UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Montaje y mejora de un scanner 3D de bajo coste basado en diseños libres

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Daniil Gavrishev

Tutor: Alberto José Pérez Jiménez

Curso 2018-2019

Resumen

Este documento recoge el procedimiento seguido para la realización de un escáner 3D partiendo de uno de código abierto ya creado y que se puede encontrar en Internet. Se detalla los pasos seguidos así como los problemas surgidos a lo largo del desarrollo, explicando las decisiones que se tomaron para solucionarlos. Además se proponen mejoras de iluminación para mejorar el proceso de escaneado y proporciona una guía para la modificación de la nube de puntos resultante en un programa de procesamiento de mallas.

Palabras clave: Escáner, Python, código abierto, 3D, bajo coste, nube de puntos.

Abstract

This document gather the procedure followed to create a 3D scanner based of an open source scanner already created and available on the Internet. It details the steps followed and the problems that took place as well as the explanation of how to solve them. Also it propose illumination related upgrades to improve the scanning process and provides a guide to modify the point cloud created with a mesh processing software.

Keywords : Scanner, Python, open source, 3D, low cost, point cloud.

Tabla de contenidos

1. Introducción

Con la reciente popularidad de las impresoras 3D{1} en todo el mundo[1], mucha gente que está interesada en crear y diseñar sus propios diseños y proyectos de forma sencilla y sin tener que recurrir a empresas especializadas que exigen un cobro por producir dichos diseños[2], tiene a su disposición una impresora 3D, ya sea en su hogar o en algún lugar donde la impresora es compartida por varios individuos.

Para poder imprimir los diseños deseados, es necesario tener un conocimiento en diseño digital o tener soltura a la hora de utilizar programas de edición 3D[3]. Esto es debido a que para poder imprimir un diseño, es necesario crear primero ese diseño con algún programa de edición digital y exportarlo al formato G-code, ya que las impresoras 3D únicamente reconocen esta extensión a la hora de la impresión. También cabe la posibilidad de contratar a algún personal especializado para el diseño del objeto[4] que tratamos de diseñar, pero esta opción contradice nuestra filosofía de no tener que contactar con ningún tipo de empresa/personal especializado, además de que al depender de otros individuos, el tiempo y el coste está fuera de nuestro control.

Por eso, la posibilidad de crear un escáner de objetos 3D que sea capaz de crear los diseños que deseamos sin tener que recurrir a agentes externos, además de ser más económico, es algo que motivará a estas personas a desarrollar sus propios escáneres 3D.

1.1. Motivación

La motivación principal para realizar este TFG{2} ha sido el interés de conocer como es el desarrollo de un escáner 3D, desde la estructura del mismo, pasando por los diferentes componentes que se requieren y la tecnología y algoritmos utilizados.

También estaba interesado en el trabajo que se debe realizar con el objeto escaneado utilizando algún programa de edición digital. Además, sentía curiosidad por aprender cómo es posible poder escanear un objeto físico en tres dimensiones. Al estar el código del *software* utilizado escrito en 'Python', me atrajo la idea de poder mejorar mis conocimientos en este lenguaje de programación ya que es un lenguaje bastante utilizado hoy en día.

Otra motivación que me llevó a realizar este TFG, es la posibilidad de diseñar y desarrollar un escáner 3D de mayor escala, o mejorando el diseño actual, para poder escanear objetos mayores. Este TFG también puede servir de guía para otras personas que deseen desarrollar este escáner 3D ya que en la documentación no se alude a los diferentes problemas que pueden surgir si se decide desarrollar el escáner sin adquirir el *kit* con los componentes.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo principal de este TFG es el de desarrollar un escáner 3D de bajo coste a partir de las especificaciones del escáner comercial 'Ciclop' de **'BQ'**, utilizando *hardware* y *software* de diseño libre y que no requiera la compra de la totalidad del *kit* que ofrecen. Además de realizar algún tipo de mejora en el escáner resultante para un funcionamiento mejorado.

1.2.2. Objetivos específicos

- Crear las piezas necesarias imprimiendolas con una impresora 3D. Adquirir las demás piezas necesarias, entre las cuales se encuentra el shield del control del motor y los láseres de **'BQ'**, diferentes tuercas y tornillos, así como el motor, los láseres, la cámara, la placa 'Arduino' y los cables.

- Montar el escáner es tu totalidad.

- Estudiar y entender el funcionamiento de cada componente, tanto hardware como software.

- Profundizar en el lenguaje de programación 'Python'.

- Una vez hecho, hacer las pruebas necesarias para ver y entender su funcionamiento.

- Hacer las modificaciones necesarias para solucionar los problemas de las pruebas, y para mejorarlo.

1.3. Estructura

La estructura del TFG se dividirá en:

En el punto **1º. Introducción** se introduce brevemente al lector a la tecnología de los escáneres 3D además de exponer los objetivos.

En el punto **2º. Contexto Tecnológico** se explica la historia, el funcionamiento y las diferentes técnicas utilizadas con escáneres 3D.

En el punto **3º. Análisis del problema** se describen las diferentes soluciones por las que se puede optar, la que últimamente fue escogida, el plan de trabajo y el presupuesto.

En el punto **4º. Diseño de la solución** se indica y describe los diferentes elementos utilizados, tanto *hardware* como *software*.

En el punto **5º. Desarrollo de la solución** se explica el desarrollo seguido para la creación del escáner 3D además de las mejoras realizadas y de un ejemplo de escaneado de un objeto.

En el punto **6º. Resultados** se expone el resultado obtenido.

En el punto **7º. Conclusiones** se analiza si se ha cumplido con los objetivos además de posibles mejoras que se pueden aplicar.

En el punto **8º. Bibliografía** se facilita información relacionada con el TFG.

En el punto **9º. Glosario** se explican algunos términos y siglas utilizadas.

2. Contexto Tecnológico

2.1. Introducción

Los primeros escáneres 3D surgieron a principios de los años 80, donde se utilizaban escáneres de contacto para recoger el mayor número de datos y crear una imágen digital del objeto. Esta técnica tardaba mucho tiempo en completarse y por eso se empezó a estudiar nuevas técnicas que permitan hacer el escaneo de objetos más rápidamente. Estos escáneres se fabricaron únicamente para aplicaciones industriales y mayoritariamente para inspecciones de superficies[5].

A finales de los años 80, se empezó a experimentar con tecnologías ópticas, que consiste en obtener una nube de puntos el objeto que se desea escanear para luego extrapolar la forma del objeto. Esta tecnología permitió reducir considerablemente el tiempo de escaneo, pero el espacio de almacenamiento de las máquinas de aquella época no permitían almacenar toda la información que se obtenía del objeto que se intentaba escanear para su posterior modificación[6].

En esta época, existían dos tipos de tecnologías ópticas:

- **Punto:** esta técnica utiliza un único punto de referencia el cual obtiene los datos, es parecido a la técnica de contacto, que se explica más adelante. Esta opción es la más lenta ya que requiere que el láser se mueva físicamente muchas veces.
- **Línea:** se utilizaba una banda de muchos puntos que pasan por el objeto de una pasada. Esta técnica satisface las exigencias tanto de velocidad como de precisión, por tanto era la más utilizada.

Las primeras aplicaciones eran las capturas de personas para la industria de la animación. **'Cyberware Laboratories'**, de Los Ángeles, fue la primera en desarrollar escaneos de rostros, unos años más tarde consiguieron escanear el cuerpo completo. En 1994, lanzaron 'REPLICA', que permitía un escaneo rápido y preciso de objetos muy detallados. **'Digibotics'** lanzó un escáner de 4-ejes que permitía un escaneo completo del objeto en una única pasada, pero era demasiado lento ya que no utilizaba la técnica de línea sino la de punto. Estos escáneres eran caros y por eso **'Immersion'** y **'Faro Technologies'** introdujeron sus escáneres de bajo coste, pero eran lentos y no podían distinguir el color de los objetos[7].

En 1996, **'3D Scanners'** combinó la técnica de línea y un brazo mecánico que se operaba de forma manual, y de esta forma consiguió que fuera rápido y flexible, capaz de producir modelos complejos con precisión y color.

Con el paso de los años, hemos conseguido mejorar la memoria de los ordenadores y por tanto, esta tecnología es la más utilizada en los diversos escáneres que existen en el mercado e incluso los que se pueden desarrollar con código abierto.

2.1.1. Contacto

Los escáneres de contacto utilizan un elemento de medida llamado palpador{3} sobre la superficie del objeto que se está escaneando. Típicamente se utiliza una punta de acero o zafiro. Los sensores que se encuentran en su interior, permiten determinar la posición en la que se encuentra el palpador en cada momento.

Los escáneres de contacto pueden ser de tres tipos[8]:

- Un sistema con un carril y con brazo rígido se sujeta firmemente perpendicularmente al objeto, y cada eje se desliza a lo largo de la pista. Estos sistemas son preferibles con objetos planos o con simples curvaturas.
- Un sistema con un brazo articulable con sensores angulares de gran precisión. La localización al final del brazo implica una serie de cálculos matemáticos complejos. Este sistema se utiliza para escanear grietas y objetos muy detallados.
- Un sistema que combina los dos tipos previamente explicados. Estos se utilizan para escanear grandes objetos que a su vez son detallados.

Un 'CMM'{4} es un ejemplo de un escáner de contacto. Este escáner puede ser muy preciso pero lento, en relación con los escáneres que utilizan tecnologías ópticas. Además, al ser de contacto, significa que el escáner está en contacto con el objeto y como consecuencia de esto, es posible que dañe el objeto que se está escaneando, cosa que no es deseable cuando se intenta escanear objetos valiosos o artefactos históricos.

2.1.2. Sin contacto

Existen dos tipos de escáneres son contacto: activos y pasivo.

Los escáneres activos emiten algún tipo de radiación o luz, y detectan el reflejo que pasa por el objeto para determinar su posición. Pueden ser de los siguientes tipos:

- Tiempo de vuelo: este tipo de escáner usa un distanciómetro láser{5}, que su función es la de determinar la distancia a la que se encuentra el objeto

cronometrando el tiempo que tarda el haz o pulso de luz en ir y volver. Esto solo produce un único punto del objeto, para completar el escaneo completo, se debe de mover el distanciómetro láser tras cada medida o utilizando un sistema de espejos rotatorios para modificar la dirección del haz.

Las ventajas de este tipo de escáner es que pueden operar a largas distancias, puede obtener muestras rápidamente y pueden medir la distancia al objeto entre 10.000 y 100.000 puntos por segundo. La precisión de este tipo de escáneres se puede perder si se encuentra con un borde, ya que la información que entiende el escáner es la de dos superficies diferentes para un mismo pulso de luz. Esto se puede solucionar utilizando un láser con una amplitud del haz más pequeña, pero la distancia a la que puede escanear objetos se vería reducida ya que la amplitud del haz se vería aumentada cuanto más larga sea la distancia. Son útiles para escanear edificios y formaciones rocosas.

- Triangulación: esta técnica utiliza un haz láser que incide en el objeto y utilizando una cámara se encuentra la ubicación del punto del láser, y dependiendo de la distancia a la que el láser incide en un objeto, el punto aparece en diferentes posiciones en el campo de visión de la cámara.

  La técnica se denomina triangulación porque el punto del haz láser, la cámara y el emisor láser forman un triángulo. La distancia del emisor láser y la cámara se conoce, al igual que los ángulos a los que se encuentran al punto, y con estos datos se puede obtener los demás valores del triángulo y conocer cada posición en el espacio.

  Al contrario que la técnica de tiempo de vuelo, los escáneres que utilizan la técnica de triangulación tienen una distancia limitada, pero con una gran precisión, en el orden de decenas de micrómetros.

  A menudo se utiliza una línea en vez de un único punto para acelerar el proceso de adquisición de datos, y es precisamente lo que se utiliza en el escáner del qual trata este TFG.

- Diferencia de fase: esta técnica mide la diferencia de fase entre la luz emitida y la recibida, que se utiliza para estimar la distancia del objeto. El haz láser emitido es continuo y de potencia modulada.

  Tiene un rango medio, situándose entre las técnicas de tiempo de vuelo y triangulación, además de tener una precisión parecida a la de la triangulación. Al modular el haz con una frecuencia constante, existe la posibilidad de haber ambigüedad en la medida de la distancia proporcional a la longitud de onda de la modulación utilizada. Esto

depende de la frecuencia utilizada y la distancia a la cual se encuentra el objeto, por tanto hay que encontrar un punto medio para ambos valores, o bien utilizar dos frecuencias diferentes. De esta forma, la frecuencia mayor se usa para calcular la distancia al punto mientras que la menor para resolver la ambigüedad de la medida.

Esta técnica tiene una velocidad de adquisición de datos muy alta, oscilando entre los 100.000 y 1.000.000 de puntos por segundo.

- Holografía conoscópica: esta técnica utiliza un haz que se refleja en la superficie del objeto pasando por un cristal birrefringente, que es un cristal con dos índices de refracción, uno fijo y el otro que es función del ángulo de incidencia del haz en la superficie del cristal.

  Al atravesar el cristal, se obtienen dos rayos paralelos que se hacen interferir utilizando una lente cilíndrica, para posteriormente capturar esa interferencia utilizando un sensor de una cámara obteniendo un patrón de franjas. La frecuencia de la interferencia determina la distancia al objeto.

  Esta técnica permite utilizar cualquier tipo de luz que sea monocromática[6], por tanto la fuente de luz no tiene porque ser un haz láser.

- Luz estructurada: esta técnica proyecta luz sobre la superficie del objeto y analiza la deformación producida por la geometría del objeto en sí. Esta técnica se basa en proyectar una línea sobre el objeto utilizando un proyector 'LCD'[7] o un láser; una cámara, desviada ligeramente del proyector de modelo, mira los puntos y calcula la distancia a la cual se encuentra cada punto en el campo de visión.

  Las ventajas de utilizar esta técnica es la velocidad y precisión, ya que se escanea varios puntos de una sola pasada, eliminando los problemas que pueden surgir con la deformación del objeto en movimiento.

- Luz modulada: esta técnica emite constantemente una luz cambiante en el objeto. Esta luz cicla con un patrón sinusoidal[8] y una cámara detecta la luz reflejada y la cantidad que el patrón de luz cambia para determinar la distancia a la que se encuentra el objeto.

Los escáneres pasivos no emiten ningún tipo de radiación, sino que se basan en detectar la radiación ambiental. Podemos encontrar los siguientes tipos:

- Estereoscópicos: este sistema utiliza dos cámaras de vídeo, ligeramente separadas, y enfocadas en el mismo objeto. Analizando las diferencias entre las imágenes de cada cámara, se puede determinar la distancia a la

que se encuentra el objeto. Este sistema es parecido a la visión de las personas.
- Fotométricos: se utiliza una única cámara pero toma múltiples imágenes bajo varias condiciones lumínicas. Esta técnica intenta invertir el modelo de formación de la imágen para recuperar la orientación de la superficie de cada *píxel*[9].
- Silueta: esta técnica utiliza fotografías realizadas a un objeto en tres dimensiones sobre un fondo bien contrastado, estirandolas y cruzandolas posteriormente para formar una aproximación del objeto.

