UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

INSTITUTO DE INGENIERÍA

DESARROLLO E IMPLEMENTACIÓN DE ALGORITMO ÓPTIMO PARA ALMACENAMIENTO Y PROCESAMIENTO DE DATOS OBTENIDOS POR SISTEMA DE BARRIDO LASER TRIDIMENSIONAL

TESIS

PARA OBTENER EL GRADO DE MAESTRÍA EN INGENIERÍA

AUTOR:

HUMBERTO ANDRADE COLLAZO

DIRECTOR DE TESIS: DR. OLEG SERGIYENKO

CODIRECTOR DE TESIS: DRA. VERA TYRSA

MEXICALI, BAJA CALIFORNIA 04 NOVIEMBRE DE 2022

Agradecimientos

Me gustaría agradecer a todas las personas que me apoyaron a continuar con mis estudios, para terminarlos y confiaron siempre en mí.

A mis padres, mi familia por ayudarme a alcanzar mis metas desde mis estudios básicos hasta mis estudios profesionales.

A mis amigos más cercanos por apoyarme y confiar en mí.

A mis compañeros de posgrado por haberme ayudado en todo momento.

A mi director de tesis, Dr. Oleg Sergiyenko por haberme instruido y compartido sus conocimientos a lo largo de mi carrera en posgrado.

A mi codirector de tesis, Dra. Vera Tyrsa por sus buenos consejos en la realización de mi trabajo.

A la universidad Autónoma de Baja California por permitirme continuar con mis estudios tanto en licenciatura como en posgrado.

Al Consejo Nacional de Ciencia Y Tecnología por darme la oportunidad de formar parte del programa de becas nacionales y poder realizar mis estudios de posgrado.

Resumen

En este trabajo de tesis de maestría se presenta un modelo para optimizar la manera en la que la información se distribuye en un sistema de barrido láser con el objetivo de mejorar la forma en la adquisición, procesamiento y almacenamiento de la información obtenida por los diferentes componentes que conforman al sistema de visión técnico mediante el desarrollo de un algoritmo que cumpla con los resultados esperados y con la finalidad de poder implementar el algoritmo en diferentes microprocesadores y brindarle la máxima capacidad al sistema de visión técnico.

Por lo que se abordarán temas como microcontroladores, protocolos de comunicación, algoritmos de programación, interfaces gráficas. Con el fin de comprender los conceptos básicos para el desarrollo de este trabajo de tesis.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1. Introducción

1.1 Sistema de visión

Los sistemas de visión se componen de sensores que son capaces de captar magnitudes externas a través de un conjunto de métodos y obtener información del medio ambiente como temperatura, sonido, luz etc. Un computador que se encarga de procesar de forma precisa y detallada los datos crudos obtenidos por los sensores en datos binarios. Los datos obtenidos son utilizados por sofisticados algoritmos y cálculos matemáticos y que se encargan de reconocer mediante análisis las características del medio que los rodea ya sea su forma, textura, medidas, etc. De esa manera será capaz de tomar decisiones sencillas como rodear un objeto u obstáculo o en tareas más complejas como realizar trabajos de ensamblaje de alta precisión de dispositivos en la industria, para reconocimiento facial y conducción de automóviles inteligentes [1] [2].

1.2 Diferentes sistemas de visión

En los diferentes sistemas de visión encontramos sistemas guiados por cámaras en la cual se capturan imágenes que son procesadas por programas sofisticados e inteligencias artificiales con el fin de diferenciar objetos como visión monocular, estéreo [3] y multicámara [4], sin embargo en los sistemas de estéreo visión es difícil obtener profundidades por lo que dependiendo de la aplicación en ocasiones es necesarios implementar un subsistema capaz de escanear el campo de visión y obtener sus profundidades, algunas de las técnicas empleadas para dicha tarea son sistemas con sensores ultrasónicos, con láser [5] [6] [7] .

Comparación de sistemas de visión (Escaneo láser contra procesamiento de imágenes por cámaras)

Ventajas

- Bajo costo
- Fácil implementación
- Funcionamiento en ambientes con poca luz

Desventajas

- Falta de autonomía
- Problemas en la certeza de los datos
- Propenso a mayor cantidad de errores en ambiente con luz

La complejidad de los sistemas de visión depende directamente de la aplicación que se requiere desempeñar. En algunas investigaciones los sistemas de visión se utilizan para reconocer diferentes tipos de plantas, implementando en su sistema un reconocimiento mediante fotografías pertenecientes a la especie de la planta [8]. En otra investigación utilizan el reconocimiento con cámaras de video para determinar el estado de madurez de legumbres como lentejas mediante el color de las mismas como se demuestra en [9]. También podemos encontrar en aplicaciones médicas mediante técnicas de procesamiento de imágenes por computadora para detectar lesiones en la piel con el fin de detectar canceres [10].

En [11] podemos ver como los sistemas de visión se implementan en la agricultura para inspección de los vegetales, frutas y granos con el fin de determinar su madurez y evaluación de estado.

1.3 Problemática

Actualmente se cuenta con un sistema de barrido óptico en el Instituto de Ingeniería de la Universidad Autónoma de Baja California (UABC) que cuenta con algunos problemas destacados como la programación de tareas innecesarias o ineficientes dentro de los microcontroladores, baja calidad de los componentes electrónicos utilizados en el circuito de control y procesamiento de datos, protocolos de comunicación poco compatibles con dispositivos actuales. Estas limitaciones afectan directamente a la velocidad de transmisión y almacenamiento de los datos, que, al ser mejorados, se obtienen ventajas como una mayor velocidad en el acceso a la información almacenada para análisis, mayor volumen de datos, mayor número de algoritmos a realizar sin afectar a la precisión de los datos obtenidos.

1.4 Justificación

Después de presentar la problemática abordada en el trabajo de investigación propuesto se mencionan los principales factores que justifican la realización de este proyecto. Principalmente, al analizar el escáner láser se identificó que se puede aumentar la velocidad en el sistema al obtener y procesar datos, ya que el almacenamiento consume una cantidad significativa de tiempo. Al atacar el problema mencionado se logrará que el sistema trabaje más rápido. Por lo tanto, se busca que los datos se transmitan eficientemente implementado con lo propuesto en las secciones siguientes. Con los resultados de esta investigación se logrará un escáner láser más eficiente generando así que su gama de aplicaciones sea más amplia. Creando nuevas líneas de investigación para trabajos de tesis de maestrías y doctorados ayudando a los campos de investigación basados en sistemas de visión de láser. Entonces, con la investigación se beneficiarán los futuros investigadores que trabajen en colaboración con el Instituto de Ingeniería de la UABC y las industrias en las cuales tiene muchas aplicaciones donde se requiere

visión máquina. Otros campos donde sería útil podrían ser en aplicaciones tales como la parametrización de mapas en 3D para localización de robot móviles donde es peligroso el acceso para el ser humano [12], medir el ángulo de inclinación de los edificios después de un gran sismo, para el área de la medicina [13] [14].

1.5 Objetivo General

Desarrollar un método y un novedoso algoritmo de procesamiento de los datos que permite acelerar almacenamiento y extracción de memoria de los datos obtenidos por sistema de triangulación dinámica con el objetivo optimizar el almacenamiento y facilitar el uso de estos datos por el sistema de control.

1.5.1 Objetivos específicos

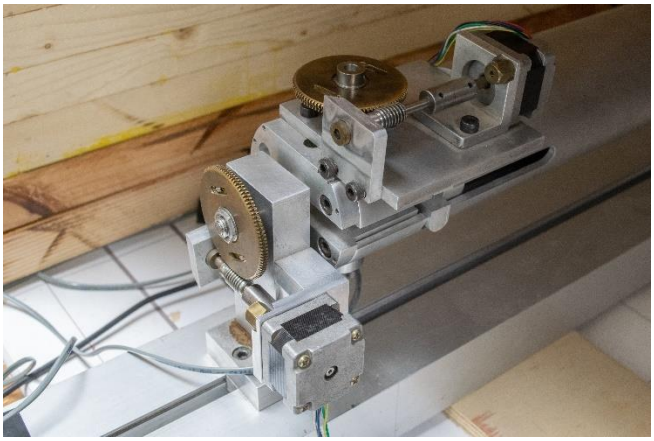
- Analizar el Sistema de Visión Técnica y sus componentes para comprender en totalidad su funcionamiento.
- Analizar las velocidades de transmisión-procesamiento de datos y velocidades de barrido.
- Analizar las fuentes físicas que alteran el comportamiento y transmisión de datos del sistema y proponer una solución que minimice la incertidumbre de los datos obtenidos.
- Investigar sobre diferentes protocolos de comunicación y algoritmos de almacenamiento con el fin de implementar el más adecuado para el registros y procesamiento de datos.
- Desarrollar un algoritmo para optimizar la forma de almacenamiento y transmisión de datos.
- Realizar pruebas y comparar el comportamiento del sistema actual y el propuesto.

- Determinar el porcentaje de mejora en el funcionamiento del sistema propuesto.
- Desarrollar una interfaz gráfica capaz de adaptarse a las diferentes características dadas de un sistema de visión técnico.
- Desarrollar un prototipo con características específicas con el fin de realizar experimentación del algoritmo propuesto.

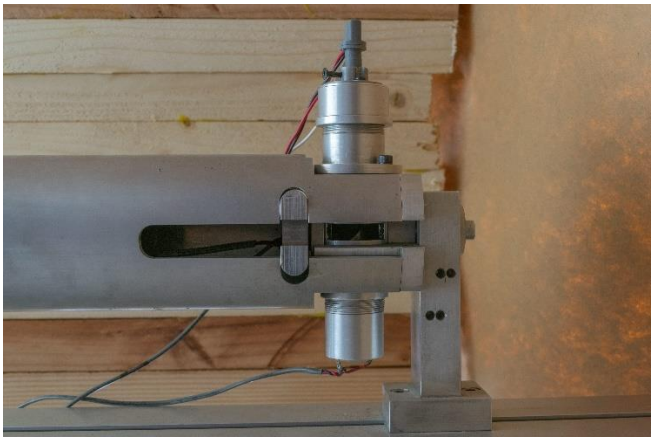
CAPÍTULO 2. Marco teórico

2.1 Sistema de visión técnico

El sistema de visión técnico es un prototipo de visión que implementa la energía de la luz láser para detectar objetos, obstáculos o superficies, el cual utiliza como técnica el principio de la triangulación dinámica para calcular el ángulo de incidencia del haz de luz láser en un objeto encontrado en su campo de visión. El sistema de visión se compone por dos partes principales que son el posicionador laáser (Positioning laser) y la apertura de escaneo (Scanning aperture) [15] [16]. Ver Figura 1.



a) Apertura de escaneo



b) Posicionador láser

Figura 1. Partes principales del sistema de visión técnico (TVS)

2.1.1 Apertura de escaneo

La apertura de escaneo, Ver Figura 2, es el componente que se encarga de obtener la cantidad de luz emitida por el haz de luz láser. Este subcomponente ésta compuesto por un espejo con un corte de 45 grados para redireccionar ortogonalmente el rayo láser hacia un fototransistor de alta velocidad (BPW77NA) [17] que actúa como sensor ya que los valores de voltaje varían de acuerdo a la cantidad de luz que es obtenida por el foto-receptor como se observa en la Figura 3 . El espejo se mantiene girando en su propio eje a una velocidad constante por el motor DC. Cuenta con un disco ranurado que nos proporcionara un contero de pulsos por revolución del motor que son capturados por un optoacoplador.

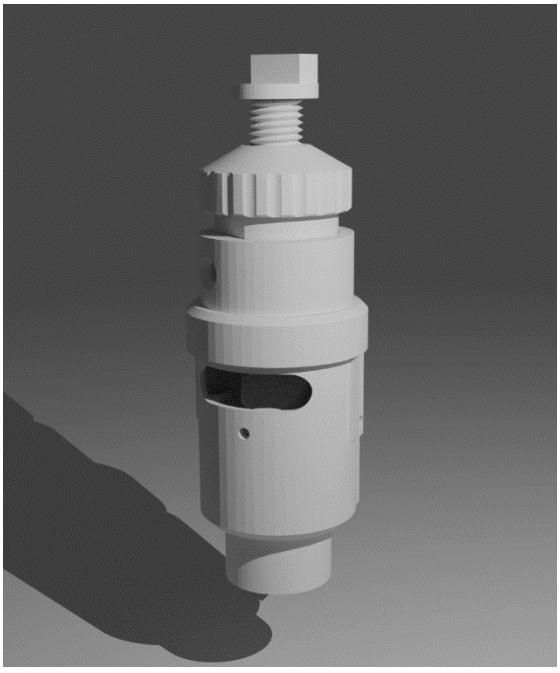


Figura 2. Diseños 3D de la apertura de escaneo del TVS.

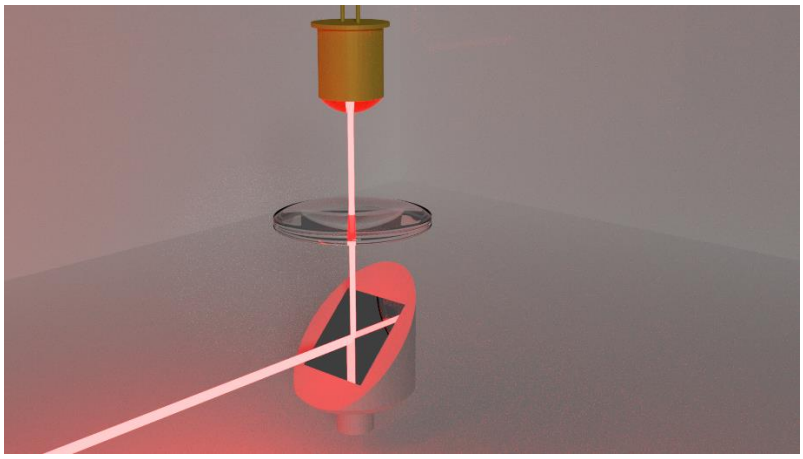


Figura 3. Principio de funcionamiento de la apertura de escaneo.

2.1.2 Posicionador láser

El posicionador láser es el componente encargado de disparar el rayo de luz láser en una determinada dirección, este utiliza un espejo con un corte de 45 grados, para reflejar el láser con un ángulo recto, de esa forma es más preciso el disparar el rayo láser desde el centro de la circunferencia y poder direccionarlo en diferentes ángulos deseados [18]. El control de la posición se logra con un motor a pasos, de esta forma es posible conocer la posición todo el tiempo y también nos será de suma importancia para el cálculo del ángulo de disparo de acuerdo a la resolución de los motores y relación de engranaje, ver Figura 4.

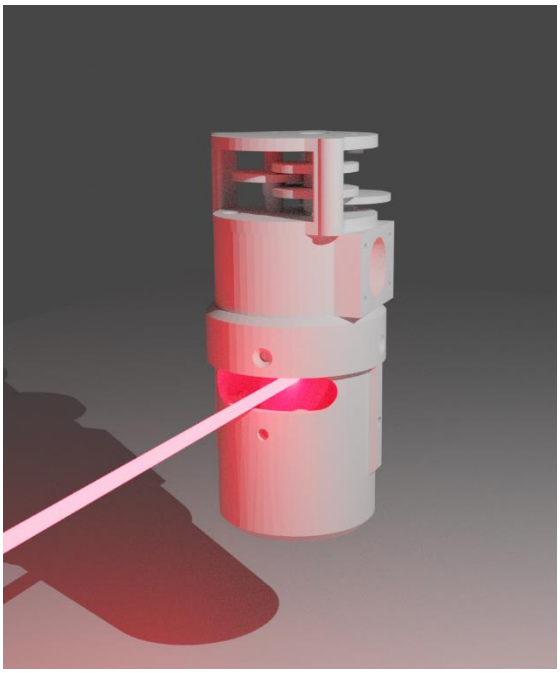


Figura 4. Diseño 3D del posicionado láser del TVS.

Conociendo las partes principales del sistema de visión técnico se demostrará el principio de triangulación dinámica del cual se hace uso para la obtención de ángulos y distancias con el fin de determinar los puntos cartesianos en el espacio.

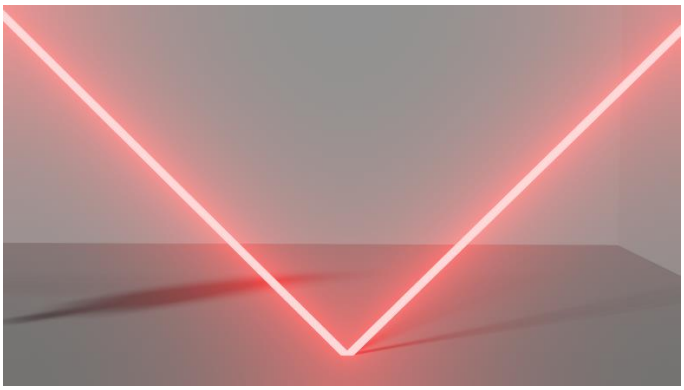
2.2 Principio de triangulación dinámica

La triangulación dinámica es utilizada para obtener coordenadas 3D de un objeto que se encuentra en el rango de visión del sistema [19]. El funcionamiento de este se basa en la reflexión de un rayo láser que es disparado desde un posicionador láser (Positioning Laser) y este haz de luz es reflejado por el objeto que se encuentra en su campo de visión, que a su vez por medio un fenómeno físico de la reflexión y refracción de la luz algunos de estos rayos viajan a través del canal óptico y la mayor parte de la energía es adquirida por el componente anteriormente mencionado en la apertura de escaneo [20].

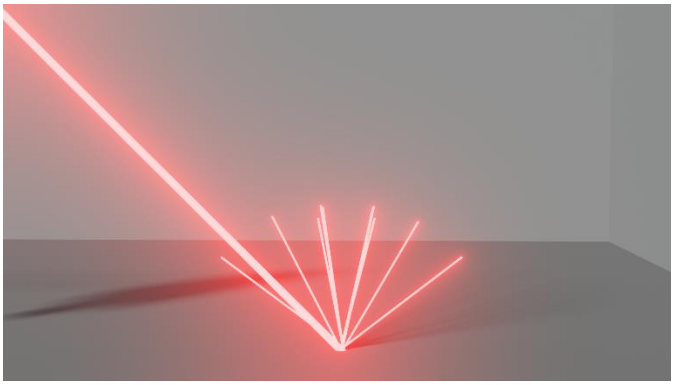
El disparo del rayo láser por el posicionador (PL) rebota en la superficie el cual genera un triángulo al incidir en la apertura de escaneo (SA) como se observa en la Figura 6. Al poder controlar el ángulo de disparo podemos generar una infinita cantidad de triángulos, de ahí su nombre de triangulación dinámica, ya que las características del triángulo cambian de acuerdo al ángulo de disparo del rayo láser, el ángulo de incidencia en la apertura de escaneo y la distancia a la que se encuentra el objeto o superficie a escanear [21].

Para obtener la distancia entre el sistema y el objeto nosotros debemos de conocer la distancia entre el posicionador láser y la apertura de escaneo. Existen algunos inconvenientes cuando se refleja la luz en una superficie, y es que existen dos tipos de reflexión de luz que son reflexión especular y reflexión difusa [22]. La reflexión especular se da cuando la luz es reflejada en una superficie con alto índice de reflexión con lo cual el ángulo de reflexión es igual al ángulo de incidencia, por otro lado, la reflexión difusa se da cuando la luz es reflejada en una superficie con bajo índice de reflexión, lo cual da a lugar a que la luz se refleje en todas las direcciones. Explicando bien estos tipos de reflexión, para el sistema de triangulación dinámica nosotros siempre deseamos que sea del tipo especular ya que tendríamos un mayor volumen de energía capturada por el sensor y por lo tanto una mejora en la precisión en los datos obtenidos por el sistema,

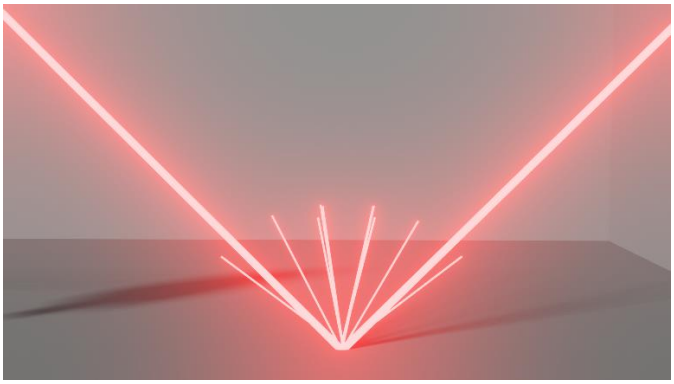
sin embargo, la mayoría de las superficies la luz reflejada presentara ambas naturalezas, en otras palabras, obtendremos reflexión mixta. Vea la Figura 5.



