
**Escuela Superior de Ciencias Experimentales y Tecnología**

**Grado en Ingeniería**

**de Tecnologías Industriales**

**Trabajo de Fin de Grado**

# BANCO DE HERRAMIENTAS PARA LA GENERACIÓN AUTOMÁTICA DE PIEZAS CAD PROGRAMADO CON PYTHON PARA HARDWARE LIBRE

**David Muñoz Bernal**

**Director: Felipe Machado Sánchez**

**Curso Académico 2019/20**

Universidad
Rey Juan Carlos

**Grado en Ingeniería de Tecnologías Industriales**

# Trabajo de Fin de grado

El presente trabajo, titulado **BANCO DE HERRAMIENTAS PARA LA GENERACIÓN AUTOMÁTICA DE PIEZAS CAD PROGRAMADO CON PYTHON PARA HARDWARE LIBRE**, constituye la memoria correspondiente a la asignatura trabajo de Fin de Grado que presenta **D. David Muñoz Bernal** como parte de su formación para aspirar al Título de Graduado en Ingeniería de Tecnologías Industriales. Este trabajo ha sido realizado en la **Escuela Superior de Ciencias Experimentales y Tecnología de la Universidad Rey Juan Carlos** en el **Departamento de Matemáticas Aplicada, Ciencia e Ingeniería de Materiales y Tecnología Electrónica** bajo la dirección de **Felipe Machado Sánchez**.

Móstoles, 7 de octubre de 2020

# Agradecimientos

Me gustaría agradecer a mi tutor, Felipe Machado, por el tiempo dedicado a este proyecto y los conocimientos que me ha enseñado. Sin él habría sido imposible realizar este trabajo, así como conocer en profundidad el mundo del software libre.

También quiero agradecer a mis padres, Javier y María José, y mi hermana, Rebeca, todo el apoyo moral que me han dado durante la realización del grado.

Cabe realizar una mención especial a Natalia por acompañarme durante la realización del grado y por la amenización de todas las horas de estudio que hemos compartido. Sin ella y su paciencia para explicar los conceptos no habría sido posible comprender algunas asignaturas.

# Índice

# Lista de acrónimos

| | |
|---|---|
| ABS | Acrylonitrile Butadiene Styrene [Acrilonitrilo Butadieno Estireno] |
| BIM | Building Information Modeling [Modelado de Información de la Construcción] |
| BREP | Boundary Representation [Representación de límites] |
| CAD | Computer Aided Design [Diseño Asistido por Ordenador] |
| CPU | Central Processing Unit [Unidad de Procesamiento Central] |
| CSG | Constructive Solid Geometry [Geometría Constructiva de Sólidos] |
| DIY | Do It Yourself [Hazlo Tú Mismo] |
| DRAM | Dynamic Random Access Memory [Memoria de Acceso Aleatorio Dinámico] |
| fco | FreeCAD Object [Objeto de FreeCAD] |
| GPU | Graphics Processing Unit [Unidad de Procesamiento Gráfico] |
| GUI | Graphics User Interface [Interfaz Gráfica de Usuario] |
| OSH | Open Source Hardware [Hardware Abierto] |
| OSS | Open Source Software [Software de Código Abierto] |
| PLA | Polylactic Acid [Ácido Poliláctico] |
| RAM | Random Access Memory [Memoria de Acceso Aleatorio] |
| UML | Unified Modeling Language [Lenguaje Unificado de Modelado] |
| WB | Workbench [Banco de Herramientas] |

# Índice de figuras

# Índice de tablas

# Índice de códigos

# 1. Resumen

Este trabajo final de grado consiste en la realización de un banco de herramientas desarrollado en lenguaje *Python* bajo licencia *LGPL-3.0* que se encuentra disponible de forma gratuita en *GitHub [52]*.

El diseño de sistemas *CAD* requiere mucho tiempo y de la realización de numerosas pruebas. En busca de reducir el tiempo invertido y las pruebas a realizar se propone la utilización del banco de herramientas creado en este proyecto. Este banco de herramientas contiene modelos paramétricos 3D con fines mecatrónicos y ópticos. Estos modelos se pueden particularizar y emplear para generar el diseño de sistema mecatrónicos a base de click de ratón.

Este banco de herramientas está desarrollado para *FreeCAD*, un programa *CAD* de licencia abierta, bajo la idea de poner al alcance de cualquier persona los modelos paramétricos que se incluyen en el banco de herramientas independientemente de las características del equipo informático con el que cuente.

Estos modelos paramétricos disponibles en el banco de herramientas están basados en un *Diseño de clases* propio. Este *Diseño de clases* ofrece una serie de características a los modelos entre las cuales destacan los puntos internos. Estos son unos puntos de interés en el modelo, como por ejemplo sería el centro del orificio de un tornillo. Con estos puntos internos se simplifica la creación de modelos pudiendo hacerlo directamente en su posición final dentro del diseño de un sistema CAD.

Si las necesidades del usuario final no se ven satisfechas con los modelos incluidos en el banco de herramientas, el usuario puede generar sus propios modelos con la biblioteca de funciones que se incluye en el repositorio del proyecto. Esta biblioteca de funciones sigue el principio de modularidad y permite realizar el diseño de modelos 3D en lenguaje *Python* con una menor cantidad de líneas de código.

De cara a que el usuario final del banco de herramienta realice un uso correcto y pueda aprovechar todas las funcionalidades que se aportan, se realiza una página web con la documentación necesaria en *ReadtheDocs [55]*. En esta página web se pueden encontrar una guía de instalación, tutoriales y videotutoriales. También se encuentran la biblioteca de funciones, con todas sus funciones en detalle, y las características de los modelos.

Junto al banco de herramientas y con el fin de demostrar su utilidad, se ha diseñado, fabricado y montado el sistema posicionador lineal de un filtro óptico *(Filter Stage)*. Además, para demostrar el correcto funcionamiento de este sistema, se ha diseñado y montado un sistema de control específico. Dicho sistema de control está basado en un microcontrolador Arduino y un conjunto de sensores que actúan en el sistema *Filter Stage* por medio de un motor paso a paso. La realización del montaje físico del sistema y su sistema de control dotan al trabajo final de grado de una vertiente aplicada de la ingeniería industrial.

Adicionalmente, el sistema *Filter Stage* se puede incluir dentro de los proyectos *Open Labware*. Estos proyectos tienen el fin de crear material de laboratorio más económico y accesible que los equipos comerciales y se promueven la ciencia abierta y la reproducibilidad de experimentos en las mismas condiciones.

Para concluir, cabe destacar la publicación del banco de herramientas en el foro oficial de *FreeCAD*. Tras esta publicación se obtuvieron propuestas de mejora por parte de los usuarios del programa que habían probado el banco de herramientas. Finalmente, el banco de herramientas ha sido aceptado como banco de herramientas externo de *FreeCAD* de manera oficial siendo publicado en el listado de la página oficial del programa *[58]*.

Asimismo, se deben destacar los conocimientos necesarios de lenguaje *Python* como del módulo *PySide2,* para la creación de la interfaz de usuario, y el módulo *Sphinx,* para la creación de la página web. Por último, hay que mencionar que el proyecto, sin contar las líneas en blanco, supone un total de *67* archivos con *28.430* líneas de código en lenguaje *Python* con *21.614* líneas de comentarios en lenguaje *Python* y *471* líneas en lenguaje *reStructuredText*.

# 2. Introducción

En los últimos años ha habido un auge en el desarrollo de proyectos caseros con dispositivos electrónicos y mecánicos. La unión de estas dos ramas, electrónica y mecánica, da lugar a la mecatrónica. Al juntarse el movimiento *DIY* (*Do It Yourself*) con la mecatrónica surge el movimiento *maker*. Este movimiento, junto con el auge de la impresión 3D y la electrónica abierta como *Arduino*, promueve la divulgación del conocimiento de la tecnología y la fabricación de dicha tecnología a bajo coste *[6][7].*

Siguiendo el mismo principio de la reducción de costes empleando la impresión 3D y la electrónica abierta surge el movimiento *Open Labware [8]*. Los dos objetivos de este movimiento son *[9]*:

1. Llegar a tener cualquier material de laboratorio con un coste sumamente inferior a las alternativas comerciales.
2. Promover la ciencia abierta, es decir, la capacidad de replicar experimentos en las mismas condiciones a bajo coste.

Partiendo del movimiento *Open Labware* y la necesidad científica de reproducir los experimentos con el menor coste posible surge la idea de facilitar a cualquier persona, sin necesidad de tener conocimientos de *CAD*, a ensamblar un sistema mecatrónico.

Se deben tener en cuenta los siguientes hechos:

- Realizar un diseño en *CAD* requiere una formación y una gran inversión de tiempo, por tanto, una herramienta capaz de simplificar esta labor sería muy útil.
- *Arduino* dispone de librerías que simplifican la adición de componentes. En base a esta idea, se ha creado una librería de modelos paramétricos que se pueden añadir a cualquier diseño. Esta librería está escrita en lenguaje *Python*.
- Disponer de librerías sin un programa en el que emplearlas resta utilidad. Por ello la librería se encuentra embebida en un banco de herramientas de *FreeCAD*. De esta forma no será necesario utilizar código, es decir, no es necesario tener conocimientos de programación para emplear la librería.

## 2.1. Ámbito científico-tecnológico

Este trabajo se puede incluir dentro de campos científicos-tecnológicos muy diferentes ya que abarca informática, diseño asistido por ordenador y mecatrónica.

### 2.1.1. Mecatrónica

La mecatrónica es un campo que integra electrónica, mecánica, diseño, control y computación para crear sistemas electromecánicos. Los sistemas mecatrónicos recogen datos por medio de los “Sensores” que se envían a la “Electrónica de control” quien se encarga de mandar una señal para actuar sobre el sistema mecánico. El esquema general de un sistema mecatrónico se puede ver en la *Figura 1 [12]*.

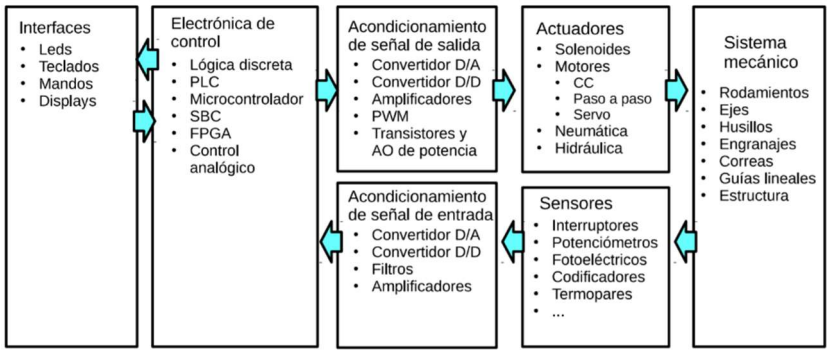


*Figura 1 - Esquema de un sistema mecatrónico [12]*

El término *mecatrónica* nace en 1969 por un ingeniero japonés que combinó los mecanismos (meca) y la electrónica (trónica). Hoy en día el término ha evolucionado y no sólo implica la unión de los mecanismos y la electrónica, sino que incluye el control y la computación como se ilustra en la *Figura 2*. Estos cuatro campos se combinan para realizar el diseño y elaboración de productos o procesos.

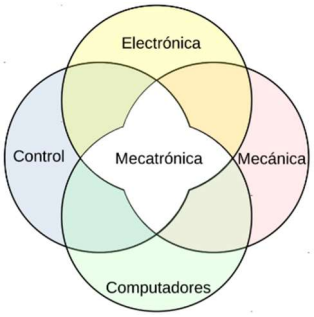


*Figura 2 – Mecatrónica [12]*

#### *2.1.1.1. Electrónica y control*

La electrónica dentro de un sistema electromecánico está compuesta por los sensores, actuadores, acondicionadores de entrada y salida de señal y la electrónica de control. La parte central del sistema electrónico suele ser un microcontrolador, *PLC* o *FPGA*, entre otros. Esta parte se encarga de actuar sobre el sistema en función de los datos que reciba de los sensores. Con el auge del movimiento *maker* se hicieron famosos los microcontroladores de electrónica abierta como *Arduino*.

#### *2.1.1.2. Mecánica*

Dentro de un sistema electromecánico, el sistema mecánico se compone de las partes móviles, las partes que transmiten movimiento y la estructura.

#### *2.1.1.3. CAD*

Se entiende por *CAD* (*Computer Aided Design*) al proceso de diseño asistido por ordenador de un modelo. Para realizar este diseño se emplean programas específicos. La invención de este tipo de programas supuso una reducción del tiempo invertido en el diseño al permitir modificar distintas dimensiones del modelo con su sistema de gráficos interactivos comparado con el proceso de diseño anterior *[13]*.

Actualmente, existen distintas técnicas de modelado 3D entre las que destacan *[14]*:

- Modelado de sólidos usando operaciones booleanas (*Constructive Solid Geometry: CSG*): los modelos se generan mediante la realización de operaciones booleanas, es decir, mediante la unión, diferencia, intersección o exclusión de dos elementos como se ilustra en la *Figura 3*.


*Figura 3 - Operaciones booleanas (Elaboración propia)*

- Modelado de sólidos usando ecuaciones integrales de límites (*Boundary representation: B-rep o BREP*): genera sólidos a partir de puntos o líneas. A partir de estos puntos se generan las ecuaciones diferenciales parciales para generar la superficie. Con este método se puede obtener una superficie más definida.

Una de las peculiaridades del *CAD* es la dificultad de modificar o reutilizar los diseños. Esta modificación puede llegar a suponer rehacer el diseño desde cero con la inversión de tiempo que esto supone. Según el proceso empleado para crear el modelo 3D será más sencilla o compleja su modificación, pudiendo llegar a ser imposible modificar el modelo. Con el objetivo de disminuir el tiempo de desarrollo de un producto se buscó la manera de reusar y hacer modificaciones de los modelos ya creados.

En un modelo paramétrico la geometría está controlada por parámetros que se pueden definir de manera dimensional, geométrica o constantes algebraicas. El uso de este tipo de modelos permite alterar el diseño modificando unos parámetros sin tener que hacer desde cero el modelo *[15]*. Por ejemplo, en la *Figura 4* se puede ver cómo en un modelo "no paramétrico" el cilindro interior no está definido por la ecuación de una circunferencia y se pierde información.

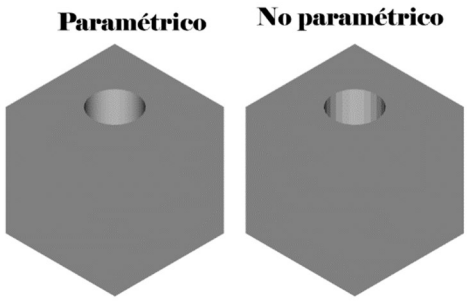


*Figura 4 - Comparativa de modelo paramétrico vs no paramétrico (Elaboración propia)*

Por ejemplo, suponiendo que se tiene un cubo de dimensiones 10x10x10mm con un agujero pasante de radio 2mm. En un modelo *CAD* convencional, primero se realizaría el modelo del cubo y después el modelo de un cilindro con la altura del cubo y el radio deseado. Cuando se tienen ambos modelos se realiza la operación booleana de resta de volúmenes para eliminar del cubo el volumen del cilindro. Si después de todo el proceso para obtener el modelo se quiere cambiar el radio del agujero, radio que depende del cilindro, no se puede y se tiene que generar el modelo de nuevo. Si realizamos el mismo cubo por medio de un modelo paramétrico se puede modificar el valor del radio sin tener que volver a hacer todo el proceso lo cual supone un ahorro de tiempo.

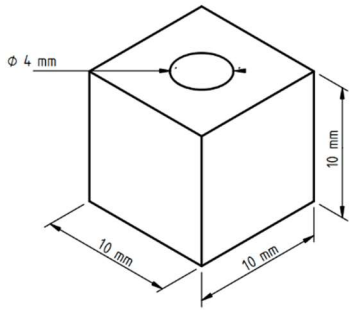


*Figura 5 - Cubo con agujero (Elaboración propia)*

El *CAD* paramétrico cobra importancia en los últimos años con la aparición de la metodología *BIM* (*Building Information Modeling* o Modelado de Información de la Construcción). Esta metodología consiste en generar un modelo matemático 3D del edificio y añadir la información útil del proyecto sobre este modelo *[16]*.

### Formatos de exportación:

La capacidad de compartir diseños *CAD* es fundamental, especialmente cuando se realizan diseños abiertos donde se debe dar la posibilidad de modificar y estudiar el diseño al gusto de cada uno. Por este motivo existen distintos formatos de archivos que permiten almacenar la información del diseño, pero no todos los formatos son capaces de compartir la misma información.

- *DXF*: formato de exportación extendido en diseños 2D.
- *STL*: formato de exportación 3D ampliamente utilizado en las impresoras 3D. Este formato exporta un mallado de triángulos, por lo que no es el mejor formato para compartir los modelos ya que se pierde la información paramétrica.
- *STEP*: es el formato más recomendado para compartir modelos 3D, además de ser estándar ISO-10303 *[17]*. Este formato incluye información de tolerancia y es capaz de manejar los ensamblajes y modelos sólidos.

### Equipos informáticos:

Los programas *CAD* necesitan unos requisitos mínimos. Para poder entender correctamente los términos que se van a utilizar en la comparativa de los programas se van a introducir distintos componentes de un equipo informático *[18][19][20]*:

- Procesador o *CPU* (*Central Processing Unit*): es la unidad de procesamiento central de un ordenador. En este microprocesador se realizan todo tipo de cálculos y se coordinan los distintos componentes del ordenador.
- *GPU* (*Graphics Processing Unit*): es la unidad de procesamiento gráfico de un ordenador. Puede estar embebida junto al *CPU* o por separado, es decir, estar en una tarjeta gráfica dedicada. Es parecida a la *CPU*, pero este microprocesador está desarrollado para realizar cálculos vectoriales y matriciales.
- Memoria *RAM* (*Random Access Memory*): se encarga de almacenar la información que necesita la *CPU* de forma temporal. Suelen ser memorias de tipo *DRAM* (*Dynamic RAM*).

### 2.1.2. Movimiento maker y *OpenLabware*

El movimiento *maker* surge de la corriente *DIY*. La unión del movimiento *maker* junto con el auge del *Open Source Hardware* (*OSH*), o *Hardware Abierto*, desencadena en el movimiento *Open Labware [8]*. Dentro de este movimiento podríamos encontrar una gran variedad de material de laboratorio con un coste sumamente inferior a las alternativas comerciales *[9]*. Además, el movimiento promueve el uso de la impresión 3D y los microcontroladores de bajo coste como *Arduino [5]*, *Beagleboard [10]* o *Raspberry Pi [11]*.

#### *2.1.2.1. Software Libre*

El *Software Libre* (*OSS*) otorga a los usuarios la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Para considerar un programa como *Software Libre* debe cumplir cuatro libertades *[1]*:

1. Libertad para ejecutar el programa con el propósito deseado.
2. Libertad para estudiar y cambiar el programa para que funcione como se quiera.
3. Libertad de distribuir copias.
4. Libertad de distribuir copias modificadas con acceso al código fuente. Esto aporta el beneficio de las modificaciones a la comunidad.

#### *2.1.2.2. Hardware libre*

El Hardware Libre u "*Open Source Hardware*" (*OSH*), al igual que el *OSS*, permite al usuario estudiar, fabricar, cambiar, compartir y vender un diseño. Para ello se debe incluir la documentación que permita fabricar o modificar el diseño *[2][3][5]*.

#### *2.1.2.3. Impresión 3D*

La impresión 3D es un método de fabricación por aportación de material. Una impresora 3D permite convertir un modelo 3D digital en un modelo 3D real mediante la adición de material en capas. Aunque existen distintos métodos para realizar dicha adición de material, el más común es la Deposición de Filamento Fundido. En este método se calienta el material deseado, normalmente materiales

termoplásticos como PLA (Ácido Poliláctico) o ABS (Acrilonitrilo Butadieno Estireno), hasta su temperatura de fusión para poder depositarse en capas *[21][22]*.

