Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

Visor 3D para placas de circuito impreso

Autor:
Álvarez-Campana Medina, Daniel

Tutor:
Mena Rodríguez, José Manuel
Departamento de Tecnología Electrónica

Valladolid, Julio 2021

Agradecimientos

Quisiera agradecer a todas las personas que me han brindado su apoyo durante la realización de este Trabajo de Fin de Grado:

A mi tutor, José Manuel, por la guía y dedicación que me ha aportado durante este proyecto.

Y también a mis amigos y familia, que me han ayudado cuando los he necesitado y que sin su apoyo emocional y comprensión, no hubiese logrado nunca llegar tan lejos.

Resumen

El objetivo final de este TFG es obtener un modelo tridimensional de una placa de circuito impreso, a partir de un archivo generado con el programa MicroSim 8.0 con formato .pca, el cual sólo puede diseñar PCBs en dos dimensiones. Haremos uso del programa KiCad, que cuenta con una funcionalidad que nos permitirá visualizar en tres dimensiones el modelo ya terminado, siempre que contenga toda la información necesaria en un archivo de formato .kicad_pcb. Para ello, hemos creado una herramienta que permite leer toda la información requerida del archivo generado por MicroSim, y reescribirla de manera automática en dicho formato, añadiendo todos los datos adicionales de los modelos tridimensionales de los componentes que se encuentren en su librería interna. Por último, la aplicación también permite generar y ampliar dicha librería de componentes y modelos, de forma que una vez añadidos, los reconocerá y procesará automáticamente para todas las futuras placas.

Palabras clave

Placa de circuito impreso, MicroSim PSpice, KiCad, visor 3D, componente electrónico.

Abstract

The final objective of this paper is to obtain a three-dimensional model of a printed circuit board, from a file generated with MicroSim 8.0 in .pca format, which can only design two-dimensional PCBs. We will make use of KiCad, which has a tool that allow us to visualize the finished model in three dimensions, as long as it contains all the necessary information in a .kicad_pcb format. For this purpose, I have created an application that allows to read all the required information from the file generated by MicroSim, and rewrite it automatically in that format, adding all the additional data for the three-dimensional models of the components included in its internal library. Finally, the application also allows to generate and expand the library of components and models so that, once added, it will automatically recognize and process them for all future boards.

Keywords

Printed circuit board, MicroSim PSpice, KiCad, 3D Viewer, electronic component.

Índice general

Índice de figuras

Índice de tablas

1. Objetivos

MicroSim PSpice es un programa de diseño electrónico que cuenta con varias funcionalidades entre las que destacan la simulación de circuitos, analógicos y digitales, optimización de circuitos, representación gráfica de ondas y la edición de placas de circuitos impresos. Por ello, constituye una herramienta fundamental tanto en el ámbito profesional, como en el educativo.

Nuestro objetivo es comprender mejor cómo se realiza el diseño de placas de circuito impreso y como se trasmite toda esa información. Para ello, examinaremos y analizaremos los archivos con el formato ".pca" que genera la aplicación PCBoards de MicroSim PSpice, que son los archivos finales que contienen toda la información necesaria para la impresión del circuito.

La dificultad radica en que MicroSim 8.0 es un software privado y la información no es tan accesible. Para ayudarnos en nuestro objetivo, vamos a ayudarnos de otro programa de diseño electrónico llamado KiCad. La principal ventaja de éste último es que, a diferencia de MicroSim, KiCad sí que es un programa de código abierto, de manera que podremos ir comparando y apoyándonos para analizar los archivos finales.

De manera práctica, y complementaria, vamos a realizar el desarrollo de un programa que convierta los archivos finales del diseño del circuito impreso de MicroSim de formato ".pca" en formato ".kicad_pcb" para que lo podamos leer desde KiCad. Este programa dispone de un visor de circuitos en tres dimensiones, lo que puede resultar una gran ayuda para los estudiantes a la hora de visualizar el resultado final y en el momento del montaje de componentes.

2. Diseño Electrónico

En la rama de la electrónica, una placa de circuito impreso es una superficie formada por rutas o pistas de un material conductor laminadas sobre una base no conductora. El circuito impreso es utilizado para conectar eléctricamente a través de las pistas un conjunto de componentes electrónicos. Generalmente las pistas están fabricadas de cobre, mientras que la base está constituida de resinas, cerámica, plástico, teflón u otros polímeros.

En el diseño de placas de circuito impreso se pueden distinguir 5 pasos claros antes de alcanzar el producto final. Dichas etapas son: Establecer las especificaciones de nuestro diseño, crear el circuito electrónico que cumpla los objetivos, realizar una serie de simulaciones y pruebas para comprobar el correcto funcionamiento, construcción de un prototipo físico, y por ultimo, volver a realizar de nuevo las pruebas y tests correspondientes a fin de garantizar su correcto funcionamiento.

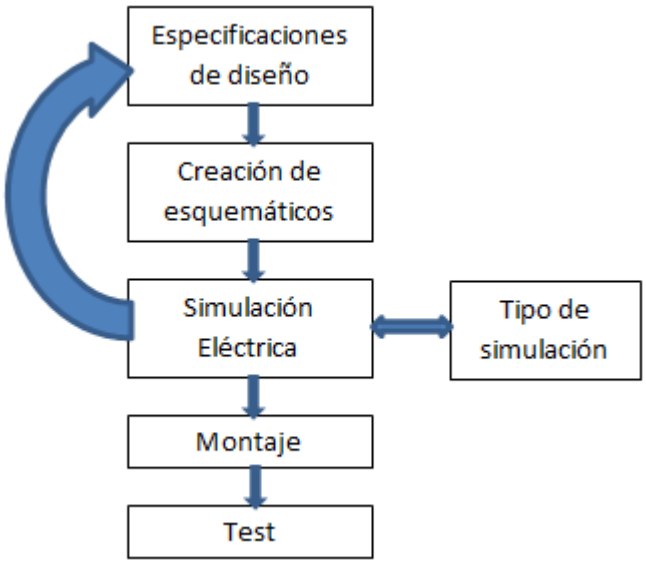

Ilustración 1: Pasos del diseño de PCBs

a) Especificaciones de diseño: Es fundamental definir bien las bases para no tener problemas durante su diseño. Empezaremos por comprender el problema y analizar los requerimientos del cliente/usuario. Después estableceremos de manera precisa el cometido de nuestra placa de circuito impreso y las especificaciones concretas que debemos cumplir. En caso necesario también estudiaremos diversos factores, tales como el espacio disponible que ocupara nuestro PCB, el presupuesto o la gama de componentes a utilizar. Otras consideraciones pueden ser el rendimiento energético, o el plazo del que disponemos. Valorando todos estos factores, realizaremos diversas propuestas de diseño, eligiendo la que más se ajuste a nuestras necesidades.

b) Creación de esquemáticos: Una vez aclarado el diseño realizaremos un diagrama de flujo que defina los distintos bloques que compondrán nuestra placa, como podrían ser la fuente de alimentación, sensores, conversores, filtros o actuadores por nombrar algunos. A continuación iremos bloque a bloque definiendo su diseño a nivel circuito. Exploraremos diversas topologías y calcularemos componentes de los circuitos individuales, investigando y seleccionando el componente que mejor se adecúe, teniendo en cuenta que tienen que interactuar y ser compatible con el resto de bloques, para mantener una coherencia y un correcto funcionamiento. Para nuestra suerte, hoy en día existen una gran variedad de programas de diseño electrónico que nos facilitan en gran medida nuestro trabajo, como pueden ser los dos con los que vamos a trabajar: MicroSim y KiCad. Ambos cuentan con una gran variedad de librerías de componentes con todas sus propiedades bien definidas, que nos permitirán luego realizar las simulaciones. En caso de que el componente no esté definido en las librerías básicas, siempre podremos buscarlo online o crear nosotros mismos su modelo.

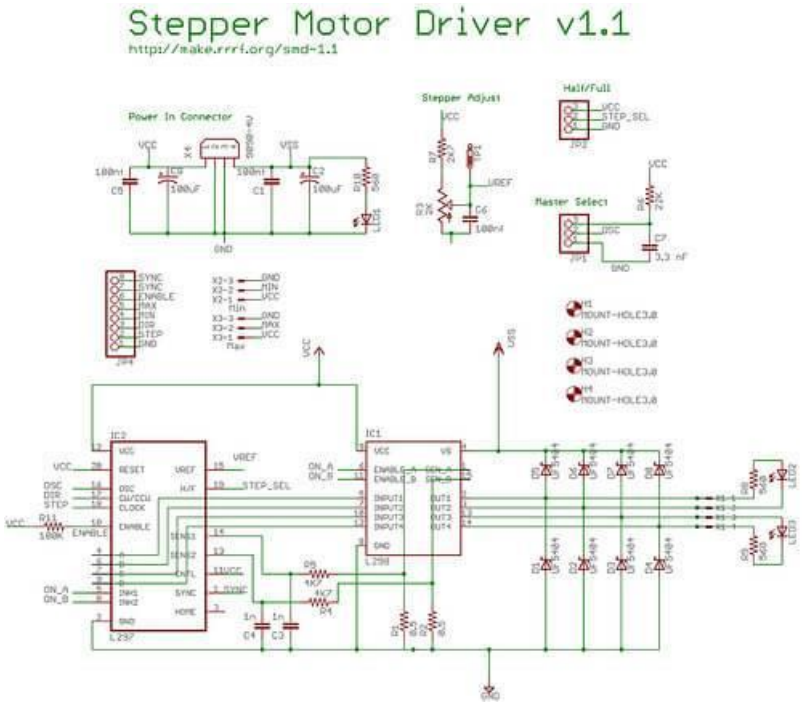

Ilustración 2: Esquemático de un motor paso a paso

c) Simulación del circuito: Este paso consiste en comprobar y asegurar que nuestro circuito cumple con todos los requerimientos definidos en la primera fase. Para ello diseñaremos una serie de simulaciones y test para verificar una a una todas las especificaciones. Debemos comprobar tanto el funcionamiento de cada bloque por separado, como el del circuito completo. Los programas de diseño vienen equipados con una amplia variedad de test y pruebas, que nos permiten simular el comportamiento real de un circuito en determinadas condiciones establecidas por el usuario. Esto facilita enormemente el trabajo de los ingenieros, al no tener que realizar una gran cantidad de tediosos cálculos y minimizando así el error humano. Deberemos analizar los resultados obtenidos, comprobando que concuerdan con lo esperado. En caso de no estar conformes, se produce un reajuste del diseño y la repetición de las pruebas hasta que todo sea satisfactorio.

d) Montaje: Estos programas cuentan con una aplicación para el diseño de la placa de circuito impreso. Deberemos definir el tamaño y forma de la placa y colocar de manera ordenada nuestros

componentes, para facilitar en mayor medida el auto enrutado. Las pistas son uniones de cobre que conectan de manera física los pines de los componentes, creando así los circuitos. El principal problema reside en que estas pistas no se pueden intersecar, porque crearía cortocircuitos y un mal funcionamiento. En caso de no ser posible, utilizaríamos pistas por ambas caras de la placa. Una vez resuelta la disposición de la placa con todos sus componentes, sólo nos queda realizar el prototipo físico.

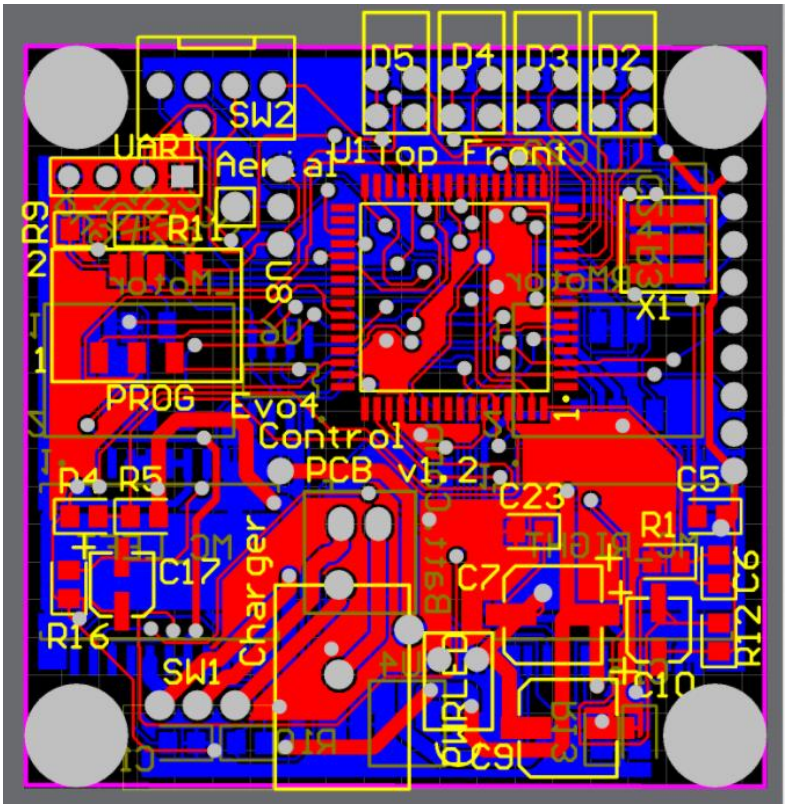

Ilustración 3: Diseño terminado de una PCB

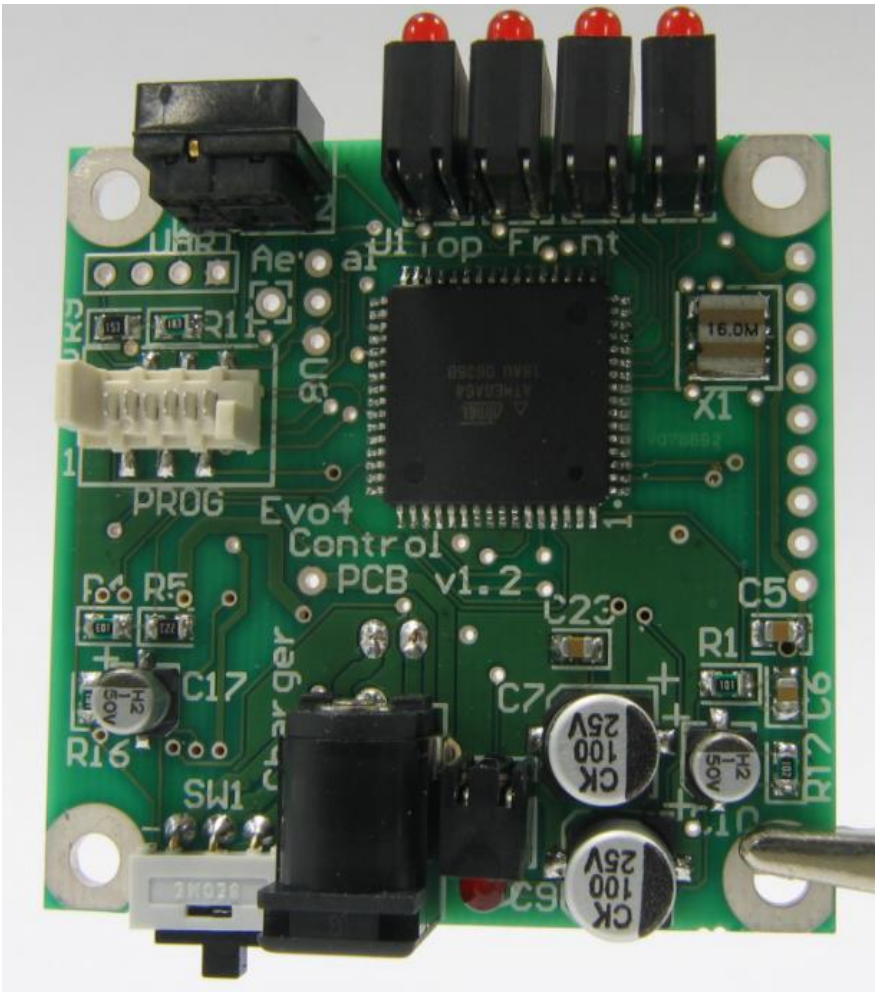

Ilustración 4: Prototipo PCB

e) Test de comprobación: Deberemos verificar el emplazamiento de todos los componentes y sus conexiones. Para finalizar volveremos a realizar todos los test necesarios para reafirmar el cumplimiento de requisitos y objetivos marcados y comprobar que coincide con los resultados obtenidos en la simulación. En caso de que algún elemento no cumpla las expectativas, realizaremos un estudio para determinar la causa y efectuaremos un rediseño del circuito, retornando a la fase de diseño de esquemáticos.

3. MicroSim PSpice

3.1. Historia

El origen se produce en la Universidad de California, Berkeley, cuando Larry Nagel y Donald Pederson desarrollan en el laboratorio de electrónica la primera versión de SPICE, 'Simulation Program with Integrated Circuits Emphasis', un simulador pensado para trabajar en estaciones de trabajo y grandes ordenadores, que podía analizar circuitos analógicos sin tener la necesidad de construirlos físicamente.

SPICE trabajaba con un fichero de texto, en el cual se alojaban una serie de declaraciones y comandos que eran leídos por SPICE, y tras comprobar que no existían errores en la sintaxis y conexiones declaradas, permitía hacer la simulación. Además incluía la opción de dejar ciertos parámetros sin especificar, tomando valores que se asignaban por defecto.

Debido a su gran utilidad, alcanzo gran fama, originándose multitud de versiones y programas similares, teniendo de base el diseño de SPICE. Una de ellas, fue creada por MicroSim Corporation en el año 1984, especialmente diseñada y adaptada para ordenadores personales y no solo entornos de trabajo en laboratorios. Una de sus versiones más usadas es la MicroSim PSpice 8.0, que es con la que trabajaremos nosotros.

3.2. Entornos de trabajo de MicroSim PSpice

MicroSim Pspice dispone de distintos entornos de trabajo, cada uno de ellos dedicado a un campo específico del diseño de placas de circuito impresas. Vamos a nombrar las más importantes:

a) *PDesign*, también conocido como DesignLab o Design Manager. Se trata de un gestor de proyectos, facilitando la gestión de los ficheros utilizados y agrupando todos los ficheros generados por las distintas herramientas a un mismo proyecto al que están asociados. En la parte izquierda de la ventana, tendremos el acceso rápido a todas los demás entornos de trabajo .

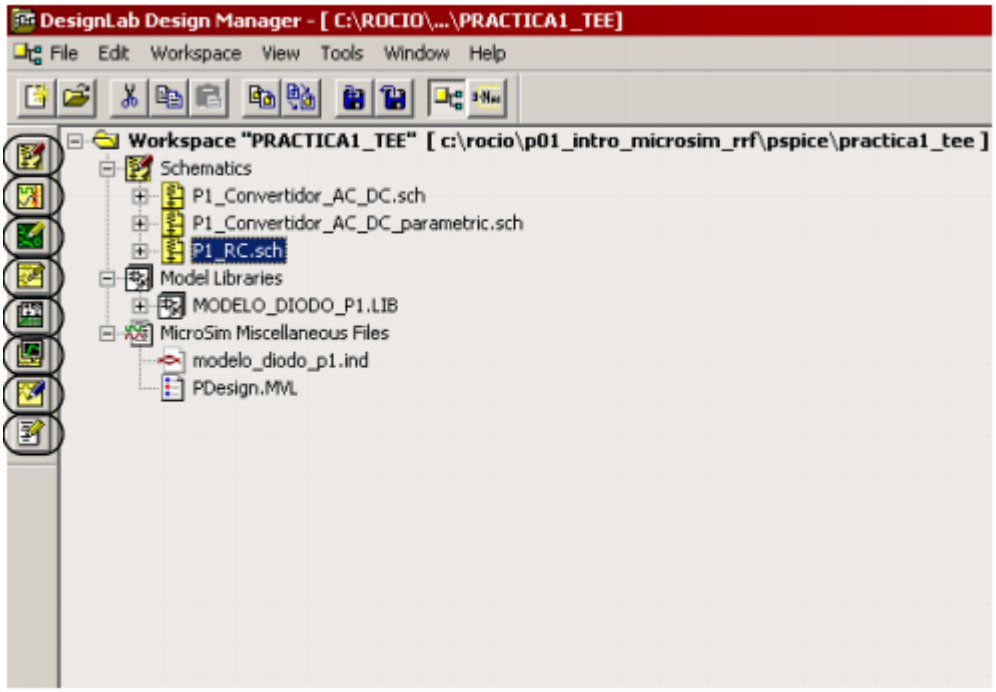

Ilustración 5: Ventana de trabajo de PDesign

b) *Schematics*: Esta aplicación gira entorno a la captura de esquemáticos. Con ella se pueden crear, editar y modificar circuitos eléctricos, definiendo los componentes y sus conexiones. Cada uno de los componentes, representados por símbolos gráficos, tienen asociado una serie de atributos, que rigen su comportamiento y características. Es fundamental asegurarse de que están correctamente definidos, para poder luego realizar las simulaciones y test y obtener resultados válidos.

Desde *Schematics* se pueden ejecutar distintos análisis del circuito, como podrían ser barridos en AC, barridos en DC, Análisis de MonteCarlo y análisis transitorios, entre otros.

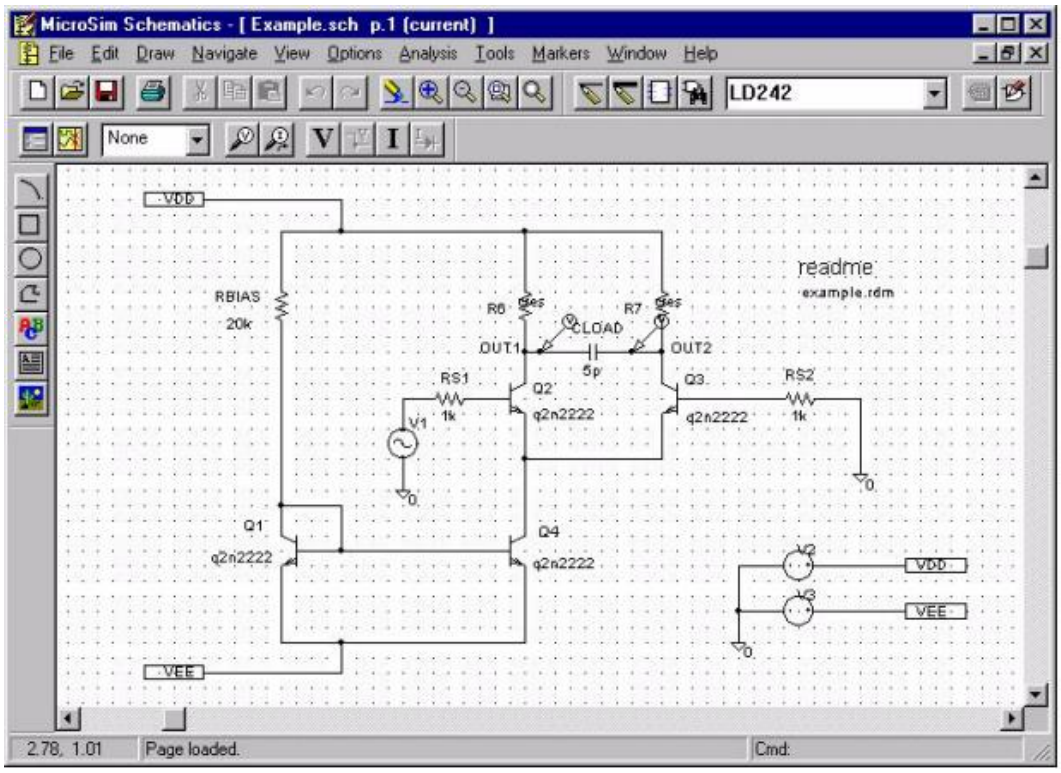

Ilustración 6: Aplicación Schematics

c) *PSpice A/D*: Es la herramienta de simulación de MicroSim Pspice. Con ella podemos simular circuitos tanto analógicos como digitales. Lee los circuitos generados por *Schematics* junto con las opciones de simulación escogidas por el usuario, genera una serie de datos como resultado.

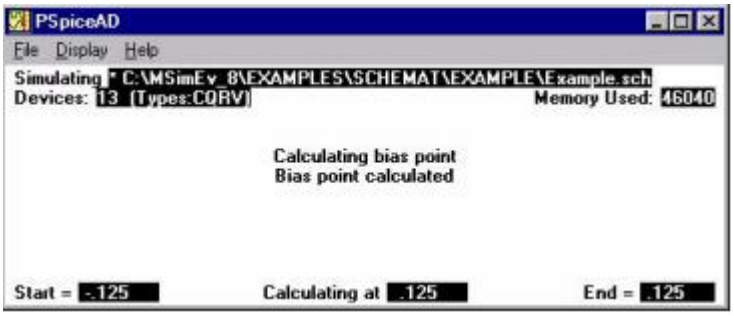

Ilustración 7: Ventana de simulación

d) *PCBoards*: En este entorno haremos la transición del esquema electrónico realizado anteriormente a un plano del modelo físico final. Definiremos el tamaño y forma de la placa, así como la disposición de todos sus componentes y como estarán interconectados entre ellos mediante pistas. Dispone de una herramienta DRC (Design Rule Check) que comprueba que se cumplen unas reglas básicas de diseño, como que estén todos los elementos bien conectados o evitar que existan pistas con muy poco espacio entre ellas. También posee un enrutador automático que genera un posible diseño de las pistas.