Una vez se ha escaneado el objeto, se consigue una nube de puntos que se debe de reconstruir para poder obtener el objeto en forma digital. Estas nubes de nubes se suelen convertir a un modelo de malla de polígonos o triángulos[10], modelos de superficie o modelos sólidos 'CAD'[11]:

- Modelos de malla de polígonos o triángulos: estos modelos se basan en unir los puntos de la nube con puntos adyacente mediante una línea recta para crear el objeto.
- Modelos de superficie: en este modelo se utiliza un conjunto de pequeñas superficies curvas, que unidas entre sí, modelan la forma del objeto escaneado. La representación más común es 'NURBS'[12].
- Modelos sólidos 'CAD': los modelos 'CAD' son totalmente editables mediante el cambio de un valor. Para obtener este modelo se pueden seguir varios enfoques; exportar la superficie 'NURBS' y que el diseñador complete el modelo 'CAD', utilizar el análisis de datos para crear el modelo basándose en las características importadas en 'CAD', o editar directamente el 'CAD' pero con un límite de puntos.

2.2. Escáneres comerciales

Estos escáneres utilizan software propio, por lo que nos se conoce el código utilizado. Actualmente existen varios escáneres en el mercado, algunos de los cuales se van a analizar a continuación. Estos no son todos lo que existen, pero al exceder considerablemente el precio de los que se van a mencionar, se van a obviar, pero el lector se puede hacerse una idea fijándose en el último escáner.

2.2.1. XYZprinting 3D-Scanner 2.0 (Fig.1)[9]

Figura 1 - XYZprinting 3D-Scanner 2.0

Este escáner, creado por una empresa Taiwanesa, es relativamente pequeño y compacto con unas dimensiones de 41 x 157 x 61 mm. y con un peso de 240 gramos, incluyendo 4 modos de uso. Utiliza una tecnología de luz estructurada y su distancia de funcionamiento está entre 30 y 50 cm. Las desventajas de este escáner es que necesita estar conectado en todo momento al ordenador mediante un cable 'USB'{13}, además de tener un tamaño de escaneo máximo de 100 x 100 x 200. Se puede obtener por un precio de 200€.

2.2.2. Structure Sensor (Fig.2)[10]

Figura 2 - Structure Sensor

Este escáner permite acoplarse directamente a un 'IPad' de **'Apple'** y después de instalar las aplicaciones necesarias, te permite escanear objetos hasta 4 horas. Este escáner tiene unas dimensiones de 109 x 24 x 18 mm. y un peso de 65 gramos. Permite hacer escaneos con una resolución de 1280 x 950 px. incluso al aire libre con luz solar directa sobre el objeto. Las desventajas es que

necesitas un 'IPad' de **'Apple'** específicamente para utilizarlo, lo cual añade un coste muy alto. Su precio ronda los 400$ (360€), sin contar la compra del 'IPad'.

2.2.3. Sense 3D scanner (Fig.3)[11]

Figura 3 - Sense 3D scanner

'3D Systems', la empresa que ha desarrollado este escáner, fue la primera en empresa en desarrollar escáneres en tres dimensiones. Tiene un tamaño de 178 x 129 x 330 mm. y es capaz de escanear objetos con un tamaño de 200 x 200 x 200 con una resolución de 1920 x 1080 px. Además no es un escáner fijo, es decir, requiere del usuario que controle el escáner para realizar el escaneado. Una de las desventajas es que únicamente funciona en sistemas Windows. Tiene un precio de 434€.

2.2.4. Matter Form V.2 (Fig.4)[12]

Figura 4 - Matter Form V.2

Creado por una empresa canadiense, **'Matter & Form'**, tiene un tamaño de 35 x 21 x 35 mm. con un peso de 1,71 Kg. A diferencia de los otros escáneres, este escáner es fijo y tiene una plataforma donde se coloca el objeto para su escaneo. Soporta varios lenguajes pero el tamaño máximo que puede llegar a escanear es de 25 x 18 cm., y funciona en sistemas 'Windows' y 'MacOS'.

2.2.5. EinScan-SE (Fig.5)[13]

Figura 5 - EinScan-SE

Este escáner, de una compañía china, fue creado gracias a una campaña de **'Kickstarter'**[14]. Utiliza tecnología de luz estructural con luces blancas e invisibles por lo que seguro para los ojos. Con un tamaño de 570 x 210 x 210 mm y un peso de 4,5 Kg, tiene dos modos de uso, uno automático y el otro manua. Para objetos mayores de 200 x 200 x 200 mm es obligatorio utilizar el modo manual. Tiene un precio superior a 1400€.

A partir de aquí, los escáneres disponibles comienzan a aumentar considerablemente su coste, llegando a los 6.000€ en algunos de ellos. Este aumento de coste significa que tienen más opciones, además de que permiten escanear objetos más grandes, con mejor resolución y son más portables, en el sentido de que algunos no requieren tener una posición estática para realizar el escaneo de un objeto.

2.3. Escáneres de código abierto

Además de los escáneres anteriormente analizados, existen otros que son de código abierto. Aquí se analizarán los escáneres que no requieren de un dispositivo móvil para funcionar, ya que existen bastantes proyectos que utilizan un dispositivo móvil tanto para capturar la imagen del objeto como para realizar

la función de ordenador, pero se ofrecerá una visión general de dichos proyectos.

2.3.1. MakerScanner 3D (Fig.6)[14]

Figura 6 - MakerScanner 3D

Este escáner es relativamente simple ya que no se necesita mucho para montarlo; la estructura básica del escáner se puede descargar de la página del desarrollador, una cámara 'USB PS3 Eye'[15], un láser y 2 pilas AA. Utiliza una técnica parecida a visión en estéreo, es decir, consiste de un láser y una cámara ligeramente descompensada a una distancia base 'B'. Cuanto más cerca el objeto al sensor, la línea del láser aparece más cerca del borde de la imagen de la cámara. Con una correcta calibración y cálculos matemáticos, se puede calcular la distancia de cada punto. Utilizando esta técnica se consiguen imágenes que requieren una edición posterior en algún programa de edición digital ya que la nube de puntos producida es bastante pobre, pero es lo suficiente como para hacerte una idea del objeto que se ha escaneado.

2.3.2. BQ Ciclop (Fig.7)[15]

Figura 7 - BQ Ciclop

Este escáner está creado por una empresa española y es totalmente de código abierto, desde las diferentes piezas que forman la estructura del escáner, pasando por los circuitos utilizados, como el código utilizado. En la web se puede encontrar los archivos necesarios para imprimir las piezas para la estructura del escáner. Utiliza una tecnología de triangulación y permite escanear objetos en color y creando una nube de puntos que se asemeja bastante al objeto escaneado. Al tener una plataforma giratoria, el escaneo es automático, eso significa que no requiere de acciones del usuario para concluir el escaneo a diferencia de otros escáneres de código abierto donde es necesario que el usuario controle el brazo o plataforma del escáner. Este escáner es el que se ha decidido sobre el cual se hará este TFG ya que además tiene relación con el grado al utilizar una impresora 3D para imprimir las piezas.

2.3.3. FabScan Pi (Fig.8)[16]

Figura 8 - FabScan Pi

Este escáner originalmente fue desarrollado como parte de una tesis de un estudiante francés en 2010. Está basado en la técnica de triangulación y soporta tanto 'Arduino' como 'Raspberry Pi'{16}. Es parecido al escáner 'Ciclop' anteriormente comentado, pero a diferencia de tener los archivos necesarios para imprimir en 3D las piezas que forman el escáner, utiliza tablas de fibra de densidad media y en la web se puede encontrar las medidas para crear la estructura del escáner. Una de las desventajas es que requiere de un *shield*{17} específico el cual hay que comprar ya que no hay diagramas referentes a su construcción, o también es posible comprar el *kit* completo donde se incluye todo lo necesario para montar el escáner.

2.3.4. Escáneres que utilizan un dispositivo móvil

También existen muchos escáneres de código abierto que utilizan dispositivos móviles. Esto escáneres utilizan tecnología de fotometría en vez de triangulación, por lo que no requieren de ningún tipo de láser para obtener datos. Una vez descargada una de las muchas aplicaciones móvil que permiten escanear objetos[17], el usuario debe manipular el dispositivo móvil mientras el programa toma imágenes del objeto para luego sacar la nube de puntos y crear la imagen digital. Los problemas que pueden surgir al utilizar este tipo de escáneres, además de que el proceso de adquisición de datos sea manual, es que el usuario no tenga un dispositivo móvil lo suficientemente bueno como para la aplicación utilizada pueda crear el objeto a partir de las imágenes tomadas, o que no tenga una cámara con suficiente resolución para tomar fotos suficientemente nítidas como para crear el objeto.

2.4. Propuesta

La propuesta que se va a presentar es la de desarrollar un escáner 3D de código abierto utilizando como referencia el escáner 'Ciclop' de **'BQ'**. Esta propuesta es un mejora considerable en cuanto al precio total de todos los componentes, además de ser de código abierto, que permite la participación de otras personas y de esta forma poder mejorar el escáner con modificaciones en el código, por ejemplo. La estructura del escáner se puede conseguir mediante impresión 3D ya que los diseños se encuentran en la web de la empresa que lo desarrolla.

3. Análisis del problema

A continuación se va a proceder a comentar las diferentes soluciones que se plantearon y la solución escogida. También se indicará el plan de trabajo que se siguió, la propiedad intelectual y el presupuesto final.

3.1. Identificación y análisis de posibles soluciones

Para la realización de este TFG, se analizaron posibles soluciones, entre las cuales se encuentran:

I. Comprar el *kit* completo disponible en la web del autor: esto significaría que la mitad del TFG ya estaría completo, pero habría que pagar en torno a los 250€ para conseguirlo, además de que en los últimos años es casi imposible conseguir un *kit* completo y habría que recurrir a vendedores de segunda mano, con los cuales no se puede estar seguro en que estado se encuentra el escáner.
II. Comprar la placa controladora desarrollada por BQ y el *shield* del escáner: esto sería comprar las dos partes más importantes del escáner que además permite configurar el escáner para que opere mediante 'bluetooth'[18]. Viendo que la placa controladora es básicamente un 'Arduino' pero con menos funciones, se decidió no elegir esta opción.
III. Comprar únicamente el *shield* que controla el motor, la placa 'Arduino Uno' y los láseres: esta opción es la elegida ya que lo único que se debe adquirir es el *shield* que controla el motor y los láseres según las especificaciones del código utilizado por el autor.
IV. No comprar nada y crear el *shield* que controla el motor y los láseres: esta última opción no requeriría comprar casi nada y por tanto se reduciría el coste material, pero el coste de desarrollo se elevaría considerablemente ya que requeriría del desarrollo del *shield*, la estructura del cual, el autor pone a disposición en la web, por lo que se podría crear el *shield* desde cero. Pero últimamente no se eligió esta opción ya que requiere aprender a crear *shields* además de pedir cada componente que forma el shield, incluyendo la FPGA[19], y que llegue a tiempo.

3.2. Solución propuesta

Como se comentó anteriormente, la opción elegida fue la de comprar únicamente el *shield* que controla el motor, la placa 'Arduino Uno' y los láseres.

Esto nos permite obtener los demás componentes del escáner creandolos nosotros mismos, como es el caso de imprimir las diferentes piezas que componen la estructura del escáner, utilizar un arduino para controlar el *software* del escáner, y conseguir las demás piezas *hardware* por un módico precio.

3.3. Plan de trabajo

3.3.1. Plan previsto

El plan de trabajo previsto para este TFG fué el siguiente:

Tarea a realizar	Tiempo	Razonamiento
Imprimir las diferentes piezas utilizando una impresora 3D	14 días	Al utilizar la impresora 3D que se encuentra en la universidad, tengo poco acceso a ella, por tanto la idea era de imprimir una pieza al día.
Montar el escáner	1-2 días	No parece ofrecer grandes dificultades.
Instalar el *software* necesario para el funcionamiento del escáner	1 día	No hay muchos programas *software* que instalar.
Familiarizarme con el programa 'Horus'	2-3 días	Al ser un programa nuevo, debo de mirar todas las opciones y ver cómo afecta al funcionamiento del escáner.
Aprender más sobre 'Python' y entender el código utilizado	7 días	Entender un código nuevo que además es de código abierto siempre es complicado, además tenía que aprender más sobre 'Python' ya que tenía las nociones básicas.
Hacer pruebas	2-3 días	Una vez esta todo montado y estoy familiarizado con el programa 'Horus', habría que hacer unas pruebas para ver que todo funciona correctamente.
Hacer las modificaciones necesarias	El tiempo sobrante	El tiempo que sobre hasta Agosto hacer las modificaciones que se crean convenientes y las que se pueda hacer.
Repetir las pruebas con las modificaciones	1 día	Una vez cada modificación esté hecha, habría que probarla.

3.3.2. Plan actual

Al final, el plan de trabajo resultante fué el siguiente:

Tarea a realizar	Tiempo	Razonamiento
Imprimir las diferentes piezas utilizando una impresora 3D	34 días	Hubo problemas con las piezas a la hora de imprimirlas, se explicará más adelante.
Montar el escáner	2 días	Según lo previsto.
Instalar el *software* necesario para el funcionamiento del escáner	1 día	Según lo previsto.
Familiarizarme con el programa 'Horus'	2 días	Según lo previsto.
Aprender más sobre 'Python' y entender el código utilizado	15 días	El código era más complejo que a simple vista, además de ser diferente en **'Github'**[20] al código que se encuentra en el programa instalado.
Hacer pruebas	3 días	Según lo previsto.
Hacer las modificaciones necesarias	14 días	Con el tiempo sobrante hubo suficiente tiempo para hacer las modificaciones básicas que se pretendían al principio del TFG, además de añadir la opción de encender y apagar el *LED* desde la interfaz del programa.
Repetir las pruebas con las modificaciones hechas	1 día	Según lo previsto.

3.4. Propiedad intelectual

Todo el *software* utilizado en este TFG tienen una licencia libre y de código abierto.

El diseño del 'Shield ZUM SCAN' está basado en la placa 'Arduino CNC Shield'[21] y está liberada bajo la licencia *'Creative Commons Attribution Share-Alike'* ('CC-BY-SA').

El programa *software* principal, 'Horus', es *software* libre liberado bajo la licencia 'GPLv2'[22].

El *firmware* del programa 'Horus', está liberado bajo la licencia 'GPLv3'[23].

3.5. Presupuesto

A continuación se mostrará el presupuesto de cada elemento utilizado en el desarrollo del escáner, así como del total.

Artículo	Precio unidad (€)	Cantidad	Subtotal
Motor Nema 17	7,55 €	1	7,55 €
Rodamiento de bolas modelo 16014	11,86 €	1	11, 86 €
Tuerca M8	0,10 €	28	2,80 €
Arandela M8	0,10 €	18	1,80 €
Tuerca M3	0,20 €	3	0,60 €
Tornillo M3 x 10mm	0,20 €	9	1,80 €
Tuerca M3 x 30mm	0,20 €	5	1,00 €
Varillas roscadas de 1 metro	1,50 €	2	3,00 €
Driver A4988	0,80 €	1	0,80 €
Fuente de alimentación	0,50 €	1	0,50 €
Cámara USB Logitech c270	20,00 €	1	20,00 €
ZUM SCAN Shield	16,00 €	1	16,00 €

Láser lineal	3,00 €	2	6,00 €
Relé de dos canales	0,90 €	1	0,90 €
LED COB 3W	0,50 €	1	0,50 €

Precio total: 75,11 €

4. Diseño de la solución

En este apartado se describirán y analizarán los diferentes componentes implicados en el funcionamiento del escáner, tanto *hardware* como *software*, además de una explicación donde se indicará como se interconecta cada elemento.