#### *2.1.2.4. Arduino*

*Arduino* es una plataforma de electrónica abierta. Las placas de *Arduino* consisten en un microcontrolador capaz de leer señales de sensores y mandar señales a actuadores en función de estas por medio de sus pines. Las placas de *Arduino* son muy económicas, lo que hace que sean perfectas para la instrumentación científica, diseño de prototipos y empezar en el mundo de la programación o la robótica.


*Figura 6 - Placa Arduino UNO [4]*

## 2.2. Comparativa de programas CAD

Dentro de los programas *CAD* debemos diferenciar entre aquellos que son de licencia propietaria y aquellos que son de licencia abierta. Para evaluar el mercado se tomarán como ejemplo tres programas de licencia propietaria y tres de licencia abierta. Los seis programas tienen la posibilidad de diseñar en 2D y 3D. Además, será necesario evaluar el coste del equipo más económico que tengan las características capaces de ejecutar el programa de forma correcta.

La información que se va a ver en los puntos *2.2.1(Programas de licencia propietaria)* y *2.2.2(Programas con licencia abierta)* se encuentra ampliada en el *Anejo I* - Programas CAD.

### 2.2.1. Programas de licencia propietaria

#### *CATIA*

*CATIA*, diseñado por *DASSAULT SYSTÈMES*, cuenta con distintas funcionalidades para, además de crear un modelo 3D, realizar análisis en distintos campos de la ingeniería *[23][25]*. La propia compañía no facilita un coste de la licencia de forma pública y redirige al usuario a un departamento de ventas. Este programa sólo funciona en sistemas operativos *Windows*.

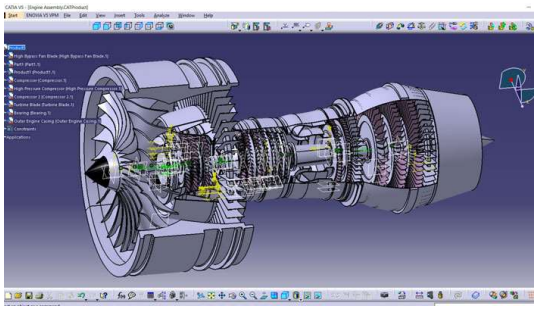

*Figura 7 - Interfaz de CATIA [24]*

### *AutoCAD*

*AutoCAD* es un software diseñado por *Autodesk* que permite realizar tanto un diseño convencional como un diseño paramétrico añadiendo restricciones geométricas o por cota. *AutoCAD* está disponible para *Windows* y *macOS* y, al igual que *CATIA*, cuenta con herramientas específicas para distintas ramas de la ingeniería *[29]*.

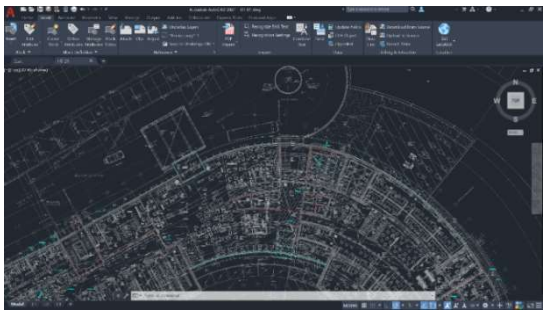

Figura *8 - Interfaz de AutoCAD [30]*

### *SolidWorks*

*SolidWorks*, al igual que *CATIA*, es un software diseñado por *DASSAULT SYSTÈMES* que cuenta con soluciones específicas para sector específico de la industria y está disponible para *Windows* y *macOS [31]*. Permite más formatos de exportación que *CATIA*, aunque su manejo de las superficies es más limitado.

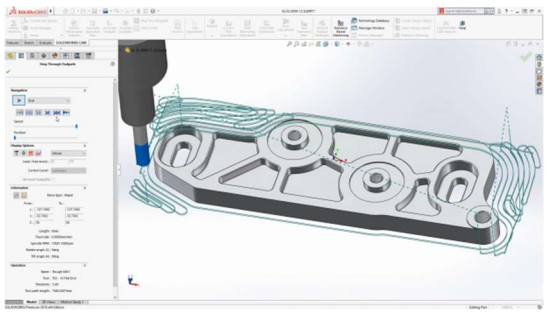

*Figura 9 - Interfaz de SolidWorks [33]*

*Resumen de los programas de licencia propietaria*

Los tres programas que se han visto permiten tanto el diseño de modelos en 3D de forma gráfica como el diseño paramétrico. Sin embargo, además de tener en cuenta el coste de la licencia es necesario valorar el coste del equipo mínimo. En el caso de *AutoCAD* es más sencillo elegir el equipo más económico, ya que aporta los datos de hardware mínimo que se deben cumplir. Por el contrario, los dos programas restantes dificultan esta labor. En el caso concreto de *CATIA* no se aporta ningún dato de requisitos mínimos, pero aporta un listado de equipos certificados.

Como se puede observar en la *Tabla 1* el programa *AutoCAD* es el más económico y supondría una inversión inicial alrededor de 3.000€ y mantener la licencia supondría una inversión anual cercana a 2.200€. Sin embargo, en estos precios no se tienen en cuenta los distintos módulos específicos que añaden herramientas.

*Tabla 1 - Comparativa programas de pago*

| | | ***CATIA*** | ***AutoCAD*** | ***SolidWorks*** |
|---|---|---|---|---|
| **Empresa** | | *DASSAULT SYSTÈMES* | *AutoDesk* | *DASSAULT SYSTÈMES* |
| **Sistemas operativos** | | *Windows* | *Windows* o *macOS* | *Windows* o *macOS* |
| **Coste licencia** | | Primer año: 9.410€* Mantenimiento: 1.680€/año* | 2.230€/año | Primer año: 6.600€ Mantenimiento: 1.500€/año |
| **Procesador** | | - | 2,5GHz | 3,3GHz |
| **Memoria RAM** | | - | 8 GB | 16GB |
| **Tarjeta gráfica** | Ancho de banda | - | 29GB/s | - |
| | VRAM | - | 1 GB | - |
| **Coste del equipo mínimo** | | 1.600€** | 800€ | 800€ |
| **Inversión inicial** | | 11.010€ | 3.010€ | 7.400€ |
| **Coste anual** | | 1.680€ | 2.230€ | 1.500€ |

*Datos originales en dólares: la conversión dólar a euro es 1$=0,84€

** Coste del equipo más económico que está certificado y se encuentra disponible a fecha 28 de agosto de 2020.

### 2.2.2. Programas con licencia abierta

*OpenSCAD*

*OpenSCAD* es un *OSS* diseñado para la creación de sólidos 3D. Cuenta con un editor de texto donde se describe el modelo con funciones matemáticas y dicho modelo aparecerá en el espacio de trabajo. Se pueden crear modelos en 2D o 3D y realizar operaciones booleanas con ellos *[36]*.

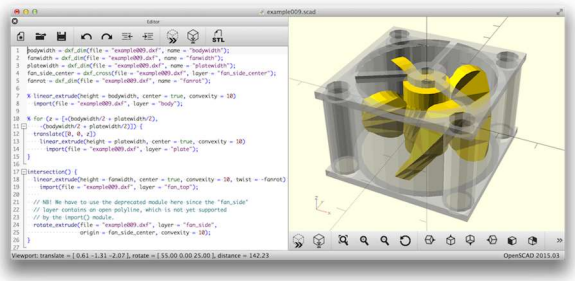

*Figura 10 - OpenSCAD [36]*

La previsualización de los modelos se realiza mediante *OpenCSG*, el cual funciona correctamente con gráficas integradas, lo que supone un menor coste del equipo *[39]*. *OpenSCAD* permite exportar el modelo en formatos tales como *STL* y *DXF*, pero no en formato *STEP*.

### *FreeCAD*

*FreeCAD* es un proyecto de *OSS* que nace en 2001 dirigido a la ingeniería mecánica y el diseño de productos. Permite realizar modelos paramétricos, añadir módulos externos programados en *C++* o *Python* y puede importar o exportar modelos en los formatos estándar *[41][42]*.

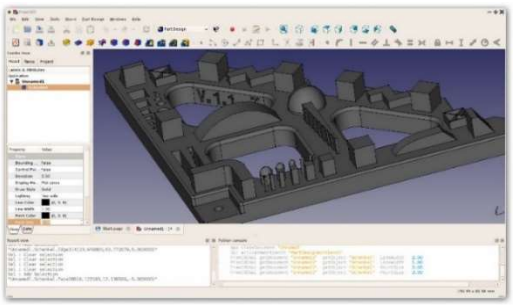

*Figura 11 - FreeCAD [41]*

### *CadQuery*

*CadQuery* nació como un módulo de *FreeCAD*, aunque después se convirtió en un programa independiente *[43]*. Al igual que *OpenSCAD*, la creación de modelos se realiza mediante un lenguaje de programación en el cual se describe el modelo con fórmulas matemáticas. Permite el modelado paramétrico y exportar en formato *STEP* y *STL [44]*.

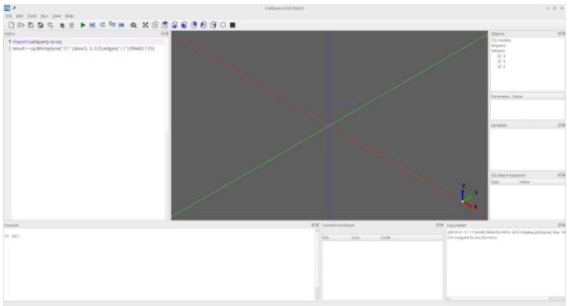

*Figura 12 - CadQuery [44]*

*Resumen de los programas de licencia abierta:*

Los tres programas de licencia abierta permiten generar modelos a partir de código sin necesidad de usar la interfaz gráfica. Sin embargo, *OpenSCAD* utiliza programación funcional que se basa en funciones matemáticas y, por tanto, es más sencillo de aprender para aquellos que no tienen conocimientos de programación. Por otro lado, *CadQuery* hace uso de una librería propia basada en *Python* para describir los modelos. El uso de esta librería propia simplifica la programación, pero el lenguaje es propio del programa. Por último, *FreeCAD* permite programar modelos en *C++* o *Python*, dos de los lenguajes de programación más usados *[45]*. Además, hay que destacar que *OpenSCAD* utiliza la librería de modelado *CGAL* basada en *CSG*, es decir que genera los modelos a partir de formas básicas usando operaciones booleanas. Por otro lado, *FreeCAD* y *CadQuery* emplean la librería de modelado *OCCT* que genera los modelos a partir de ecuaciones. Por tanto, con *OCCT* se obtendrán modelos de mayor calidad que con *CSG*, ya que esta última trabaja con mallas y, por tanto, pierde la información paramétrica. También es importante tener en consideración que *FreeCAD* cuenta con interfaz gráfica (*GUI*) mientras que *OpenSCAD* y *CadQuery* no. Por último, si se pretende crear hardware libre, es importante poder compartir los modelos sin perder ningún tipo de información. Por tanto, es importante poder exportar los modelos en formato *STEP*.

Las principales características de los tres softwares libres se recogen en la *Tabla 2*.

*Tabla 2 - Comparativa de programas gratuitos*

| | ***OpenSCAD*** | ***FreeCAD*** | ***CadQuery*** |
|---|---|---|---|
| **Software libre** | Sí | Sí | Sí |
| **Interfaz gráfica** | No | Sí | No |
| **Lenguaje de programación** | Lenguaje propio de programación funcional | *C++* *Python* | Lenguaje propio basado en *Python* |
| **Librería de modelado** | *CGAL* | *OCCT* | *OCCT* |
| **Formatos exportación estándar** | No exporta formato *STEP* | Sí | Sí |
| **Módulos externos** | No | Sí | No |

### 2.2.3. Comparativa de los programas

Para elegir el programa se deben seguir las recomendaciones del *OSH*. Esto implica prescindir de los programas de licencia propietaria, además de por su elevado coste, por la imposibilidad de modificarlos, es decir, por no poder añadir nuevas utilidades al software *[46]*. Cabe remarcar que, aunque se prescinda de los programas de licencia propietaria para este fin, dichos programas tienen mejores prestaciones para fines profesionales.

Todos los softwares libres por evaluar permiten el modelado paramétrico mediante programación, por lo que se tienen que valorar otras características para elegir el software a emplear en la *Tabla 3*.

Tanto *OpenSCAD* como *CadQuery* carecen de interfaz gráfica interactiva de forma que es necesario aprender el lenguaje de programación propio de cada uno. Además, se debe tener en cuenta la posibilidad de exportar los modelos en formatos estándar como *STEP*.

*FreeCAD*, por el contrario, sí que cuenta con interfaz gráfica, por lo que será más sencillo para usuarios sin conocimientos de programación. Además, *FreeCAD* permite, a cualquier usuario, añadir módulos externos programados en *Python*.

*Tabla 3 - Comparativa de programas CAD*

| | ***OpenSCAD*** | ***FreeCAD*** | ***CadQuery*** | ***CATIA*** | ***AutoCAD*** | ***SolidWorks*** |
|---|---|---|---|---|---|---|
| **Coste licencia** | Gratuito | Gratuito | Gratuito | 9.410€ | 2.230€ | 6.600€ |
| **Mantenimiento de licencia** | Gratuito | Gratuito | Gratuito | 1.680€ | 2.230€ | 1.500€ |
| **Sistemas Operativos** | *Windows, macOS, Linux* | | | *Windows* | *Windows, macOS* | |
| **Coste equipo (prestaciones)** | 200€ (bajas) | 200€ (bajas) | 200€ (bajas) | 1600€ (altas) | 800€ (medias) | 800€ (medias) |
| **CAD Paramétrico** | No | Sí | Sí | Sí | Sí | Sí |
| **Formatos de exportación** | *DXF* *STL* | *DXF* *STL* *STEP* | *DXF* *STL* *STEP* | *DXF* | *DXF* *STL* *STEP* | *DXF* *STL* *STEP* |
| **Modificable** | Sí | Sí | Sí | No | No | No |
| **Interfaz gráfica** | No | Sí | No | Sí | Sí | Sí |
| **Módulos externos** | No | Si | No | No | No | No |

Por tanto, se usará *FreeCAD* dado que tiene interfaz gráfica, permite añadir módulos externos (bancos de herramientas) programados en *Python* y permite exportar en formatos estándar como *STEP*, *STL* y *DXF*. Además, es *OSS*, un requisito indispensable para desarrollar *OSH* y que permite que cualquiera pueda modificar los diseños.

# 3. Objetivos

El objetivo principal del proyecto es la realización de un banco de herramientas para *FreeCAD*. Tanto el banco de herramientas como los modelos paramétricos que en él se encuentran están realizados en el lenguaje de programación *Python*. En el desarrollo del proyecto se deben cumplir los siguientes objetivos:

- Diseñar un *Software Libre* que simplifique la creación de *CAD*.
- Tener una biblioteca de modelos 3D paramétricos que sean accesibles desde su código fuente.
- Desarrollar un banco de herramientas con una interfaz gráfica que incluya la biblioteca de modelos 3D y permita la personalización del modelo deseado por parte del usuario.
- Creación de un sistema que permita realizar la composición entre modelos 3D de manera sencilla desde el banco de herramientas desarrollado.
- Dar la posibilidad de exportar los modelos de la biblioteca en formato *STL* en la posición más adecuada para su impresión 3D.
- Desarrollar una biblioteca de funciones que faciliten la programación de futuros modelos 3D.
- Desarrollar una documentación accesible de forma pública tanto de la biblioteca de modelos como de la biblioteca de funciones.

Cumpliendo estos objetivos se facilita el diseño de sistemas mecatrónicos en *CAD* para aquellos con poca experiencia en el diseño. Además, se permite la reproducción de los experimentos científicos a bajo coste, haciendo más accesible la realización de estos experimentos en zonas con bajos recursos.

# 4. Solución Técnica

El proyecto consta de dos vertientes muy distintas, pero complementarias. Por una parte, está el banco de herramientas, un software que permite la creación de modelos 3D paramétricos por medio de una interfaz gráfica, la cual se verá en profundidad en el apartado *4.1(Mechatronic Workbench)*. Por otra parte, se tiene el modelo de *OSH*, *Filter Stage.* Este sistema es un posicionador lineal de un filtro óptico que se ha creado con los modelos 3D del banco de herramientas mencionado que se tratará en el apartado *4.2 (Sistemas realizados con el Workbench)* y que servirá como aplicación y demostración del banco de herramientas.

La realización del diseño de un sistema en *CAD* requiere de mucho tiempo y de la realización de muchas pruebas. Con el fin de reducir el tiempo invertido en la realización del diseño y las pruebas se propone la utilización del banco de herramientas creado en este proyecto.

Asimismo, para complementar las competencias impartidas en el grado se realiza el montaje físico del sistema *Filter Stage*, la implementación de la electrónica y el diseño del sistema de control.

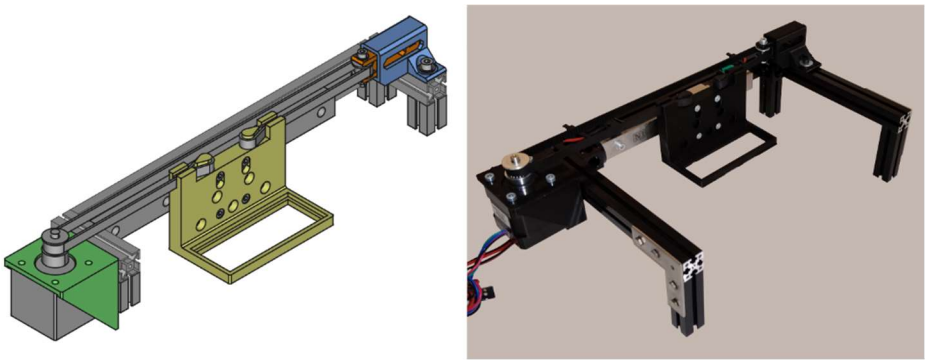

*Figura 13 - Modelo CAD y modelo real (Elaboración propia)*

El objetivo final de este proyecto es la creación de un banco de herramientas que permita a los usuarios la creación, utilización y modificación de los modelos paramétricos desarrollados en lenguaje *Python*. Para ello es necesario programar una interfaz gráfica sencilla de usar por un usuario inexperto sin conocimiento de diseño *CAD*.

Por último, cabe comentar la publicación del banco de herramientas en el foro oficial de *FreeCAD [56]*. Tras la publicación en el foro algunos usuarios han realizado peticiones de mejora y utilidades que serán parte de las líneas futuras del proyecto.

## 4.1. Mechatronic Workbench

El banco de herramientas *Mechatronic* o "*Mechatronic Workbench*" es un banco de herramientas diseñado en lenguaje *Python 3.8* y el módulo gráfico *PySide2*. Este workbench (*WB*) es *OSS*. Para poder

considerar el trabajo como *OSS* y *OSH* se ha creado una web en *ReadtheDocs [48]* con la información necesaria para la instalación, así como una web con datos sobre los modelos y pequeños tutoriales. También se encuentra disponible el código fuente desde el repositorio "*Mechatronic Documentation*" de *GitHub [47]*. Por último, se realizaron varios videotutoriales para mostrar el *WB* en funcionamiento y que se encuentran disponibles en *YouTube [49]*.

El *WB* permite seleccionar entre sus 21 modelos, 2 ensamblajes y 2 funciones. Cuando se selecciona un modelo, ensamblaje o función se genera un menú con distintas opciones en la ventana de *FreeCAD "TASKS"*. Las ventanas propias de la interfaz de *FreeCAD* se pueden ver en el apartado *FreeCAD* dentro del *Anejo I* - Programas CAD. Para facilitar la comprensión del funcionamiento se va a realizar la demostración de funcionamiento con el modelo *Filter Stage*.