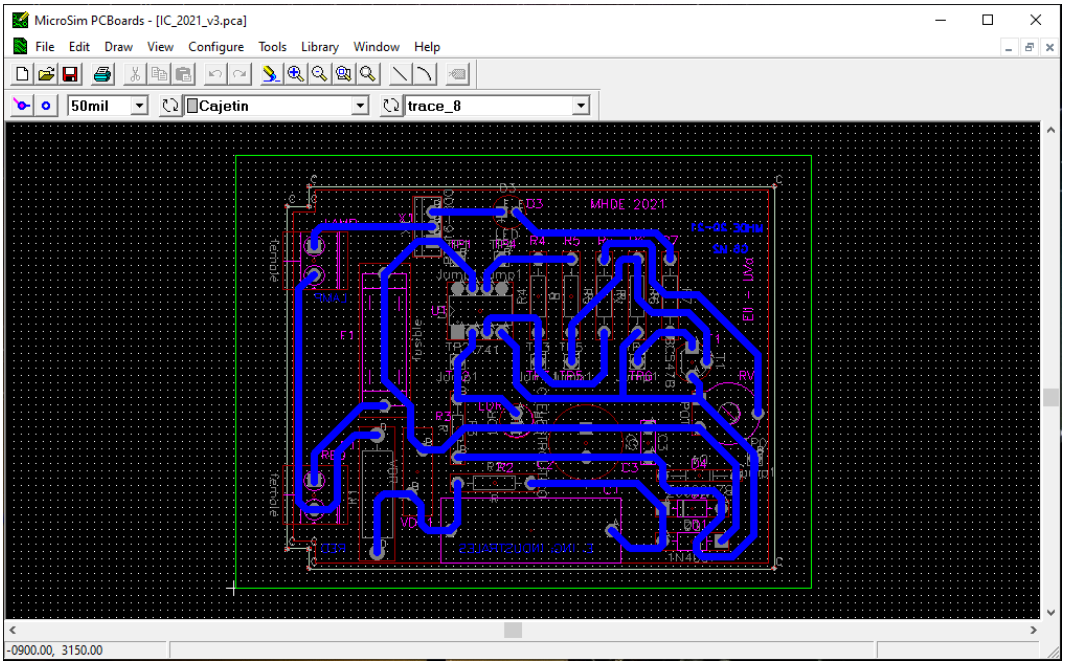

Ilustración 8: Ventana de trabajo de PCBoards

4. KiCad

4.1. Historia

KiCad es un programa de software libre cuyo propósito es el diseño de circuitos impresos y su posterior conversión a placa de circuito impreso.

Fue desarrollado en 1992 por Jean-Pierre Charras, cuando trabajaba en la IUT ("Instituts Universitaires de Technologie") de Grenoble, Francia. Desde entonces ha ido evolucionando gracias a colaboradores voluntarios y donaciones externas, entre las que cabe destacar el CERN, la Organización Europea para la Investigación Nuclear.

4.2. Entornos de trabajo de KiCad

Este programa usa un entorno integrado para las distintas etapas del proceso de diseño: Capturas esquemáticas, diseños de circuitos impresos y visualización de componentes. Se pueden añadir librerías con componentes, incluyendo su información importante como características, propiedades, huella, medidas del componente físico,... Estas aplicaciones cumplen las mismas funcionalidades que sus homólogas de MicroSim PSpice.

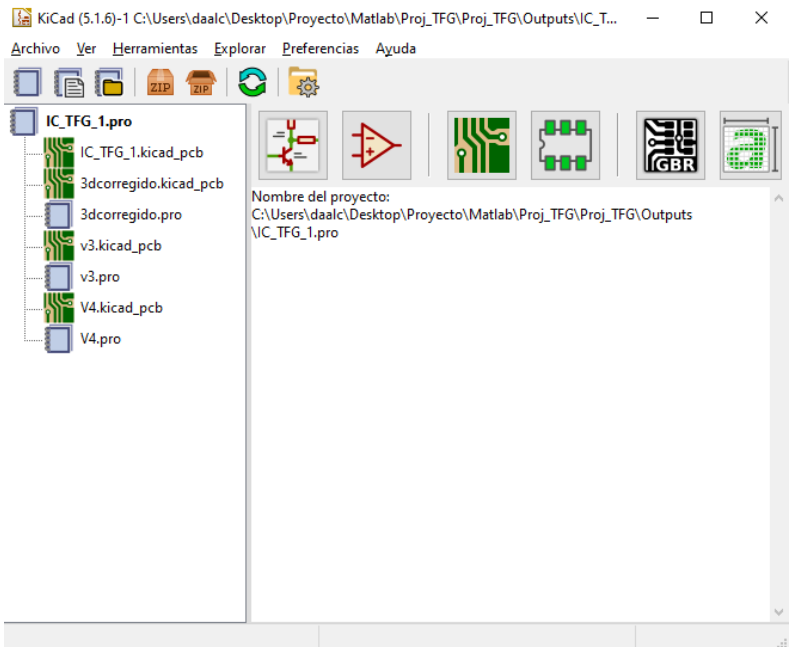

Ilustración 9: KiCad project manager

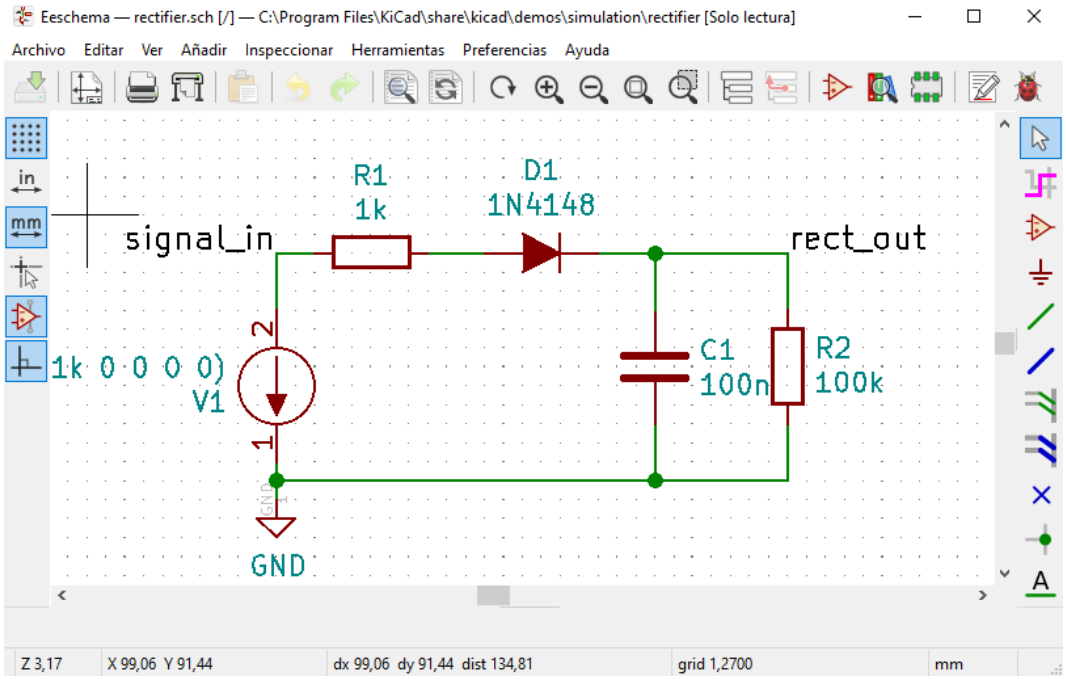

Ilustración 10: Editor de esquemáticos Eeschema

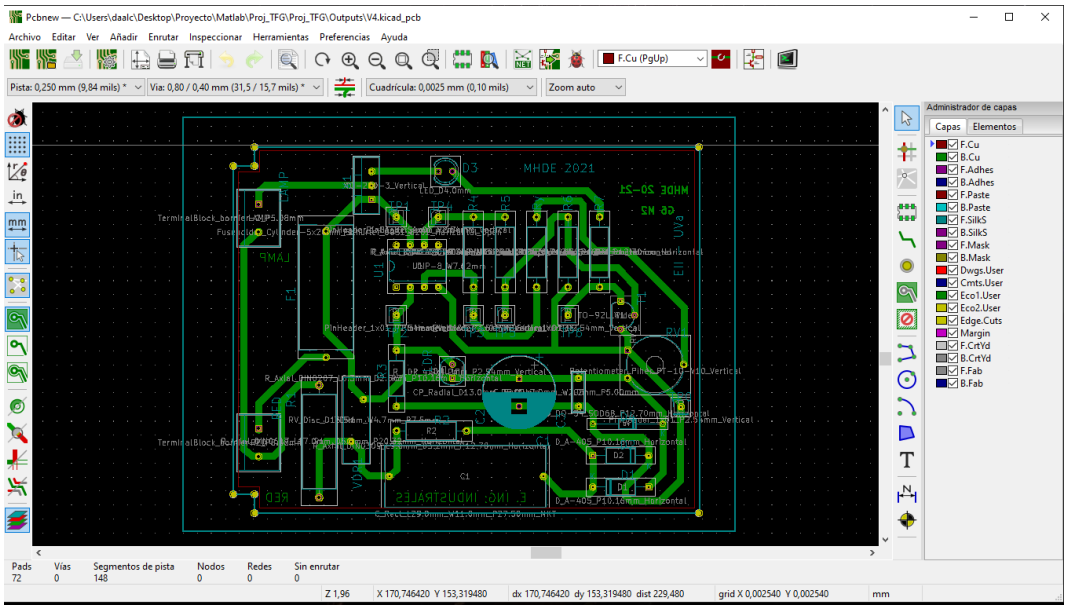

Ilustración 11: Editor de PCB Pcbnew

5. Elementos de una PCB

Nuestro objetivo, como habíamos mencionado anteriormente, es crear una aplicación que permita transferir toda la información necesaria de un archivo ".pca" generado por PCBoards de MicroSim 8.0, a un archivo "kicad_pcb" que pueda ser leído por KiCad y que nos permita obtener un modelo tridimensional de la placa.

Para ello, es fundamental saber cómo se representan los datos en ambos formatos, y cómo se transforman de uno a otro. El proceso de aprendizaje consistió en añadir los distintos elementos por separado e ir modificando el valor de sus variables, para así poder identificarlas en el fichero correspondiente.

Vamos a ir analizando individualmente cada uno de los elementos que conforman la placa de circuito impreso y estudiando sus características.

5.1. Sistemas de coordenadas

Lo primero es aclarar el sistema de coordenadas utilizado por cada programa para definir las posiciones de cualquier elemento. Todo sistema de coordenadas viene definido por un origen y por unos ejes de coordenadas. Otro dato importante que hay que tener presente, es que MicroSim trabaja en pulgadas y KiCad en mm.

5.1.1. MicroSim

En MicroSim, el origen está situado en la esquina inferior izquierda de la siguiente manera:

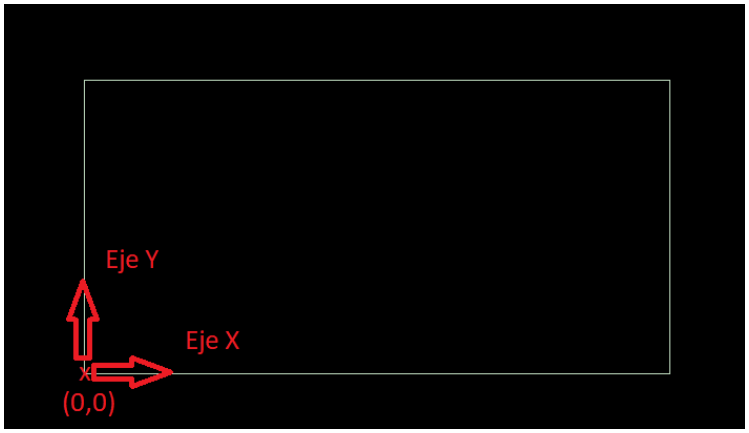

Ilustración 12: Ejes de coordenadas en MicroSim

5.1.2. KiCad

En KiCad, nuestros diseños aparecen dentro de una plantilla con cajetilla como se observa en la siguiente imagen:

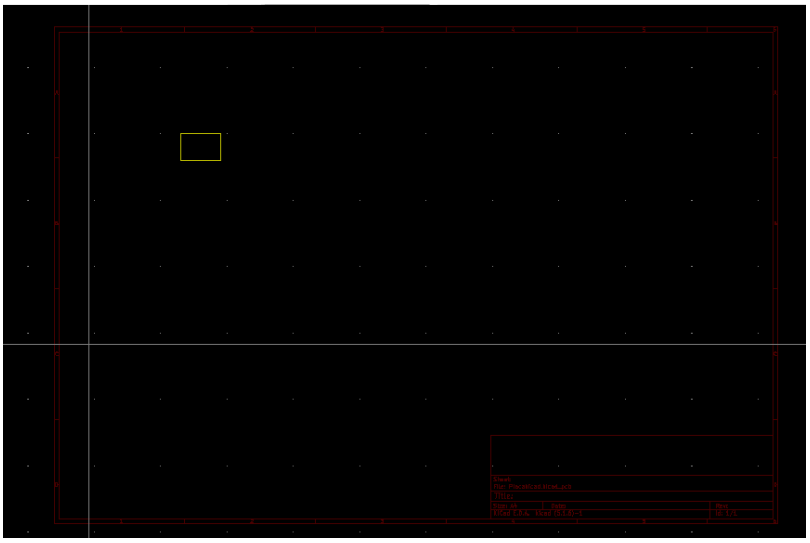

Ilustración 13: Plantilla de diseño de Pcbnew

El sistema de coordenadas se define con su origen fuera de la plantilla en la esquina superior izquierda y el eje Y tiene un sentido opuesto al utilizado por MicroSim.

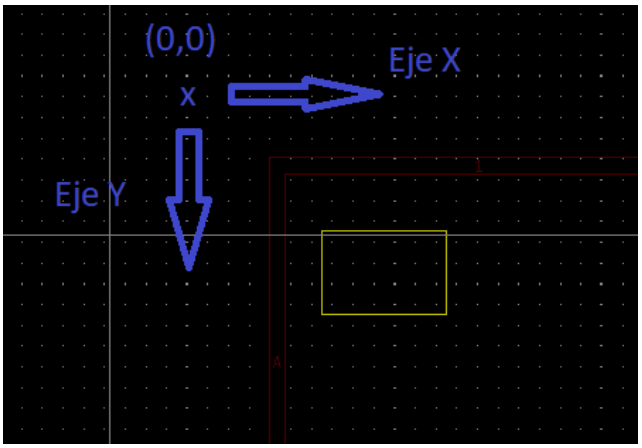

Ilustración 14: Ejes de coordenadas en KiCad

5.1.3. Transformación

Nuestro objetivo es representar la placa centrada en la plantilla de KiCad, para ello aplicaremos el siguiente cálculo a todas las coordenadas:

$$x_{KiCad} = 25.4 * x_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho\ placa$$

$$y_{KiCad} = 190 - 25.4 * y_{MicroSim} + 25.4 * \frac{1}{2} * alto\ placa$$

5.2. Borde de la placa

Ahora vamos a estudiar los límites físicos que marcan el margen de la placa de circuito impreso. Ese límite viene definidos por un conjunto de líneas dentro de una capa específica del editor de PCBs.

5.2.1. MicroSim

En MicroSim, la capa utilizada para definir los bordes se llama *"BoardOutline"*.

Su estructura es la siguiente:

*line BoardOutline *posXo posYo posXf posYf grosor* 0

```
*line BoardOutline 0.51969 2.74035 3.69469 2.74035 0.01 0
*line BoardOutline 3.69469 2.74035 3.69469 0.12785 0.01 0
*line BoardOutline 3.69469 0.12785 0.51969 0.12785 0.01 0
*line BoardOutline 0.51969 0.12785 0.51969 0.26535 0.01 0
*line BoardOutline 0.51969 0.26535 0.36969 0.26535 0.01 0
*line BoardOutline 0.36969 0.26535 0.36969 2.60285 0.01 0
*line BoardOutline 0.36969 2.60285 0.51969 2.60285 0.01 0
*line BoardOutline 0.51969 2.60285 0.51969 2.74035 0.01 0
```

Ilustración 15: Ejemplo de borde de placa en MicroSim

Al ser una línea, las variables que la definen son los componentes en el eje X y eje Y de su posición inicial y de su posición final, y el grosor de dicha línea.

5.2.2. KiCad

La capa designada para el límite de la placa utilizada en Pcbnew, dentro de KiCad, es la denominada *"Edge.Cuts"* y tiene la siguiente:

(gr_line (start *posXo posYo*) (end *posXf posYf*) (layer Edge.Cuts) (width *grosor*))

```
(gr_line (start 210.9726 153.5739) (end 291.6176 153.5739) (layer Edge.Cuts) (width 0.254))
(gr_line (start 291.6176 153.5739) (end 291.6176 219.9314) (layer Edge.Cuts) (width 0.254))
(gr_line (start 291.6176 219.9314) (end 210.9726 219.9314) (layer Edge.Cuts) (width 0.254))
(gr_line (start 210.9726 219.9314) (end 210.9726 216.4389) (layer Edge.Cuts) (width 0.254))
(gr_line (start 210.9726 216.4389) (end 207.1626 216.4389) (layer Edge.Cuts) (width 0.254))
(gr_line (start 207.1626 216.4389) (end 207.1626 157.0664) (layer Edge.Cuts) (width 0.254))
(gr_line (start 207.1626 157.0664) (end 210.9726 157.0664) (layer Edge.Cuts) (width 0.254))
(gr_line (start 210.9726 157.0664) (end 210.9726 153.5739) (layer Edge.Cuts) (width 0.254))
```

Ilustración 16: Ejemplo de borde de placa en KiCad

5.2.3. Transformación

Conociendo las coordenadas de todas las líneas que conforman la silueta de la placa, podemos calcular el ancho y alto de la placa:

$$ancho = x_{max} - x_{min}$$

$$alto = y_{max} - y_{min}$$

Siendo el X_{max} y X_{min}, los valores máximos y mínimos de todas las coordenadas X de todas las líneas, y de igual manera para Y_{max} e Y_{min}.

Utilizando la transformación de coordenadas aprendidas anteriormente, y pasando el grosor de pulgadas a milímetros, obtenemos:

$$posXo_{KiCad} = 25.4 * posXo_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posXf_{KiCad} = 25.4 * posXf_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posYo_{KiCad} = 190 - 25.4 * posYo_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$posYf_{KiCad} = 190 - 25.4 * posYf_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$grosor_{KiCad} = 25.4 * grosor_{MicroSim}$$

5.3. Líneas

Aquí englobamos todas aquellas líneas que no pertenezcan al borde de la placa, por lo que todas sus propiedades y ecuaciones de transformación se mantienen.

5.3.1. MicroSim

Su estructura será:

*line *capa posXo posYo posXf posYf grosor* 0

```
*line SilkBot 1.1825 1.4525 1.1825 1.4 0.009 0
*line SilkBot 1.16 1.4525 1.16 1.43 0.009 0
*line SilkBot 1.1375 1.4525 1.1375 1.4 0.009 0
*line SilkBot 1.115 1.4525 1.115 1.4 0.009 0
*line SilkBot 1.07375 1.4525 1.07375 1.4 0.009 0
*line SilkBot 1.085 1.43 1.10375 1.43 0.009 0
*line SilkBot 1.05125 1.4 1.05125 1.4525 0.009 0
*line SilkBot 1.025 1.4525 1.05125 1.4525 0.009 0
```

Ilustración 17: Ejemplo de líneas en MicroSim

5.3.2. KiCad

De igual manera, tenemos en KiCad la estructura:

(gr_line (start *posXo posYo*) (end *posXf posYf*) (layer *capa*) (width *grosor*))

Como se puede ver en la Ilustración 18:

```
(gr_line (start 2.893014e+02 1.526112e+02) (end 2.893014e+02 2.189052e+02) (layer F.SilkS) (width 2.540000e-01))
(gr_line (start 2.893014e+02 2.189052e+02) (end 2.090374e+02 2.189052e+02) (layer F.SilkS) (width 2.540000e-01))
(gr_line (start 2.090374e+02 2.189052e+02) (end 2.090374e+02 2.150952e+02) (layer F.SilkS) (width 2.540000e-01))
(gr_line (start 2.090374e+02 2.150952e+02) (end 2.052274e+02 2.150952e+02) (layer F.SilkS) (width 2.540000e-01))
(gr_line (start 2.052274e+02 2.150952e+02) (end 2.052274e+02 1.564212e+02) (layer F.SilkS) (width 2.540000e-01))
(gr_line (start 2.052274e+02 1.564212e+02) (end 2.090374e+02 1.564212e+02) (layer F.SilkS) (width 2.540000e-01))
(gr_line (start 2.090374e+02 1.564212e+02) (end 2.090374e+02 1.526112e+02) (layer F.SilkS) (width 2.540000e-01))
(gr_line (start 2.279985e+02 1.862535e+02) (end 2.279985e+02 1.875870e+02) (layer F.SilkS) (width 2.286000e-01))
(gr_line (start 2.274270e+02 1.862535e+02) (end 2.274270e+02 1.868250e+02) (layer F.SilkS) (width 2.286000e-01))
(gr_line (start 2.268555e+02 1.862535e+02) (end 2.268555e+02 1.875870e+02) (layer F.SilkS) (width 2.286000e-01))
(gr_line (start 2.262840e+02 1.862535e+02) (end 2.262840e+02 1.875870e+02) (layer F.SilkS) (width 2.286000e-01))
```

Ilustración 18: Ejemplo de líneas en KiCad

5.3.3. Transformación

Aquí aplicaremos las mismas transformaciones que hemos visto anteriormente. La única diferencia es saber el nombre de la capa correspondiente de KiCad donde vamos a trasladarla, según la Tabla 1.

$$posXo_{KiCad} = 25.4 * posXo_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posXf_{KiCad} = 25.4 * posXf_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posYo_{KiCad} = 190 - 25.4 * posYo_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$posYf_{KiCad} = 190 - 25.4 * posYf_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$grosor_{KiCad} = 25.4 * grosor_{MicroSim}$$

Capa de MicroSim	Capa equivalente de KiCad
SilkTop	F.SilkS
SilkBot	B.SilkS
Solder	B.Cu
Otras	Dwgs.User

Tabla 1: Equivalencia de tablas

5.4. Arcos

Los arcos son segmentos de una circunferencia, definidos a partir de sendos puntos sobre dicha circunferencia.

5.4.1. MicroSim

En MircroSim se definen a partir del origen de la circunferencia (*origenX*, *origenY*), el radio de dicha circunferencia (*R*), y sus ángulos inicial y final: alfa, omega (α, Ω).

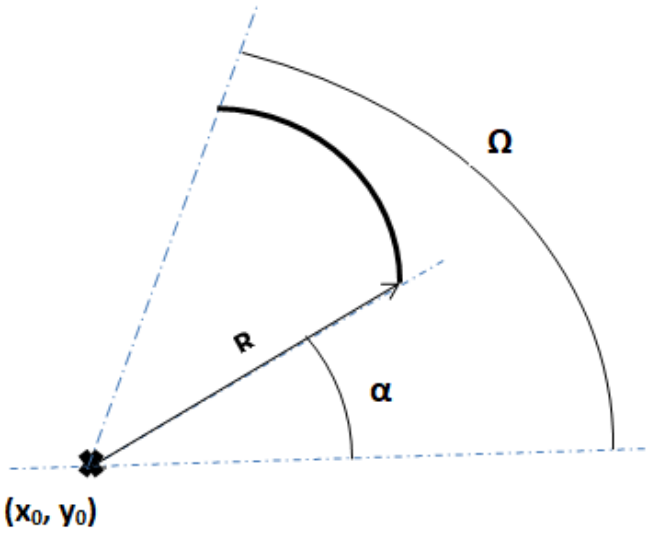

Ilustración 19: Definición de un arco en MicroSim

Su manera de representarlo es la siguiente:

*arc *capa origenX origenY R α Ω grosor* 0

OrigenX e origenY se corresponden en la Ilustración 19 con x_o e y_o respectivamente.

```
*arc SilkTop 0.55 1.55 0.206155 345.964 90 0.01 0
```

Ilustración 20: Ejemplo de arco en MicroSim

5.4.2. KiCad

En KiCad los arcos se definen de una manera distinta, con el origen de la circunferencia (*origenX*, *origenY*), radio (*R*), punto inicial de la circunferencia (*posX*, *posY*) y ángulo del arco (β):

(gr_arc (start *origenX origenY*) (end *posX posY*) (angle β) (layer *capa*) (width *grosor*))

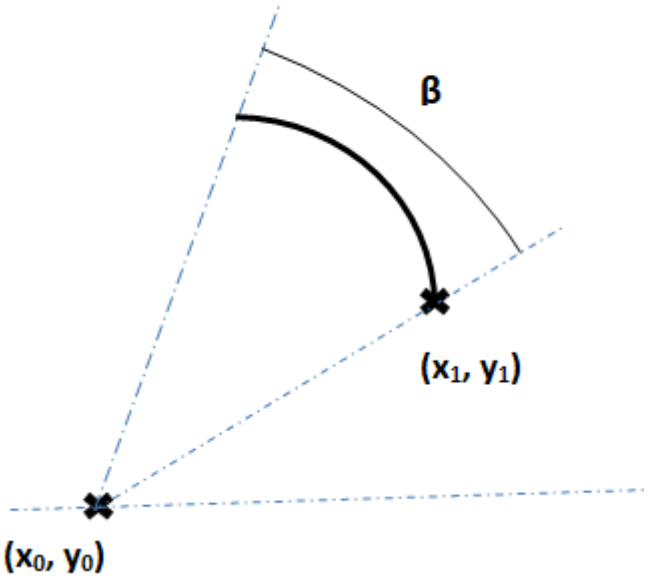

Ilustración 21: Definición de un arco en KiCad

OrigenX e origenY se corresponden en la Ilustración 21 con x_0, y_0, y posX e posY con x_1, y_1 respectivamente.

```
(gr_arc (start 2.117425e+02 1.838087e+02) (end 3.348183e+02 1.956246e+02) (angle 3.457578e+02) (layer F.SilkS) (width 2.540000e-01))
```

Ilustración 22: Ejemplo de arco en KiCad

5.4.3. Transformación

Para realizar la transformación, además del cambio de unidades y sistemas de referencia, deberemos aplicar unos simples cálculos trigonométricos.

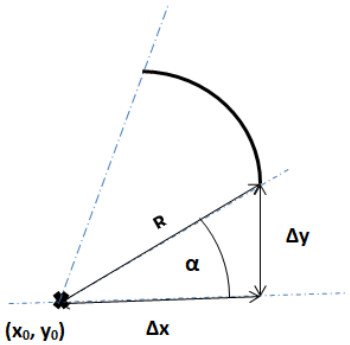

Ilustración 23: Representación trigonométrica para la transformación

$$\Delta x = R * \cos \alpha$$
$$\Delta y = R * \text{sen } \alpha$$

Operando y sustituyendo tenemos entonces:

$$posX = origenX + radio * \cos \alpha$$
$$posY = origenY + radio * \text{sen } \alpha$$
$$\beta = \ \Omega - \ \alpha$$
$$posX_{KiCad} = 25.4 * posX_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$
$$posY_{KiCad} = 190 - 25.4 * posY_{MicroSim} + 25.4 * \frac{1}{2} * alto$$
$$grosor_{KiCad} = 25.4 * grosor_{MicroSim}$$

5.5. Textos

En este apartado englobamos todos los textos añadidos por el usuario. No se incluyen las referencias de los componentes o su modelo, si vienen fijadas al propio componente.