(a)



(b)



(c)

Figura 5. Tipos de reflexiones a) Reflexión especular, b) Reflexión difusa, c) Reflexión mixta

Explicado el funcionamiento de triangulación dinámica debemos hacer uso del misma y para ello utilizaremos las ecuaciones planteadas para obtener la distancia del centro de nuestro sistema al objeto en nuestro campo de visión.

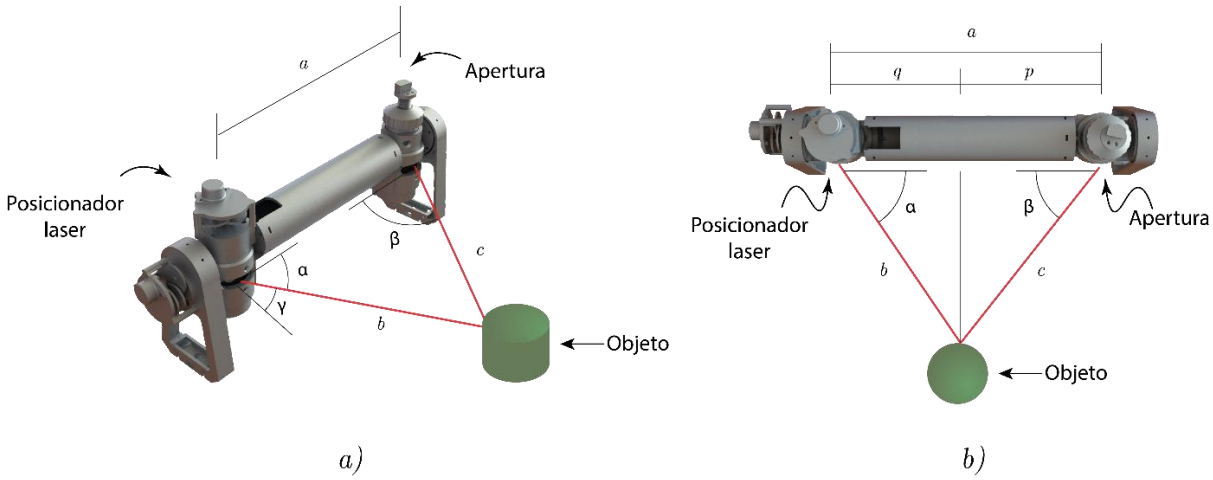


Figura 6. Principio de triangulación dinámica

2.3 Cálculo de las coordenadas 3D

Ecuación para el cálculo de la distancia desde el punto de incidencia del haz de luz láser hasta el sistema de visión técnico.

$$d_{ij} = a\frac{\sin B_{ij}\sin C_{ij}}{\sin\left[180° - \left(B_{ij} + C_{ij}\right)\right]} \tag{1}$$

Para obtener las coordenadas cartesianas se utilizaron las siguientes fórmulas en base a la teoría de senos para sus respectivos componentes para x Ecuación (1), para y Ecuación (3), Ecuación (4) y para z la Ecuación (5) [15].

$$x_{ij} = a\frac{\sin B_{ij}\sin C_{ij}\cos\sum_{j=i}^{n}\beta_j}{\sin\left[180° - \left(B_{ij} + C_{ij}\right)\right]} \tag{2}$$

$$y_{ij} = a\left(\frac{1}{2} - \frac{\sin B_{ij}\cos C_{ij}}{\sin\left[180° - \left(B_{ij} + C_{ij}\right)\right]}\right) \; cuando\colon B_{ij} \leq 90° \tag{3}$$

$$y_{ij} = -a\left(\frac{1}{2} - \frac{\sin B_{ij}\cos C_{ij}}{\sin\left[180° - \left(B_{ij} + C_{ij}\right)\right]}\right) \; cuando\colon B_{ij} \geq 90° \tag{4}$$

$$z_{ij} = a\frac{\sin B_{ij}\sin C_{ij}\sin\sum_{j=i}^{n}\beta_j}{\sin\left[180° - \left(B_{ij} + C_{ij}\right)\right]} \tag{5}$$

Los datos crudos obtenidos que son los pulsos contados por el optoacoplador y el fototransistor deben de ser procesados mediante algoritmos aritméticos para lo cual se requiere de una unidad de procesamiento como algún microcontrolador que sea capaz de realizar dicha tarea.

2.4 Microcontroladores

Los microcontroladores son dispositivos que se componen de tres partes principales: el CPU (Central Processing Unit por sus siglas en inglés), la memoria y los periféricos de entradas y salidas [23] [24].

El CPU o la unidad de procesamiento central es la encargada de procesar toda la información por medio de circuitos que se encargan de realizar los procesos matemáticos y lógicos estos circuitos compuestos son conocidos como la unidad aritmética y lógica también como ALU (Arithmetic and Logic Unit por sus siglas en inglés).

Los periféricos de entradas y salidas se encargan de interactuar con el exterior y de igual manera desde los procesos de información hacia el exterior. Existen entradas digitales y análogas de las cuales las digitales pasan de estado alto a bajo con un nivel de voltaje por lo regular de 5V. En las entradas y salidas análogas el microcontrolador es capaz de reproducir señales de valores continuos dependiendo de la precisión que posea el microcontrolador. Ver Figura 7. Otras de las funciones de los periféricos son para la recepción y transmisión de datos a partir de protocolos de comunicación integrados, cabe mencionar que estos protocolos son incluidos normalmente en microcontroladores de gama media en adelante.

La memoria dentro de un procesador es una parte muy importante puesto que es donde se almacenan todos los datos de ejecución de un programa, existen dos tipos de memorias en los procesadores, la memoria volátil y la memoria no volátil. La memoria volátil es un área donde se guardan datos temporales, esta parte de la memoria la mayoría del tiempo es utilizada por el procesador para almacenar información de las variables y procesos que se están ejecutando en tiempo real eso quiere decir que en cuanto el proceso que se está ejecutando estos espacios de memoria están siendo ocupados y una vez termina, el espacio reservado es liberado siendo capaz de albergar información nueva para otros procesos [24]. Estas memorias son conocidas como memorias RAM (Random Access Memory por sus siglas en inglés). Por otro lado,

existen las memorias que almacenan el programa y estas pueden quedar grabadas indefinidamente, aunque carezcan de fuente una energía. Estas memorias son conocidas como memorias ROM (Read-Only Memory por sus siglas en inglés) donde se almacena el código que se ejecutara cada vez que sea energizado el procesador.

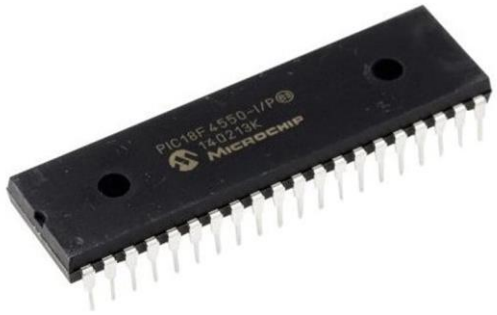


Figura 7. Microcontrolador PIC18F4550

Todos los datos obtenidos mediante las técnicas matemáticas y formulas deben ser almacenados y para ello es necesario de un algoritmo capaz de enviar y recibir el volumen de datos que contendrá y para ello es requerido la implementación de un protocolo de comunicación adecuado para nuestra aplicación de enviar y recibir datos del fototransistor y el optoacoplador para su procesamiento en una unidad de cómputo.

2.5 Protocolos de comunicación

Los protocolos de comunicación son formas estructuradas de enviar información a través de un canal de comunicación en el cual dos o más dispositivos son capaces de intercambiar datos de forma asíncrona o síncrona [25].
Existen algunos protocolos de comunicación muy conocidos como los siguientes que se mencionan a continuación.

- USB (Universal Serial Bus)
- Serial
- SPI (Serial Peripheral Interface)
- I2C (Inter-Integrated Circuit)
- MODBUS
- Profinet
- BacNET

Cada uno de ellos esta específicamente desarrollados para ser utilizados en tareas distintas, ya sea para la industria que requiere de conexiones de largas distancia y donde el volumen de datos es alto, puesto que la mayoría de las veces son para el control de máquinas CNC y robots articulados que realizan acciones complejas. Por otro lado, existen protocolos de comunicación para tarjetas de control que son de corta distancia, pero son mucho más rápidos en el intercambio de información [26] [27].

2.5.1 Protocolos síncronos

Estos protocolos de comunicación están sincronizados en la transmisión de datos mediante una señal de reloj, y es de suma importancia ya que es la forma en la que los dispositivos codifican la información recibida. Algunas de las ventajas de este tipo de protocolos es que admiten velocidades de transmisión mucha más altas en comparación con los protocolos asíncronos. También requiere de una configuración de Maestro-Esclavo. Algunos de estos protocolos son el I2C, SPI y USB.

2.5.2 Protocolos asíncronos

Los protocolos asíncronos se transmiten mediante bytes y estos requieren de una inicialización para comenzar la transmisión en consecuencia también requieren de una señal de parada, una de las ventajas es que requieren de pocos cables para el inter conexión entre dispositivos, también pueden ser transmitidos a largas distancias, aunque los fabricantes que utilizan estos tipos de protocolos deben especificar la distancia máxima para evitar pérdidas de información [28]. Como desventajas encontramos que son más susceptibles al ruido y por esa razón existen algoritmos que nos ayudan a minimizar las pérdidas de información por ejemplo añaden un bit de paridad con el cual el dispositivo receptor realiza cálculos para determinar si se ha alterado un bit en la transmisión de información. Otros protocolos como MODBUS, Profinet implementan un código para el cálculo de errores conocido como CRC (Cyclic Redundancy Check por sus siglas en inglés) [29]. Algunos protocolos asíncronos son Serial, BACnet, Profinet.

2.6 Almacenamiento de información

En la actualidad la cantidad de información utilizada es cada vez mayor siendo necesaria cada vez más espacio para almacenar el volumen de datos, como el ejemplo de las fotografías digitales donde en una foto existen muchos componentes que determinan el espacio que ocupa en almacenamiento como la resolución de la cámara, el formato de compresión, el mapa de colores y el tipo de mapa de bits, solo por mencionar algunos. Resulta casi imposible calcular el tamaño exacto de una imagen digital, pero esta se puede estimar con los datos disponibles mencionados anteriormente. Una imagen digital con una resolución de 12MP (Mega pixeles) con una compresión de 1/6 necesita de un espacio estimado de 6MB. Y para las cámaras modernas de 24MP son capaces de adquirir fotografías de aproximadamente de entre 12MB y 15MB.

Con el avance de la tecnología día a día las resoluciones de las cámaras aumentan y no solo eso, sino que la mayoría de aplicaciones requieren cada vez más un mayor volumen de información por lo que también es necesario incrementar el espacio de almacenamiento de los dispositivos en un tamaño compacto y que sea de rápido acceso para manejar el volumen de datos solicitado [30].

En nuestro caso los sistemas de visión requieren de una gran cantidad de espacio de almacenamiento para todo tipo de datos relacionado con el medio ambiente. Como en el caso de sistemas de estéreo visión los cuales utilizan la información de las imágenes que capturan. Siendo información de las imágenes en pixeles, que conlleva el tono de cada color, el formato. Y en el caso de los sistemas de visión por escáner láser, estos pueden almacenar matrices de los puntos obtenidos con el fin de recrear un objeto en 3D [31].

CAPÍTULO 3. Metodología

3.1 Incremento en la velocidad de trabajo del microcontrolador

Cuando se habla de microcontroladores algo que es imprescindible es la velocidad de trabajo del mismo, ya que dependiendo de nuestra aplicación seremos capaces de realizar una cantidad determinada de tareas, pero para ello es necesario saber seleccionar el adecuado.

Algunos microcontroladores poseen características como el aumentar su velocidad de trabajo utilizando un multiplicador de frecuencia con el fin de realizar más tareas en un lapso de tiempo. El microcontrolador PIC18F4550 utilizado para el sistema de visión técnico posee dicha característica el cual aumentara su velocidad de trabajo máxima hasta cuatro veces siendo esta de 48MHz.

Al aumentar la velocidad de trabajo del microcontrolador podrá ejecutar más instrucciones en el mismo intervalo de tiempo. Con esto nosotros debemos programar el microcontrolador como nos especifica el fabricante, con una configuración de oscilador de HSPLL (High-Speed Crystal/Resonator with PLL enabled por sus siglas en inglés) [32].

Para realizar dicha tarea primero debemos de dividir la frecuencia de entrada en oscilador primario en nuestro caso utilizaremos un cristal de cuarzo de 20Mhz por lo que para obtener los 4Mhz debemos dividir la frecuencia en cinco partes por lo que cargaremos el valor 5 en el registro CONFIG1L siendo los bits 0-2 donde se encuentra asignado PLLDIV0-PLLDIV2. La frecuencia resultante de 4Mhz que a su vez pasa por un bloque que la multiplica a 96MHz. Estos 96Mhz son utilizados para configurar los periféricos de comunicación USB si se desea utilizar este módulo USB 2.0 que esta incorporado en el microcontrolador ver Figura 10.

Teniendo los 96Mhz debemos dividir nuevamente para alimentar al CPU, esto es posible gracias a un bloque adicional de PLL Postscaler denominado CPUDIV el cual

también debemos configurar siendo que los bits a programar son CPUDIV0-CPUDIV1 que se encuentran en el registro CONFIG1L asignados a los bits 3-4. Ver Figura 8.

U-0	U-0	R/P-0	R/P-0	R/P-0	R/P-0	R/P-0	R/P-0
—	—	USBDIV	CPUDIV1	CPUDIV0	PLLDIV2	PLLDIV1	PLLDIV0
bit 7							bit 0

Figura 8. Registro CONFIG1L del microcontrolador PIC8F4550

Los divisores de frecuencia o preescaladores son circuitos diseñados para dividir una frecuencia de entrada determinada y como resultado obtenemos f/n donde f es la frecuencia de entrada y n es un numero entero. Ver Figura 9.

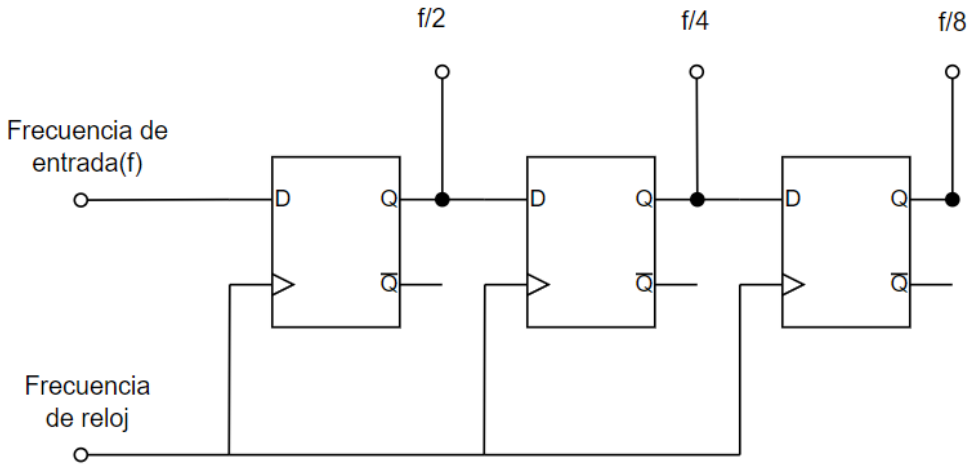


Figura 9. Circuito divisor de frecuencias

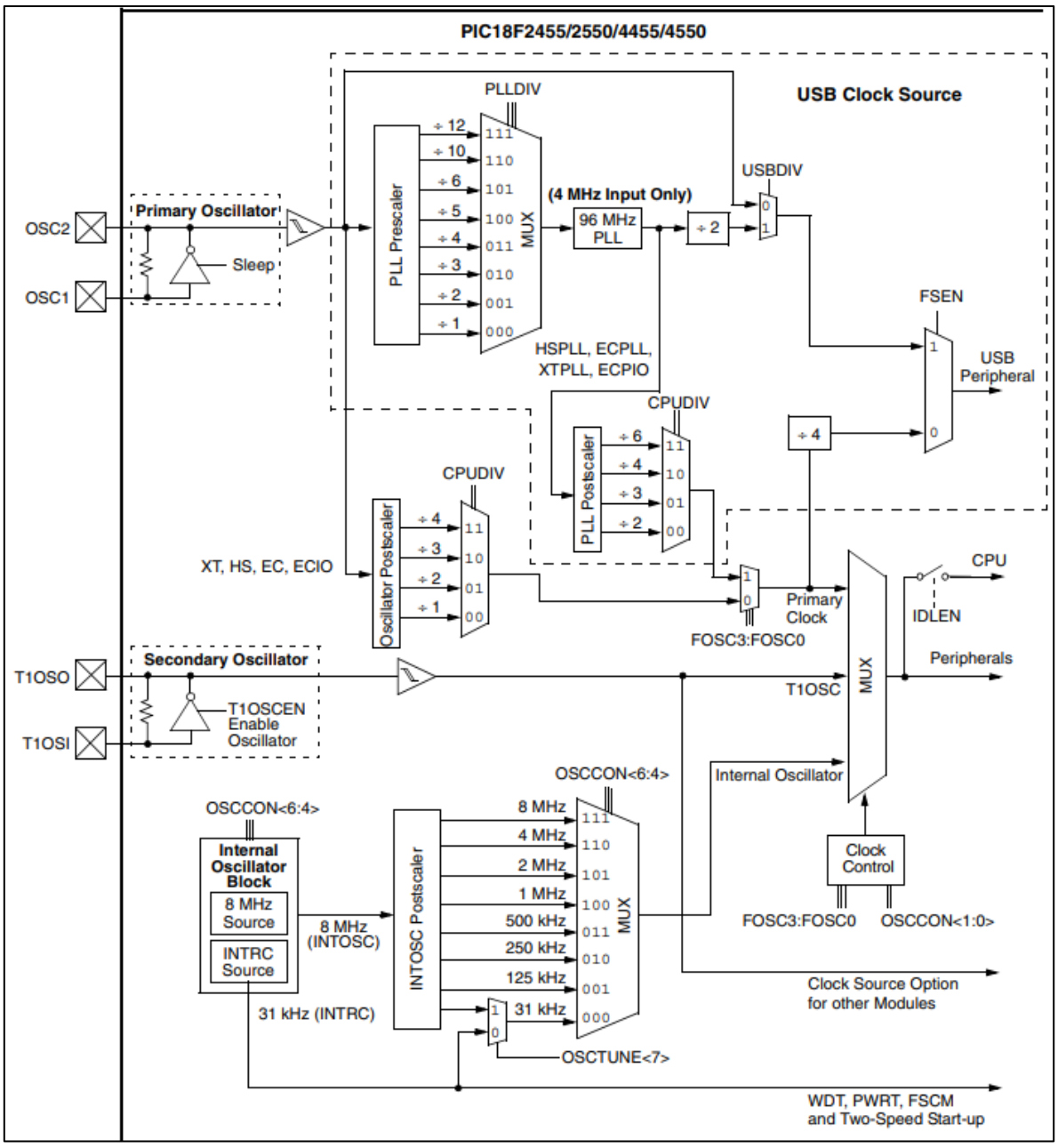


Figura 10. Diagrama de registros para configuración de oscilador

Con el incremento de la velocidad del microcontrolador podemos manejar un volumen de datos en el mismo intervalo de tiempo siendo de 5MHz en cada instrucción antes y siendo ahora de 12Mhz cada instrucción con un aumento del 140% lo cual nos proporciona una mayor cantidad de muestras por segundo y podríamos generar una

señal virtual de 20kHz sin disminuir la velocidad de procesamiento de la información obtenida que veremos en el siguiente tema.