Primero se selecciona el *WB* dentro de *FreeCAD*. Para ello, en la parte superior se debe clicar en el menú desplegable y seleccionar la opción "*Mechatronic*" como se ilustra en la *Figura 14*. Al seleccionar el *WB* se carga la barra de herramientas con todos modelos y funciones disponibles. Asimismo, en la barra superior del programa se cargan los menús propios del *WB*: *Parts*, *Optic*, *Systems* y *Functions*.

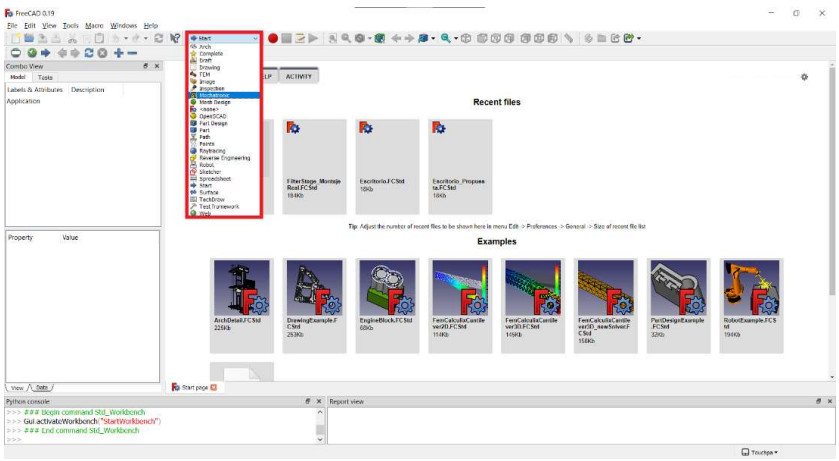


*Figura 14 - Paso 1: Seleccionar el worbench (Elaboración propia)*

Siguiendo con el ejemplo, para crear el *Filter Stage* se debe seleccionar el modelo desde su icono en la barra de herramientas o desde el menú *Systems* como se ve en la *Figura 15*. Al seleccionar este modelo se crea un menú en la ventana *TASKS* como se muestra en la *Figura 16*. Este menú tiene distintas opciones que permiten definir los principales parámetros del sistema como el tamaño de filtro (ancho y largo), el ancho de los perfiles de aluminio, el tamaño de los tornillos y el tamaño del motor.

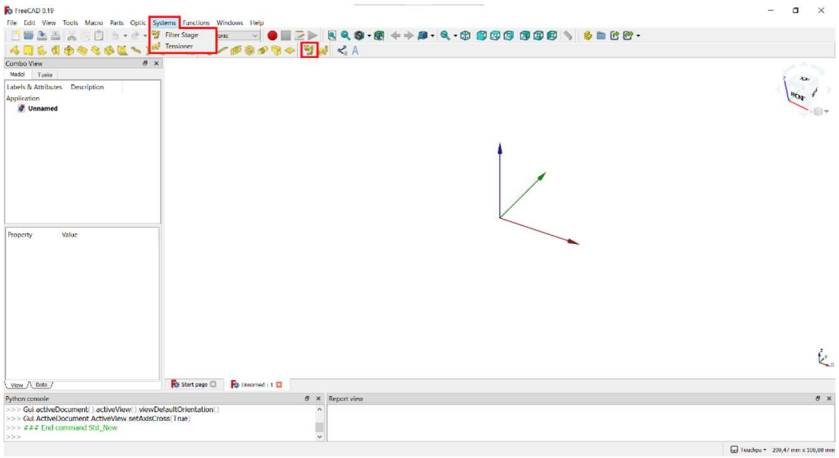

*Figura 15 – Paso 2: Seleccionar el modelo FilterStage (Elaboración propia)*

En el caso del ejemplo que se está viendo los valores que se introducen son los indicados en la *Tabla 4*.

*Tabla 4 - Valores de los parámetros*

| Parámetros | Valor |
|---|---|
| **Move distance** | 150 |
| **Filter Length** | 75 |
| **Filter Width** | 30 |
| **Base width** | 15 |
| **Tensioner stroke** | 20 |
| **Wall thick** | 3 |
| **Nut type** | M3 |
| **Motor size** | 14 |
| **Rail high Motor holder** | 35 |
| **Motor holder thickness** | 3 |

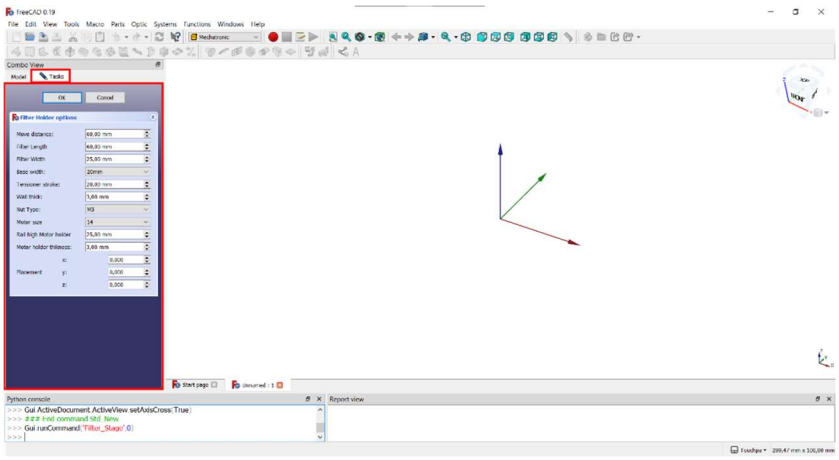

*Figura 16 – Paso 3: Menú del Filter Stage (Elaboración propia)*

Una vez se introducen los valores se debe clicar en el botón "*OK*" y el modelo se genera en el espacio de trabajo, como se puede ver en la *Figura 17*.

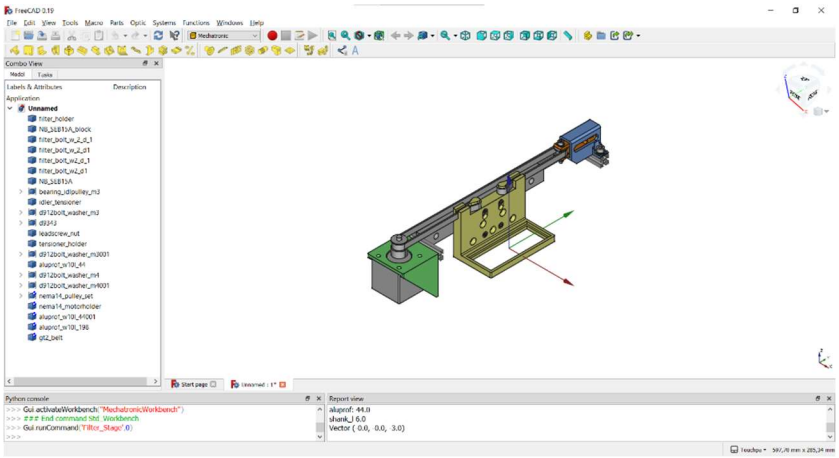

*Figura 17 - Paso 4: Modelo del Filter Stage (Elaboración propia)*

El modelo del *Filter Stage* que se ha generado no tiene ningún soporte en el modelo *CAD* generado. Por tanto, falta añadir 4 perfiles de aluminio que hagan de soporte. Para añadir los perfiles al modelo se debe seleccionar el modelo desde el icono como se ilustra en la *Figura 18*.

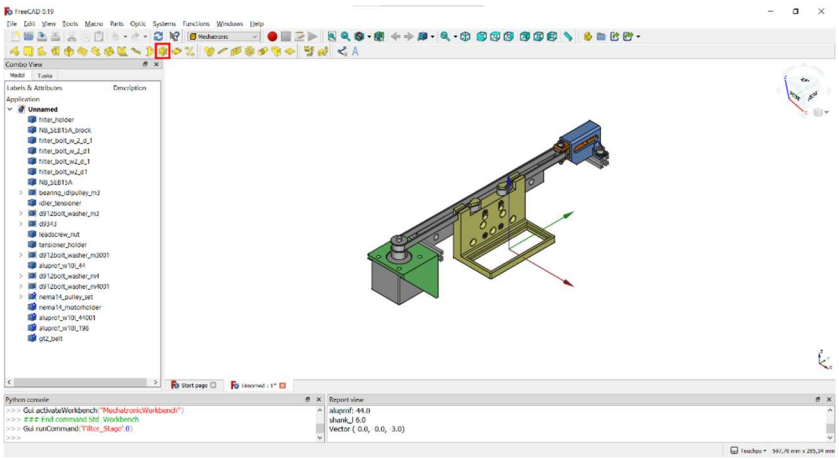

*Figura 18 - Paso 5: Selección del modelo Aluminium Profile (Elaboración propia)*

Al seleccionar el modelo se genera el menú en la ventana *TASKS* como se muestra en la *Figura 19*. Este menú se caracteriza por tener dos sub-menús: *Aluminium Profile Options* y *Advance Placement*. El sub-menú *Aluminium Profile Options* tiene las distintas opciones del modelo que se ha seleccionado. El sub-menú *Advance Placement* es útil para añadir modelos en puntos internos de otros modelos. Si en el sistema se tuviera algún modelo con puntos internos sería tan sencillo como seleccionar el modelo que se quiere tomar de referencia y seleccionar el punto interno de interés. Para comprobar que se han seleccionado los puntos internos correctos se puede pulsar el botón *Show Point* que mostrará en color verde dicho punto. Si el punto deseado es el correcto sólo queda fijar esta posición en el modelo que se va a generar. Para ello se debe pulsar sobre el botón *Set this position to the model*. En este caso se prescinde de este último sub-menú.

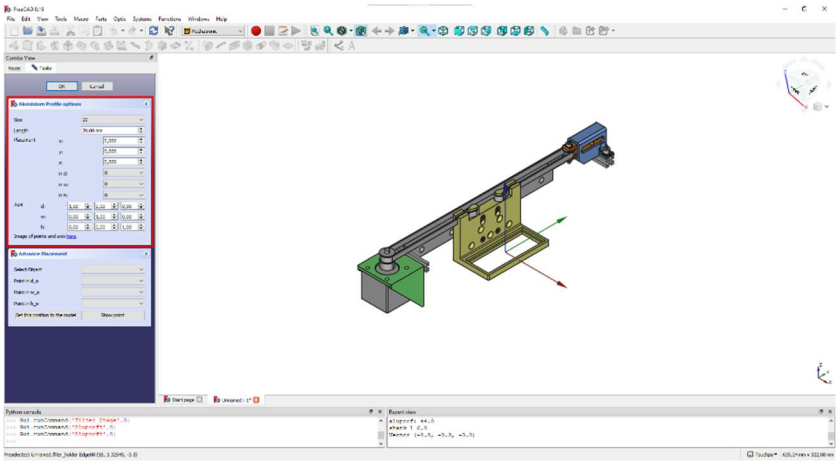

*Figura 19 - Paso 6: Menú Tasks del Aluminium Profile (Elaboración propia)*

Para colocar el nuevo perfil los valores del menú serán:

- Tamaño *(Size)*: 15
- Longitud *(Length)*: 20mm
- Posición *(Placement)*: se fijará con el ratón pulsando con el botón izquierdo sobre la posición deseada. En este caso será la esquina inferior izquierda del perfil de aluminio situado más a la derecha. Dicha esquina devuelve la siguiente posición en el espacio:
    - X: -7,0
    - Y: 115,1
    - Z: 14,9

  También será necesario fijar los puntos internos (in d, in w, in h) en función del punto del modelo que se quiera situar en dicha posición. Para seleccionar estos puntos internos es necesario mirar la imagen de los ejes y puntos internos del modelo. Esta imagen se encuentra en un link en la parte inferior del sub-menú. Consultando dicha imagen los valores que se deben seleccionar son:
    - In d: 5
    - In w: -3
    - In h: -3
- Ejes *(Axis)*: los ejes, al igual que los puntos internos, dependen de cada modelo, por lo que es necesario consultar la imagen. Por la posición y cómo se han seleccionado los puntos internos se deben fijar los siguientes vectores para cada eje:
    - d: (0,0,1)
    - w: (0,1,0)
    - h: (-1,0,0)

Una vez introducidos todos los valores se debe clicar en el botón *OK* y el modelo aparecerá en la posición deseada como se ve en la *Figura 20*.

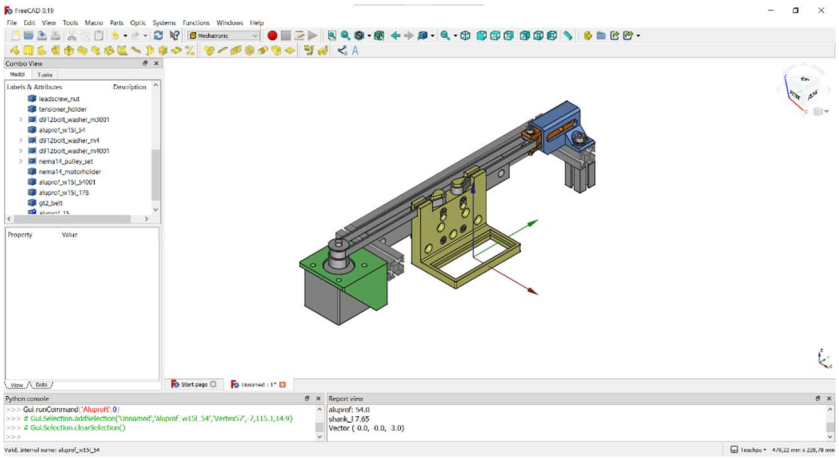

*Figura 20 - Paso 7: Modelo FilterStage con un soporte (Elaboración propia)*

Para completar el soporte de perfiles de aluminio queda repetir el proceso que se acaba de ver con los tres perfiles restantes. Sin embargo, dado que los perfiles que se quieren son idénticos al que ya se ha generado, se va a copiar el perfil ya creado y se va a emplear la función *Assembly* para cambiar su posición.

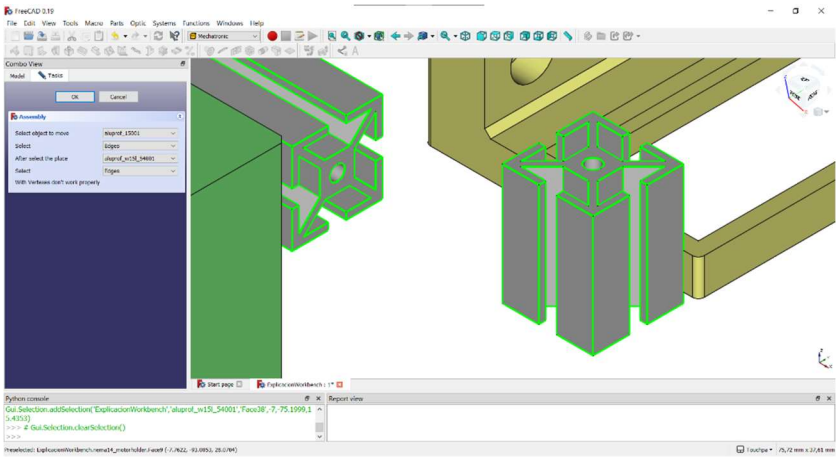

*Figura 21 – Paso 8: Menú Assembly (Elaboración propia)*

Para emplear esta función se selecciona en el menú el modelo que se quiere mover y el modelo de referencia de la posición. Una vez seleccionados los modelos, se selecciona la parte de estos que se quiere resaltar y se pulsa el botón *OK*, en este caso se seleccionarán los bordes (*edges*) como se ve en la *Figura 21*. Por último, se debe seleccionar el borde del modelo que se quiere mover y, manteniendo pulsada la tecla *CONTROL*, el borde del modelo donde se quiera posicionar la anteriormente seleccionada y pulsar otra vez en el botón *OK*.

El resultado final sería el que se puede ven en la *Figura 22*.

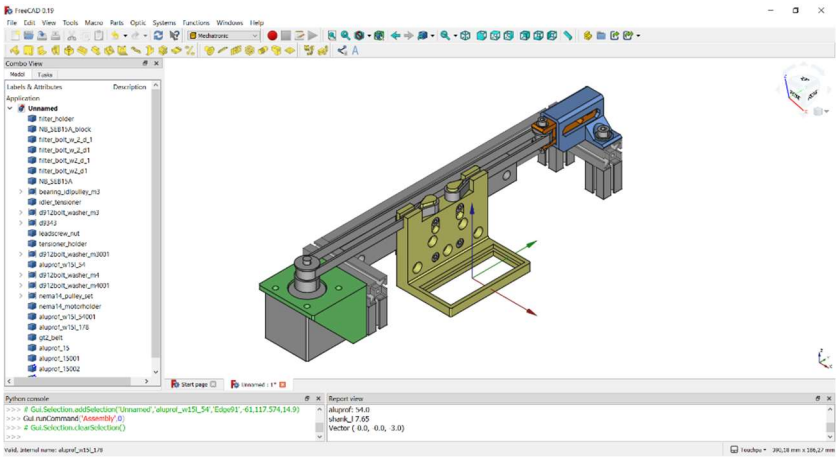

*Figura 22 - Paso 9: Sistema FilterStage con soporte (Elaboración propia)*

### 4.1.1. Biblioteca de Modelos

La biblioteca de modelos se divide en dos partes. La primera parte consta de modelos con finalidad mecánica y la segunda con finalidad óptica. Ambas bibliotecas se pueden ver en la *Figura 23.*

*Figura 23 - Mechatronic Workbench (Elaboración propia)*

Los componentes de las bibliotecas se enumeran en la *Tabla 6, Tabla 7* y *Tabla 13* junto a los tiempos medios de generación en dos equipos portátiles de gamas y prestaciones muy distintas representadas en la *Tabla 5*.

*Tabla 5 - Prestaciones de los equipos de pruebas*

| Prestaciones portátil | Coste (€) | Procesador | RAM | Gráfica integrada | Gráfica dedicada |
|---|---|---|---|---|---|
| **Alta** | 1500 | AMD Ryzen 7 4800HS a 2.9GHz | 16 GB | AMD Radeon Graphics 7 a 1600 MHz | Nvidia RTX 2060 |
| **Baja** | 300 | Intel Celeron N4000 a 1,10GHz | 8 GB | UHD Intel 600 a 200 MHz | No tiene |

Para medir el tiempo de generación de los modelos se ha empleado la función *time.time()* con la cual se obtiene el tiempo actual. Empleando esta función antes y después de la *función_del_modelo* se calcula el tiempo en realizar el modelo. Para poder obtener un tiempo medio de generación del modelo se generan 100 modelos. El tiempo de generación de cada modelo se almacena en la variable *suma*. Cuando se han realizado los 100 modelos, se toma el tiempo total y se calcula la media. La metodología que se acaba de explicar está contenida en el código que se muestra en el *Código 1*.

```
import time
suma = 0
for i in range(100):
    t1 = time.time()

    función_del_modelo

    t2 = time.time()
    suma = suma + (t2-t1)
media = suma/100
print(media)
```

*Código 1 - Generación de 100 modelos y cálculo de tiempo medio*

Como se ve en la *Tabla 6* y la *Tabla 7* los tiempos entre los dos equipos, a pesar de las grandes diferencias entre prestaciones, no son sustanciales en la mayoría de los modelos ya que se tratan de diferencias de décimas de segundo. En casos más complejos, como el modelo "Carcasa de rodamiento lineal", la diferencia de tiempos aumenta hasta 1,2 segundos. A pesar de ser un tiempo más elevado, se puede considerar insignificante en comparación con el tiempo que tardaría un usuario inexperto en generar manualmente dicho modelo. Así mismo, existen casos inusuales, como el modelo "Soporte de eje", donde se observa que el equipo con bajas prestaciones es más rápido que el equipo de altas prestaciones.