5.5.1. MicroSim

La estructura del texto en MicroSim es la siguiente:

*text 0.006
@attribute TEXT= *texto*
display [*capa*] *capa posX posY alineación ángulo tamaño_letra grosor* 0 0 0

```
*text 0.006
@attribute TEXT=MHDE
display  [SilkTop] SilkTop 0.9 2.65 4 0 0.06 0.012 0 0 0
```

Ilustración 24: Ejemplo de texto en MicroSim

5.5.2. KiCad

El texto toma la siguiente forma:

(gr_text "*texto*" (at *posX posY angulo*) (layer *capa*)
(effects (font (size *tamaño_letra tamaño_letra)* (thickness *grosor*))))

```
(gr_text "MHDE" (at 2.208230e+02 1.558370e+02 0) (layer F.SilkS)
(effects (font (size 1.524000e+00 1.524000e+00) (thickness 3.048000e-01))))
```

Ilustración 25: Ejemplo de texto en KiCad

5.5.3. Transformación

Para la correcta transformación usaremos los cambios de coordenadas y el cambio de unidades:

$$posX_{KiCad} = 25.4 * posX_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posY_{KiCad} = 190 - 25.4 * posY_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$\acute{a}ngulo_{KiCad} = \acute{a}ngulo_{MicroSim}$$

$$tamaño\_letra_{KiCad} = 25.4 * tamaño\_letra_{MicroSim}$$

$$grosor_{KiCad} = 25.4 * grosor_{MicroSim}$$

Para las capas, véase la Tabla 1.

5.6. Pasantes

Los pasantes, o taladros, son agujeros mecanizados que atraviesan la placa de un extremo a otro. Se utilizan principalmente para anclar la placa a otras superficies.

5.6.1. MicroSim

Las pasantes se definen como:

*hole Component Solder *posX posY radio* 0.005

Las capas Component y Solder son las capas origen y final, como siempre van a ser pasantes, es decir, atraviesan todas las capas, siempre van a ser las capas más externas.

```
*hole Component Solder 0.436 0.3195 0.03 0.005
```

Ilustración 26: Ejemplo de pasante en MicroSim

5.6.2. KiCad

En KiCad no se dispone de un símbolo del circuito electrónico para representarlo. La solución es incluir un componente llamado "Mounting Hole" que hemos adaptado a nuestras necesidades.

```
(module MountingHole_2.1mm (layer F.Cu) (tedit 5B924765)
  (at posX posY 0)
  (descr "Mounting Hole 2.1mm, no annular")
  (tags "mounting hole 2.1mm no annular")
  (attr virtual)
  (fp_circle (center 0 0) (end 0.762 0) (layer Cmts.User) (width 0.15))
  (fp_circle (center 0 0) (end 0.762 0) (layer F.CrtYd) (width 0.05))
  (pad "" np_thru_hole circle (at 0 0) (size radio radio) (drill radio) (layers *.Cu *.Mask))
)
```

Ilustración 27: Ejemplo de pasante en KiCad

5.6.3. Transformación

Aplicando la misma transformación de coordenadas y unidades:

$$posX_{KiCad} = 25.4 * posX_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posY_{KiCad} = 190 - 25.4 * posY_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$radio_{KiCad} = 25.4 * radio_{MicroSim}$$

5.7. Pistas

Una pista es un camino de material conductor (cobre) sobre una base no conductora, que conecta físicamente los pines de los componentes en una placa, de manera que exista continuidad eléctrica entre ellos.

5.7.1. MicroSim

La forma de definir las pistas en MicroSim:

@seg Solder *posXo posYo posXf posYf grosor* 0.025 -

```
@seg Solder 2.476 1.817 2.476 1.617 0.05 0.025 -
@seg Solder 2.386 2.107 2.386 1.957 0.05 0.025 -
@seg Solder 2.386 1.957 2.476 1.867 0.05 0.025 -
@seg Solder 2.476 1.867 2.476 1.817 0.05 0.025 -
```

Ilustración 28: Ejemplo de pistas en MicroSim

5.7.2. KiCad

Las pistas en el fichero .kicad_pcb se describen de la siguiente forma:

(segment (start *posXo posYo*) (end *posXf posYf*) (width *grosor*) (layer B.Cu))

```
(segment (start 2.814274e+02 1.696292e+02) (end 2.826974e+02 1.683592e+02) (width 1.270000e+00) (layer B.Cu) )
(segment (start 2.826974e+02 1.683592e+02) (end 2.826974e+02 1.607392e+02) (width 1.270000e+00) (layer B.Cu) )
(segment (start 2.750774e+02 1.696292e+02) (end 2.814274e+02 1.696292e+02) (width 1.270000e+00) (layer B.Cu) )
(segment (start 2.735534e+02 1.769952e+02) (end 2.712674e+02 1.792812e+02) (width 1.270000e+00) (layer B.Cu) )
```

Ilustración 29: Ejemplo de pistas en KiCad

5.7.3. Transformación

Aplicando la misma transformación de coordenadas y unidades:

$$posXo_{KiCad} = 25.4 * posXo_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posXf_{KiCad} = 25.4 * posXf_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posYo_{KiCad} = 190 - 25.4 * posYo_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$posYf_{KiCad} = 190 - 25.4 * posYf_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$grosor_{KiCad} = 25.4 * grosor_{MicroSim}$$

5.8. Vías

Son agujeros, o puentes, cuyo diámetro interior está recubierto de material conductor. Se utilizan para conectar eléctricamente pistas de distintas capas de la placa de circuito impreso.

5.8.1. MicroSim

Su estructura es:

@via Component Solder *posX posY tipo_pad angulo*

De la variable *tipo_pad* hablaremos más adelante.

```
@via Component Solder 1.225 1.975 rnd-060-040 0
```

Ilustración 30: Ejemplo de vía en MicroSim

5.8.2. KiCad

En KiCad se define de la siguiente manera:

(via (at *posX posY*) (size *radio_ext*) (drill *radio_int*) (layers F.Cu B.Cu) (net 0))

Ambos radios vienen determinados por el tipo de pad. Por ejemplo:

```
(via (at 2.209500e+02 1.773000e+02) (size 6.000000e-01) (drill 4.000000e-01) (layers F.Cu B.Cu) (net 0))
(via (at 2.209500e+02 1.715850e+02) (size 6.000000e-01) (drill 4.000000e-01) (layers F.Cu B.Cu) (net 0))
```

Ilustración 31: Ejemplo de vía en KiCad

5.8.3. Transformación

Aplicando la misma transformación de coordenadas y unidades:

$$posX_{KiCad} = 25.4 * posX_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posY_{KiCad} = 190 - 25.4 * posY_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

5.9. Componentes

Son los dispositivos que forman los circuitos electrónicos, conectados entre sí mediante pistas. Cada uno realiza una función concreta, según el tipo de componente.

5.9.1. MicroSim

En MicroSim, los componentes vienen definidos en dos partes. La primera es el “footprint” o huella. Esta huella corresponde con la representación gráfica del componente en los circuitos y planos electrónicos. Por norma general, suele coincidir con la planta del dispositivo, es decir, con la proyección ortogonal de la cara superior.

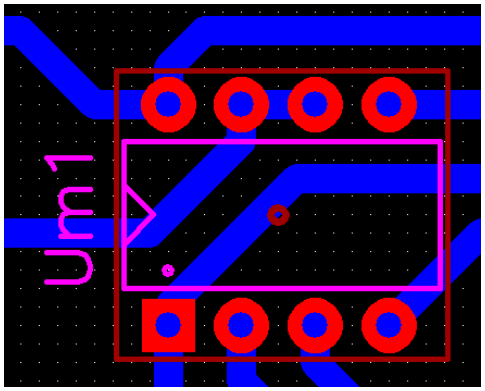

Ilustración 32: Vista de un footprint en Pcbnew

Las huellas se definen como un conjunto de líneas que permiten definir esa representación gráfica.

```
*footprint DIP8 8 THRU
@refdes 4
display  [AssyDrwTop] AssemblyTop -0.08 0.15 4 90 0.06 0.008 0 0 0
display  [SilkTop] SilkTop -0.08 0.15 4 90 0.06 0.008 0 0 0
@comptype 4
display  [AssyDrwTop] AssemblyTop 0.15 -0.15 4 0 0.06 0.008 0 0 0
@graphics
line [SilkTop] -0.06 0.05 -0.06 0.25 0.008 0
line [SilkTop] -0.06 0.25 0.37 0.25 0.008 0
line [SilkTop] 0.37 0.25 0.37 0.05 0.008 0
line [SilkTop] 0.37 0.05 -0.06 0.05 0.008 0
line [SilkTop] -0.06 0.19 -0.02 0.15 0.008 0
line [SilkTop] -0.02 0.15 -0.06 0.11 0.008 0
line [AssyDrwTop] -0.06 0.05 -0.06 0.25 0.008 0
line [AssyDrwTop] -0.06 0.25 0.37 0.25 0.008 0
line [AssyDrwTop] 0.37 0.25 0.37 0.05 0.008 0
line [AssyDrwTop] 0.37 0.05 -0.06 0.05 0.008 0
line [AssyDrwTop] -0.06 0.19 -0.02 0.15 0.008 0
line [AssyDrwTop] -0.02 0.15 -0.06 0.11 0.008 0
line [BoundaryTop] -0.07 -0.045 -0.07 0.345 0.008 0
line [BoundaryTop] -0.07 0.345 0.38 0.345 0.008 0
line [BoundaryTop] 0.38 0.345 0.38 -0.045 0.008 0
line [BoundaryTop] 0.38 -0.045 -0.07 -0.045 0.008 0
centroid [BoundaryTop] 0.15 0.15
arc [SilkTop] 0 0.075 0.005 0 0 0.008
arc [AssyDrwTop] 0 0.075 0.005 0 0 0.008
@pins
pin 5 0.3 0.3 0 rnd-075-035 thru
pin 6 0.2 0.3 0 rnd-075-035 thru
pin 7 0.1 0.3 0 rnd-075-035 thru
pin 8 0 0.3 0 rnd-075-035 thru
pin 2 0.1 0 0 rnd-075-035 thru
pin 3 0.2 0 0 rnd-075-035 thru
pin 4 0.3 0 0 rnd-075-035 thru
pin 1 0 0 0 sq-070-035 thru
```

Ilustración 33: Ejemplo de footprint en KiCad

Esta huella va asociada a todos los componentes DIP8 en este caso.
Ahora necesitamos saber las coordenadas y orientación de cada componente dentro de la placa.

```
*component 555B DIP8 Um1 1.725 2.825 0 Component NONE
@attribute REFDES=Um1
display  [AssyDrwTop] AssemblyTop -0.08 0.15 4 90 0.06 0.008 0 0 0
display  [SilkTop] SilkTop -0.105 0.15 4 90 0.06 0.008 0 0 0
@attribute TYPE_NAME=555B
display  [AssyDrwTop] AssemblyTop 0.15 -0.15 4 0 0.06 0.008 0 0 0
```

Ilustración 34: Ejemplo de datos de componente de MicroSim

Los datos que buscamos se encuentran en la primera línea.

*component *nombre_componente nombre_footprint nombre_referencia posX posY angulo* Component NONE

Nombre_componente corresponde al nombre del modelo del componente físico. En este ejemplo es un 555B dentro de los integrados de 8 patillas.

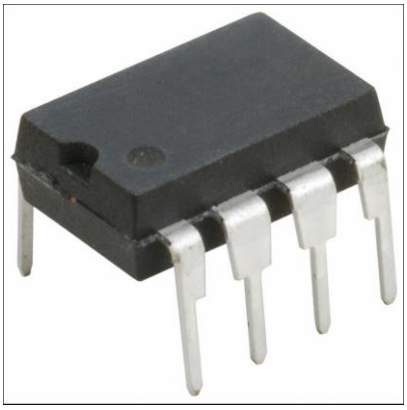

Ilustración 35: Encapsulado de 8 patillas

Nombre_footprint es el nombre de la huella utilizada para representar gráficamente a este componente. En este caso utilizará el footprint denominado DIP8.

Nombre_referencia es un nombre que se asigna a cada componente individual para poder distinguirlos y diferenciarlos en el caso de que existan múltiples unidades de ese modelo. En este ejemplo se le ha llamado Um1.

posX, posY y *ángulo* sirven para definir la posición y orientación del componente. Un dato muy importante a tener en cuenta, es que (*posX* , *posY)* marcan la posición del pin 1 (o pad) en la PCB. Todas las líneas que definen el footprint usan este mismo pad como origen de coordenadas.

5.9.2. KiCad

En KiCad de igual manera tenemos el footprint definido como un conjunto de líneas, y también tienen como origen de coordenadas la posición de su pad 1 correspondiente. En la segunda línea podemos ver la información de posición y orientación del componente. Esta posición corresponde de nuevo a la posición donde encontraremos el pad 1.

(at *posX posY angulo*)

```
)(module DIP-8_W7.62mm (layer F.Cu) (tedit 5A02E8C5)
  (at 233.65 155.71 90)
  (descr "8-lead though-hole mounted DIP package, row spacing 7.62 mm (300 mils)")
  (tags "THT DIP DIL PDIP 2.54mm 7.62mm 300mil")
  (fp_text reference Um1 (at 3.81 -2.667 0) (layer F.SilkS)
    (effects (font (size 1.524 1.524) (thickness 0.2032)))
  )
  (fp_text value DIP-8_W7.62mm (at 3.81 9.95) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (fp_arc (start 3.81 -1.33) (end 2.81 -1.33) (angle -180.000000) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.635 -1.27) (end 6.985 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 -1.27) (end 6.985 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 8.89) (end 0.635 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 8.89) (end 0.635 -0.27) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 -0.27) (end 1.635 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 2.81 -1.33) (end 1.16 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 -1.33) (end 1.16 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 8.95) (end 6.46 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 8.95) (end 6.46 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 -1.33) (end 4.81 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start -1.1 -1.55) (end -1.1 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start -1.1 9.15) (end 8.7 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 9.15) (end 8.7 -1.55) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 -1.55) (end -1.1 -1.55) (layer F.CrtYd) (width 0.05))
  (pad 5 thru_hole circle (at 7.62 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 6 thru_hole circle (at 7.62 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 7 thru_hole circle (at 7.62 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 8 thru_hole circle (at 7.62 0) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 2 thru_hole circle (at 0 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 3 thru_hole circle (at 0 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 4 thru_hole circle (at 0 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 1 thru_hole rect (at 0 0) (size 0.70 0.70) (drill 0.35) (layers *.Cu *.Mask))

  (fp_text user %R (at 3.81 3.81) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (model ${KISYS3DMOD}/Package_DIP.3dshapes/DIP-8_W7.62mm.wrl
    (at (xyz 0 0 0))
    (scale (xyz 1 1 1))
    (rotate (xyz 0 0 0))
  )
```

Ilustración 36: Ejemplo de footprint en MicroSim

Recordemos que nuestro objetivo final es obtener una imagen tridimensional de la placa final, y los footprint solo nos proporcionan la representación plana. Para solucionarlo, tendremos que incluir para cada componente la ruta donde está guardado el modelo en 3D de dicho componente, junto a unas variables para ajustar en caso de que sea necesario, tales como posición, escala de tamaño o rotación.

```
)(module DIP-8_W7.62mm (layer F.Cu) (tedit 5A02E8C5)
  (at 233.65 155.71 90)
  (descr "8-lead though-hole mounted DIP package, row spacing 7.62 mm (300 mils)")
  (tags "THT DIP DIL PDIP 2.54mm 7.62mm 300mil")
  (fp_text reference Um1 (at 3.81 -2.667 0) (layer F.SilkS)
    (effects (font (size 1.524 1.524) (thickness 0.2032)))
  )
  (fp_text value DIP-8_W7.62mm (at 3.81 9.95) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (fp_arc (start 3.81 -1.33) (end 2.81 -1.33) (angle -180.000000) (layer F.SilkS) (width 0.12
  (fp_line (start 1.635 -1.27) (end 6.985 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 -1.27) (end 6.985 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 8.89) (end 0.635 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 8.89) (end 0.635 -0.27) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 -0.27) (end 1.635 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 2.81 -1.33) (end 1.16 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 -1.33) (end 1.16 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 8.95) (end 6.46 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 8.95) (end 6.46 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 -1.33) (end 4.81 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start -1.1 -1.55) (end -1.1 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start -1.1 9.15) (end 8.7 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 9.15) (end 8.7 -1.55) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 -1.55) (end -1.1 -1.55) (layer F.CrtYd) (width 0.05))
  (pad 5 thru_hole circle (at 7.62 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 6 thru_hole circle (at 7.62 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 7 thru_hole circle (at 7.62 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 8 thru_hole circle (at 7.62 0) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 2 thru_hole circle (at 0 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 3 thru_hole circle (at 0 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 4 thru_hole circle (at 0 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 1 thru_hole rect (at 0 0) (size 0.70 0.70) (drill 0.35) (layers *.Cu *.Mask))

  (fp_text user %R (at 3.81 3.81) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (model ${KISYS3DMOD}/Package_DIP.3dshapes/DIP-8_W7.62mm.wrl
    (at (xyz 0 0 0))
    (scale (xyz 1 1 1))
    (rotate (xyz 0 0 0))
  )
```

Ilustración 37: Modelo 3D en KiCad

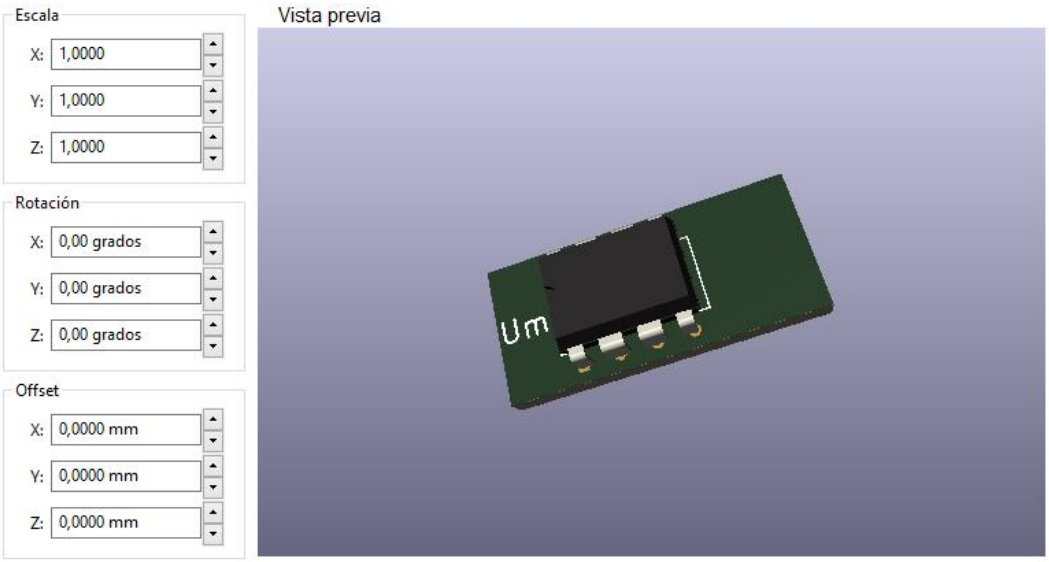

Ilustración 38: Ejemplo tridimensional en KiCad

5.9.3. Transformación

Para las transformaciones nos vamos a ayudar de las librerías de KiCad donde vienen ya totalmente definidos los modelos. En caso de que el componente específico que estemos buscando no se halle en dichas librerías habrá que buscarlo en internet. Entraremos más en detalle más adelante. Para adaptar estos modelos, deberemos adaptarlos modificando la posición y ángulo final. Aplicando la misma transformación de coordenadas y unidades:

$$posX_{KiCad} = 25.4 * posX_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posY_{KiCad} = 190 - 25.4 * posY_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$\acute{a}ngulo_{KiCad} = \acute{a}ngulo_{MicroSim}$$

5.10. Referencias de Componentes

Denominados referencia a la denominación que damos a cada componente individual para poder distinguirlo de los demás. Estas referencias pueden ser asignadas por el usuario componente a componente, aunque los programas de diseño electrónico pueden hacer lo manera automática siguiendo un conjunto de reglas internas. Por ejemplo numerará todas las resistencias de la placa como R1, R2, R3,... siguiendo un orden de arriba abajo y de izquierda a derecha según su posición en la placa.

Para mantener la coherencia, vamos a trasladar toda la información exacta de esas referencias: Su nombre, posición y orientación, tamaño de letra y grosor del texto.

5.10.1. MicroSim

Anteriormente en la Ilustración 38, habíamos visto donde se encontraba la referencia. Para conocer sus propiedades deberemos fijarnos en el display de la capa SilkTop. Estas posiciones también son relativas al pad 1. Su estructura es:

display [SilkTop] SilkTop *posX posY alin ángulo tamaño_letra grosor* 0 0 0

```
*component 555B DIP8 Um1 1.725 2.825 0 Component NONE
@attribute REFDES=Um1
display  [AssyDrwTop] AssemblyTop -0.08 0.15 4 90 0.06 0.008 0 0 0
display  [SilkTop] SilkTop -0.105 0.15 4 90 0.06 0.008 0 0 0
@attribute TYPE_NAME=555B
display  [AssyDrwTop] AssemblyTop 0.15 -0.15 4 0 0.06 0.008 0 0 0
```

Ilustración 39: Localización de la referencia en MicroSim

5.10.2. KiCad

```
)(module DIP-8_W7.62mm (layer F.Cu) (tedit 5A02E8C5)
  (at 233.65 155.71 90)
  (descr "8-lead though-hole mounted DIP package, row spacing 7.62 mm (300 mils)")
  (tags "THT DIP DIL PDIP 2.54mm 7.62mm 300mil")
  (fp_text reference Um1 (at 3.81 -2.667 0) (layer F.SilkS)
    (effects (font (size 1.524 1.524) (thickness 0.2032)))
  )
  (fp_text value DIP-8_W7.62mm (at 3.81 9.95) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (fp_arc (start 3.81 -1.33) (end 2.81 -1.33) (angle -180.000000) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.635 -1.27) (end 6.985 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 -1.27) (end 6.985 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 8.89) (end 0.635 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 8.89) (end 0.635 -0.27) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 -0.27) (end 1.635 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 2.81 -1.33) (end 1.16 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 -1.33) (end 1.16 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 8.95) (end 6.46 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 8.95) (end 6.46 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 -1.33) (end 4.81 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start -1.1 -1.55) (end -1.1 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start -1.1 9.15) (end 8.7 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 9.15) (end 8.7 -1.55) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 -1.55) (end -1.1 -1.55) (layer F.CrtYd) (width 0.05))
  (pad 5 thru_hole circle (at 7.62 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 6 thru_hole circle (at 7.62 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 7 thru_hole circle (at 7.62 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 8 thru_hole circle (at 7.62 0) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 2 thru_hole circle (at 0 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 3 thru_hole circle (at 0 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 4 thru_hole circle (at 0 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 1 thru_hole rect (at 0 0) (size 0.70 0.70) (drill 0.35) (layers *.Cu *.Mask))

  (fp_text user %R (at 3.81 3.81) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (model ${KISYS3DMOD}/Package_DIP.3dshapes/DIP-8_W7.62mm.wrl
    (at (xyz 0 0 0))
    (scale (xyz 1 1 1))
    (rotate (xyz 0 0 0))
  )
```

Ilustración 40: Datos de referencia en KiCad

Como podemos observar en el texto remarcado, en KiCad tiene el siguiente formato:

(fp_text reference *nombre_referencia* (at *posX posY angulo*) (layer F.SilkS)
  (effects (font (size *tamaño_letra tamaño_letra*) (thickness *grosor*)))
)

5.10.3. Transformación

Aplicando la misma transformación de coordenadas y unidades:

$$posX_{KiCad} = 25.4 * posX_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posY_{KiCad} = 190 - 25.4 * posY_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$ángulo_{KiCad} = ángulo_{MicroSim}$$

$$grosor_{KiCad} = 25.4 * grosor_{MicroSim}$$

5.11. Pads

Los Pads, también llamados pines, isletas o nodos, se encargan de realizar la conexión entre las patillas de los componentes y las pistas de la placa. Los pads tienen tres características fundamentales:

- Tecnología: Pueden ser THT (Through Hole Technology) o SMD (Surface Mounting Device). Los THT cuentan con un tubo de material conductor, en el cual se sueldan las patillas del componente una vez insertadas, mientras que los SMD no cuentan con un pasante, si no que se sueldan directamente el componente con la placa.

- Forma: Según su forma geométrica, podríamos clasificarlos en: pads cuadrados, pads rectangulares, pads ovalados y pads circulares, siendo estos últimos los más utilizados.

- Tamaño: Dependiendo de su forma y tipo se necesitarán un número de medidas para poder definir el pad de forma inequívoca. Para definir un círculo o un cuadrado solo necesitamos una medida, mientras que para los rectángulos y óvalos necesitaremos dos. Además si el pad fuera de tipo THT, necesitamos una medida adicional para definir el tamaño del taladro (drill).

Los modelos incluidos en las bibliotecas de KiCad ya vienen con sus pads correspondientes incorporados. Sin embargo, hemos querido aumentar la flexibilidad y transmitir la información de todos los pads de MicroSim a KiCad. De esta forma, si en nuestro modelo MicroSim queremos cambiar el tamaño, forma o posición de cualquier pad para ajustarlo mejor a nuestras necesidades, ese cambio se verá reflejado en KiCad.