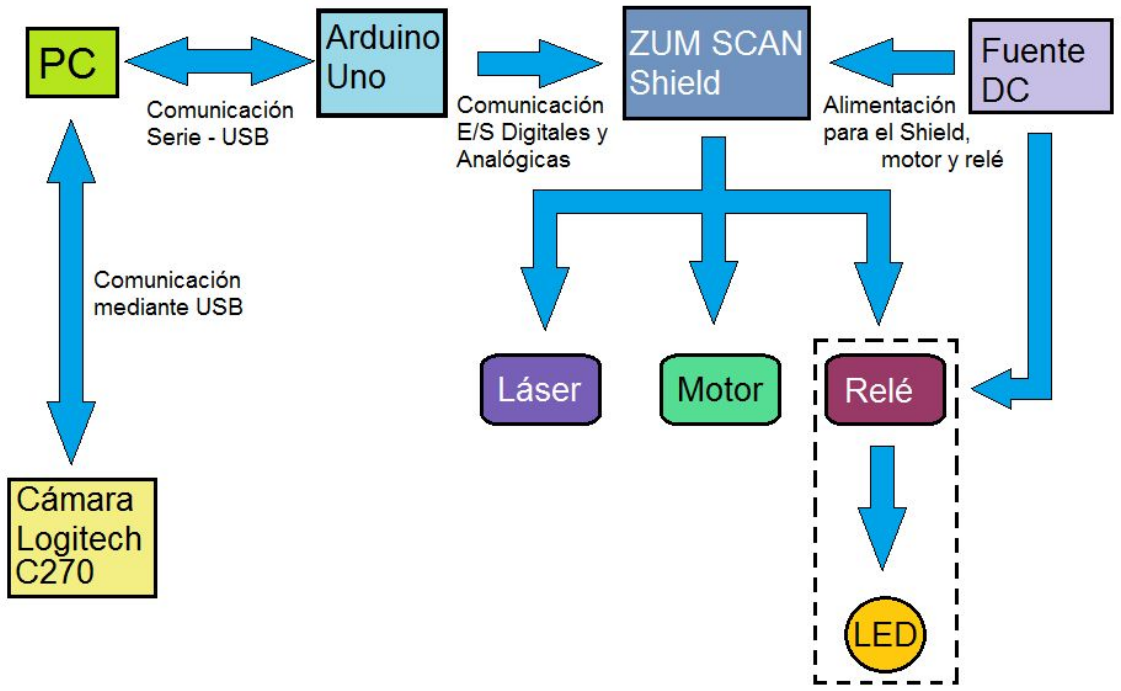


Figura 9 - Diagrama de bloques

El programa *software* utilizado, 'Horus', se encuentra en el PC[24] y es el encargado de enviar órdenes al *shield* para activar y desactivar los láseres, motor y relé por medio de comandos 'G-code', así como de indicarle a la cámara cuando debe de capturar una imágen. El 'Arduino Uno' hace la función de puente entre el programa y el *shield* al permitir la comunicación entre ambos. Se dispone además de una fuente de alimentación que da potencia y permite el funcionamiento del motor y del relé. Tanto el relé como el *LED*[25], se añadirán más adelante como mejoras del escáner. Este proceso se puede ver en la Figura 9.

Cuando el escaneo comienza, el programa le indica la *shield* que debe de encender uno de los láseres y luego le indica a la cámara que haga una captura. Vuelve a repetir el proceso pero esta vez indicando que debe encender el otro láser después de apagar el anteriormente encendido. A continuación, le indica al *shield* que apague el láser actual para sacar una captura del objeto en sí para conseguir la textura, es en este momento que se enciende el *LED* para dar luz al

objeto. Esta sería la mejora añadida para mejorar el proceso de captura de la textura del objeto. Luego se le indica que mueva el motor los grados especificados, apaga el *LED* y vuelve a comenzar de nuevo el proceso de escaneo. Así continúa hasta haber dado una vuelta completa de 360º, en este momento se detiene y se obtiene la imagen digital del objeto escaneado.

4.1. Hardware

4.1.1. Piezas impresas 3D (Fig.10)

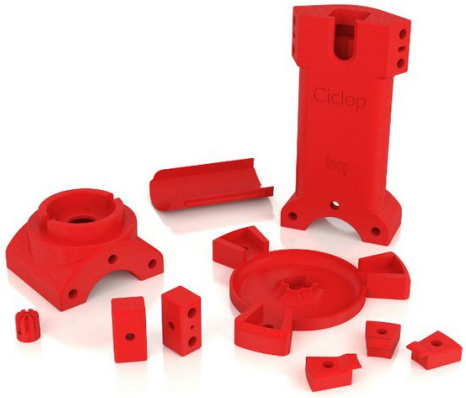


Figura 10 - Piezas impresas 3D

Las piezas que componen la estructura del escáner han sido impresas con una impresora 3D. Los ficheros que contienen el diseño en formato *.stl* se encuentran en la página web de la empresa que lo ha desarrollado. En los diseños aportados, faltaría la parte superior de la plataforma giratoria que se ha tenido que diseñar a parte e imprimirla.

4.1.2. Rodamiento de bolas modelo 16014 (Fig.11)

Figura 11 - Rodamiento de bolas

El rodamiento de bolas se ha utilizado para hacer que la plataforma del escáner gire. Tiene un diámetro de 110 mm y una altura de 13 mm.

4.1.3. Tornillería (Fig.12)

Figura 12 - Tornillería

Las diferentes tuercas, tornillos y varillas roscadas se han utilizado para unir las piezas impresas entre sí para formar el escáner además de sujetar el motor y los láseres en su sitio.

4.1.4. Superficie antideslizante (Fig.13)

Figura 13 - Superficie antideslizante

Una superficie antideslizante para que el objeto que se pretende escanear no se mueva durante el movimiento de la base.

4.1.5. Motor paso a paso Nema 17 con conector (Fig.14)

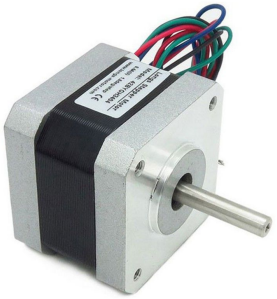


Figura 14 - Motor paso a paso

El motor paso a paso 'Nema 17' es el que se encarga de hacer girar la plataforma del escáner. Es necesario un *driver*[26] para controlar los pasos del motor. El motor está compuesto por una serie de bobinas que se magnetizan y desmagnetizan para mover el rotor de su interior según los pasos que se requieran.

4.1.6. Driver A4988 (Fig.15)

Figura 15 - Driver A4988

El *driver* que permite controlar los pasos del motor se encuentra en el *shield*. Tiene cinco modos de funcionamiento: *full-step*, *half-step*, *quarter-step*, *eight-step* y *sixteenth-step*. Al driver se le envía un pulso que a su vez este le indica al motor que debe moverse un paso.

A continuación, en la figura 16, se muestra un diagrama de los diferentes pines del *driver*:

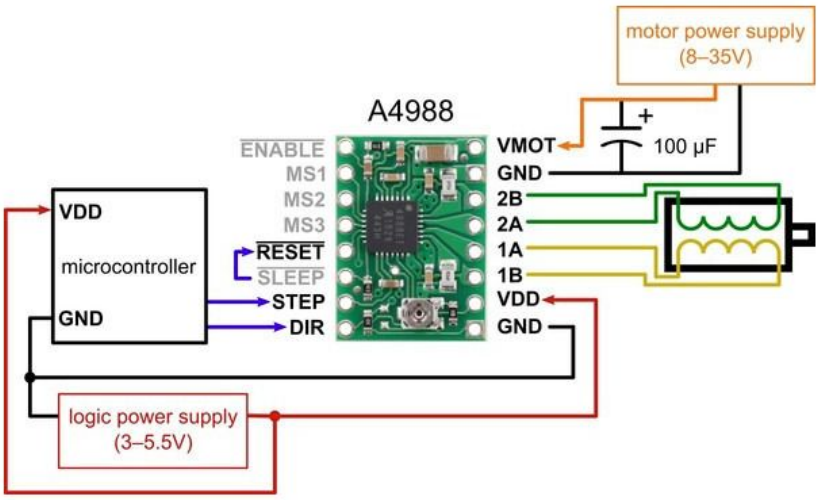


Figura 16 - Diagrama driver

4.1.7. Cable USB (Fig.17)

El cable 'USB' conecta la placa 'Arduino Uno' al 'PC'.

Figura 17 - Cable USB

4.1.8. Fuente de alimentación (Fig.18)

La fuente de alimentación de 12V se utiliza para dar potencia al *shield* para que se pueda mover el motor y encender los láseres. Posteriormente se modificó el cable para añadirle una salida adicional para poder incorporar el relé al diseño final.

Figura 18 - Fuente de alimentación

4.1.9. Cámara USB Logitech C270 (Fig.19)

La cámara 'Logitech C270' se utiliza para capturar la imagen de los láseres y del objeto para conseguir su color. Se conecta directamente al 'PC' y el programa 'Horus' detecta automáticamente la cámara si es la única que se tiene conectada.

Figura 19 - Cámara USB Logitech C270

4.1.10. Arduino Uno (Fig.20)

Figura 20 - Arduino Uno

La placa 'Arduino Uno', conectada al 'PC', permite al programa 'Horus' interactuar con el shield. Se requiere subir a la placa el código de 'Grbl' para que funcione la comunicación entre ambos dispositivos, se explicará más adelante el procedimiento a seguir.

4.1.11. ZUM SCAN Shield (Fig.21)

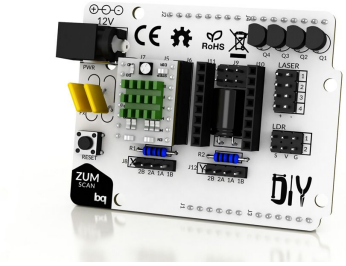


Figura 21 - ZUM SCAN Shield

Este *shield* permite la comunicación con el motor y los láseres del escáner, y posteriormente con el *LED*. Está basada en la placa 'Arduino CNC Shield'. Además permite el control de dos motores paso a paso y cuatro láseres, por lo que permite aplicarle futuras mejoras.

4.1.12. Láser lineal (Fig.22)

Figura 22 - Láser lineal

El láser lineal que se ha instalado, permite al escáner obtener la forma física del objeto. El escáner dispone de dos láseres, por lo que se obtienen dos imágenes del objeto que se solapan para conseguir un resultado mejor. La línea que presenta el láser se obtiene al pasar el haz por una lente cilíndrica.

4.1.13. Módulo relé de dos canales (Fig.23)

Figura 23 - Módulo relé de dos canales

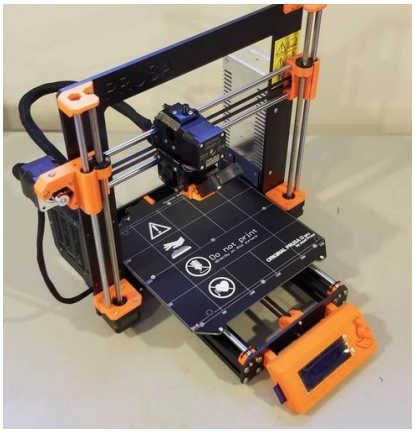
Relé de dos canales utilizado para permitir el encendido y apagado del *LED*. Requiere de una fuente de alimentación, masa y una de una señal que le indique cuando debe permitir el paso de corriente. Esta señal se implementó como un comando en 'G-code'.

4.1.14. LED (Fig.24)

Este *LED* se utiliza para dar iluminación a la hora de capturar los datos referentes a la textura del objeto. Sería recomendable utilizar un *LED* de baja potencia.

Figura 24 - LED

4.1.15. Impresora 3D Prusa i3 MKII (Fig.25)



La impresora 3D con la que se imprimió todas las piezas, fue la Prusa i3 MKII. Esta impresora es la más conocida y por tanto de las más utilizadas, por lo que encontrar soluciones a los diferentes problemas que puedan surgir, es relativamente sencillo. Además, todas las piezas están disponibles en Internet en formato *.stl* por lo que es posible imprimir dichas piezas en caso de deterioro.

Figura 25 - Impresora 3D Prusa i3 MKII

4.2. Software

4.2.1. Horus (Fig.26)

HORUS
3D SCANNING FOR EVERYONE

'Horus' es el programa *software* principal del escáner. Se encarga de realizar todas las tareas necesarias para escanear. Es de código abierto y está escrito en el lenguaje de programación 'Python'.

Figura 26 - Horus

El programa está dividido en tres *workbench* o vistas. En la primera vista, *"Scanning workbench"*, podemos apretar el botón de *"play"* y el programa se encarga de escanear el objeto. En la segunda vista, *"Control workbench"*, podemos controlar y modificar diferentes parámetros de los diferentes elementos presentes. Aquí podemos interactuar directamente con elementos como el motor o los láseres enviando comandos 'G-code'. Por último, en la tercera vista, *"Calibration workbench"*, podemos calibrar manualmente el escáner sin necesidad de ejecutar el *"wizard"* que se ofrece.

4.2.2. MeshLab (Fig.27)

Figura 27 - Meshlab

'MeshLab' es un programa de procesamiento geométrico que se utiliza principalmente para tratar mallas, como por ejemplo, nubes de puntos que se obtienen al escanear un objeto para conseguir un objeto más nítido.

'MeshLab' está escrito en 'C++' y es de código abierto, por lo que su uso es gratuito y los usuarios pueden modificar el código fuente para mejorar el programa. También cabe la posibilidad de utilizar el programa como una biblioteca en otros programas.

4.2.3. Blender (Fig.28)

Figura 28 - Blender

'Blender' es un programa de herramientas *software* para gráficos de ordenador 3D. Permite crear y modificar diseños en 3D con una amplia serie de opciones, desde modelado 3D, pasando por simulaciones de fluidos y humo, hasta animaciones completas. Está escrito en los lenguajes de programación 'C', 'C++' y 'Python'.

4.2.4. Python (Fig.29)

Figura 29 - Python

'Python' es un lenguaje de programación interpretado y de alto nivel, que además soporta orientación a objetos, programación por procedimientos y programación funcional. El lenguaje es de código abierto y actualmente hay una gran cantidad de usuarios que utilizan y modifican el lenguaje para crear nuevas bibliotecas para ser utilizadas junto a 'Python'.

El programa principal del escáner, 'Horus', está completamente escrito en 'Python', y como se ha comentado anteriormente, al ser de código abierto, se ha podido modificar el código para añadir posteriores mejoras tanto a la apariencia del programa como al funcionamiento.

4.2.5. FreeCAD (Fig.30)

Figura 30 - FreeCAD

'FreeCAD' es una aplicación gratuita y de código abierto de diseño asistido por computador en tres dimensiones. Está escrito en 'Python' y 'C++', y al ser de código abierto, es posible modificar el programa. 'FreeCAD' soporta diferentes formatos, entre los cuales se encuentra el formato *.stl*, que es el formato que se necesita para poder crear un objeto con 'G-code', que es el formato utilizado por las impresora 3D para poder imprimir objetos.

Este programa se utilizó para rediseñar uno de los diseños del escáner y para crear otro diseño que no aparece en la documentación de la web del escáner.

4.2.6. Slic3r (Fig.31)

Figura 31 - Slic3r

'Slic3r' es un *software* 3D gratuito que se utiliza para generar archivos 'G-code' a partir de ficheros 'CAD', con formato *.stl* o *.obj*. Esto se realiza para generar un fichero que la impresora 3D sea capaz de entender.

'Slic3r' se utilizó para generar los ficheros 'G-code' de todas las piezas con formato *.stl* que se encuentran en la web del autor del escáner 'Ciclop', además de los diseños diseñados y rediseñados.