*Tabla 6 - Modelos de la librería mecánica y sus tiempos de generación*

| **Modelos mecánicos** | **Tiempo medio (s) Prestaciones altas** | **Tiempo medio (s) Prestaciones bajas** |
|---|---|---|
| **Soporte de eje *(Shaft holder)*** | 0,298 | 0,176 |
| **Soporte polea loca *(Idler holder)*** | 0,207 | 0,231 |
| **Escuadra *(Bracket)*** | 0,358 | 0,211 |
| **Soporte de Motor *(Motor holder)*** | 0,388 | 0,340 |
| **Motor *(motor)*** | 0,417 | 0,274 |
| **Soporte de interruptor *(Switch holder)*** | 0,609 | 0,823 |
| **Carcasa de rodamiento lineal *(Linear bear housing*** | 2,080 | 3,296 |
| **Soporte general de escuadra *(Hall Stop)*** | 0,256 | 0,438 |
| **Soporte de filtro *(Filter holder)*** | 0,890 | 0,838 |
| **Abrazadera de correas *(Belt clamp)*** | 0,090 | 0,163 |
| **Soporte de sensor *(Sensor holder)*** | 0,301 | 0,475 |
| **Perfil de aluminio *(Aluminium profile)*** | 0,024 | 0,047 |
| **Guía lineal *(Linear guide)*** | 0,105 | 0,195 |
| **Tornillos, tuercas y arandelas *(Bolt, nut, washer)*** | 0,035 | 0,064 |

Todos los modelos tienen unas características propias que se almacenan junto al modelo y se verán en el apartado *4.1.2 (Características de los modelos)*. Además, los modelos están generados usando la biblioteca de funciones básicas que se verá en el apartado *4.1.7 (Biblioteca de funciones básicas)*. Asimismo, cualquier usuario puede emplear la librería de funciones para crear sus propios modelos siguiendo la explicación que se verá en el apartado *4.1.6 (Diseño de Modelos 3D en lenguaje Python).*

*Tabla 7 - Modelos de la librería óptica y sus tiempos de generación*

| Modelos ópticos | Tiempo medio (s) Prestaciones altas | Tiempo medio (s) Prestaciones bajas |
|---|---|---|
| **Lente de tubo** | 0,114 | 0,079 |
| **Base LCPB1M** | 0,243 | 0,190 |
| **Placa** | 0,101 | 0,138 |
| **CageCube** | 0,387 | 0,888 |
| **ThLed30** | 0,159 | 0,137 |
| **PrizLed** | 0,079 | 0,120 |
| **BreadBoard** | 0,131 | 0,193 |

### 4.1.2. Características de los modelos

En el apartado anterior se ha visto que el *WB* tiene varios modelos mecánicos y ópticos. Los modelos se realizaron inicialmente siguiendo el *Diseño de Clases* realizado por Felipe Machado, aunque posteriormente se modificó el *Diseño de Clases* para añadir nuevas características. El diseño de clases se verá en profundidad en el apartado *4.1.8 (Diseño de Clases)*.

Los parámetros básicos que tienen todos los modelos realizados en cualquiera de los *Diseños de Clases* son:

- Nombre: este nombre será el que vea el usuario en la interfaz gráfica de *FreeCAD*.
- Forma.
- Punto origen: es el origen del sistema de coordenadas interno. Se define con un punto interno en cada eje interno como: pos_o = (pos_d, pos_w, pos_h).
- Posición: lugar en el espacio en el que se sitúa el origen del modelo.
- Ejes internos: conjunto de tres vectores ortonormales que definen el sistema de coordenadas interno del modelo. Estos ejes están definidos por las direcciones de profundidad (eje d - depth), ancho (eje w - width) y alto (eje h - height).
- Puntos internos: conjunto de puntos característicos en las tres direcciones de los ejes internos. Están definidos junto con el modelo y son puntos de interés como el centro de un orificio donde se puede situar un tornillo.
- Eje de simetría: dado que muchos de los modelos son simétricos respecto a alguno de sus ejes, los puntos internos se generan sólo en la mitad positiva, es decir, en el sentido positivo del eje correspondiente. Por tanto, es necesario definir un parámetro booleano (verdadero o falso) para cada eje que determine si el modelo es simétrico o no respecto a él.

### 4.1.3. Ensamblajes

Además de los modelos individuales, es posible que el usuario quiera generar un sistema formado por varios modelos individuales. Esto es lo que se denominará ensamblaje. Para ello el *WB* tiene un menú

de ensamblajes *(Assemblies)*. Dentro de este menú se encuentran dos sistemas ya ensamblados con un fin específico, que están representados en la *Tabla 8* junto a los tiempos de generación en dos equipos con características distintas, que se encuentran representadas en la *Tabla 5*.

El sistema "*Tensioner*" es parte del *Filter Stage*, aunque se incluye por separado para poder generar únicamente este conjunto.

El sistema "*Filter Stage*" se explicará en detalle en el apartado *4.2 (*Sistemas realizados con el Workbench*)*.

*Tabla 8 – Ensamblaje y sus tiempos de generación*

| Modelos ópticos | Tiempo medio (s) Prestaciones altas | Tiempo medio (s) Prestaciones bajas |
|---|---|---|
| **Filter Stage** | 3,011 | 3,884 |
| **Tensioner** | 1,031 | 1,449 |

En caso de que se desee añadir un sistema, se podría realizar en lenguaje *Python* y añadirse a este menú. De esta forma el sistema sería accesible para cualquier persona sin conocimientos de programación y, en caso de ser parte de un experimento, sería fácil de reproducir con las mismas condiciones.

### 4.1.4. Tipos de usuarios

A la hora de usar *FreeCAD* se pueden distinguir dos tipos de usuarios.

- Usuario básico, sin conocimientos de programación.
- Usuario avanzado, con conocimientos de programación.

El usuario básico, el más común, hará uso de la interfaz gráfica de usuario (*GUI*) de *FreeCAD*. Por tanto, la interfaz que tenga el *WB* deberá ser intuitiva y sencilla de usar. El diseño del *GUI* se verá en profundidad en el apartado *4.1.5 (Diseño de GUI)*.

El usuario avanzado, además de usar el *GUI*, dados sus conocimientos de programación, podrá usar la consola de *Python*. Para este tipo de usuario, además de la interfaz gráfica se proporciona una librería de funciones propia del proyecto. Con esta librería de funciones podrá crear sus propios modelos de una forma más sencilla. La forma de crear estos modelos se verá en el apartado *4.1.6 (Diseño de Modelos 3D en lenguaje Python)*.

### 4.1.5. Diseño de GUI

El *GUI* es la parte gráfica del programa con la que el usuario va a interactuar y es la parte a la cual se ha dedicado un mayor tiempo. De la interfaz de *FreeCAD* sólo se modificarán la barra de herramientas,

que contendrá los botones del *WB*, y la ventana *TASKS*, donde aparecerán las opciones correspondientes a cada modelo paramétrico como se ejemplifica en la *Figura 24*.

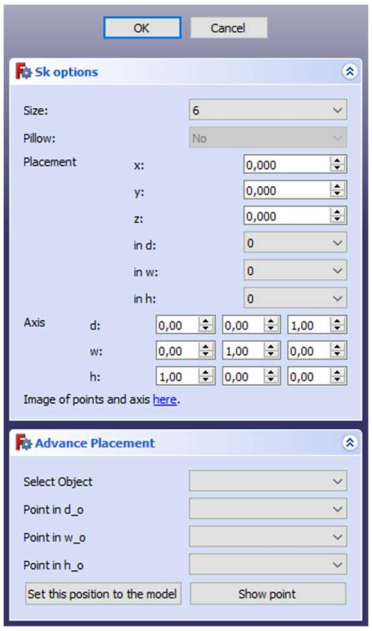


*Figura 24 - Interfaz final del modelo SK (Elaboración propia)*

La interfaz de *FreeCAD* se genera en lenguaje *Python* con el módulo *PySide2*. Este módulo permite generar distintos tipos de objetos interactivos llamados *Widgets*. Los más usados en la interfaz serán los mostrados en la *Figura 25*.

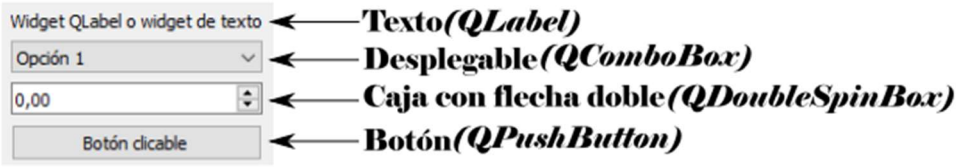


*Figura 25 - Widgets más usados (Elaboración propia)*

Además, *PySide2* da la posibilidad de colocar los *Widgets* con distintos diseños. A estos distintos diseños se les llamará *layout*. El más usado es el *layout* de cuadrícula por ser muy sencillo de usar ya que sólo necesita el número de la fila y el de la columna donde se desea colocar el *widget*. En este tipo de *layout* todos los *widgets* tienen el mismo tamaño, lo cual puede producir la superposición de estos cuando el *layout* no tenga el espacio suficiente para todos los *widgets*. Como se puede ver en *Figura 26* el *layout* de cuadrícula tiene un *layout* representado en color azul claro y un conjunto de *widgets* encima de este representados en color azul oscuro.

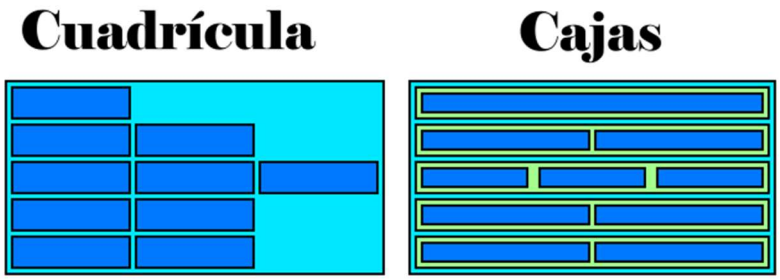


*Figura 26 - Comparativa de layout (Elaboración propia)*

Para solventar los problemas de superposición de *widgets* se optó por usar un *layout* de cajas. El *layout* de cajas se genera en distintos niveles de forma que cada *widget* tiene un tamaño en función del nivel en el que esté. En concreto el *layout* está generado en 3 niveles que se van a explicar del inferior al superior de forma ascendente:

1. Un *layout* de caja vertical, de color azul claro, será el nivel inferior sobre el que situar otros *layout*.
2. *Layouts* de cajas horizontales, de color verde. Se generan tantos como líneas se quieran poner en el menú.
3. Los *widgets*, de color azul oscuro, se sitúan sobre los *layouts* horizontales.

De esta forma los *widgets* contenidos en un *layout* se reparten por igual el espacio de éste y, por tanto, se evita la superposición de *widgets.*

En la *Figura 27* se muestra el cambio que supuso el cambio de un *layout* a otro.


*Figura 27- Cambio de layout (Elaboración propia)*

### 4.1.6. Diseño de Modelos 3D en lenguaje Python

El diseño de modelos 3D se puede hacer mediante la interfaz gráfica, pero para poder añadirlos al *WB* deberán generarse en lenguaje *Python*. La manera de generar estos modelos será por medio de la biblioteca de funciones básicas cuyo código fuente está disponible en el repositorio del proyecto *[52]*.

Lo primero será generar una clase con el nombre que se quiere, como ejemplo usaremos la clase *Washer()*. Esta clase hereda de *ShpCylHole,* por ello se encuentra entre los paréntesis como se muestra a continuación:

*class Washer (ShpCylHole)*

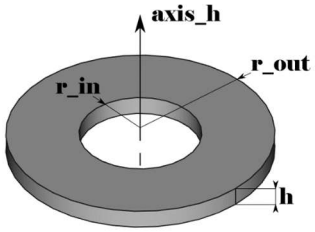


*Figura 28 - Arandela y sus dimensiones (Elaboración propia)*

Esta clase contendrá una función principal llamada *init()*. Esta función será la encargada de generar el modelo 3D como el que se muestra en la *Figura 28*. Además, como se quiere que la función sea paramétrica, deberá tener como parámetros el radio exterior (*r_out*), radio interior (*r_in*) y altura (*h*). Luego, adicionalmente, tiene los parámetros de eje de la altura (*axis_h*), posición en la altura del origen (*pos_h*), tolerancia (*tol*) y posición (*pos*). Esta función se vería de la siguiente forma:

```
def __init__(self, r_out, r_in, h, axis_h, pos_h, tol = 0, pos = V0):
```

Para que el modelo almacene como propiedades su posición y su nombre, ambas deben asignarse en el código.

```
self.name = 'washer'
self.position = pos
```

Para generar la forma del modelo se debe llamar a la función *ShpCylHole* con los parámetros característicos de la arandela que se va a realizar.

```
ShpCylHole.__init__(self, r_out = r_out, r_in = r_in, h = h, axis_h = axis_h, pos_h = pos_h
xtr_r_in = tol_r, xtr_r_out = - tol_r, pos = self.pos, name = self.name)
```

Con estos pasos ya se tiene una forma creada, pero esta forma aún no es visible en *FreeCAD*, por lo que se tiene que crear un objeto *FreeCAD*.

```
super.create_fco()
```

La función "*super*" hace referencia a la clase de la que se hereda. En este caso, como se mencionó anteriormente, la clase que se está creando, "*Washer*", hereda de la clase "*ShpCylHole*". Por tanto, cuando se pone "*super*" se hace referencia a "*ShpCylHole*". Así mismo, "*ShpCylHole*" hereda de la clase "*Obj3D*", la clase básica del *Diseño de Clases*.

Por último, quedaría situar el objeto *FreeCAD* (*fco*) en la posición que tiene como propiedad.

```
self.fco.Placement.Base = self.position
```

Como resumen, para crear un objeto hay que crear una *clase* que contenga una *función constructor*. Esta función será la encargada de crear la forma, generar el objeto *FreeCAD* y situarlo en su posición. De esta forma, el código de *Python* quedaría como se puede ver en el *Código 2.*

```
class Washer (ShpCylHole)
   def __init__(self, r_out, r_in, h, axis_h, pos_h, tol = 0, pos = V0):
      self.name = 'washer'
      self.pos = FreeCAD.Vector(0,0,0)
      self.position = pos
      ShpCylHole.__init__(self, r_out = r_out, r_in = r_in, h = h, axis_h = axis_h,
            pos_h = pos_h, xtr_r_in = tol_r, xtr_r_out = - tol_r, pos = self.pos,
            name = self.name)
      super().create_fco()
      self.fco.Placement.Base = self.position
```

*Código 2 - Clase para la creación de una arandela*

En caso de que se quiera crear un objeto que sea la unión de varias formas básicas se ha diseñado una función que permite sumar o restar volúmenes de manera simple. Esta función está incluida dentro de la clase "*Obj3D*" y necesita solamente dos parámetros. El primer parámetro será la forma, mientras que el segundo parámetros determinará si la forma suma (1) o resta volumen (0).

*Obj3D.add_child(self, forma, 1, nombre)* - Para sumar volumen

*Obj3D.add_child(self, forma, 0, nombre)* - Para restar volumen

Por ejemplo, para crear una placa con una perforación se tendrían dos formas básicas: un tetraedro y un cilindro. El tetraedro generará un volumen que se quiere añadir al modelo, mientras que el cilindro tendrá un volumen que se quiere eliminar para hacer la perforación. Es necesario especificar las posiciones que van a ocupar estos volúmenes, ya que la perforación puede realizarse en cualquier parte de la placa. Para el ejemplo se realizará la perforación en el centro de la placa.

Después de generar ambos volúmenes se deben sumar, para ello se diseñó la función *make_parent* que permite realizar todas las sumas y restas de volúmenes necesarias con una única línea.

Por tanto, la clase que genera la placa perforada quedaría como se puede ver en el *Código 3*.

```
class PlacaPerforada (Obj3D)
   def __init__(self, d, w, h, r, pos = V0):
      axis_d = VX
      axis_w = VY
      axis_h = VZ
      self.name = 'placa_perforada'
      self.position = pos
      super().__init__(axis_d, axis_w, axis_h, name)
      posicion_perforacion = FreeCAD.Vector(d/2, w/2, 0)
      placa = addBox(d, w, h, 'placa')
      cilindro = ShpCyl(r, h, pos = posicion_perforacion)
      super().add_child(placa, 1, 'placa')
      super().add_child(cilindro, 0, 'cilindro')
      super().make_parent(self.name)
      super().create_fco(self.name)
      self.fco.Placement.Base = self.position
```

*Código 3 - Clase de para la creación de una placa perforada*

### 4.1.7. Biblioteca de funciones básicas

La biblioteca de funciones básicas permite generar modelos básicos con una única línea de código en *Python*. Con esta biblioteca se simplifica la creación de nuevos modelos en lenguaje *Python* y reduce la cantidad de líneas de código que se deben escribir siguiendo el principio de modularidad.

```
doc = FreeCAD.ActiveDocument                                addbox(x,y,z,name)
x0 =  0
x1 =  x
y0 =  0
y1 =  y
z0 = 0
p00 = FreeCAD.Vector (x0,y0,z0)
p10 = FreeCAD.Vector (x1,y0,z0)
p11 = FreeCAD.Vector (x1,y1,z0)
p01 = FreeCAD.Vector (x0,y1,z0)
sq_list = [p00, p10, p11, p01]
square =  doc.addObject("Part::Polygon",name + "_sq")
square.Nodes =sq_list
square.Close = True
square.ViewObject.Visibility = False
box = doc.addObject ("Part::Extrusion", name)
box.Base = square
box.Dir = (0,0, z)
box.Solid = True
doc.recompute()
```

*Código 4 – Sin función VS con función*

Por ejemplo, generar un tetraedro sería tan sencillo como escribir *"addBox(x,y,z,name)"* dónde *x*,*y*,*z* serían largo, ancho y alto del tetraedro correspondiente. Adicionalmente, como los objetos de *FreeCAD* necesitan un nombre, se da la opción de poner uno con la argumento "*name*". La diferencia de líneas que se deben escribir se puede ver en el *Código 4* .

La biblioteca consta de 101 funciones documentadas tanto en código como en la web de forma que la librería pueda ser utilizada por cualquiera. Además, al estar documentado cumple las recomendaciones de *SSO*.

### 4.1.8. Diseño de Clases

El *Diseño de Clases* es la base de todos los modelos, por este motivo es muy importante que esté bien realizado y contenga todas las características que se desean implementar. Este diseño, para poder ser comprendido, se realiza en *Unified Modeling Lenguaje* (*UML*) o *Lenguaje Unificado de Modelado*.

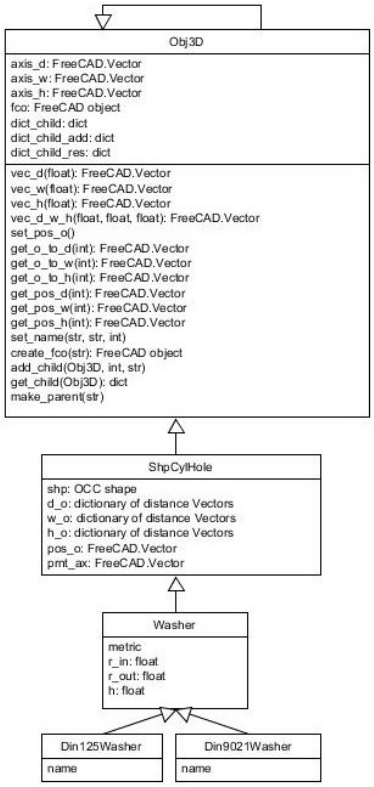


*Figura 29 - Diseño de Clases en UML (Elaboración propia)*

El diseño empleado finalmente es muy sencillo y se puede ver en la *Figura 29* y consiste en una clase principal denominada "*Obj3D*". Esta clase tiene un conjunto de propiedades y funciones propias que tendrán todos los modelos 3D. Estas propiedades son las mismas que se han explicado en el apartado *4.1.2 (Características de los modelos).* De esta clase principal heredan el resto de clases, es decir, las propiedades y funciones que tiene esta clase las tendrán el resto de clases que dependan de esta. La herencia entre clases se ilustra por medio de una flecha. El origen de dicha flecha marca la clase que hereda las propiedades y funciones de la clase en la que finaliza la flecha.