5.11.1. MicroSim

Dentro de la descripción de cada footprint, en el apartado @pins se describen todos los pines con la siguiente estructura:

pin *num_pin posX posY angulo medidas_pin tipo_pin*

```
*footprint DIP8 8 THRU
@refdes 4
display  [AssyDrwTop] AssemblyTop -0.08 0.15 4 90 0.06 0.008 0 0 0
display  [SilkTop] SilkTop -0.08 0.15 4 90 0.06 0.008 0 0 0
@comptype 4
display  [AssyDrwTop] AssemblyTop 0.15 -0.15 4 0 0.06 0.008 0 0 0
@graphics
line [SilkTop] -0.06 0.05 -0.06 0.25 0.008 0
line [SilkTop] -0.06 0.25 0.37 0.25 0.008 0
line [SilkTop] 0.37 0.25 0.37 0.05 0.008 0
line [SilkTop] 0.37 0.05 -0.06 0.05 0.008 0
line [SilkTop] -0.06 0.19 -0.02 0.15 0.008 0
line [SilkTop] -0.02 0.15 -0.06 0.11 0.008 0
line [AssyDrwTop] -0.06 0.05 -0.06 0.25 0.008 0
line [AssyDrwTop] -0.06 0.25 0.37 0.25 0.008 0
line [AssyDrwTop] 0.37 0.25 0.37 0.05 0.008 0
line [AssyDrwTop] 0.37 0.05 -0.06 0.05 0.008 0
line [AssyDrwTop] -0.06 0.19 -0.02 0.15 0.008 0
line [AssyDrwTop] -0.02 0.15 -0.06 0.11 0.008 0
line [BoundaryTop] -0.07 -0.045 -0.07 0.345 0.008 0
line [BoundaryTop] -0.07 0.345 0.38 0.345 0.008 0
line [BoundaryTop] 0.38 0.345 0.38 -0.045 0.008 0
line [BoundaryTop] 0.38 -0.045 -0.07 -0.045 0.008 0
centroid [BoundaryTop] 0.15 0.15
arc [SilkTop] 0 0.075 0.005 0 0 0.008
arc [AssyDrwTop] 0 0.075 0.005 0 0 0.008
@pins
pin 5 0.3 0.3 0 rnd-075-035 thru
pin 6 0.2 0.3 0 rnd-075-035 thru
pin 7 0.1 0.3 0 rnd-075-035 thru
pin 8 0 0.3 0 rnd-075-035 thru
pin 2 0.1 0 0 rnd-075-035 thru
pin 3 0.2 0 0 rnd-075-035 thru
pin 4 0.3 0 0 rnd-075-035 thru
pin 1 0 0 0 sq-070-035 thru
```

Ilustración 41: Datos de los pads en MicroSim

Tipo Pad	Formato
THT cuadrado	sq-*lado-drill*
THT rectangular	rect-*lado1-lado2-drill*
THT circular	rnd-*diametro-drill*
THT ovalado	oval-diametro1-*diametro2-drill*
SMD cuadrado	sq-*lado*
SMD rectangular	rect-*lado1-lado2*
SMD circular	rnd-*diametro*
SMD ovalado	oval-diametro1-*diametro2*

Tabla 2: Equivalencias de pads

5.11.2. KiCad

Los pad se escriben de forma similar en KiCad:

```
)(module DIP-8_W7.62mm (layer F.Cu) (tedit 5A02E8C5)
  (at 233.65 155.71 90)
  (descr "8-lead though-hole mounted DIP package, row spacing 7.62 mm (300 mils)")
  (tags "THT DIP DIL PDIP 2.54mm 7.62mm 300mil")
  (fp_text reference Um1 (at 3.81 -2.667 0) (layer F.SilkS)
    (effects (font (size 1.524 1.524) (thickness 0.2032)))
  )
  (fp_text value DIP-8_W7.62mm (at 3.81 9.95) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (fp_arc (start 3.81 -1.33) (end 2.81 -1.33) (angle -180.000000) (layer F.SilkS) (width 0.12)
  (fp_line (start 1.635 -1.27) (end 6.985 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 -1.27) (end 6.985 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 6.985 8.89) (end 0.635 8.89) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 8.89) (end 0.635 -0.27) (layer F.Fab) (width 0.1))
  (fp_line (start 0.635 -0.27) (end 1.635 -1.27) (layer F.Fab) (width 0.1))
  (fp_line (start 2.81 -1.33) (end 1.16 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 -1.33) (end 1.16 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 1.16 8.95) (end 6.46 8.95) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 8.95) (end 6.46 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start 6.46 -1.33) (end 4.81 -1.33) (layer F.SilkS) (width 0.12))
  (fp_line (start -1.1 -1.55) (end -1.1 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start -1.1 9.15) (end 8.7 9.15) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 9.15) (end 8.7 -1.55) (layer F.CrtYd) (width 0.05))
  (fp_line (start 8.7 -1.55) (end -1.1 -1.55) (layer F.CrtYd) (width 0.05))
  (pad 5 thru_hole circle (at 7.62 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 6 thru_hole circle (at 7.62 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 7 thru_hole circle (at 7.62 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 8 thru_hole circle (at 7.62 0) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 2 thru_hole circle (at 0 2.54) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 3 thru_hole circle (at 0 5.08) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 4 thru_hole circle (at 0 7.62) (size 0.75 0.75) (drill 0.35) (layers *.Cu *.Mask))
  (pad 1 thru_hole rect (at 0 0) (size 0.70 0.70) (drill 0.35) (layers *.Cu *.Mask))
  (fp_text user %R (at 3.81 3.81) (layer F.Fab)
    (effects (font (size 1 1) (thickness 0.15)))
  )
  (model ${KISYS3DMOD}/Package_DIP.3dshapes/DIP-8_W7.62mm.wrl
    (at (xyz 0 0 0))
    (scale (xyz 1 1 1))
    (rotate (xyz 0 0 0))
  )
```

Ilustración 42: Datos de los pads en KiCad

Su estructura es:

(pad *num_pad* throu_hole *forma_pad* (at *posX posY*) (size *dim1 dim 2*) (drill *drill*) (layers *.Cu *.Mask)

Si no tuviese drill, es decir, fuese SMD, cambia ligeramente:

(pad *num_pad* smd *forma_pad* (at *posX posY*) (size *dim1 dim2*) (layers *.Cu *.Mask)

En caso de los pads circulares y cuadrados, *dim1* y *dim2* son idénticas.

5.11.3. Transformación

Aplicando la misma transformación de coordenadas y unidades:

$$posX_{KiCad} = 25.4 * posX_{MicroSim} + 240 - 25.4 * \frac{1}{2} * ancho$$

$$posY_{KiCad} = 190 - 25.4 * posY_{MicroSim} + 25.4 * \frac{1}{2} * alto$$

$$dim1_{KiCad} = 25.4 * dim1_{MicoSim}$$

$$dim2_{KiCad} = 25.4 * dim2_{MicroSim}$$

$$drill_{KiCad} = 25.4 * drill_{MicroSim}$$

6. Funcionamiento nivel usuario de la aplicación

En este apartado vamos a explicar el funcionamiento de nuestra aplicación desarrollada paso a paso, de manera clara y concisa. Para apoyarnos usaremos como ejemplo práctico un archivo ".pca" correspondiente al diseño de un interruptor dinámico, como se muestra en la Ilustración 43.

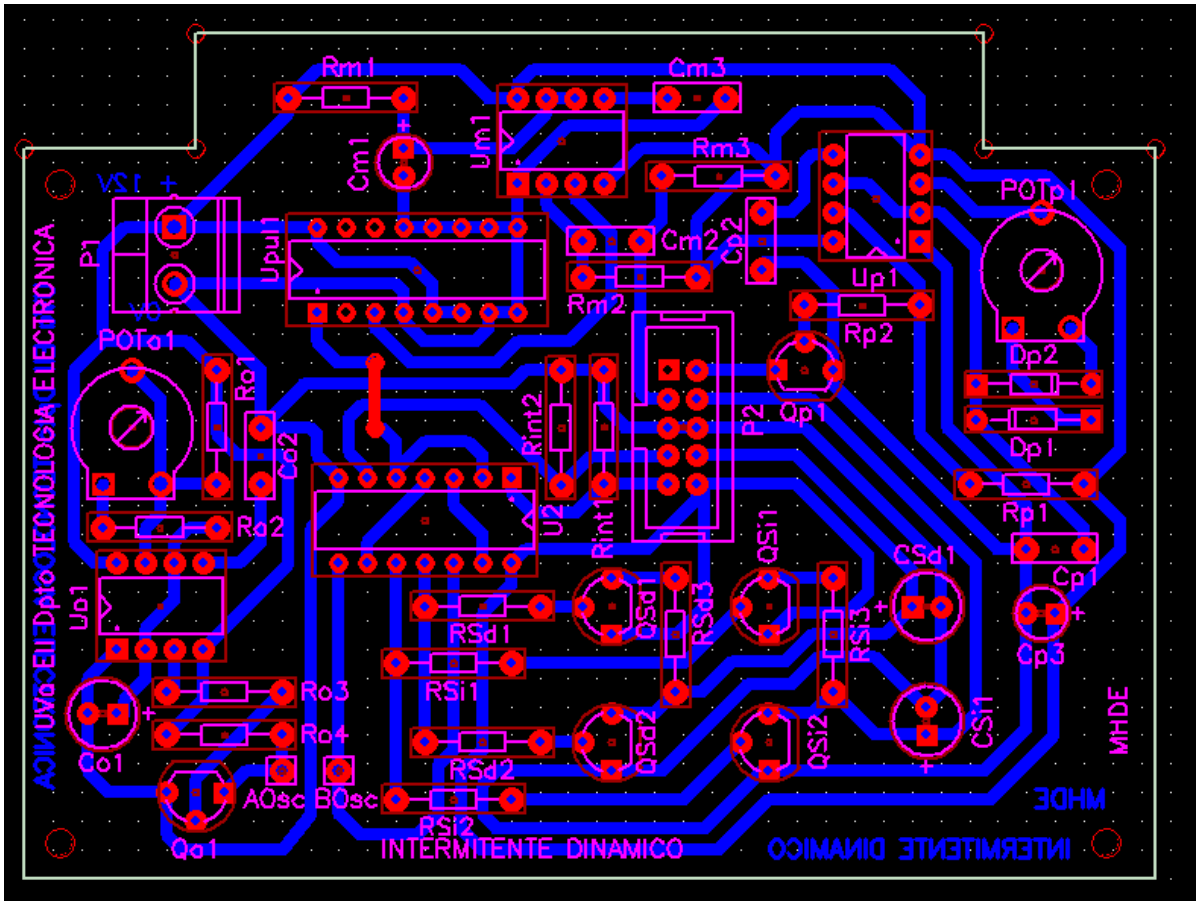

Ilustración 43: Diseño de un intermitente dinámico en MicroSim

6.1. Instalación

Para poder trabajar con el programa, lo primero que deberemos hacer es ejecutar el instalador y seguir las instrucciones pertinentes.

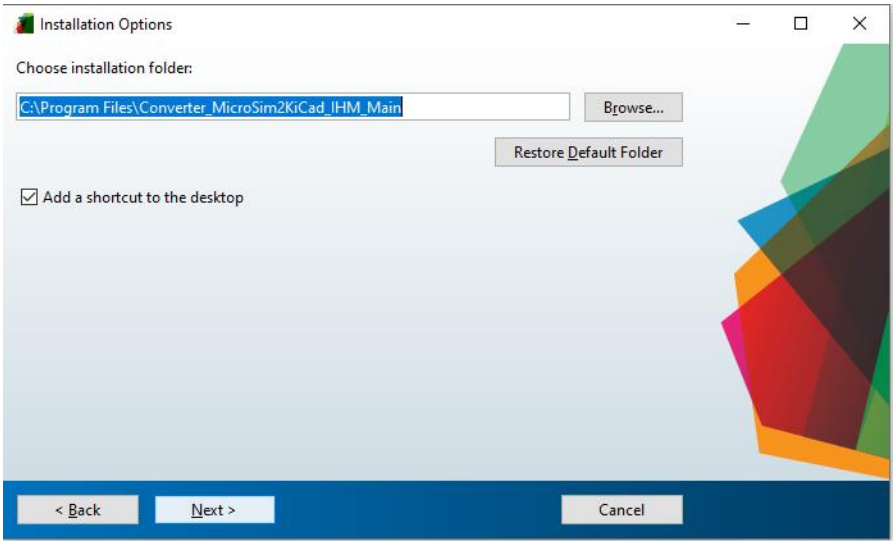

Ilustración 44: Ventana del instalador de la aplicación

El instalador permite crear un acceso rápido en el escritorio para su fácil acceso, y nos permitirá elegir la ruta de instalación. A mayores requiere la descarga e instalación de Matlab Runtime, una serie de librerías que permiten el correcto funcionamiento de la herramienta. Este paso de la instalación requiere de una conexión a internet y tarda varios minutos. En caso de que la máquina donde estemos realizando la instalación, ya posea estos archivos, el instalador lo detectará automáticamente.

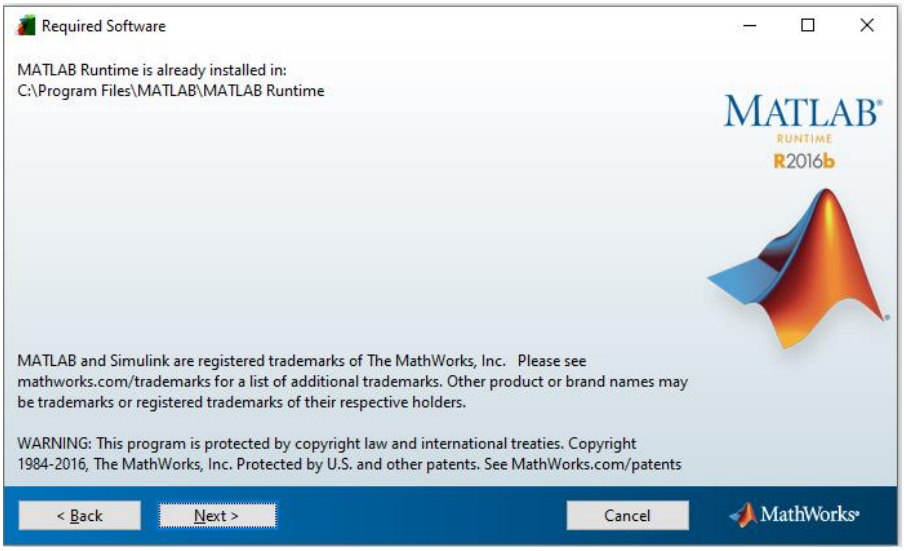

Ilustración 45: Detección de Matlab Runtime en la máquina

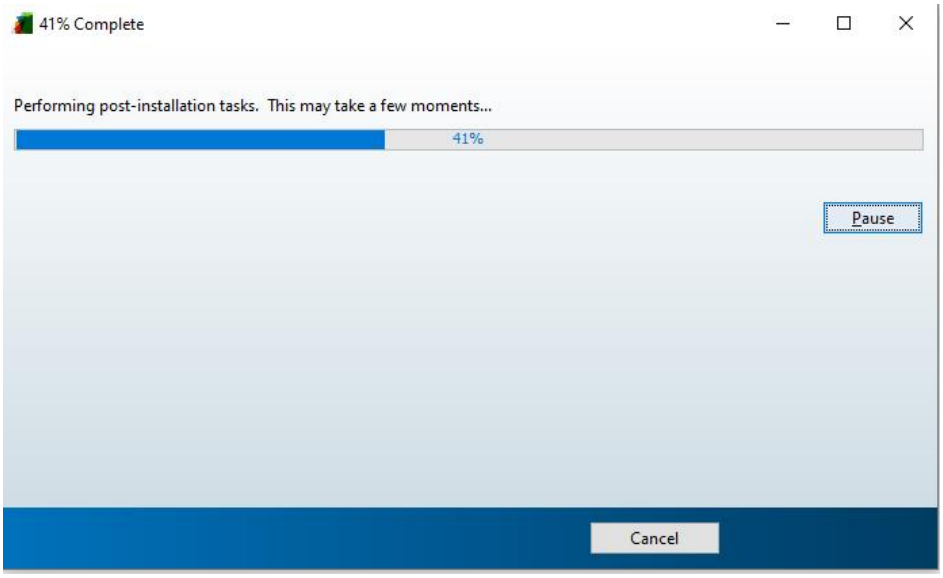

Ilustración 46: Estado de la instalación

El programa necesita una carpeta que utiliza como destino para guardar toda la información interna, incluyendo las librerías de componentes. Para ello se crea la ruta C:\CacheMatlab y con una variable de entorno se fija como la ruta que utiliza de memoria cache. El propio programa gestiona automáticamente todos estos pasos la primera vez que se ejecuta.

Una vez finalizado el proceso, ya podremos hacer uso de la aplicación desarrollada.

Ilustración 47: Icono de acceso directo

6.2. Cargar Archivo

En el panel de carga (figura X) tenemos todo lo necesario para leer los archivos ".pca" que hemos generado con PcBoards de MicroSim.

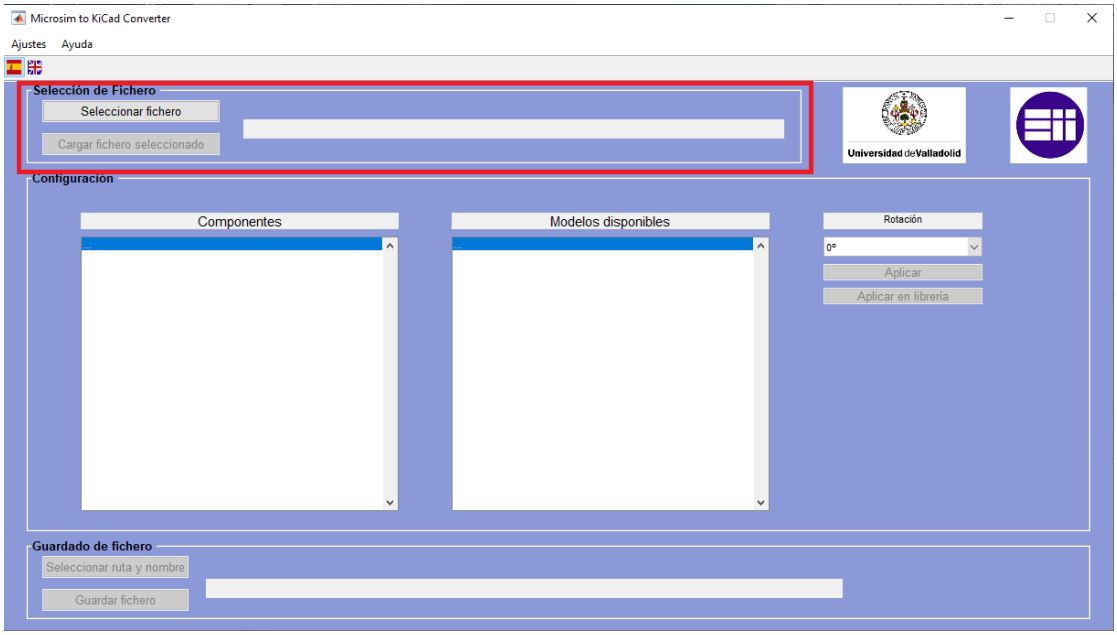

Ilustración 48: Ventana principal de la aplicación

Lo primero es pulsar el botón "Seleccionar Fichero". Se abrirá una ventana pidiendo seleccionar el archivo que queremos cargar, donde podremos navegar a través de los directorios para encontrar dicho archivo. Sólo se admitirán archivos con el formato correcto, es decir, ".pca".

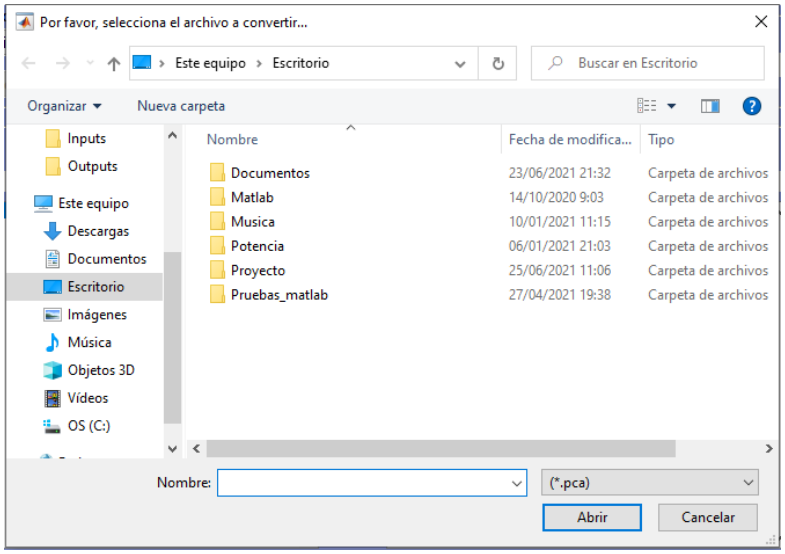

Ilustración 49: Pantalla de selección de fichero

Buscaremos el directorio donde se encuentre el archivo a convertir, en nuestro caso será INT_DINAM_2021.pca y hacemos clic en Abrir.

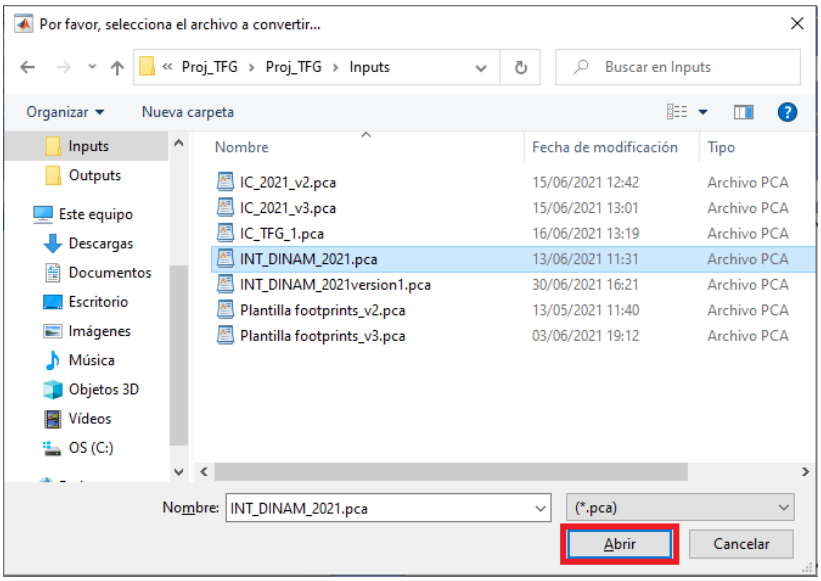

Ilustración 50: Selección de archivo

Ahora el programa ya tiene localizado el fichero, y aparecerá su ruta completa en el campo de la derecha. Asimismo, el botón de "Cargar Fichero Seleccionado" pasará a estar habilitado.

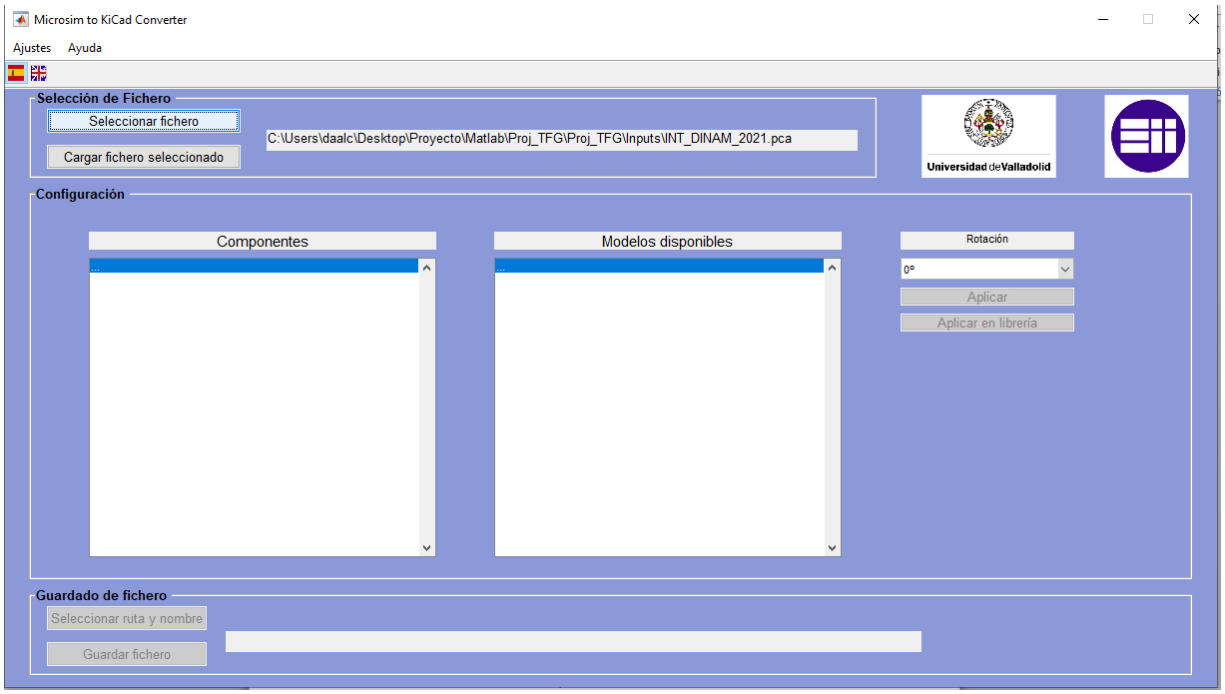

Ilustración 51: Carga de fichero

Ya solo nos queda pulsar en el botón “Cargar Fichero Seleccionado” para que la aplicación lea toda la información necesaria y la guarde internamente. En el panel de “Configuración” aparecerán todos los componentes encontrados en la lista de la izquierda, y también se habilitarán las opciones de rotación y de selección del archivo creado.

6.3. Selección de Modelos

Ahora centraremos nuestra atención en las dos listas del panel central “Configuración”. Como hemos mencionado, en la lista de la izquierda aparecerán nombrados todos los componentes que se han encontrado en la librería interna de nuestro programa. Este nombre corresponde al nombre asignado a ese componente en MicroSim PSpice.
En la lista de la derecha se mostrarán todos los modelos de KiCad asociados al componente de la lista izquierda seleccionado actualmente.

