cogemos el ultimo dato del ciclo de lectura

%Separamos los datos en un nuevo vector
datosnuevos = datos(a:b);

%Reducimos el numero de valores a 2094 puntos
Puntos=1:2048+32+14;
Nuevos = interp1(1:b-a+1,datos(a:b,2),Puntos);
plot (Puntos,Nuevos);

%Hallamos el valor del máximo
posmax= find(Nuevos == max(Nuevos));
p=posmax-32
d= distancia(posmax, dsl, theta);
```