Para comprender correctamente el diseño, se va a estudiar el ejemplo representado en la *Figura 29*.

- Los modelos *Din 125* y *Din 9021* son arandelas con un nombre específico que determina métrica, radio interior, radio exterior y altura o grosor.
- Ambos tipos de arandelas se pueden incluir dentro de una clase genérica "*Washer*" (*Arandela*). Esta clase específica contiene los atributos de métrica (*metric*) radio interior (*r_in*), radio exterior (*r_out*) y altura (*h*).
- Una arandela se puede considerar un cilindro con agujero interior (*ShpCylHole*) ya que un cilindro con agujero tiene un radio interior, radio exterior y altura. Además, esta clase (*ShpCylHole*) tiene una forma concreta, uno puntos internos (*d_o*, *w_o*, *h_o*), una posición y un eje óptimo de impresión (*prnt_ax*).

- Por último, el cilindro con agujero interior se puede considerar un *Obj3D* con unos ejes internos (*axis_d*, *axis_w*, *axis_h*) y tiene la posibilidad de ser un objeto de *FreeCAD* (*fco*). Además, si este objeto estuviera compuesto por un conjunto de objetos, tendría un diccionario con los objetos que lo componen, es decir, tendría un diccionario de hijos (*dict_child*) que serían *Obj3D*.

El ejemplo que se ha visto se podría reproducir con una estructura similar para generar todos los modelos que se incluyen en la librería de modelos.

Como se ha mencionado antes, el diseño por el que se ha optado es muy sencillo, especialmente si se compara con el *Diseño de Clases* anterior que se puede ver en la *Figura 30*. El principal inconveniente que presenta el diseño anterior, el de la *Figura 30*, es la herencia múltiple. Como anteriormente se ha visto, una clase puede heredar las propiedades y funciones de otra. En el caso de la herencia múltiple, la clase hereda las propiedades y funciones de dos o más clases. Esta clase de herencia se daba ya que existían dos clases importantes: *Obj3D* y *SinglePart*. La clase *Obj3D* no generaba objetos en *FreeCAD* (fco) sino que era una característica propia de *SinglePart*. Por tanto, los modelos eran componentes por sí solos *(SinglePart)* con una posición dentro del espacio de trabajo de *FreeCAD*, pero también eran *Obj3D* con unos ejes internos y unas funciones propias. Además, los *SinglePart* podían ser parte de un conjunto de piezas *(PartsSet)*.

Por ejemplo, una arandela (*Washer*) es un cilindro con agujero (*ShpCylHole*) al igual que en el nuevo diseño, pero también será un componente (*SinglePart*). Esta dependencia de dos clases distintas está contraindicada a no ser que sea completamente necesaria. Por este motivo, además de la sencillez, se decidió aplicar el nuevo *Diseño de Clases*.

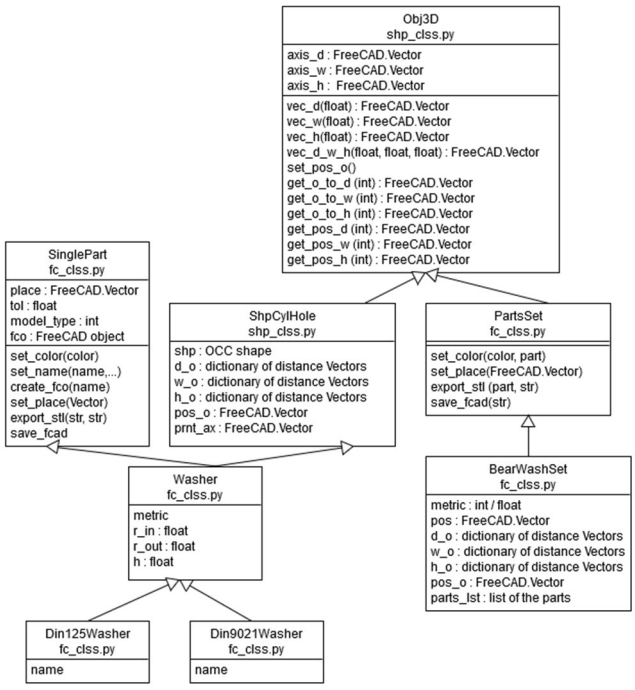


*Figura 30 - Diseño UML inicial de Felipe Machado (Elaboración propia)*

### 4.1.9. Documentación del código

En cualquier tipo de *SSO* es esencial realizar una documentación que permita a cualquier usuario entender cómo se debe usar. Además, dado que en el caso de este proyecto se crean unas librerías, para que el usuario sea capaz de usarlas de forma inequívoca, debe estar cada modelo o función documentada.

Inicialmente se decidió realizar una página web usando *GitHubPages*, parte de la plataforma de *GitHub*. Dicha página web era muy sencilla y estaba realizada en lenguaje *HTML* y *CSS*, los principales lenguajes de programación de páginas web.

Para implementar una mejor documentación se optó por usar "*ReadtheDocs*". Esta plataforma simplifica la documentación de software y la gestiona en versiones sin la necesidad de emplear *HTML* ni *CSS*. Para poder usar esta plataforma se debe emplear el módulo de Python *Sphinx*. *Sphinx* permite generar páginas web de forma sencilla usando *Markdown* o *reStructuredText*, lenguajes que permiten dar formato a los textos de forma rápida.

A pesar de que *Markdown* ya se había empleado en la documentación básica realizada para *GitHub*, se decidió usar *reStructuredText*. Esta decisión está motivada porque *Markdown* no soporta muchas características avanzadas de *Sphinx*.

Además, *Sphinx* cuenta con el módulo "*autodoc*" que le permite leer los comentarios en código *Python*. Esto permite documentar cada modelo o función únicamente en código, ahorrando el tiempo correspondiente a crear la documentación en la web. Asimismo, cuando se realice un cambio en el código se verá reflejado en la web de forma automática.

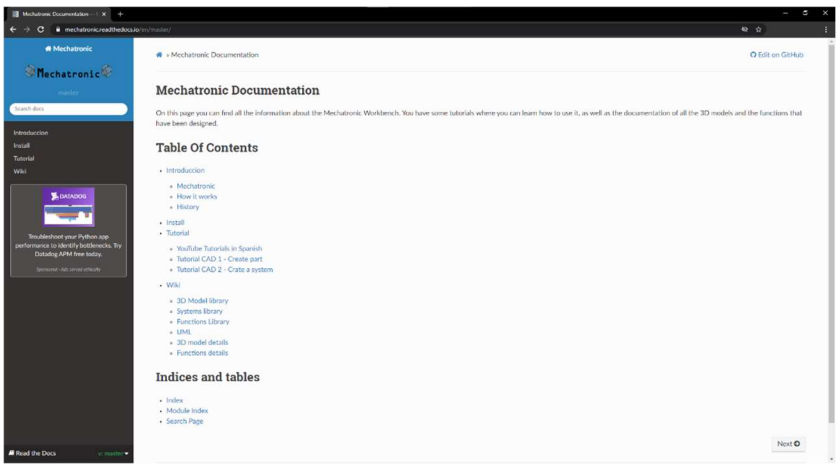


*Figura 31 - Página web en ReadtheDocs (Elaboración propia)*

La página web está realizada en inglés y tiene cinco apartados:

- Principal: Contiene una breve explicación del contenido de la página y el índice del contenido
- Introducción: explica brevemente los dos tipos de usuario y cómo debería proceder cada uno de ellos. También añade un apartado de historia donde se explica cómo surgió el proyecto.

- Instalación: en este apartado se encuentran explicadas las dos formas de instalación del WB. También se facilita, por medio de un enlace directo, la descarga de *FreeCAD* y de *Mechatronic Workbench*.
- Tutoriales: En este apartado se incluyen embebidos los dos videotutoriales en español y dos tutoriales escritos con apoyo en imágenes.
- Wiki: este apartado es el más amplio. En él se incluyen los siguientes subapartados:
    - Librería de modelos: listado de todos los modelos con imágenes.
    - Librería de sistemas: listado de los sistemas incluidos.
    - Librería de funciones: listado de todas las funciones.
    - Diseño de clases: breve explicación del diseño de clases.
    - Librería de modelos detallada: se encuentran documentadas las clases que generan los modelos.
    - Librería de funciones detallada: se encuentran las funciones documentadas.

## 4.2. Sistemas realizados con el Workbench

### Filter Stage

El *Filter Stage* (posicionador lineal de un filtro óptico) es un proyecto que se puede enmarcar dentro de *Open-Labware*. Es un proyecto original de Felipe Machado que se encuentra disponible en GitHub como modelo ensamblado o por piezas para *FreeCAD [50]* y como para *OpenScad [51]*, sólo los modelos para impresión.

El *Filter Stage* consta de cuatro modelos imprimibles:

- Soporte del filtro (Filter Holder) – distancia de movimiento 150mm – ancho 30mm – largo 75mm
- Soporte del tensionador (Tensioner Holder)
- Tensor de la polea (Idler Pulley)
- Soporte del motor (Motor Holder) - 35mm de alto – 3mm espesor – nema 14

Además de los modelos imprimibles, para su montaje serán necesarios los siguientes componentes:

- Placa Arduino UNO
- Guía lineal
- Sensor final de carrera x2
- Botón x2
- Pololu A4988
- Protoboard

- Perfiles de aluminio
- Tornillos, tuercas y arandelas
- Correa dentada
- Fuente de alimentación externa de 12V

A continuación, se explican brevemente los pasos a seguir para realizar el ensamblaje que se muestran en *Figura 32*. Dicho ensamblaje se encuentra en detalle en el *(Anejo III - Ensamblaje del* sistema Filter Stage*)*.

1. Montar la **guía lineal**, el **soporte del motor** y el **soporte del tensionador** sobre los **perfiles de aluminio** de longitudes 200mm, 150mm y 150mm correspondientemente. Posteriormente se deben unir dichos perfiles entre sí.
2. Añadir los **perfiles de aluminio** de longitud 50mm a la estructura anterior.
3. Montar el **soporte del filtro** sobre la **guía lineal**, el **motor** en el **soporte del motor**, la **polea** en el **tensionador** y después el **tensionador** sobre el **soporte del tensionador**.
4. Por último, colocar la **correa dentada** entre el tensionador y el motor y situar un **sensor final de carrera** en cada extremo de la **guía lineal**.

Tras realizar el montaje de la estructura queda realizar el montaje eléctrico. Para ello es necesario consultar la información del *Pololu A4988 [53]*. Es importante colocar una resistencia de *Pull Down*, tanto en los finales de carrera como en los botones para establecer un valor lógico cuando los interruptores no están activos. Sin estas resistencias el valor no se puede determinar con certeza. Al colocar estas resistencias el pin está conectado a tierra (0 lógico), ya que son de *Pull Down*, y cuando se active el botón llegarán 5V al pin (1 lógico).

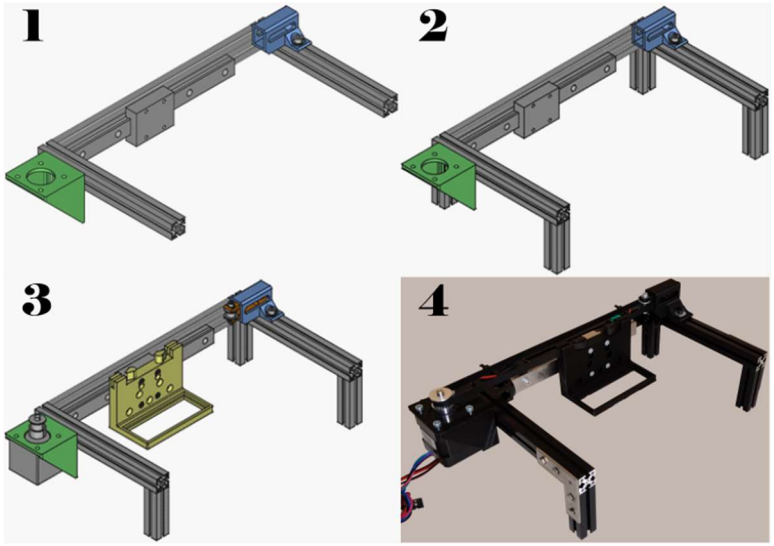


*Figura 32 - Pasos básicos de montaje (Elaboración propia)*

Así mismo, es importante situar un condensador de $100\mu F$ a la entrada de la fuente de alimentación externa del controlador del motor. Al colocar el condensador se suple la demanda extra de energía necesaria para iniciar el movimiento del motor paso a paso.

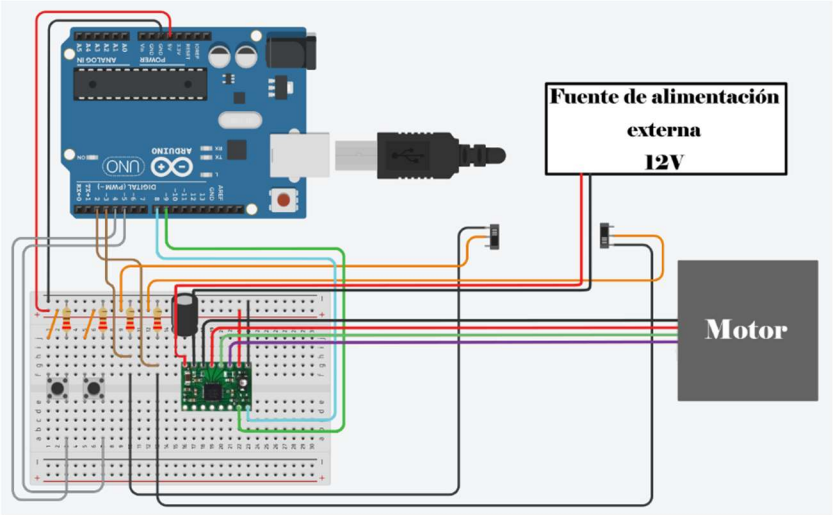


*Figura 33 - Esquema del montaje eléctrico (Elaboración propia)*

Este montaje eléctrico es una aplicación específica con el fin de demostrar el funcionamiento del sistema. Este sistema sería posible modificarlo en función del tipo de aplicación que se requiera.

Por último, se debe realizar el código de control del sistema que funcionará en la placa de Arduino Uno. Dicho código de control será una máquina de estados que controlará el movimiento del filtro con un motor paso a paso. El motor podrá mover el filtro a derecha o izquierda en función del estado en el que se encuentre. Los estados del sistema de control desarrollados serán:

- Parado: cuando el filtro no se mueva, es decir, que el motor esté parado, siempre que no se encuentre en ninguno de los dos extremos, se encontrará en este estado.
- Moviendo a la derecha: mientras se mantenga pulsado el botón derecho el filtro se moverá en este sentido, es decir, el motor girará.
- Moviendo a la izquierda: mientras se mantenga pulsado el botón izquierdo el filtro se moverá en este sentido, es decir, el motor girará.
- Parado tope derecho: si el sistema se encuentra en el estado "moviendo a la derecha" y llega al sensor final de carrera derecho se entrará en este estado. En este estado el motor se parará y sólo podrá salir de él cuando se pulse el botón izquierdo.
- Parado tope izquierdo: si el sistema se encuentra en el estado "moviendo a la izquierda" y llega al sensor final de carrera izquierdo se entrará en este estado. En este estado el motor se parará y sólo podrá salir de él cuando se pulse el botón derecho.

## 4.3. Resultados

Como se ha visto en los apartados anteriores, realizar un diseño *CAD* supone una gran inversión de tiempo, pero esta inversión de tiempo puede ser menor gracias al *WB* desarrollado. Por otro lado, para realizar un diseño *CAD* hace falta tener experiencia tanto en el programa que se va a emplear como en la realización de estos diseños. La experiencia necesaria para realizar los diseños con el *WB* será menor, ya que da la posibilidad de realizar todo el diseño por medio de la interfaz gráfica.

Asimismo, con los tiempos de creación de modelos vistos en la *Tabla 6*, la *Tabla 7* y la *Tabla 8* se puede llegar la conclusión de que no es necesario un equipo informático de altas prestaciones para generar los modelos.

Existen sistemas comercializados con características similares a las del sistema propuesto, Filter Stage, como se puede ver en la *Tabla 9*. Los sistemas comerciales tienen un coste superior, en parte debido a los componentes de mayor calidad que se usan en estos sistemas. Sin embargo, el sistema propuesto tiene como ventaja principal la particularización del sistema, así como la posibilidad de remplazar un componente dañado.

*Tabla 9 - Sistemas comerciales*

| **Sistemas** | **Coste** |
|---|---|
| **150 mm Motorized Linear Translation Stage** | 2.232,88€ |
| **150 mm Translation Stage with Stepper Motor** | 2.625,35€ |
| **Sistema propuesto: *Filter Stage*** | 75,88€ |

El sistema propuesto es mucho más económico que los sistemas comerciales lo que permitiría la reproducción de experimentos. Así mismo, los sistemas comerciales tienen un sistema de control propio mientras que el sistema propuesto se puede controlar de varias formas, ya sea con el equipo informático o con otras formas que se quieran. Por ejemplo, en el caso de la aplicación específica desarrollada, por medio de botones.

El presupuesto para la realización del sistema propuesto se encuentra en la *Tabla 10*. Los precios de los componentes son orientativos ya que pueden variar. Precios tomados a día 28 de septiembre de 2020.

*Tabla 10 - Presupuesto del sistema propuesto*

| Componente | Unidades | Coste unitario (€) | Coste (€) |
|---|---|---|---|
| **Perfil de aluminio 20x50mm** | 4 | 0,1627 | 0,65 |
| **Perfil de aluminio 20x150mm** | 2 | 0,4882 | 0,98 |
| **Perfil de aluminio 20x200mm** | 1 | 0,6951 | 0,65 |
| **Final de carrera mecánico** | 2 | 0,9740 | 1,95 |
| **Motor paso a paso Nema 14 – Pololu 1209** | 1 | 13,1000 | 13,10 |
| **Polea GTR2 – 20 dientes** | 1 | 0,6500 | 0,65 |
| **Polea dentada GTR2 – 6mm – 1mm** | 1 | 1,0000 | 1,00 |
| **Arduino UNO** | 1 | 17,7000 | 17,70 |
| **Pololu A4988** | 1 | 1,8900 | 1,89 |
| **Condensador electrolítico** | 1 | 0,2866 | 0,29 |
| **Guía lineal - SSEBZ16-150** | 1 | 33,2100 | 33,21 |
| **Piezas impresas** | 0,055 | 13,8500 | 1,57 |
| **Arandelas din 125** | 2 | 0,0009 | 0,01 |
| **Arandelas din 9021** | 4 | 0,0014 | 0,01 |
| **Tuercas** | 27 | 0,0016 | 0,04 |
| **Tornillos** | 30 | 0,0725 | 2,18 |
| | | **Total:** | 75,88 |

La bobina filamento de PLA (1Kg) tiene un coste de 13,85€, por tanto, 1g de PLA tiene un coste de 0,01385€.

*Tabla 11 - Coste de impresión de modelos*

| Modelo | Tiempo | Cantidad de material | Coste material (€) |
|---|---|---|---|
| **Soporte motor** | 2h y 20min | 16g PLA – 2,07m | 0,22 |
| **Soporte del tensionador** | 1h y 12min | 8g PLA – 0,98m | 0,11 |
| **Tensionador** | 32m | 3g PLA – 0,44m | 0,04 |
| **Soporte del filtro** | 3h y 38min | 28g – 3,58m | 0,39 |
| | | **Total:** | 0,76 |

Con la realización del banco de herramientas y del prototipo del sistema *Filter Stage* se pueden considerar demostradas las competencias específicas de la *Tabla 12*.

