Instituto Tecnológico de Costa Rica

Carrera de Ingeniería Mecatrónica

Desarrollo de un evaluador previo a la aplicación de pruebas automatizadas, diseño de circuitos impresos y accesorio mecánico para el montaje de una interfaz formal.

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura

Daniel de Jesús Aguirre Sosa

2014067945

Cartago, 19 de abril del 2021

Declaratoria de Autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema y asesoramiento técnico de miembros de la empresa Boston Scientific y S.R.L. y el asesoramiento del profesor Rodolfo Piedra del Tecnológico de Costa Rica.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Cartago, 19 de abril 2021

Daniel de Jesús Aguirre Sosa
Céd: 6-04330173

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica. Por lo tanto, este trabajo cuenta con el aval para ser defendido ante el jurado evaluador, como requisito para optar por el título de Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: Daniel de Jesús Aguirre Sosa

Proyecto: Desarrollo de un evaluador previo a la aplicación de pruebas automatizadas, diseño de circuitos impresos y accesorio mecánico para el montaje de una interfaz formal.

RODOLFO JOSE PIEDRA CAMACHO (FIRMA)

Digitally signed by RODOLFO JOSE PIEDRA CAMACHO (FIRMA)
Date: 2021.04.14 00:42:08 -06'00'

Ing. Rodolfo José Piedra Camacho

Asesor

Cartago,19 de abril 2021.

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: Daniel de Jesús Aguirre Sosa

Proyecto: Desarrollo de un evaluador previo a la aplicación de pruebas automatizadas, diseño de circuitos impresos y accesorio mecánico para el montaje de una interfaz formal.

Miembros del jurado evaluador

JAIME ALONSO MORA MELENDEZ (FIRMA)
Digitally signed by JAIME ALONSO MORA MELENDEZ (FIRMA)
Date: 2021.04.20 15:38:37 -06'00'

MSc. -Ing. Jaime Mora Meléndez

Jurado

JUAN CARLOS BRENES TORRES (FIRMA)
Firmado digitalmente por JUAN CARLOS BRENES TORRES (FIRMA)
Fecha: 2021.04.20 12:44:44 -06'00'

MSc. -Ing. Juan Carlos Brenes Torres

Jurado

ANA MARIA MURILLO MORGAN (FIRMA)
Firmado digitalmente por ANA MARIA MURILLO MORGAN (FIRMA)
Fecha: 2021.04.19 16:29:24 -06'00'

Ing. Ana María Murillo Morgan

Jurado

Los miembros de este jurado dan fe de que el presente proyecto final de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

Cartago, 19 de abril 2021

Resumen

En la industria medica los programas que asisten a los profesionales en intervenciones cardiacas deben ser verificados de manera exhaustiva, el equipo de Rhythmia se especializa en realizar pruebas de software y hardware al sistema de mapeo intracardiaco desarrollado por la empresa Boston Scientific, en la actualidad se ha impulsado la automatización de las pruebas dependientes de hardware pero para poder lograrlo se requiere que el entorno de pruebas tenga la estabilidad y características necesarias para poder ser aprobado como entorno fiable.

En este documento de detalla el proceso de diseño de 3 partes primordiales en el desarrollo de la infraestructura del entorno de ejecución de las pruebas dependientes de hardware las cuales son un código que evalúa el estado y conexión de los 4 principales componentes del entorno de pruebas, placas de circuitos impresos que permiten simular la conexión y desconexión automática de los dispositivos del sistema de mapeo que se incluyen dentro del plan de pruebas del entorno y un accesorio de resguardo para propiciar orden, ergonomía y protección.

Los diseños se validaron aplicando pruebas de carácter unitario y funcional a nivel de código, simulaciones analítico-enfocadas, comprobaciones matemáticas y evaluación de estados críticos de las distintas partes para asegurar el cumplimiento de los requerimientos de diseño.

Palabras clave: Software de evaluación, circuito de conmutación, accesorio de resguardo, entorno de pruebas.

Summary

In the medical industry, the programs that assist professionals in cardiac interventions must be verified exhaustively, the Rhythmia team specializes in performing software and hardware tests to the intracardiac mapping system developed by Boston Scientific. Recently, the automation of hardware tests has been promoted, but to achieve this, the test environment must have the necessary stability and characteristics to be approved as a reliable environment.

This document details the design process of 3 essential parts in the development of the infrastructure of the execution environment of the hardware-dependent tests that are a code that evaluates the status and connection of the 4 main components of the test environment, PCBs that allow to simulate the automatic connection and disconnection of the mapping system devices that are included in the official test plan and a mechanical accessory to ensure order, ergonomics, and protection.

The designs were validated by applying unitary and functional tests at the code level, analytical-focused simulations, mathematical checks, and evaluation of critical states of the different parts to ensure compliance with the design requirements.

Keywords: Evaluation software, switching circuit, backup accessory, test environment, protection accessory.

Dedicado.

A Mercedes Rodríguez y Carla Sosa
que siempre me han amado y apoyado incondicionalmente.
Sin ustedes no estaría hoy acá.

Agradecimiento

A mis padres Carla Sosa y Ronny Cruz que estuvieron siempre apoyándome todo el camino y brindándome fortaleza para seguir adelante en los momentos más difíciles, a mi hermano Vincent Cruz que siempre me alegro los días cuando más lo necesitaba.