Para pruebas experimentales en la velocidad del microcontrolador se desarrolló un código sencillo que genera un cambio de estado en una terminal para poder observarlo en el osciloscopio. Primeramente, se tomará la señal con la programación de los registros sin el PLL para poder observar la velocidad de trabajo en cada ciclo como se observa en la Figura 12. Posteriormente se programará el microcontrolador nuevamente implementando los registros PLL del oscilador primario configurándolo para una velocidad máxima de 48MHz.

```
#include <18f4550.h>
#fuses HS, NOWDT, NOLVP, NOPROTECT, NOPUT

#use delay(clock=20M)

void main(){
      set_tris_a(0x00);
      set_tris_b(0x03);
      set_tris_c(0x80);
   while(1){
      output_toggle(PIN_B7);
   }
}
```

Figura 11. Código en C para generar cambio de estado en cada ciclo de trabajo

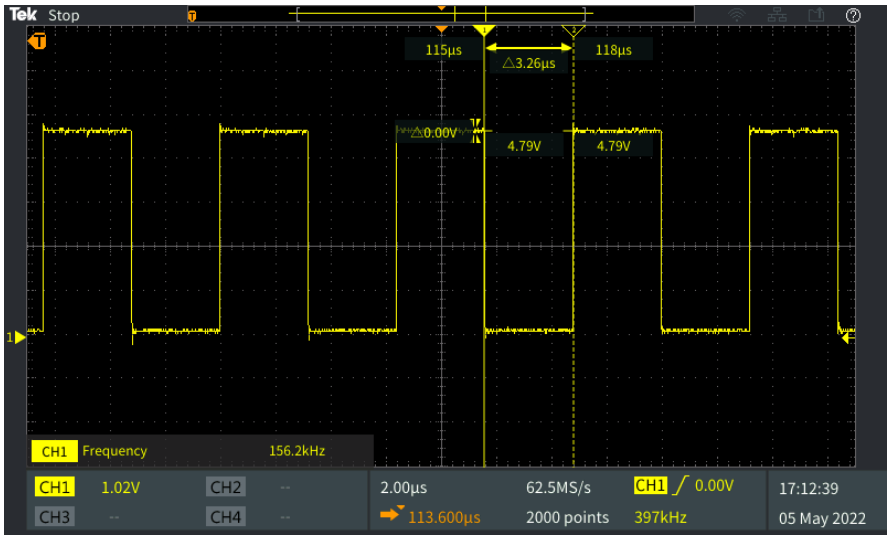


Figura 12. Gráfica representativa de señal generada de 156.2kHz

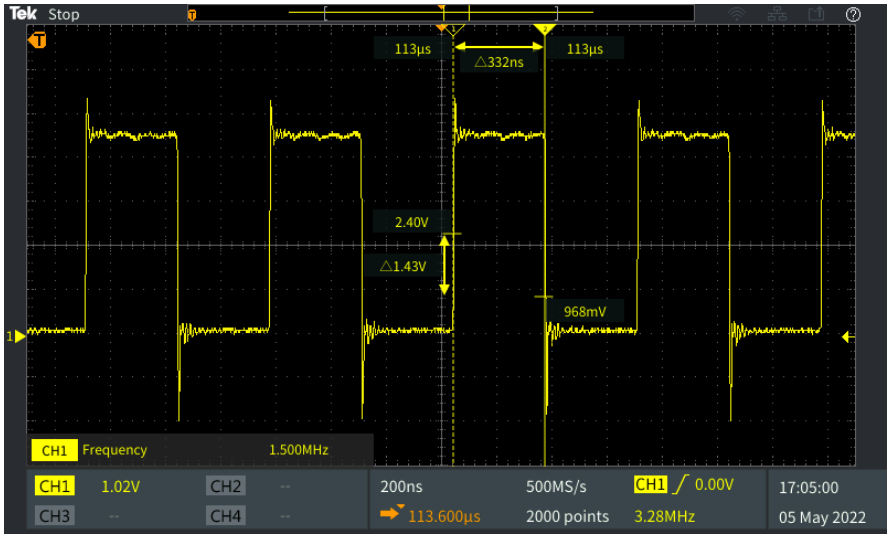


Figura 13. Gráfica representativa de señal generada de 1.5MHz

Como se ilustra en la Figura 12 nosotros podemos observar una señal cuadrada de 156.2kHz, aunque teniendo en cuenta que el programa se ejecuta cada vez para hacer un cambio de estado en la terminal, la frecuencia total resultante sustituyendo en la ecuación (6), donde f_T es la frecuencia total y f_o es la frecuencia obtenida resultando en 312.4kHz

$$f_T = 2f_O \tag{6}$$

Si sustituimos en la ecuación (7) donde f_T es la frecuencia total y T es el tiempo nosotros obtenemos un periodo de 3.203µS por instrucción.

$$T = \frac{1}{f_T} \tag{7}$$

En la Figura 13 podemos observar la gráfica generada por el microcontrolador con el programa utilizando los registros PLL, sustituyendo en la ecuación (6) obtenemos una frecuencia total de 3MHz y sustituyendo en la ecuación (7) obtenemos un periodo de 333.3nS por cada instrucción.

Algunas técnicas como circuitos externos dedicados específicamente a la tarea de generar una señal cuadrada de 20kHz (Microcontroladores, oscilador astable con circuito integrado NE555) [33] con el fin de contar los pulsos mediante los algoritmos programados. Con la optimización de la velocidad de trabajo del microcontrolador crearemos una señal virtual de 20kHz sin la necesidad de circuitos externos. De esta forma obtenemos mediante software la señal de referencia necesaria para el cálculo del ángulo en que la apertura de escaneo (SA) detecte el rayo láser reflejado.

3.2 Generación virtual de señal cuadrada de 20kHz

Como se explicó en 2.1.1 la apertura de escaneo contiene un espejo con un corte de 45 grados el cual se encuentra girando en su propio eje, ese espejo tiene un disco ranurado el cual nos permite detectar las revoluciones del motor DC el cual nos marca el inicio para empezar a contar los pulsos por la señal de referencia.

La mayoría de los microprocesadores existentes en el mercado poseen algunas características en común como los temporizadores. Para generar una señal virtual de 20kHz haremos uso de uno de los módulos del microcontrolador PIC18F4550.

El microcontrolador internamente cuenta con cuatro temporizadores, estos temporizadores tienen la capacidad de operar al mismo tiempo que el programa principal y cuando el temporizador termina su cuenta se desborda. El desborde del registro es cuando el temporizador pasa de su máximo nivel de almacenamiento a cero, es decir el registro del timer 0 es de 8 o 16 bits dependiendo de la aplicación en la que se desea trabajar, en nuestro caso en particular lo programaremos con 16 bits, lo que significa que su registro puede almacenar máximo 2^{16} o sea, 65535 por lo que cada conteo esta dado por un intervalo de tiempo y nosotros deseamos generar un retardo, lo que significa que sumaremos el tiempo de cada cuenta hasta que obtengamos el tiempo requerido. Para obtener el retardo se ha planteado la siguiente ecuación (8) para el temporizador de 8 bits y la ecuación (9) para 16 bits.

$$T_R = \left(\frac{4 * Preescaler}{F_{osc}} (2^8 - TMR0) \right) \tag{8}$$

Donde T_R es el tiempo de retardo requerido para generar en el microcontrolador.

$$T_R = \left(\frac{4 * Preescaler}{F_{osc}} (2^{16} - TMR0) \right) \tag{9}$$

En la ecuación (9) planteada para obtener un tiempo de retardo de acuerdo a que el ciclo de trabajo del microcontrolador es de un cuarto de la frecuencia de entrada, en otras palabras, sustituyendo en la ecuación (9) si trabajamos el microcontrolador

nosotros podemos obtener un retardo de hasta 12Mhz, entonces cada conteo tiene un lapso de 83.3nS si utilizamos el timer0 con un registro de 16 bits eso nos da como resultado 5.46mS como retardo máximo. Con los resultados obtenidos podemos generar una señal cuadrada de mínimo 183.10Hz y máxima de 12MHz. Despejando la ecuación (9) obtenemos la siguiente ecuación (10) que nos ayudara a cargar el valor en los registros TMR0H y TMR0L [32] del microcontrolador como se observa en la Figura 14 un fragmento de código para cargar el valor en el registro del timer0.

$$TMR0 = 2^{16} - \frac{T_R * F_{osc}}{4 * Preescaler} \tag{10}$$

Donde F_{osc} es la frecuencia de trabajo del microcontrolador, *TMR0* es el registro del timer0 en el microcontrolador, T_R es el tiempo de retardo y *Preescaler* es el preescalador que divide la frecuencia de trabajo que es un número entero.

```
#int_timer0
void timer0_interrupt(){
      output_toggle(PIN_B7);
      set_timer0(65525);
}
```

Figura 14. Código en C que muestra la generación de un pulso de 20kHz.

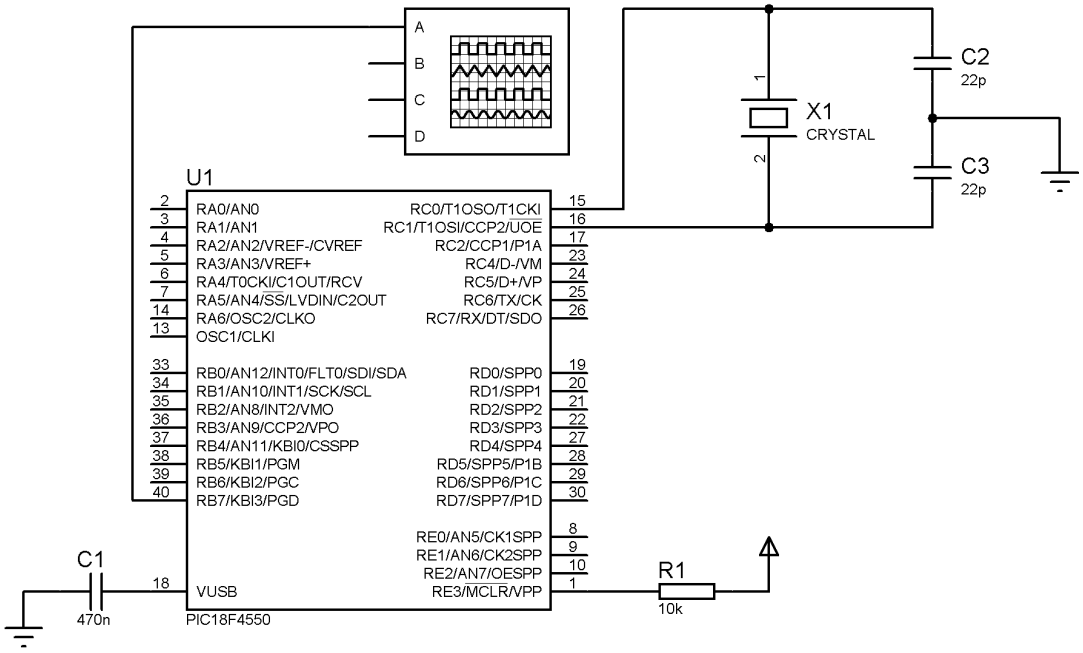


Figura 15. Diagrama de conexión con osciloscopio para generación de señal cuadrada de 20kHz.

Como se ilustra en la Figura 15 se realizaron pruebas de funcionamiento armando el circuito mostrado, el cual se generará un programa en C para obtener la señal de salida y monitorearla en el osciloscopio además que también será simulada en Proteus, un software para simulación de circuitos [34].

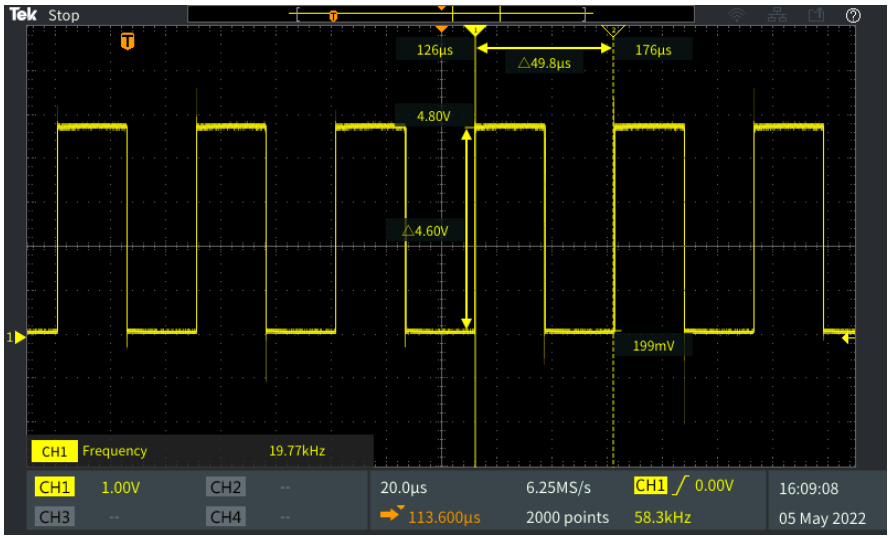


Figura 16. Gráfica de señal generada por microcontrolador PIC18F4550 en osciloscopio.

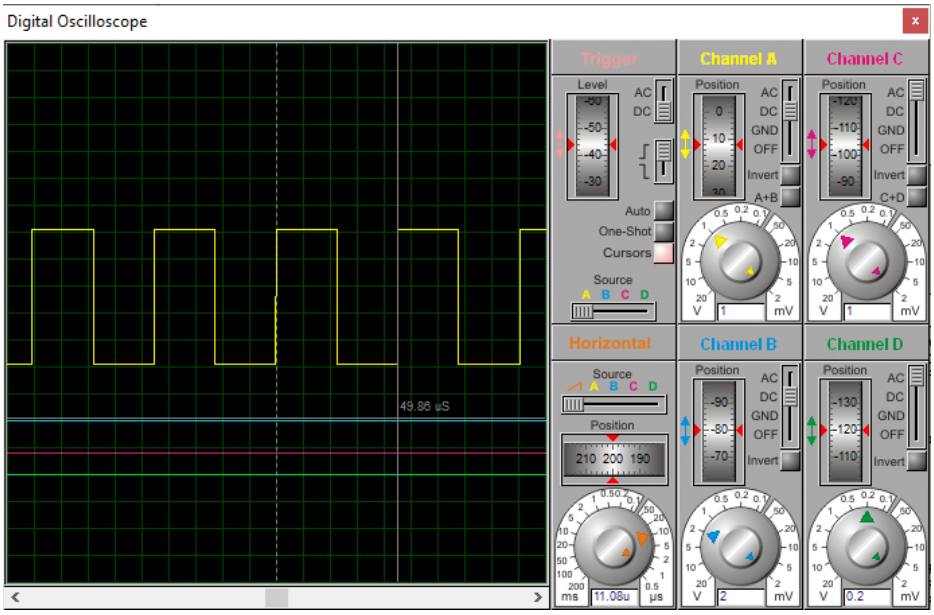


Figura 17. Simulación de señal de 20KHz.

En la Figura 17 observamos una gráfica generada en el simulador Proteus el cual nos ayuda a observar los resultados experimentales, siendo estos coherentes con los resultados obtenidos en experimentación.

3.3 Adquisición y procesamiento de señal obtenida por apertura de escaneo

Es imprescindible tener un buen método de adquisición de datos cuando se trata de sistemas que están estrechamente relacionados con el medio ambiente, por lo que es importante reducir al mínimo la incertidumbre en la toma de información ya sea, distancia, temperatura, intensidad de luz, presión etc.

Para el TVS utilizamos un rayo láser que se refleja en una determinada superficie u objeto que se encuentra en nuestro campo de visión, nuestro fototransistor es el encargado de captar la intensidad de luz recibida y posteriormente ser interpretada por un microcontrolador o por algún circuito para compensar o transformar la señal.

El microcontrolador PIC18F4550 posee dos comparadores análogos internos con los cuales es posible programar de diferentes maneras (dos comparadores independientes, referencia común con salida digital, un comparador independiente, etc.) [32], en nuestro caso utilizaremos dicho comparador para procesar la señal recibida del fototransistor.

De acuerdo a la hoja de datos especificada por el fabricante utilizaremos la configuración de dos comparadores independientes y utilizaremos el comparador denominado como C1 que se encuentra asignado a los pines RA0/AN0 y RA3/AN3/Vref+. En la Figura 18 se observan las conexiones de acuerdo a los pines del microcontrolador.

Para obtener la señal del fototransistor VBPW77NA se ha planteado el siguiente circuito obteniendo el diagrama que se observa en la Figura 18.

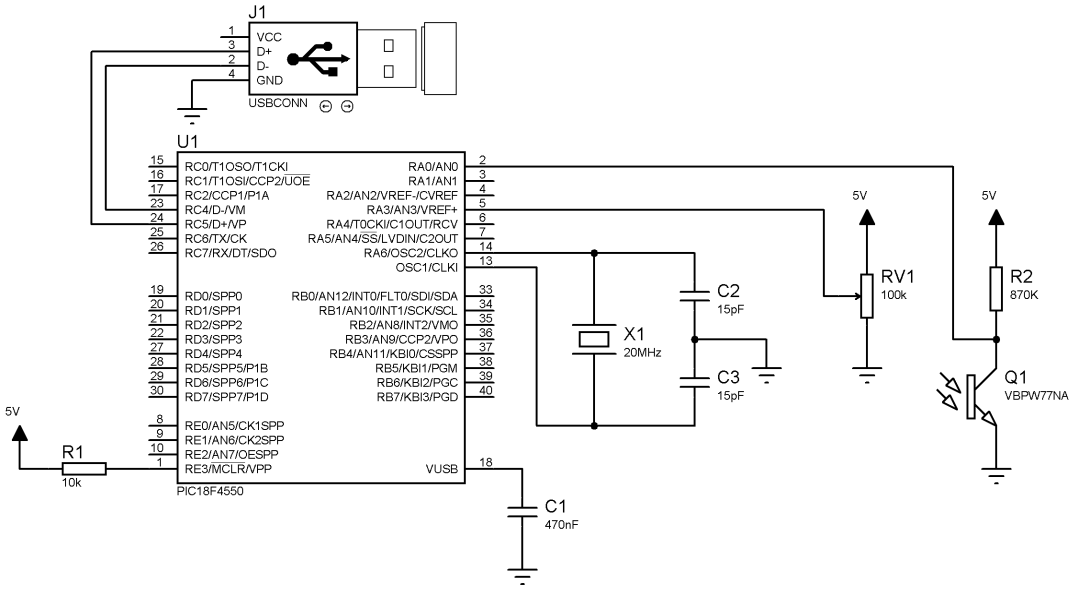


Figura 18. Diagrama eléctrico para captura de señal de fototransistor VBPW77NA

Las características de los comparadores están descritas por el fabricante en la hoja de datos en el cual tenemos un tiempo de respuesta de 150ns en comparación con el circuito integrado LM311 que es de 165ns [35] claramente una diferencia de 15ns siendo el comparador interno del PIC18F4550 más veloz. En la Figura 19 se observa el diagrama de conexiones realizado con circuitería externa.