4.2.7. Grbl (Fig.32)

Figura 32 - Grbl

'Grbl' es una alternativa de bajo coste, de alto rendimiento, para controlar motores paso a paso. Es compatible con 'Arduino Uno'. Está escrito en 'C' y permite mantener pulsos de 30kHz de forma estable.

4.2.8. G-Code (Fig.33)

Figura 33 - G-Code

'G-code' es un lenguaje de programación de control numérico, es el más utilizado para controlar máquinas automatizadas.

Con este lenguaje, se le indica a las máquinas computerizadas las acciones que deben de llevar a cabo, como moverse a cierta posición, a que velocidad se deben de mover o que partes mover. Los comandos de 'G-code' también se pueden llamar códigos preparatorios, y sería cualquier comando que empiece por una G, por ejemplo G01, G28 o G99. El funcionamiento de un programa con

‘G-code’ se resume en que el programa le envía una serie de comandos a la máquina la cual procede a realizar las acciones indicadas.

En este TFG, ‘G-code’ se ha utilizado para indicarle al motor cuando moverse y a que velocidad, además de indicarle a los láseres, y posteriormente al *LED*, cuando encenderse y apagarse.

5. Desarrollo de la solución

En este apartado se va a explicar el desarrollo del escáner, desde la impresión de las piezas, hasta la modificación final del objeto escaneado con un programa de edición digital. Además de mencionar y explicar las mejoras realizadas.

5.1. Desarrollo

El primer paso realizada fue acceder a la web del escáner 'Ciclop' de **'BQ'** para conseguir los diseños de las piezas de la estructura del escáner que se deben de imprimir y hacer una lista de todos los materiales necesarios para montarlo. Una vez obtenidos los diseños y los demás materiales (láseres, tornillería, motor, *shield*...) se procede a imprimir las piezas que forman la estructura del escáner con una impresora 3D.

Primeramente, se descargaron los diseños de las piezas y se utilizó el programa 'Slic3r' para obtener los ficheros de las piezas con formato *.gcode*. Se hicieron varias pruebas para encontrar la configuración correcta para estas piezas y el escáner utilizado (Fig.34).

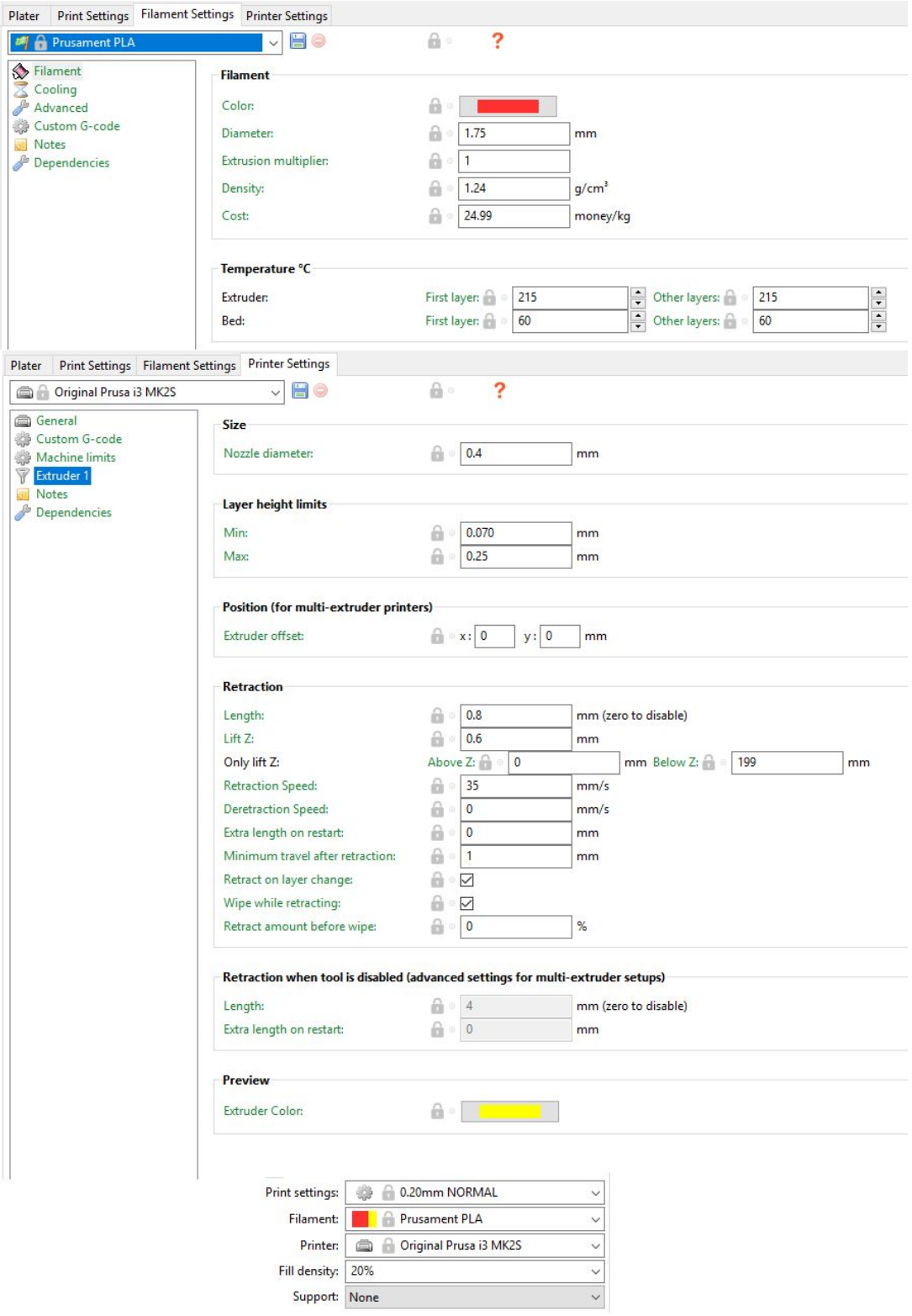


Figura 34 - Configuración Slic3r

Una vez configurado, se procedió a imprimir todos las piezas del escáner. Durante la impresión de las piezas, surgieron varios problemas que ralentizaron el progreso del desarrollo. Con una de las piezas, sobre la mitad de la pieza, el filamento de la impresora no consiguió adherirse al resto de la pieza y como

resultado, el resto de la pieza no finalizó y se desecho el resultado final. Para corregir esto, hubo que cambiar la velocidad a la cual la impresora expulsa aire para enfriar el filamento.

Con otra pieza (Fig.35), sobre la base de esta, la impresora imprimió fuera de la pieza creando la sensación de que el extrusor{27} fue movido por fuentes externas. Se volvió a intentar imprimir la misma pieza por si ese fuera el caso, pero se imprimió de forma parecida, por lo que se dedujo que el problema estaba en la generación de 'G-code' del diseño. Se probó a generar el 'G-code' otra vez, y esta vez finalmente se imprimió correctamente.

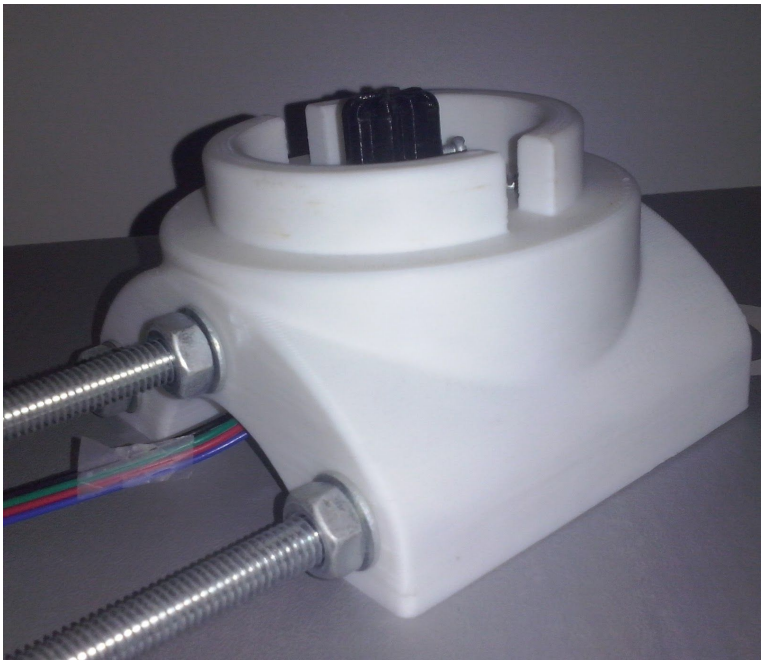


Figura 35 - Pieza 3D correcta

La pieza más grande se imprimió con una impresora 'Prusa i3 MKIII', que es una versión mejorada de la que se utilizó para las otras piezas, ya que se trataba de una pieza grande y con bastantes huecos y se temía que la impresora actual no consiguiera imprimir la pieza correctamente.

La última pieza que dió problemas a la hora de imprimir, fue la pieza que cubre la circuitería de la parte trasera. Al ser la pieza bastante estrecha, la impresora tenía problemas para imprimirla ya que únicamente podía imprimir dos líneas de filamento, como se puede apreciar en la figura 36. La solución fue aumentar la distancia entre ambas líneas de filamentos para que se imprima mejor, pero finalmente se decidió no incluir la pieza ya que con las mejoras que se incluyeron al final, la pieza no resultaba adecuada.

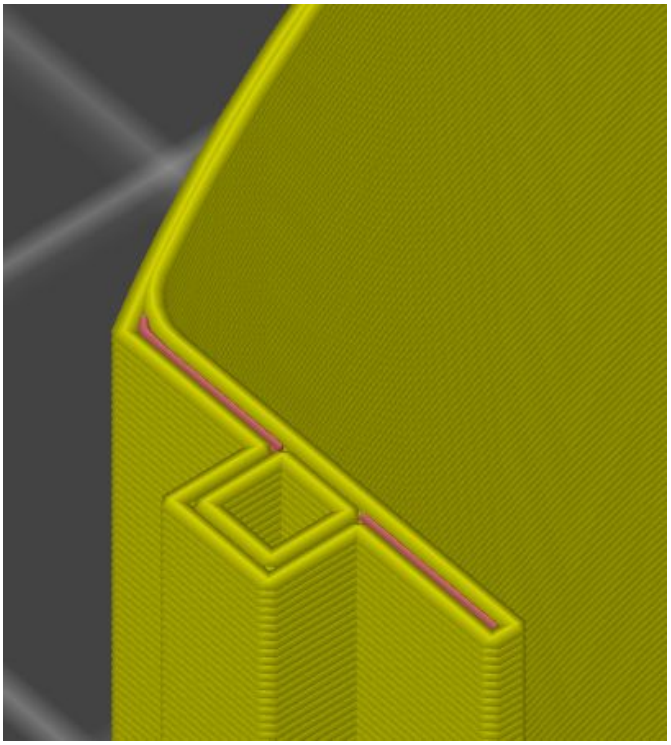


Figura 36 - Dos líneas de filamento

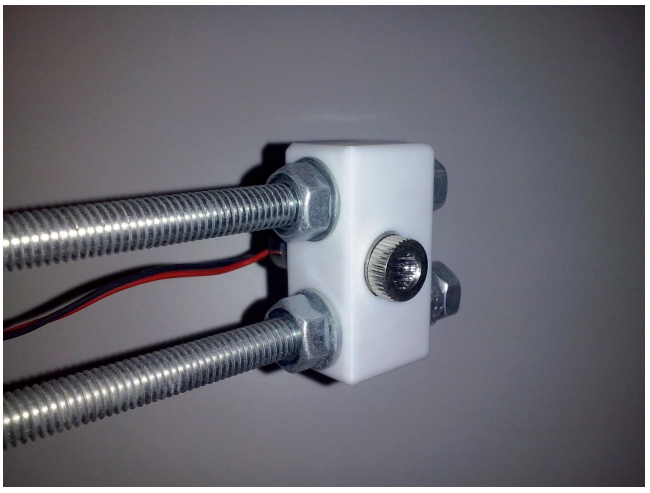


Figura 37 - Pieza que sujeta los láseres

Mientras se llevaba a cabo la impresión de las piezas, se probó a introducir los láseres en las piezas que los sujetan (Fig.37). Se apreció que los láseres no encajaban en las piezas, por lo que no se podían utilizar. La solución fue volver a rediseñar la pieza utilizando el programa 'FreeCAD'. Se diseñó la misma pieza desde cero, ya que el autor solamente incluyó las piezas con formato *.stl*, aumentando ligeramente los huecos para que los láseres quepan. Además, en los diseños aportados, no se encontraba la base para la base giratoria del escáner, con lo que se requirió diseñar e imprimir la pieza. Se comenzó obteniendo la posición de los huecos de las tuercas partiendo del soporte de la base que si que

se puede encontrar en la web con los demás diseños, se buscó un punto intermedio entre los tres huecos y finalmente se creó una base con una circunferencia de 200 mm (Fig.38 y Fig.39).

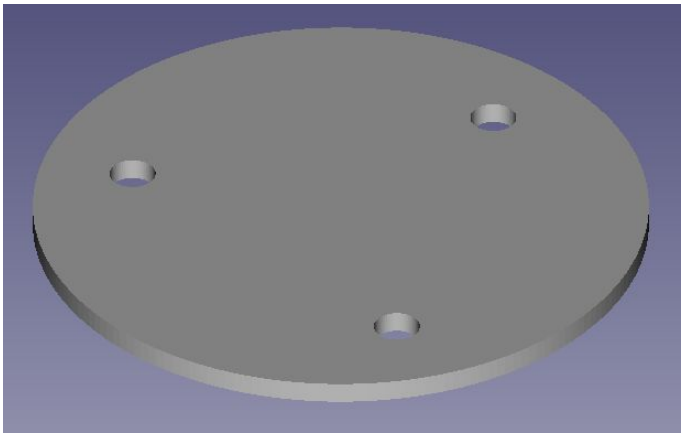


Figura 38 - Base hecha en FreeCAD

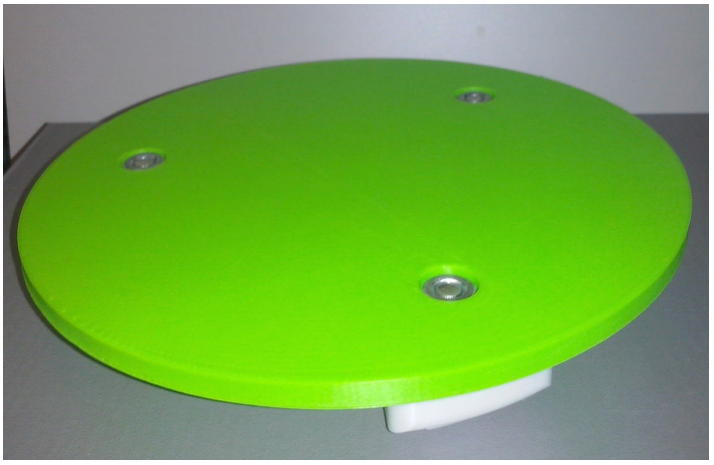


Figura 39 - Base resultante

A sus vez, se empezó a cortar las varillas roscadas de un metro a las especificaciones marcadas por el autor en la web.

