CONACYT

CIDESI®

CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

PROYECTO INDUSTRIAL TERMINAL

Diseño, modelado y construcción de escáner 3D por proyección láser

PARA OBTENER LA ESPECIALIDAD DE:

Tecnólogo en Mecatrónica

PRESENTA:

Ing. José Gerardo Gómez Méndez

TUTOR ACADÉMICO Y DE PLANTA:

Dr. Leonardo Barriga Rodríguez

Querétaro, Qro., 2016

Resumen

Este proyecto consiste en el diseño asistido por computadora y la fabricación de un dispositivo mediante Hardware libre que permita la obtención de imágenes de objetos en 360º con la finalidad de ser utilizadas posteriormente para la reconstrucción tridimensional mediante el método de triangulación por proyección láser.

Índice

Listado de Figuras

Listado de Tablas

I. Introducción

En la actualidad existe una necesidad de obtener modelos tridimensionales de objetos, tal y como lo plantea B. Girod, a diferencia de los modelos bidimensionales, ellos representan información geométrica de la forma y textura de un objeto, y por tanto es mucho más útil que la información bidimensional que representa únicamente una simple reflexión local del mismo [1]. Un modelo 3D consta básicamente de una descripción numérica del objeto que puede ser utilizado para renderizar imágenes del mismo desde puntos arbitrarios y bajo condiciones arbitrarias de iluminación [2].

Se utilizan en diferentes ámbitos como el reconocimiento de objetos, aplicaciones de inspección y ensamblaje industrial de partes, calibración, navegación robótica autónoma, diagnósticos médicos, cartografía, digitalización de objetos y documentos históricos [3], aplicaciones multimedia, el comercio electrónico, los videojuegos [4], ingeniería inversa, realidad virtual, entre otras; sin embargo, un problema crítico sigue siendo el elevado costo de dispositivos para escanear con una alta calidad.

Una característica primordial del modelo obtenido es que debe ser editable, para proveer la capacidad de utilizar objetos físicos como el punto de inicio para el diseño de nuevos objetos en sistemas de modelado por computadora [2], adicionalmente estos deben poder ser rotados, escalados, deformados, observados mediante diferentes condiciones de luz, permitir cambios en su apariencia como color o material y observada la interacción del modelo con otros modelos en el ambiente [4].

Existen diversas técnicas para lograr una reconstrucción tridimensional, pero una de ellas está siendo muy utilizada actualmente por ingenieros que parte de la

captura de imágenes bidimensionales para procesarlas y obtener un modelo en tres dimensiones. Esta técnica llamada triangulación óptica reflectante sin contacto, presume tener un costo sumamente menor en comparación con los dispositivos actuales y una buena exactitud.

Para ello es necesario el obtener imágenes del objeto a digitalizar en cada uno de sus ángulos de visión, reflejando en él una luz estructurada, que en este caso es una línea láser y obtener parámetros certeros de distancia entre el emisor y el receptor, para posteriormente pasar a la etapa de procesamiento de imágenes en donde se lleva a cabo dicha reconstrucción.

Es por tanto que el objetivo de este proyecto se enfoca en la primera necesidad, en la obtención digital de imágenes, que además provea de un mecanismo eficiente y sencillo de utilizar, donde pueda ser posible adicionalmente, la alteración en el ángulo de rotación del objeto, así como la resolución del conjunto de fotografías.

II. Planteamiento del Problema

Actualmente existen diversos dispositivos comerciales que permiten digitalizar objetos tridimensionales; sin embargo, pese a su eficiencia, son sumamente costosos y no permiten probar con los nuevos algoritmos de reconstrucción, ya que utilizan los propios del fabricante. Así mismo se han investigado por más de dos décadas nuevos algoritmos y técnicas de reconstrucción que permitan disminuir el costo y el procesamiento necesario. Para lograr esto es necesario en una primera instancia la obtención de las características físicas del objeto a digitalizar, por lo tanto surge una necesidad imperativa de tener un dispositivo que permita facilitar la adquisición de dichos datos para posteriormente aplicar estos métodos.

III. Justificación

Uno de los métodos de reconstrucción que se ha estado utilizando en los últimos años es la técnica óptica de reflexión sin contacto, haciendo uso de la triangulación espacial con base a la proyección de un láser lineal. Esta tiene enormes ventajas, partiendo por la adquisición de datos que sólo requiere una cámara web y un láser lineal, teniendo como mayor beneficio la buena calidad que entrega posterior al post-procesamiento de los datos.

Es así como en este proyecto se diseñará un dispositivo que facilite tanto a científicos como investigadores la obtención de imágenes mediante parámetros establecidos con una buena calibración y que además disminuya la traslación de los objetos sobre un punto específico que generalmente se da cuando se hacen pruebas y que no permite hacer comparaciones certeras entre técnicas y algoritmos de reconstrucción.

IV. Objetivos

IV.I Objetivo General

Desarrollar un dispositivo de bajo costo basado en proyección de línea láser para la obtención de imágenes para digitalización de geometrías de superficies de objetos en 3D.

IV.II Objetivos Específicos

- Hacer el diseño asistido por computadora del prototipo de escáner tridimensional
- Construir un equipo basado en hardware libre para el control de movimiento y rotación de los objetos a digitalizar.
- Escribir el programa computacional que permitirá obtener las imágenes y rotar el objeto a escanear.
- Realizar pruebas de calibración del dispositivo.
- Escribir y detallar los planos para poder reconstruir el dispositivo.
- Defender el proyecto ante comité y obtener el título en la Especialidad de Tecnólogo en Mecatrónica.

V. Hipótesis

Es posible desarrollar un dispositivo de bajo costo basado en proyección de línea láser para la obtención de imágenes para digitalización de geometrías de superficies de objetos en 3D.

VI. Antecedentes

La digitalización de geometrías de superficies en objetos en tercera dimensión no es algo nuevo; de hecho, desde los años 80 se comenzó con investigaciones para obtener mediante imágenes dichas reconstrucciones de la superficie de un objeto determinado. En 1983 y 1985 respectivamente, R.A. Jarvis [5] y P. Besl [6] comenzaron con técnicas de visión por computadora para poder obtener las perspectivas mediante sensores de rango.

Sin embargo, la técnica para digitalizar objetos en 3D mediante análisis de imágenes con proyección láser en el contorno del cuerpo, fue desarrollada varios años después, puesto que cada uno de los componentes que la integran se desarrolló por separado. Un ejemplo de esto son los trabajos para triangulación de profundidades de G. Bickel [7] y W. Dremel [8]. Por su parte en 1986, se publicó en la novena conferencia internacional de láser un trabajo acerca de un radar mediante imágenes con láser [9].

En esa misma década, se diseñaron de forma separada sistemas de escaneo 3D con láser por J. Wang [9] y en los laboratorios NRC se construyó un sensor de imagen de rango [10]. El problema que se comenzó a observar fue que no había una buena precisión en la posición del objeto y por tanto las medidas no eran del todo certeras por lo que se hicieron varias investigaciones al respecto como las de Blais [11], Idesawa [12] y Steinbichler [13].

Posteriormente en 1995 Beraldin [14] y Blais [15] comenzaron con la reconstrucción digital mediante sistemas de visión para crear objetos computarizados en 3D pero con geometrías básicas, siendo presentado en 1996 durante la cuarta conferencia europea de visión por computadora, el desarrollo de una reconstrucción

que permitía obtener superficies confiables mediante la unión de varias imágenes de rango [16].

Otro problema surgió debido a que los datos 3D muchas veces no estaban estructurados por la complejidad que se tiene al trabajar de forma simultánea con varias imágenes, así que comenzaron a desarrollar algoritmos para corregir este tipo de fallas como lo hecho por M. Algori [17], J. Neugebauer [18] [19], Wheeler [20], Whitaker [21], M. Reed [22] y K. Klein [23].

Un inconveniente más que se encontró para el análisis de imágenes es el de la calibración de la cámara, que puede afectar considerablemente el resultado final si ésta no es óptima. Uno de los primeros trabajos para corregir esto fue el de R. Y. Tsai [24] con un eficiente método en 1996, así mismo se siguió tratando de mejorar como en los trabajos de Niem [25] y Zhang [26] para tener como resultado en 2003 en un método que permitía la autocalibración las cámaras [27]. Por último en 2005 se hizo un elaborado trabajo en cuanto a la comparación de las técnicas de calibración de cámaras digitales por D. Pizarro [28].

Por otro lado, entre 1995 y 2002 se ha continuado utilizado el análisis de imágenes en un sinfín de aplicaciones y desarrollando nuevas tecnologías para el escaneo tridimensional por diferentes métodos. Por ejemplo se encuentran los trabajos de W. Seales [29], H. Rushmeier [30], N. Amenta [31], E. Trucco [32] y U. Yilmaz [33] o el de M. Polleyfeys [34] que logró hacer la adquisición de los datos mediante un dispositivo "hand-held".

Ya para 1999 hubo una enorme preocupación por la reconstrucción de imágenes a color, presentando un artículo sobre eso F. Schmitt [35] en la conferencia internacional de procesamiento de imágenes, así como los trabajos de K. Pulli [36],

F. Chen [37] y J. González [38] para poder manejar de forma simultánea imágenes con color y con rangos.

Como anteriormente se mencionó, una de las partes importantes en la reconstrucción es la triangulación, la cual permite calcular las distancias, y que en el periodo comprendido entre 1995 y 1999 se lograron desarrollar mejores triangulaciones, haciendo uso del análisis del espacio-tiempo y haciéndolas dependientes de los datos, esto se ve en "Better Optical Triangulation through Spacetime Analysis" [39] y "Best data-dependent triangulations" [40].

Para finales del siglo XX se comenzaron a utilizar en conjunto todos los métodos, técnicas y algoritmos antes mencionados para construir diversos dispositivos que fueran portables y de bajo costo para la digitalización tridimensional, tal es el caso de Y. Matsumoto [41]. Así mismo se comenzó con la utilización de luz estructurada por D. Caspi [42] y E. Horn [43]. Un trabajo bastante interesante fue en el año 2000 que mediante este tipo de técnicas de visión por computadora, M. Levoy pudo escanear una estatua de Miguel Ángel [44].

Posteriormente de 2001 a la fecha se han continuado desarrollando este tipo de dispositivos y en especial los basados en proyección de luz estructurada y análisis de las imágenes mediante cámaras digitales, tal es el caso de A. Mülayim [45], C. Rocchini [46], U. Yilmaz [4], S. Winkelbach [47], H. Reyes [48] y C. Zamora [3], donde todos ellos coinciden en la utilización de una sola cámara fotográfica y un emisor led en forma de línea recta de color verde o roja con longitudes de onda desde los 500 hasta los 700 nanómetros.

El cambio principal en cada uno de los prototipos es la altura y ángulo de cada uno de los componentes. La cámara en posición frontal al objeto con un ángulo

menor a los 10º es lo más común, teniendo el láser apuntando a 45º; sin embargo, como se puede ver en dichos artículos, la posición, distancia y ángulo depende del sensor utilizado.

VII. Marco teórico

El escaneo tridimensional ha sido investigado por varias décadas y por tanto existen un sinnúmero de técnicas y algoritmos para lograrlo. En las Figuras 1 y 2 se muestra la taxonomía de los principales sistemas para la adquisición semi-automática de la forma de objetos en tres dimensiones, en donde la seleccionada en gris corresponde a la metodología para la cual este proyecto pretende enfocarse.

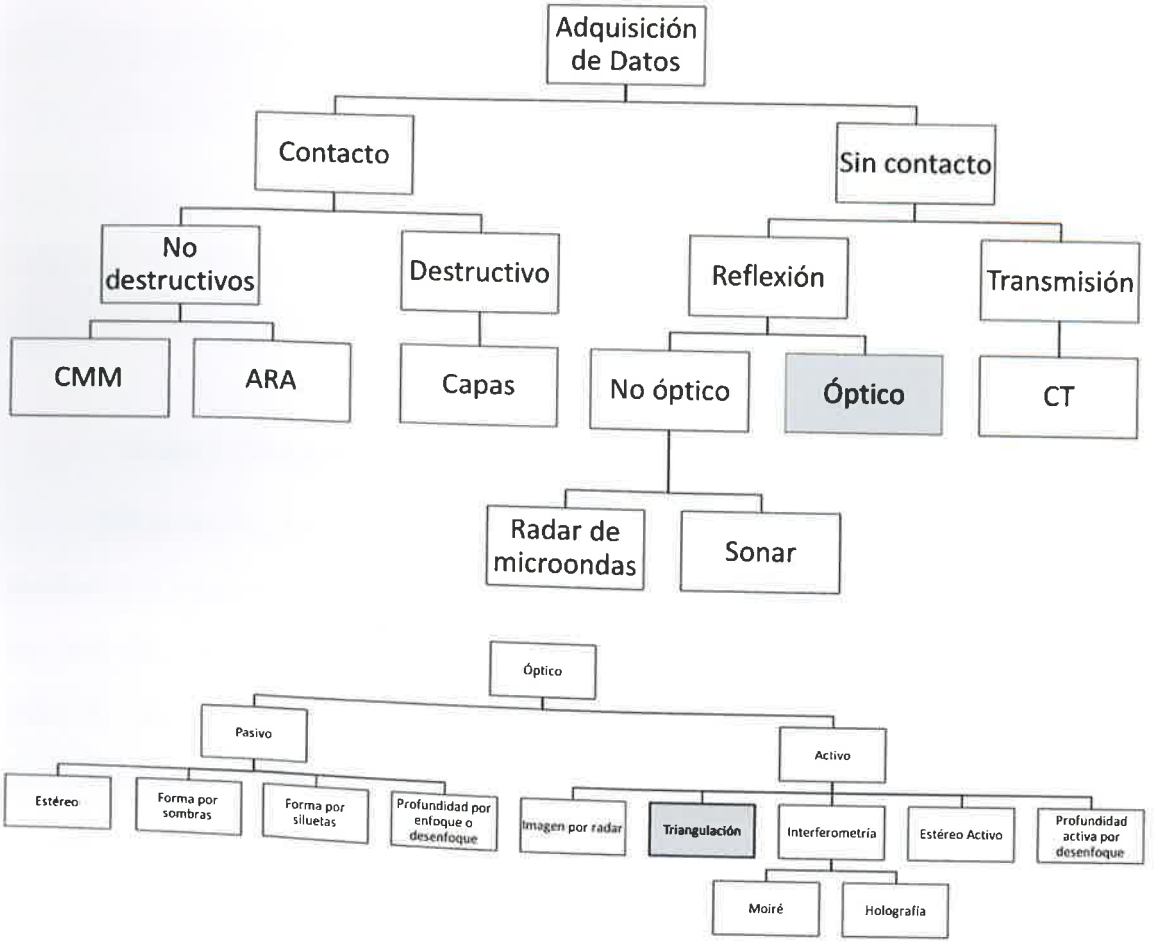


Figura 1 Taxonomía de los sistemas para la adquisición automática o semi-automática de la forma de objetos tridimensionales, de acuerdo con B. Curless [49].

Los sistemas ópticos activos son métodos de adquisición de datos sin contacto, lo cual es deseable en la mayoría de los casos, y pueden realizar los cálculos de las coordenadas del objeto mediante diversos algoritmos, uno de ellos es la triangulación. Esta clase de dispositivos constan de en un emisor, el cuál proyecta una luz estructurada en algún patrón específico, y un sensor, que generalmente es una cámara digital que tiene como objetivo adquirir las imágenes para poder analizar posteriormente en ellas el patrón de distorsión debido a la forma del objeto a digitalizar. En la Figura 2 se aprecia a detalle lo anteriormente descrito.

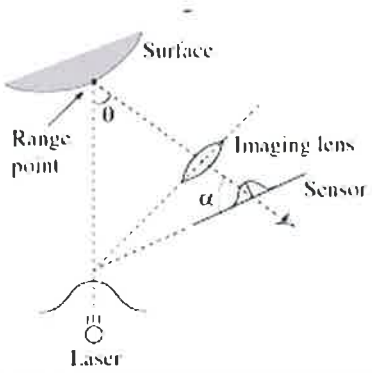


Figura 2 Triangulación mediante sensor óptico, de acuerdo con B. Curless [49].

De acuerdo con C. Rocchini, la información de profundidad es reconstruida mediante la técnica de triangulación, para lo cual es necesario que el emisor produzca luz [46]. Sin embargo esta luz puede ser coherente (los rayos de la fuente luminosa salen de ella paralelos entre sí [50]) o de luz incoherente (sus rayos son convergentes antes de cruzarse y divergentes después de los cruces [50]), pero debe seguir de manera forzosa un patrón, ya sea de punto, línea, cruz o incluso más complejos.

Para que el patrón elegido sea de utilidad, tiene que cumplir con ciertas características entre las que destacan: no puede ser alterado drásticamente por la reflectividad del objeto y la detección de la forma consistente del patrón, así como la reconstrucción de las características de indexación debe ser precisa y sencilla.

Una vez que se tiene seleccionado el emisor, el patrón que proyectará, así como la cámara a utilizar se fijan en las posiciones determinadas para la triangulación. Es indispensable conocer con exactitud la distancia al punto de origen de ambos. Con esto es posible lograr la reconstrucción, siendo la forma más sencilla la toma de fotografías conforme el objeto va rotando sobre un eje determinado, el cuál no puede variar su distancia respecto al emisor y sensor. Otra solución para lograr la reconstrucción espacial surge de proyectar planos sobre la superficie, sin embargo la complejidad para ello crece demasiado.

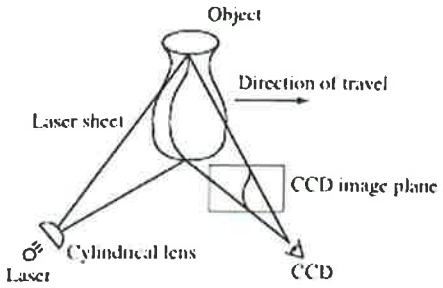


***Figura 3** Digitalización de objeto mediante la triangulación óptica activa, de acuerdo con B. Curless [49].*

VIII. Diseño y Construcción del Prototipo

VIII.I Materiales

VIII.I.I Dispositivos Adquiridos

Para construir un prototipo funcional se utilizaron los materiales que se describen a continuación.

1. Emisor

Figura 4 Módulo emisor de láser de línea roja.