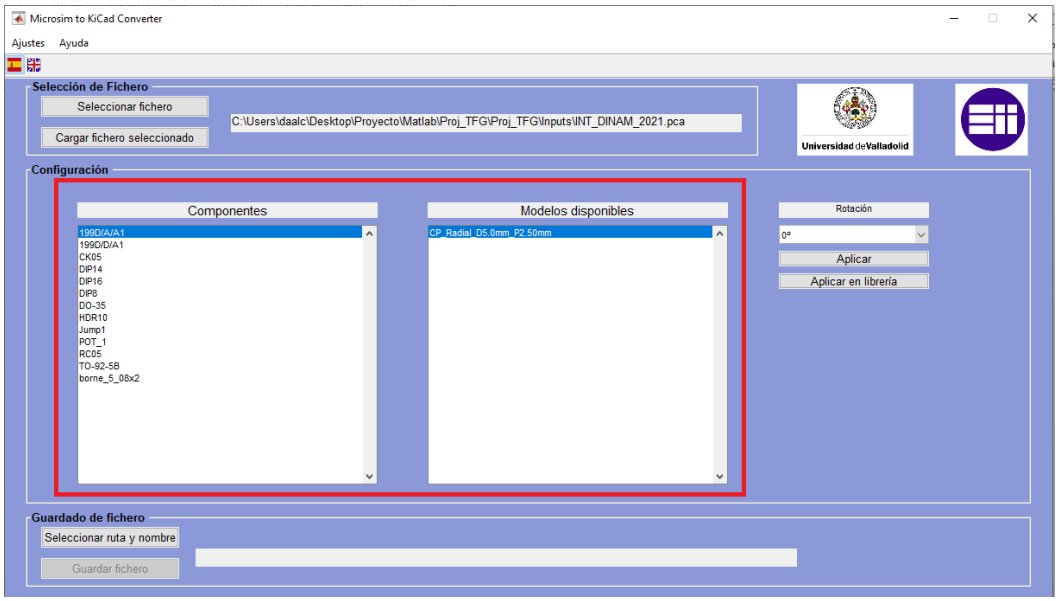

Ilustración 52: Lista de componentes y modelos asociados

En la mayoría de los casos, para cada componente de MicroSim corresponde un único modelo de KiCad. Sin embargo, puede ocurrir que haya varios que correspondan al mismo footprint. Esto es debido a que MicroSim trabaja con planos (dos dimensiones) y nosotros queremos obtener un modelo tridimensional. Como no tenemos datos sobre la altura de los componentes o su forma pueden existir varios modelos que encajen en las dimensiones del footprint. Un caso claro son los condensadores verticales, de los cuales conocemos su diámetro y su distancia entre pines, pero puede haber varios de distintas alturas.

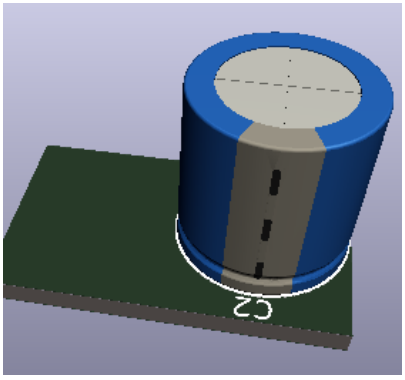

Ilustración 53: Modelo de un condensador

En nuestro ejemplo, nos pasa un caso similar con el condensador CK05.

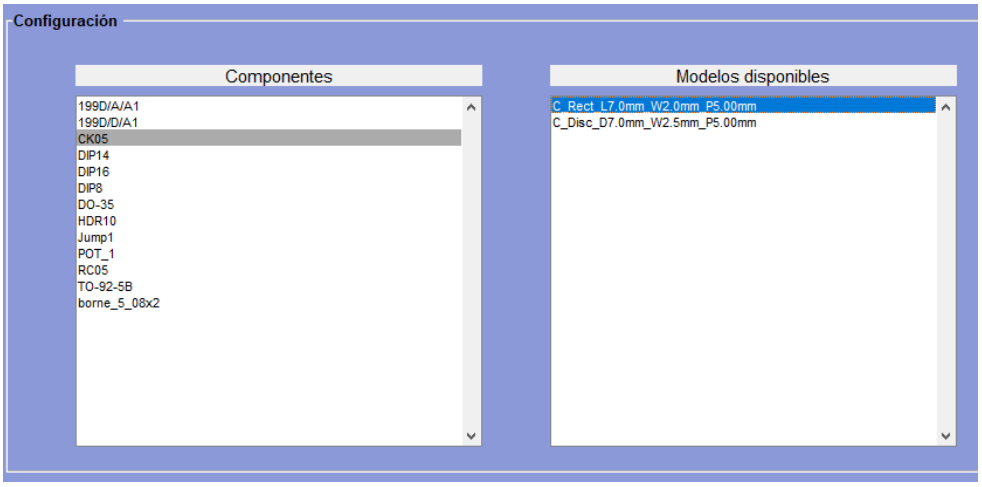

Ilustración 54: Selección de modelos

Durante la creación del prototipo de la placa de este interruptor dinámico, percibimos que este componente disponía de dos tipos disponibles que encajaban en nuestro footprint: Encapsulados y de disco.

Ilustración 55: Distintos encapsulados

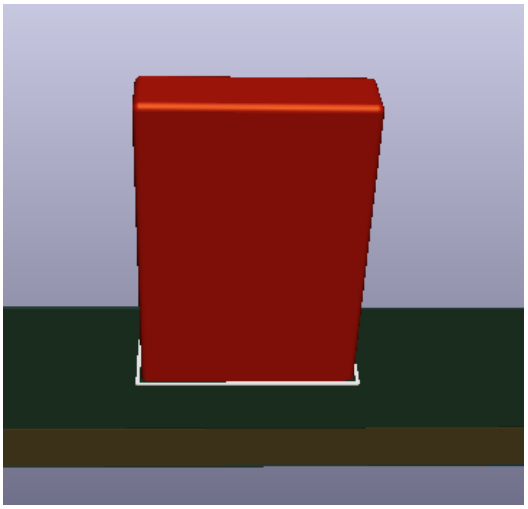

Ilustración 56: Modelo 3D de encapsulado

Ilustración 57: Condensadores de disco

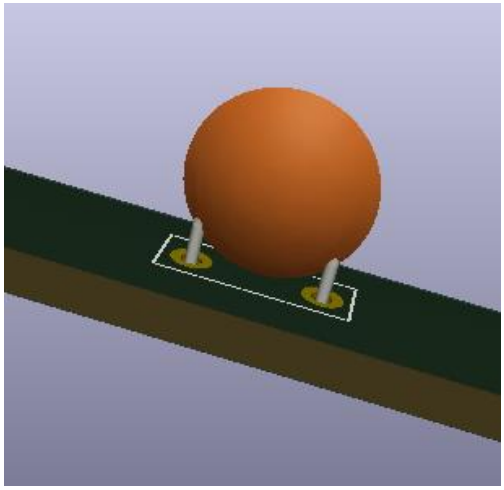

Ilustración 58: Modelo 3D de un condensador de disco

Al disponer de ambos modelos en nuestra librería, tenemos la libertad de poder escoger que modelo queremos aplicar a cada placa.

Por defecto siempre aparecerá seleccionado el primer modelo disponible de la lista, pero siempre es aconsejable revisar todos los componentes y comprobar que, si hubiese varios modelos, seleccionamos el más acorde a nuestros intereses.

6.4. Guardar Archivo

Una vez satisfechos con nuestras elecciones, vamos a guardar el archivo final para comprobar los resultados obtenidos. Para ello tenemos en el panel de "Guardado de Fichero" dos botones.

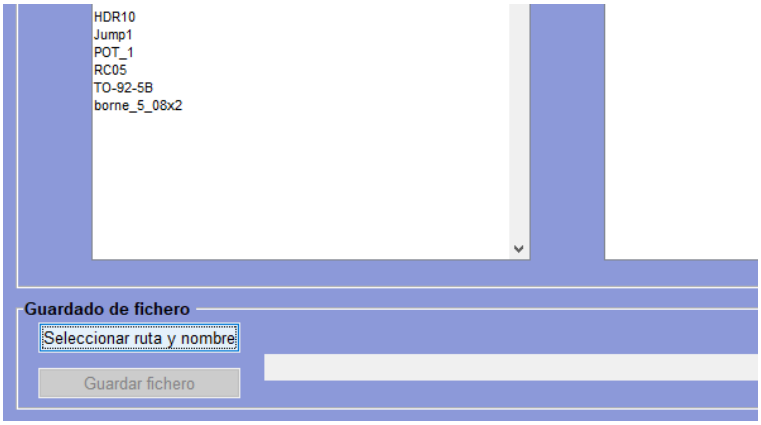

Ilustración 59: Guardar fichero

El primero creará una ventana que nos pedirá el nombre y la ruta del archivo que creará el programa, de manera muy similar a seleccionar el archivo que cargábamos.

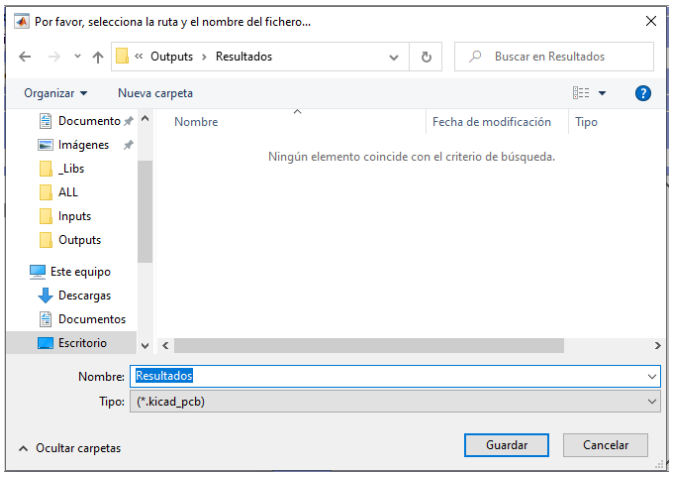

Ilustración 60: Nombre del archivo final

Una vez escogido la carpeta y el nombre, hacemos clic en “Guardar”. Aparecerá la ruta completa en el campo de la derecha, y se habilitará el botón de “Guardar Fichero”.

Ilustración 61: Ruta de guardado

Por último solo queda pinchar en “Guardar Fichero” y, tras un mensaje de confirmación, ya tendremos creado nuestro archivo “.kicad_pcb” con los modelos tridimensionales implementados.

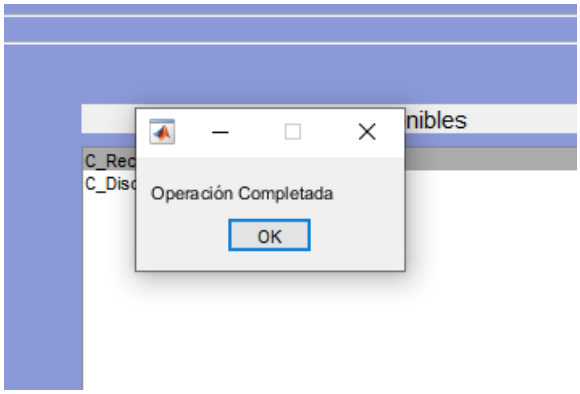

Ilustración 62: Mensaje de confirmación

6.5. Visualización de Resultados

Para visualizar el resultado final, tan sólo tendremos que abrir el archivo recién creado con el programa KiCad. Al abrirlo, visualizaremos el plano de la placa con todos sus elementos y propiedades, que hemos transferido desde el archivo de MicroSim.

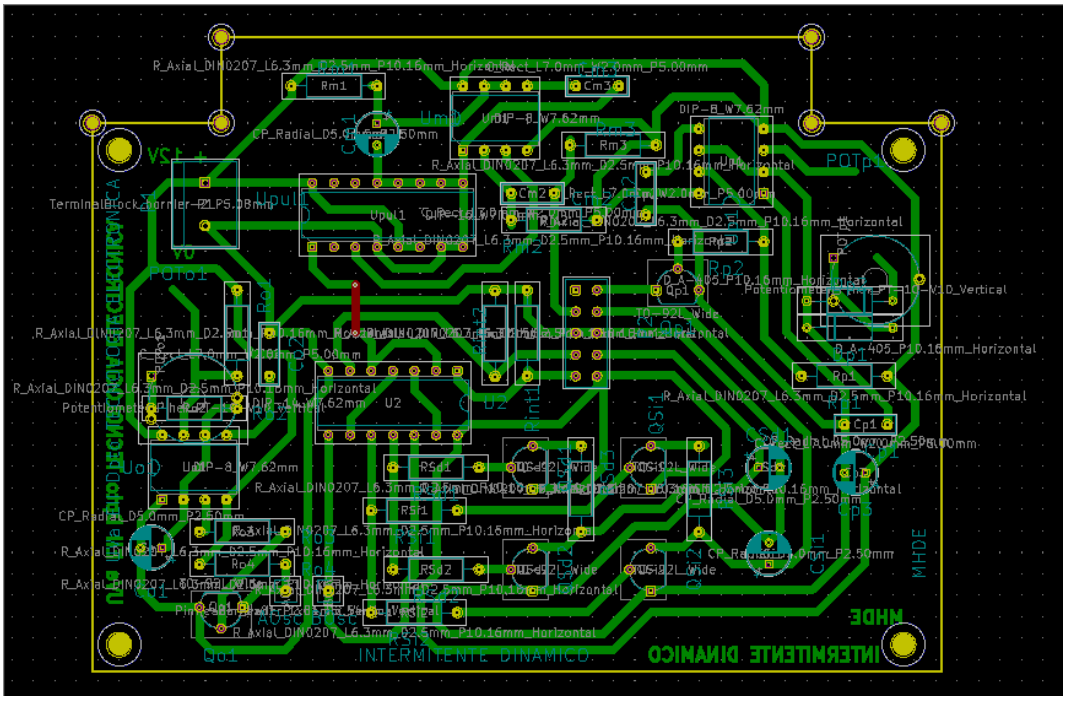

Ilustración 63: Visualización de la PCB generada

Para poder observar el modelo tridimensional, en la barra de herramientas de KiCad, haremos clic en la opción “Ver” y luego en “Visor 3D”.

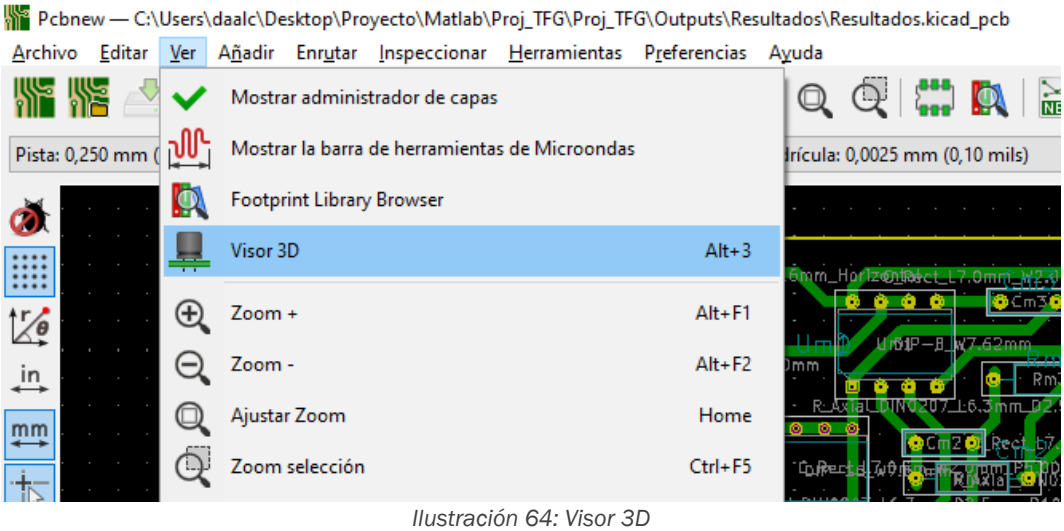

Ilustración 64: Visor 3D

Tras unos segundos, veremos el resultado final. Podemos movernos por la placa y ampliar la vista si queremos fijarnos en una parte concreta.

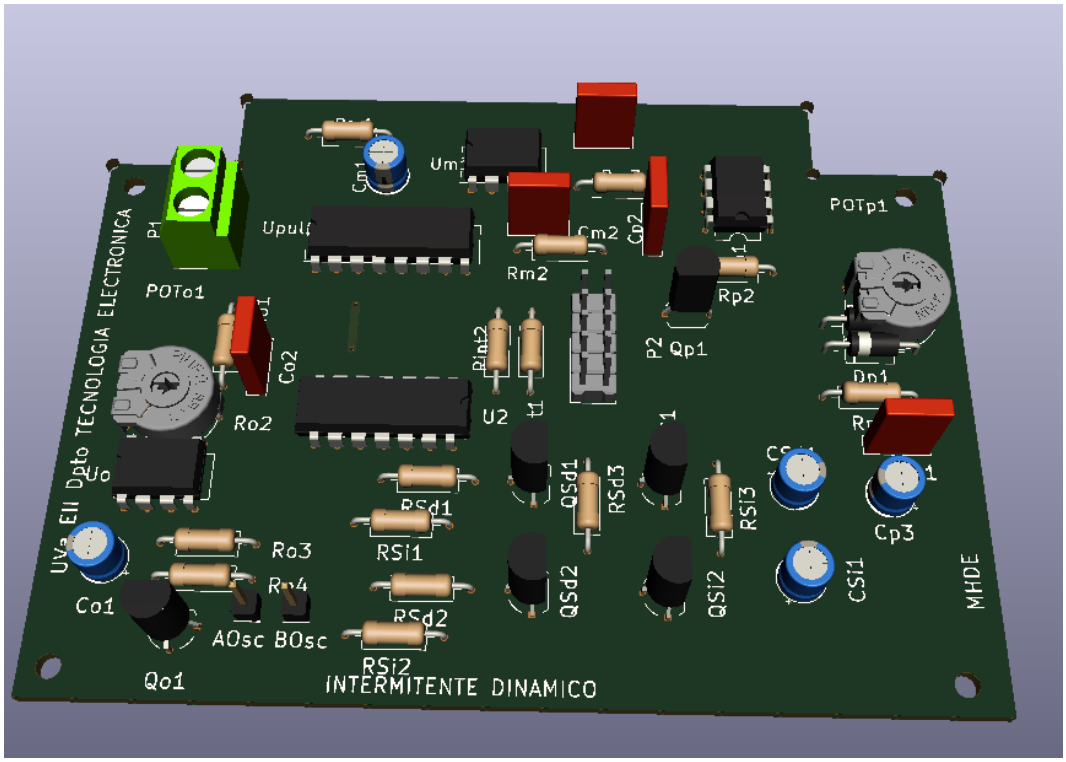

Ilustración 65: Modelo tridimensional de la PCB

En el caso de que no se visualicen las pistas ni la capa superior de la placa, como se muestra en la figura X, iremos a “Opciones de visualización” dentro del menú “Preferencias”. Allí marcaremos la opción de representación “Modo estilo real” y aceptamos.

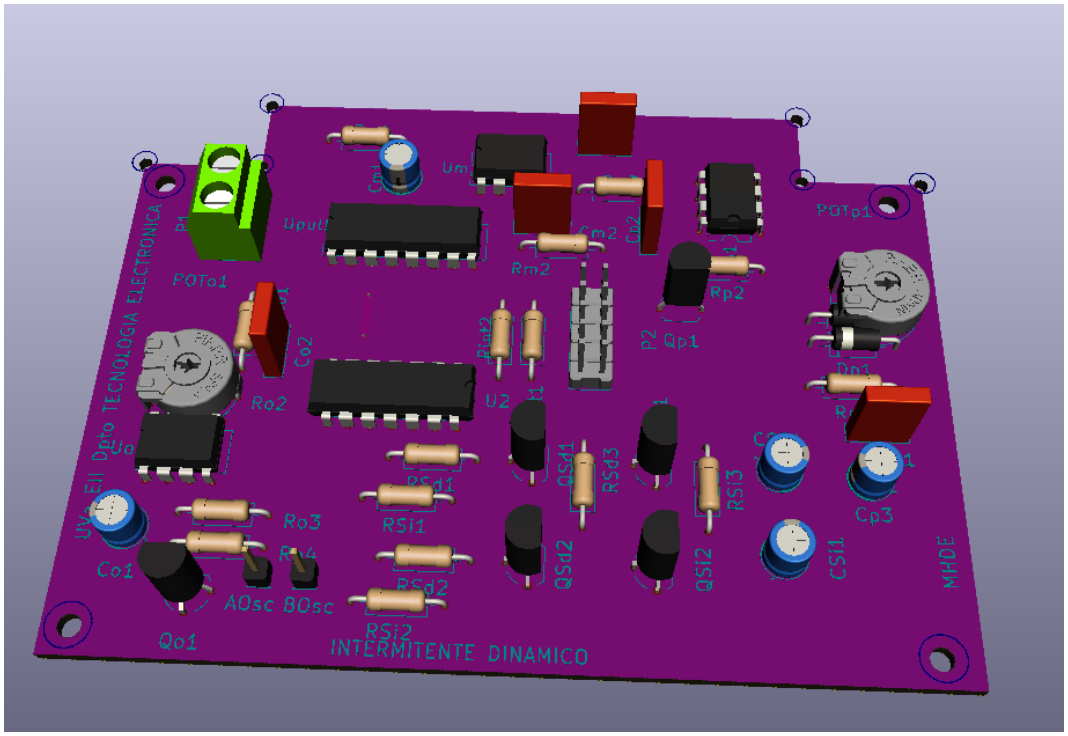

Ilustración 66: Modo de visualización no real

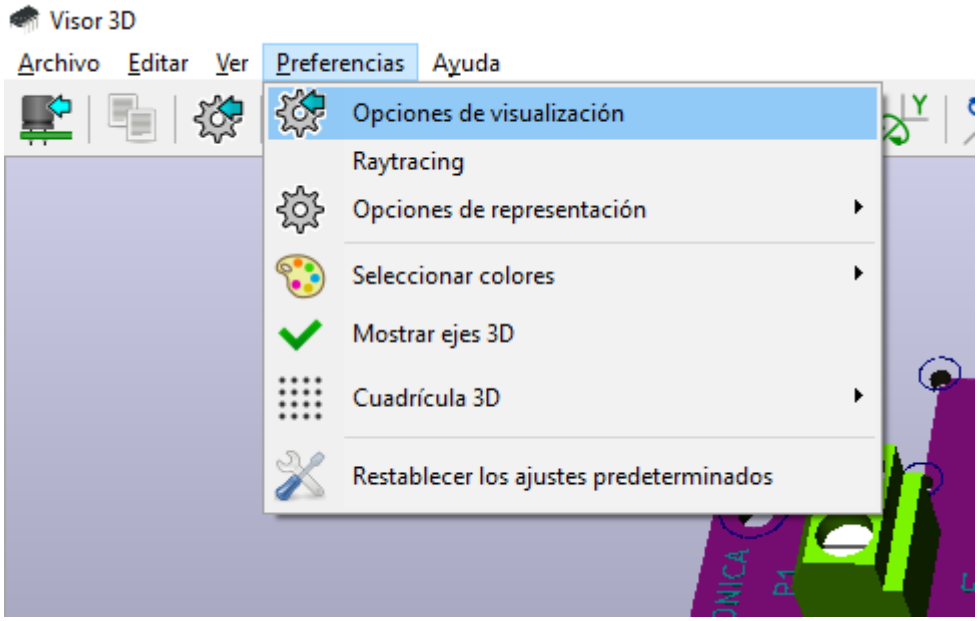

Ilustración 67: Opciones de visualización

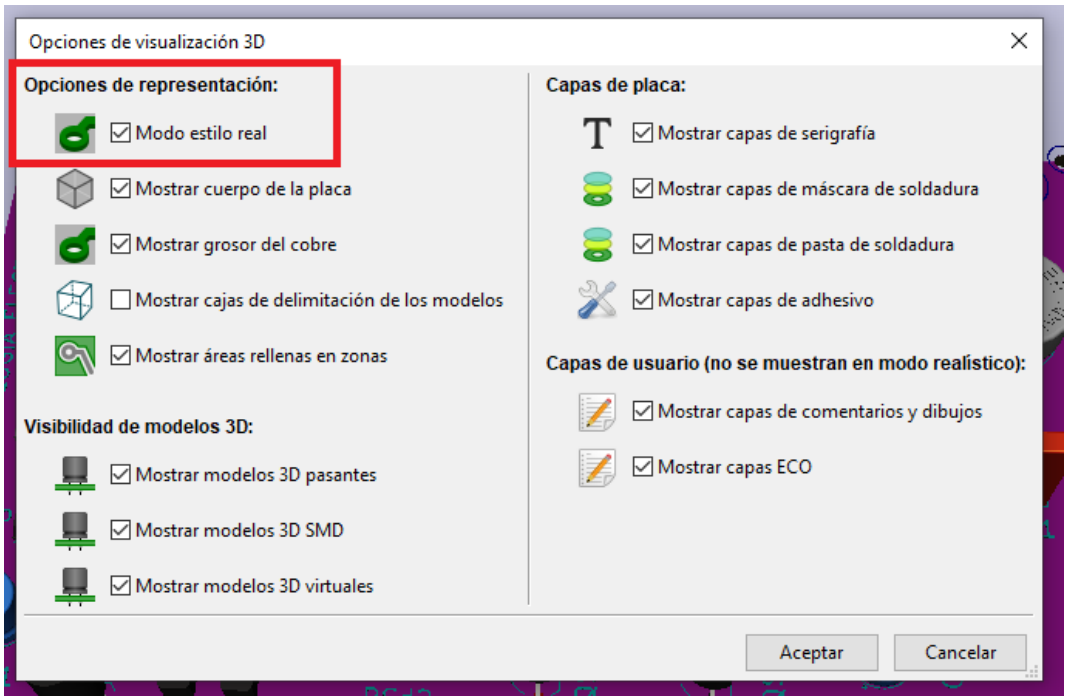

Ilustración 68: Modelo estilo real

6.6. Rotación de Componentes

Es posible que el footprint de MicroSim y el modelo de KiCad, estén definidos con orientaciones distintas. En nuestro ejemplo, puede verse claramente que en el resultado final, los potenciómetros no están colocados en el mismo sentido que en el diseño original, y no coinciden los pads con las pistas, e incluso se superponen con otros componentes.