Al finalizar la impresión de las piezas y el serrado de las varillas, se procedió a montar el escáner con todos los componentes del escáner. Esta parte fue relativamente fácil y no hubo ningún problema. En cuanto a la circuitería, cuando se intentó montar el *shield* sobre la placa de 'Arduino Uno', el conector de alimentación está montado en la misma posición que la entrada del cable que conecta la placa con el 'PC', por lo que los pines del *shield* sobresalen por la otra parte y no se podía montar correctamente sobre la placa 'Arduino'. Se procedió a cortar lo máximo posible de esos pines para poder montar el *shield* sobre la placa (Fig.40).

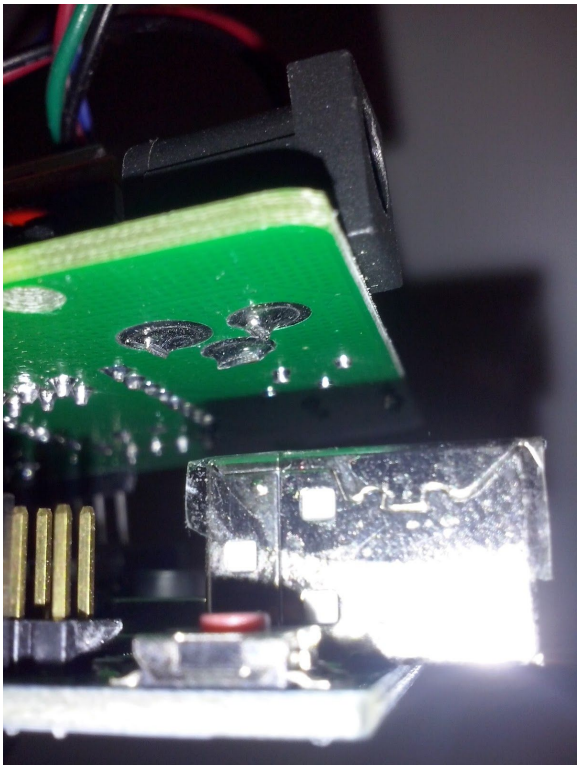


Figura 40 - Pines cortados

Posteriormente se procedió a instalar el *software* necesario para el funcionamiento del escáner. Comenzando por instalar el programa 'Horus', se siguió las rutas indicadas en la web hasta llegar al enlace desde donde se descarga el archivo ejecutable para la instalación del programa[18]. En otra página de la misma web, se puede encontrar un archivo *.hex* al final de la misma que nos servirá más adelante[19]. Luego, según lo recomendado por el autor, se descargó e instaló los *drivers* actualizados de la cámara 'Logitech'. Por último se instaló 'Grbl', para ello se tuvo que buscar un poco para encontrar las instrucciones para compilar 'Grbl'[20]. Se siguieron las instrucciones marcadas y se compiló correctamente el *software*.

Con todo lo necesario descargado e instalado, se procede a ejecutar el programa 'Horus'. Lo primero que deberíamos de hacer, es usar el *"wizard"* que viene incluido en el programa para calibrar el escáner y tener un escaneo óptimo. Pero antes de proceder con este paso, es necesario acceder a las preferencias para poder configurar las diferentes opciones. Este menú no se encuentra en el *"wizard"* y se debe de realizar antes de ejecutarlo. Las preferencias se encuentran en el menú de *"Edit"* y seleccionamos el submenú *"Preferences"* (Fig.41). Aquí podemos modificar opciones como la cámara que se va a utilizar, ya que si se tiene otra cámara conectada, es probable que el programa utilice esta cámara, además de indicarle al programa que placa debe de seleccionar y el

firmware que debemos de subir a la placa, es aquí donde necesitaremos el *firmware* anteriormente descargado.

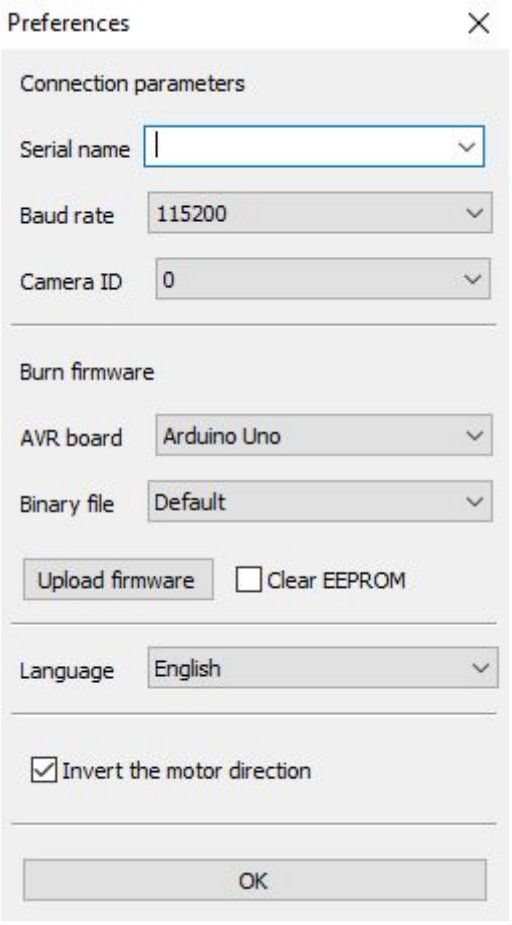


Figura 41 - Preferencias

Una vez configurado los parámetros, ejecutamos el *"wizard"* y lo primero que vemos es que se requiere de un patrón. Este patrón no se encuentra en la web junto a los demás componentes y se tuvo que buscar. Se encontró una web[21] donde se puede descargar e imprimir el patrón, que es un patrón típico con forma de tablero de ajedrez que se utiliza para calibrar.

Ahora, con el patrón montado en la plataforma y este en la base del escáner, podemos proceder a ejecutar el *"wizard"* y calibrar el escáner.

A continuación se hizo una prueba rápida para comprobar que todo funcionaba correctamente. Se notó que el motor no funcionaba correctamente, hacia mucho ruido y el movimiento de cada paso era muy brusco. El problema se hallaba en que el driver del motor no estaba configurado para funcionar a 16 pasos. Los zócalos para el *driver* del motor del *shield* tiene 6 pines, los cuales modifican el funcionamiento del motor, y es necesario poner unos puentes para habilitar la opción del funcionamiento a 16 pasos.

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

Figura 42 - Pines driver paso a paso

Como se puede apreciar en la figura 42, se requiere que todos los pines se encuentran en *High* para poder tener la opción de funcionamiento a 16 pasos.

Con todos estos problemas resultos, se procede a realizar una prueba de escaneado. Para esto, después de realizar la calibración del escáner, el programa empieza por encender uno de los láseres, captura la imágen del objeto y determina la textura del mismo. A continuación hace lo mismo con el otro láser, y luego hace una captura del objeto sin los láseres para capturar el color del objeto. Luego rota la plataforma los grados que le hemos indicado, y vuelve a repetir el proceso anterior hasta que complete una vuelta de 360º.

Se hizo una prueba y se consiguió el siguiente objeto (Fig.43):

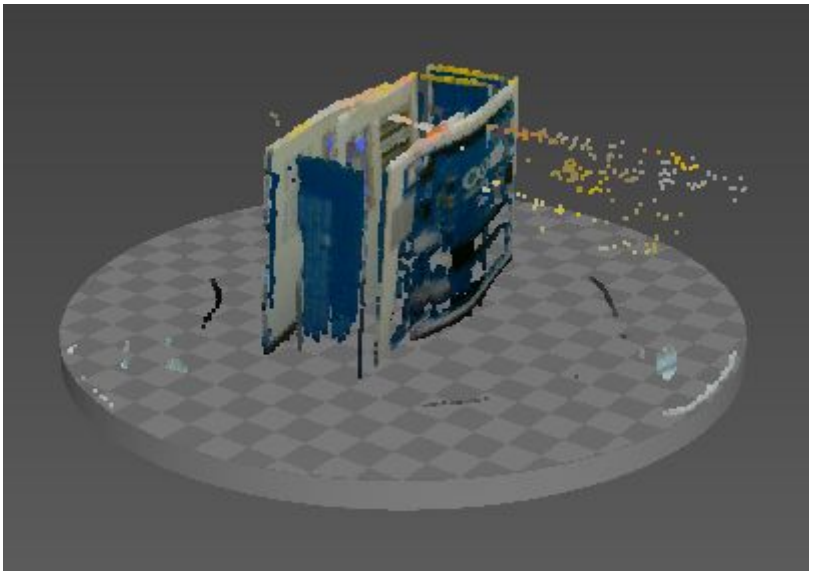


Figura 43 - Resultado de escaneado 1

Por último, se ajustaron todas las opciones disponibles y se procedió a hacer otra prueba, teniendo como resultado el siguiente objeto (Fig.44):

Figura 44 - Resultado de escaneado 2

Como se puede apreciar, la mala iluminación da como resultado que parte del objeto no se consiga capturar los puntos necesarios para formar la nube de puntos en esa parte.

5.2. Mejoras realizadas

Como resultado de la segunda prueba de escaneado, se dedujo que la iluminación es un factor importante en cuanto a conseguir un buen escaneado. Por lo cual, la primera mejora que se realizó fue proveer al escáner de una cáscara con el interior de un color uniforme, blanco en este caso, para conseguir un entorno oscuro. Con esto conseguimos que la cámara consiga los datos de los láseres con más precisión. Ahora faltaría incorporar una fuente de luz para poder conseguir los datos de la textura del objeto. Para esto se integró una iluminación *LED* a la estructura con un relé para poder controlar el encendido y apagado del *LED*. Al tener cuatro entradas para controlar los láseres, ya se están utilizando dos, así que se escogió la tercera entrada para conectar una de las entradas del relé, la que se encarga de indicar cuando debe de encenderse y apagarse. La alimentación y la masa se conectó a los pines de los conectores de sensor analógico que tiene integrado el *shield*. El resultado de estas conexiones se puede ver en la figura 45 y figura 46:

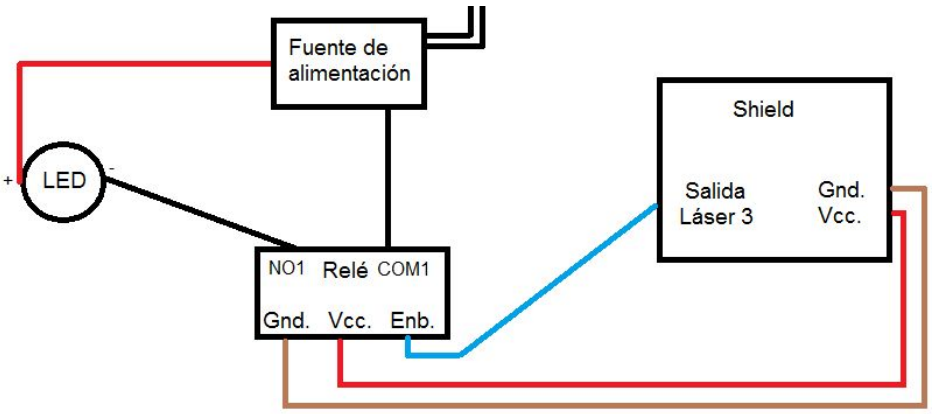


Figura 45 - Diagrama de conexiones

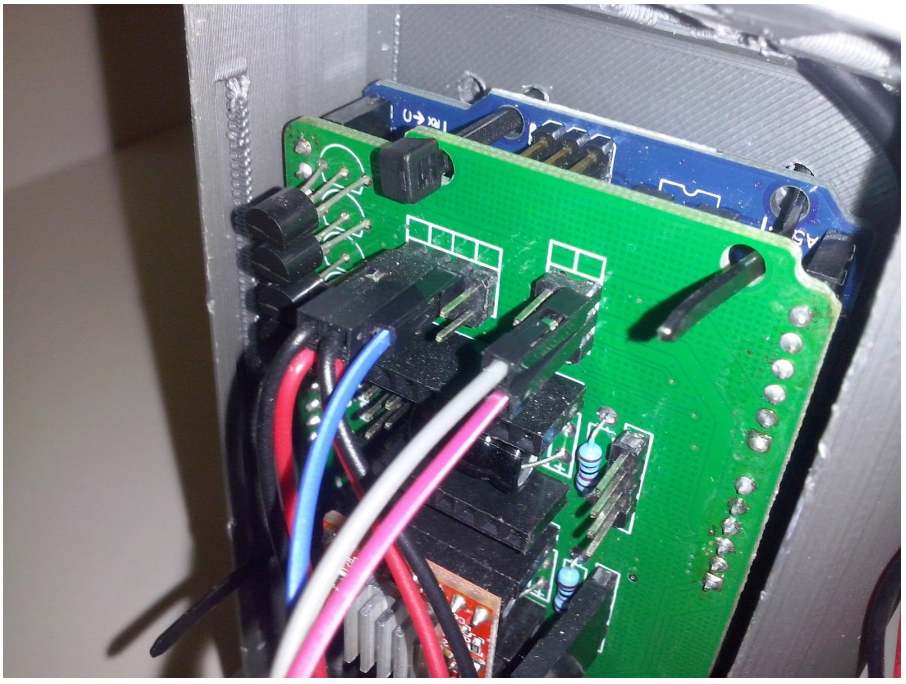


Figura 46 - Fotografía de las conexiones

A continuación se modificó el código del programa 'Horus' para indicarle cuándo debe de encender y apagar el *LED*. Primero se empieza por estudiar y entender el código del programa. Después de estudiar el código que se encuentra en la web de 'Github', se modificó el código de esa web, ya que debería de ser el mismo código que el que se encuentra en el programa final, pero cuando se intentó cargar de nuevo el programa, se produjo un error porque no se encontraban ciertas funciones. Resultó que el código de la web de 'Github' era diferente al del programa que se descargó, esto no es mucho problema pero si un inconveniente ya que se tuvo que volver a estudiar el código, esta vez del programa descargado. Muchas funciones no varían, pero la estructura del código e incluso clases enteras, cambian drásticamente.

Una vez entendido el código, se procedió a modificar dicho código.

Se buscó las funciones que activan y apagan los láseres y se creó una nueva función para activar y apagar el led mandando comandos 'G-code' (Fig. 47).

```
def setLEDOn(self):
    return self._sendCommand("M71T3")

def setLEDOff(self):
    return self._sendCommand("M70T3")
```

Figura 47 - Funciones añadidas

A continuación se buscó donde se activan los láseres en el proceso de escaneado. Se modificó para que el *LED* esté encendido en todo momento y se apague cuando la cámara inicie el proceso de captura de los láseres para conseguir los datos de la geometría del objeto. De esta forma se evita que el retardo que tiene la cámara a la hora de capturar una imagen nueva, afecte a al proceso total.

```
#-- Capture images
imgRaw = self.driver.camera.captureImage(flush=True, flushValue=flush)
self.driver.board.setLEDOff()                                          #TFG_cambio
time.sleep(0.25)

if self.pcg.useLeftLaser:
    self.driver.board.setLeftLaserOn()
    self.driver.board.setRightLaserOff()
    imgLaserLeft = self.driver.camera.captureImage(flush=True, flushValue=flush)
else:
    imgLaserLeft = None

if self.pcg.useRightLaser:
    self.driver.board.setRightLaserOn()
    self.driver.board.setLeftLaserOff()
    imgLaserRight = self.driver.camera.captureImage(flush=True, flushValue=flush)
else:
    imgLaserRight = None

time.sleep(0.25)
self.driver.board.setLEDOn()                                           #TFG_cambio
```

Figura 48 - Modificación encendido LED

Como se puede apreciar en la figura 48, la tercera línea apaga el *LED*, se espera 0.25 segundos, captura los datos de ambos láseres, espera otros 0.25 segundos y por último se enciende el *LED* para capturar los datos de la textura. Los tiempos de espera son necesarios ya que si se enciende el *LED* demasiado pronto, puede interferir en la captura de datos de los láseres.