Este láser fue utilizado como el emisor, debido a que proporciona una luz estructurada, con un patrón lineal que cumple con las características antes descritas. Sus características principales se detallan a continuación

Potencia de salida: Min 2,5 mW, 3.0mW típicos, Max 5.0mW

Corriente requerida: Min10mA, típicos 20mA, Max 25mA

Voltaje de funcionamiento: Min 2.3V DC, típicos 4.5V DC, Max 8.0V DC

Longitud de onda: 650nm

Ancho del punto enfocado: <2 mm a 3 metros de distancia.

Prueba práctica para la cabeza de punto: reflejo visible al objetivo aleatorio > 1000m.

2. Sensor

Figura 5 Cámara web Logitech® c170.

Para el sensor se utilizó la cámara web mostrada en la Figura 5 por las características que ésta posee y que cumple con los requerimientos necesarios. A continuación se detallan algunas de sus especificaciones técnicas.

Resolución: 5 megapíxeles.

Tecnología Logitech Fluid Crystal™: Ajusta automáticamente la frecuencia de cuadro, la nitidez, la saturación de color y el audio para obtener las mejores imágenes estáticas y en movimiento.

Tipo de enfoque: Enfoque constante.

Protocolo de comunicación: Hi-Speed USB 2.0 certified.

3. Motor

Para la rotación constante y la captura de fotografías en cada uno de sus ángulos es necesario un motor con el torque y exactitud suficiente para moverlo. En la Figura 6 se muestra el motor a pasos utilizado con su respectivo driver, y posteriormente se encuentran las especificaciones técnicas.

Figura 6 Motor a pasos 28BYJ-48 con driver.

Voltaje nominal 5v DC

Cantidad de Polos: 4

Relación de la variación de velocidad 1/64

Ángulo del paso 5,625 ° / 64

Resistencia 200Ω ± 7% (25 ° C)

Tracción idle-in> Frecuencia 600Hz

Tracción idle-oute> Frecuencia 1000Hz

Torque con traccion> 34.3 mN/m (120Hz)

Torque de posicionamiento automático> 34.3 mN/m

Resistencia aislada> 10M (500V)

Potencia eléctrica aislada 600VAC/1mA/1s

4. Computadora

El dispositivo utilizado para controlar el movimiento del motor, la toma de la secuencia de fotos y el servidor Web para controlarlo remotamente es una Raspberry Pi 1 modelo B como la mostrada en la Figura 7. Gracias a sus diversos puertos de entrada y salida, su bajo consumo energético, su capacidad de procesamiento y su arquitectura abierta fue seleccionada.

Figura 7 Raspberry Pi 1 modelo B.

Características técnicas

SoC (System on a Chip): Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB).

CPU: ARM 1176JZF-S a 700 MHz.

Juego de instrucciones: RISC de 32 bits.

GPU: Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1, 1080p30 H.264/MPEG-4 AVC.

Memoria (SDRAM): 512 MB compartidos con la GPU.

Entradas de vídeo: Conector MIPI CSI.

Salidas de vídeo: Conector RCA (PAL y NTSC), HDMI, Interfaz DSI para panel LCD.

Salidas de audio: Conector de 3.5 mm, HDMI.

Almacenamiento integrado: SD / MMC / ranura para SDIO.

Conectividad de red: 10/100 Ethernet (RJ-45).

Periféricos de bajo nivel: 8 x GPIO, SPI, I^2C, UART.

Consumo energético: 700 mA, (3.5 W).

Fuente de alimentación: 5 V vía Micro USB o GPIO header.

Sistemas operativos soportados: GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, RISC OS.

VIII.I.II Tornillería

En la Tabla 1 se detalla cada uno de los tornillos, varillas, tuercas y arandelas utilizados en la fabricación del prototipo, teniendo como uso principal la fijación de los componentes. Se pueden utilizar tanto milimétricos como estándar, siempre y cuando el diámetro y la longitud sean similares a lo que se describirá a continuación. En la sección de "cantidad y función" se menciona en donde fueron utilizados, aunque también se puede ver eso a detalle en los anexos 1-19.

Tornillos				
Nombre	**Diámetro mm**	**Longitud mm**	**Tipo de Cabeza**	**Cantidad y función**
Tornillo 1	2.7	16.3	Cilíndrica Ranurada	4: Sujetar driver para motor a la placa de acrílico. 2: Sujetar la Raspberry Pi a la placa de acrílico.
Tornillo 2	3.4	52.2	Cabeza Phillips	4: Sujetar el soporte de la cámara por un lado a la placa de acrílico y por el otro a la tarjeta electrónica a esta última.

Tornillo 3	3.4	25	Cabeza Phillips	2: Sujetar el motor a pasos en el soporte para motor.
Varilla roscada	6.35	250	Sin cabeza	2: Unir el soporte para el motor con la placa principal de acrílico, dejando entre ellos una separación de 170 mm.
Tuercas y Arandelas				
Nombre	**Diámetro Interior mm**	**Diámetro Exterior mm**	**Cantidad y función**	
Tuerca pequeña	2.7	5.5	6: Acoplar tornillo 1 a la placa de acrílico (4 en driver para motor y 2 en Raspberry Pi).	
Tuerca mediana	3.4	7.9	4: Acoplar tornillo 2 a la placa de acrílico (del soporte para motor). 8: Acoplar los mismos tornillos a la tarjeta electrónica.	
Tuerca grande	6.35	11	4: Acoplar varilla roscada al soporte para el motor. 4: Acoplar varilla roscada a la placa principal de acrílico.	

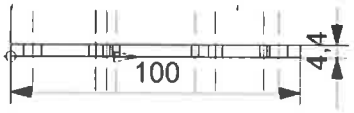
Arandela pequeña	2.7	4.7	12: Soporte de carga de apriete para todas las tuercas pequeña y cabezas de tornillo 1.
Arandela mediana	3.4	9.6	16: Soporte de carga de apriete para todas las tuercas medianas y cabezas de tornillo 2.
Arandela grande	6.35	18.6	8: Soporte de carga de apriete para todas las tuercas grandes.
Arandela extra grande	10.8	34.8	4: Soporte de carga de apriete para todas las tuercas grandes y al tener un excedente en la parte inferior, permite nivelar el dispositivo, funcionando como calza.

Tabla 1 Tornillería utilizada.

VIII.I.III Piezas impresas en 3D

Para el ensamble del prototipo fue necesario diseñar cuatro soportes mediante CAD (Diseño asistido por computadora) para ser posteriormente fabricadas en una impresora 3D utilizando Acrilonitrilo Butadieno Estireno (ABS) y dos placas de acrílico de 4mm de espesor para unir a ellas todos los soportes y los dispositivos.

a) Placa principal de acrílico



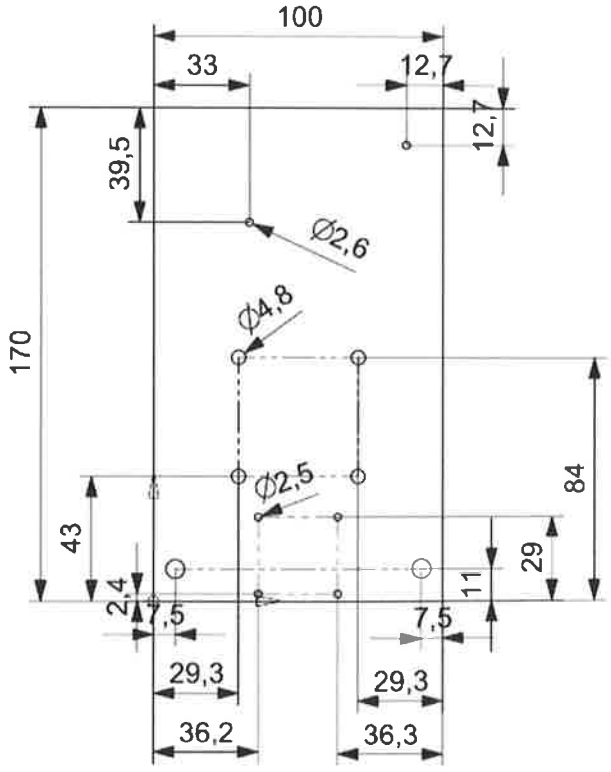


Figura 8 Perspectiva superior y frontal de la placa de acrílico.

En la Figura 8 se puede apreciar la placa que corresponde al soporte principal del escáner con sus respectivos hoyos para la tornillería. En ella irán el driver del motor, soporte para la cámara, placa para el láser, Raspberry Pi, la placa electrónica y las varillas para el motor.

b) Placa de acrílico para soporte del láser

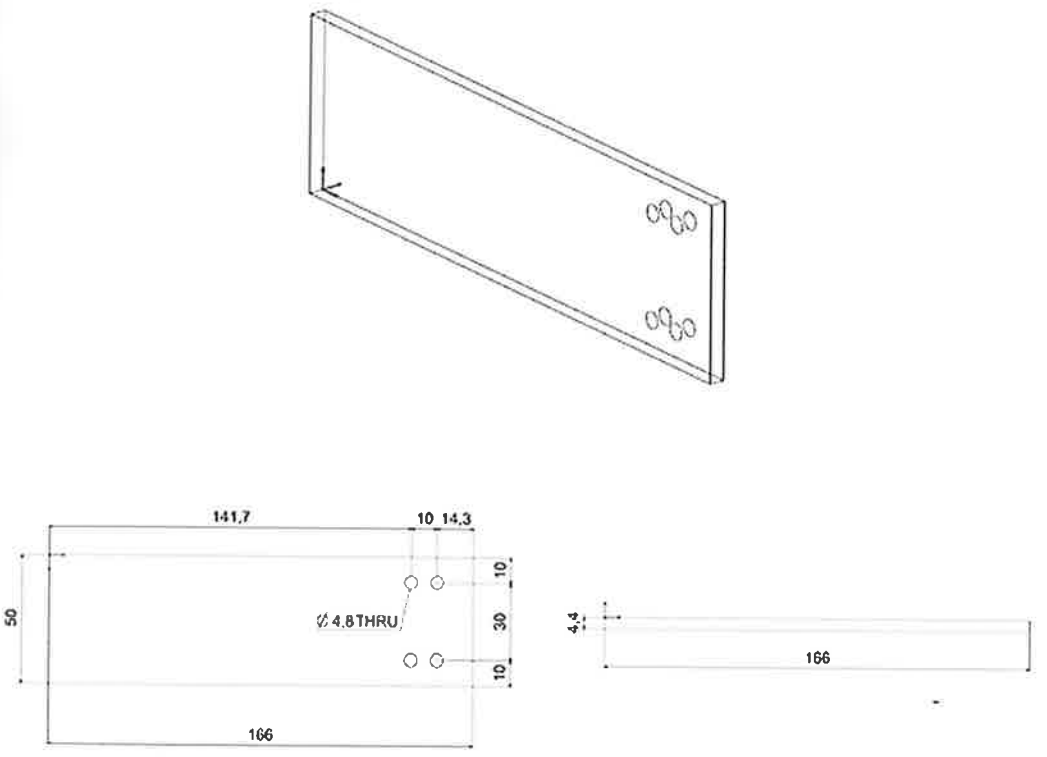


Figura 9 Perspectiva isométrica, frontal y superior de la placa de acrílico para soporte del láser.

La función de la placa de la Figura 9 es estar unida a la placa de la Figura 8 para mantener el soporte de la Figura 10 fijo ya que este último es el encargado de apuntar el láser a 45° del objeto a digitalizar. Cabe destacar que la distancia de 141.7 mm del borde izquierdo al centro de los hoyos superior e inferior izquierdos es lo que

permite que la proyección de la línea láser quede exactamente en el centro del plato giratorio.

c) Soporte del láser

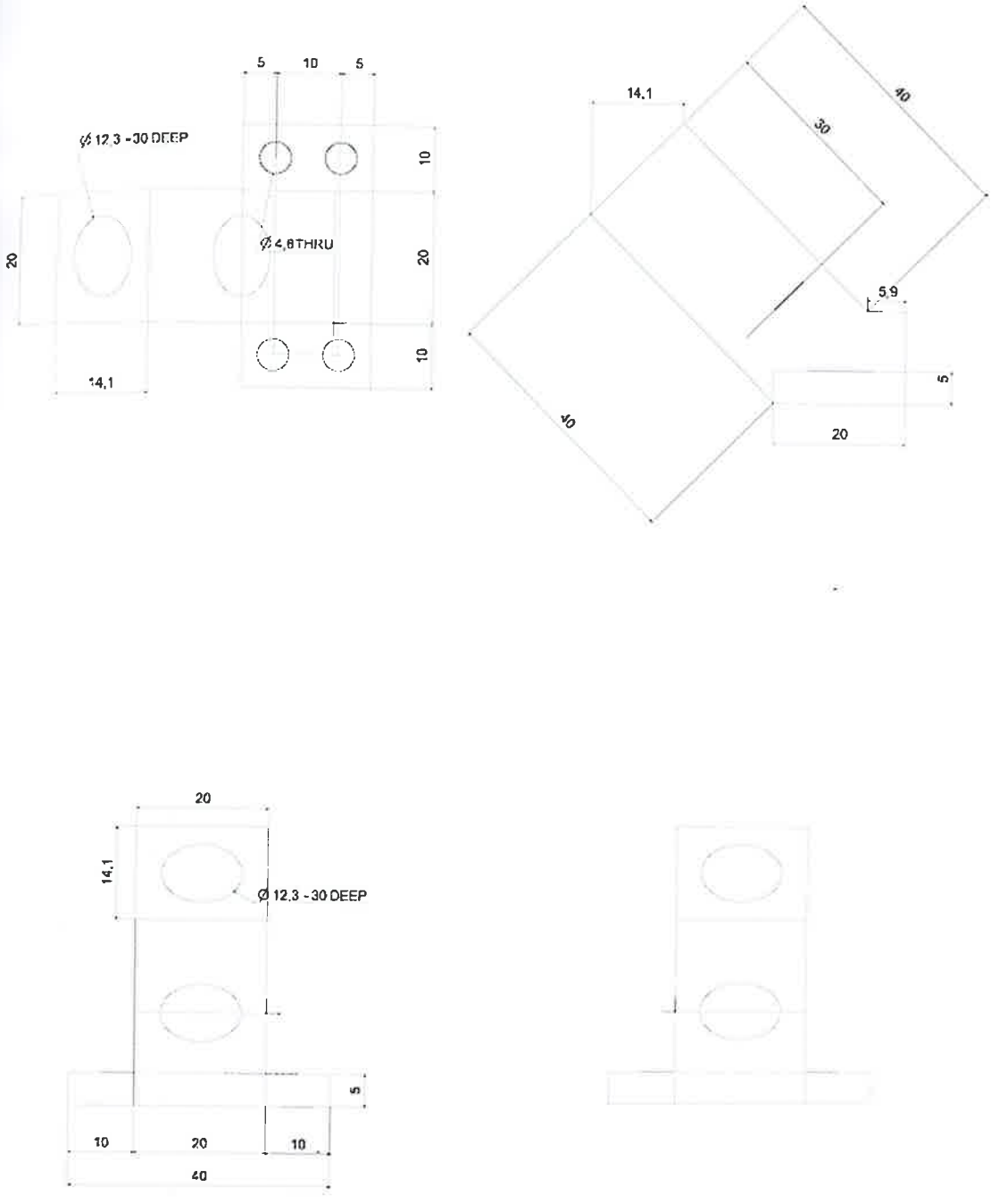


Figura 10 Perspectiva superior, frontal, derecha e izquierda del soporte para el láser.

La pieza mostrada en las Figura 10 y Figura 11 corresponde al soporte para el láser que como se mencionó anteriormente lo mantendrá apuntado con una inclinación de 45° hacia el objeto a digitalizar. Esto hace referencia a la Figura 2 de la página 13 que tiene como objetivo lograr la triangulación para los cálculos de distancia.

Figura 11 Vista frontal del soporte para el láser impreso en 3D.

d) Soporte de la cámara

La pieza mostrada en las Figura 12 y 13 se unió a la placa principal de acrílico mediante los tornillos de 6/32" y su función es mantener la cámara estable a una distancia exacta de 118 mm del centro del plato giratorio. Está diseñada para las medidas exactas de la cámara web Logitech® c170 mostrada en la Figura 5 en la página 16.

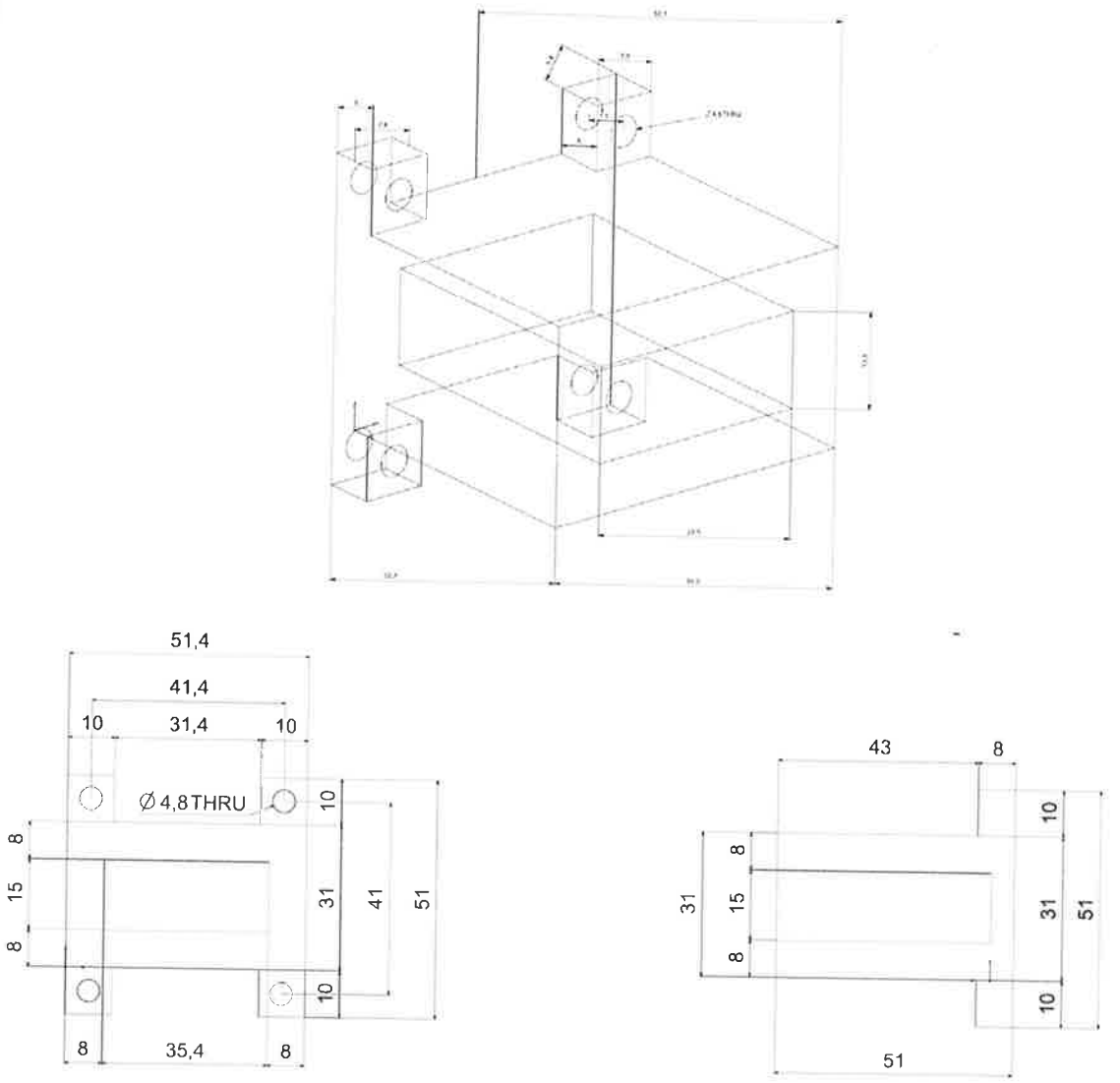


Figura 12 Perspectiva trimétrica, izquierda y trasera del soporte para la cámara.