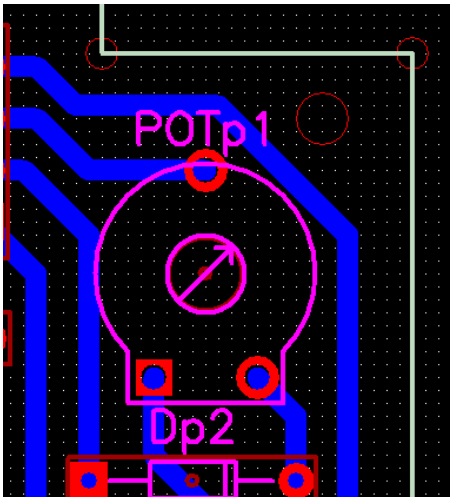

Ilustración 69: Posición original del potenciómetro

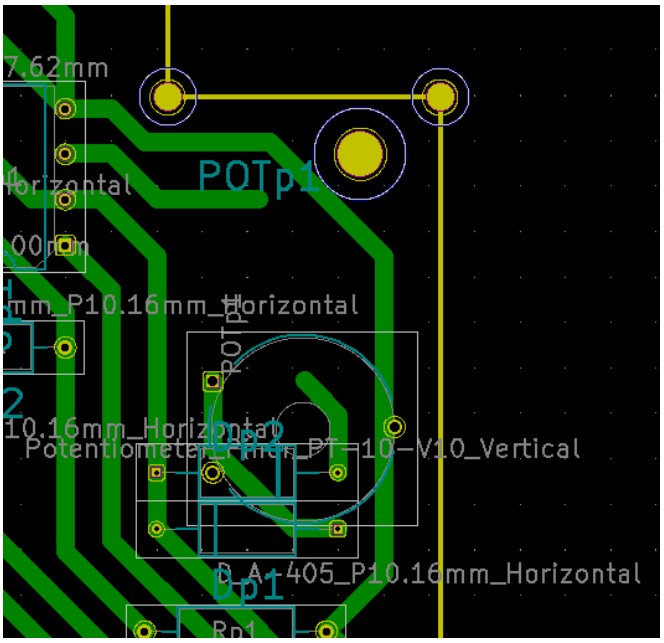

Ilustración 70: Posición del potenciómetro en el archivo generado

Este tipo de desajustes no se pueden realizar de manera automática, ya que es imposible que el programa detecte la orientación del diseño original del footprint creado en MicroSim.

Para solucionar este problema, el usuario deberá comprobar una vez obtenido el resultado final si hay algún componente que su orientación no corresponda con el original. Una vez identificado el componente problemático, deberemos hacer uso de la funcionalidad de rotación.

Dentro del panel “Configuración” a la derecha, tenemos el menú de rotación.

Deberemos seleccionar en las listas, el componente y modelo al que vayamos a modificar su ángulo inicial, y el giro que realizará. En nuestro caso buscamos el potenciómetro y queremos que realice un giro de 90º. Cabe destacar que todos los giros se consideran en sentido anti horario. Si quisiéramos girar 90º en sentido horario, sería lo mismo que girar 270º o, como está definido en la aplicación, -90º.

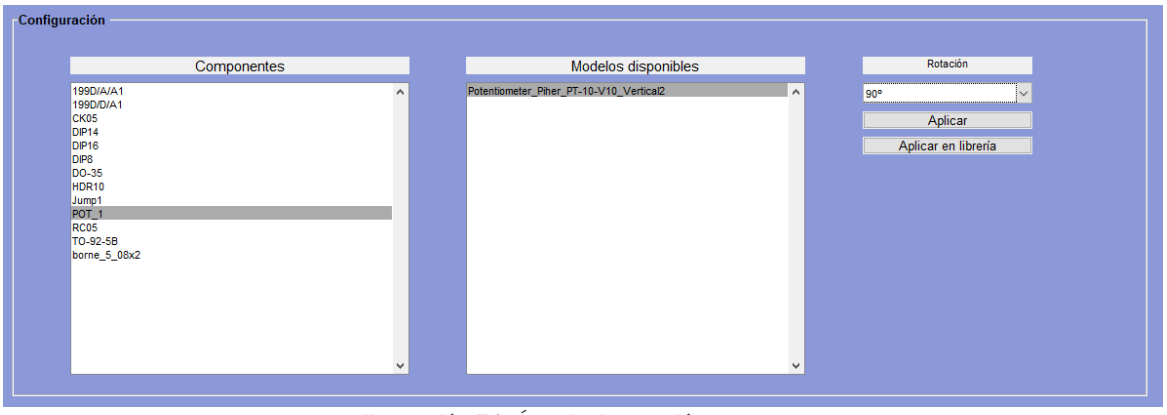

Ilustración 71: Ángulo de rotación

Una vez seleccionados el componente, modelo y giro, tenemos dos opciones: “Aplicar” y “Aplicar en librería”. Si pulsamos en “Aplicar” ese giro será efectivo para todos esos modelos, pero tan sólo en el archivo que estamos creando. Además recibiremos un mensaje de confirmación.

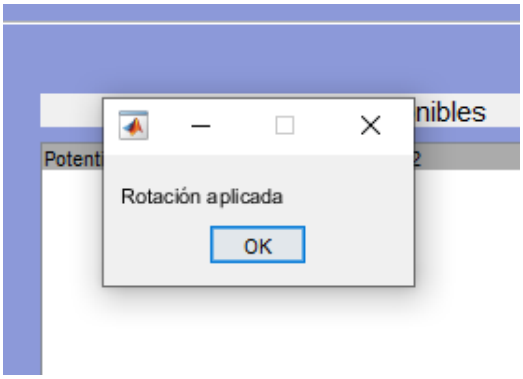

Ilustración 72: Mensaje de confirmación

Esto nos permite volver a guardar el archivo “.kicad_pcb” y volver a visualizar la placa, para comprobar que el giro ha sido realizado y si era el ángulo correcto.

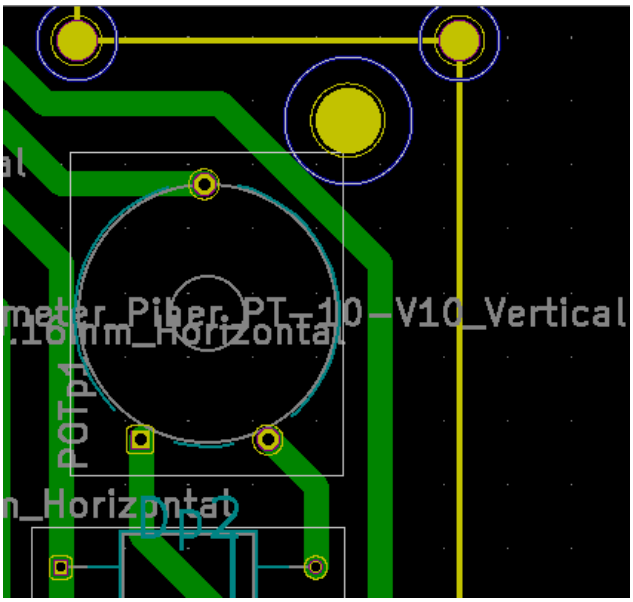

Ilustración 73: Potenciómetro en su posición correcta

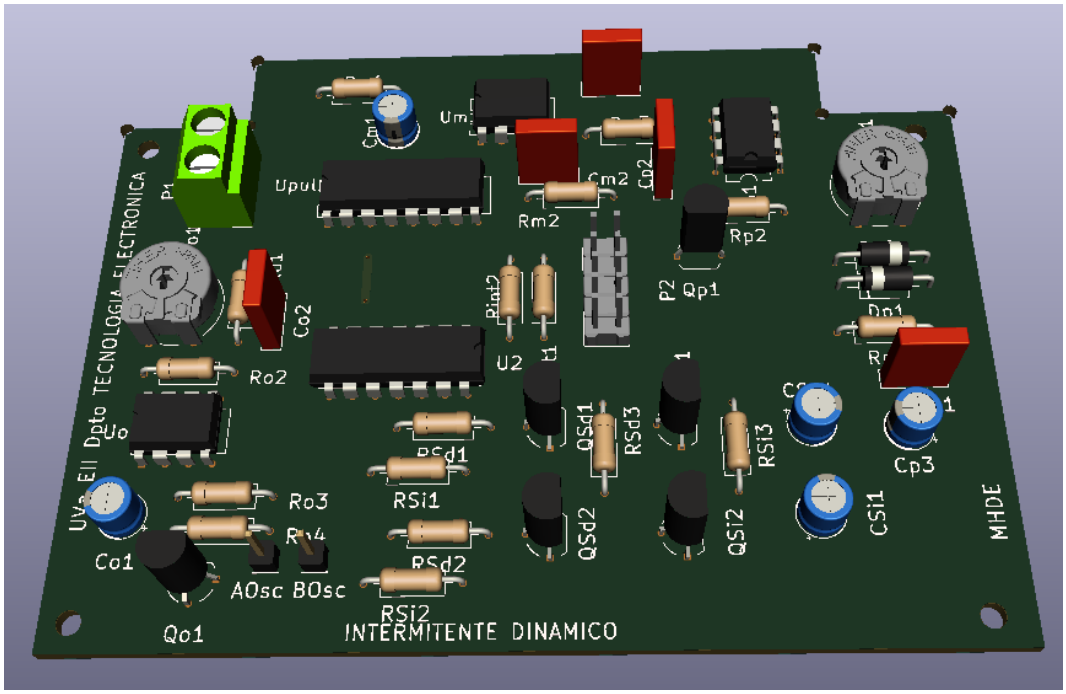

Ilustración 74: Cara superior del modelo 3D de la placa corregida

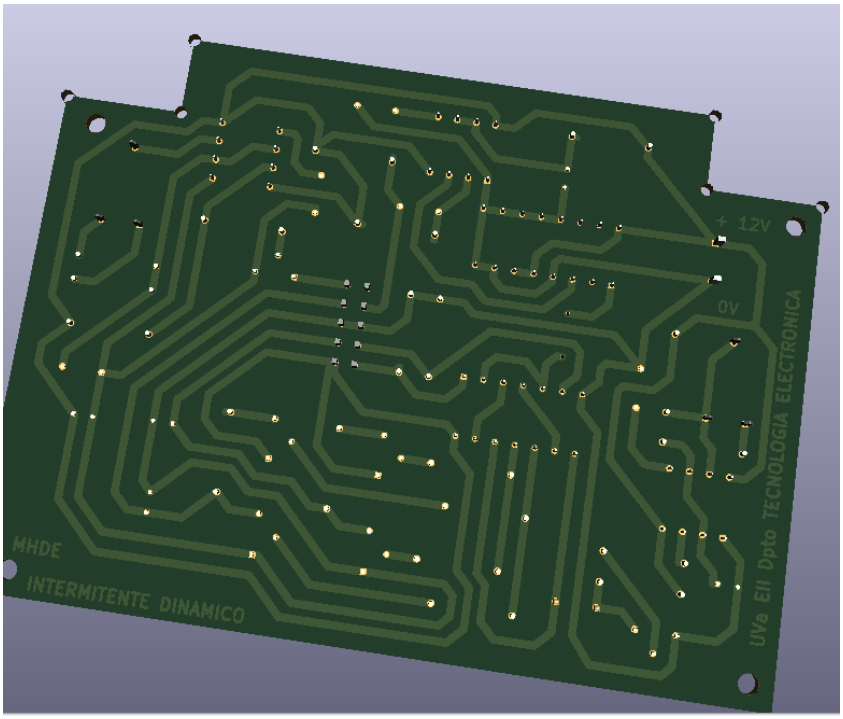

Ilustración 75: Cara inferior del modelo 3D de la placa corregida

Una vez confirmado que la rotación es la acertada, podemos utilizar el botón "Aplicar en librería" para que ese modelo seleccionado siempre aparezca con esa rotación por defecto, corrigiendo así el problema inicial. Este ángulo no marca la orientación final, si no que se suma al ángulo inicial que tenga marcado en la librería, por defecto 0º.

A continuación mostramos en la Ilustración 76 una imagen del prototipo físico construido a partir del archivo utilizado en este ejemplo, para que podamos apreciar la precisión del modelo tridimensional con el resultado final.

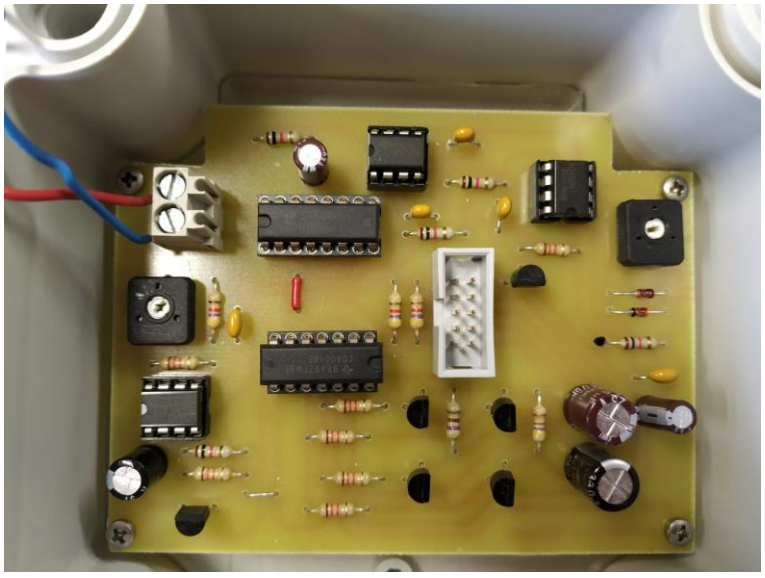

Ilustración 76: Prototipo físico final

6.7. Selección de idioma

El programa viene implementado en dos idiomas, castellano e inglés. Para cambiar de idioma, tan sólo habrá que pulsar en la bandera correspondiente, situadas en la esquina superior izquierda, bajo la barra de herramientas.

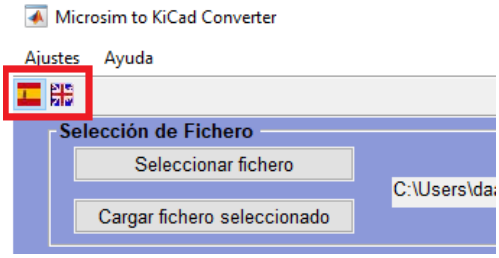

Ilustración 77: Selección de Idioma

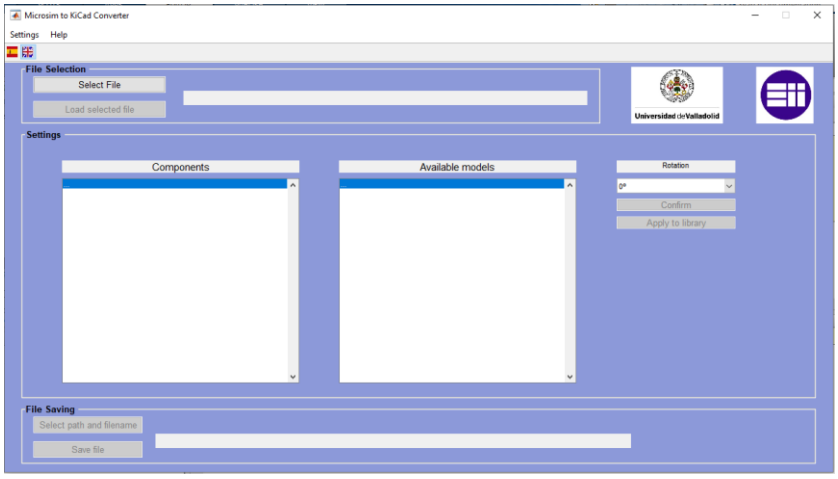

Ilustración 78: Ventana principal traducida al inglés

6.8. Guía de Usuario

En la barra de herramientas, dentro de “Ayuda”, encontramos la opción “Guía de Usuario”, que nos abrirá automáticamente un PDF con un manual de usuario donde poder aprender el funcionamiento de esta herramienta.

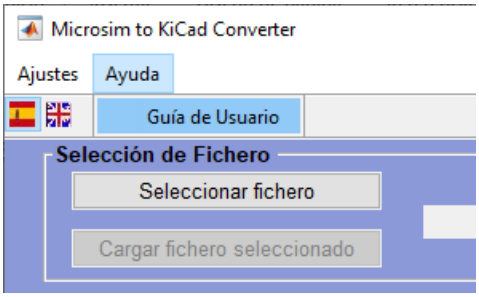

Ilustración 79: Guía de Usuario

6.9. Añadir componentes a la librería

Puede darse el caso de que alguno de los componentes existentes en la placa no se encuentren definidos en la biblioteca actual. Cuando esto ocurra, al presionar el botón de "Cargar fichero seleccionado" aparecerá una ventana emergente advirtiéndonos sobre ello, y enumerando todos los componentes que no se hayan encontrado.

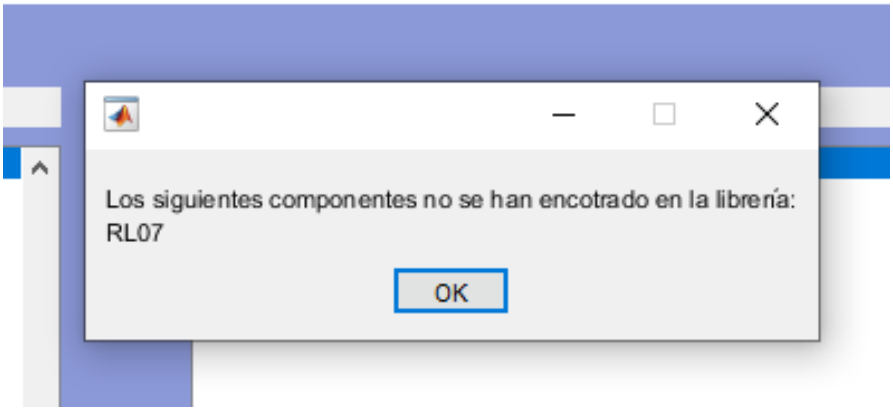

Ilustración 80: Mensaje de componentes no encontrados

La herramienta está diseñada para poder ampliar su librería de forma progresiva, de modo que una vez añadido un nuevo componente ya lo reconocerá de manera automática para todos los futuros archivos con los que trabajemos. A continuación se describirá el proceso para añadir un nuevo componente.
Lo primero que necesitamos es el modelo de KiCad que corresponda con el footprint de MicroSim que estamos buscando. Tenemos dos posibles sitios donde buscarlo.

El primero es dentro de las propias librerías básicas de KiCad. Para ello, abrimos el KiCad Eeschema y añadimos cualquier componente, por ejemplo una resistencia. Ahora accedemos a "Asignar Huellas…" dentro de la opción "Herramientas" de la barra de opciones superior.

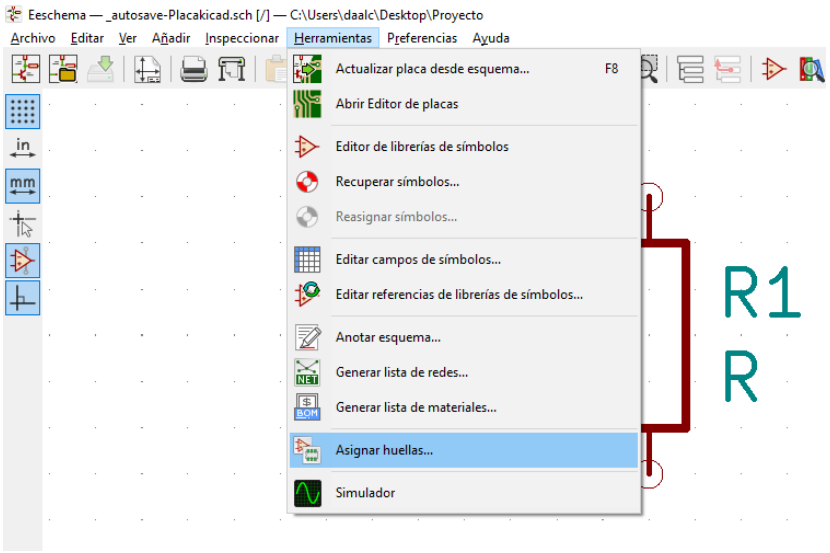

Ilustración 81: Asignación de huellas

Tras cargar todos los modelos de la librería, proceso que puede llevar unos minutos, aparecerá la siguiente ventana.

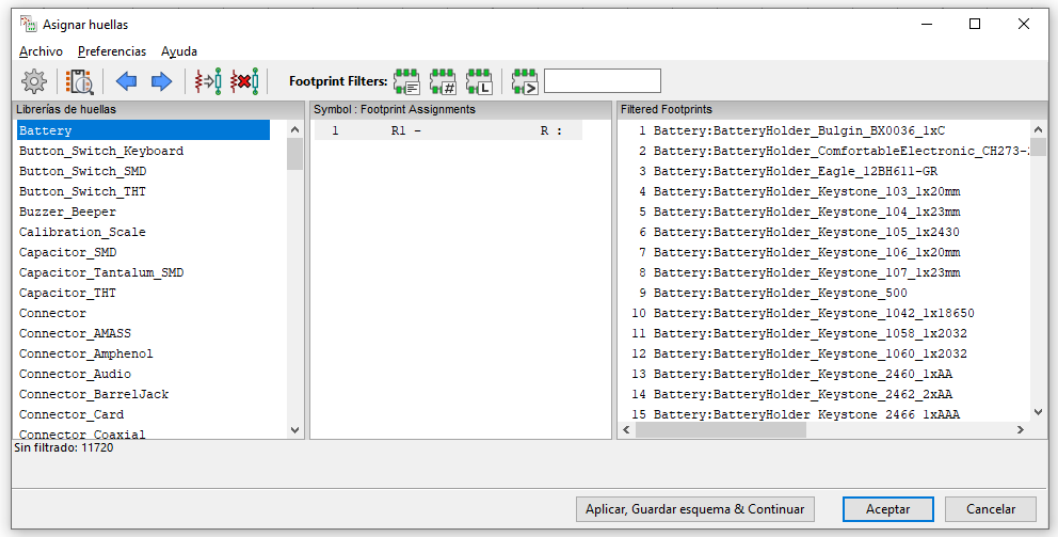

Ilustración 82: Librerías de footprint de KiCad

En la parte izquierda, bajo la etiqueta “Librería de huellas”, se hayan distintas categorías de componentes para facilitar la búsqueda, como pueden ser resistencias, condensadores, diodos, conectores, etc. En la parte derecha, bajo “Filtered Footprints” se encuentran los modelos dentro de esas categorías. También se pueden filtrar y buscar modelos por palabras clave o número de patillas.

En nuestro caso estamos buscando RL07, que es una resistencia THT, asique seleccionamos esa categoría. Para saber cuál es el modelo correspondiente de los muchos existentes, necesitamos saber las medidas de su footprint. Para ello, lo podemos medir desde PcBoards de MicroSim.

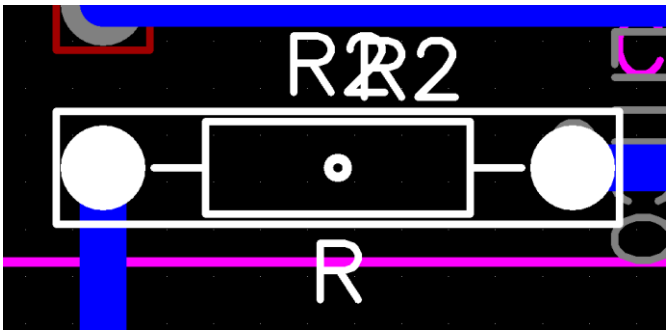

Ilustración 83: Footprint de RL07 en MicroSim

Recordemos que MicroSim trabaja en pulgadas, y KiCad en milímetros. Para que no haya confusiones vamos a “Tools” y pinchamos en “Options”.

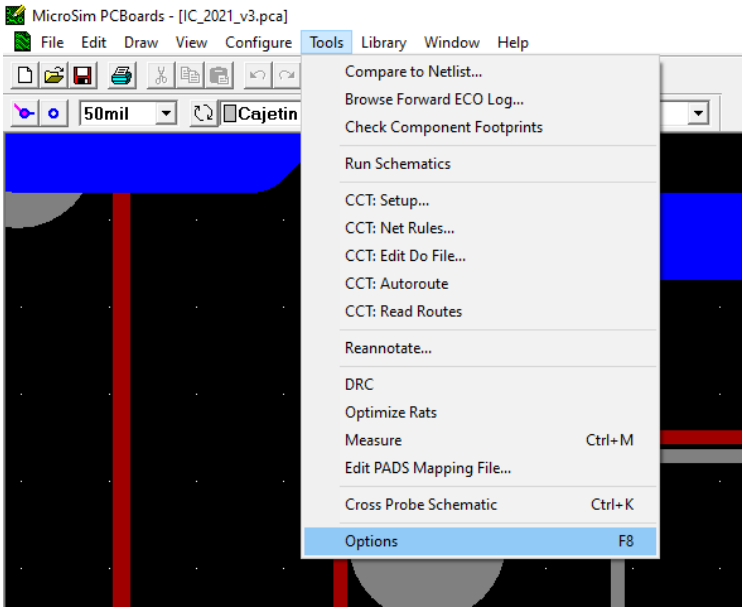

Ilustración 84: Menú de herramientas

Dentro de la ventana emergente, marcamos como unidad de medida los milímetros.

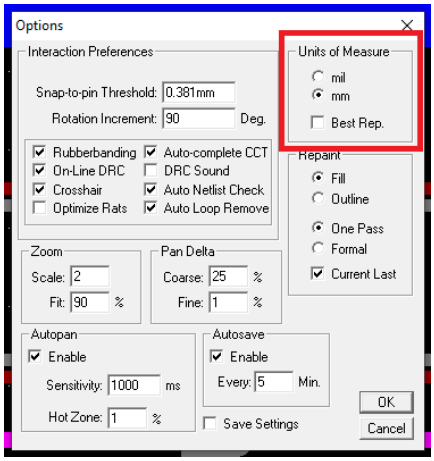

Ilustración 85: Mediciones en milímetros

Ahora usaremos la funcionalidad de medida, dentro de “Tools” y “Measure”, o Ctrl+M como acceso rápido.

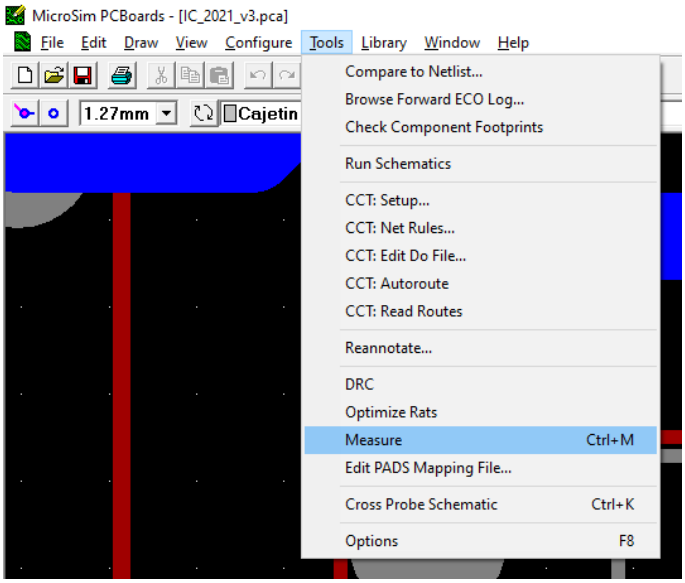

Ilustración 86: Herramienta de medición en MicroSim

Con las medidas de nuestro componente anotadas, volvemos a la asignación de huella de KiCad. Aunque la mayoría de los modelos tienen nombres significativos incluyendo sus medidas, es mejor asegurarse. Para ello, una vez encontrado el posible candidato hacemos clic derecho sobre él y “View footprint”.