*Tabla 12 - Competencias específicas*

| Competencias específicas |
|---|
| **CE3**. Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería. |
| **CE5**. Capacidad de visión espacial y conocimiento de las técnicas de representación gráfica, tanto por métodos tradicionales de geometría métrica y geometría descriptiva, como mediante las aplicaciones de diseño asistido por ordenador. |
| **CE12**. Conocimientos de los fundamentos de la electrónica. |
| **CE13**. Conocimientos sobre los fundamentos de automatismos y métodos de control. |
| **CE27**. Conocimiento de los fundamentos y aplicaciones de la electrónica analógica. |
| **CE28**. Conocimiento de los fundamentos y aplicaciones de la electrónica digital y microprocesadores. |
| **CE30**. Conocimiento aplicado de instrumentación electrónica. |
| **CE31**. Capacidad para diseñar sistemas electrónicos analógicos, digitales y de potencia. |
| **CE32**. Conocimiento de los principios de regulación automática y su aplicación a la automatización industrial. |
| **CE33**. Capacidad para diseñar sistemas de control y automatización industrial |

## 4.4. Líneas futuras

Existen muchas propuestas para ampliar o mejorar el proyecto. Algunas de las propuestas se ven a continuación:

- Por una parte, se podrían añadir más modelos paramétricos a la librería y por tanto al banco de herramientas, así como diseñar y añadir al banco de herramientas nuevos sistemas.
- Otra línea de mejora del software sería la adición de propiedades en los modelos actuales permitiendo la modificación de dichas propiedades desde la interfaz gráfica. La modificación de estas propiedades desde la interfaz gráfica haría que el modelo cambiara sin la necesidad de borrar dicho modelo y generar otro con las propiedades deseadas.
- También se podría mejorar la interfaz gráfica incluyendo la imagen del modelo con sus ejes y puntos internos sin necesidad de consultar una página web.
- Generar una nueva herramienta que permita añadir puntos internos a cualquier modelo, ya sea propio del banco de herramientas o no. Esta herramienta ha sido solicitada por la comunidad de *FreeCAD* que ha probado el banco de herramientas.
- Mejorar la función de ensamblaje añadiendo la opción de seleccionar los puntos internos. Además, se podría hacer que el menú fuera más interactivo de modo que se redujera el número de pulsaciones del ratón que tiene que realizar el usuario.
- Realizar la configuración de la página web para permitir seleccionar entre español o inglés así cómo realizar la traducción al castellano de dicha web.

# 5. Conclusiones

En este proyecto se ha diseñado un banco de herramientas que simplifica la creación de *CAD*. Dicho banco de herramientas incluye la biblioteca de modelos 3D, una funcionalidad para facilitar la composición entre modelos y otra funcionalidad para exportar los modelos en formatos *STL*. El código fuente del banco de herramientas, la biblioteca de modelos 3D y la biblioteca de funciones se encuentran disponibles en *GitHub* bajo licencia *LGLP-3.0* convirtiéndose así en *Software Libre*. También se ha desarrollado en *ReadtheDocs* la documentación pertinente para la instalación y utilización del banco de herramientas, donde también se encuentran documentadas la biblioteca de modelos y la biblioteca de funciones. Cabe destacar la inclusión del *WB* en el listado oficial de bancos de herramientas externos que se encuentra en la página de FreeCAD *[58]*. Esta inclusión es un reconocimiento por parte de los desarrolladores del programa y la comunidad de usuarios al trabajo que se ha realizado.

Adicionalmente se ha comprobado que el banco de herramientas funciona correctamente en equipos informáticos de gama baja permitiendo la creación de sistemas mecatrónicos en *CAD* con una inversión inicial reducida.

Tras la comparativa del sistema *Filter Stage* con los sistemas comerciales se ha comprobado que el sistema *Filter Stage* cumple con la misma función que los sistemas comerciales con un coste menor. Además, permite realizar el control del sistema con el método que se desee dada la gran personalización que se puede realizar. Asimismo, el usuario dispone de la documentación necesaria para reparar el sistema con un bajo coste. Por el contrario, los sistemas comerciales tienen su propio sistema de control y en caso de rotura sería complejo de sustituir cualquier componente estropeado. Cabe destacar que los sistemas comerciales tienen componentes de mayor calidad que el sistema *Filter Stage*.

Además, se demuestra el funcionamiento del sistema *Filter Stage* con el montaje del prototipo y la implementación del sistema de control. Con este prototipo también se demuestra la aplicabilidad del banco de herramientas desarrollado y dota al trabajo de una vertiente de aplicación de los conocimientos aprendidos durante el grado de ingeniería industrial.

Asimismo, se debe destacar el aprendizaje que se ha realizado en lenguaje *Python* y sus módulos *PySide2* y *Sphinx*. También hay que mencionar que el proyecto, sin líneas en blanco, supone un total de *67* archivos con *28.430* líneas en lenguaje *Python* con *21.614* líneas de comentarios y *471* líneas en lenguaje *reStructuredText*.

# 6. Bibliografía


[1] GNU, Free Software [*Consulta: 06 de abril de 2020*] *Disponible en:* https://www.gnu.org/philosophy/free-sw.es.html

[2] OSHWA [Consulta: 06 de abril de 2020] *Disponible en:* https://www.oshwa.org/definition/

[3] OpenHardware. Standardisation of Practices in Open Source Hardware *Disponible en:* https://openhardware.metajnl.com/articles/10.5334/joh.22/

[4] Arduino [Consulta: 06 de abril de 2020] *Disponible en:* https://www.arduino.cc/

[5] Arduino, Open Source Hardware [Consulta: 06 de abril de 2020] *Disponible en:* https://www.arduino.cc/en/Main/FAQ#toc3

[6] GUTIÉRREZ, Raúl Tabarés. La importancia de la cultura tecnológica en el movimiento maker. *arbor*, 2018, vol. 194, no 789, p. 471

[7] Alvarellos Navarro, S., García Sáez, C., Vaquero Rubio, E., Fernández Ajero, J., Grana Fernández, J., Salvador Polo, S., González de Opazo, C., González del H. *Manual de supervivencia Maker*. España, 20 de septiembre de 2015. ISBN 978-84-608-3488-5 *Disponible en:* https://manualsupervivenciamaker.com/manual/

[8] Open-Labware [Consulta: 20 de julio de 2020] *Disponible en:* https://open-labware.net

[9] Artículo Open Labware: Baden T, Chagas AM, Gage G, Marzullo T, Prieto-Godino LL, Euler T. Open labware: 3-D printing your own lab equipment. PLoS Biol. 2015; 13(3):e1002086. https://doi.org/10.1371/journal.pbio.1002086 PMID: 25794301

[10] Beagleboard [Consulta: 20 de julio de 2020] *Disponible en:* https://beagleboard.org

[11] Raspberry Pi [Consulta: 20 de julio de 2020] *Disponible en:* https://www.raspberrypi.org

[12] Felipe Machado. Apuntes de Mecatrónica *Disponible en:* https://zenodo.org/record/4041987

[13] J.C.Torres. *Diseño asistido por ordenador* [Consulta: 8 de julio de 2020] *Disponible en:* https://lsi.ugr.es/~cad/teoria/Tema1/RESUMENTEMA1.PDF

[14] Geometric Modelling: Theorical and Computacional Basic towards Advance CAD Applications.

[15] Jorge D. Camba, Manuel Contero, P. Company. *Parametric CAD modeling: An Analysis of Strategies for Design Reusability*. ScienceDirect, 22 de enero de 2016, ISSN 0010-4485. *Disponible en:* http://repositori.uji.es/xmlui/bitstream/handle/10234/153425/company2016.pdf

[16] Zigurat. Máster BIM manager [Consulta: 14 de julio de 2020] *Disponible en:* http://www.cype.net/pdfs/encuentros/2014/3777_catalogo_mbim.pdf

[17] The STEP standard for Product Data Exchange [Consulta: 29 de agosto de 2020] *Disponible en:* https://asmedigitalcollection.asme.org/computingengineering/article-

*abstract/1/1/102/445236/Introduction-to-ISO-10303-the-STEP-Standard-for?redirectedFrom=fulltext*

[18] ÁNGEL, RUBIO GONZÁLEZ Miguel, et al. Introducción a la Informática básica. Editorial UNED, 2017.

[19] Nvidia: Diferencia entre CPU y GPU [Consulta: 18 de julio de 2020] *Disponible en: https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/*

[20] Intel: Diferencia entre CPU y GPU [Consulta: 18 de julio de 2020] *Disponible en: https://www.intel.com/content/www/us/en/products/docs/processors/cpu-vs-gpu.html*

[21] JORQUERA ORTEGA, Adam. Fabricación digital: Introducción al modelado e impresión 3D. Ministerio de Educación, Cultura y Deporte, 2016.

[22] Alfonso Álvarez, T.S. Automatismos y Robótica Industrial: Fundamentos en impresión 3D con Prusa i3 Steel.

[23] DASSAULT SYSTÈMES: CATIA [Consulta: 8 de julio de 2020] *Disponible en*: *https://www.3ds.com/es/productos-y-servicios/catia/productos/*

[24] Skyfilabs, imagen de la interfaz de CAD. *Disponible en: https://www.skyfilabs.com/project-ideas/latest-projects-based-on-catia*

[25] DASSAULT SYSTÈMES: Portafolio de CATIA [Consulta: 8 de julio de 2020] *Disponible en*: *https://www.3ds.com/es/productos-y-servicios/catia/productos/catia-v5/portfolio/*

[26] 3DSMAN: CATIA Pricing. [Consulta: 14 de julio de 2020] *Disponible en*: *https://3dsman.com/catia-pricing/*

[27] DASSAULT SYSTÈMES: Certified Workstations [Consulta: 14 de julio de 2020] *Disponible en*: *https://www.3ds.com/support/hardware-and-software/hardware-and-software-configurations/?woc=%7B"operating%20system"%3A%5B"operating%20system%2Fwindows%2010%2064-bit"%5D%2C"releases"%3A%5B"releases%2Fv5-6r2020"%5D%7D*

[28] Nvidia: Recomendación de tarjeta gráfica [Consulta: 14 de junio de 2020] *Disponible en*: *https://www.nvidia.com/es-es/design-visualization/quadro/selector/*

[29] AutoDesk: AutoCAD. [Consulta: 14 de julio de 2020] *Disponible en*: *https://www.autodesk.es/products/autocad/features?plc=ACDIST&term=1-YEAR&support=ADVANCED&quantity=1#internal-link-2d-features*

[30] 2acad, imagen de AutoCAD. *Disponible en: https://www.2acad.es/wp-content/uploads/Rendimiento-de-gr%C3%A1ficos-AutoCAD-2021.jpg*

[31] DASSAULT SYSTÈMES: SolidWorks. [Consulta: 14 de julio de 2020] *Disponible en: https://www.solidworks.com/choosing-solidworks*

[32] CIMWORKS: Distribuidor oficial de SolidWorks [Consulta: 14 de julio de 2020] *Disponible en: https://www.cimworks.es/precios-solidworks/*

[33] SolidWorks: blog [Consulta: 14 de julio de 2020] *Disponible en:* https://blogs.solidworks.com/solidworkslatamyesp/solidworks-blog/solidworks/solidworks-2018-herramientas-para-desarrollar-tu-potencial/

*[34]* CADEM: Diferencia entre CATIA y SolidWorks [Consulta: 9 de septiembre de 2020] *Disponible en:* https://www.cadems.es/diferencias-entre-catia-y-solidworks/

[35] DASSAULT SYSTÈMES: Requisitos SolidWorks [Consulta: 14 de julio de 2020] *Disponible en:* https://www.solidworks.es/sw/support/SystemRequirements.html

[36] OpenSCAD [Consulta: 14 de julio de 2020] *Disponible en:* https://www.openscad.org/about.html

[37] Documentación OpenSCAD [Consulta: 14 de julio de 2020] *Disponible en:* https://en.wikibooks.org/w/index.php?title=OpenSCAD_User_Manual/General&oldid=3675948

[38] Programación funcional [Consulta: 14 de julio de 2020] *Disponible en:* https://www.lexico.com/definition/functional_programming

[39] OpenCSG [Consulta: 16 de julio de 2020] *Disponible en:* http://opencsg.org

[40] FreeCAD [Consulta: 15 de julio de 2020] *Disponible en:* https://www.freecadweb.org

[41] FreeCAD: Sobre FreeCAD [Consulta: 15 de julio de 2020] *Disponible en:* https://wiki.freecadweb.org/About_FreeCAD

[42] FreeCAD: Características [Consulta: 15 de julio de 2020] *Disponible en:* https://wiki.freecadweb.org/Feature_list

[43] GitHub: CadQuery [Consulta: 30 de julio de 2020] *Disponible en:* https://github.com/dcowden/cadquery

[44] CadQuery [Consulta: 15 de julio de 2020] *Disponible en:* https://cadquery.readthedocs.io/en/stable/intro.html#

[45] Lenguajes más buscado en Google en el último año [Consulta el 28 de agosto de 2020] *Disponible en:* https://trends.google.es/trends/explore?geo=ES&q=%2Fm%2F05z1_,%2Fm%2F0jgqg,%2Fm%2F07sbkfb,%2Fm%2F02p97,%2Fm%2F03g20

[46] Machado F, Malpica N, Borromeo S (2019) Parametric CAD modeling for open source scientific hardware: Comparing OpenSCAD and FreeCAD Python scripts. PLOS ONE 14(12): e0225795. https://doi.org/10.1371/journal.pone.0225795

[47] GitHub: Repositorio Mechatronic. *Disponible en:* https://github.com/davidmubernal/Mechatronic

[48] ReadtheDocs: Documentación de Mechatronic Workbench. *Disponible en:* https://mechatronic.readthedocs.io/en/master/

[49] YouTube: videos sobre Mechatronic Workbench. *Disponible en:* https://www.youtube.com/playlist?list=PLJAGaIjAPiFIkdTY4OOOegZvmtumLL3OK

[50] GitHub: Felipe Machado – FreeCAD Filter Stage [Consulta: 3 de agosto de 2020] *Disponible en: https://github.com/felipe-m/freecad_filter_stage*

[51] GitHub: Felipe Machado – OpenScad Filter Stage [Consulta: 3 de agosto de 2020] *Disponible en: https://github.com/felipe-m/oscad_filter_stage*

[52] GitHub: David Muñoz – Workbench Mechatronic: Versión 0.1. *Disponible en: https://github.com/davidmubernal/Mechatronic_Documentation/commit/5a8ad975e25c87418b8809f487bfb774f03f7236*

[53] Pololu A4988: Datasheet. *Disponible en: https://www.pololu.com/product/1182*

[54] Clements P, Garlan D, Bass L, Stafford J, Nord R, Ivers J, et al. Documenting software architectures: views and beyond. 2nd Ed. Boston. Pearson Education; 2010.

[55] Documentación ReadtheDocs. [Consulta: 17 de febrero de 2020] *Disponible en: https://docs.readthedocs.io/en/stable/intro/getting-started-with-sphinx.html*

[56] Foro FreeCAD: Publicación Mechatronic Workbench. *Disponible en: https://forum.freecadweb.org/viewtopic.php?f=9&t=44498*

[57] Machado F, Práctica de introducción al diseño electrónico digital de componentes discretos. *Disponible en: https://zenodo.org/record/4064284#.X3iz22gzZGo*

[58] FreeCAD: external workbench *Disponible en: https://wiki.freecadweb.org/External_workbenches*

# 7. Anejos

## Anejo I - Programas CAD

### Programas de licencia propietaria

#### *CATIA*

*CATIA*, diseñado por *DASSAULT SYSTÈMES*, cuenta con distintas funcionalidades para crear y modificar superficies (Shape Design & Styling), realizar el ensamblaje de modelos mecánicos (Mechanical Design), realizar el análisis de estructuras (Infrastructure) e integrar esquemas eléctricos en productos electromecánicos (Electre) entre otros *[23][25]*. En la página oficial de *DASSAULT SYSTÈMES* no se encuentra el coste de este software y para saberlo sería necesario contactar con su departamento de ventas. Según *3DSMan* en 2017 el coste una licencia sería de 11.200$ el primer año y 2.000$ los años consecutivos para mantener la licencia *[26]*.

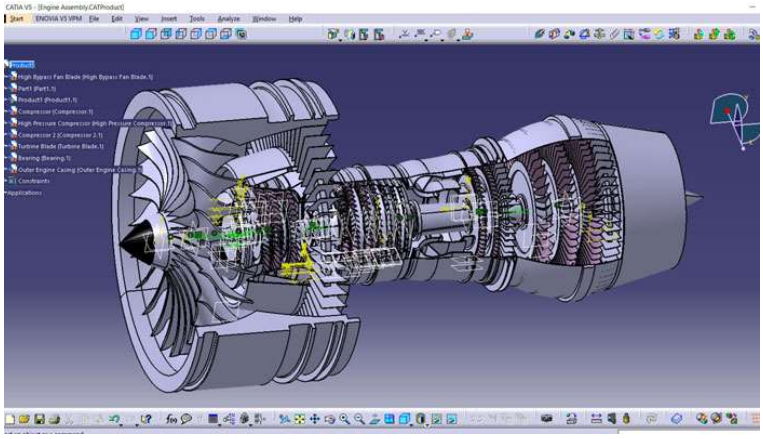

*Figura 34 - Interfaz de CATIA [24]*

Además de este coste, a pesar de estar desactualizado, deberíamos añadir el coste del equipo donde funcionará. Al igual que con el coste, *DASSAULT SYSTÈMES* no aporta unos requisitos mínimos del equipo. Sin embargo, tiene unos equipos validados para el software *[27]*. Cabe destacar que todos los equipos cuentan con Graficas *Nvidia Quadro* o *AMD Radeon Pro*. Este tipo de tarjetas gráficas están diseñadas para mejorar el rendimiento de estos programas lo cual encarece el precio de estas.

Para tener una idea del coste del equipo más económico recomendado se selecciona el ordenador *Dell Precision 3550* con un coste de 1598,57€ y las siguientes características:

- Procesador *Intel Core i7-10510U* a *1.8Ghz*
- Tarjeta gráfica *Nvidia Quadro P520* con *2Gb de VRAM*
- Memoria RAM de 8Gb

#### *AutoCAD*

*AutoCAD* es un software diseñado por *Autodesk* que permite realizar un diseño convencional y un diseño paramétrico añadiendo restricciones geométricas o por cota. *AutoCAD* está disponible para *Windows* y *macOS*.

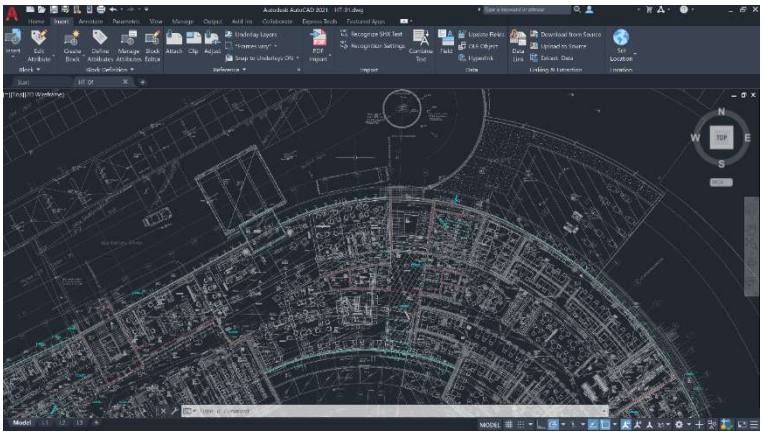

*Figura 35 - Interfaz de AutoCAD [30]*

Además, cuenta con la posibilidad de realizar la renderización de los modelos 3D para obtener una apariencia más realista.

Existen también otros conjuntos de herramientas en función de los sectores específicos de la industria: *AutoCAD MAP*, *AutoCAD Electrical*, *AutoCAD Architecture*, *AutoCAD MEP*, *AutoCAD Plant 3D*, *AutoCAD Mechanical*.