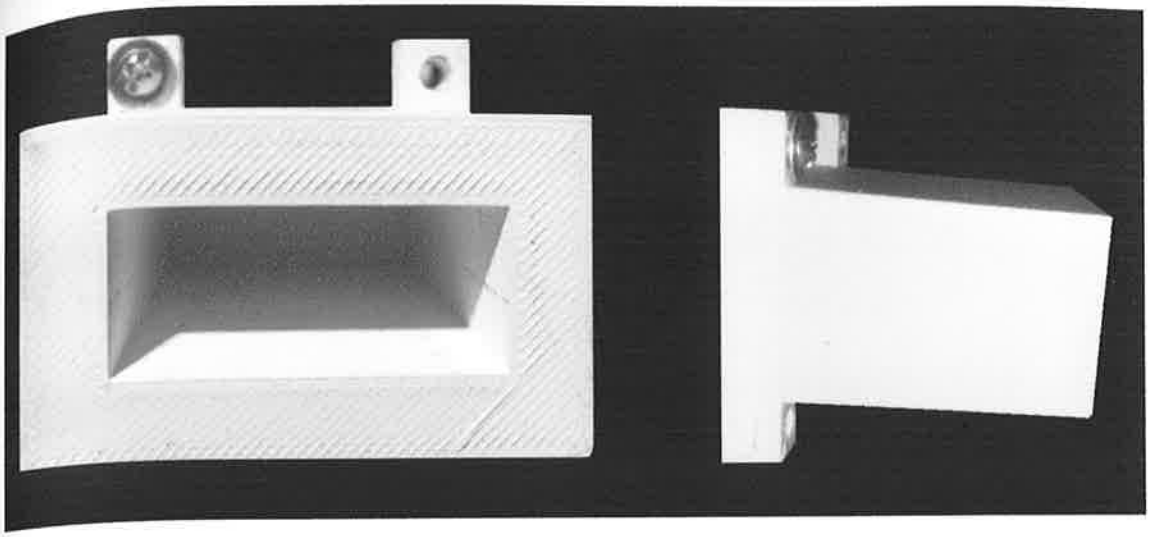


Figura 13 Vista frontal y lateral del soporte para la cámara impreso en 3D.

e) Soporte del motor

El soporte del motor está diseñado para ser impreso de igual manera en 3D; sin embargo, para la construcción del prototipo se utilizó madera. Su función es mantener estable al motor y de igual forma que la pieza anterior, está diseñada para que la distancia del centro del motor al lente de la cámara sea de 118 mm.

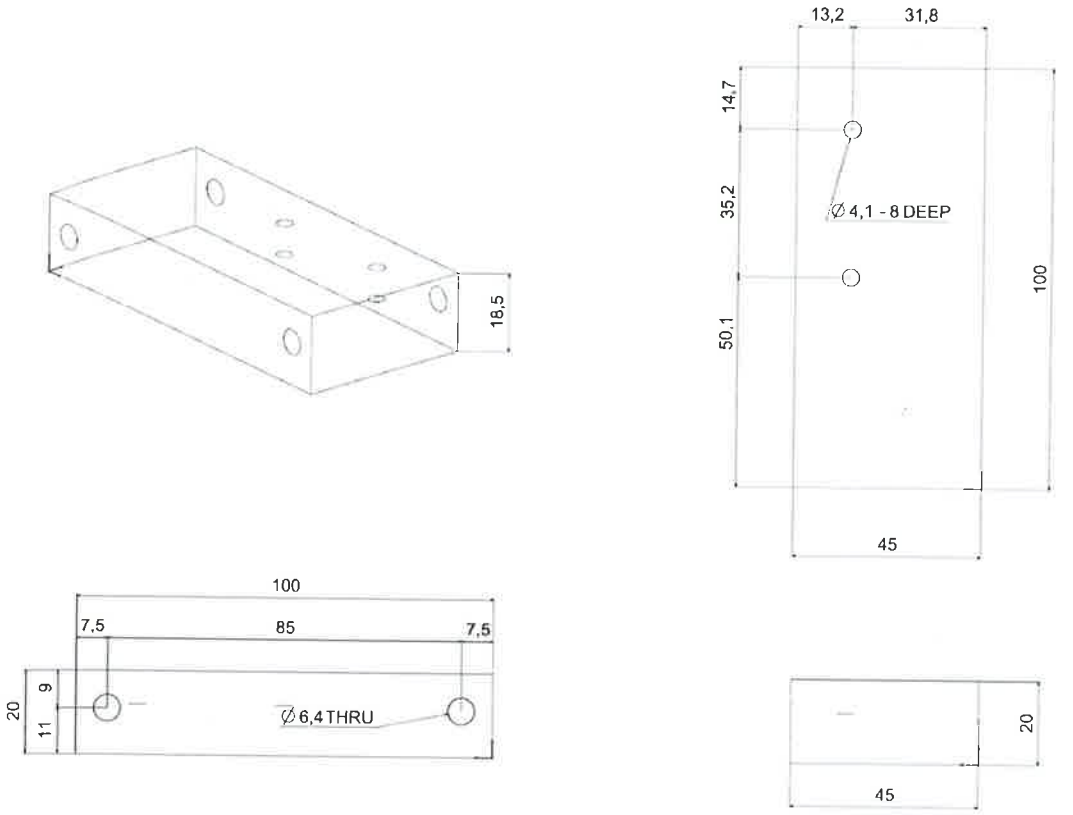


Figura 14 Perspectiva trimétrica, superior, frontal e izquierda del soporte para el motor

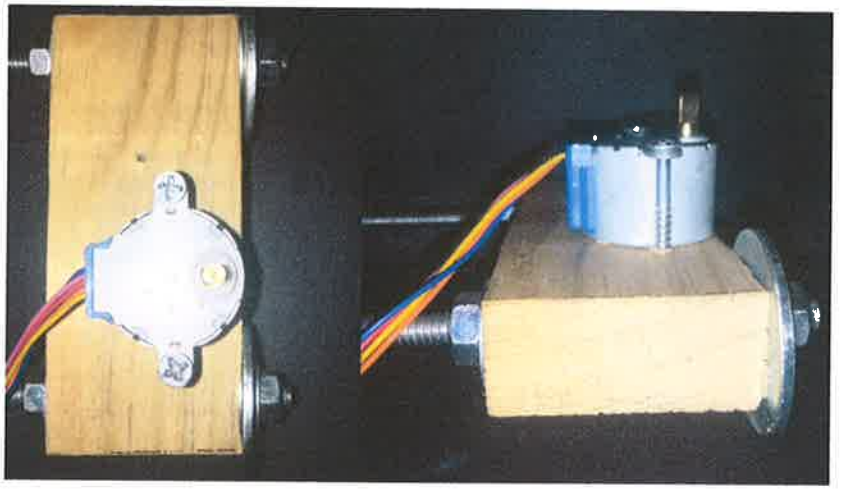


Figura 15 Vista superior y lateral del soporte para el motor

f) Plato giratorio

Esta pieza sostiene el objeto a digitalizar y permite su rotación en 360°. El color indicado es negro mate para que no refleje la luz láser. En la parte inferior se encuentra una abertura diseñada especialmente para la forma y tamaño que tiene el eje del motor a pasos 28BYJ-48. Sus planos se encuentran en la Figura 16 y la pieza impresa se muestra en la Figura 17.

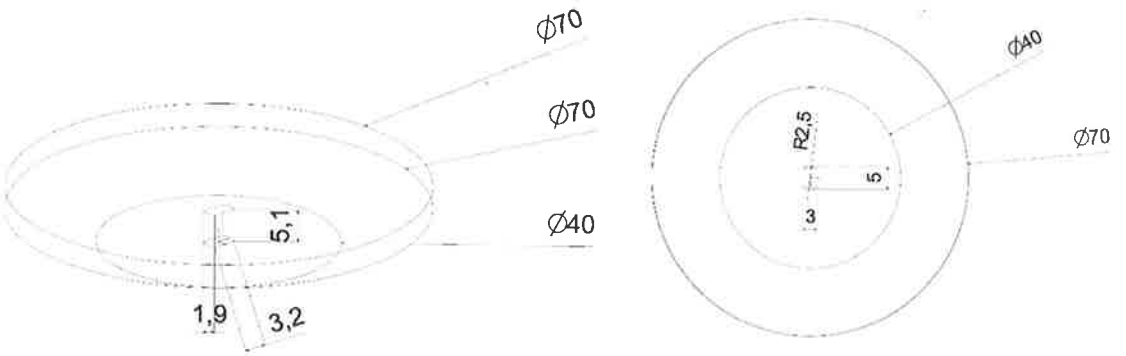


Figura 16 Perspectiva trimétrica e inferior del plato giratorio

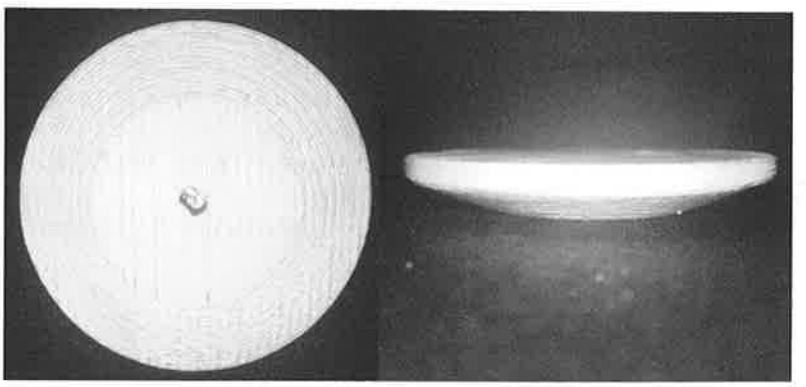


Figura 17 Vista inferior y frontal del plato giratorio impreso en 3D

VIII.II Software Utilizado

Siemens PLM Software NX (ex Unigraphics)

Todas las piezas, soportes y el plato rotatorio fueron diseñados asistidamente por computadora utilizando esta herramienta para posteriormente poder ser impresas en 3D.

Raspbian

El sistema operativo elegido para utilizarse dentro de la Raspberry Pi fue Raspbian, el cual es un sistema operativo gratuito basado en Debian y optimizado para la misma, que adicionalmente cuenta con utilidades y más de 35,000 paquetes. En él se introdujo el programa principal para controlar la captura de imágenes y el movimiento del motor, así como la aplicación gráfica para controlarlo de forma remota.

Python

El programa principal está escrito en Python y se encuentra incluido en los anexos. Una de las ventajas de haber elegido Raspbian como sistema operativo es que ya cuenta con Python 2 y 3 instalados.

Fswebcam

Esta es una aplicación simple para cámaras web que permite capturar con una sola instrucción una imagen en PNG o JPG con una amplia variedad de parámetros de configuración y que funciona de manera extraordinaria con Python.

Apache Server

Este servidor HTTP de código abierto es el encargado de permitir abrir la aplicación Web desarrollada con HTML5, JavaScript y CSS3 desde cualquier otro dispositivo conectado a la red lo que provee de una interfaz sencilla de manejo.

PHP Server

El servidor de PHP fue utilizado para recibir las peticiones generadas desde la aplicación Web y enviarlas al programa principal en Python, así como para obtener la respuesta y comunicarla al usuario.

VIII.III Ensamble de los componentes

En las Figura 18 y Figura 19 se muestra el ensamble final de los componentes enumerados a continuación, y en los Anexos 1-12 se muestran todas las perspectivas. Para la realización del prototipo final se colocaron exactamente de la misma manera y con las distancias descritas en el apartado VIII.I.III.

1. Emisor
2. Motor
3. Driver para motor a pasos
4. Computadora
5. Placa principal de acrílico
6. Placa de acrílico para soporte del láser
7. Tarjeta electrónica
8. Soporte del láser
9. Soporte de la cámara
10. Soporte del motor
11. Plato giratorio

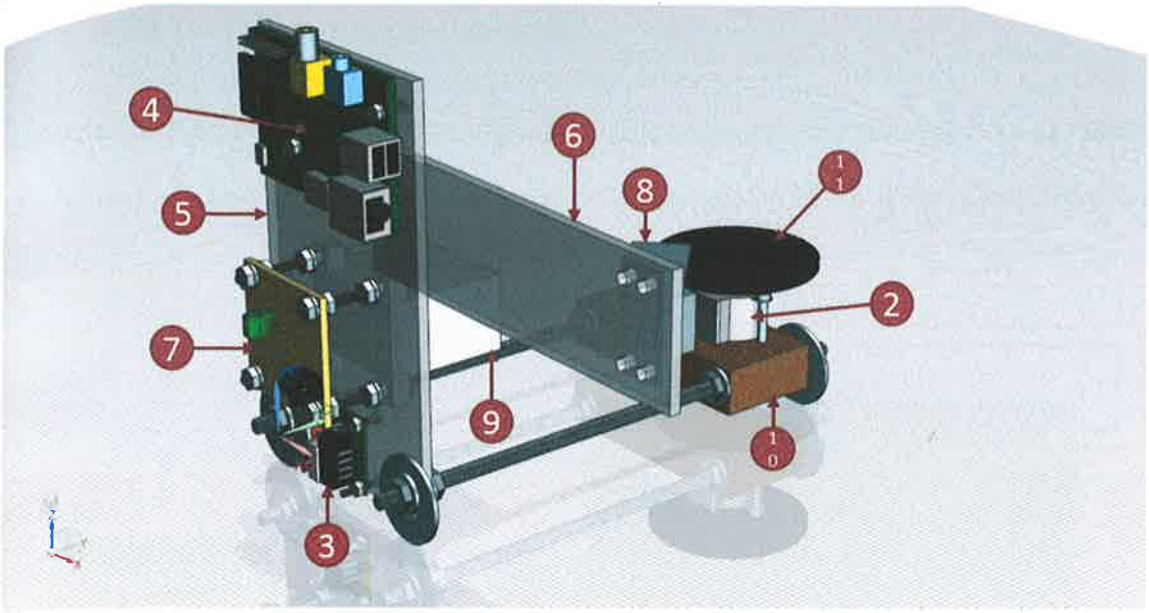


Figura 18 Ensamble de los componentes. Perspectiva Trimétrica

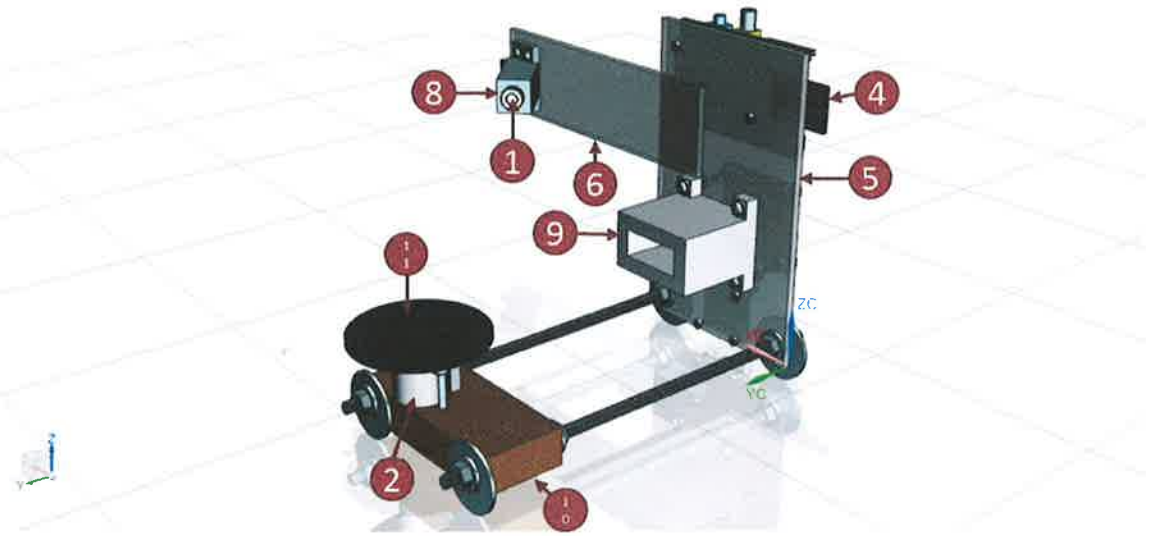


Figura 19 Ensamble de los componentes. Perspectiva Trimétrica con giro de 180° en el eje z

VIII.IV Diagrama de Conexión

A continuación en la Figura 20 se adjunta el diagrama de conexión para los circuitos eléctricos y el cableado del motor a pasos de la Raspberry Pi al driver y del driver al motor.