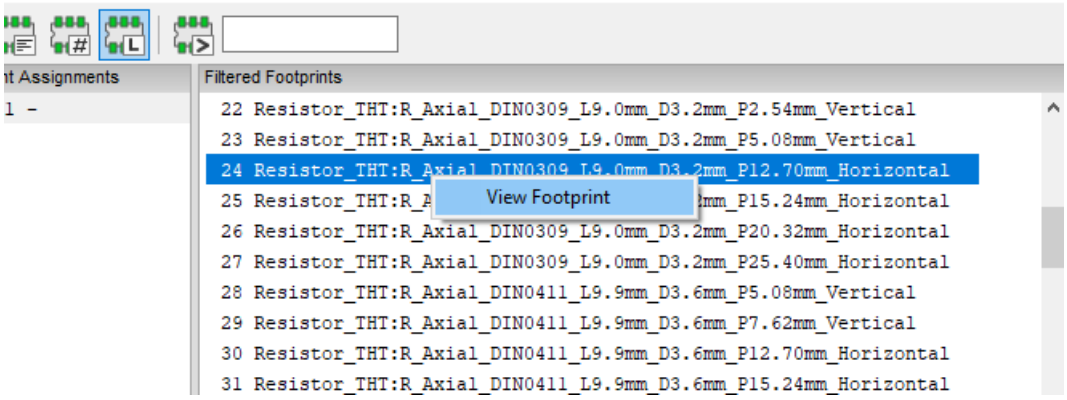

Ilustración 87: Ver footprint detallado

En la nueva ventana podremos estudiar el footprint con detenimiento y utilizar una regla de la barra de herramientas izquierda para poder verificar que todas las medidas se corresponden con nuestro footprint de MicroSim.

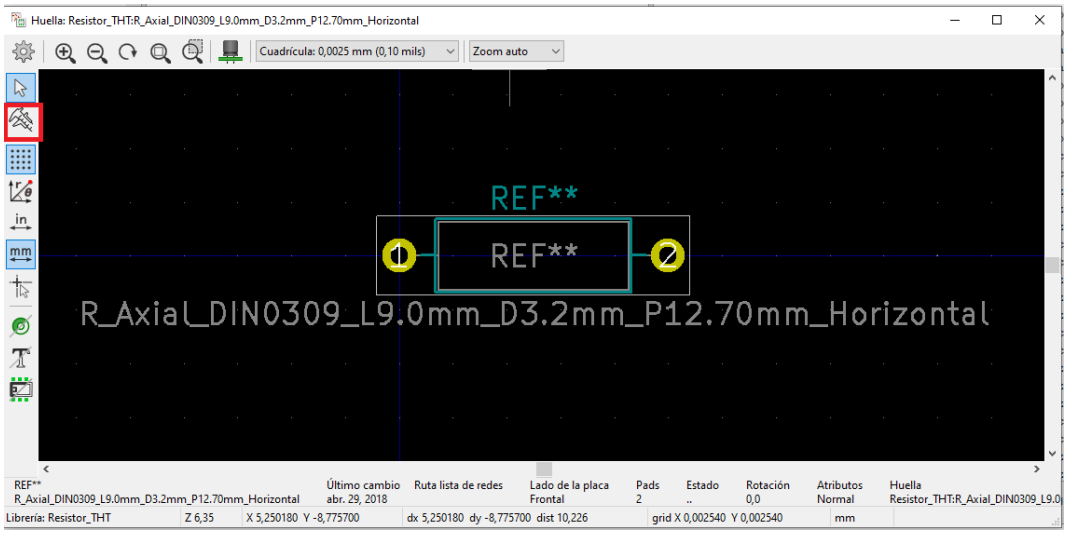

Ilustración 88: Herramienta de medición en KiCad

También tenemos una herramienta en la parte superior para poder ver su modelo en tres dimensiones.

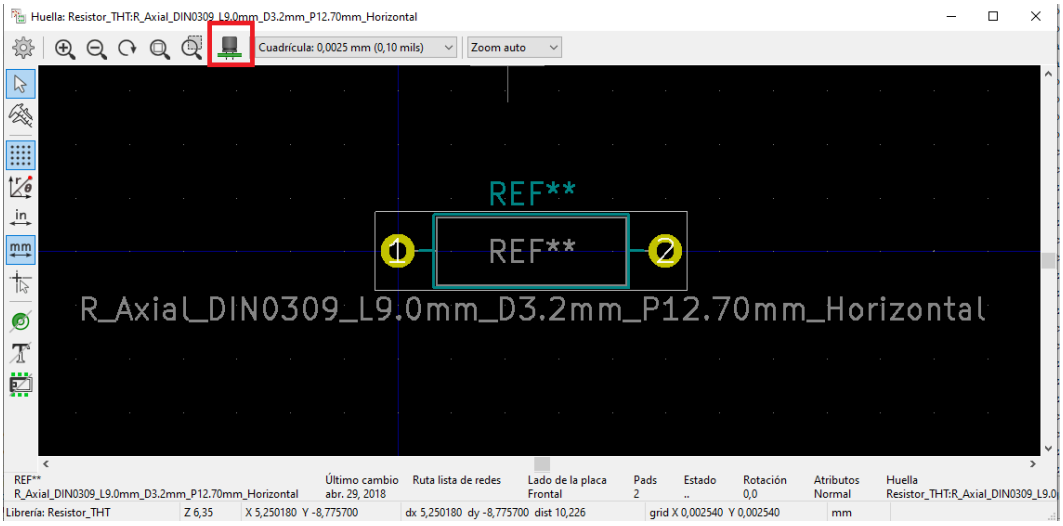

Ilustración 89: Herramienta Visor 3D

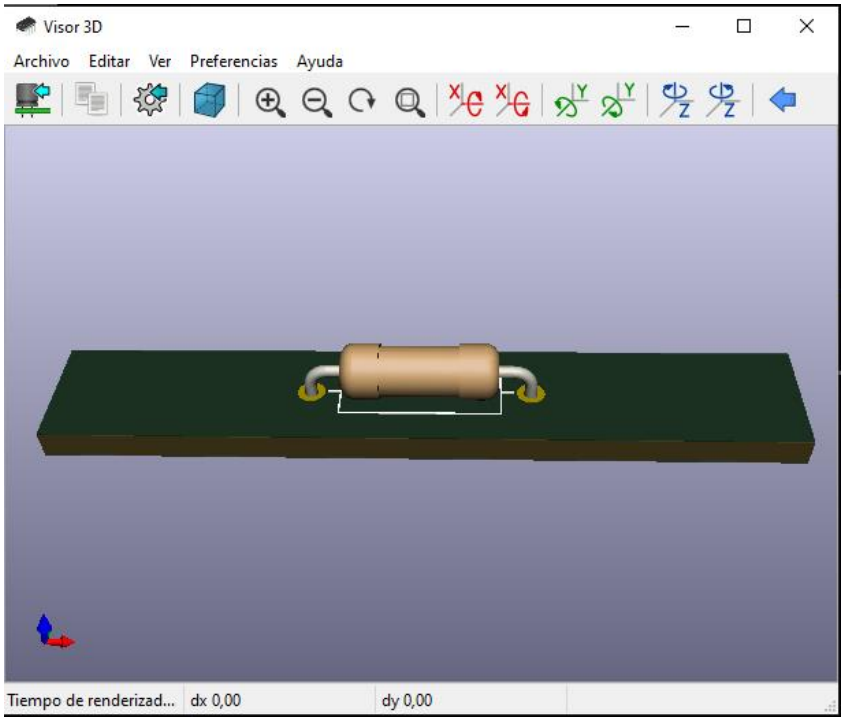

Ilustración 90: Modelo 3D de la resistencia RL07

Una vez encontrado el modelo debemos ir a la ruta donde se encuentra ese archivo. Para ello vamos al directorio donde esté instalado KiCad y seguimos el siguiente camino: C:\Program Files\KiCad\share\kicad\modules
Ahora vamos a la carpeta de la categoría donde se encontraba el modelo y lo buscamos.

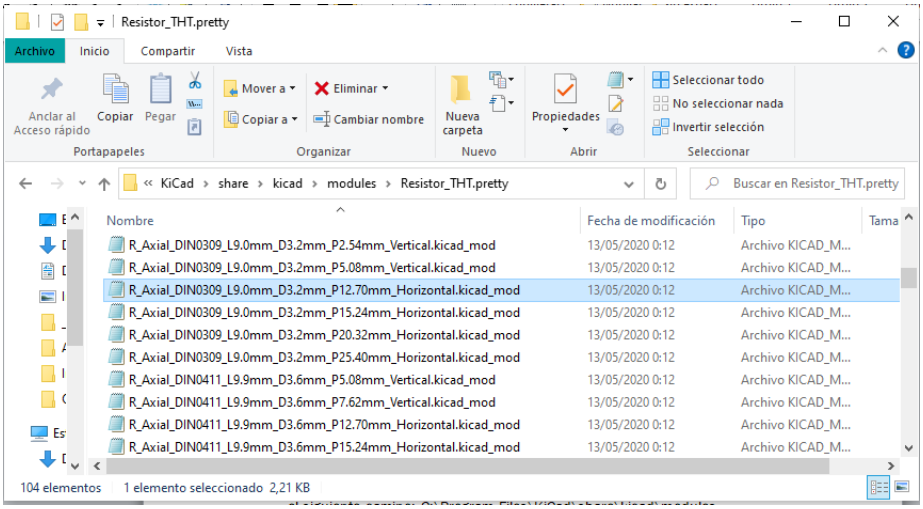

Ilustración 91: Archivo de la librería de KiCad

Deberemos copiar ese archivo y pegarlo en la carpeta "_Libs" dentro del directorio de nuestra herramienta. La ruta completa es:
C:\CacheMatlab\mcrCache9.1\Conver0\Archivos_Libs
Esta ruta es fácilmente accesible desde la aplicación, accediendo a "Abrir ruta de librería" dentro del menú "Ajustes".

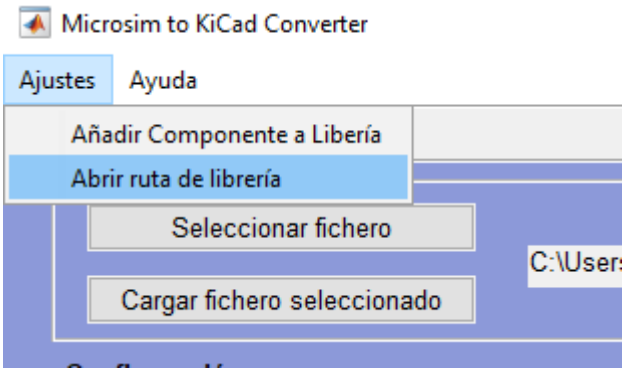

Ilustración 92: Abrir ruta de librería

Ahora que ya tenemos el modelo sólo queda volver a la ventana principal y el menú "Ajustes", hacer clic en "Añadir Componente a Librería".

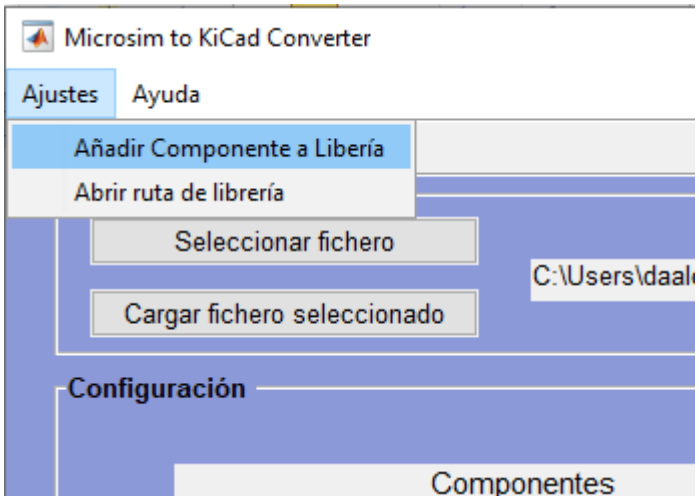

Ilustración 93: Añadir Componente a Librería

En la nueva ventana emergente deberemos rellenar los campos de nombre modelo MicroSim, en nuestro caso era el RL07, y en modelo KiCad, el nombre del archivo que acabamos de copiar, R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal, y presionamos aceptar. Es importante asegurarse de que ambos nombres están bien escritos o no el programa no será capaz de encontrarlos.

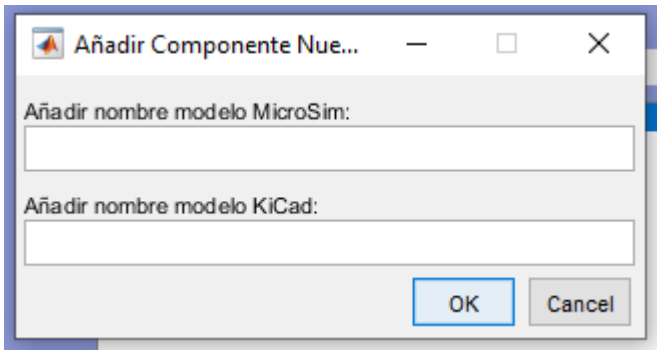

Ilustración 94: Ventana emergente para añadir componente

El componente ya se encuentra en nuestra librería y será reconocido en futuros proyectos.

7. Funcionamiento interno de la aplicación

En este capítulo vamos a explicar cómo hemos logrado las distintas funcionalidades de la herramienta a nivel técnico, sin entrar al código. En caso de querer entrar más en detalle, se adjunta todo el código con los comentarios necesarios para poder entender el flujo del mismo.

7.1. Diagrama de flujo

El código se ha estructurado de una manera muy lineal y con bloques muy bien definidos. Esto simplifica mucho la programación y las interacciones entre sus funciones. Además nos permite actualizar el código y ampliar sus funcionalidades en futuras versiones de una manera lógica sin tener que alterar el código ya escrito ni su estructura.

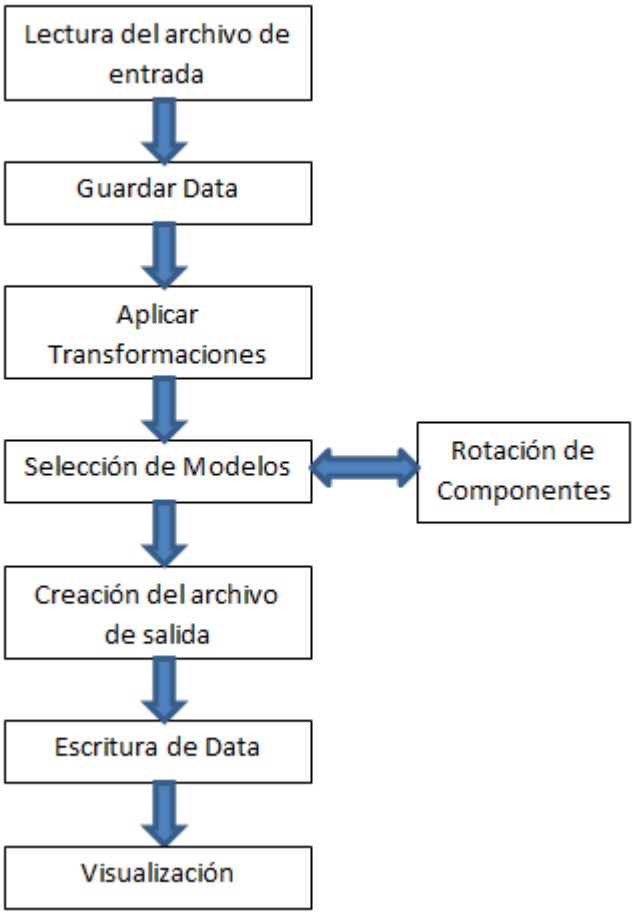

Ilustración 95: Diagrama de Flujo

7.2. Lectura de archivo

En este bloque, se pide al usuario que seleccione el archivo que queremos cargar. Una vez localizado en la ruta obtenida, el fichero se abre como un archivo de texto.

Dentro de toda la información que está en su interior, queremos obtener los datos necesarios para poder reescribirlos de tal forma que KiCad pueda comprenderlo y el resultado sea idéntico. Para ello nos basamos en la búsqueda de unas palabras clave para determinar el número de la línea donde se encuentra esa información. Por ejemplo, cada pista siempre está definida en una línea que comienza por los caracteres *@seg*. En la tabla X podemos encontrar todas las palabras clave que nos ayudan a encontrar la línea con la información pertinente.

Elemento a buscar	Palabras clave
Borde de placa	*line BoardOutline
Línea	*line
Arco	*arc
Texto	*text
Taladro	*hole
Componente	*component
Vía	@via
Pista	@seg

Tabla 3: Búsqueda de palabras clave

Una vez encontrado el índice (el número de la línea donde se encuentra esa palabra clave) utilizamos los conocimientos aprendidos en el capítulo 5 para buscar todas las variables necesarias para definir inequívocamente el elemento en cuestión, y las guardamos en la memoria interna.

7.3. Transformaciones

En este bloque nos dedicamos a realizar todas las transformaciones matemáticas de todas las variables de cada elemento de la placa, aplicando las fórmulas vistas en el capítulo 5. Por ello, nos resultó de gran ayuda haber realizado un estudio previo en profundidad y tan sólo tenemos que programar dichas fórmulas, manteniendo el formato de las variables para que no existan desajustes.

7.4. Selección de Modelos

Para tratar en profundidad el bloque de selección de modelos, hay que introducir previamente un fichero fundamental para el desarrollo de esta aplicación. Se trata de un simple fichero de texto, llamado "Correspondencias.txt" que se encuentra en la carpeta "_Libs" junto con los archivos de todos los modelos de KiCad.

En un principio, la aplicación iba a contener una tabla de equivalencias en su memoria interna, que sirviera de enlace entre el nombre del modelo MicroSim y el nombre de modelo KiCad, donde existiesen un conjunto de componentes definidos como los comúnmente más usados, según mi tutor. Sin embargo, esto planteaba un serio problema si en nuestra placa existía algún componente que no estuviese presente en dicho listado. Para ampliar ese listado, habría que entrar al código y modificarlo para añadir componentes, y luego compilarlo de nuevo. Para solucionarlo, se ideó que esa tabla estuviese escrita en un archivo externo, al cual el código pudiese acceder siempre que lo necesitase, y que fuese de fácil acceso para el usuario si en caso de que necesitase manipularlo.

En este fichero, las correspondencias se estructuran por filas de la siguiente manera:

Nombre_Componente_MicroSim Nombre_Modelo_KiCad Angulo_inicial

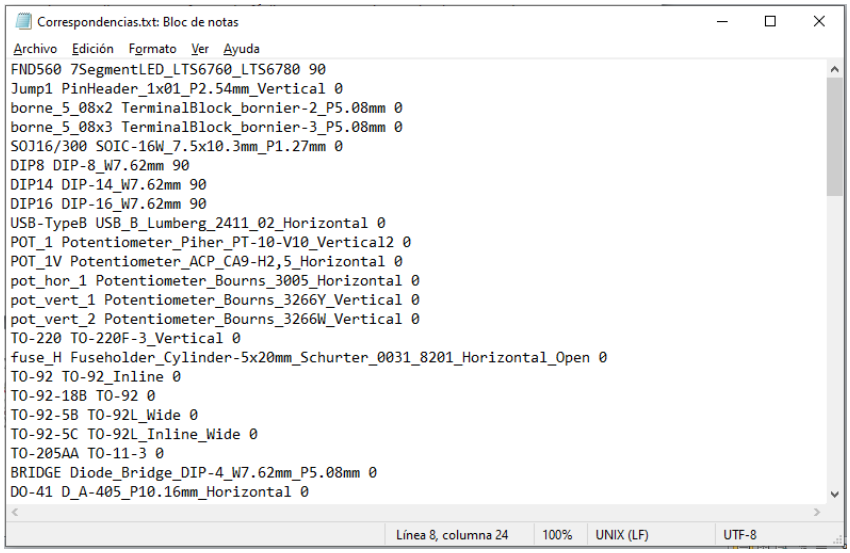

Ilustración 96: Archivo de correspondencias

Recordemos que tanto el *Nombre_Componente_MicroSim*, como el *Nombre_Modelo_KiCad* tienen que corresponderse con aquellos nombres asignados de manera precisa, incluyendo mayúsculas y minúsculas.

Una vez explicado el fichero de Correspondencias, podemos volver a la selección de archivos.

Durante la lectura del archivo .pca, obtuvimos el nombre de todos los componentes que se hayan presentes en la placa. Estos nombres se comparan con la primera columna del fichero Correspondencias.txt para clasificarlos en componentes encontrados y componentes ausentes. Los componentes ausentes se indicarán al usuario mediante una ventana emergente. Por otra parte los componentes encontrados se mostrarán en la lista izquierda de la ventana principal de la aplicación. Para cada componente, se establecerán los posibles modelos asociados, que corresponden a los nombres de la segunda columna del fichero Correspondencias.txt cuya primera columna coincida con el componente en cuestión.

Por defecto, el programa asigna el primer modelo encontrado de cada componente como el modelo escogido. Sin embargo, el usuario puede escoger el modelo que el elija simplemente seleccionándolo en la lista. El programa lee esa selección y la guarda en su memoria.

7.5. Rotación

Con esta funcionalidad podemos añadir una rotación base a todos los modelos seleccionados para corregir el posible desajuste que existiese en el diseño de los footprint entre ambos MicroSim y KiCad. Este cambio se aplica solo al archivo que generemos en ese momento y no es cambio permanente, lo que nos permite hacer pruebas hasta estar satisfechos con el resultado.

El código lee el modelo seleccionado, y a todos los modelos existentes en esa placa le sumara el ángulo escogido al ángulo guardado en el fichero de Correspondencias.txt (tercera columna) más el ángulo definido en el archivo .pca para determinar su orientación final.

7.6. Rotación Permanente

Una vez que estemos seguros de cuál es el ángulo necesario para corregir ese desajuste previo, podremos aplicar el cambio de una manera permanente. El código busca el modelo seleccionado en el fichero Correspondencias.txt (segunda columna) y le suma al ángulo inicial (tercera columna) el valor marcado en la casilla de rotación. Ese ángulo resultante se divide entre 360 y nos quedamos con el resto, de forma que los valores finales posibles sólo son 0, 90, 180 ó 270. Esto nos evita que aparezcan números negativos, o mayores de 360, que puedan interferir con el correcto funcionamiento de nuestra aplicación.

7.7. Guardar Archivo

En este bloque crearemos un archivo con formato .kicad_pcb que contenga todos los datos necesarios para poder hacer una réplica exacta de la placa creada en MicroSim PSpice, que pueda ser leída por KiCad y que contenga toda la información para su representación tridimensional, que es realmente el objetivo final de esta aplicación.

Lo primero es la cabecera del archivo. Las primeras líneas contienen información básica sobre el tamaño de la plantilla, las definiciones de las capas y sus estilos, las redes y demás configuraciones iniciales. Todo esto lo tenemos escrito en un fichero llamado plantilla.txt, que el código copia en la ruta de guardado y con el nombre escogido por el usuario. Este será el archivo .kicad_pcb final e iremos escribiendo todo a continuación del texto copiado de la plantilla.

Utilizando la estructura ya estudiada en el capítulo 5, iremos escribiendo todos los elementos, sustituyendo las variables por su valor numérico final, con los cambios y transformaciones ya aplicados. Este método funciona con los bordes de la placa, líneas, arcos, textos, taladros, pistas y vías.

Para los componentes, la metodología no es tan sencilla. Vamos a ayudarnos de la librería de modelos de KiCad, o de modelos descargados de internet, o incluso creados por el usuario (siguiendo el mismo formato que los demás modelos, por supuesto).

Para cada componente que exista en la placa, el código busca el modelo seleccionado por el usuario (por defecto, el primer modelo de la lista) en la carpeta “_Libs”, abre ese archivo y copia su contenido. Después añadimos una línea adicional tras la primera línea, donde insertamos los datos de su

posición y orientación. Buscamos la línea que contenga la palabra REF**, y sustituimos esa y la siguiente por todos los datos del nombre de referencia correspondientes. También modificamos todas las líneas que incluyen información de los pads, con la que hemos leído y modificado del fichero .pca, prestando especial cuidado con usar el formato correspondiente según el tipo y forma de los pads, asegurándonos también que todos los giros que se aplican al componente (orientación en la placa, rotación inicial de la librería y rotación aplicada por el usuario) se aplique también de manera correcta a la orientación de los pads.

Cuando ya están todos los elementos escritos sólo nos queda escribir el cierre del archivo para que mantenga el formato correcto.

7.8. Añadir Componente

Esta funcionalidad se encarga de añadir los nombres introducidos por el usuario en el fichero de Correspondencias.txt mencionado anteriormente. Este fichero puede ser modificado de manera manual abriéndolo con cualquier editor de texto, pero hay que tener cuidado de mantener el formato intacto.

Si no encontramos el componente en las librerías básicas de KiCad, tendremos que buscarlo online. Buscando el nombre del modelo, por lo habitual, aparecerán varias páginas web de proveedores donde nos permiten descargar el modelo 2D (el footprint) y el modelo 3D. Necesitaremos ambos.
Recordemos que los footprint tienen que guardarse en la carpeta "_Libs" de nuestra aplicación, mientras que los modelos 3D se guardan en su carpeta correspondiente dentro de la ruta:

C:\Program Files\KiCad\share\kicad\modules\packages3d

Como ejemplo, añado un enlace a un proveedor que nos permite descargarnos los modelos de un conector de 10 pines:

IPT1-120-09-S-D footprint & symbol by Samtec | SnapEDA

Otros problemas que pueden surgir, es que el modelo aparezca en la librería de KiCad, pero no exista modelo tridimensional, porque aún no están completos o implementados. Añado un enlace a una página donde se encuentra la mayoría de dichos componentes. También podemos descargarlos de proveedores, como hemos explicado.

kicad-library/modules/packages3d at master · KiCad/kicad-library · GitHub

Otra opción es acudir a bibliotecas de modelos 3D, donde podremos encontrar diversos modelos, por ejemplo la siguiente:

Popular models | 3D CAD Model Collection | GrabCAD Community Library

Por último destacar que, con todos los conocimientos de cómo se definen las líneas, arcos y pads, el usuario podría modificar modelos ya existentes e incluso crearlo de cero. Mientras tome de referencia la estructura de los otros modelos, nuestro programa será capaz de interpretarlos y añadirlos a la placa de forma satisfactoria.