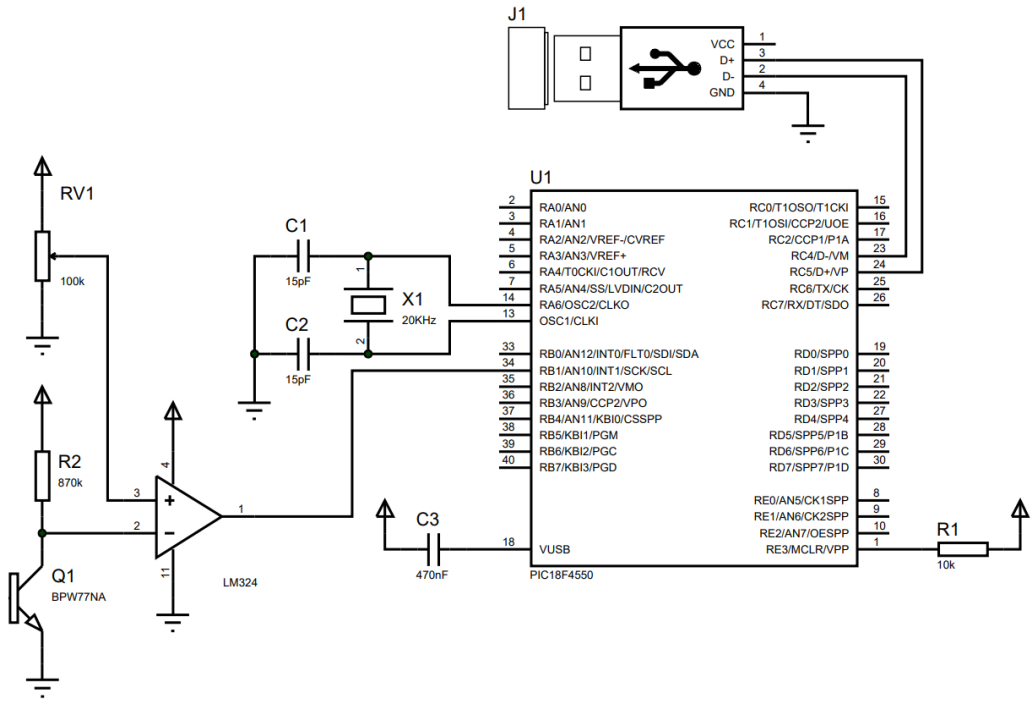


Figura 19. Circuito con comparador externo para detección de señal de fototransistor BPW77NA.

El comparador interno C1 mencionado anteriormente será utilizado para detectar los cambios de estado en la señal recibida y así determinar el centro de la señal recibida por el fototransistor.

En este caso se realizaron dos configuraciones diferentes, en el primer caso utilizamos el comparador para detectar el nivel de voltaje establecido por el potenciómetro RV1 como se observa en la Figura 20, en este caso depende de la conexión del fototransistor y el potenciómetro, dependiendo como deseamos detectar la señal ya sea con flanco de subida o flanco de bajada como se observa en la Figura 21.

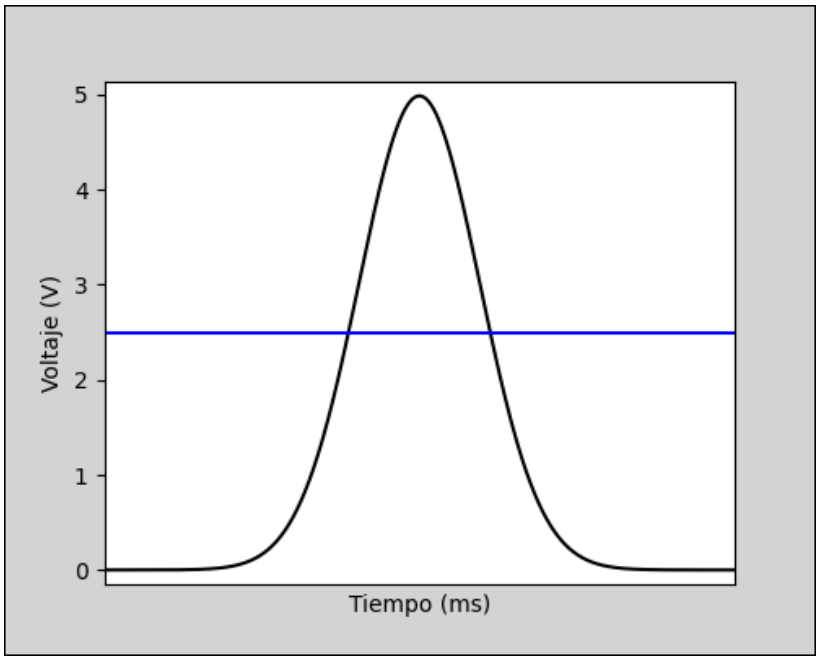
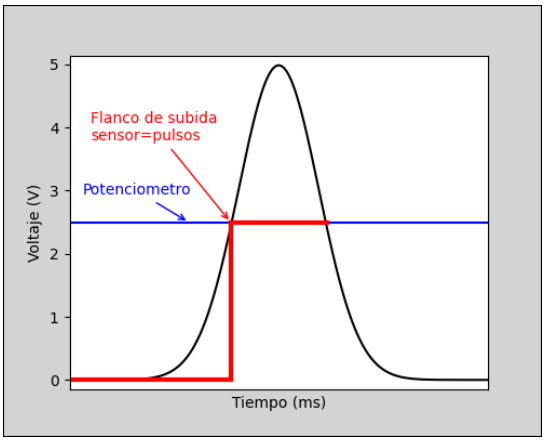


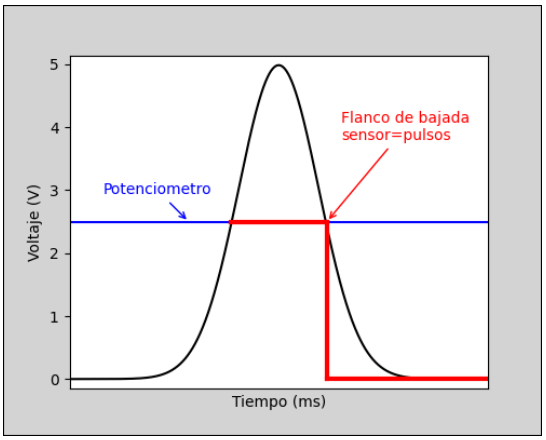
Figura 20. Representación de la señal del fototransistor(negra) y voltaje de comparación(azul).

El circuito utilizado en la Figura 19 el cual utiliza un comparador externo para detectar el flanco de subida producido por la señal del fototransistor se puede considerar que tiene algunas desventajas, las cuales son la utilización de tiempo en el programa para esperar la señal del fototransistor, esto puede solventarse utilizando una interrupción externa.

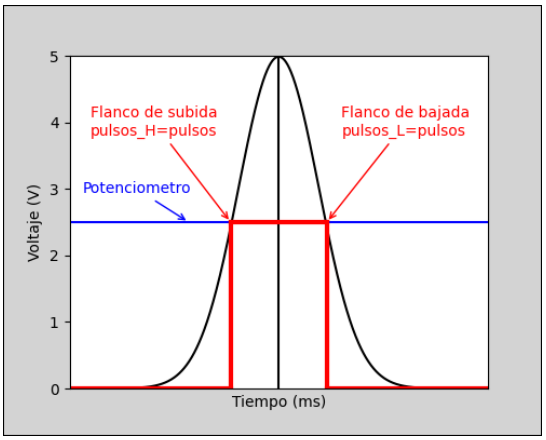
En este caso es posible detectar la señal de acuerdo a un nivel de voltaje establecido por el potenciómetro RV1, ya que no conlleva muchos recursos de procesamiento y es sencillo de implementar en cuanto a circuitos externos, sin embargo, presenta inconvenientes de procesamiento ya que nosotros deseamos tomar el centro de la señal ya que ahí es donde se concentra la mayor parte de la energía reflejada por la superficie [19], lo cual es imposible obtener con la configuración mencionada anteriormente.



a)



b)



c)

Figura 21. Gráficas obtenidas en las configuraciones del comparador. a) Flanco de subida. b) Flanco de bajada. c) Cambio de estado en salida de comparador

Como se observa en la Figura 21c la detección de los flancos de subida y los flancos de bajada son utilizados para el cálculo y obtener el centro de la señal gaussiana generada por el sensor.

```
#int_comp
void comparator_interrupt(){
   output_toggle(PIN_B7);
   if(!C1OUT && flagC){
      phototransistorL = pulsos;
      flagC = 0;
   }
   if(C1OUT && !flagC){
      phototransistorH = pulsos;
      flagC = 1;
   }
}
```

Figura 22. Código de interrupción por cambio de estado en comparador

Utilizando las interrupciones generadas por los cambios de estado en el comparador C1 del microcontrolador el cual nos brinda la ventaja de ejecutar código en cada cambio de estado en el C1, en otras palabras, cada vez que se genere un cambio ya sea de subida o de bajada se generará una interrupción en el programa y se ejecutará un código que nosotros hemos establecido para obtener los pulsos contados en los flancos de subida y en los flancos de bajada que serán almacenados en variables y obtener el resultado con la siguiente ecuación establecida donde $pulsos_F$ son los pulsos contados en el centro de la señal gaussiana por el fototransistor, $pulsos_H$ son los pulsos contados por el fototransistor en el flanco de subida y $pulsos_L$ son los pulsos contados por el fototransistor en el flanco de bajada.

$$pulsos_F = \frac{pulsos_H - pulsos_L}{2} \tag{11}$$

En la Figura 22 tenemos el fragmento del código que se encarga de asignar el valor de los pulsos contados por la señal de referencia de 20kHz (mencionado en 3.2) para determinar la posición del rayo láser y una vez obtenidos los valores sustituimos en la

ecuación (11). Con esto se obtiene una mayor presión al obtener el centro de la señal, aunque claro también depende de la señal generada por el fototransistor.

Teniendo en cuenta que mediante programación hemos generado una señal de referencia y también obtenemos la señal del fototransistor, también es importante tener un control sobre los motores a pasos el cual también es posible mediante programación, haremos uso de la disponibilidad de los puertos del microcontrolador para generar la secuencia de control como se verá en el siguiente tema.

3.4 Generador de secuencia para movimiento de motores a pasos

El sistema de barrido láser utiliza motores a pasos para el movimiento en los ejes, lo cual facilita el cálculo y obtención de mediciones de forma precisa. Para la rotación completa de un motor a pasos es necesario ingresarle una secuencia de pasos que pueden variar de acuerdo a la circunstancia ya que algunas secuencias ofrecen más precisión, así como otras que ofrecen más torque al motor que se utilizan para mecanismos que requieren más fuerza para mover una carga, en nuestro caso nosotros podemos utilizar la secuencia que se muestra en la Tabla 1 puesto que solo necesitamos de una fase, la secuencia FULL STEP de dos fases se suelen utilizar en aplicaciones que requieren torque, por lo que el gasto energético es mayor y no nos es útil en nuestro caso ya que lo que requerimos es precisión.

La secuencia para los motores a pasos de la forma paso completo una fase como se muestra en la Tabla 1, paso completo dos fases como se muestra en la Tabla 2 y el medio paso como se muestra en la Tabla 3.

Pasos	**Bobina 1A**	**Bobina 2A**	**Bobina 1B**	**Bobina 2B**
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Tabla 1. Secuencia para un movimiento FULL STEP de una fase.

Pasos	**Bobina 1A**	**Bobina 2A**	**Bobina 1B**	**Bobina 2B**
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Tabla 2. Secuencia para un movimiento FULL STEP de dos fases.

Pasos	Bobina 1A	Bobina 2A	Bobina 1B	Bobina 2B
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Tabla 3. Secuencia para un movimiento HALF STEP (Medio paso).

Para generar el movimiento del motor necesitamos generar una serie de combinaciones dentro del software del microcontrolador dado que el programa se encuentra ejecutando varias cosas simultáneamente como la obtención del ángulo de incidencia del láser, el control del motor y la generación de la frecuencia de 20kHz dentro del mismo, como se mencionó en 3.4 el microcontrolador PIC18F4450 está diseñado con 4 temporizadores y se optó por utilizar el TIMER3, ya hemos hecho uso del TIMER0 para la generación de un señal cuadrada con una frecuencia de 20kHz como se demostró en 3.3, el TIMER1 se utilizó para el tiempo de muestreo del controlador PID, el TIMER2 se utilizó para generar un PWM(Pulse Width Modulation por sus siglas en ingles) para el control de velocidad del motor de la apertura de escaneo.

Para generar un retardo en la secuencia de los motores a pasos requerimos aunque tenemos algunos inconvenientes al generar un tiempo de retardo para este temporizador ya que si usamos el reloj interno que es de 48MHz no nos es posible crear el tiempo suficiente, por lo que el fabricante nos especifica que es posible introducir una señal de reloj externa para generar los retardos [32], Como se observa en la Figura 10 tenemos dos osciladores principales el cual utilizaremos un cristal de cuarzo de 32.786kHz que es necesario para generar periodos de retardo más grandes.

Para determinar el periodo que genera una frecuencia de 32.768kHz utilizaremos la siguiente formula:

$$T = \frac{1}{F} = \frac{1}{32768} = 0.0000030517\ segundos = 30.51\mu s$$

Con la utilización del timer3 integrado en el microcontrolador se utilizó un cristal de cuarzo de 32.768kHz para generar retardos que van desde 30.51µs hasta 6 segundos. Nos resulta un periodo de 30.51µs como mínimo para generar retardos. El timer3 tiene un registro (TMR3H y TMR3L) [32] de 16 bits el cual nos da la posibilidad de almacenar un máximo de 2^{16} que son 65536 valores distintos, si nosotros multiplicamos el periodo generado por el cristal de cuarzo de 32.768kHz por el número máximo del registro del timer3 obtendremos una cantidad máxima de 6 segundos que puede generar cuando no utilizamos un divisor de frecuencias (preescalador) de 8, al programarlo en el microcontrolador el timer3 tiene un preescalador que divide la frecuencia en partes, de esta forma es posible crear periodos de tiempo más grandes como se observa en la Figura 23.

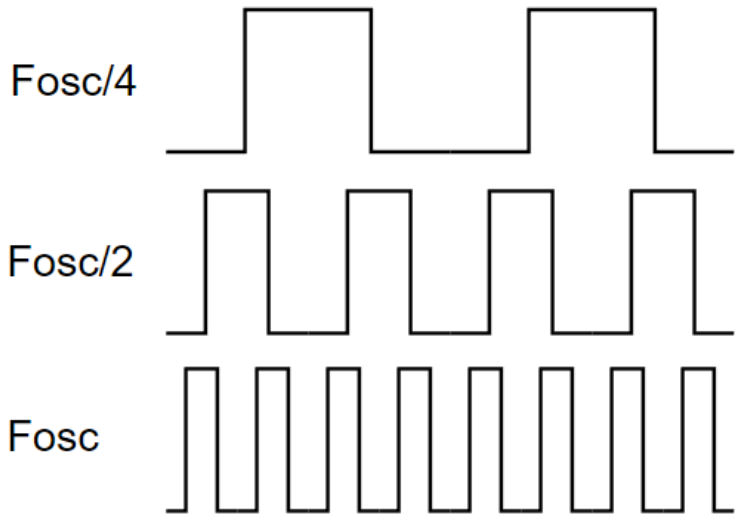


Figura 23.División de frecuencias por un preescalador.

En la Figura 24 se puede apreciar los fragmentos de códigos escritos para generar las señales necesarias para el funcionamiento del motor a pasos de acuerdo al modo de operación. Estas constantes son arreglos de datos que contienen la combinación en la secuencia correcta, de tal manera que al ejecutar el código de la secuencia solo es necesario incrementar un contador y el puerto D del microcontrolador será igual a dicha secuencia como se aprecia en la Figura 25.

```
const int full_step_1[4] = {0x01,0x02,0x04,0x08};
const int full_step_2[4] = {0x03,0x06,0x0c,0x09};
const int half step[8] = {0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09};
```

Figura 24. Fragmento de código de secuencias para motores a pasos.

Al implementar este código en el microcontrolador PIC18F4550 es posible descartar la necesidad de utilizar microcontroladores adicionales para cada motor, de esa forma es posible ahorrar en gastos por componentes y distribuir mejor el espacio para una tarjeta PCB embebida con todos los componentes necesarios para el funcionamiento del sistema de barrido láser.

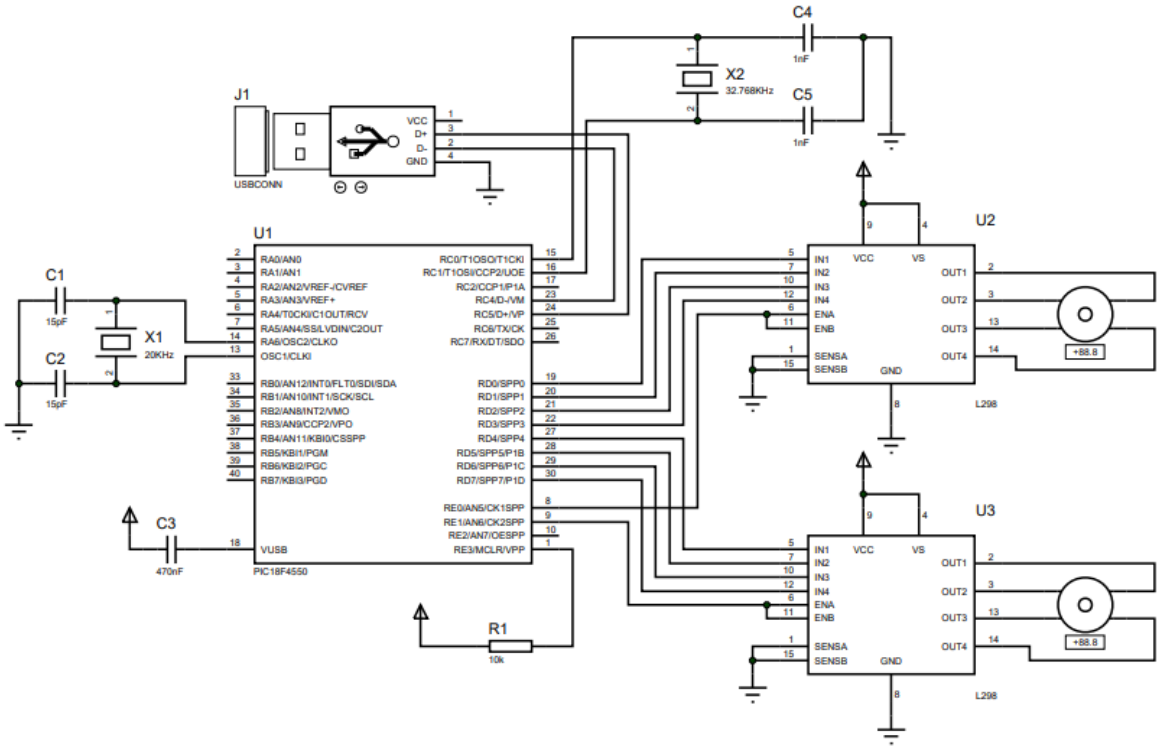


Figura 25. Diagrama eléctrico de conexión de motores a pasos.

Como el fabricante nos específica que podemos utilizar diferentes tipos de preescaladores para generar una señal de menor frecuencia y así poder contar los flancos de subida, al sumar un número determinado de flancos podemos generar una interrupción cada cierto tiempo.

Para calcular el tiempo necesario para generar la interrupción se utilizará la siguiente ecuación (12).

$$T_3 = 2^{16} - \frac{T * F_{osc}}{Preescaler} \quad (12)$$

Si nosotros deseamos un periodo de 12ms para cada paso nosotros cargaremos el valor al registro del TIMER3 que nos resulte en la ecuación (12), en este caso en particular nos resultó 65372.16 redondeando al valor más cercano obtenemos 65372, nosotros cargaremos el valor al registro del timer3, entonces comenzara a contará los pulsos de subida hasta que se desborde (Que pase de 65535 a 0) y generará una interrupción que será el cambio de pasos en el motor a pasos.

Para cargar el valor a dicho registro se utiliza la siguiente instrucción:

```
set_timer3(65372);
```

Figura 26. Código de carga a registro del TIMER3

3.5 Transmisión y recepción de datos con la PC

Para transmitir datos a través de un sistema de control a una computadora es necesario utilizar protocolos de comunicación adecuados de acuerdo a la aplicación que se desea. Para nuestra aplicación utilizaremos el protocolo USB-CDC (Universal Serial Bus -Communication Devices Class) el cual nos permite crear un puerto virtual del tipo serial en una PC

Con esa implementación de software es posible omitir circuitería de adaptadores de voltaje como el MAX232 el cual solo podría hacer que se pierda información en el momento de transmitir los datos. Con el avance de la tecnología cada vez es menos frecuente la aparición de puertos DB-9 por lo que la comunicación USB es mucho más accesible para nuestras aplicaciones.