El coste de este software es de 279€ al mes, 2.227€ al año o 6.014€ durante tres años.

A este coste, habría que sumarle el coste del equipo básico con los siguientes requisitos mínimos *[29]*:

- Procesador de 2.5 GHz
- 8Gb de memoria RAM
- Tarjeta gráfica con un ancho de banda de 29 Gb/s y 1Gb de memoria dedicada VRAM.

El equipo básico que cumple estas propiedades se encuentra en la *Tabla 13* y tiene un coste de 730,74€

*Tabla 13 - Presupuesto de equipo básico AutoCAD*

| Componente | Modelo | Coste |
|---|---|---|
| **Procesador** | Intel Core i3-9100F | 67,99€ |
| **Tarjeta gráfica** | Nvidia Quadro P400 2Gb | 151,00€ |
| **Memoria RAM** | Kingston HyperX Fury 8Gb | 34,00€ |
| **Placa Base** | Gigabyte H310M | 55,99€ |
| **SSD** | Kingston A400 de 240Gb | 33,99€ |
| **HDD** | Seagate Barracuda 1Tb | 37,99€ |
| **Caja** | Tacens Anima USB 3 | 18,98€ |
| **Fuente de alimentación** | Cooler Master 500W | 46,98€ |
| **Monitor** | Philips 21.5” 1080p | 84,99€ |
| **Teclado y ratón** | L-Link Combo teclado + ratón | 5,99€ |
| **Sistema Operativo** | Windows 10 | 117,95€ |
| **Montaje** | | 44,94€ |
| **Instalación del Sistema Operativo** | | 29,95€ |
| | **Total:** | 730,74€ |

### *SolidWorks*

*SolidWorks* es un software diseñado por *DASSAULT SYSTÈMES* que cuenta con soluciones específicas para cada sector de la industria y está disponible para *Windows* y *macOS [31]*.

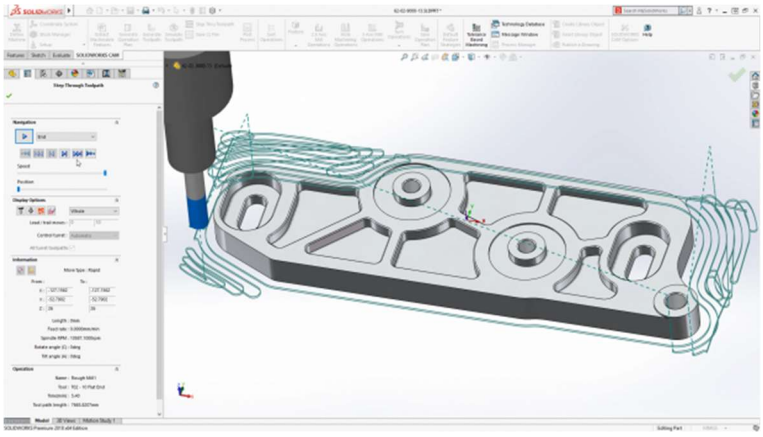

*Figura 36 - Interfaz de SolidWorks [33]*

El coste de la versión estándar es de 6.600€ más 1.500€ al año para mantener la licencia *[32]*.

Para este software el equipo recomendado debe tener los siguientes requisitos *[35]*:

- Procesador de 3.3 GHz
- 16 Gb de memoria RAM
- Tarjeta gráfica *Nvidia Quadro* o *AMD Radeon Pro*

El equipo básico que cumple estas propiedades se encuentra en la *Tabla 14* y tiene un coste de 764,74€

*Tabla 14 - Presupuesto de equipo básico SolidWorks*

| Componente | Modelo | Coste |
|---|---|---|
| **Procesador** | Intel Core i3-9100F | 67,99€ |
| **Tarjeta gráfica** | Nvidia Quadro P400 2Gb | 151,00€ |
| **Memoria RAM** | Kingston HyperX Fury 2x8Gb | 68,00€ |
| **Placa Base** | Gigabyte H310M | 55,99€ |
| **SSD** | Kingston A400 de 240Gb | 33,99€ |
| **HDD** | Seagate Barracuda 1Tb | 37,99€ |
| **Caja** | Tacens Anima USB 3 | 18,98€ |
| **Fuente de alimentación** | Cooler Master 500W | 46,98€ |
| **Monitor** | Philips 21.5” 1080p | 84,99€ |
| **Teclado y ratón** | L-Link Combo teclado + ratón | 5,99€ |
| **Sistema Operativo** | Windows 10 | 117,95€ |
| **Montaje** | | 44,94€ |
| **Instalación del Sistema Operativo** | | 29,95€ |
| | **Total:** | 764,74€ |

## Programas gratuitos

### *OpenSCAD*

*OpenSCAD* es un *OSS* diseñado para la creación de sólidos 3D. Está disponible para distribuciones *Linux*, *Windows* y *macOS* a través de su página web bajo la licencia *GPLv2*.

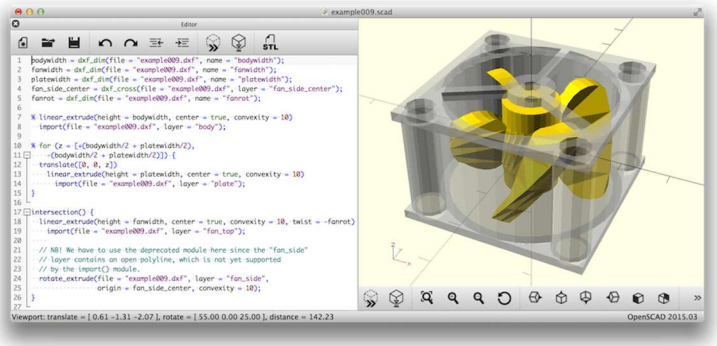

*Figura 37 – OpenSCAD [36]*

Cabe destacar que la interfaz de este software tiene dos ventanas. En la ventana izquierda hay un editor de texto donde se debe describir el modelo. En la ventana de la derecha se encuentra el espacio de trabajo con el modelo 3D renderizado. Para crear el modelo 3D tenemos la opción de generar sólidos 3D y unirlos con operaciones booleanas o la opción de generar formas en 2D y extruirlas, generar un volumen con ellas *[36][37]*.

Para describir los modelos se emplea un lenguaje de programación funcional (Functional Programming language) propio del software que consiste en secuencias de funciones matemáticas, en lugar de un conjunto de comandos *[38]*. De esta forma se reducen las líneas necesarias para escribir una acción y se vuelve más legible para aquellas personas que no tengan conocimientos en programación. La previsualización de los modelos se realiza mediante *OpenCSG*, el cual funciona correctamente con gráficas integradas, lo que supone un menor coste del equipo [39]. Además, al usar *OpenCSG* los modelos se generan a partir de operaciones booleanas con formas básicas. *OpenSCAD* permite exportar el modelo en formatos tales como *STL* y *DXF* pero no en formato *STEP*.

El equipo necesario puede ser bastante económico ya que es posible ejecutar el programa en una Raspberry Pi 4 con 4Gb de memoria RAM. Sin embargo, para asegurar que se tenga la suficiente memoria RAM como para realizar ensamblajes de muchos modelos se elige la *Raspberry Pi* con 8GB de memoria RAM.

*Tabla 15 - Presupuesto OpenSCAD*

| Componente | Modelo | Coste |
|---|---|---|
| **Placa** | Raspberry Pi 4 con 8GB de RAM | 83,99€ |
| **Tarjeta SD** | MicroSD 32Gb | 13,00€ |
| **Fuente de alimentación** | Fuente de alimentación oficial 15.3W USB-C | 8,95€ |
| **HDMI a micro-HDMI** | Adaptador | 2,40€ |
| **Monitor** | Philips 21.5” 1080p | 84,99€ |
| **Teclado y ratón** | L-Link Combo teclado + ratón | 5,99€ |
| | **Total:** | 199,62€ |

### *FreeCAD*

*FreeCAD* es un proyecto de *OSS* que nace en 2001 dirigido a la ingeniería mecánica y el diseño de productos. Cuenta con las siguientes características *[40][41]*:

- Modelos paramétricos
- Posibilidad de añadir módulos programados en *C++* o *Python*
- Capacidad para importar o exportar en distintos formatos estándar: *STEP*, *OBJ*, *DXF*, *SVG* y *STEP* entre otros.

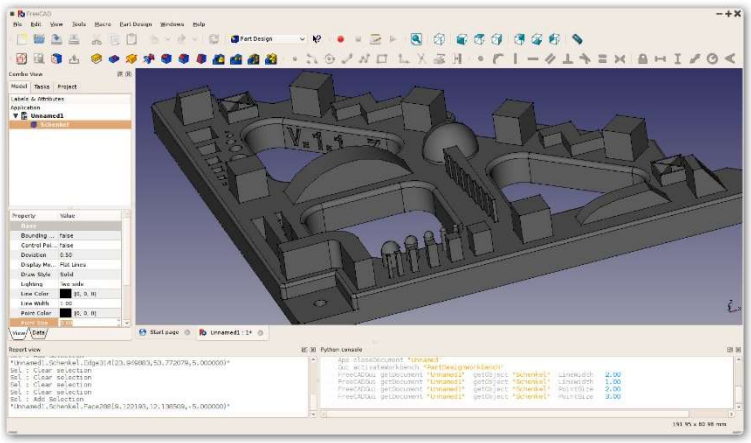

*Figura 38 - FreeCAD [41]*

También cuenta con distintos módulos para realizar dibujos en 3D, simular el movimiento de un brazo robótico, dibujo técnico para realizar vistas de modelos, módulo de arquitectura que permite metodología BIM y fabricación asistida por ordenador (CAM) *[42]*. La interfaz básica de *FreeCAD*, que se puede ver en la ilustración 4, consta de una ventana central con la representación gráfica en 3D del modelo y una ventana lateral con los modelos 3D en vista de árbol. Si se selecciona un modelo de la vista de árbol veremos en la parte inferior de la misma ventana las propiedades que tiene el modelo.

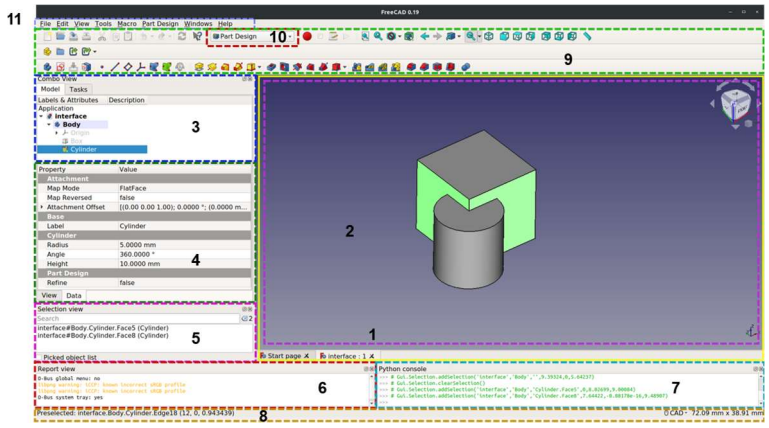


*Figura 39 - Interfaz de FreeCAD [42]*

Adicionalmente, FreeCAD tiene distintas ventanas para tener un mayor control del software que se ven representadas en la *Figura 39*. La ventana 6 nos permite ver el informe del software, en ella veremos

distintos mensajes sobre el funcionamiento de las operaciones que se realicen. La ventana 7 es la consola de *Python*, desde esta ventana se puede interactuar con el software desde comandos. La consola de *Python* resulta útil para un usuario con conocimientos de programación que requiera de un mayor control.

Además, está basado en librerías de *OSS*:

- Empleo del *kernel* Open Cascade Technology (*OCCT*)
- Herramienta de gráficos 3D Coin3D
- Interfaz gráfica desarrollada con *Qt*
- Posibilidad de escribir código en *Python*

*Tabla 16 - Presupuesto FreeCAD*

| Componente | Modelo | Coste |
|---|---|---|
| **Placa** | Raspberry Pi 4 con 8GB de RAM | 83,99€ |
| **Tarjeta SD** | MicroSD 32Gb | 13,00€ |
| **Fuente de alimentación** | Fuente de alimentación oficial 15.3W USB-C | 8,95€ |
| **HDMI a micro-HDMI** | Adaptador | 2,40€ |
| **Monitor** | Philips 21.5” 1080p | 84,99€ |
| **Teclado y ratón** | L-Link Combo teclado + ratón | 5,99€ |
| | **Total:** | 199,62€ |

### *CadQuery*

*CadQuery* nació como un módulo de *FreeCAD*, aunque después se convirtió en un programa independiente *[43]*. *CadQuery* cuenta con una librería de *Python* para construir modelos en 3D. Al igual que *OpenSCAD*, la generación de modelos se realiza mediante un lenguaje de programación en el cual se describe el modelo. Está basado en el kernel *OCCT* y permite exportar los modelos en formato *STEP* y *STL*. Con este software se pueden realizar modelados paramétricos *[44]*.

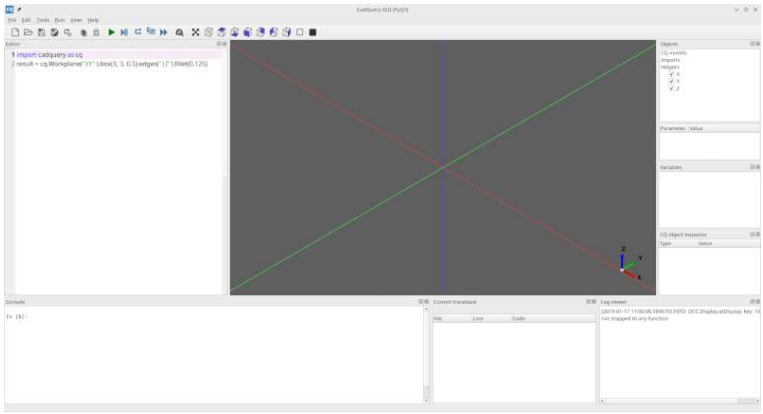

*Figura 40 - CadQuery [44]*

# Anejo II - Versiones del GUI

Para entender bien cada parte del *GUI* diseñado se va a realizar el análisis de este desde la primera versión hasta la versión final. Se valorarán todos los cambios en un modelo con interfaz sencilla, *SK*, y compleja, *Aluprof Bracket*. Además, se verán los nuevos modelos añadidos y funcionalidades.

## *Versión 0.1 (V-0.1)*

Aunque esta versión tiene fecha el 29 de junio de 2019 en el repositorio, los archivos a los que hace referencia fueron publicados el 26 de marzo de 2019 *[52]*. Esta versión funcionaba con *FreeCAD 0.17* sobre el lenguaje *Python 2.7* y contaba con 6 modelos y la función para imprimirlos en formato *STL*.

*Figura 41 - ToolBar V-0.1 (Elaboración propia)*

En esta versión el menú era muy sencillo permitiendo modificar los modelos de forma muy básica.

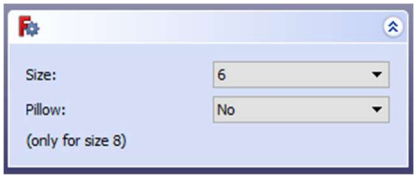


*Figura 42 - Ventana Tasks SK V-0.1 (Elaboración propia)*

En modelos con mayor complejidad y variables dependientes entre sí era difícil de comprender cuales tendrían efecto sobre el modelo o no. Por ejemplo, en la *Figura 43*, al seleccionar el tipo *2 profiles*, podríamos modificar la opción *Reinforce*. Sin embargo, las opciones *Flap* y *Dist between profiles* no tendrían ningún efecto sobre el tipo de modelo que hemos seleccionado.

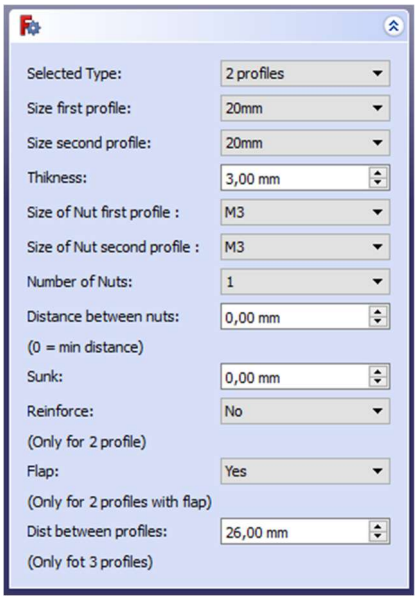


*Figura 43 - Ventana Tasks Aluprof Bracket V-0.1 (Elaboración propia)*

### *Versión 0.1.4 (V-0.1.4)*

En esta versión, con fecha 29 de junio de 2019, se añaden 4 modelos y un sistema completo. Además, los iconos se han cambiado al color predefinido por *FreeCAD* para los modelos *CAD* (amarillo) y las funciones (azul). Sin embargo, el cambio más importante en esta versión es la actualización a *FreeCAD 0.18*. Esta actualización supuso pasar de *Python 2.7* a *Python 3.6*, por lo que hubo que modificar gran parte del programa. Al cambiar la versión del lenguaje se debe proceder a la actualización de los conocimientos, es decir, se debe revisar la documentación del lenguaje para ver los cambios y poder aplicarlos. Este cambio de versión supuso la modificación de cómo se debía llamar a unas funciones, lo que supuso revisar todo el trabajo realizado hasta el momento para adaptarlo.

*Figura 44 - ToolBar V-0.1.4 (Elaboración propia)*

### *Versión 0.2.0 (V-0.2.0)*

Esta versión, con fecha 16 de julio de 2019, incluye un nuevo modelo, pero su mayor novedad es la función *Assembly* representado con el icono de la letra "A".

*Figura 45 - ToolBar V-0.2.0 (Elaboración propia)*

Esta función permite realizar el ensamblaje de modelos de forma más sencilla. Esta función necesita que selecciones dos modelos o dos líneas que pertenezcan a estos. El primer modelo seleccionado será el que se mueva, mientras que el segundo modelo seleccionado será el que marque la posición a la que se mueve el primer modelo.

Por ejemplo, se quiere mover el modelo *Aluprof Bracket*, situado a la derecha, encima del *Sk*, situado a la izquierda. Para ello seleccionamos primero la línea inferior del agujero central del *Bracket* y seguidamente, manteniendo la tecla *CONTROL* pulsada, seleccionamos la línea superior del agujero central del *Sk* (*Figura 46*). Clicamos en el botón "*OK*" y la pieza *Bracket* se situará en su nueva posición como se puede ver en la *Figura 47*.

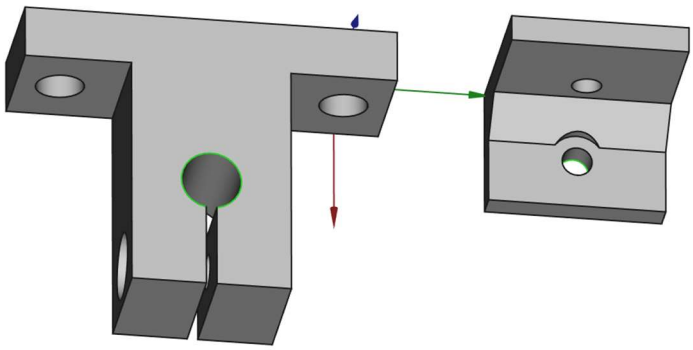

*Figura 46 - Selección del modelo a mover y su posición (Elaboración propia)*

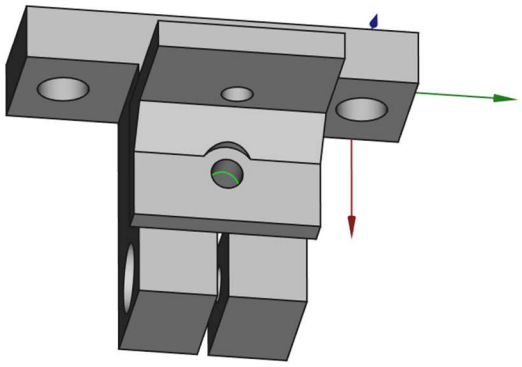

*Figura 47 - Modelos en su nueva posición (Elaboración propia)*

### *Versión 0.2.1 (V-0.2.1)*

Después de la actualización anterior ya se permitía hacer ensamblajes, pero para montar un sistema entero hacen falta elementos que hagan de soportes y métodos de unión. Por eso, el 12 de septiembre de 2019 se realiza una actualización para añadir modelos que, aunque no están pensados para imprimirse, serán útiles para realizar dichos ensamblajes. Dichos modelos se corresponden con un perfil de aluminio, tornillos, arandelas y tuercas. Todos estos elementos se usan en el modelo ensamblado del *Filter Stage*.