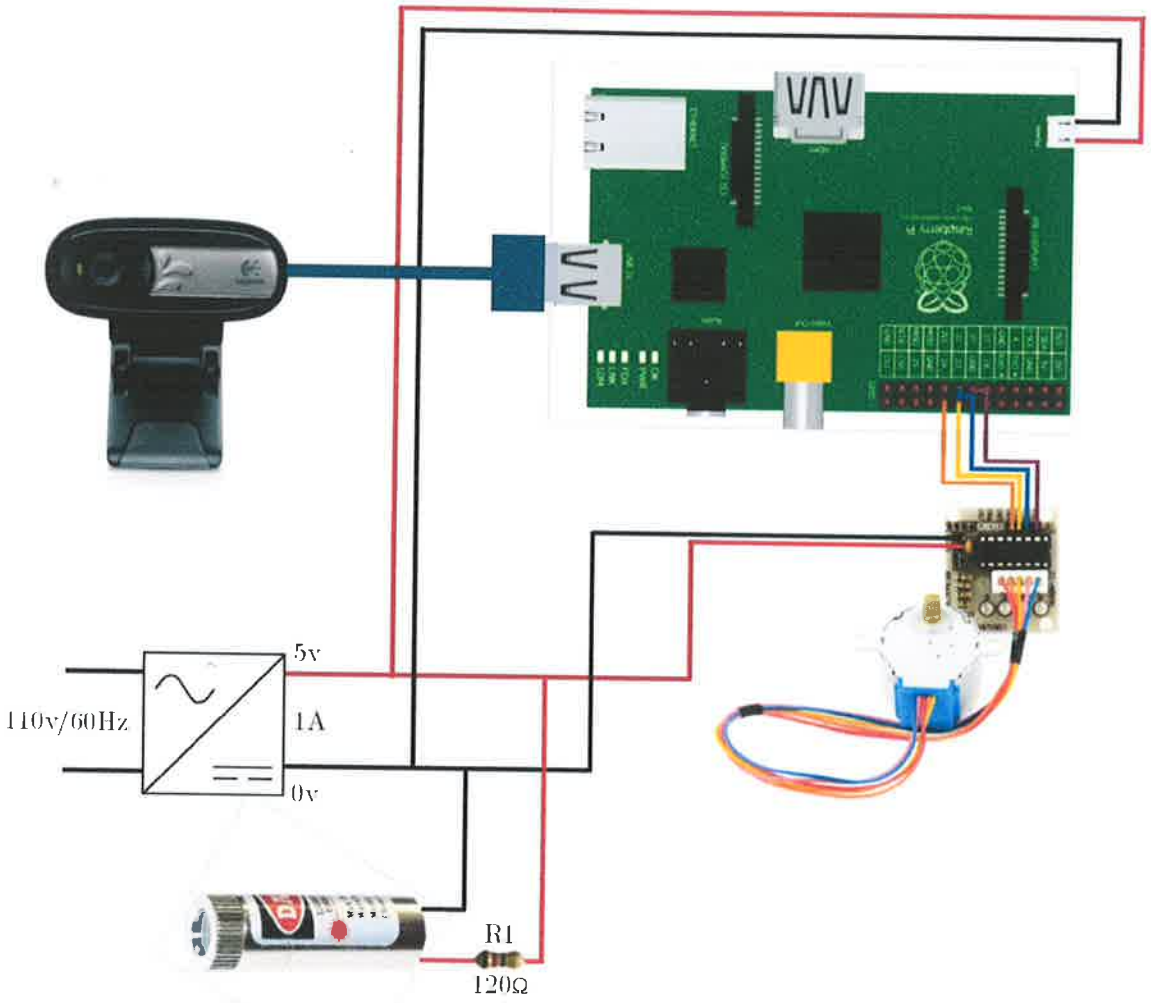


Figura 20 Diagrama de conexión

VIII.V Código

Como anteriormente se mencionó, se utilizaron distintos lenguajes de programación y varios archivos de código para todas las funcionalidades del sistema. Sin embargo, el código más relevante fue el elaborado en Python ya que a través de la información que recibe de la aplicación web (resolución, cantidad de imágenes a obtener y señal de activación) es que controla tanto el motor como la cámara y devuelve una carpeta en formato .tar.gz que contiene el conjunto de imágenes.

En las Figura 21 y Figura 22 se anexa dicho código, el cual además contiene las configuraciones iniciales que se deben hacer, así como comentarios que indican los procesos más relevantes.

```python
'''
        Antes de comenzar el programa es necesario introducir los siguientes comandos en la terminal de la Raspberry Pi

        Paso 1  sudo apt-get install fswebcam
        Paso 2  chmod +x programa.py
'''
#!/usr/bin/python
import sys
import os
import RPi.GPIO as GPIO
import time
from datetime import datetime

#Estados de los pines para controlar el motor a pasos
def estados(estado):
        if estado == 7:
                GPIO.output(13,False)
                GPIO.output(15,False)
                GPIO.output(16,False)
                GPIO.output(18,True)
        elif estado == 6:
                GPIO.output(13,False)
                GPIO.output(15,False)
                GPIO.output(16,True)
                GPIO.output(18,True)
        elif estado == 5:
                GPIO.output(13,False)
                GPIO.output(15,False)
                GPIO.output(16,True)
                GPIO.output(18,False)
        elif estado == 4:
                GPIO.output(13,False)
                GPIO.output(15,True)
                GPIO.output(16,True)
                GPIO.output(18,False)
        elif estado == 3:
                GPIO.output(13,False)
                GPIO.output(15,True)
                GPIO.output(16,False)
                GPIO.output(18,False)
        elif estado == 2:
                GPIO.output(13,True)
                GPIO.output(15,True)
                GPIO.output(16,False)
                GPIO.output(18,False)
        elif estado == 1:
                GPIO.output(13,True)
                GPIO.output(15,False)
                GPIO.output(16,False)
                GPIO.output(18,False)
        elif estado == 0:
                GPIO.output(13,True)
                GPIO.output(15,False)
                GPIO.output(16,False)
                GPIO.output(18,True)

GPIO.cleanup()
GPIO.setmode(GPIO.BOARD)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
GPIO.output(13,False)
GPIO.output(15,False)
GPIO.output(16,False)
GPIO.output(18,False)
#Se toman dos imagenes y posteriormente se borran. Con la finalidad de que el sistema este listo puesto que siempre
#las primeras dos fotografias que toma, contienen errores
os.system("fswebcam -i 0 -d /dev/video0 -r 640x480 -q --no-timestamp --no-underlay --no-banner /var/www/garbage/load1.jpg")
time.sleep(3)
os.system("fswebcam -i 0 -d /dev/video0 -r 640x480 -q --no-timestamp --no-underlay --no-banner /var/www/garbage/load2.jpg")
time.sleep(3)
os.system("sudo rm /var/www/garbage/load1.jpg")
os.system("sudo rm /var/www/garbage/load2.jpg")

delay = 0.005
count = 1
now = datetime.now()
get = sys.argv[1]
num = int(get[0])
resolucion = get[1:]
carpeta = '%s%s%s%s%s%s' % (now.hour, now.minute, now.second, now.day, now.month, now.year)
os.system("mkdir /var/www/scannerImages/"+carpeta)
instruccion = "fswebcam -i 0 -d /dev/video0 -r "+str(resolucion)+" -q --no-timestamp --no-underlay --no-banner /var/www/scannerImages/"+carpeta+"/"
extension = ".jpg"
conteoFotos = 0;

#para 64 imagenes
if num == 1:
        conteoFotos+=1
        os.system(instruccion+str(conteoFotos)+extension)
        time.sleep(1)
        for count in range(0,512):
                if (((count%2)==0) and (count<512) and (count>0)):
                        conteoFotos+=1
                        time.sleep(0.5)
                        os.system(instruccion+str(conteoFotos)+extension)
```

Figura 21 Código de programación "scanner.py" Parte 1

```
                        time.sleep(0.5)
                for x in range(0,8):
                        estados(x)
                        time.sleep(delay)
    # Escanea 128 imagenes
elif num == 2:
        conteoFotos+=1
        os.system(instruccion+str(conteoFotos)+extension)
        time.sleep(1)
        for count in range(0,512):
                if (((count%4)==0) and (count<512) and (count>0)):
                        conteoFotos+=1
                        time.sleep(0.5)
                        os.system(instruccion+str(conteoFotos)+extension)
                        time.sleep(0.5)
                for x in range(0,8):
                        estados(x)
                        time.sleep(delay)
    # Escanea 256 imagenes
elif num == 3:
        conteoFotos+=1
        os.system(instruccion+str(conteoFotos)+extension)
        time.sleep(1)
        for count in range(0,512):
                if (((count%2)==0) and (count<512) and (count>0)):
                        conteoFotos+=1
                        time.sleep(0.5)
                        os.system(instruccion+str(conteoFotos)+extension)
                        time.sleep(0.5)
                for x in range(0,8):
                        estados(x)
                        time.sleep(delay)
    # Escanea 512 imagenes
elif num == 4:
        for count in range(0,512):
                conteoFotos+=1
                time.sleep(0.5)
                os.system(instruccion+str(conteoFotos)+extension)
                time.sleep(0.5)
                for x in range(0,8):
                        estados(x)
                        time.sleep(delay)
    # Escanea 1024 imagenes
elif num == 5:
        for count in range(0,512):
                conteoFotos+=1
                time.sleep(0.5)
                os.system(instruccion+str(conteoFotos)+extension)
                time.sleep(0.5)
                for x in range(0,8):
                        if x==4:
                                conteoFotos+=1
                                time.sleep(0.5)
                                os.system(instruccion+str(conteoFotos)+extension)
                                time.sleep(0.5)
                        estados(x)
                        time.sleep(delay)
    # Escanea 2048 imagenes
elif num == 6:
        for count in range(0,512):
                conteoFotos+=1
                time.sleep(0.5)
                os.system(instruccion+str(conteoFotos)+extension)
                time.sleep(0.5)
                for x in range(0,8):
                        if x==2 or x==4 or x ==6:
                                conteoFotos+=1
                                time.sleep(0.5)
                                os.system(instruccion+str(conteoFotos)+extension)
                                time.sleep(0.5)
                        estados(x)
                        time.sleep(delay)
    # Escanea 4096 imagenes
elif num == 7:
        for count in range(0,512):
                conteoFotos+=1
                time.sleep(0.5)
                os.system(instruccion+str(conteoFotos)+extension)
                time.sleep(0.5)
                for x in range(0,7):
                        estados(x)
                        time.sleep(delay)
                        conteoFotos+=1
                        time.sleep(0.5)
                        os.system(instruccion+str(conteoFotos)+extension)
                        time.sleep(0.5)

print("scannerImages/"+str(carpeta)+".tar.gz")
os.system("tar -zcvf /var/www/scannerImages/"+str(carpeta)+".tar.gz /var/www/scannerImages/"+str(carpeta))
#Terminar Programa
GPIO.output(11,False)
GPIO.output(35,False)
GPIO.output(36,False)
GPIO.output(18,False)
GPIO.cleanup()
sys.exit()
```

Figura 22 Código de programación "scanner.py" Parte 2

X. Descripción de uso del dispositivo

Para encender el dispositivo, sólo es necesario conectarlo a una fuente de alimentación que entregue 5v a por lo menos 1.0A en corriente continua. Posteriormente se conecta la Raspberry Pi a la red de área local (LAN) y se obtiene la dirección IP. Al escribir esta dirección en cualquier navegador de internet, el servidor Apache instalado en ella, direccionará automáticamente a la pantalla que se muestra en la Figura 23.

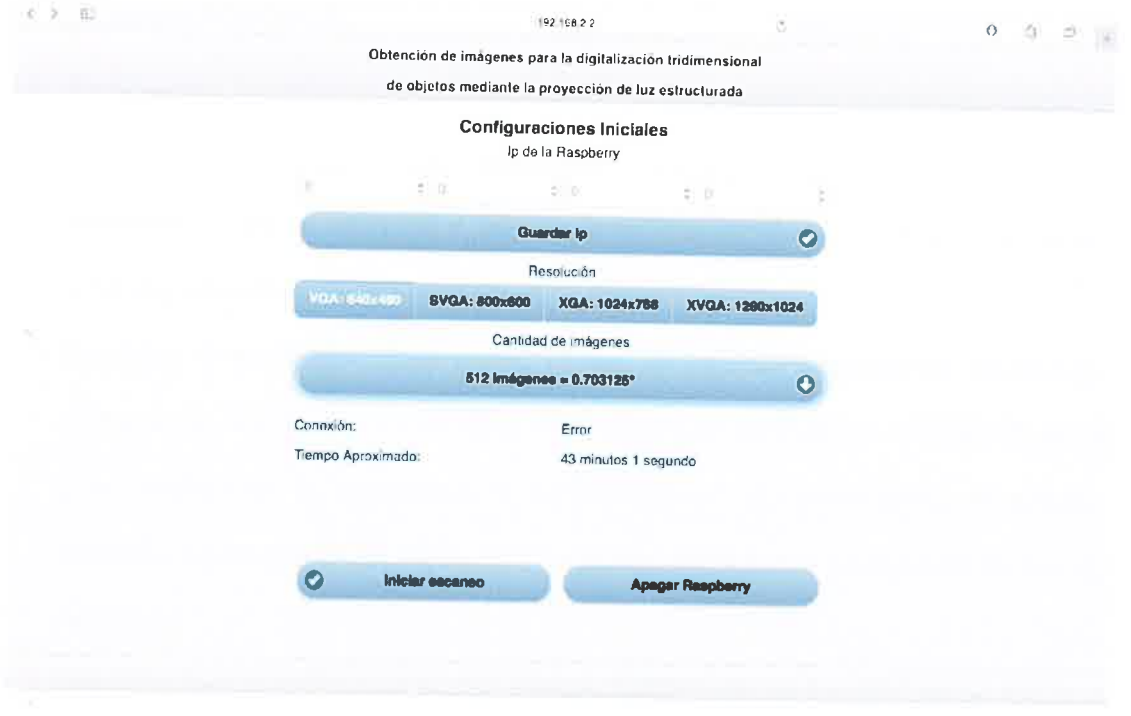


Figura 23 Aplicación web para el control del escáner.

Ya con la aplicación cargada en el navegador es necesario establecer algunos parámetros iniciales para el escaneo. A continuación se describen los puntos resaltados en la Figura 24 que refieren a todas las posibles configuraciones que se tienen para las imágenes a obtener.

1. Configuración de la IP. En estos cuatro espacios se anotan los números en formato decimal referentes a la dirección IP de la Raspberry Pi, que es la misma que se escribió en la barra de navegación.
2. Guardar IP: Una vez que ejecutado el punto anterior, se presiona este botón el cual tiene como finalidad almacenar en el sistema en forma de "cookie" la dirección IP. Sólo es necesario presionarlo la primera vez que se configura o si se requiere actualizar la misma. Si esta coincide con la dirección de la Raspberry Pi, entonces el punto 5 mostrará "OK" como en la Figura 24, de lo contrario publicará "Error", tal y como se puede apreciar en la Figura 23.
3. Resolución: En este apartado se selecciona la resolución que tendrá cada una de las imágenes una vez obtenidas. Cabe destacar que la resolución predeterminada es 640x480 y esta es la recomendada; sin embargo, es posible utilizar cualquiera de las otras tres.
4. Cantidad de imágenes: Este punto es uno de los más importantes ya que de él dependerán dos factores cruciales: el tiempo que tardará en realizar el escaneo y la calidad de la digitalización tridimensional. Al presionarlo, se abrirá la pantalla mostrada en la Figura 25 y se podrá seleccionar cualquiera de las siete opciones.

   Debido a que se utilizó un motor a pasos 28BYJ-48, en función de los pasos que gire se calculan los grados de rotación para el plato, es por eso que los valores de estos últimos no son números enteros; sin embargo, el margen de error es muy bajo. En la Tabla 2, se pueden apreciar a detalle cada una de las opciones posibles.

Número de imágenes a obtener	Grados de rotación por imagen	Tiempo aproximado para completar todo el ciclo
64	5.625	4 minutos y 45 segundos
128	2.8125	8 minutos y 50 segundos
256	1.40625	11 minutos y 50 segundos
512	0.703125	22 minutos y 42 segundos
1024	0.3515625	44 minutos
2048	0.17578125	1 hora y 18 minutos
4096	0.087890625	3 horas y 36 minutos

Tabla 2 Configuraciones posibles para cantidades de imágenes.

5. Conexión: Realiza la prueba de conexión cada que carga la página o que se Guarda una IP en el punto 2. Si no muestra "OK" significa que la conexión falló y por tanto no se puede realizar un escaneo.
6. Tiempo aproximado: Aquí muestra el tiempo que el sistema calcula que tardará el en completar todo el ciclo de obtención de imágenes en función de la cantidad seleccionada en el punto 4. Los valores posibles se mostraron en la Tabla 2.
7. Iniciar escaneo: Una vez que todos los parámetros iniciales están configurados correctamente y que se corroboró que exista comunicación entre la aplicación y el dispositivo, se puede presionar este botón para comenzar el escaneo. En la Figura 26 se muestra el cambio en la pantalla una vez iniciado el escaneo y las tres nuevas opciones que aparecen:

a. Hora de finalización: Con base a la cantidad de imágenes seleccionadas y la hora a la que se comienza el escaneo, el sistema calcula una hora aproximada a la que terminará.

b. Porcentaje realizado: Aquí se puede apreciar en tiempo real el progreso del escaneo en una escala del 0-100%.

c. Paro de emergencia: Este botón tiene la finalidad interrumpir el ciclo en caso de que hubiera algún error. Una vez presionado se requiere de aproximadamente 5 minutos para poder volver a utilizar el escáner.

8. Apagar Raspberry: Con este botón es posible mandar una señal de apagado en caso de que se requiera.

Cuando el sistema completa todo el ciclo, mostrará un mensaje como el de la Figura 27 y al presionarse "Aceptar" la pantalla cambiará tal y como se aprecia en la Figura 28. A continuación se describen los últimos tres cambios:

d. Se muestra la hora real de finalización

e. Se indica que se completó el 100%

f. Descargar Imágenes: Se habilita este botón que al presionarse, descarga un archivo comprimido con la extensión .tar.gz y que contiene el total de las imágenes en formato jpeg.

Figura 24 Aplicación web para el control del escáner. Configuraciones iniciales.