Además de modificar el código para configurar el funcionamiento del *LED*, se decidió cambiar la interfaz de una de las vistas, concretamente la vista *"Control workbench"*, para añadir una opción que permite encender y apagar el *LED* con un botón. Se creó la siguiente clase para añadir un panel a la vista (Fig.49):

```
class LedControl(ExpandablePanel):          #TFG_cambio
    """"""
    def __init__(self, parent):
        """"""
        ExpandablePanel.__init__(self, parent, _("LED Control"), hasUndo=False, hasRestore=False)

        self.driver = Driver.Instance()

        self.clearSections()
        section = self.createSection('led_control')
        section.addItem(ToggleButton, 'led_button')

    def updateCallbacks(self):
        section = self.sections['led_control']
        section.updateCallback('led_button', (self.driver.board.setLEDOn, self.driver.board.setLEDOff))
```

Figura 49 - Función botón LED

Además de añadir varias funciones para que funcione todo correctamente (Fig.50 y Fig.51):

```
self.controls.addPanel('camera_control', CameraControl(self.controls))
self.controls.addPanel('laser_control', LaserControl(self.controls))
self.controls.addPanel('led_control', LedControl(self.controls))          #TFG_cambio
self.controls.addPanel('ldr_value', LDRControl(self.controls))
self.controls.addPanel('motor_control', MotorControl(self.controls))
self.controls.addPanel('gcode_control', GcodeControl(self.controls))
```

Figura 50 - Panel de LED

```
setting('left_button', '', str, 'basic', _('Left'), False)
setting('right_button', '', str, 'basic', _('Right'), False)
setting('led_button', '', str, 'basic', _('LED'), False)          #TFG_cambio
setting('move_button', '', str, 'basic', _('Move'), False)
setting('enable_button', '', str, 'basic', _('Enable'), False)
setting('gcode_gui', '', str, 'advanced', _('Send'), False)
setting('ldr_value', '', str, 'advanced', _('Send'), False)
```

Figura 51 - elemento añadido para que reconozca el panel

5.3. Ejemplo de escaneado

Para empezar a escanear, conectamos la cámara y la placa 'Arduino Uno' al PC y conectamos la fuente de alimentación. Accedemos al programa y nos dirigimos a la pestaña de ***"Preferences"*** y configuramos los diferentes parámetros, además de cargar el *firmware*. Una vez configurado, seleccionamos ***"File > ***"*** y nos aparecerá el *wizard* para la calibración del escáner. Seguimos los pasos y calibramos el escáner como se muestra en las figuras 52, 53 y 54.

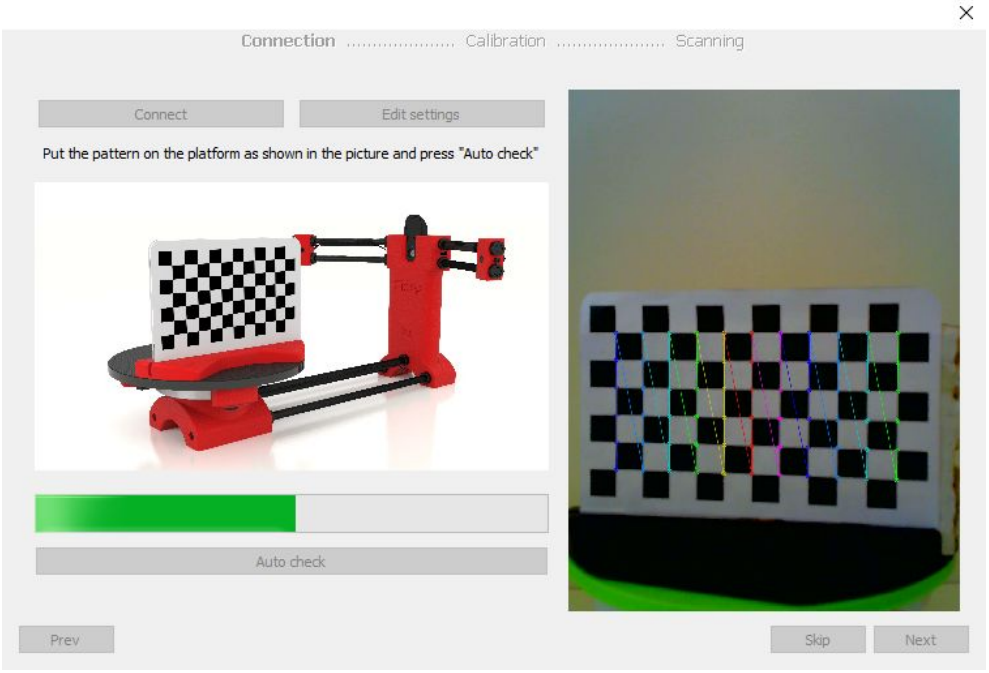


Figura 52 - Imagen de calibración

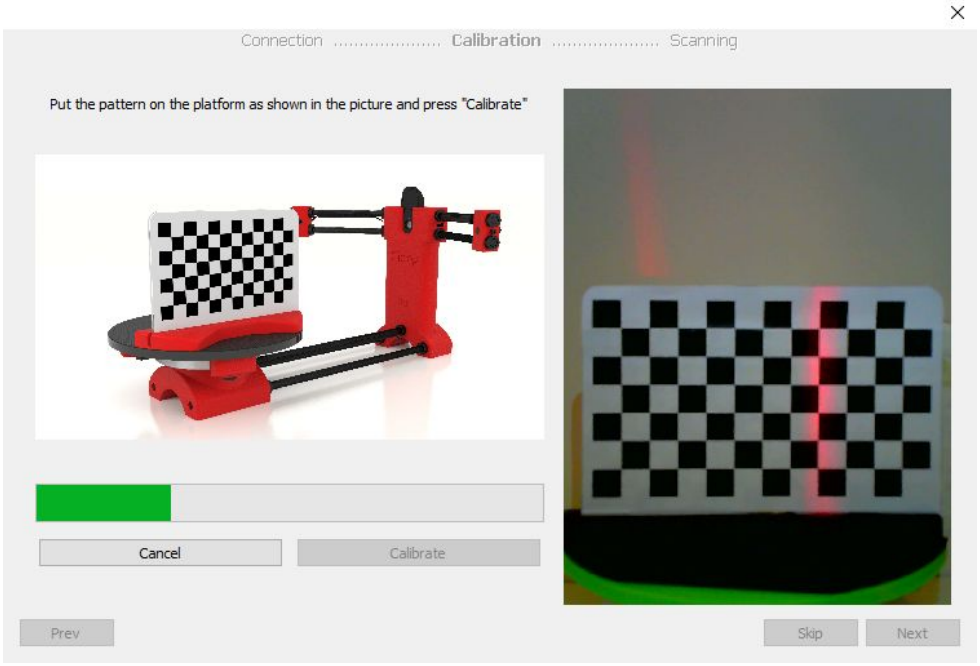


Figura 53 - Imagen de calibración

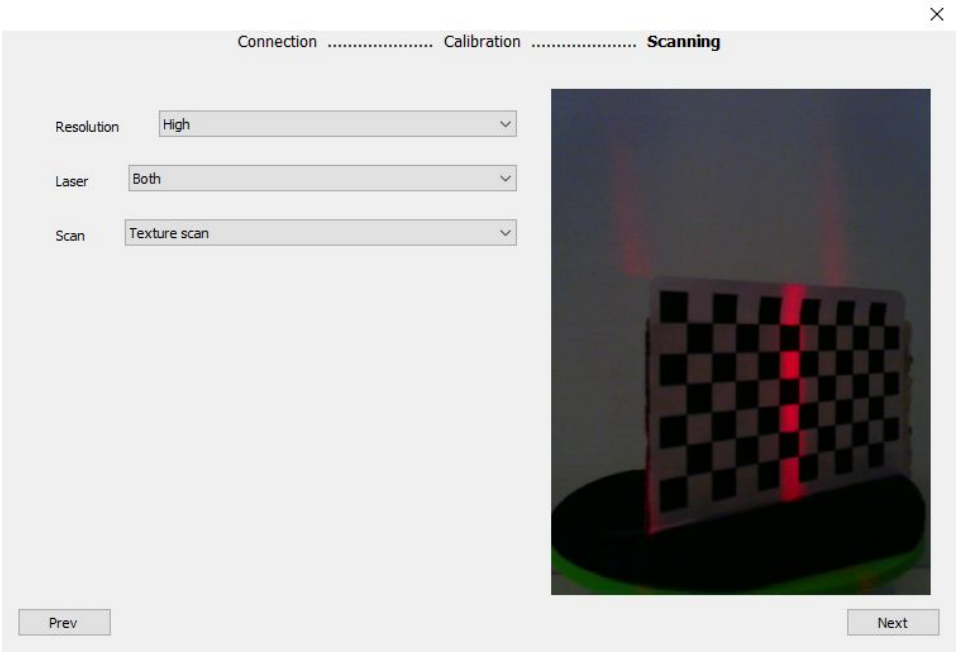


Figura 54 - Imagen de calibración

Ahora procedemos a escanear un objeto que hayamos colocado sobre la base giratoria. Seleccionamos ***"Scanning workbench"*** en la parte superior derecha (Fig.55), y a continuación apretamos sobre el icono del *play*, en la parte superior izquierda (Fig.56).

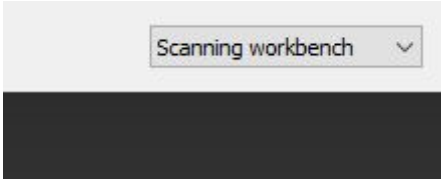


Figura 55 - Scanning workbench

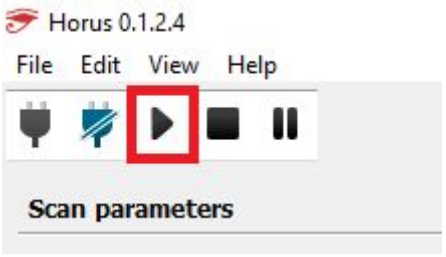


Figura 56 - Icono de play

Ahora solo nos queda esperar hasta que el proceso de escaneado termine y podremos guardar la nube de puntos obtenida seleccionando ***"File > Save model"***. Solo nos queda modificar la nube de puntos.

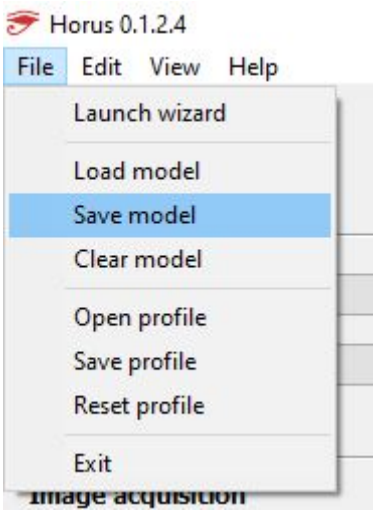


Figura 57 - Guardar modelo

5.4. Modificación de la nube de puntos

Como paso final, después de escanear el objeto y conseguir la nube de puntos, utilizaremos el programa 'MeshLab' para modificar la nube de puntos para conseguir un resultado final óptimo.

Primeramente abrimos el programa e importamos el fichero que contiene la nube de puntos **(*"File > Import Mesh..."*)**(Fig.58).

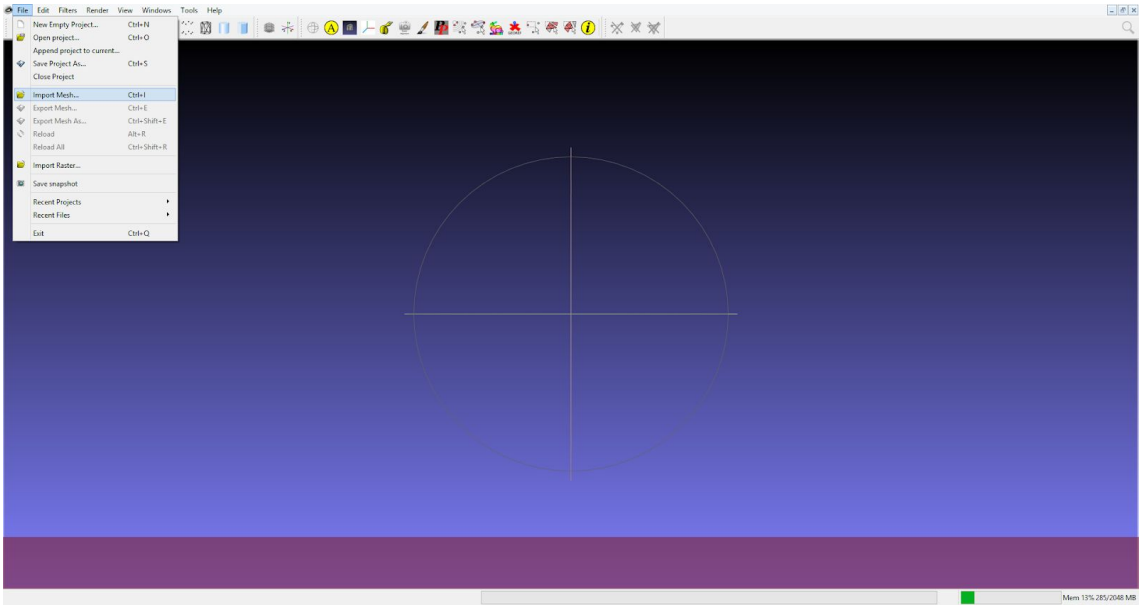


Figura 58 - Import Mesh

Después nos dirigimos a ***"Filters > Normals, Curvatures and Orientation"*** y escogemos la opción ***"Compute normals for point sets"***(Fig.59). Esto lo que nos permite es capturar el grupo de puntos de la nube

para crear planos, que a su vez, estos planos se usan para crear las normales. En la ventana que nos aparece, en el campo de ***"Neighbour num"***, que se refiere al número de vecinos que se usarán para calcular las normales del objeto, le indicamos cuantos vecinos queremos, en este caso, pondremos 15, lo demás lo dejamos por defecto, y apretamos ***"Apply"***. Esto puede tardar un rato.