8. Posibles mejoras para futuras versiones

Al haberse diseñado y programado la aplicación de una manera tan lineal y organizada, es muy sencillo implementar más funcionalidades que complementen la estructura base creada.

Una posible mejora sería añadir una ventana donde mostrar una imagen del modelo actualmente seleccionado, para que el usuario tuviera una guía más visual que facilitase su elección. Dicha ventana podría integrarse justo debajo del menú de rotación y cargaría imágenes que deberían descargarse y guardarse de la misma forma que los modelos de la biblioteca.

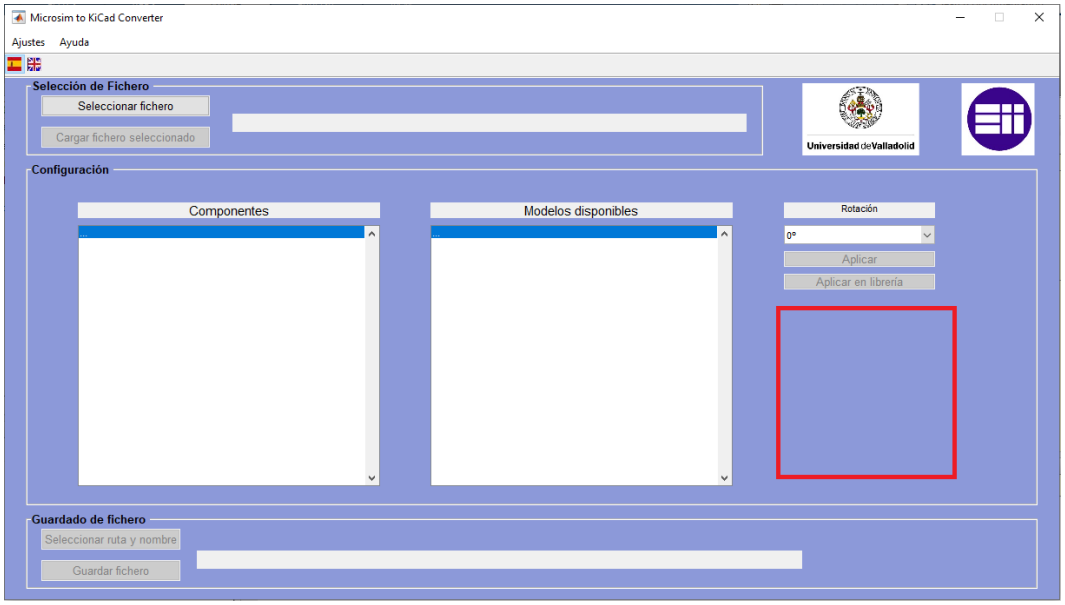

Ilustración 97: Posible localización de pre visualización de modelos

Otra funcionalidad que podría ser de ayuda al usuario, es añadir una opción dentro de la barra de herramientas que abriese en una ventana de Windows la carpeta donde se encuentra la librería interna de la aplicación, para ahorrar al usuario el tiempo y esfuerzo de tener que buscarla manualmente.

9. Resultados y conclusiones

Una vez creada la aplicación, se realizaron pruebas con distintas placas para asegurarnos de su correcto funcionamiento. Pudimos observar que, con tan solo añadir los componentes que no estuviesen en nuestra librería, el programa generaba la placa tridimensional con total precisión. Con el uso, esta librería crecerá de manera gradual, aumentando así la probabilidad de que la herramienta genere el fichero final con menor esfuerzo por parte del usuario. A continuación mostramos varios ejemplos de las placas generadas con MicroSim 8.0 y los resultados finales obtenidos por ambas capas para apreciar todos los detalles.

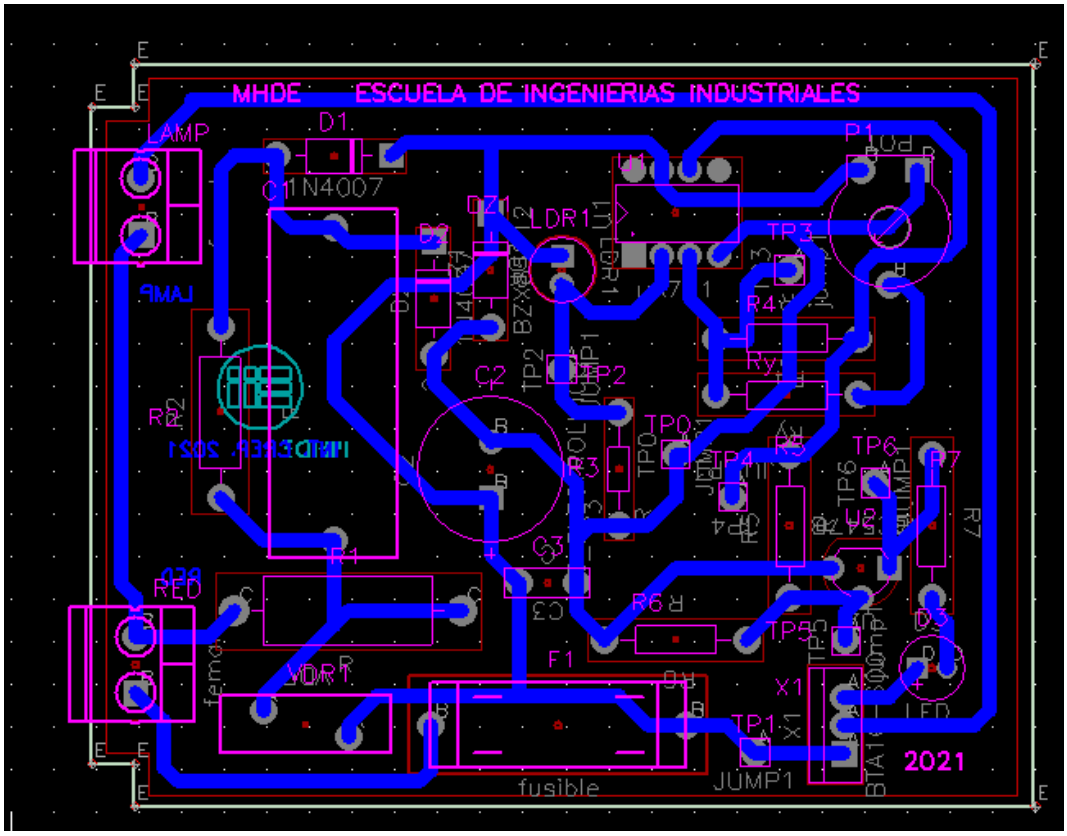

Ilustración 98: IC_TFG_1.pca en MicroSim

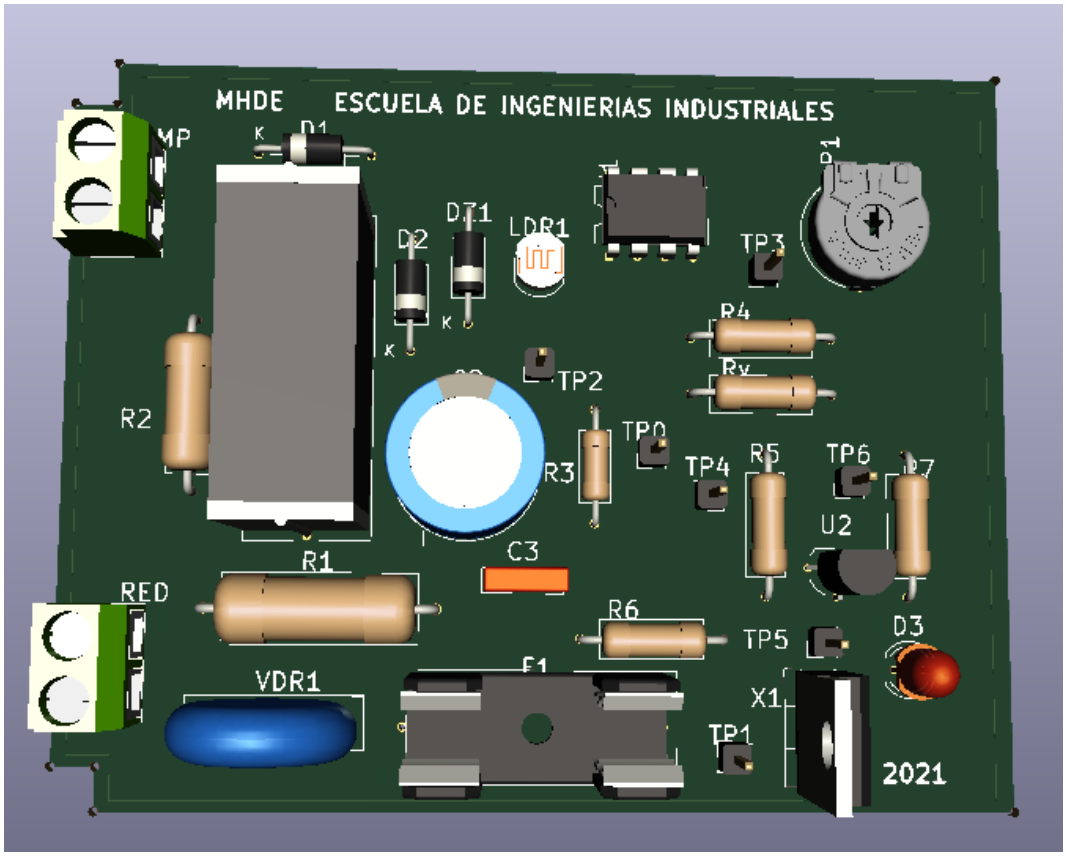

Ilustración 99: Visualización cara superior IC_TFG_1.kicad_pcb en KiCad

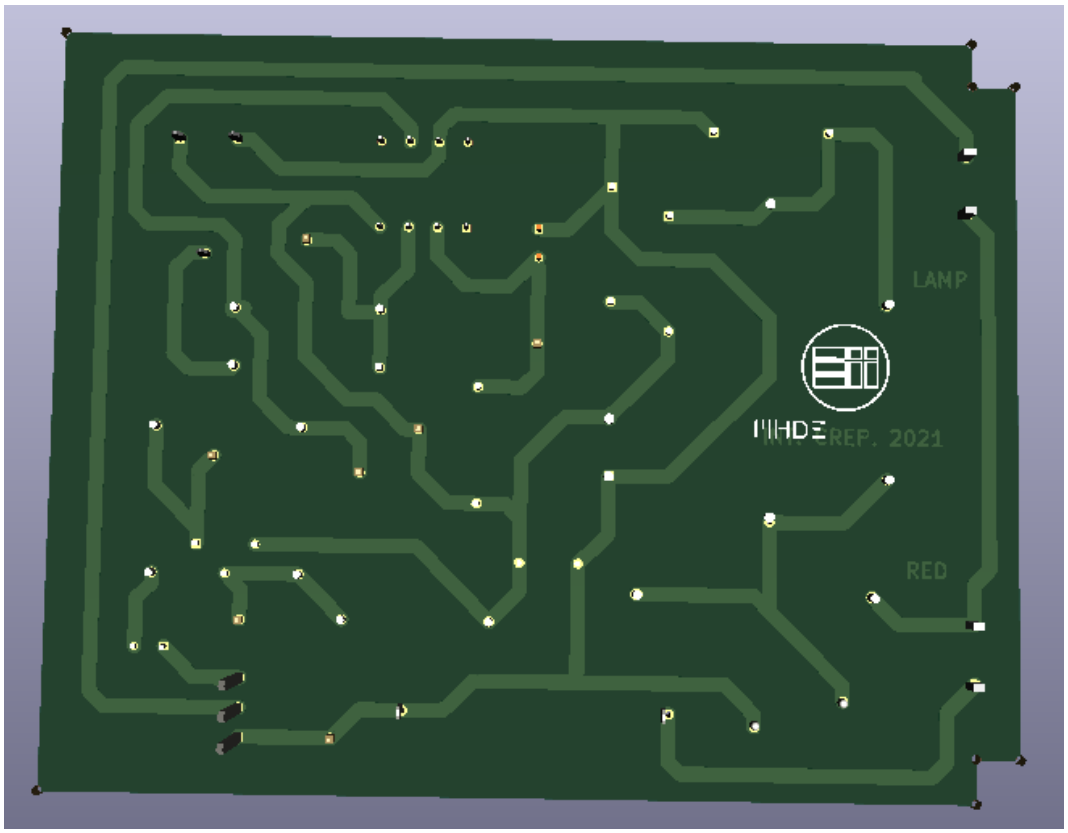

Ilustración 100: Visualización cara inferior IC_TFG_1.kicad_pcb en KiCad

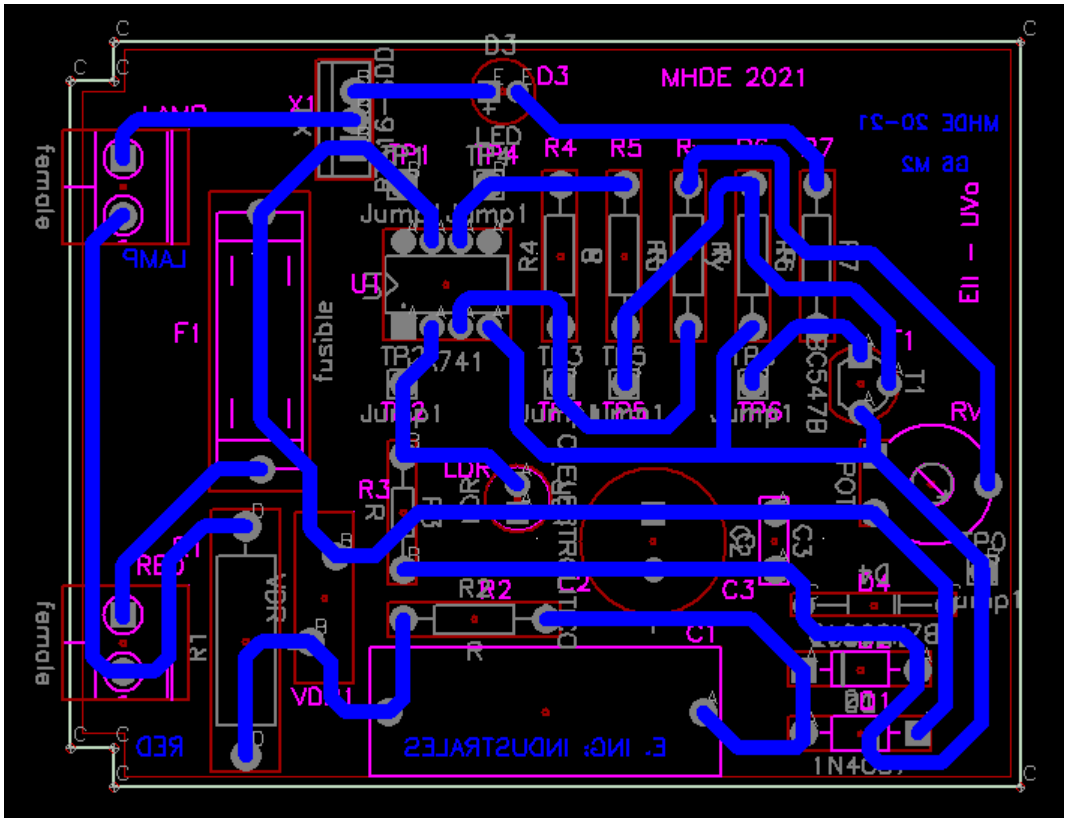

Ilustración 101: IC_2021.pca en MicroSim

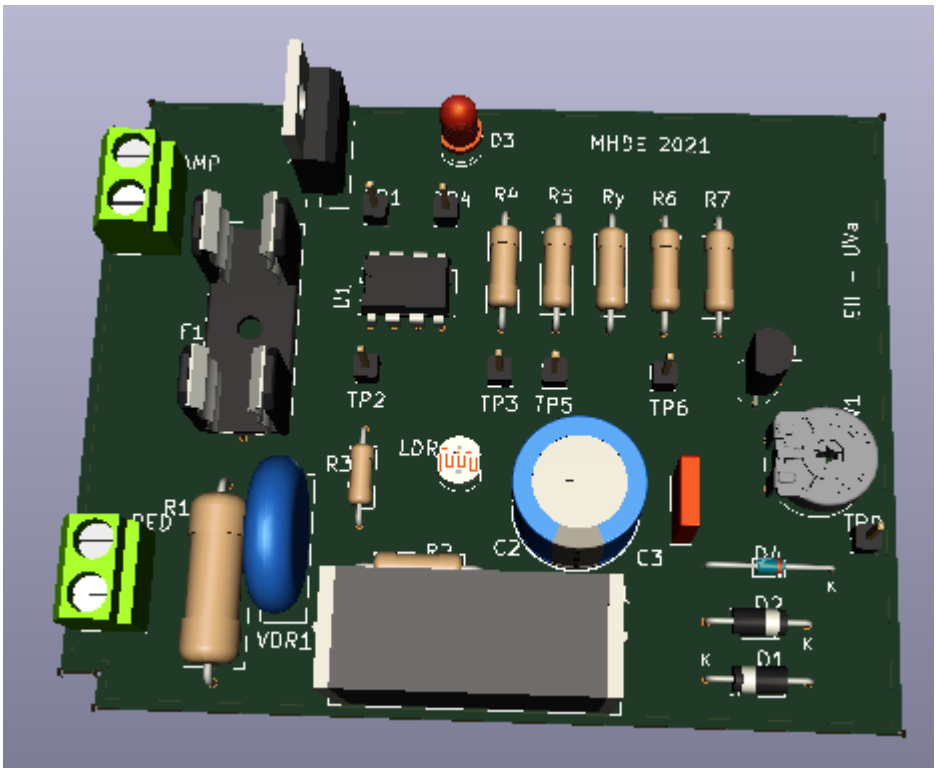

Ilustración 102: Visualización cara superior IC_2021 en KiCad

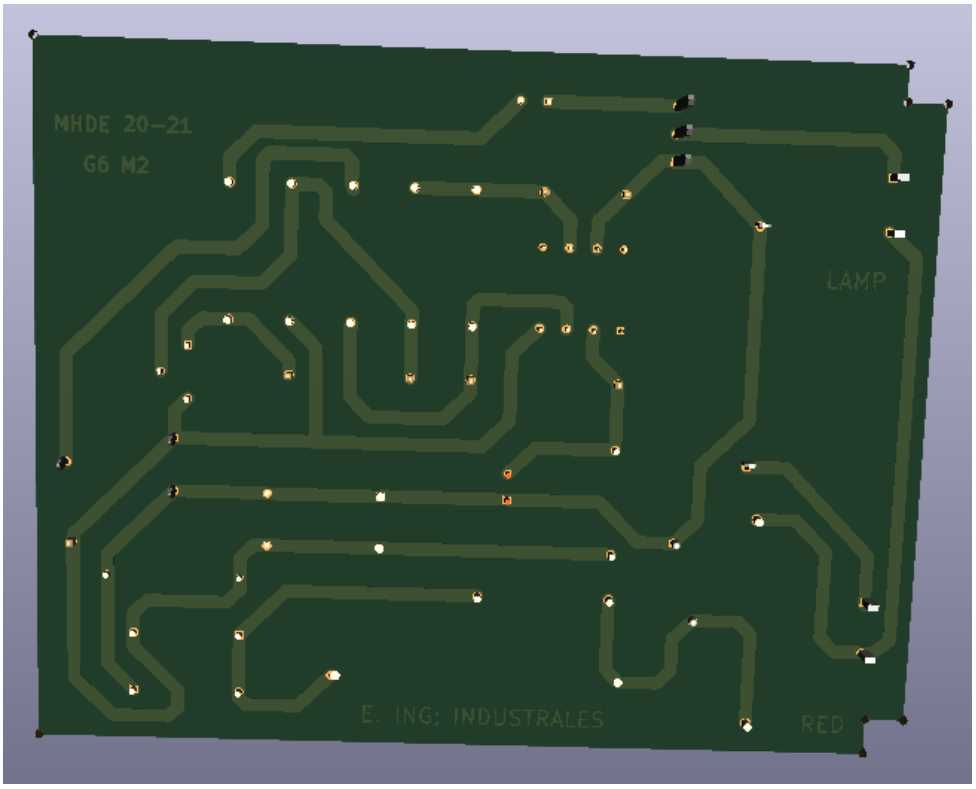

Ilustración 103: Visualización cara inferior IC_2021 en KiCad

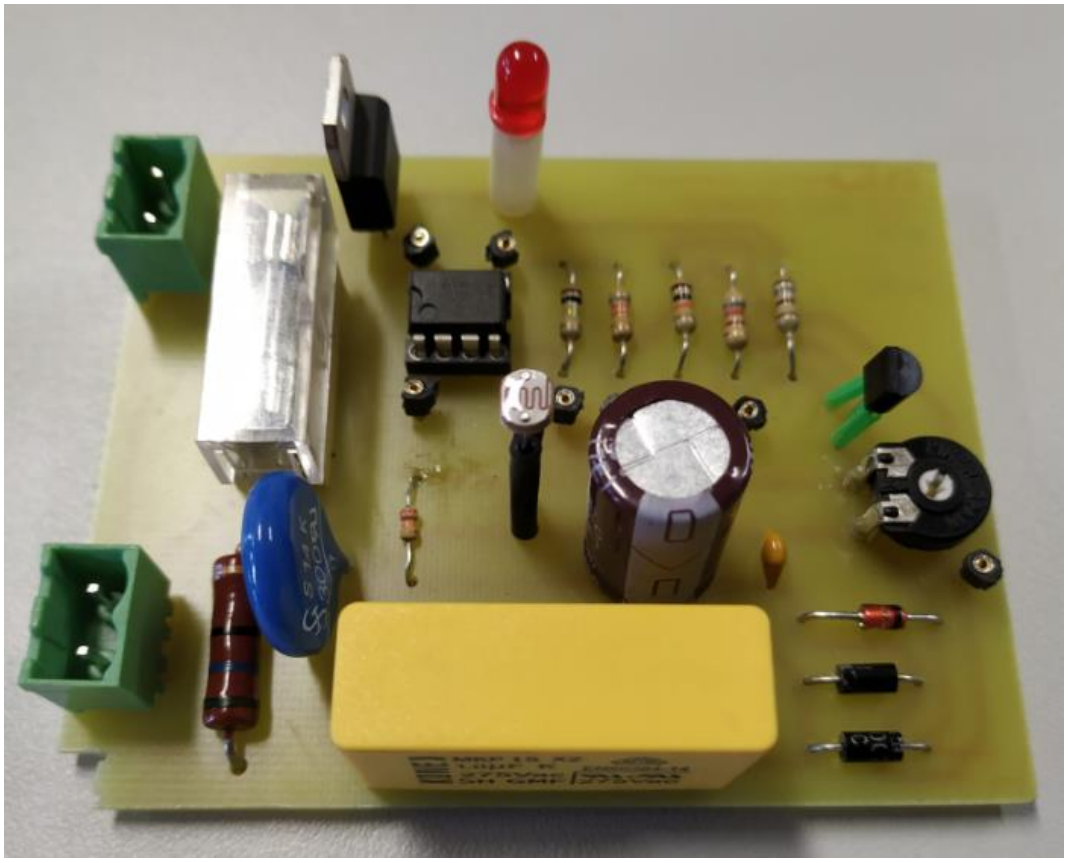

Ilustración 104: Prototipo físico final de la placa IC_2021

Consideramos que los objetivos han sido alcanzados, puesto que la aplicación desarrollada cumple con su función de obtener la vista tridimensional de la placa de circuito impreso original manteniendo todos sus elementos y características. Además, la interacción creada entre la herramienta y la librería de modelos creada soluciona de manera eficaz el problema de tener que contar con una base de componentes limitados, sin tener que acceder al código base para ampliar el número de componentes.

Las funcionalidades adicionales, como el reajuste de orientación de componentes, añadir nuevos modelos a la librería desde la aplicación o la variedad de idiomas, permite al usuario un uso más completo, sencillo e intuitivo de la herramienta.

Por otra parte, el estudio en profundidad realizado sobre los archivos de formatos “.pca” y “.kicad_pcb” me han permitido llegar a entender, saber interpretar e incluso modificar dichos archivos con soltura sin tener que acceder a sus interfaces correspondientes.

10. Bibliografía y referencias

- DPTO. ELECTRÓNICA Y ELECTROMAGNETISMO. Introducción a MicroSim PSpice®: Descripción del entorno, Universidad de Cádiz
  Design_Eval_8.fm (mfbarcell.es)

- MicroSim Schematics User Guide
  Schematics User's Guide (mfbarcell.es)

- DPTO. ELECTRÓNICA Y ELECTROMAGNETISMO. Simulación de circuitos con Pspice, Universidad de Sevilla
  01Intro_Microsim.PDF (csic.es)

- Electronic Design Methodology In Lab Experiments
  Universidad de Sevilla. Escuela Universitaria Politécnica
  002.p65 (csic.es)

- FABIÁN ACQUATICCI. Introducción al Diseño de Circuitos Electrónicos, Universidad de Buenos Aires
  intro1_v13 (uba.ar)

- Miguel Pareja Aparicio. Diseño y desarrollo de circuitos impresos con KiCad, 2010

- Creating a 3D model for Kicad
  Creating a 3D model for Kicad - Workshopshed

11. Anejos

En los anejos adjuntos a esta memoria podemos encontrar:

1. El código fuente de la aplicación. Esto incluye la función principal y todas las funciones secundarias que utiliza, así como los diversos archivos necesarios para su funcionamiento (como guía de usuario, o logos utilizados en la interfaz).

2. El archivo “Instalador.exe”. Se trata de un ejecutable que permite la instalación completa de la aplicación para cualquier usuario.

3. El archivo “Manual_Usuario”, donde se describe paso a paso todo el funcionamiento de la aplicación de manera clara y concisa.

4. El archivo “Lista_Modelos.xlsx”, un Excel que contiene en una tabla todos los componentes comprendidos actualmente en la librería interna del programa.