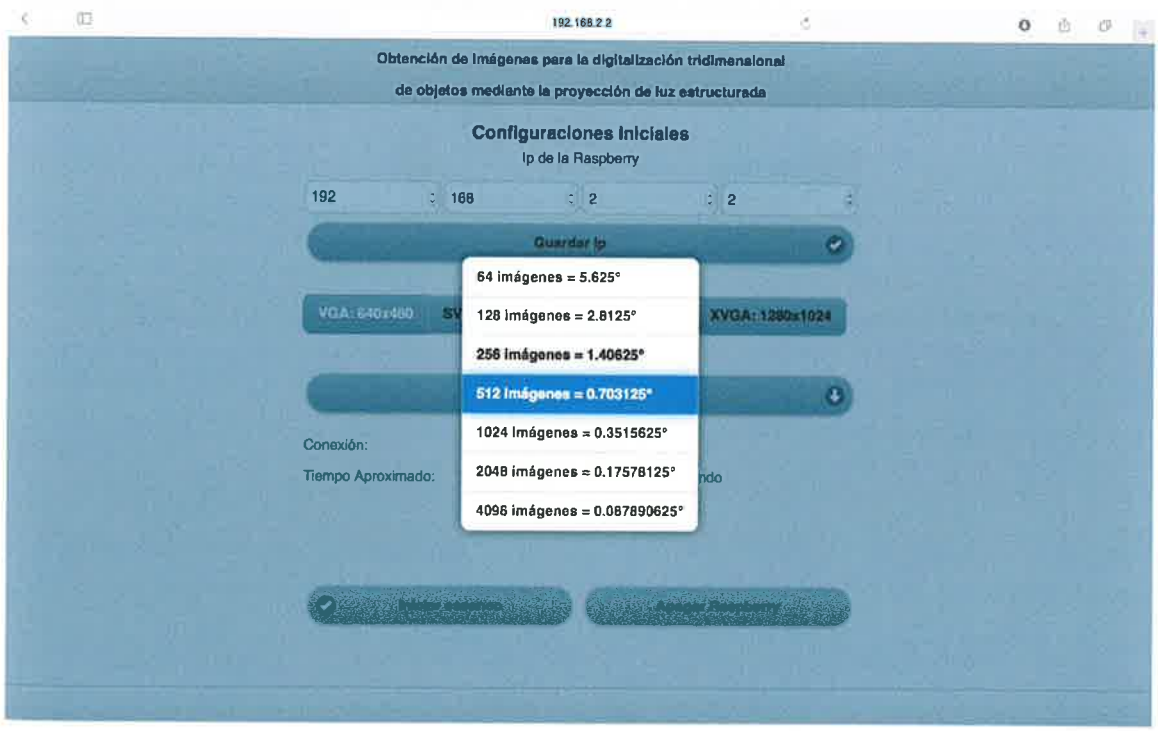


Figura 25 Aplicación web para el control del escáner. Selección de imágenes a obtener.

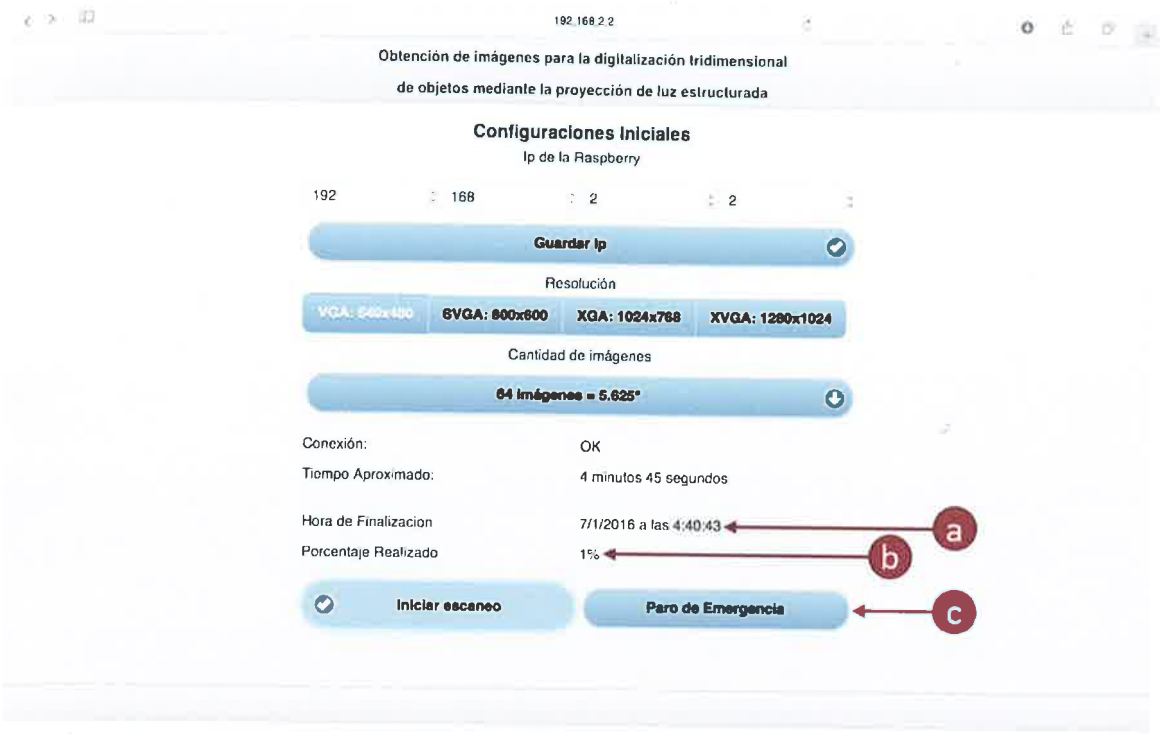


Figura 26 Aplicación web para el control del escáner. Escaneo en progreso.

Figura 27 Aplicación web para el control del escáner. Escaneo finalizado mensaje.

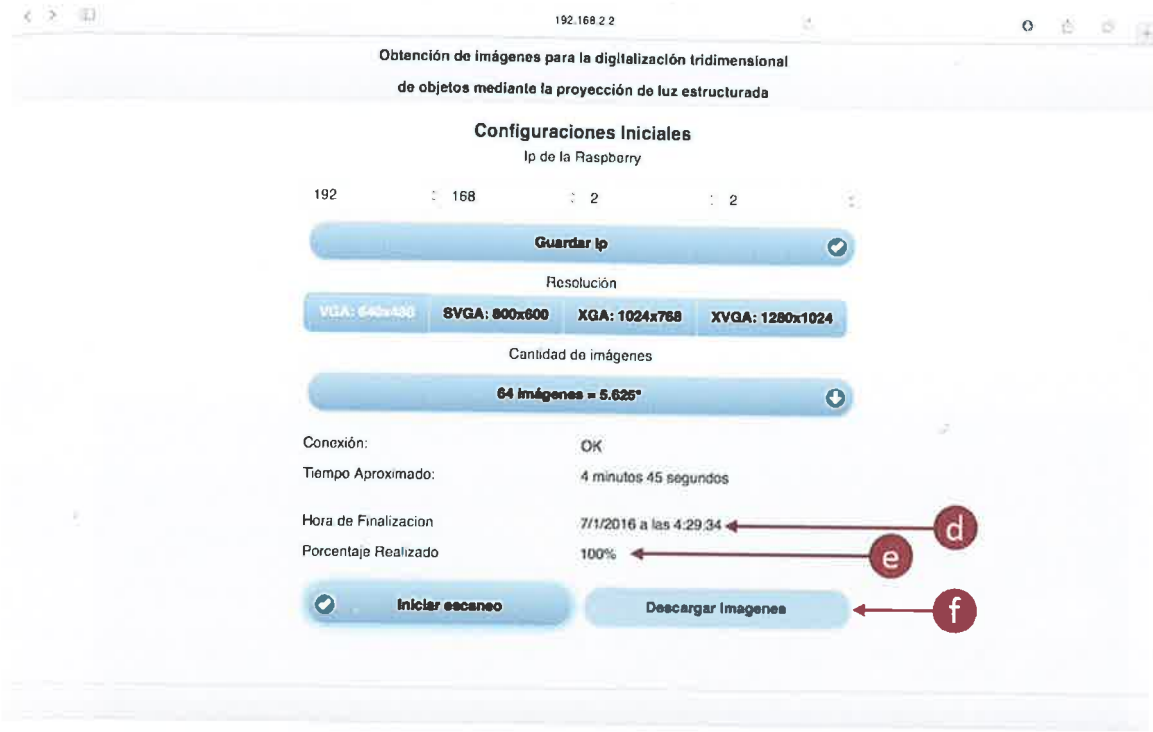


Figura 28 Aplicación web para el control del escáner. Escaneo finalizado, descarga de imágenes.

XI. Resultados

A continuación, en la Figura 29 se muestra el prototipo fabricado, con él se realizaron las pruebas para la obtención de las imágenes. Está elaborado con las mismas especificaciones que se muestran en los planos de la sección VIII.I.III y con los mismos dispositivos de la sección VIII.I.I.

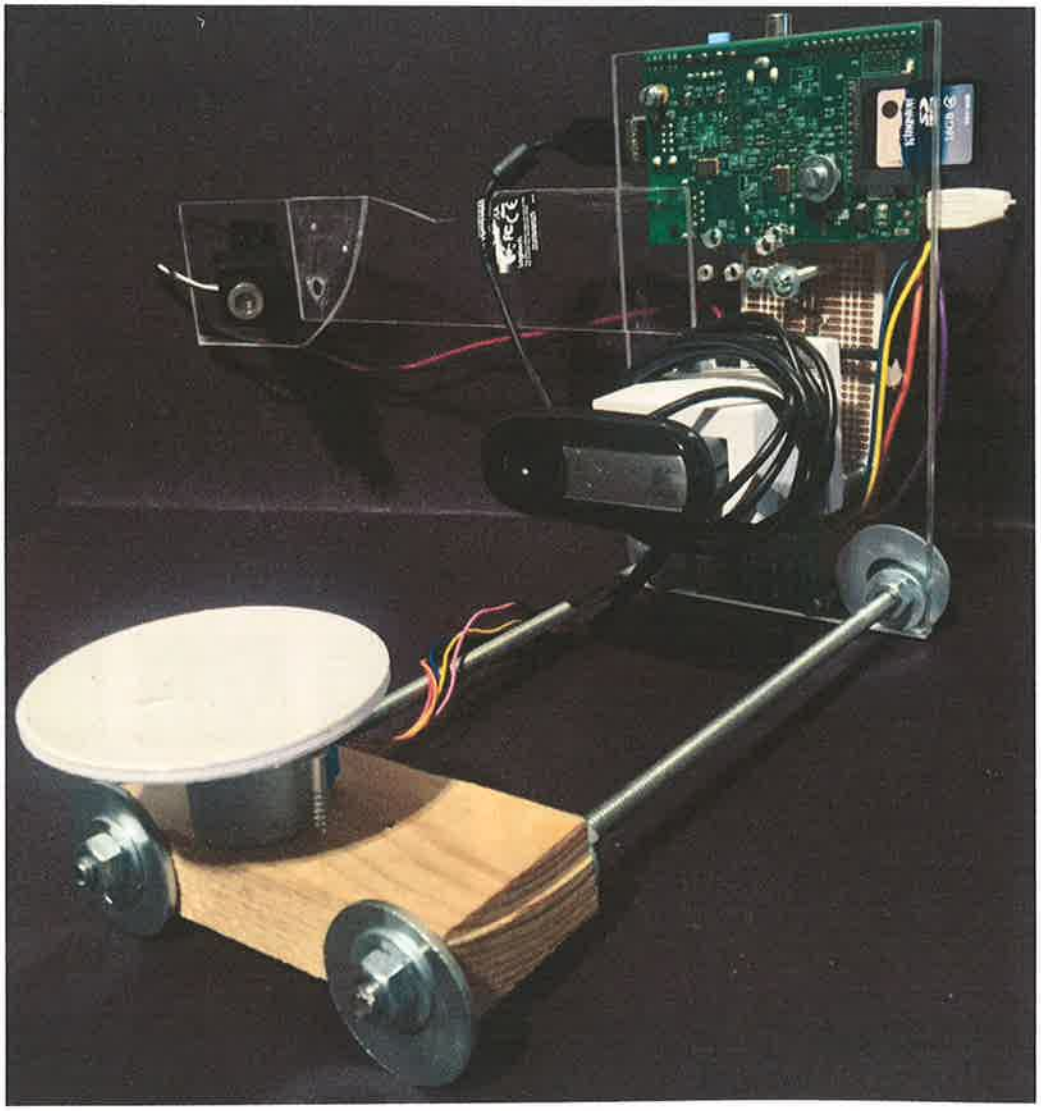


Figura 29 Perspectiva Isométrica rotada a 180° en el eje z del prototipo construido

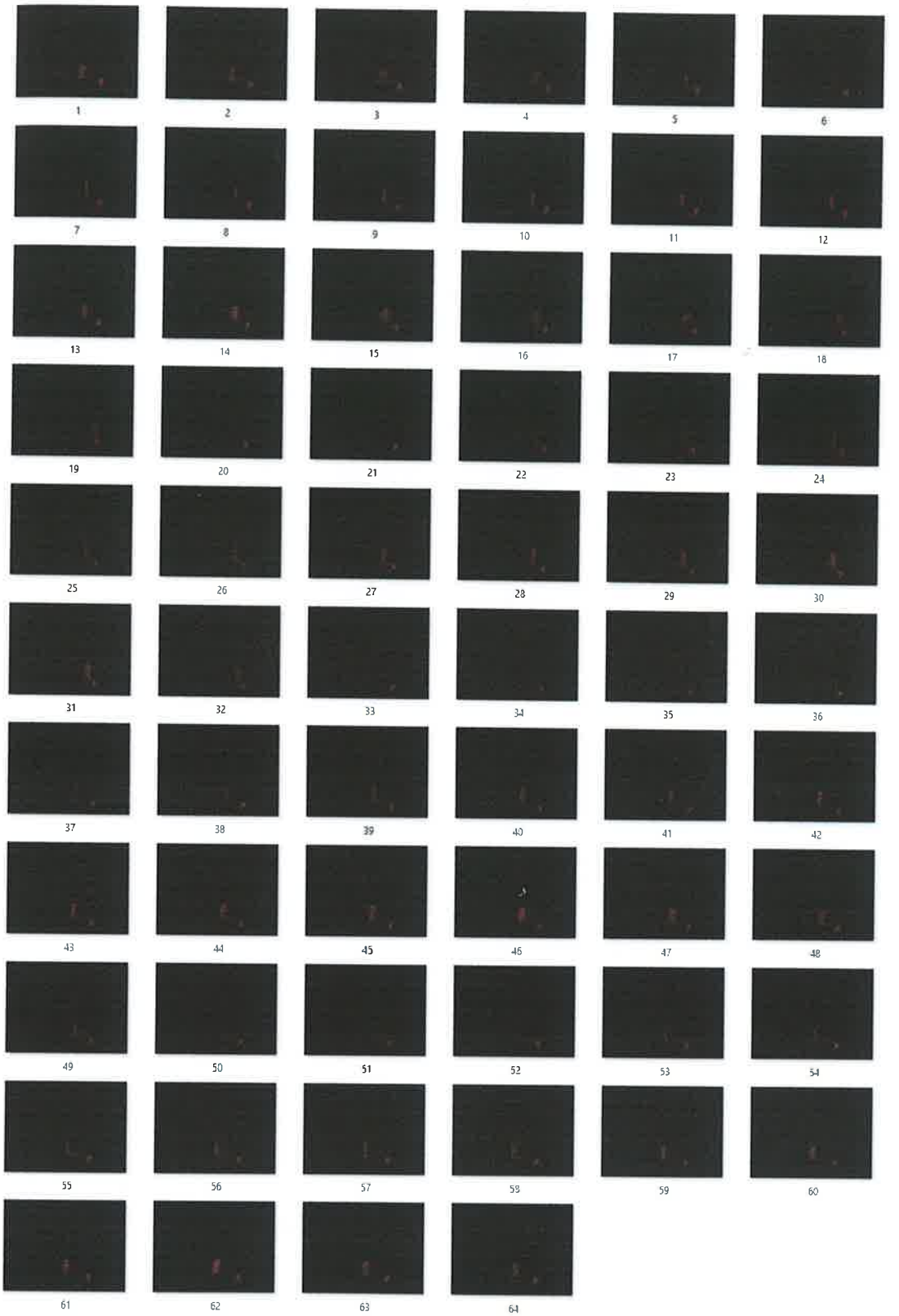


Figura 31 Conjunto de 64 imágenes tomadas con el prototipo.

Haciendo uso del prototipo de la Figura 29 y de la aplicación mostrada en el apartado X en la página 37, se realizó un escaneo del objeto referente a la Figura 30 que es un prisma cuadrado de color gris, fabricado en PLA (poliácido láctico) y que tiene 40mm de base por 10mm de altura. La resolución utilizada fue de 640x480 pixeles y se tomaron 64 imágenes. En la Figura 31 se puede apreciar el resultado de este escaneo.

Figura 30 Objeto escaneado.

XII. Conclusiones

En el presente trabajo se presentaron las herramientas para la construcción de un dispositivo de adquisición de imágenes, que utiliza únicamente tecnología de consumo (cámara, computadora, láser comercial) y software libre. Una vez que se complemente con un algoritmo de reconstrucción tridimensional que haga uso de las fotografías tomadas por él mismo, proporcionará un mecanismo eficiente y de bajo costo para el escaneo de objetos en tres dimensiones.

Aunque fue un buen comienzo la fabricación de un dispositivo que provea a los ingenieros enfocados a la visión por computadora de una herramienta para obtener imágenes con un patrón de luz estructurada y con amplias opciones de configuración, existen varios problemas detectados.

El primero de ellos es la condición de la luminosidad, ya que para que obtener imágenes útiles, se tiene que trabajar en total oscuridad para que se pueda detectar bien la proyección de la línea láser. Aunado a esto, los objetos a digitalizar no pueden de ninguna manera reflejar la luz, puesto que de hacerlo el patrón se distorsionará y se obtendrán lecturas erróneas.