El microcontrolador PIC18F4550 tiene un módulo de comunicación USB2.0 que puede ser programada con el fin de transmitir y recibir datos. Como se explicó anteriormente el microcontrolador se encuentra trabajando con una frecuencia interna de 48MHz lo cual es necesario para el uso del módulo USB a una alta velocidad de transmisión de hasta 1.17MBytes/s. Ver Figura 27.

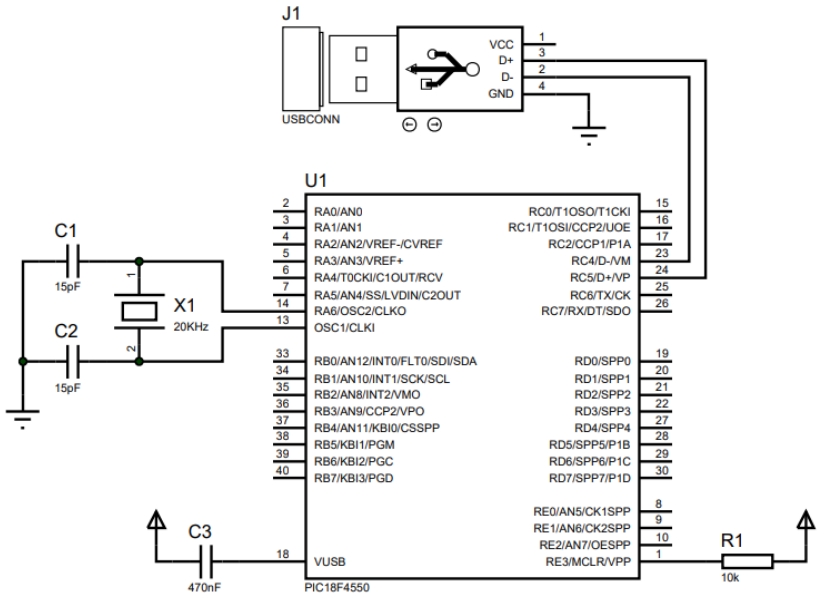


Figura 27. Conexion de PIC18F4550 con protocolo USB20.0 a la PC

3.6 Memoria interna del microcontrolador PIC18F4550

Como se mencionó el capítulo 2.4 los microcontroladores poseen de una memoria ROM, sin embargo, existen diferentes tipos de memorias ROM como las memorias PROM (Programmable Read-Only Memory), EPROM (Erasable Programmable Read-Only Memory) y EEPROM (Electrically Erasable Programmable Read-Only).

Nosotros nos enfocaremos en las memorias EEPROM el cual nuestro microcontrolador posee una memoria EEPROM de 256 Bytes [32], esta memoria la utilizaremos para almacenar información sobre la configuración de nuestro sistema.

3.7 Programación de algoritmos para procesamiento de datos

Para el control del TVS es necesario implementar algoritmos que disminuyan la cantidad de trabajo al operador para tomar datos experimentales con esto es imprescindible el desarrollo de algoritmos mediante programación capaz de realizar dicha tarea.

Gracias a la implementación de los comandos de control el programa es capaz de adaptarse a las necesidades del usuario para tomar datos experimentales. Dentro del código de programación se añadieron comandos con el fin de tener un control más robusto del sistema, de esa manera es capaz de adaptarse a cualquier situación que se presente como cambios en el diseño mecánico del TVS como implementación de motores con una mayor resolución, la distancia entre SA y PL para aplicaciones donde el tamaño es un problema. Entre los comandos programados podemos encontrar, la dirección de rotación de los motores a pasos que controlan la dirección de disparo del rayo láser, el tiempo entre pasos, tipo de secuencia de pasos para los motores, etc.

A continuación, se explican algunos de los comandos establecidos para el control del sistema de visión técnico.

laser_full_step_one: Establece el tipo de secuencia en paso completo una fase en el posicionador láser.

laser_full_step_two: Establece el tipo de secuencia paso completo dos fases en el posicionador láser.

laser_half_step: Estable el tipo de secuencia de medio paso en el posicionador láser.

z_full_step_one: Estable el tipo de secuencia en full step una fase en el eje Z.

z_full_step_two: Estable el tipo de secuencia en full step dos fases en el eje Z.

z_half_step: Estable el tipo de secuencia en half step en el eje Z.

motor_z_inv: Invierte la secuencia de pasos del motor del eje Z.

motor_laser_inv: Invierte la secuencia de pasos del motor del posicionador láser.

steps_time: Establece el tiempo entre pasos (en milisegundos).

laser_steps: Indica pasos a recorrer en el posicionador láser.

Kp: Establece el kp del controlador PID.

Ki: Establece el ki del controlador PID.

Kd: Establece el kd del controlador PID.

St: Establece el tiempo de muestreo del controlador PID.

Setpoint: Establece la consigna del controlador PID.

z_steps: Indica pasos a recorrer en la dirección del eje Z.

info: Muestra al usuario la información del controlador.

Con la adición de estos comandos será necesario enviarlos por el canal de comunicación hacia el microcontrolador como se menciona en 3.5, cabe mencionar que el posible crear cualquier comando para realizar una acción determinada.

3.8 Almacenamiento en memoria y creación de archivos

Todos los sistemas de adquisición de datos manejan grandes volúmenes de información y es importante almacenarlos para un procesado y toma de decisiones de acuerdo a la tarea que se busca ejecutar o mejorar, en nuestro caso es importante almacenar todos los datos obtenidos para nuestro sistema de visión técnico. Existen datos que debemos procesar para obtener características del medio en el que se encuentra, obstáculos, texturas etc. Toda esa información dada por el fototransistor es procesada para determinar ángulos de incidencia, disparo y distancias con la ley de senos [36]. Ese volumen de información será almacenado con un formato específico para un rápido acceso y guardado.

Debemos tener en consideración la forma de guardado de los datos, por lo que si deseamos enviar el TVS en lugares de poco acceso para el ser humano el volumen de información no será el óptimo para generar mapas o nubes de puntos con el fin de graficarlos en una computadora, para ello se desarrolló un algoritmo para el almacenamiento de información ya sea por parte de la interfaz gráfica como el propio sistema de control del TVS.

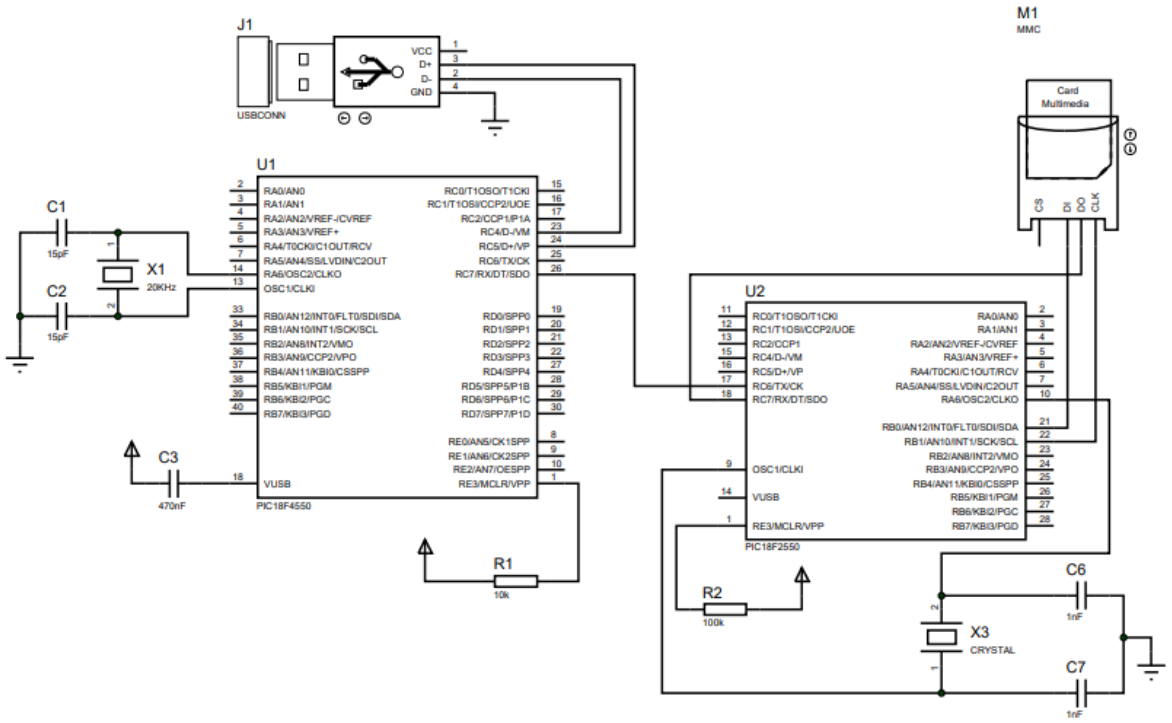


Figura 28. Diagrama de conexión con PIC18F2550 para almacenamiento.

Para el almacenamiento se utilizó un microcontrolador diferente debido a su alto consumo de recursos, ya que para escribir en una memoria SD es necesario comunicarse por medio de protocolo SPI, el cual disponemos en ambos microcontroladores. En la Figura 28 podemos observar el conexionado para la comunicación entre los microcontroladores por medio del puerto Serial.

CAPÍTULO 4. Diseño y desarrollo de interfaz gráfica

4.1 Interfaz gráfica

En la actualidad la mayoría de controles de sistemas se manejan gracias a una interfaz gráfica por lo cual es necesario de utilizar un lenguaje de programación adecuado para su utilización. Para el desarrollo de este interfaz se ha optado por utilizar el lenguaje de programación de Python por la fácil implementación y su gran escala de utilización a nivel mundial. Python es un lenguaje muy flexible que por su sencilles puede parecer pseudocodigo, es un lenguaje orientado a objetos y cuenta con una gran variedad de librerías es considerado como un lenguaje de muy alto nivel, ver Figura 29. También es denominado como open-source lo cual es de gran utilidad para escribir código libre [37].

Existen diversas aplicaciones escritas en Python como el sistema operativo Ubuntu, Blender un programa para modelado 3D. También es ampliamente utilizado en centros de investigación como el CERN (Organización Europea para la investigación nuclear) por sus librerías exclusivamente dedicadas a la investigación y análisis de datos [38].

Para el control del TVS se desarrolló un software para la toma de datos y control del mismo como se observa en la Figura 30

Figura 29. Logo del lenguaje de Python

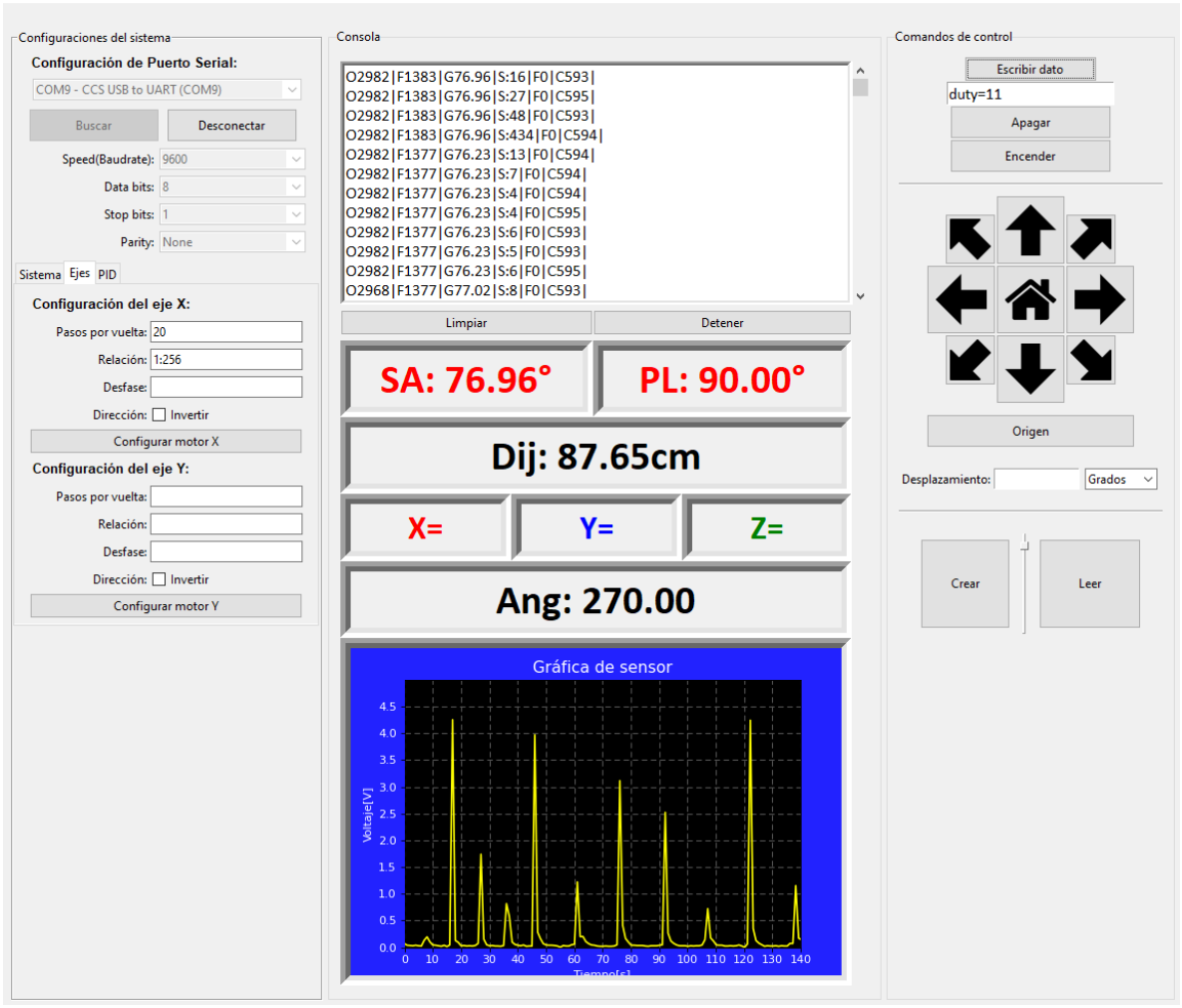


Figura 30. Interfaz gráfica para el control de sistema de visión técnico (TVS)

Para el desarrollo de esta aplicación se utilizó el lenguaje de programación Python debido a su gran utilización en la actualidad y su gran versatilidad [37], la aplicación desarrollada recibe los datos obtenidos por el TVS, y a su vez podemos controlar todo el sistema incluso en el futuro se podrían programar algoritmos de barrido para obtener diferentes resultados. El lenguaje de Python tiene diferentes librerías para el desarrollo de interfaces gráficas entre ellas Tkinter, PyQt, wxPython solo por mencionar algunas de ellas [39].

La librería para el diseño gráfico que se utilizo fue Tkinter que nos ofrece diferentes widgets para la visualización de información, así como etiquetas, cuadros de texto y para el control de diferentes eventos como botones, y deslizadores.

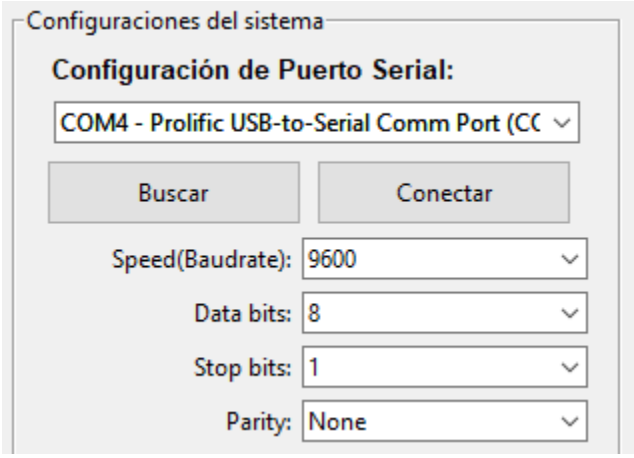


Figura 31. Panel de configuración para el puerto serial.

En la Figura 31 tenemos el panel para configuración de comunicación, esto nos ayuda en caso de tener varios puertos COM en nuestra computadora, el cual no da la libertad de seleccionar el puerto utilizado y configurarlo de acuerdo a la velocidad de transmisión y recepción de datos, el tamaño de datos a transmitir, los bits de parada, y la paridad que nos ayuda a detectar errores en la comunicación. Para el control del protocolo de comunicación serial RS-232 existe una librería en Python llamada PYSERIAL el cual nos proporciona una conexión para transmitir y recibir datos vía puerto serie [40].

La idea de diseñar una interfaz gráfica es la versatilidad para el control ya sea en el procesamiento y almacenamiento de los datos obtenidos, para diferentes análisis y pruebas existen muchas variables que pueden ser alteradas en el control de nuestro sistema de visión técnico como la velocidad del motor DC, la velocidad de cambio de rotación de los motores a pasos, la frecuencia de los pulsos de referencia, etc. Cada uno de estos parámetros puede ser modificado mediante la interfaz y se muestran algunos de dichos parámetros en la Figura 30.

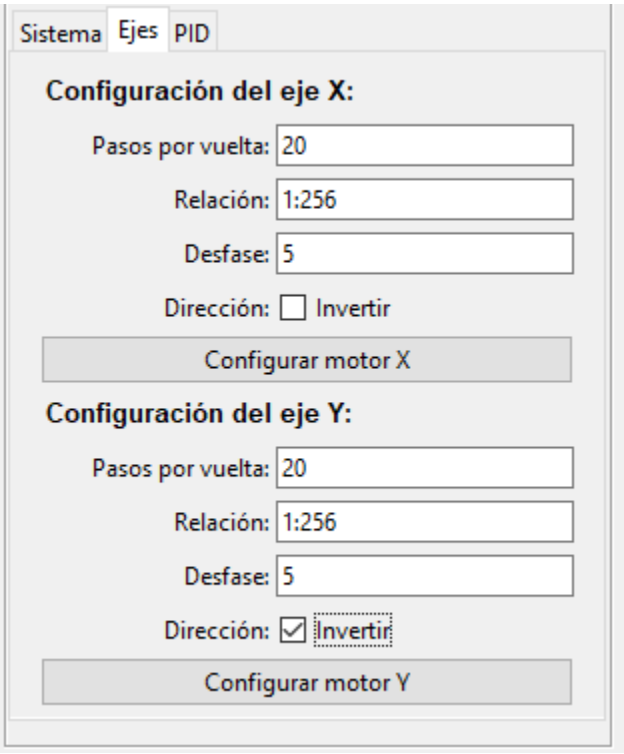


Figura 32. Panel de configuración de motores a pasos

En la Figura 32 podemos observar el panel de configuraciones de movimiento de los ejes, esto abarca el control que se va utilizar en los motores a pasos, como el número de pasos, la relación de engranaje (en caso de requerirlo), el desfase (si existe) y también la posibilidad de invertir el giro del motor en caso de una mala conexión, de esta forma evitaremos cambiar conexiones todo gracias a contar con el software.

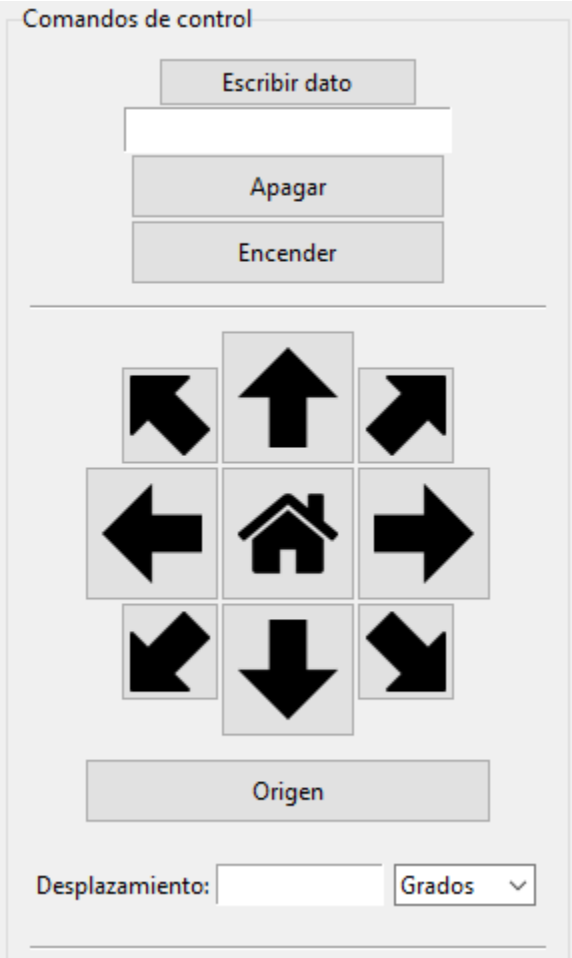


Figura 33. Panel de control de posición.