*Figura 48 - ToolBar V- 0.2.1 (Elaboración propia)*

### *Versión 0.2.2 (V-0.2.2.)*

En esta versión, además de los modelos con finalidad mecánica, se introducen un conjunto de modelos con finalidad óptica.

*Figura 49 - ToolBar V-0.2.2 (Elaboración propia)*

### *Versión 0.2.3. (V-0.2.3.)*

Esta versión no añade modelos, pero supone un cambio importante en el *GUI* permitiendo al usuario elegir la posición y orientación de los modelos. Además, podrá fijar el centro del modelo en el punto interno que desee. También, se añade una *URL* para consultar la imagen que se encuentra disponible en línea con los distintos puntos internos y direcciones de los ejes.

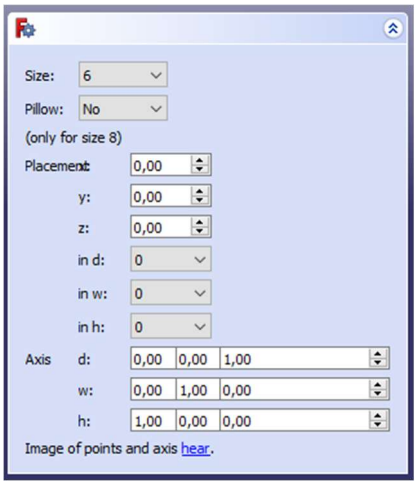


*Figura 50 - Ventana Tasks SK V-0.2.3. (Elaboración propia)*

Como se puede apreciar, tanto en la *Figura 50* como en la *Figura 52*, el *GUI* de esta versión, a pesar de permitir la colocación y orientación de los modelos, no representa de forma óptima y uniforme los distintos campos. Esto se debe al tipo de *layout* que se encargaba del posicionamiento en las versiones anteriores y en esta. Dicho *layout* genera una cuadrícula donde se deben colocar los widgets que se quieren visualizar. Dicha cuadrícula puede no ser homogénea a lo largo del *GUI*, es decir, que puede cambiar el número de elementos que hay en cada fila. Por ejemplo, en el *GUI* de la *Figura 50* se podría representar con la cuadrícula de la *Figura 51*. Pero este tipo de *layout* no distribuye el espacio total del menú entre todos los widgets que contiene cada fila, por ese motivo se pueden ver widgets en una misma fila con tamaños muy distintos.

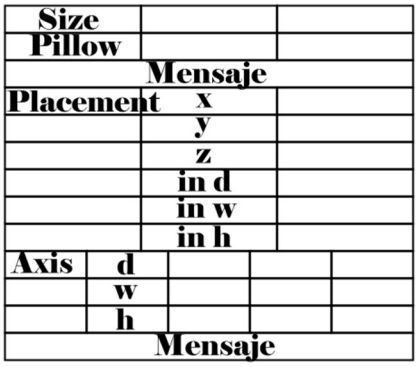


*Figura 51 - Cuadrícula del GUI (Elaboración propia)*

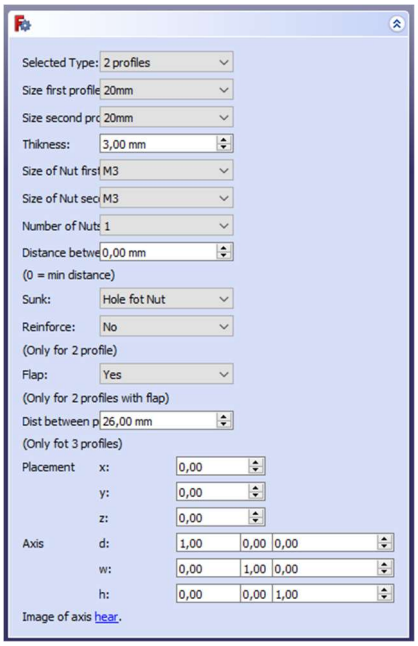


*Figura 52 - Ventana Tasks Aluprof Bracket V-0.2.3. (Elaboración propia)*

### *Versión 1.0.0 (V-1.0.0)*

Con la versión 1.0.0, la primera versión que tiene los requisitos que se querían alcanzar, el *GUI* cambia en gran medida.

Ahora la ventana *Tasks* tiene dos menús distintos. El primero es similar al visto en la versión anterior, pero cambia la forma en la que se disponen los widgets para mejorar la interacción con ellos. En la versión anterior hablamos de un *layout* que generaba una cuadrícula. En esta versión se usan dos tipos de *layout* que generan una línea (fila o columna) donde podremos colocar los widgets que se quieren visualizar. Por ejemplo, para generar un diseño como el representado en la *Figura 53* usando el *layout* de cuadrícula todos los widgets deberán ocupar el mismo espacio sin tener en cuenta su contenido. Sin embargo, usando el sistema de *layout* de cajas se deberán generar distintos *layouts* unos dentro de otros. Es decir, habrá un *layout* principal vertical que contenga 5 *layouts* horizontales. Estos últimos serán los que contengan los widgets a mostrar. De este modo, el espacio ocupado por los widgets dependerá del tamaño del *layout* secundario, los horizontales, y el tamaño de estos del *layout* principal, el vertical.

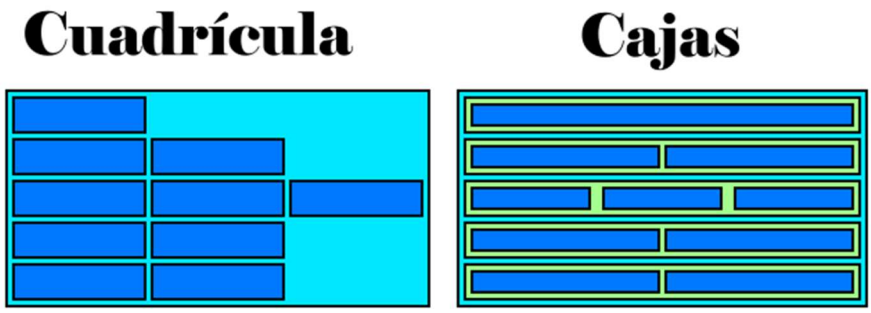


*Figura 53 - Comparativa de layout (Elaboración propia)*

También se genera una lógica sobre el *GUI* que permite habilitar o deshabilitar opciones. Es decir, la opción *Pillow* sólo se encuentra disponible para el tamaño 8, por tanto, si el tamaño seleccionado no es "8" no se podrá interactuar con esta opción. Con este sistema se impide que el usuario pueda generar errores.

Además, se incorpora una función que enlaza la posición que se quiere fijar en el modelo con la posición del cursor del ratón sobre la ventana de visualización de *FreeCAD*. De este modo, el usuario no necesita introducir la posición haciendo uso del teclado, simplemente debe clicar sobre la posición en la que desea colocar el nuevo modelo (cualquier punto del espacio, vértice, línea o cara de un modelo ya existente) y dicha posición quedará fijada. En el caso de que el usuario quiera cambiar de posición deberá volver a clicar y el seguimiento del cursor volverá a habilitarse.

El segundo menú, *Advance Placement*, sólo tendrá efecto cuando exista un modelo con puntos internos. Con este menú el usuario podrá seleccionar un modelo ya creado y sus puntos característicos. Si se clica sobre *Show point* se mostrará en pantalla un punto verde en la posición seleccionada por el usuario. El botón *Set this position to the model* toma la posición del punto seleccionado y la introduce como posición del modelo que se va a generar.  Con este menú se pueden generar ensamblajes de modelos de forma rápida y directa.

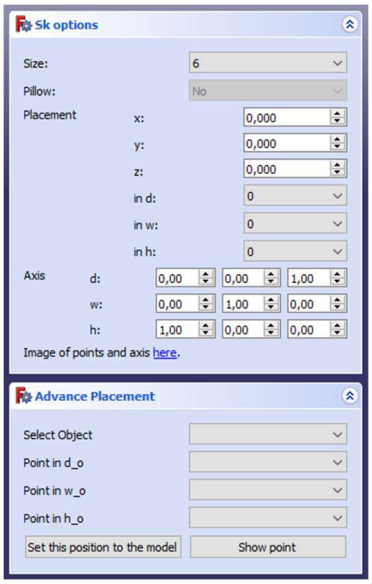


*Figura 54 - Ventana Tasks SK V-1.0.0 (Elaboración propia)*

## Anejo III - Ensamblaje del sistema Filter Stage

En el apartado *4.2* se detalla brevemente cómo realizar el ensamblaje del sistema *Filter Stage*. Con el fin de proporcionar una guía de ensamblaje completa, se detalla a continuación el paso a paso del ensamblaje del sistema:

1. Fijar la **guía lineal** al **perfil de aluminio 15x200mm**.

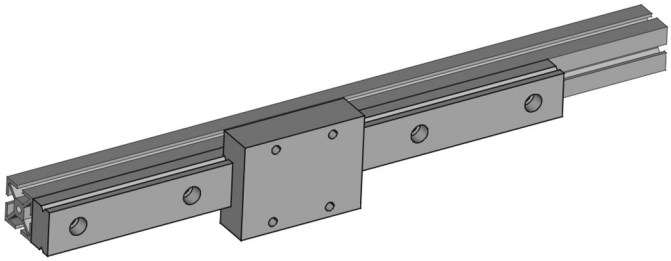

*Figura 55 - Paso 1: Guía lineal sobre perfil 15x200mm (Elaboración propia)*

2. Fijar el **soporte del motor** a uno de los **perfiles de aluminio 15x150mm**.

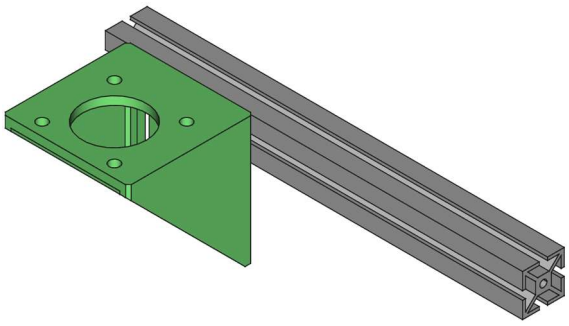

*Figura 56 -Paso 2: soporte motor sobre perfil 15x150mm (Elaboración propia)*

3. Fijar el **soporte del tensionador** al **perfil de aluminio 15x150mm** restante.

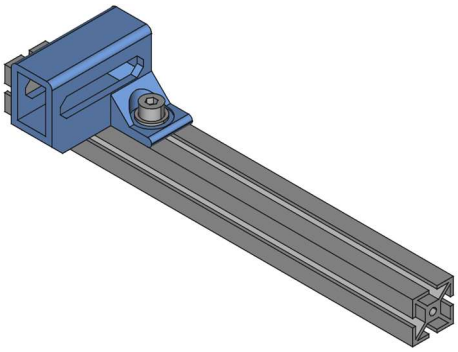

*Figura 57 - Paso 3: soporte del tensionador sobre perfil 15x150mm (Elaboración propia)*

4. **Unir** los tres **perfiles de aluminio** entre sí.

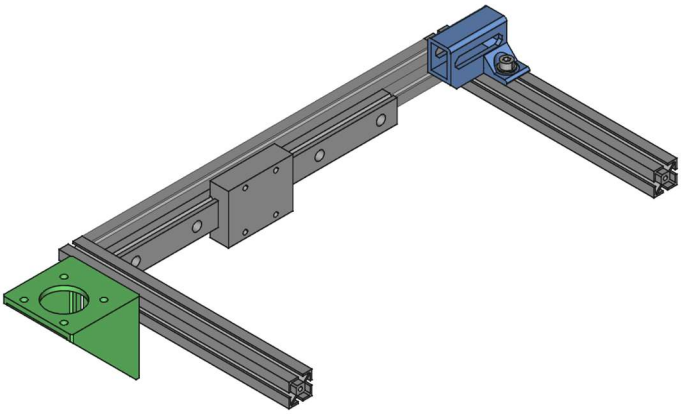

*Figura 58 - Paso 4: unión de los 3 perfiles (Elaboración propia)*

5. Añadir los cuatro **perfiles de aluminio 15x50mm** al sistema como patas para soportar la estructura.

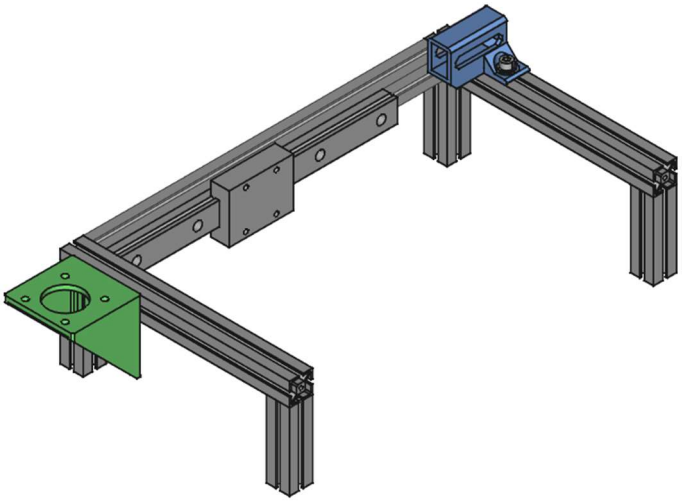

*Figura 59 - Paso 5: modelo con perfiles de apoyo (Elaboración propia)*

6. Montar el **soporte del filtro** sobre la **Guía lineal**.

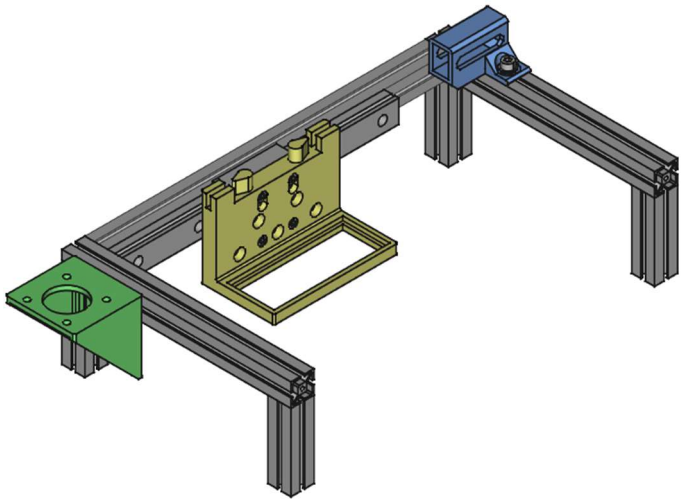

*Figura 60 - Paso 6: montaje del soporte del filtro (Elaboración propia)*

7. Montar la **polea** en el **tensionador** y el tensionador sobre el **soporte del tensionador**.

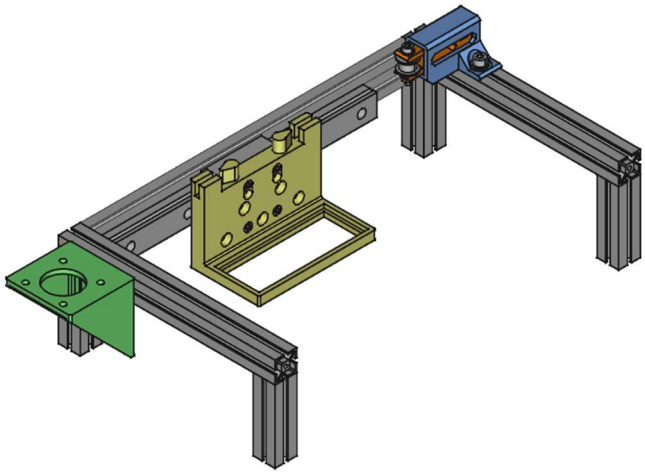

*Figura 61 - Paso 7: montaje del tensionador y la polea en su soporte (Elaboración propia)*

8. Montar el **motor** en el **soporte del motor**.

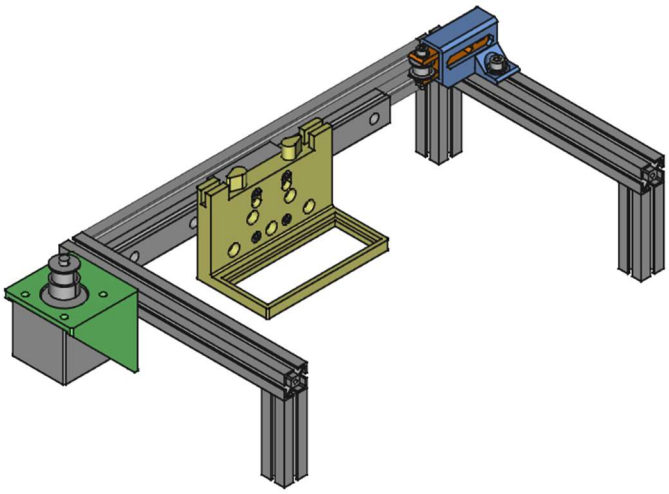

*Figura 62 - Paso 8: montaje del motor (Elaboración propia)*

9. Colocar la **correa dentada** entre el tensionador y el motor.

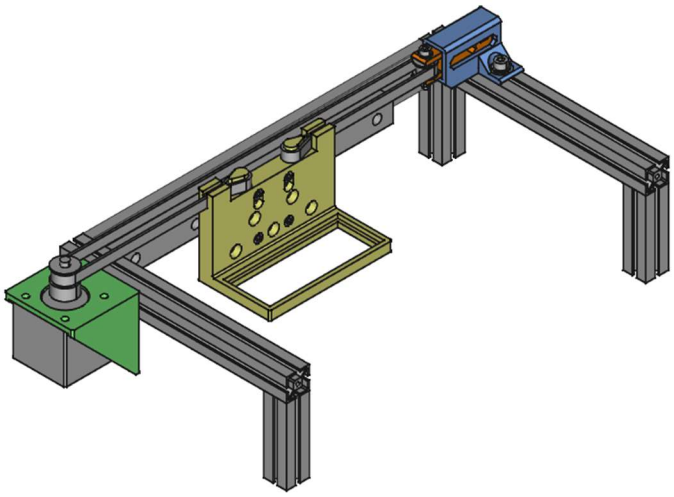

*Figura 63 - Paso 9: montaje de la correa (Elaboración propia)*

10. Situar un **sensor final de carrera** en cada extremo de la **guía lineal**.

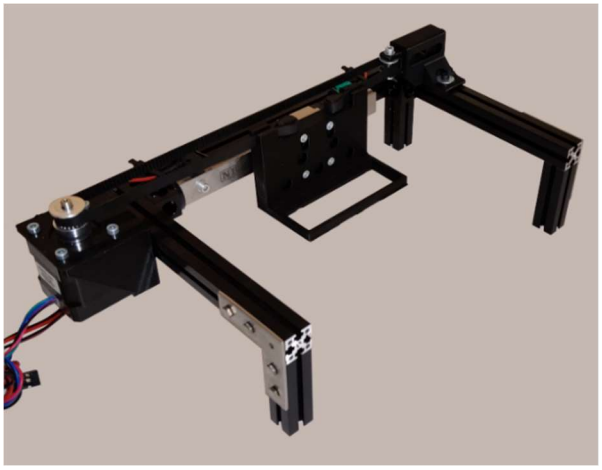

*Figura 64 - Montaje final (Elaboración propia)*