Por último, si bien las fotografías que el sistema actual captura serán útiles en reconstrucciones tridimensionales, es indispensable la utilización de otro láser linear, con una longitud de onda diferente que permita disminuir el ancho de la línea proyectada, ya que la resolución de los modelos digitalizados está en función de ella.

Si se logran corregir estos problemas, diseñar un buen algoritmo de reconstrucción y calibrar correctamente el dispositivo, sí es posible obtener datos certeros acerca de la forma geométrica de objetos utilizando una arquitectura abierta con dispositivos comerciales; sin embargo, la fiabilidad y eficiencia aún se encuentra lejos de los escáneres profesionales que se encuentran en el mercado.

XIII. Referencias

[1] B. Girod, G. Greiner y H. Niemann, Principles of 3D Image Analysis and Synthesis, Stanford: Springer Science & Business Media, 2013.

[2] F. Bernardini y H. Rushmeier, «The 3D Model Acquisition Pipeline,» *Computer Graphics,* vol. 21, nº 2, pp. 149-172, 2002.

[3] C. T. Zamora, D. C. Muñoz, J. C. P. Ortega, S. T. Arriaga, E. G. Hurtado y S. L. C. Magdaleno, «Sistema para digitalización y modelado 3D de objetos, mediante proyección laser utilizando hardware de arquitectura abierta,» de *IX Congreso Internacional sobre Innovación y Desarrollo Tecnológico*, Cuernavaca, México, 2011.

[4] U. Yilmaz, A. Y. Mulayim y V. Atalay, «An Image-Based Inexpensive 3d Scanner,» *International Journal of Image and Graphics,* 2002.

[5] R. A. Jarvis, «A perspective on range finding techniques for computer vision,» *IEEE Trans. Pattern Anal. Mach. Intell,* pp. 122-139, 1983.

[6] P. J. Besl, «Active, optical range imaging sensors,» *Mach. Vision Appl,* pp. 127-152, 1988.

[7] G. Bickel, G. Hausler y M. Maul, «Triangulation with expanded range of depth,» *Opt. Eng,* p. 975–977 , 1985.

[8] W. Dremel, G. Heusler y M. Maul, «Triangulation with a large dynamical range,» de *Proc. SPIE*, 1986.

[9] J. Y. Wang, «Imaging laser radar—an overview,» de *Proc. 9th Intl. Conf. Laser'86*, 1986.

[10] M. Rioux, F. Blais, J. A. Beraldin y P. Boulanger, «Range imaging sensors development at NRC laboratories,» de *Proc. IEEE Workshop Interpretation 3D Scenes*, 1989.

[11] F. Blais, M. Rioux y J.-A. Beraldin, «Practical considerations for a design of a high precision 3-D laser scanner system,» de *Proc. SPIE*, 1988.

[12] M. Idesawa, «High-precision image position sensing methods suitable for 3-D measurement,» *Opt. Lasers Eng,* pp. 3-4, 1989.

[13] H. Steinbichler, «Method and apparatus for ascertaining the absolute coordinates of an object». Estados Unidos Patente 5,289,264 , 1994.

[14] J.-A. Beraldin, S. F. El-Hakim y F. Blais, «Performance evaluation of three active vision systems built at the National Research Council of Canada,» de *Proceedings of the Optical 3D Measurement Techniques III*, Vienna, 1995.

[15] G. Blais y M. D. Levine, «Registering Multiview range data to create 3D computer objects,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* p. 820–824, 1995.

[16] A. Hilton, A. Stoddart, J. Illingworth y T. Windeatt, «Reliable surface reconstruction from multiple range images,» de *Fourth European Conference on ComputerVision*, 1996.

[17] M.-E. Algorri y F. Schmitt, «Surface reconstruction from unstructured 3D data,» *Computer Graphics Forum,* p. 47–60, 1996.

[18] J. Neugebauer y K. Klein, «Adaptive triangulation of objects reconstructed from multiple range images,» *IEEE Visualization '97, Late Breaking Hot Topics,* 1997.

[19] P. J. Neugebauer, «Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images,» *International Journal of Shape Modeling,* p. 71–90, 1997.

[20] M. Wheeler, Y. Sato y K. Ikeuchi, «Consensus surfaces for modeling 3D objects from multiple range images,» de *Sixth International Conference on ComputerVision, IEEE*, 1998.

[21] R. T. Whitaker, «A level-set approach to 3D reconstruction from range data,» *International Journal of Computer Vision,* p. 203–231, 1998.

[22] M. Reed y P. Allen, «3D modeling from range imagery: an incremental method with a planning component,» *Image and Vision Computing,* p. 99– 111, 1999.

[23] P. Neugebauer y K. Klein, «Texturing 3D models of real world objects from multiple unregistered photographics views,» de *Computer Graphics Forum*, 1999.

[24] R. Y. Tsai, «An efficient and accurate camera calibration technique for 3d machine vision,» de *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Florida, USA , 1996.

[25] W. Niem y J. Wingbermühle, «Automatic reconstruction of 3D objects using a mobile monoscopic camera,» de *Proc. International Conference on Recent Advances in 3D Imaging and Modeling*, 1997.

[26] Z. Zhang, «A flexible new technique for camera calibration,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 1330-1334, 2000.

[27] J. Isern González, «Estudio experimental de métodos de calibración y autocalibración de cámaras,» *UPGC,* 2003.

[28] D. A. Pizarro, P. Campos y C. L. Tozzi, «Comparación de técnicas de calibración de cámaras digitales,» *Rev. Fac. Ing., Univ. Tarapacá,* 2005.

[29] W. B. Seales y O. D. Faugeras, «Building 3-dimensional object models from image sequences,» *Computer Vision and Image Understanding* , p. 308–324, 1995.

[30] H. Rushmeier, F. Bernardini, J. Mittleman y G. Taubin, «Acquiring input for rendering at appropriate level of detail: digitizing a Piet`a,» de *Proceedings of the 9th Eurographics Workshop on Rendering*, Vienna, Austria, 1998.

[31] N. Amenta, S. Choi, T. K. Dey y N. Leekha, «A simple algorithm for surface reconstruction,» de *Proceedings of 16th ACM Symposium on Computational Geometry*, 2000.

[32] E. Trucco y A. Verri, «Introductory to techniques for 3-D computer vision,» *Prentice Hall,* 1998.

[33] U. Yılmaz, A. Y. Mülayim y V. Atalay, «Reconstruction of three dimensional models from real images,» de *Proc. International Symposium on 3D Data Processing Visualization and Transmission*, 2002.

[34] M. Polleyfeys, R. Koch, M. Vergauwen y L. V.Gool., «Hand-held acquisition of 3D models with a video camera,» de *Proceeding of the 2nd International Conference on 3D Digital Imaging and Modeling*, Ottawa, Canada, 1999.

[35] F. Schmitt y Y. Yemez, «3D color object reconstruction from 2D image sequences,» de *Proc. International Conference on Image Processing*, 1999.

[36] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro y W. Stuetzle, «View-based rendering: visualizing real objects from scanned range and color data,» de *Proceedings of the 8th Eurographics Workshop on Rendering*, St Etienne, France, 1997.

[37] F. Chen, G. M. Brown y M. Song, «Overview of three-dimensional shape measurement using optical methods,» *Opt. Eng,* pp. 10-22, 2000.

[38] J. C. A. González y H. I. M. Castillo, «Análisis comparativo de algoritmos de reconstrucción 3D basados en visión para la obtención de trayectorias,» de *XVI Congreso Internacional Anual De La SOMIM*, Monterrey, Nuevo León, México, 2010.

[39] B. Curless y M. Levoy, «Better Optical Triangulation through Spacetime Analysis,» 1995.

[40] L. Alboul, G. Kloosterman, C. Traas y R. v. Damme, «Best data-dependent triangulations,» Twente, 1999.

[41] Y. Matsumoto, H. Terasaki, K. Sugimoto y T. Arakawa, «A portable threedimensional digitizer,» de *Proc. International Conference on Recent Advances in 3D Digital Imaging and Modeling*, 1997.

[42] D. Caspi, N. Kiryati y J. Shamir, «Range imaging with adaptive color structured light,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 470-480, 1998.

[43] E. Horn y N. Kiryati, «Toward optimal struc-tured light patterns,» *Image and Vision Computing* , pp. 87-97, 1999.

[44] M. Levoy, «M. Levoy The digital Michelangelo project: 3D scanning of large statues,» de *Proceedings of SIGGRAPH 00, Computer Graphics Proceedings, Annual Conference Series*, 2000.

[45] A. Y. Mülayim, Y. Yemez, F. Schmitt y V. Atalay, «Rotation axis extraction of a turn table viewed by a fixed camera,» de *Proc. Vision Modeling and Visualization*, 1999.

[46] C. Rocchini, P. Cignoni, C. Montani, P. Pingi y R. Scopigno, «A low cost 3D scanner based on structured light,» *Istituto di Scienza e Tecnologie dell'Informazione (ISTI) Consiglio Nazionale delle Ricerche, C.N.R,* 2001.

[47] S. Winkelbach, S. Molkenstruck y F. M. Wahl, «Low-Cost Laser Range Scanner and Fast Surface Registration Approach,» *Springer Berlin Heidelberg,* 2006.

[48] H. C. Reyes, F. M. Ibarra y J. C. P. Ortega, «Caracterización de un sistema de reconstrucción 3D de bajo costo, utilizando proyección de línea láser,» de *6º Congreso Internacional en Innovación y Desarrollo Tecnológico*, Cuernavaca, México, 2008.

[49] B. Curless y S. Seitz, «3D Photography,» de *ACM Siggraph '00 Course Notes*, 2000.

[50] J. Arentsen, «Luz coherente. Rayos laser,» de *Luz, egos y universos*, Santiago, Andrés Bello, 1985, p. 257.

XIV. Anexos

Anexo 1

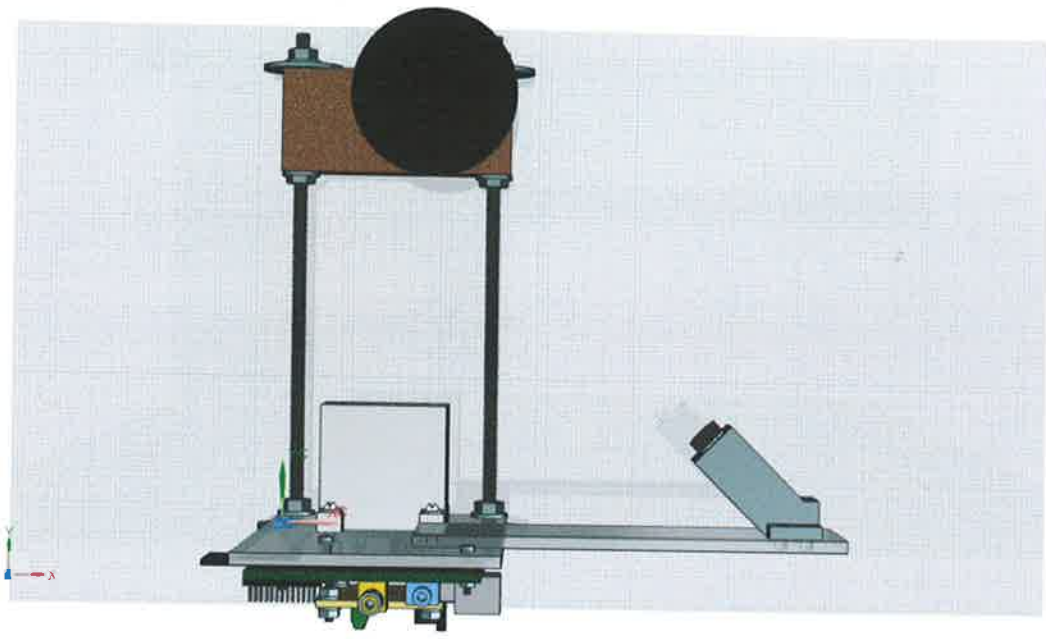


Figura 32 Perspectiva superior del modelo asistido por computadora del prototipo.

Anexo 2

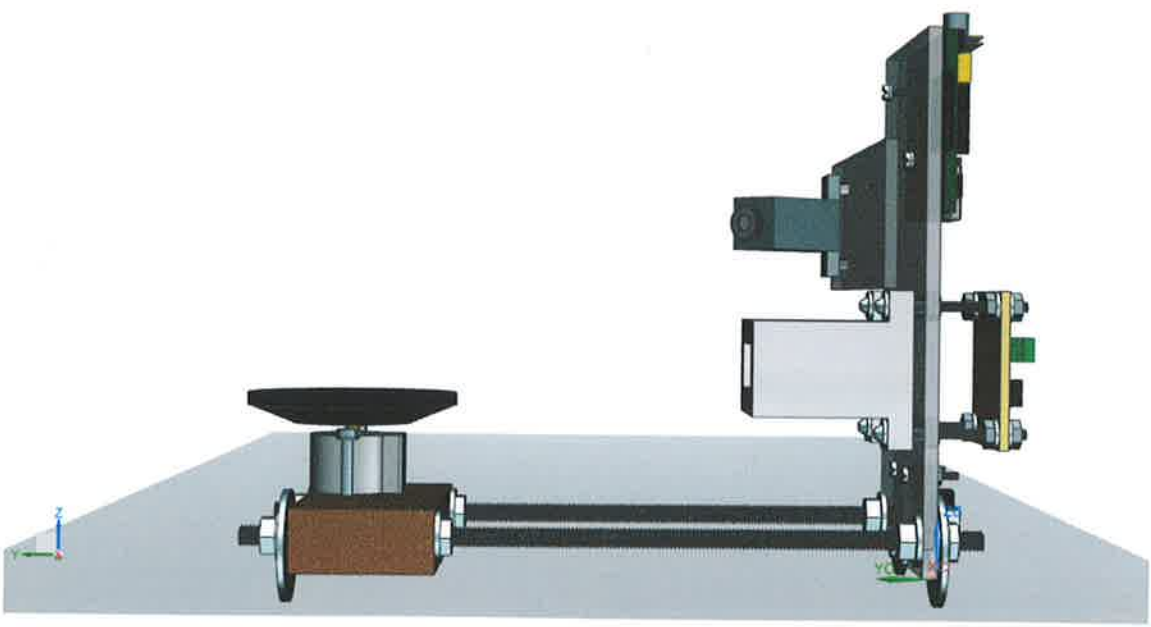


Figura 33 Perspectiva izquierda del modelo asistido por computadora del prototipo.

Anexo 3

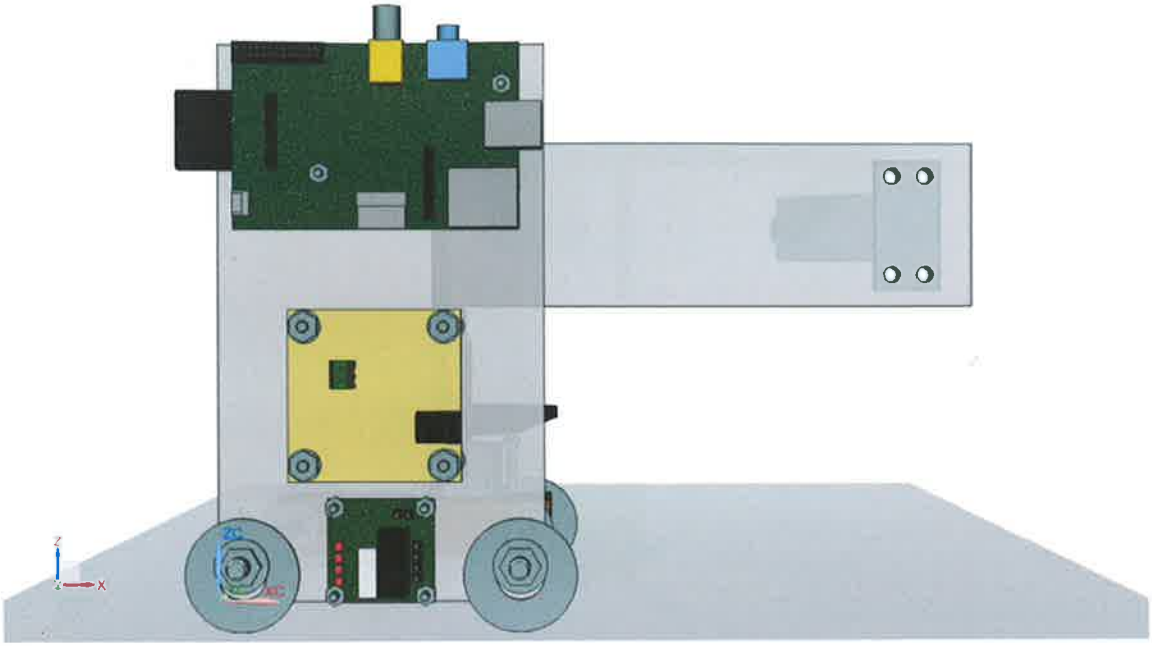


Figura 34 Perspectiva frontal del modelo asistido por computadora del prototipo.

Anexo 4

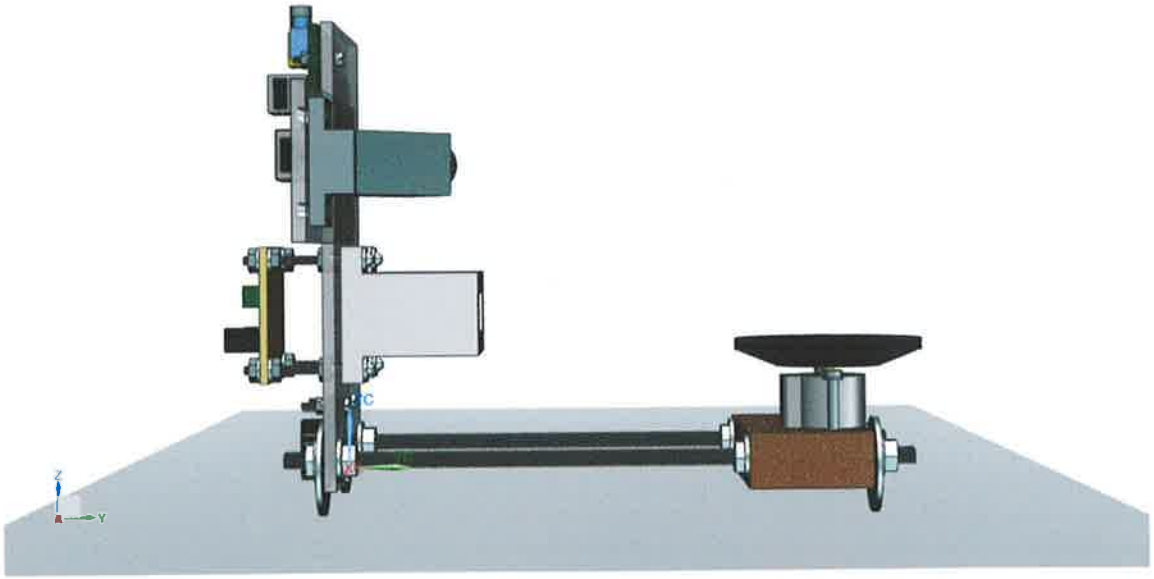


Figura 35 Perspectiva derecha del modelo asistido por computadora del prototipo.