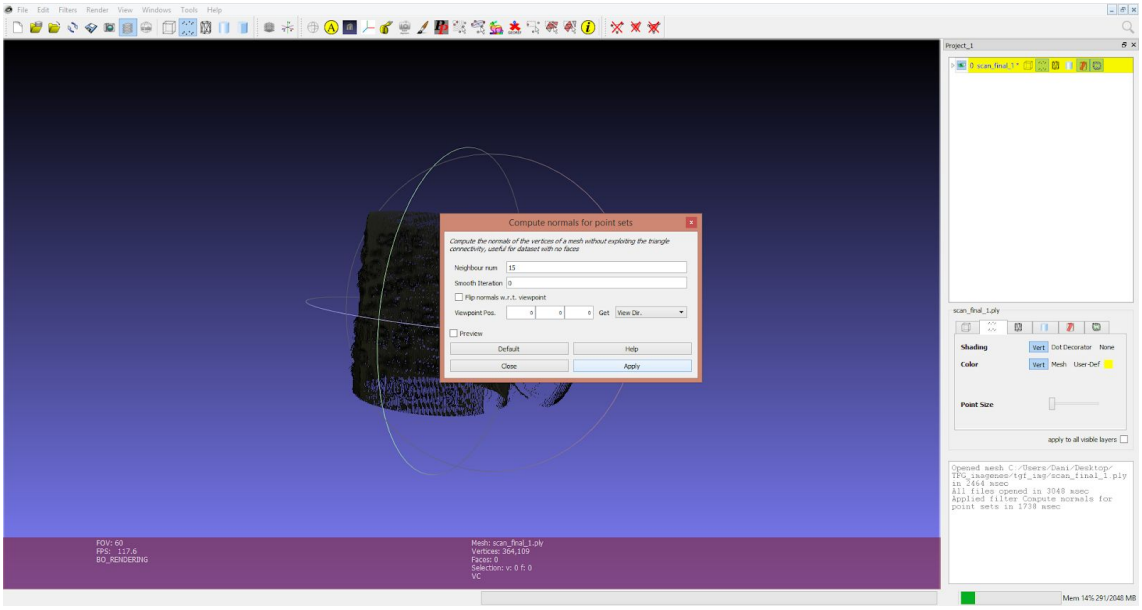


Figura 59 - Compute normals for point set

Cerramos la ventana anterior y ahora tendremos que mallar la nube de puntos, para eso escogemos la opción en **"*Filters* > *Remeshing, Simplification and Reconstruction* > *Screened Poisson Surface Reconstruction*"**. En la ventana que nos aparece, en el campo de ***"Reconstruction Depth"***(Fig.60), le indicamos un valor de 12 y apretamos ***"Apply"***.

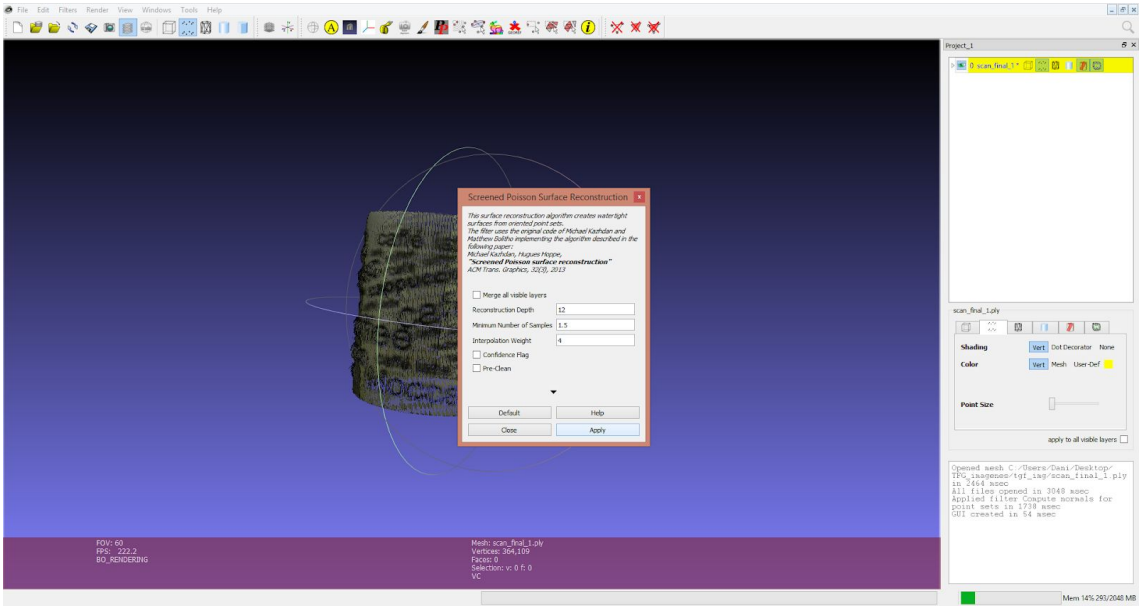


Figura 60 - Reconstruction Depth

A continuación tendremos que extraer el objeto del mallado resultante, seleccionamos ***"Filters > Selection > Select faces with edges longer than..."***. Esto nos permite elegir las caras grandes del mallado y eliminarlas. Si nos aparece una ventana, le indicamos algo como 0.15 y apretamos ***"Apply"***, pero si nos aparece un mensaje como se puede apreciar en la figura 61, podemos obviar este paso.

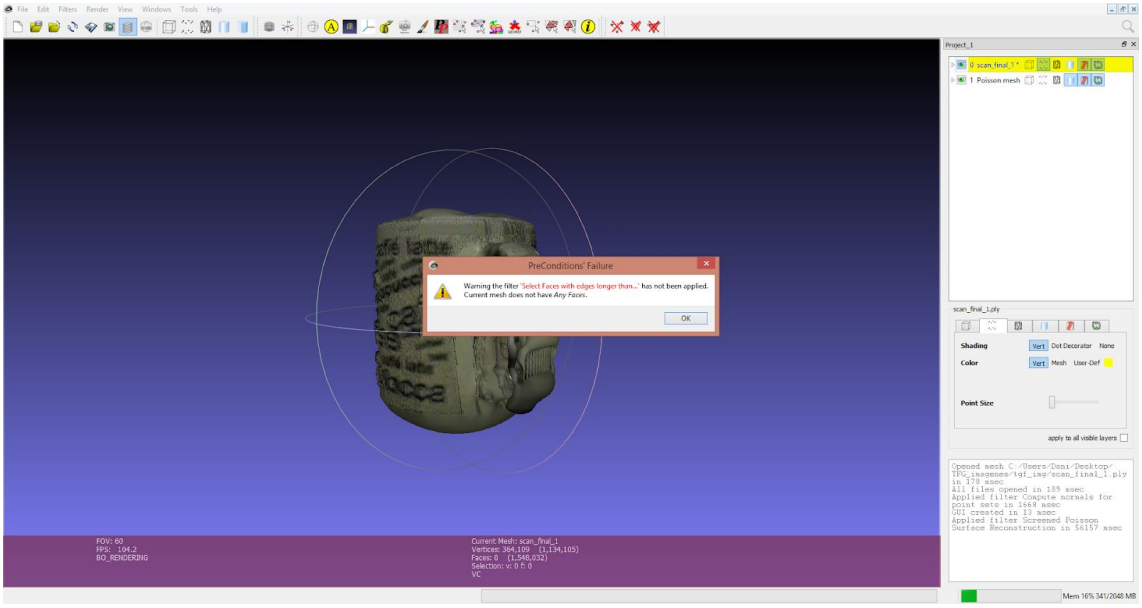


Figura 61 - Select faces with edges longer than

En la parte derecha, tendremos un menú, en el apartado ***"Color"***, seleccionamos la opción ***"Mesh"***. Esto nos permite ver la malla sin la textura (Fig.62).

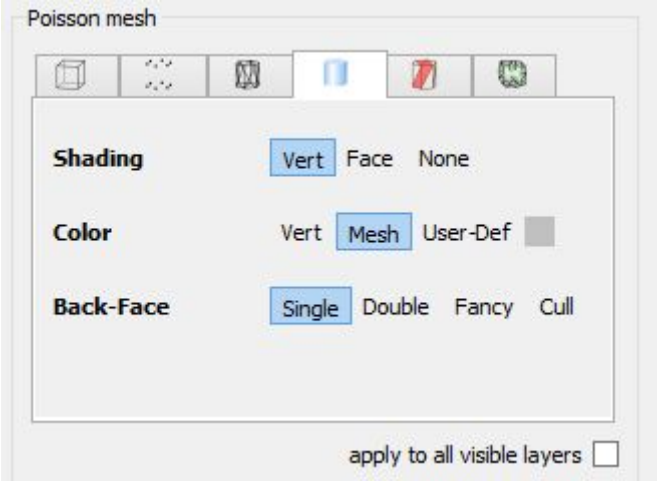


Figura 62 - Visualización de la capa

Ahora seleccionamos la herramienta ***"Select Faces in a rectangular region"*** en la parte superior(Fig.63) y eliminamos vértices o caras que no queramos en el objeto final, como por ejemplo puntos capturados que no pertenezcan al objeto.

Figura 63 - Select faces in a rectangular region

A continuación, seleccionamos ***"Filters > Smoothing, Fairing and Deformation > Taubin Smooth"*** y con la herramienta anterior, seleccionamos los puntos que queremos suavizar como se puede apreciar en la figura 64 y figura 65.

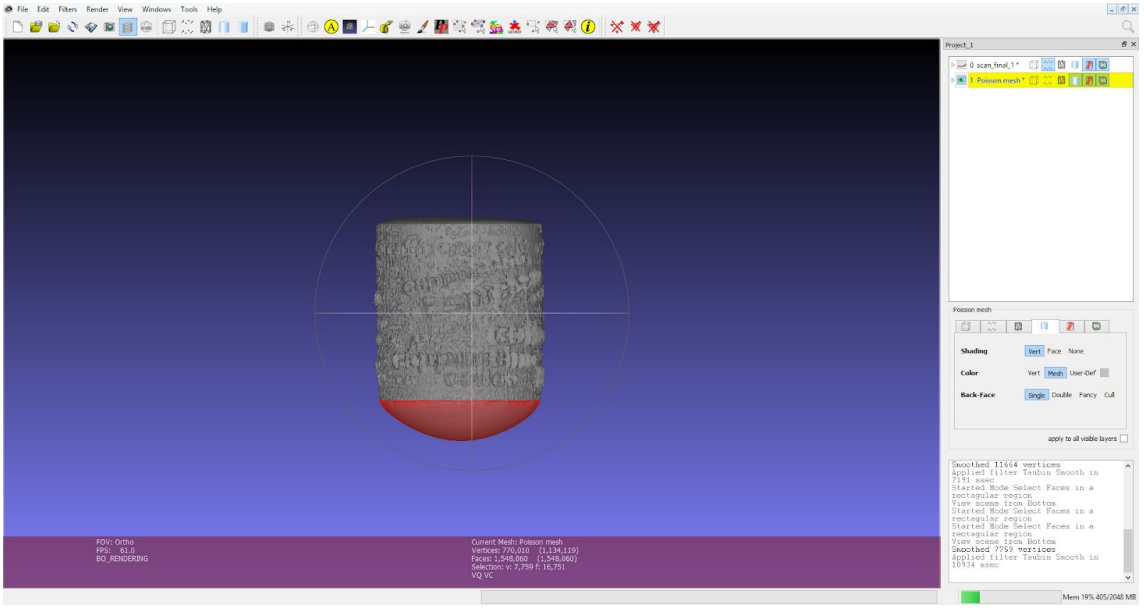


Figura 64 - Antes de suavizar

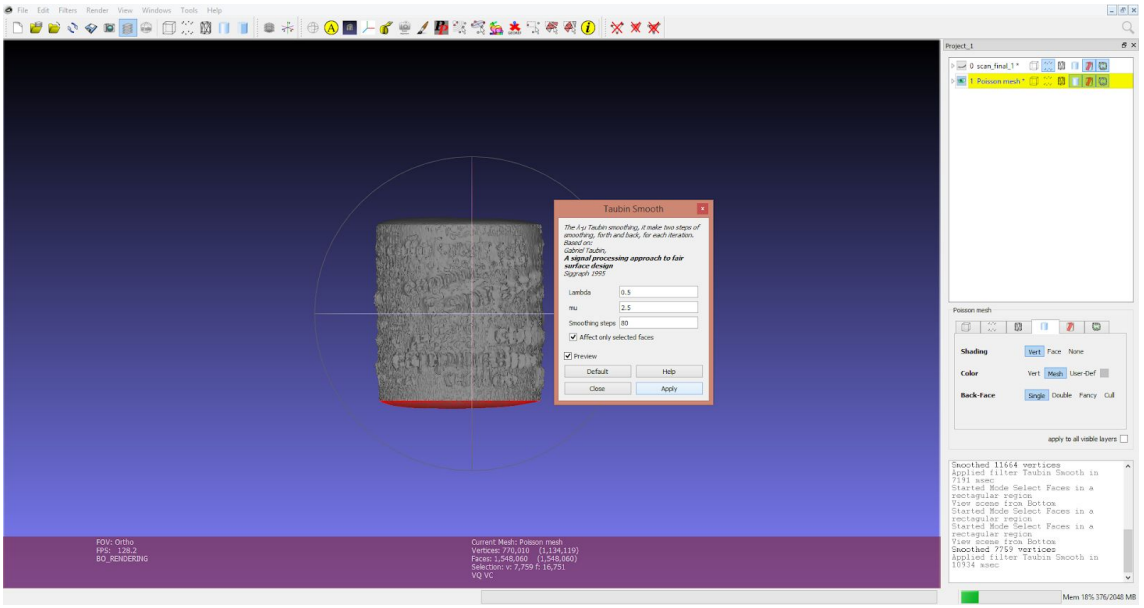


Figura 65 - Suavizado

Ahora vamos a cerrar los huecos que quedan en la malla, para eso nos dirigimos a ***"Filters > Remeshing, Simplification and Reconstruction"*** y seleccionamos ***"Close Holes"***. En la ventana que nos aparece, en el campo de ***"Max size to be closed"*** (Fig.66), el número que ponemos puede variar según los huecos que queremos cubrir y que tamaño tendran.

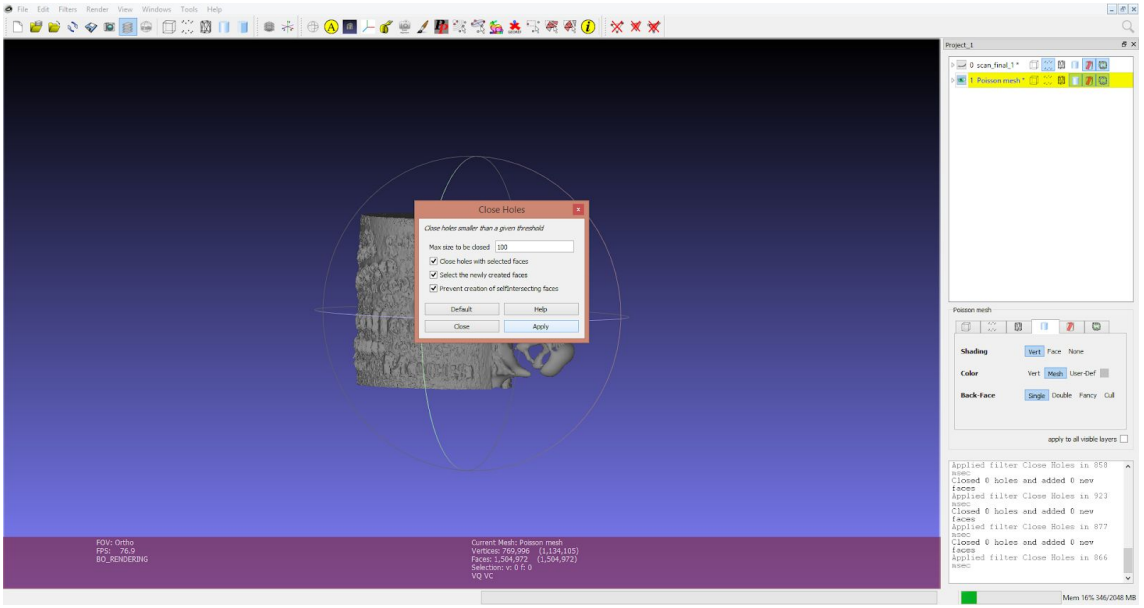


Figura 66 - Close holes

Por último, podemos comprimir el número de caras de la malla para ocupar menos y de esta forma conseguir que el archivo del objeto final tenga un peso menor. Para eso vamos a ***"Filters > Remeshing, Simplification and Reconstruction > Simplification: Quadric Edge Collapse***

Decimation" y en la ventana que nos aparece, en el campo ***"Target number of Faces"*** vamos reduciendo el número de caras, por ejemplo a la mitad (Fig.67).