También contamos con un control de posición el cual podemos ver en la Figura 33 el cual tenemos disponible algunos botones que nos ayudan a cambiar la dirección de disparo de rayo láser, el cual tenemos la ventaja de desplazar el disparo en grados o en pasos del motor, también se cuenta con un botón con la opción de establecer un origen, en posteriores actualizaciones se espera poder tener un control más amplio como el de apagar o encender el láser, para el ahorro de energía y así poder tener un sistema con mayor tiempo de trabajo.

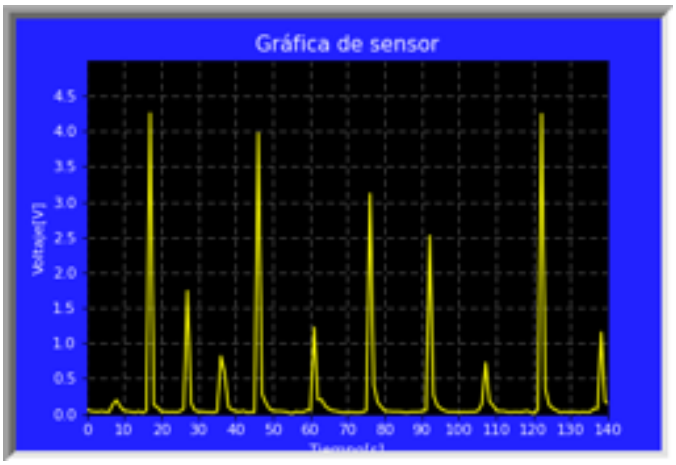


Figura 34. Señal graficada del fototransistor.

Para la visualización de datos por la interfaz gráfica se ha incluido un monitor, en el cual podemos observar en la Figura 34. Para la programación de esta herramienta se utilizaron dos librerías Matplotlib el cual es de gran ayuda para la creación de gráficas de diferentes tipos siendo algunas dinámicas [41]. También la librería de Numpy el cual se utiliza ampliamente en el campo de la investigación ya que es posible crear matrices y arreglos de datos matemáticos pudiendo realizar ecuaciones complejas [42].

Con la interfaz gráfica desarrollada con los medios para comunicación con el usuario es necesario poder procesar la información recibida y mostrarla en los apartados ya mencionados anteriormente, por lo que es importante determinar los algoritmos mediante programación de todos los procesos que se llevaran a cabo por software como se verá en el siguiente tema.

4.2 Formato de comandos recibidos por sistema de visión técnico

Como se mencionó en 3.7 disponemos de una lista de comandos elaborada para el control del TVS, lo cual nos da una gran posibilidad de programar comandos para realizar tareas específicas.

```
PIC18F4550>> O1500|F570|G90|S3000|C780\n
```

Figura 35. Sentencia para envió de variables por microcontrolador PIC18F4550

Para una mejor eficiencia del programa se ha establecido un formato para el envío de datos por el canal de comunicación, a continuación, se explica a detalle el funcionamiento del mismo.

Hemos seleccionado letras en mayúsculas para representar el valor de las variables que deseamos enviar, ó para los pulsos contados por el optoacoplador, F para los pulsos contados por el fototransistor, G para grados siendo este calculado dentro del microcontrolador, S para la lectura del canal análogo ADC del microcontrolador que va desde 0 hasta 1023, siendo 0 para 0V y 1023 para una lectura máxima de 5V. El cual están separados por una línea vertical como este carácter '|' como se observa en la Figura 35.

Con el formato que hemos establecido es más fácil interpretar los datos para almacenarlos por medio de la interfaz gráfica y esto gracias a el reciclado del código encargado de extraer los valores de la cadena mandada por el microcontrolador. Con esta implementación de este formato es posible enviar información diferente añadiendo una letra, sílaba, palabra o símbolo seguido del valor requerido sin espacio.

4.3 Adquisición y procesamiento de datos

Como se mencionó en 3.7 y en 4.2 obtenemos los datos con el formato establecido explicado, la interfaz gráfica se encarga de procesarlos mediante algoritmos de programación con el cual podemos crear plantillas con los comandos establecidos. En la Figura 36 observamos un diagrama de flujo que sigue el microcontrolador para controlar la apertura de escaneo cuando requerimos tomar datos experimentales.

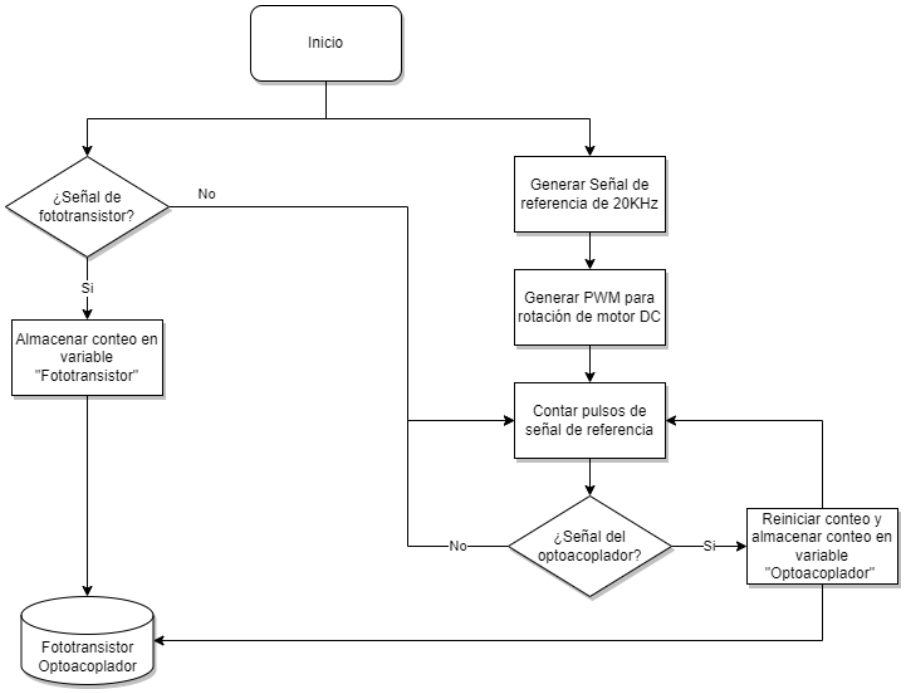


Figura 36. Diagrama de flujo para apertura de escaneo.

En la Figura 37 podemos observar un diagrama de flujo que es usado para la comunicación del TVS con la computadora, el cual hace uso del diagrama de flujo mostrado anteriormente en la Figura 36. Lo que podemos hacer uso de otros diagramas de flujo para realizar diversas tareas.

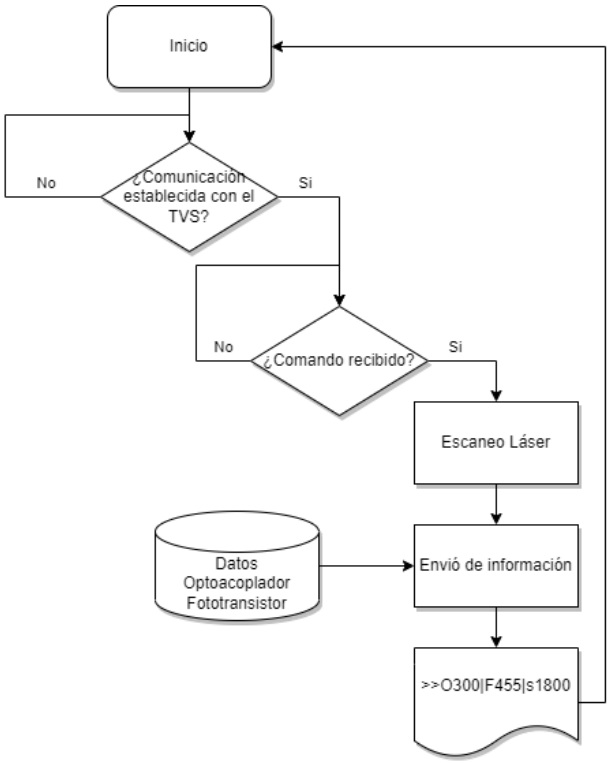


Figura 37. Diagrama de flujo para el envío de datos

CAPÍTULO 5. Desarrollo de prototipo

Con el fin de comparar la eficiencia de los algoritmos de obtención de datos, ya sea con un prototipo en 3D o uno más profesional se ha optado por el desarrollo de un prototipo experimenta con diferentes características mecánicas y electrónicas

Para el desarrollo de este prototipo 3D se utilizó el programa especializado en diseño mecánico SOLIDWORKS, el cual nos ayuda en la elaboración de un diseño y simulación de los componentes con las dimensiones con las cuales requerimos como se muestra en la Figura 38.

El prototipo nombrado mini TVS fue impreso en 3D con plástico PLA de 1.75mm Ver Figura 40.

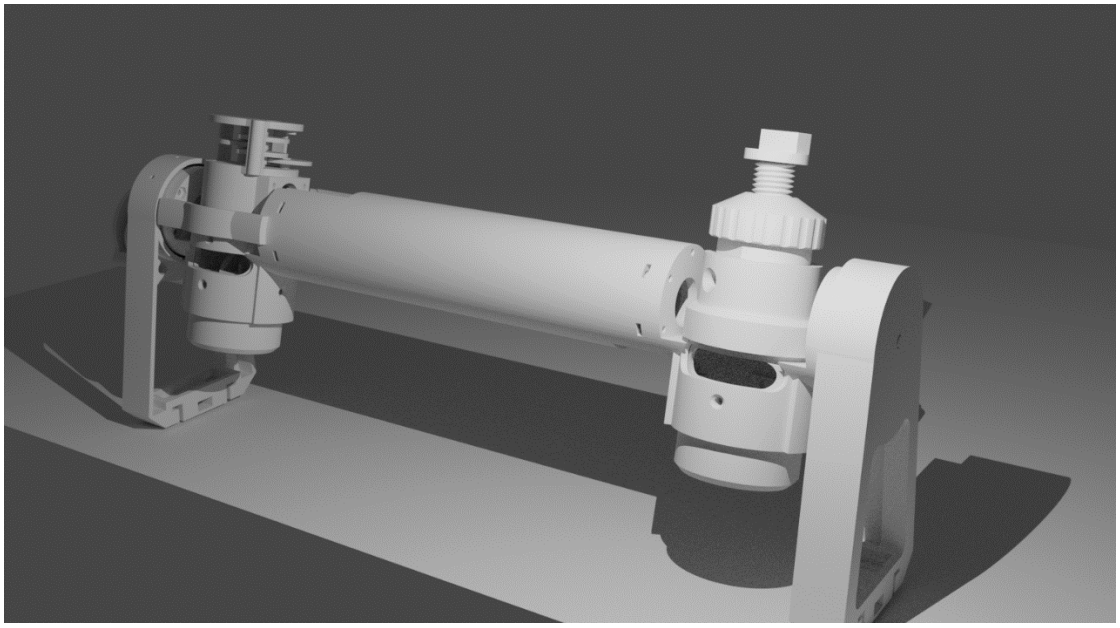


Figura 38. Diseño 3D de prototipo de sistema de visión técnico.

Para la elaboración de la apertura de escaneo fue necesario conseguir los componentes ideales de acuerdo al tamaño planteado, como el motor mabushi RF-310T-11400 que se observa en la Figura 39 con unas dimensiones de 24.5 mm de diámetro y el cual

posee un eje extendido de 20mm ideal para montar el espejo con corte de 45° y el disco ranurado junto con el optoacoplador.

Figura 39. Motor DC modelo RF-310T-11400

En cuanto a las características mecánicas del posicionador láser, se utilizó un motor a pasos genérico de 15mm de diámetro con una resolución de 20 pasos por revolución, lo cual es difícil para nuestro uso debido a su baja resolución, por lo que se diseñó una transmisión reductora con una relación de 1:256 para incrementar su precisión y su torque con lo cual nos brinda 5120 pasos por revolución y una resolución de 0.0703° por paso.

5.1 Características de prototipo TVS en impresión 3D

Para el prototipo 3D se consideraron las medidas de 20cm de largo para una experimentación en lugares de difícil acceso. El material de construcción fue de PLA (Poly-Lactic Acid por sus siglas en inglés), un polímero biodegradable que es muy sencillo para trabajar debido a su bajo índice de fundición que es de entre 190°C y 220°C además de que no emite gases tóxicos, cosa que hacen los materiales técnicos como el ABS y ASA [43]. Entre las aplicaciones encontramos de uso doméstico, en la industria alimentaria, biomédicas, médicas como orales, ortopédicas, craneofaciales para cirugías plásticas etc. Las técnicas empleadas para el uso de este polímero podemos encontrar la inyección por moldeo, extrusión, moldeo por soplado, termoformado etc. [44]. Para este prototipo es fácil implementar modificaciones para realizar distintos tipos de experimentación, como la longitud del TVS, siendo diseñadas de mayor o menor longitud. ver Figura 40.

Una de las ventajas principales de este prototipo es que al ser de un tamaño relativamente pequeño es posible usar componentes de bajo consumo como el láser, los motores a pasos para el posicionamiento, lo cual se traduce en reducción de consumo de corriente y eso nos resulta beneficioso ya que podemos implementar este prototipo en robots móviles y hacer más eficiente el uso de la energía disponible para un mayor tiempo de funcionamiento en experimentaciones.

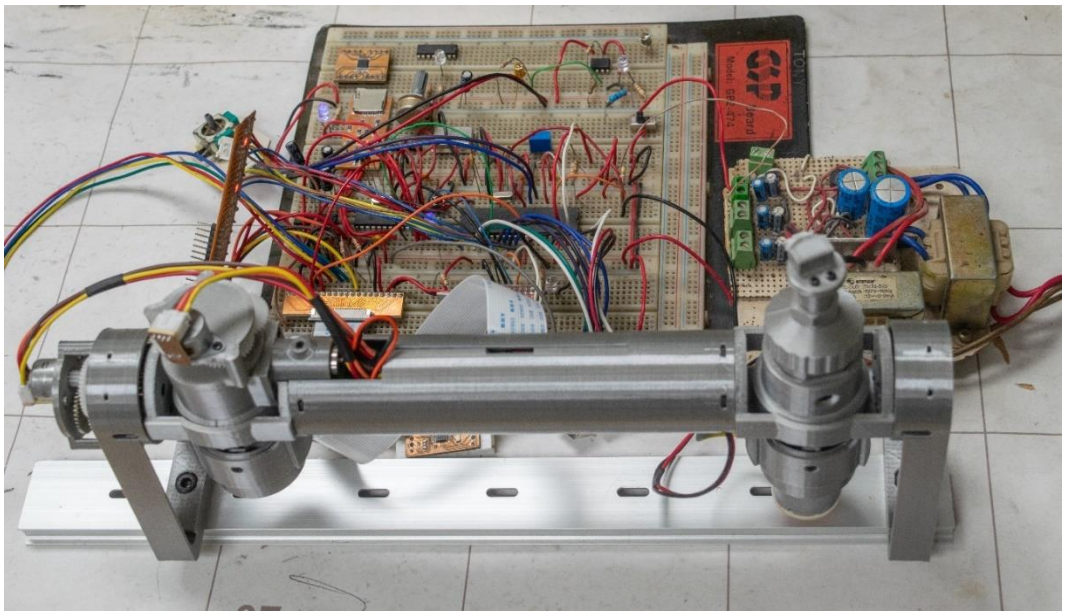


Figura 40. Prototipo impreso en 3D.

A continuación, se explican a detalle algunas de las características del prototipo 3D.

- Longitud de 20 centímetros
- Láser de 5V con una potencia de 10mW (teórico)
- Motores a pasos con una resolución de 20 pasos por revolución
- Relación de engranaje de 1:256 (posicionador láser)
- Motor DC de 6V corriente nominal de 120mA
- Fototransistor modelo VBPW77NA
- Peso aproximado de 528g.

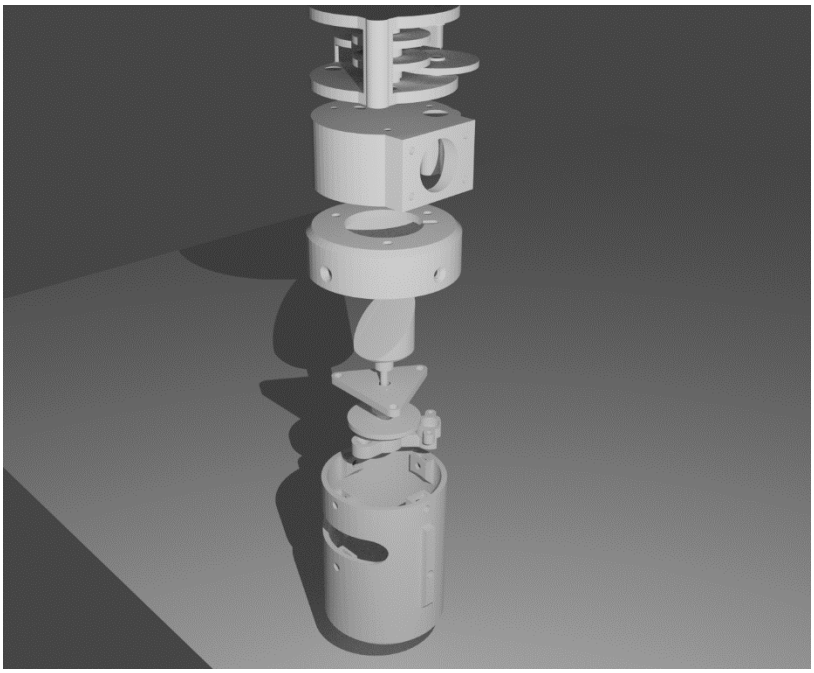


Figura 41. Vista explotada de la apertura de escaneo.

Para el desarrollo del mini TVS se diseñaron aproximadamente 14 piezas para la apertura de escaneo, ver Figura 41, 12 piezas para el posicionador láser, y 6 piezas para la estructura que da soporte al prototipo, con un total de 32 piezas diseñadas, excluyendo motores, engranes y piezas móviles.

CAPÍTULO 6. Experimentación y resultados

Para la experimentación se siguió un procedimiento para comprobar posibles errores como se muestra en la Figura 42 el cual al realizar una experimentación se documentan los resultados y si estos van de acuerdo a lo esperado, se sigue el diagrama hasta lograr tener resultados consistentes.