Anexo 5

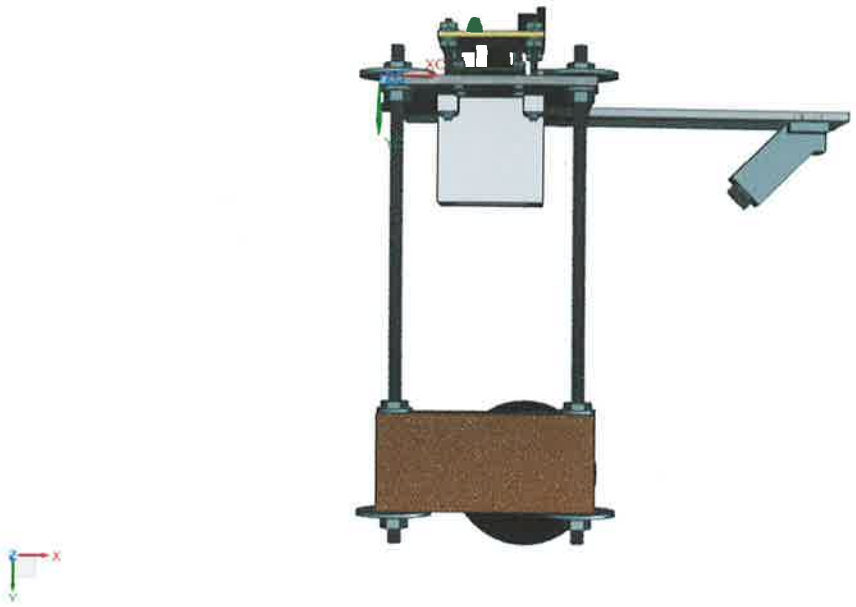


Figura 36 Perspectiva inferior del modelo asistido por computadora del prototipo.

Anexo 6

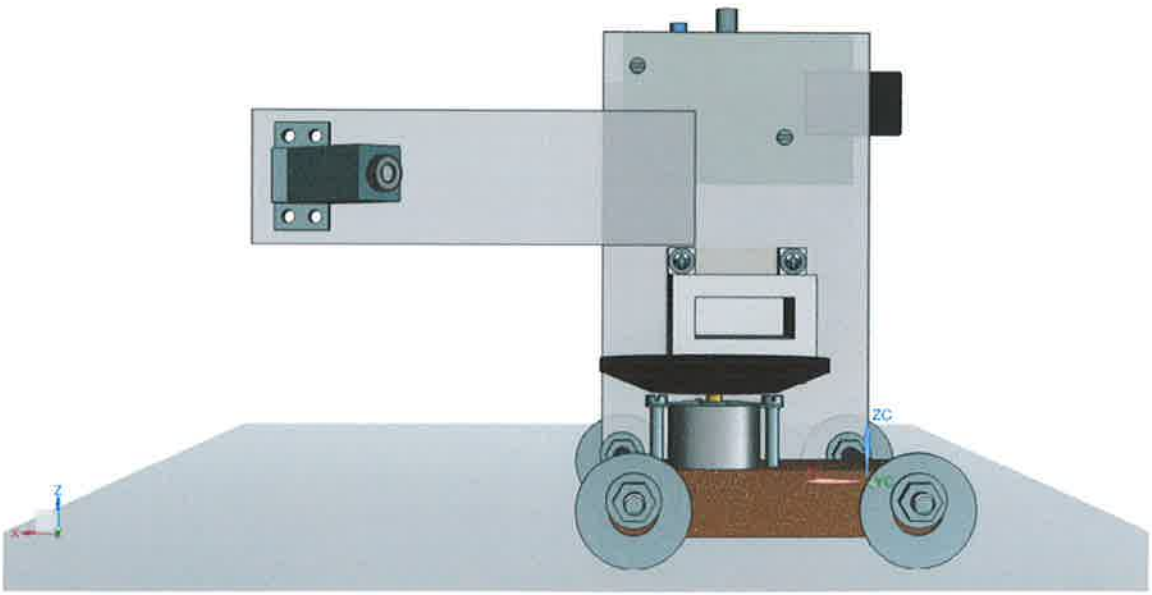


Figura 37 Perspectiva trasera del modelo asistido por computadora del prototipo.

Anexo 7

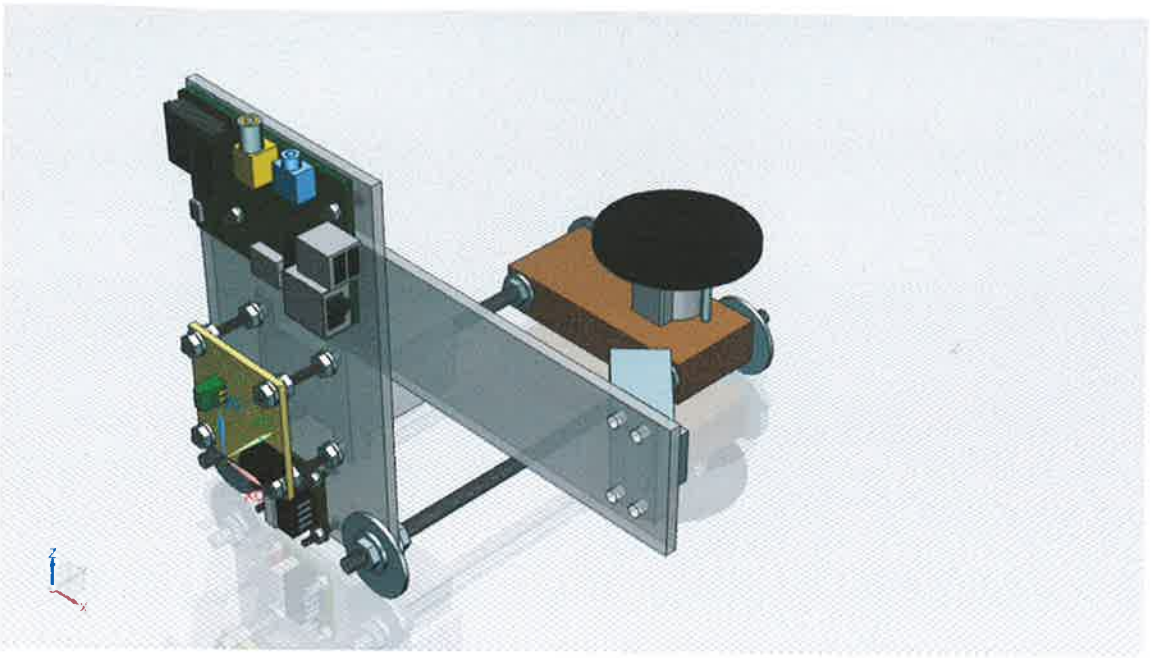


Figura 38 Perspectiva isométrica del modelo asistido por computadora del prototipo.

Anexo 8

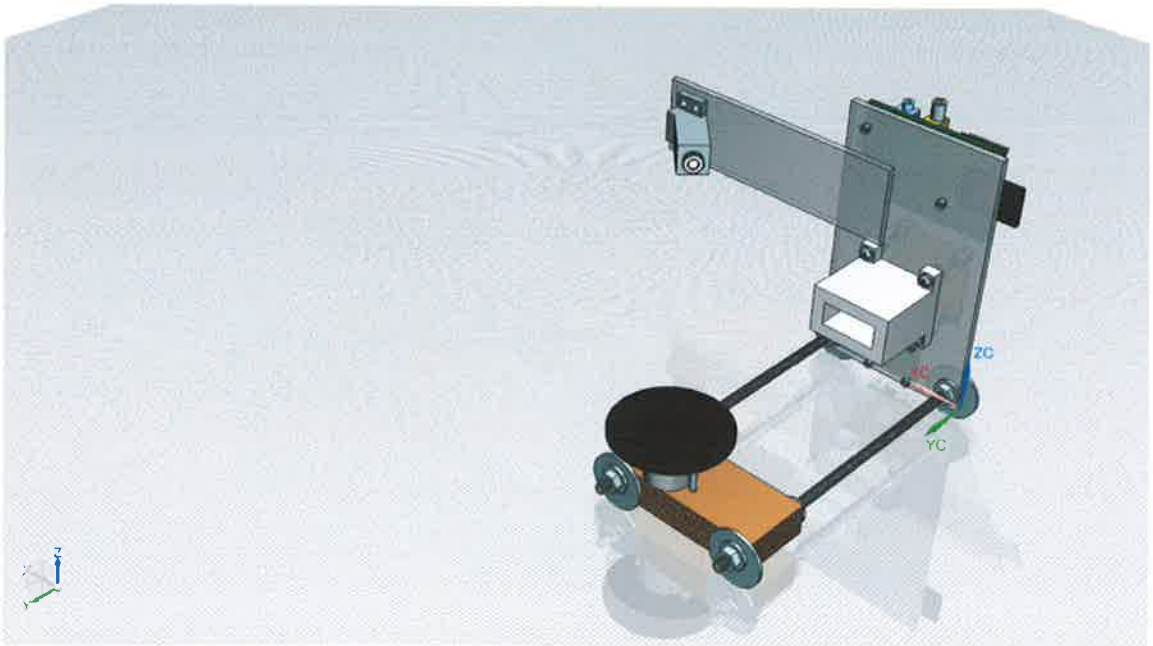


Figura 39 Perspectiva isométrica rotada 180° del modelo asistido por computadora del prototipo.

Anexo 9

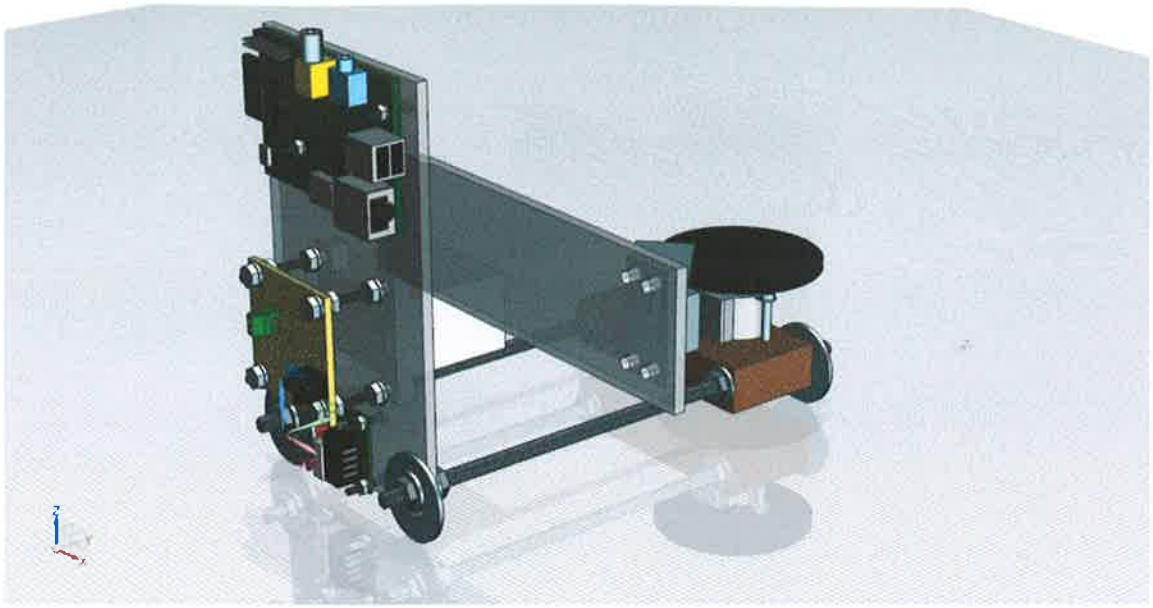


Figura 40 Perspectiva trimétrica del modelo asistido por computadora del prototipo.

Anexo 10

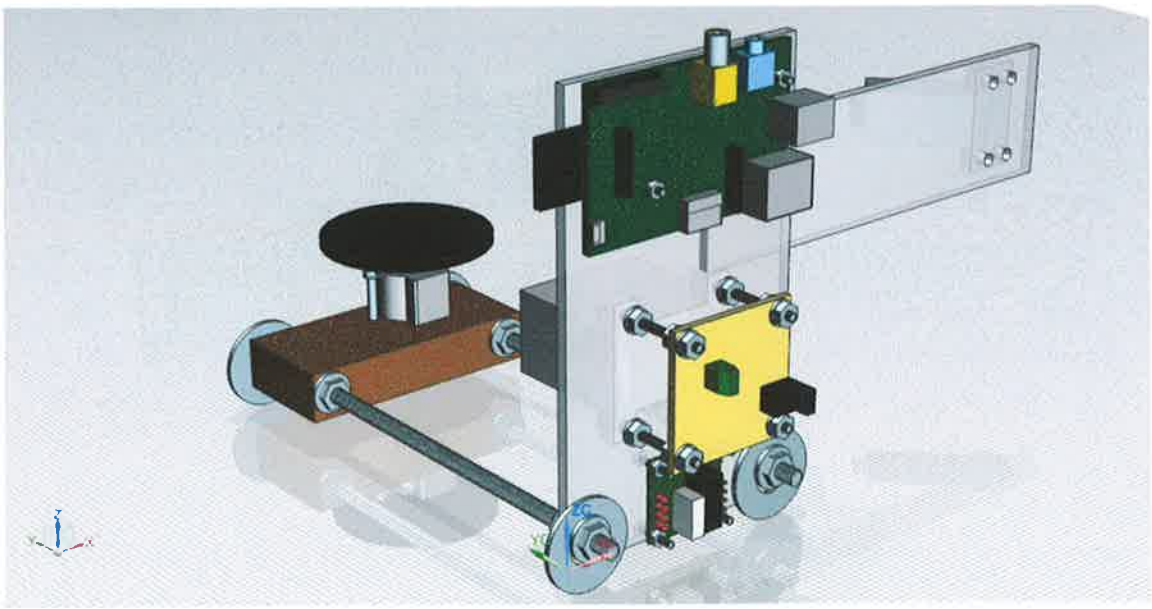


Figura 41 Perspectiva trimétrica rotada 90° del modelo asistido por computadora del prototipo.

Anexo 11

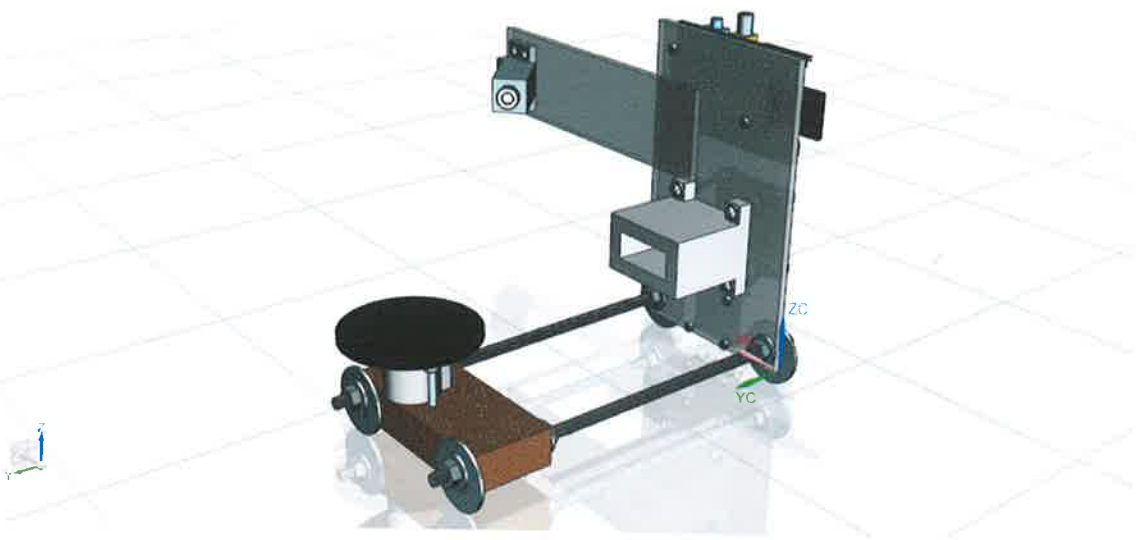


Figura 42 Perspectiva trimétrica rotada 90° del modelo asistido por computadora del prototipo.

Anexo 12

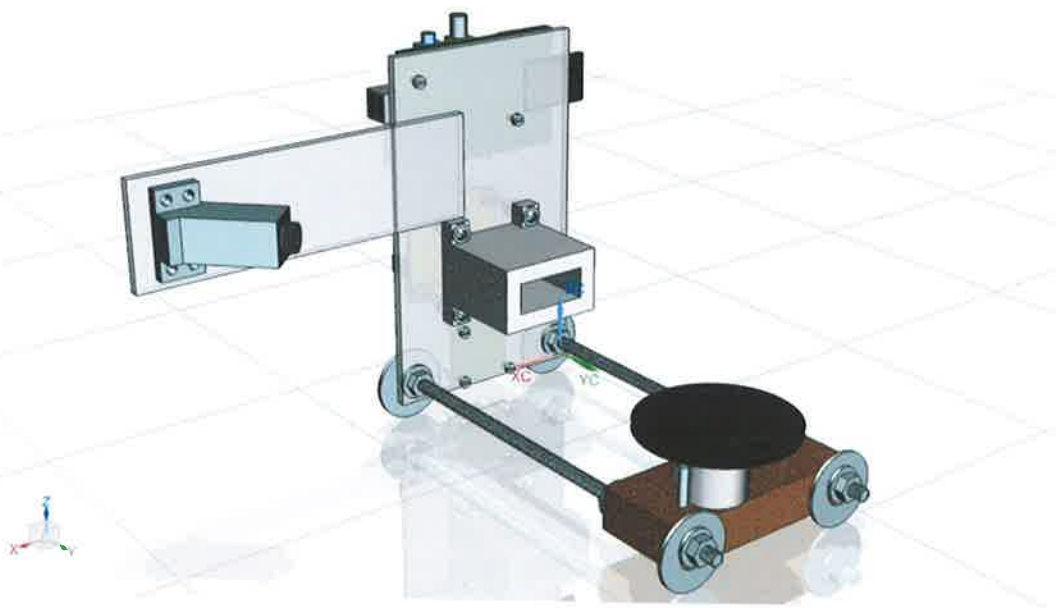


Figura 43 Perspectiva trimétrica rotada 90° del modelo asistido por computadora del prototipo.