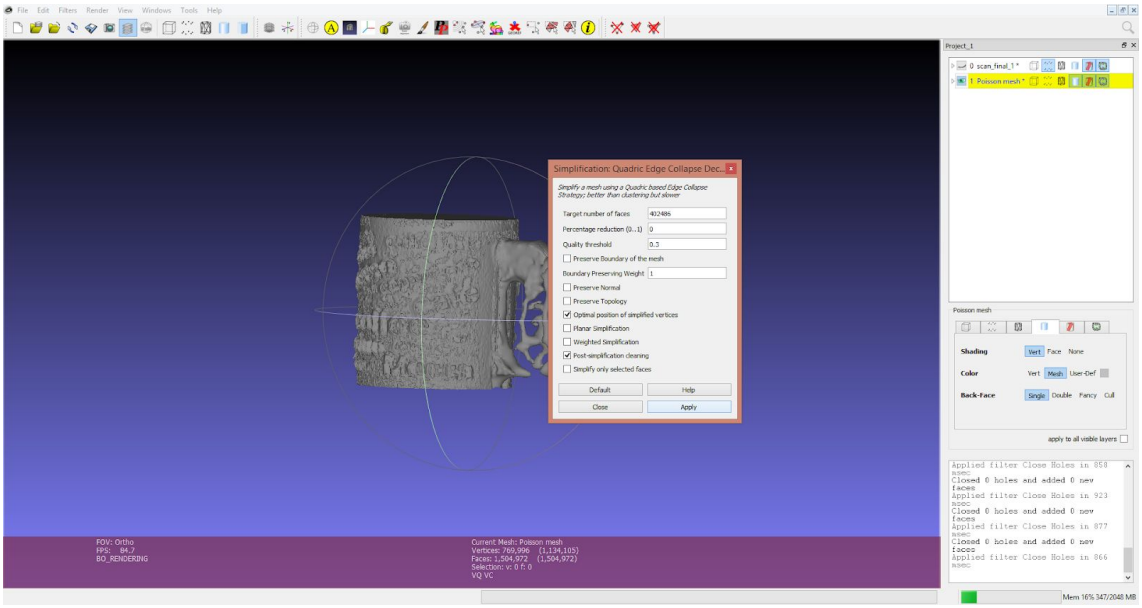


Figura 67 - Simplification

Cada vez que apliquemos este filtro, reducimos más el peso del objeto final. Se puede repetir este proceso las veces que se desee.

6. Resultados

El resultado final del escáner 3D se puede apreciar en la figura 68.

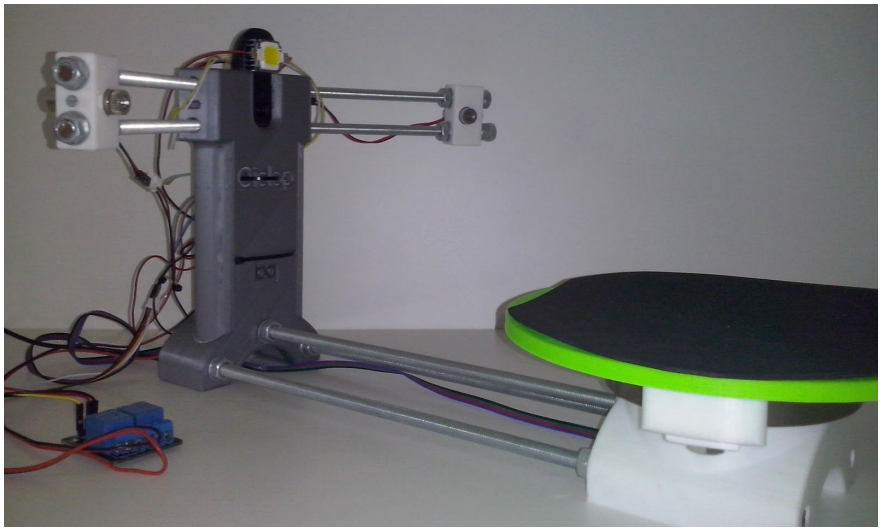


Figura 68 - Escáner 3D

Al comparar los objetos escaneados anteriormente con uno escaneado con las mejoras añadidas, podemos observar que la diferencia es bastante visible(Fig.69).

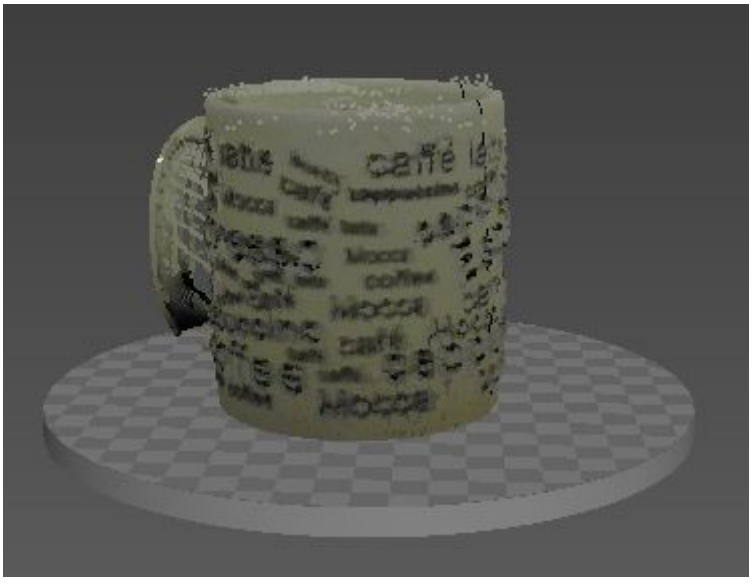


Figura 69 - Objeto escaneado final

Se puede observar como ya no hay puntos fuera del objeto ya que el escáner está dispuesto de una carcasa que limita la distancia a la que la cámara recoge los

datos. Además no se observan huecos significativos en el objeto gracias a la iluminación mejorada. Siguen existiendo algunos huecos que son producto de la técnica utilizada, ya que la técnica de triangulación tiene limitaciones que son observables cuando se capturan los puntos en superficies donde dos o más caras coinciden.

Las ventajas de este escáner, a parte de que cuesta menos que comprar el kit completo, serían que una vez desarrollado, al no pesar casi nada, se puede colocar prácticamente en cualquier lugar y conseguir el mismo resultado siempre ya que únicamente se requiere calibrar el escáner una vez y no es necesario adaptarse a los diferentes entornos de iluminación. Tampoco requiere de que el usuario controle nada del escáner una vez el proceso de escaneado ha comenzado. Además la nube de puntos resultante es prácticamente idéntica al objeto que se ha escaneado, por lo que se pueden conseguir escaneados bastante fidedignos.

Alguna desventaja, sería que el relé emite un sonido cada vez que se enciende y apaga que puede resultar molesto. La técnica utilizada tiene dificultades a la hora de capturar puntos con colores rojos, que sería el color de los láseres, y con superficies reflectantes, ya que se trata de una técnica óptica.

7. Conclusiones

Una vez terminado el proyecto y realizadas las mejoras, se observan los objetivos que se planteaban al principio y se puede afirmar que se han cumplido con todos los objetivos planteados. Se desarrolló un escáner utilizando componentes adquiridos por separado y *software* de código abierto, se profundizó en el lenguaje de programación 'Python' para poder modificar el *software* del escáner, y se realizaron mejoras al escáner aplicando diferentes métodos.

A nivel personal, he podido estudiar y utilizar el lenguaje de programación 'Python', que es un lenguaje que me interesa mucho por sus casi infinitas posibilidades de crear diferentes programas, y se puede observar en este TFG, ya que con 'Python' se ha creado tanto la tecnología que permite el escaneado, como la interfaz del programa 'Horus'. Además he podido aprender cómo es posible poder escanear un objeto en tres dimensiones, que es una de mis motivaciones a la hora de escoger el TFG.

7.1. Relación del trabajo desarrollado con los estudios cursados

En este TFG, a parte de las asignaturas obligatorias cursadas a lo largo de estos años y que han tenido un impacto en mayor o menor medida, algunas asignaturas optativas han sido claves en cuanto al desarrollo de dicho TFG, como por ejemplo, se han imprimido piezas 3D, diseñado piezas con un programa 'CAD' y utilizado el lenguaje 'G-code', correspondiente a la asignatura de "Diseño y Fabricación 3D". Se modificó código en 'Python', que correspondía a la asignatura de "Seguridad en los Sistemas Informáticos", donde se enseñó lo básico de 'Python'. Se añadió mejoras físicas por medio de adición de componentes electrónicos como es el caso del relé y el led en las mejoras realizadas además de modificar parte de la fuente de alimentación para dar potencia a los varios dispositivos, esto corresponde a la asignatura de "Diseño de interfaces y periféricos".

7.2. Posibles mejoras

Una vez desarrollado el proyecto, es posible realizar varias mejoras, entre las cuales:

- Cambiar el modo de conexión entre la placa 'Arduino' y el 'PC' por medio de un módulo 'bluetooth'. Esta mejora se pensó realizar pero en el momento de implementarlo, al utilizar una placa 'Arduino Uno' y no la

placa utilizada por el autor, es complicado añadir la funcionalidad del 'bluetooth' ya que a la placa de 'Arduino' se le añade el *shield* con lo que se bloquean las entradas de la placa y no se puede añadir el módulo. La única forma que hay de añadir esta funcionalidad es la de modificar el *shield* para dejar libre varias entradas para incluir el módulo 'bluetooth'.

- Añadir alguna forma de editar la nube de puntos automáticamente sin tener que recurrir a programas externos. Esto se podría conseguir adquiriendo el programa 'MeshLab' como biblioteca para el programa 'Horus' y creando una función que edite la nube de puntos.

8. Bibliografía

[1] Artículo sobre el aumento del uso de impresoras 3D - *Forbes*

https://www.forbes.com/sites/louiscolumbus/2018/05/30/the-state-of-3d-printing-2018/#2bfaeed67b0a

[2] Diferentes empresas que ofrecen algún servicio de impresión 3D - *el economista*

https://empresite.eleconomista.es/Actividad/IMPRESION-3D/

[3] Algunos programas de edición 3D - *All 3DP*

https://all3dp.com/1/best-free-3d-modeling-software-3d-cad-3d-design-software/

[4] Las mismas empresas pueden ofrecer servicios de diseño - *el economista*

https://empresite.eleconomista.es/Actividad/IMPRESION-3D/

[5] PDF descargable con la historia del escaneado 3D - *ResearcheGate - Mostafa A-B Ebrahim*

https://www.researchgate.net/publication/267037683_3D_LASER_SCANNERS_HISTORY_APPLICATIONS_AND_FUTURE

[6] Breve historia sobre el escaneado 3D - *Matter and Form*

https://matterandform.net/blog/a-brief-history-of-3d-scanning

[7] Artículo con referencia a las diferentes empresas de la época - *Modena*

https://www.modena.co.za/history-of-3d-scanners/

[8] Enlace a la wikipedia sobre escaneado 3D - *Wikipedia*

https://en.wikipedia.org/wiki/3D_scanning

[9] XYZprinting 3D Scanner 2.0 - *XYZprinting*

https://www.xyzprinting.com/en/product/3d-scanner-2-0

[10] Structure Sensor - *Structure*

https://structure.io/

[11] Sense 3D scanner - *3D Systems*

https://www.3dsystems.com/shop/sense

[12] Matter Form V.2 - *Matter and Form*

https://matterandform.net/store/products/MFS1V2

[13] EinScan-SE - *EinScan*

https://www.einscan.com/desktop-3d-scanners/einscan-se/

[14] MakerScanner 3D - *makerscanner*

http://www.makerscanner.com/

[15] BQ Ciclop - *BQ*

https://www.bq.com/en/support/ciclop/support-sheet

[16] FabScan Pi - *fabscan*

https://fabscan.org/

[17] Aplicaciones para dispositivos móviles para escaneado 3D - *All 3DP*

https://all3dp.com/2/5-best-3d-scanner-apps-for-your-smartphone/

[18] Instalación de Horus - *Horus*

https://horus.readthedocs.io/es/release-0.2/source/installation/windows.html

[19] Firmware de Horus - *BQ*

http://diwo.bq.com/horus-fw-released/

[20] Grbl - *Github*

https://github.com/gnea/grbl/wiki/Compiling-Grbl

[21] Patrón cuadriculado - *thingiverse*

https://www.thingiverse.com/thing:1054078

9. Glosario

(1) **3D**: Tres dimensiones.

(2) **TFG**: Trabajo de fin de grado.

(3) **Palpador**: vástago que permite medir distancias mediante el contacto con un objeto.

(4) **CMM**: *Coordinate measuring machine*, máquina de medición por coordenadas. Máquina que utiliza un palpador para recoger coordenadas de un objeto.

(5) **Distanciómetro láser**: dispositivo electrónico de medición que permite calcular la distancia desde un punto a otro.

(6) **Luz monocromática**: una luz que está formada por componentes de un solo color, tiene una sola longitud de onda.

(7) **LCD**: *Liquid Crystal Display*, pantalla de cristal líquido. Pantalla delgada formada por píxeles delante de una fuente de luz o reflectora.

(8) **Patrón sinusoidal**: Patrón que describe una onda sinusoidal.

(9) **Píxel**: Un punto físico en una imagen que se utiliza como unidad de medición.

(10) **Malla de polígonos o triángulos**: Una superficie generado por sistemas de vértices posicionados en un espacio virtual con coordenadas.

(11) **CAD**: *Computer-Aided Design*, diseño asistido por computador. Utilización de diferentes programas gráficos para crear una serie de imágenes para crear otra mayor o diferente.

(12) **NURBS**: *Non-Uniform Rational B-Spline*, B-splines racionales no uniformes. Modelo matemático utilizado en la computación gráfica para generar y representar curvas y superficies.

(13) **USB**: *Universal Serial Bus*, bus universal en serie. Bus de comunicación entre computadores, periféricos y dispositivos electrónicos.

(14) **Kickstarter**: Sitio web de financiación pública para proyectos.

(15) **USB PS3 Eye**: Cámara lanzada por SONY para su consola PlayStation.

(16) **Raspberry Pi**: Ordenador de placa reducida parecido a Arduino.

(17) **Shield**: Es un tipo de tarjeta de expansión para Arduino que permite expandir la capacidad de computación del mismo.

(18) **Bluetooth**: Dispositivo que permite la transmisión de datos entre diferentes dispositivos mediante radiofrecuencia.

(19) **FPGA**: *Field-programmable gate array*, arreglo de compuertas programable en el campo. Dispositivo programable que contiene bloques lógicos que permite crear y configurar sistemas.

(20) **Github**: Sitio web "forja" utilizado principalmente para almacenar código fuente de programas.

(21) **Arduino CNC Shield**: PCB que permite el control de motores paso a paso.

(22) **GPLv2**: *GNU General Public License*, licencia pública general de GNU. Licencia de derechos de autor para software libre y código abierto. Esta es la versión dos.

(23) **GPLv3**: Esta sería la versión tres.

(24) **PC**: *Personal computer*, ordenador personal.

(25) **LED**: *Light-Emitting Diode*, diodo emisor de luz. Fuente de luz constituida por un material semiconductor con dos terminales.

(26) **Driver**: Controlador o manejador de dispositivos. Programa informático que permite al sistema operativo interactuar con un periférico.

(27) **Extrusor**: Elemento en las impresoras 3D que permite extruir el filamento.