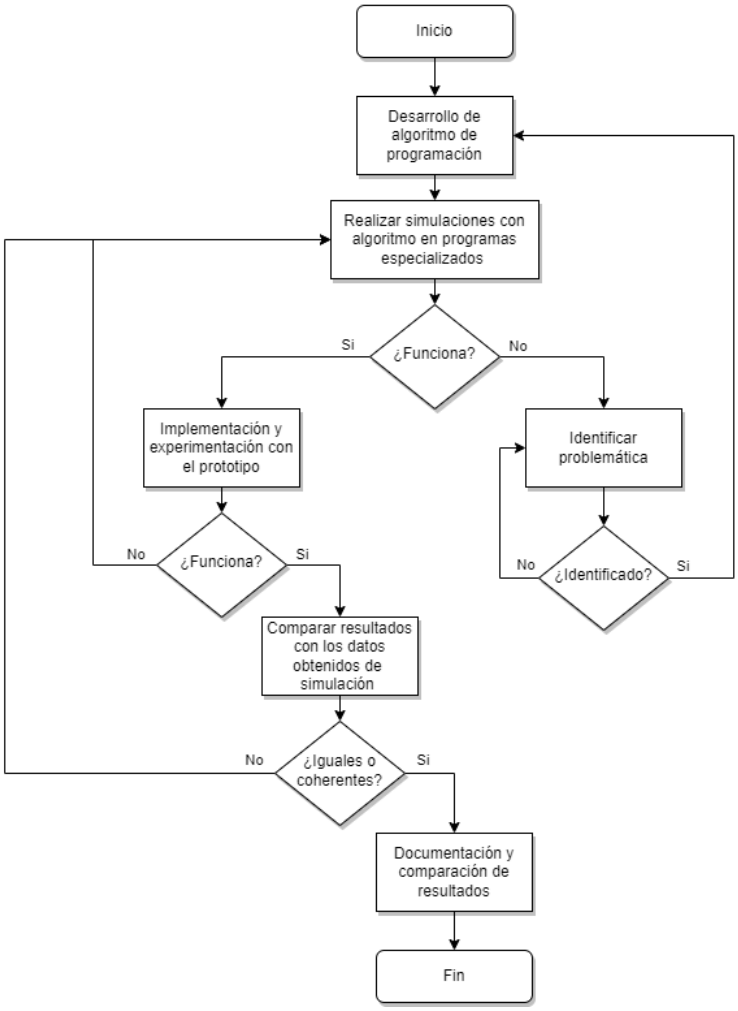


Figura 42. Diagrama de flujo para realizar experimentación.

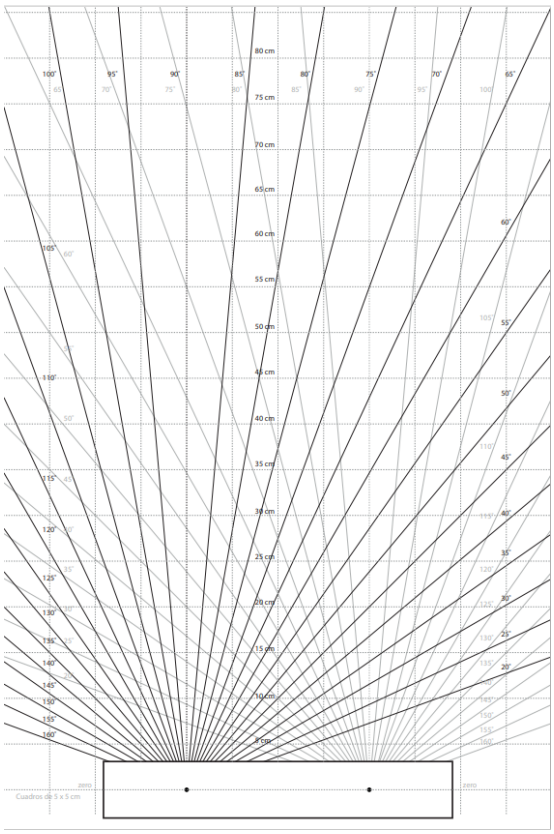


Figura 43. Hoja de referencia con ángulos marcados de acuerdo al diseño mecánico del TVS.

Para comparar los resultados se ha replicado el experimento de tomas medidas y obtener el valor del ángulo de incidencia del rayo láser en la apertura de escaneo, el experimento se ha hecho a partir de 100 muestras, a continuación, se puede observar en la Tabla 4 las medidas y los errores obtenidos en la Figura 43 podemos observar una hoja de 30x90 cm donde se colocó el TVS y manualmente se posiciona el láser en determinado ángulo, siendo de 77° el ángulo de incidencia en la apertura de escaneo a una distancia de aproximadamente de 89cm .

A continuación, se muestra la Tabla 4 donde obtenemos los resultados de las 100 muestras tomadas, donde registramos el número de pulsos contados por el optoacoplador y fototransistor de acuerdo a la señal de referencia como se menciona en 3.3.

			Angulo incidencia (Grados)			
	Optoacoplador	Fototransistor	Valor medido	Valor real	Error	Error relativo
1	3423	1568	74.9080	77.00	-2.092	2.717%
2	3423	1568	74.9080	77.00	-2.092	2.717%
3	3423	1573	75.4338	77.00	-1.566	2.034%
4	3423	1576	75.7493	77.00	-1.251	1.624%
5	3423	1576	75.7493	77.00	-1.251	1.624%
6	3423	1576	75.7493	77.00	-1.251	1.624%
7	3423	1576	75.7493	77.00	-1.251	1.624%
8	3423	1576	75.7493	77.00	-1.251	1.624%
9	3423	1577	75.8545	77.00	-1.145	1.488%
10	3423	1577	75.8545	77.00	-1.145	1.488%
11	3419	1577	76.0486	77.00	-0.951	1.236%
12	3419	1577	76.0486	77.00	-0.951	1.236%
13	3419	1577	76.0486	77.00	-0.951	1.236%
14	3419	1578	76.1538	77.00	-0.846	1.099%
15	3419	1578	76.1538	77.00	-0.846	1.099%
16	3419	1578	76.1538	77.00	-0.846	1.099%
17	3415	1578	76.3485	77.00	-0.652	0.846%
18	3415	1578	76.3485	77.00	-0.652	0.846%
19	3415	1577	76.2430	77.00	-0.757	0.983%
20	3419	1608	79.3127	77.00	2.313	3.003%
21	3415	1609	79.6164	77.00	2.616	3.398%
22	3406	1609	80.0646	77.00	3.065	3.980%
23	3406	1609	80.0646	77.00	3.065	3.980%
24	3406	1609	80.0646	77.00	3.065	3.980%
25	3419	1571	75.4168	77.00	-1.583	2.056%
26	3419	1571	75.4168	77.00	-1.583	2.056%
27	3419	1571	75.4168	77.00	-1.583	2.056%
28	3419	1571	75.4168	77.00	-1.583	2.056%
29	3419	1571	75.4168	77.00	-1.583	2.056%
30	3419	1571	75.4168	77.00	-1.583	2.056%
31	3419	1580	76.3644	77.00	-0.636	0.825%

32	3425	1583	76.3883	77.00	-0.612	0.794%
33	3425	1583	76.3883	77.00	-0.612	0.794%
34	3425	1573	75.3372	77.00	-1.663	2.159%
35	3425	1573	75.3372	77.00	-1.663	2.159%
36	3425	1573	75.3372	77.00	-1.663	2.159%
37	3425	1583	76.3883	77.00	-0.612	0.794%
38	3425	1583	76.3883	77.00	-0.612	0.794%
39	3425	1583	76.3883	77.00	-0.612	0.794%
40	3430	1583	76.1458	77.00	-0.854	1.109%
41	3430	1573	75.0962	77.00	-1.904	2.472%
42	3430	1573	75.0962	77.00	-1.904	2.472%
43	3430	1573	75.0962	77.00	-1.904	2.472%
44	3430	1573	75.0962	77.00	-1.904	2.472%
45	3425	1573	75.3372	77.00	-1.663	2.159%
46	3430	1573	75.0962	77.00	-1.904	2.472%
47	3430	1573	75.0962	77.00	-1.904	2.472%
48	3435	1606	78.3144	77.00	1.314	1.707%
49	3430	1606	78.5598	77.00	1.560	2.026%
50	3430	1606	78.5598	77.00	1.560	2.026%
51	3515	1620	75.9175	77.00	-1.083	1.406%
52	3515	1620	75.9175	77.00	-1.083	1.406%
53	3515	1620	75.9175	77.00	-1.083	1.406%
54	3515	1620	75.9175	77.00	-1.083	1.406%
55	3515	1620	75.9175	77.00	-1.083	1.406%
56	3517	1620	75.8231	77.00	-1.177	1.528%
57	3517	1616	75.4137	77.00	-1.586	2.060%
58	3515	1616	75.5078	77.00	-1.492	1.938%
59	3517	1616	75.4137	77.00	-1.586	2.060%
60	3513	1616	75.6020	77.00	-1.398	1.816%
61	3513	1616	75.6020	77.00	-1.398	1.816%
62	3513	1616	75.6020	77.00	-1.398	1.816%
63	3513	1620	76.0120	77.00	-0.988	1.283%
64	3513	1620	76.0120	77.00	-0.988	1.283%
65	3517	1620	75.8231	77.00	-1.177	1.528%
66	3514	1604	74.3256	77.00	-2.674	3.473%

67	3514	1604	74.3256	77.00	-2.674	3.473%
68	3514	1604	74.3256	77.00	-2.674	3.473%
69	3514	1604	74.3256	77.00	-2.674	3.473%
70	3506	1619	76.2407	77.00	-0.759	0.986%
71	3506	1619	76.2407	77.00	-0.759	0.986%
72	3506	1619	76.2407	77.00	-0.759	0.986%
73	3504	1619	76.3356	77.00	-0.664	0.863%
74	3506	1619	76.2407	77.00	-0.759	0.986%
75	3504	1614	75.8219	77.00	-1.178	1.530%
76	3506	1614	75.7273	77.00	-1.273	1.653%
77	3506	1614	75.7273	77.00	-1.273	1.653%
78	3506	1614	75.7273	77.00	-1.273	1.653%
79	3506	1614	75.7273	77.00	-1.273	1.653%
80	3506	1614	75.7273	77.00	-1.273	1.653%
81	3506	1614	75.7273	77.00	-1.273	1.653%
82	3506	1614	75.7273	77.00	-1.273	1.653%
83	3506	1614	75.7273	77.00	-1.273	1.653%
84	3506	1614	75.7273	77.00	-1.273	1.653%
85	3501	1607	75.2442	77.00	-1.756	2.280%
86	3501	1607	75.2442	77.00	-1.756	2.280%
87	3501	1607	75.2442	77.00	-1.756	2.280%
88	3501	1605	75.0386	77.00	-1.961	2.547%
89	3501	1607	75.2442	77.00	-1.756	2.280%
90	3501	1607	75.2442	77.00	-1.756	2.280%
91	3501	1607	75.2442	77.00	-1.756	2.280%
92	3501	1611	75.6555	77.00	-1.344	1.746%
93	3501	1611	75.6555	77.00	-1.344	1.746%
94	3501	1611	75.6555	77.00	-1.344	1.746%
95	3503	1611	75.5609	77.00	-1.439	1.869%
96	3503	1609	75.3554	77.00	-1.645	2.136%
97	3510	1617	75.8462	77.00	-1.154	1.499%
98	3510	1617	75.8462	77.00	-1.154	1.499%
99	3510	1612	75.3333	77.00	-1.667	2.165%
100	3510	1615	75.6410	77.00	-1.359	1.765%

Tabla 4. Comparación de datos obtenidos por apertura de escaneo (SA)

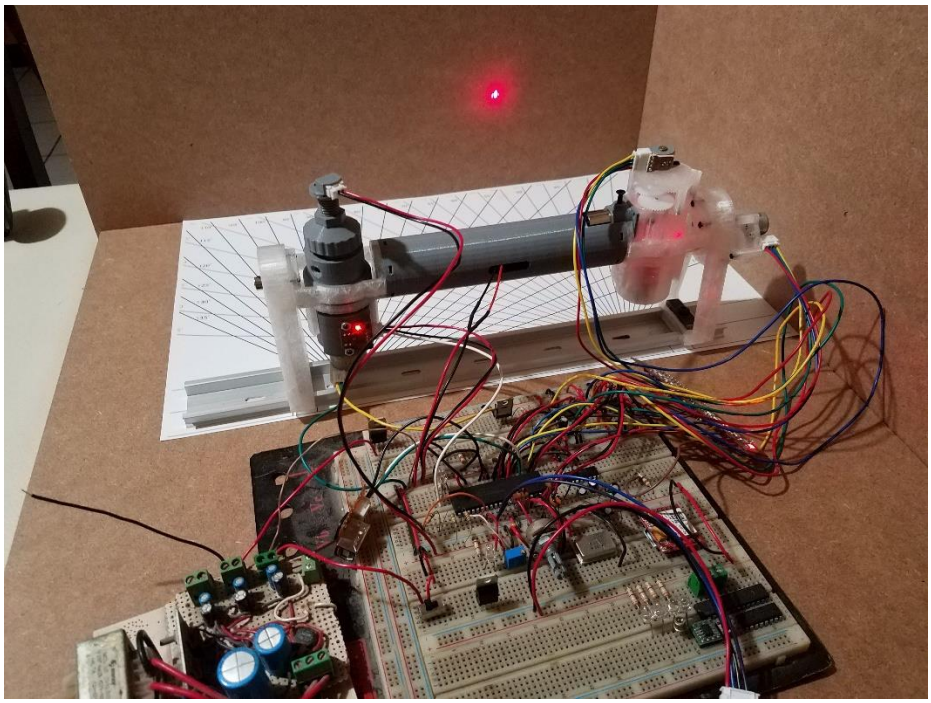


Figura 44. Experimentación con prototipo mini TVS.

De los resultados generados por el prototipo obtenemos el siguiente histograma de datos y podemos sacar algunas conclusiones como que el valor más obtenido se encuentra entre 75.15° y 75.97°, aunque también se presentan algunos valores dispersos debido a variables externas desconocidas y esto genera una curva no deseada.

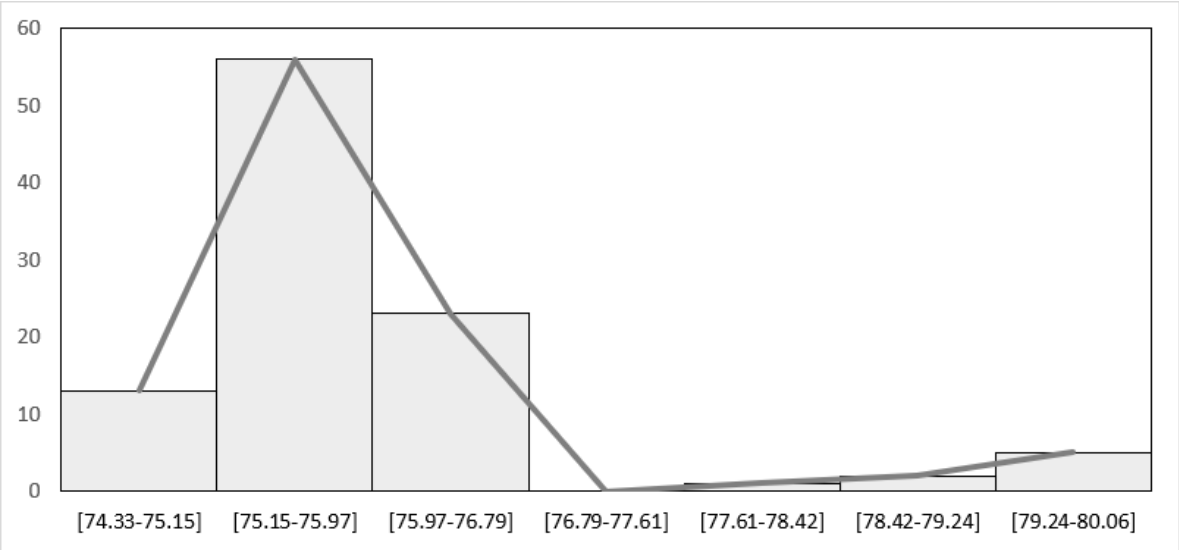


Figura 45. Histograma de datos de ángulo de incidencia de apertura de escaneo

En los resultados experimentales nos enfocamos directamente en el ángulo de incidencia en la apertura de escaneo, debido a su gran importancia en el cálculo de distancias y coordenadas cartesianas. Como se mencionó en 3.1 el incremento de la velocidad de trabajo del microcontrolador nos proporciona mayor toma de información y programación de algoritmos sin comprometer seriamente la precisión del sistema como la generación de la señal de referencia mencionado en 3.2, y como hemos visto en 3.3 la adquisición de la señal del fototransistor nos da una mayor resolución debido a la precisión en obtener el centro de la señal generada por el fototransistor.

Trabajo futuro

Con la implementación de los algoritmos, es posible programar diferentes microcontroladores para poder obtener los mismos resultados, aunque debemos tener en consideración las limitaciones que pueda poseer, como protocolos de comunicación con el USB o puerto serial, también en cuanto a las interrupciones externas, temporizadores entre otras. Existen algunos microprocesadores modernos que poseen mayor velocidad de trabajo, sin duda alguna eso mejorara la eficiencia del sistema de control, aunque existen algunos microprocesadores basados en tecnologías más avanzadas sino que ahora son diseñadas con más de un núcleo como la Raspberry pi pico, o con velocidades superiores a los 200MHz como la ESP32 y además con dos núcleos lo cual es de gran utilidad para programar procesos diferentes en cada uno de los núcleos sin afectar el rendimiento del programa.

Conclusiones

Durante la realización de este trabajo de maestría en ingeniería se ha logrado los siguientes resultados principales.

He desarrollado un algoritmo con el cual el procesamiento de los datos es más eficiente aprovechando las capacidades máximas del microcontrolador, utilizando los temporizadores internos podemos generar una señal cuadrada de 20kHz virtualmente mediante software sin la necesidad de circuitos externos, incluso podemos programar frecuencias más altas y más bajas en caso de requerirlas.

Como se mencionó en 3.1, 3.3 y se demostró en la sección 6 las lecturas obtenidas por el sistema de barrido láser tienen un error de presión de hasta $\pm 5\%$ de los valores reales y de acuerdo al algoritmo programado es posible implementarlo en otros tipos de microcontroladores de Atmega, Raspberry pi pico, Arduino etc. De tal forma que somos capaces de aprovechar el máximo uso del mismo teniendo todo centralizado en un solo procesador para la obtención de datos.

Con la implementación del algoritmo hemos obtenido una mejor distribución de comunicación entre microcontrolador y computadora, como se demuestra en el 3.5, se mejoró el enlace y tener un canal de comunicación único siendo este el puerto USB-CDC con velocidades de transmisión de hasta 1.17 kbytes por segundo.

Entre las limitaciones del sistema de barrido láser se encuentran las diferentes distribuciones de tarjetas de control ya sea para controlar los motores a pasos o para transformar la señal de entrada del sensor, también para la comunicación del microcontrolador con la computadora, para ello se ha solventado con la utilización de los propios recursos de interfaz de microcontrolador de los cuales está diseñado como las interrupciones externas que se programaron para contabilizar el inicio del algoritmo de revolución dadas por espejo que se encuentra en la apertura de escaneo para el cálculo del ángulo en el que el espejo detecta la reflexión del rayo láser en la superficie que se encuentra en el campo de visión del sistema. También los temporizadores

internos que pueden trabajar como programas en paralelo mientras se ejecuta el código de transmisión de datos por el puerto serie en el hilo principal del microcontrolador.

El almacenamiento de información dentro de la memoria EERPROM del microcontrolador podría ser de gran utilidad en el uso de robot móviles ya que el robot puede obtener la información del sistema, por lo que será muy útil tener en cuenta donde se almacenan los datos dentro del mismo. También nos brinda de una ventaja y es que es posible almacenar datos inclusive si nuestro sistema pierde la energía eléctrica.

En cuanto al procesamiento de los datos obtenidos se utilizó un lenguaje de programación de alto nivel que nos brinda una rápida programación e implementación de nuevo código para diferentes campos de investigación de procesamiento de datos, ya sea el mapeado de superficies etc. Con una gran versatilidad para configurar los parámetros del sistema, como por ejemplo el número de pasos de los motores a pasos, el tipo de paso que se requiere para tomar datos, puesto que se puede obtener diferentes presiones cuando se desea posicionar el disparo del rayo láser, etc.

Fue creada una interfaz gráfica utilizando las ventajas de diferentes librerías desarrolladas específicamente para el análisis de datos como Pandas, Numpy entre otras y estas son compatibles con el formato de MATLAB y también para diseño de gráficos dinámicos como Matplotlib. El cual nos es de gran utilidad para la visualización de información en forma de gráficas y tablas como fue demostrado en el capítulo 5 y 6 donde observamos los datos obtenidos del fototransistor graficados.

Referencias

[1] S. Tsugawa, «Vision-based vehicles in Japan: Machine vision system and driving control systems,» *IEEE Transactions on industrial electronics,* pp. 398-405, 1994.

[2] J. C. Rodríguez Quiñonez, «Médicion de parámetros biométricos por medio de barrido láser dinámico,» 2013.