Anexo 13

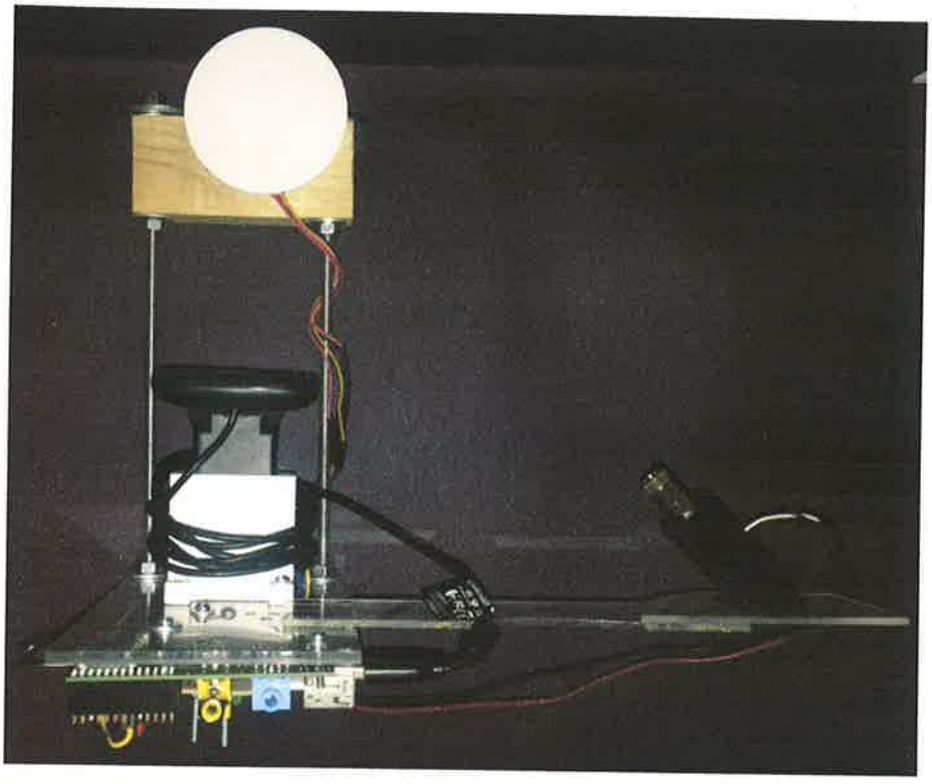


Figura 44 Perspectiva superior del prototipo construido

Anexo 14

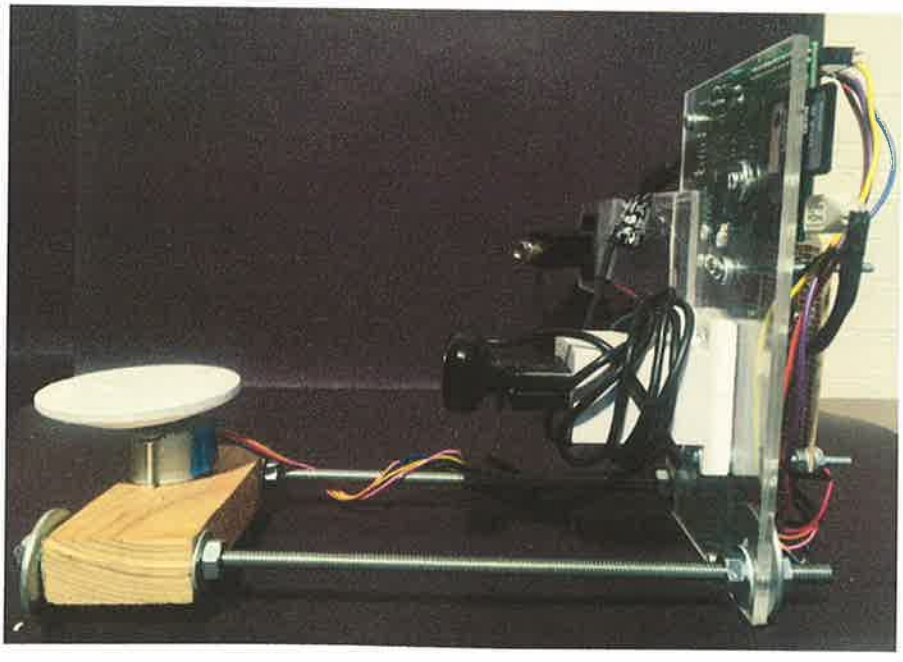


Figura 45 Perspectiva izquierda del prototipo construido

Anexo 15

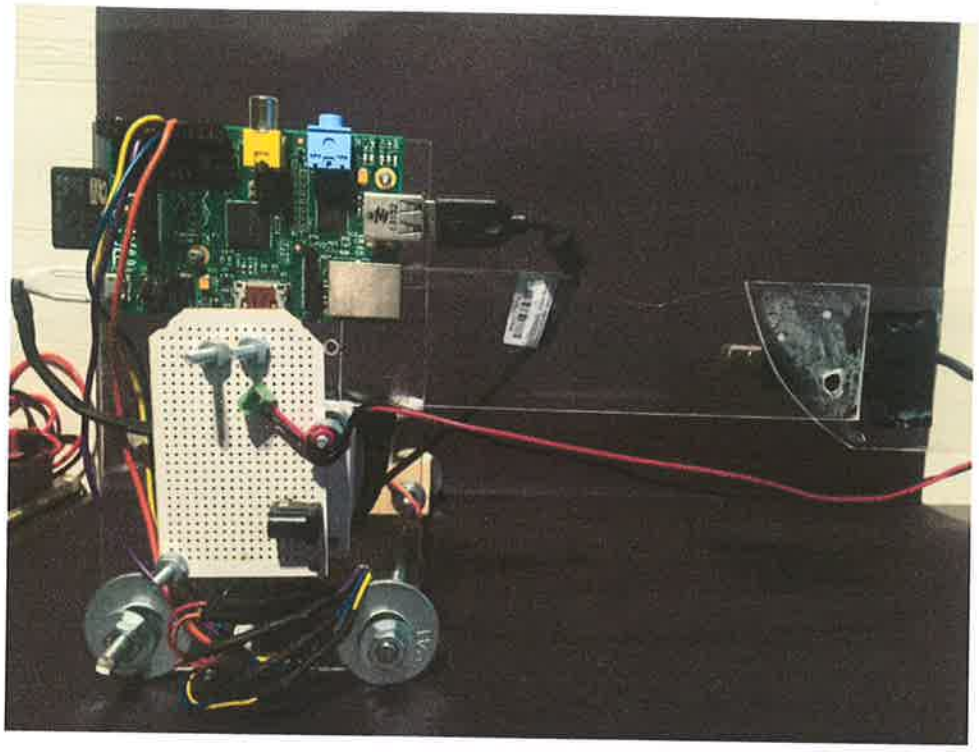


Figura 46 Perspectiva frontal del prototipo construido

Anexo 16

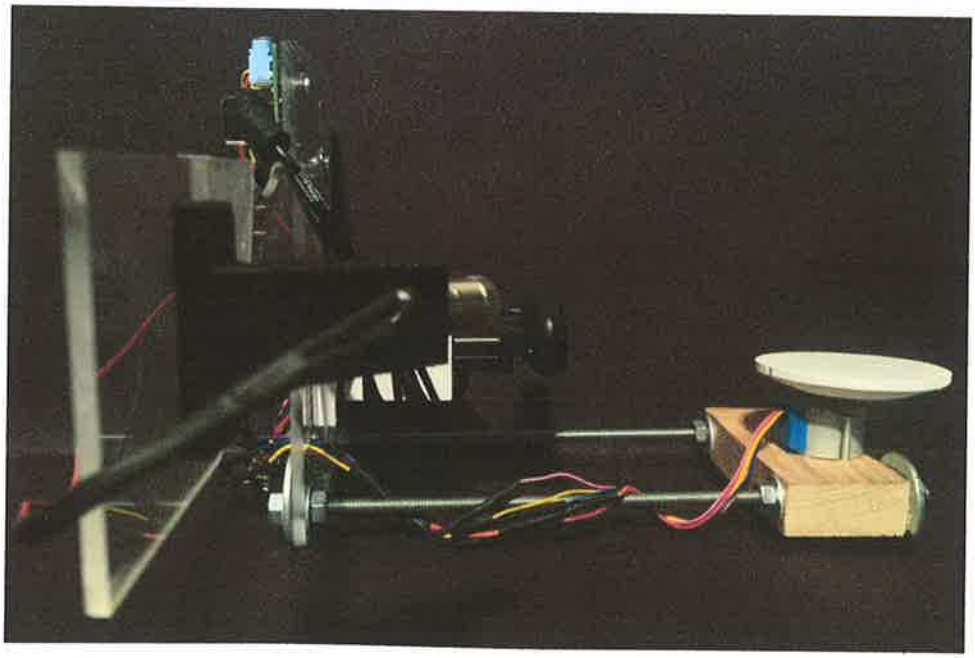


Figura 47 Perspectiva derecha del prototipo construido

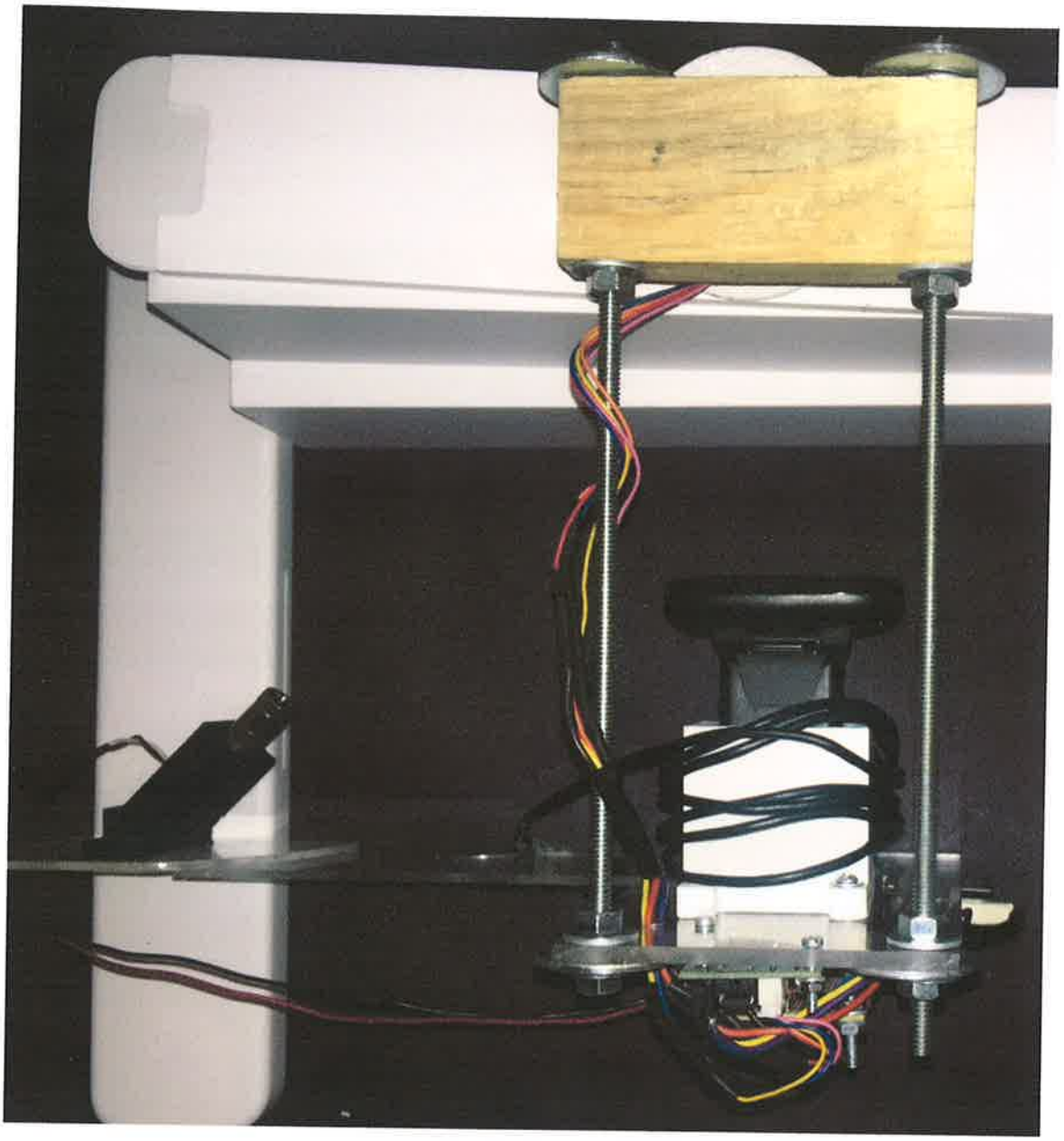


Figura 48 Perspectiva inferior del prototipo construido

Anexo 18

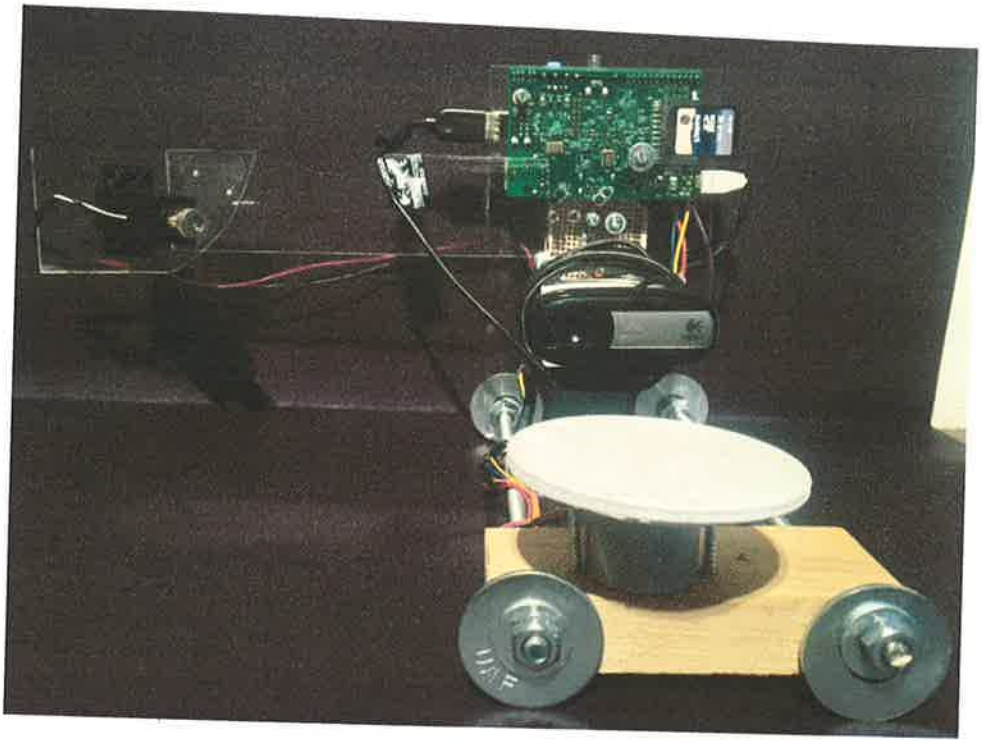


Figura 49 Perspectiva trasera del prototipo construido

Anexo 19

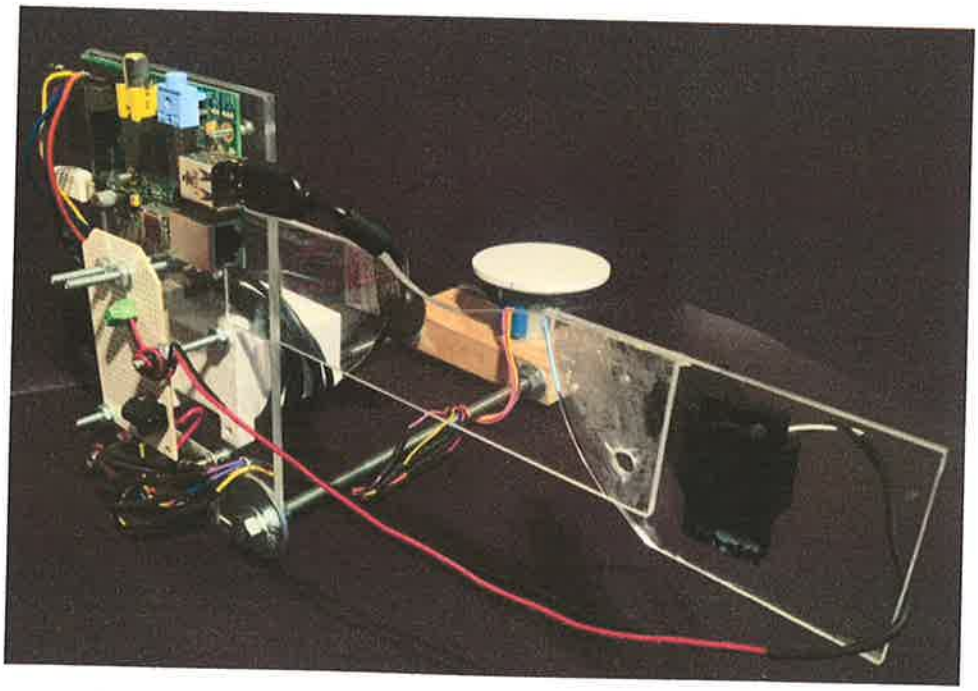


Figura 50 Perspectiva isométrica del prototipo construido