[3] A. Broggi, P. Cerri y P. C. Antonello, «Multi-resolution vehicle detection using artificial vision,» *IEEE Intelligent Vehicles Sysmposium,* pp. 310-314, 2004.

[4] N. Ayache, Artificial vision for mobile robots: stereo vision and multisensory perception, MIT Press, 1991.

[5] M. Ivanov, L. Lindner, O. Sergiyenko , J. C. Rodríguez Quiñonez y M. Rivas López, «Mobile robot path planning using continuous laser scanning,» *Optoelectronics in machine vision-based theories and applications,* pp. 338-372, 2019.

[6] J. Li, M. Chen, X. Jin, Y. Chen, Z. Dai y Z. Ou, «Calibration of a multiple axes 3-D laser scanning system consisting of robot, portable laser scanner and turntable,» *Optik,* vol. 122, pp. 324-329, 2011.

[7] K. Ramani y Y. Fang, «Computers mimic human perception of 3-D shapes,» vol. 45, 2011.

[8] N. Kumar, P. Belhumeur, A. Biswars, D. W. Jacobs, W. J. Kress y I. C. López, «Leafsnap: A computer vision system for automatic plant species identification.,» *European conference on computer vision,* pp. 502-516, 2012.

[9] M. A. Shahin y S. J. Symons, «A machine vision system for granding lentils,» *Canadian biosystem engineering,* vol. 43, pp. 7.7-7.14, 2001.

[10] I. Maglogiannis y C. N. Doukas, «Overview od advanced computer vision systems for skin lesions characterization,» *IEEE transaction on information technology in biomedicine,* pp. 721-733, 2009.

[11] T. Brosnan y D. W. Sun, «Inspection and grading of agricultural and food products by computer vision systems,» *Computers and electronics in agriculture,* pp. 193-213, 2002.

[12] O. Sergiyenko, W. Flores Fuentes y P. Mercorelli, Machine vision and navigation, Springer International Publishing, 2020.

[13] J. C. Rodríguez Quiñonez, O. Sergiyenko, L. C. Básaca Preciado, V. Tyrsa, A. G. Gurko, M. A. Podrygalo y D. Hernández Balbuena, «Optical monitoring of scoliosis by 3D medical laser scanner,» *Optics and lasers in engineering,* vol. 54, pp. 175-186, 2014.

[14] J. C. Rodríguez Quiñonez, O. Sergiyenko, F. F. Gonzales Navarro, L. C. Básaca Preciado y V. Tyrsa, «Surface recognition improvement in 3D medical laser scanner using Levenberg-Marquardt method,» *Signal Processing,* vol. 93, nº 2, pp. 378-386, 2013.

[15] L. Lindner , O. Sergiyerko, J. C. Rodríguez Quiñonez, M. Rivas López, D. Hernández Balbuena, W. Flores Fuentes, V. Tyrsa y F. N. Murrieta Rico, «Mobile robot vision system using continuos laser scanning for industrial application,» 2016.

[16] O. Real Moreno, J. C. Rodríguez Quiñonez, O. Sergiyenko, L. C. Básaca Preciado, D. Hernández Balbuena, M. Rivas López y W. Flores Fuentes, «Accuracy improvement in 3D laser scanner based on dinamic triangulation for autonomous navigation system,» pp. 1602-1608, 2017.

[17] Vishay Semiconductors, «BPW77NA, BPW77NB,» 2008. [En línea]. Available: https://www.vishay.com/docs/81527/bpw77n.pdf.

[18] M. Rivaz López, J. C. Rodríguez Quiñonez, O. Sergiyenko, W. Flores Fuentes, R. Ráscon Carmona y L. Lindner, «Improve three-dimensional point localization accuracy in stereo vision systems using a novel camera calibration method,» *International Journal of advanced robotic systems,* vol. 17, 2020.

[19] W. Flores Fuentes, M. Rivas López, O. Sergiyenko, J. C. Rodríguez Quiñonez, D. Hernández Balbuena y J. Rivera Castillo, «Energy center detection in light scanning sensors for structural health monitoring accurary enhancement,» pp. 2355-2361, 2014.

[20] J. C. Rodríguez Quiñonez, O. Sergiyenko, V. Tyrsa, L. C. Básaca Preciado, M. Rivas López, D. Hernández Balbuena, p y M. Peña Cabrera, «3D body & medical scanners technologies: Methodology and spatial discriminations,» pp. 13-317, 2011.

[21] O. Sergiyenko, V. Trysa, L. C. Basaca Preciado, J. C. Rodríguez Quiñonez, W. Hernandez, J. Nieto Hipólito, M. Rivaz López y O. Starostenko, «Electromechanical 3D Optoelectronics Scanners: Resolution Constrainst and possible ways of improvement,» 2011.

[22] L. C. Básaca Preciado, O. Sergiyenko, J. C. Rodríguez Quiñoez, V. Tyrsa, M. Rivas López y O. Starostenko, «Optical 3D laser measurement system for navigation of autonomous mobile robot,» *Optics and lasers in engineering,* pp. 159-169, 2014.

[23] E. M. Pérez, Microcontroladores PIC: sistema integrado para el autoaprendizaje, Marcombo, 2007.

[24] F. Valdéz y R. Pallás Areny, Microcontroladores fundamentos y aplicaciones con PIC, vol. 1149, Marcombo, 2007.

[25] S. Armstrong, «USART, SPI, and I2C: serial communication protocols.,» *Programming PIC Microcontrollers with XC8,* pp. 209-276, 2018.

[26] V. E. Vargas, C. A. Baena y A. F. Salcedo, «Design and Implementation of a Wired Intercommunication Prototype for Hospital,» 2018.

[27] E. E. Ramos Noriega, «Desarrollo de un módulo de comunicación serial-ethernet para aplicaciones de monitoreo y control remoto,» 2002.

[28] N. G. Forero Saboya, «Normas de comunicación en serie RS-232, RS-422 y RS-484,» *Revista ingenio libre,* pp. 86-94, 2017.

[29] S. Tamboli, M. Rawale , R. Thoraiet y S. Agashe, «Implementation of MODBUS RTU and Modbus TCP communication using Siemens S7-1200 PLC for batch process,» *2015 International conference on smart technologies and management for computing, communication, controls, energy and materiasl,* pp. 258-263, 2015.

[30] M. Morris Mano y J. F. Gonzales, Diseño digital, Pearson Education, 2003.

[31] SanDisk, «SanDisk Secure Digital Card,» 2003. [En línea]. Available: https://www.convict.lu/pdf/ProdManualSDCardv1.9.pdf.

[32] Microchip Technology Inc, «PIC18F2455/2550/4455/4550 Datasheet,» 2006. [En línea]. Available: https://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf.

[33] TEXAS INSTRUMENTS, «LM555 Timer,» 2015.

[34] E. Garcia Breijo, Compilador C CCS y simulador PROTEUS para microcontroladores PIC, Marcombo, 2012.

[35] Texas Instruments, «LM111, LM211, LM311 Differential Comparators,» 2017. [En línea]. Available: https://www.ti.com/lit/ds/symlink/lm311.pdf?ts=1653011804538&ref_url=https%253A%252F%252Fgoogle.com.

[36] M. Rivas López, O. Sergiyenko, M. Aguirre, L. Devia, V. Tyrsa y I. Rendón, «Spatial data acquisition by laser scanning for robot or SHM task,» *2008 IEEE International Symposium on industrial Electronics,* pp. 1458-1462, 2008.

[37] Python, «Python,» 2001-2022. [En línea]. Available: https://www.python.org/.

[38] I. Challenger Perez, R. Díaz y R. A. Becerra García , «El lenguaje de programación Python,» *Ciencias Holguin,* vol. 20, pp. 1-13, 2014.

[39] Python, «Tkinter Interface Python for Tcl/Tk,» [En línea]. Available: https://docs.python.org/es/3/library/tkinter.html.

[40] «PySerial,» 2001-2020. [En línea]. Available: https://pyserial.readthedocs.io/en/latest/index.html.

[41] «Matplotlib,» 2012-2022. [En línea]. Available: https://matplotlib.org/stable/index.html.

[42] «Numpy,» 2022. [En línea]. Available: https://numpy.org/.

[43] P. McKeown y M. D. Jones, «The chemical recycling of PLA,» vol. 1, 2020.

[44] L. T. Sin, Polylactic acid: PLA biopolymer technology and applications, 2012.

Anexos

```
#int_timer3
void timer3_interrupt(){
   static signed int counter_laser = 0, counter_z = 0;
   unsigned int16 time;
   time = (unsigned int16)(65535.0 - (16384.0 * ((float)steps_time *
0.001)));
   if(adc_laser_value / 150 > 0)
      motor_laser_rotation? counter_laser++ : counter_laser--;
   else if(adc_laser_value / 150 < 0)
      motor_laser_rotation? counter_laser-- : counter_laser++;
   if(laser_steps != 0)
      if(laser_steps < 0){
         laser_steps++;
         motor_laser_rotation? counter_laser-- : counter_laser++;
      }else{
         laser_steps--;
         motor_laser_rotation? counter_laser++ : counter_laser--;
      }

   if(steps_laser_sequence == 0 || steps_laser_sequence == 1)
      counter_laser = limits(counter_laser,0,3);
   else if(steps_laser_sequence == 2)
      counter_laser = limits(counter_laser,0,7);
   step_laser = counter_laser;

   if(adc_z_value / 150 > 0)
      motor_z_rotation? counter_z++ : counter_z--;
   else if(adc_z_value / 150 < 0)
      motor_z_rotation? counter_z-- : counter_z++;
```

Figura 14. Código de programación para el control de motores a pasos

```
#include <18f4550.h>
#device ADC= 10
#fuses HSPLL, NOWDT, NOLVP, NOPROTECT, NOPUT, PLL5, USBDIV, CPUDIV1
#use delay(clock=48M)
#use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7)
#use fast_io(b)
#include <PID.c>
#include <string.h>
#include <string2.c>
#include <stdlib.h>

#byte porta=0xf80
#byte portb=0xf81
#byte portc=0xf82
#byte portd=0xf83
#byte porte=0xf84

#define USB_CDC_ISR() received_data()
void received_data();
#define USB_CABLE_IS_ATTACHED() input(PIN_B4)
#include <usb_cdc.h>

unsigned     int32     pulsos=0,     pulsos_f=0,     motor_pulses=0,
motor_pulses_f=0,      phototransistor=0,      phototransistorL=0,
phototransistorH=0;
PIDcontroller PIDcontrol;
float kp=10,ki=1,kd=0,st=0.2;
float setpoint = 60;
float maxLimit = 1000, minLimit = 0;
int8 duty;
float rpm;
int16 duty_value;
int16 phototransistor1=0, actual_photo, photo=0, anterior_photo=0,
comparator1=0;

#define SIZE_RX_BUFFER 30
char rx_buffer[SIZE_RX_BUFFER]="";
unsigned int count=0;
short bufferFI=FALSE;
char valor[10], atributo[10], comando[30];

int full_step_one[4]={0x01,0x02,0x04,0x08};
int full_step_two[4]={0x03,0x06,0x0c,0x09};
int half_step[8]={0x01,0x03,0x02,0x06,0x04,0x0c,0x08,0x09};
signed int step_laser=0, step_z=0;

signed int16 motor_speed = 0, adc_laser_value = 0, adc_z_value = 0;
unsigned int16 adc_x = 0, adc_y = 0;
unsigned int8 steps_laser_sequence = 0, steps_z_sequence = 0;
short motor_z_rotation=0, motor_laser_rotation=0;
unsigned int16 steps_time = 15, steps_time_test;
signed int16 laser_steps = 0, z_steps = 0;
```

```
int1 flagC=0;

#int_ext
void conteo(){
   pulsos_f=pulsos;
   motor_pulses+=1;
   pulsos=0;
}

#int_timer0
void timer0_interrupt(){
   //output_toggle(PIN_B7);
   pulsos+=1;
   set_timer0(254);
}

#int_timer1
void timer1_interrutp(){
   motor_pulses_f=motor_pulses;
   motor_pulses=0;
   rpm = ((float)motor_pulses_f / st) * 60.0;
   set_timer1((unsigned int16)(65536.0 - ((st * 32768.0) / 1.0)));
   PIDcompute(&PIDcontrol, setpoint, read_adc());
   output_toggle(PIN_B5);
}
signed int limits(signed int var, signed int nlim, int plim){
   if(var >= (plim + 1))
      return nlim;
   else if(var <= (nlim - 1))
      return plim;
   else
      return var;
}
#int_timer3
void timer3_interrupt(){
   static signed int counter_laser = 0, counter_z = 0;
   unsigned int16 time;
   time = (unsigned int16)(65535.0 - (16384.0 * ((float)steps_time
* 0.001)));
   if(adc_laser_value / 150 > 0)
      motor_laser_rotation? counter_laser++ : counter_laser--;
   else if(adc_laser_value / 150 < 0)
      motor_laser_rotation? counter_laser-- : counter_laser++;
   if(laser_steps != 0)
      if(laser_steps < 0){
         laser_steps++;
         motor_laser_rotation? counter_laser-- : counter_laser++;
      }else{
         laser_steps--;
         motor_laser_rotation? counter_laser++ : counter_laser--;
      }
```

```
   if(steps_laser_sequence == 0 || steps_laser_sequence == 1)
      counter_laser = limits(counter_laser,0,3);
   else if(steps_laser_sequence == 2)
      counter_laser = limits(counter_laser,0,7);
   step_laser = counter_laser;

   if(adc_z_value / 150 > 0)
      motor_z_rotation? counter_z++ : counter_z--;
   else if(adc_z_value / 150 < 0)
      motor_z_rotation? counter_z-- : counter_z++;

   if(steps_z_sequence == 0 || steps_z_sequence == 1)
      counter_z = limits(counter_z,0,3);
   else if(steps_z_sequence == 2)
      counter_z = limits(counter_z,0,7);
   step_z = counter_z;

   set_timer3(time);
   //set_timer3(65371);   //10ms
   //set_timer3(32768);   //2s
}
#int_comp
void comparator_interrupt(){
   output_toggle(PIN_B7);
   if(!C1OUT && flagC){
      phototransistorL = pulsos;
      flagC = 0;
   }
   if(C1OUT && !flagC){
      phototransistorH = pulsos;
      flagC = 1;
   }
}

#int_ad
void analog_conversion(){
      if (actual_photo + 150 < anterior_photo){
         photo = pulsos;
         anterior_photo = 0;
      }
}
void received_data(){
   //output_toggle(PIN_B7);
   char data;
   if(usb_cdc_kbhit()){
      if(count < SIZE_RX_BUFFER){
         data = usb_cdc_getc();
         if(data == '\n'){
            rx_buffer[count] = NULL;
            bufferFI = true;
            count = 0;
         }else{
```

```
            rx_buffer[count] = data;
            count++;
            bufferFI = false;
            if(count >= SIZE_RX_BUFFER - 1){
               rx_buffer[count] = NULL;
               count = 0;
               bufferFI = true;
            }else{
               rx_buffer[count] = NULL;
            }
         }
      }
   }
}
int1 toogle_bit(int1 bit){
   if(bit)
      bit=0;
   else
      bit=1;
   return bit;
}
void asign_values(char *cmd){
   char value[10];
   if(countchr(cmd, '=') > 0)
      split(cmd, value, findchr(cmd,'='));
   switch(cmd){
      case "kp": kp = atof(value); break;
      case "ki": ki = atof(value); break;
      case "kd": kd = atof(value); break;
      case "st": ST = atof(value); break;
      case "setpoint": setpoint = atof(value); break;
      case "laser_full_step_one": steps_laser_sequence = 0; break;
      case "laser_full_step_two": steps_laser_sequence = 1; break;
      case "laser_half_step": steps_laser_sequence = 2; break;
      case "z_full_step_one": steps_z_sequence = 0; break;
      case "z_full_step_two": steps_z_sequence = 1; break;
      case "z_half_step": steps_z_sequence = 2; break;
      case "info": printf(usb_cdc_putc,"kp: %.2f | ki: %.2f | kd:
%.2f | setpoint: %.2f\n\r",kp,ki,kd,setpoint); break;
      case         "motor_z_inv":         motor_z_rotation         =
toogle_bit(motor_z_rotation); break;
      case      "motor_laser_inv":      motor_laser_rotation      =
toogle_bit(motor_laser_rotation); break;
      case "steps_time": steps_time = atoi(value);break;
      case "laser_steps": laser_steps = atol(value);break;
      case "z_steps": z_steps = atol(value);break;
      case "mk_file": printf("mk_file=%s\n\r",value);break;
      case "write_data": printf("write_data=%s\n\r",value);break;
      case "close_file": printf("close_file\n\r");break;
      case "motor": printf("motor=%s\n\r",value);break;
      case                                          "motor_dir_inv":
printf("motor_dir_inv=%s\n\r",value);break;
```

```
      case "duty": printf("duty=%s\n\r",value);break;
      case "save":break;
      default:  printf(usb_cdc_putc,"Comando  no  reconocido\n\r");
break;
   }
}
void get_command(){
   char str_commands[30], str_command[20];

   if(countchr(rx_buffer,',') > 0){
      strcpy(str_commands,rx_buffer);
      strcpy(rx_buffer,'\0');
   }else{
      strcpy(str_command,rx_buffer);
      strcpy(rx_buffer,'\0');
      asign_values(str_command);
   }

   while(countchr(str_commands,',') > 0){
      rsplitr(str_commands,                              str_command,
findchr(str_commands,','));
      asign_values(str_command);
      if(countchr(str_commands,',') == 0)
         asign_values(str_commands);
   }
   bufferFI = false;
}
unsigned int sequence(unsigned int8 value, unsigned int steps){
   switch(value){
         case 0:return full_step_one[steps];break;
         case 1:return full_step_two[steps];break;
         case 2:return half_step[steps];break;
   }
}
void main(){

   usb_cdc_init();
   usb_init_cs();
   enable_interrupts(int_ext_L2H);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256|RTCC_8_BIT);
   setup_timer_1(T1_EXTERNAL|T1_CLK_OUT|T1_DIV_BY_1);
   setup_timer_2(T2_DIV_BY_16,74,1);
   setup_timer_3(T3_EXTERNAL|T3_DIV_BY_2);
   enable_interrupts(int_timer0);
   enable_interrupts(int_timer3);
   enable_interrupts(int_comp);
   enable_interrupts(global);

   setup_adc_ports(AN0_TO_AN7);
   setup_adc(ADC_CLOCK_DIV_64);
   set_adc_channel(7);
```

```
   setup_comparator(A0_A3_NC_NC_OUT_ON_A4);
   set_timer0(254);
   set_timer1(55706);
   set_timer3(65000);

   set_tris_a(0x0f);
   set_tris_b(0x13);
   set_tris_c(0x81);
   set_tris_d(0x00);
   set_tris_e(0x07);
   output_low(PIN_B7);           //code for test

   while(1){
      phototransistor = ((phototransistorH + phototransistorL) / 2);
      duty_value = ((74.0 + 1.0) * 4.0 * ((float)duty/100));
      set_pwm1_duty(duty_value);

      usb_task();
      if(bufferFI)
         get_command();
      if(usb_enumerated()){

printf(usb_cdc_putc,"O%lu|F%lu|G%.2f|S:%lu|F%lu|C%lu|\n\r",
pulsos_f,     phototransistor,     (((float)(phototransistor     *
360)/((float)pulsos_f)))-90,actual_photo,photo,comparator1);
      }

      set_adc_channel(5);
      delay_us(100);
      adc_x = read_adc();
      adc_z_value = (0.97751710 * ((float)adc_x)) - 500.0;
      set_adc_channel(6);
      delay_us(100);
      adc_y = read_adc();
      adc_laser_value = (0.97751710 * ((float)adc_y)) - 500.0;
      set_adc_channel(7);
      delay_us(100);
      anterior_photo = actual_photo;
      actual_photo = read_adc();
      delay_us(150);

      set_adc_channel(0);
      delay_us(100);
      comparator1 = read_adc();
      delay_us(600);

      portd       =       sequence(steps_z_sequence,step_z)       |
sequence(steps_laser_sequence,step_laser)<<4;
   }
}
```