A mis abuelos Victoria Rodríguez y Ronald Silva que nunca me dieron la espalda y siempre me tendieron la mano para ayudarme en los momentos que más lo necesitaba.

A mis bisabuelos Carlos Campos y Mercedes Rodríguez que son una parte de mi corazón, me amaron y apoyaron incondicionalmente en todo mi camino educativo.

A mis amigos Felipe Rodríguez, Octavio Sánchez, Adriana Rojas, Francisco Monge, Francini Solorzano, Jesús Castro y Mauro Zumbado les agradezco por su apoyo, ustedes alegraron e iluminaron todo mi camino en la universidad.

Al profesor Rodolfo Piedra Camacho por su excelente apoyo y guía en el desarrollo de este trabajo.

A mis jefes Sharon Tang y Priyank Soni por darme la oportunidad de desarrollar este proyecto y brindarme una experiencia laboral invaluable.

Índice General

Índice de Tablas

Índice de figuras

Lista de abreviaturas

Abreviaturas

EU	Estados Unidos
R&D	Research and Development
SiS	Signal Station
PCB	Placa de circuito impreso
EP	Electrofisiología
ECG	Electrocardiograma
NI	National Instruments
DPDT	Double Pole Double Throw
2PST	Two Pole Single Throw
RTMD	Rhythmia Mapping
API	Interfaz de aplicación
SP	Stable Point

1. Introducción

1.1 Entorno del proyecto

Boston Scientific es una empresa estadounidense fabricante de dispositivos médicos especializada en áreas como radiología, cardiología, electrofisiología y neuro modulación, entre otras. La primera sede de Boston Scientific en Costa Rica abrió sus puertas en el 2004 en la zona franca de Heredia (Global Park), actualmente se cuenta con más de 1453 colaboradores.

El área en la cual el proyecto se desarrolla es electrofisiología la cual busca entender la manera en que las señales eléctricas se mueven a través del corazón, se utiliza en personas cuyos cuerpos producen señales eléctricas inefectivas o caóticas que causan que el corazón lata incorrectamente. Boston Scientific desarrolla múltiples herramientas utilizadas en el diagnóstico y tratamiento, unas de las principales herramientas desarrolladas son los catéteres de ablación, diagnóstico y mapeo. También cuenta con el desarrollo de múltiples equipos, el principal equipamiento para mapeo y navegación es el “RHYTHMIA HDx Mapping System” el cual se desarrolla y se prueba en Cambridge, EU.

Desde el 2018 se creó una división costarricense del equipo “Rhythmia” en las oficinas de Heredia en el departamento de R&D (Research and Development), esta división cuenta con 5 ingenieros que trabajan en coordinación con el equipo de “Rhythmia” ubicado en Cambridge, Estados Unidos.

Ambos equipos son encargados del testeo formal del programa “Rhythmia Mapping System” así como de todos los dispositivos utilizados para el funcionamiento del programa. Las actividades del equipo de testeo de Costa Rica se dividen en 2 tipos el testeo manual y el testeo automático.

Las actividades de prueba manual involucran aquellas que requieren interacción humana como conectar o desconectar dispositivos, encender y apagar unidades o estaciones de señalización y que también requieren verificación a través de líneas de comando o respuestas visuales que actualmente están disponibles en entornos no automatizados. Algunas de estas actividades son la ejecución de

pruebas informales (simulacros), verificación y registro de errores, verificación de la sustentabilidad de las herramientas utilizadas en la operación y prueba del software, y la verificación del diseño final del software que será lanzado al mercado.

Las actividades de automatización implican todas aquellas que transforman en automáticas, pruebas que se realizaban manualmente con el fin de reducir tiempos de ejecución de los protocolos de verificación los cuales son un conjunto de pruebas que se utilizan para asegurar el bienestar de múltiples componentes.

Algunas de estas actividades incluyen análisis estático de código, análisis de fugas de memoria, análisis de rendimiento, análisis de cobertura de código, ejecuciones de automatización de pruebas nocturnas y desarrollo de casos de prueba para las independientes y dependientes del hardware, así como todo el mantenimiento de las herramientas, scripts y prueba ya desarrollada

El testeo manual y automatizado puede ser dividido en independientes “hardware” y dependientes “hardware”. La principal diferencia entre estas 2 categorías recae en el requerimiento de equipo para la ejecución de las pruebas. Si estas requieren de todos los equipos de Rhythmia como el SiS, generador de ablaciones, catéteres, sistema de grabación y estaciones de trabajo son consideradas dependientes de “hardware”, por otro lado, si las pruebas únicamente requieren de la estación de trabajo y flujo de datos (previamente adquiridos) es considerado independiente de “hardware”.

Actualmente el equipo de automatización ha automatizado 432 pruebas de 1111 pruebas independientes de “hardware” y 17 pruebas de 453 dependientes de “hardware”. Esta diferencia se debe a varias razones, una de ellas es la infraestructura necesaria para automatizar casos dependientes de “hardware”.

1.2 Definición del problema

1.2.1 Planteamiento del problema u objeto de estudio.

Hoy en día en el Departamento de Rythmia, que pertenece a Boston Scientific, las pruebas formales entre estaciones de trabajo y algunos dispositivos médicos como catéteres y consola se están ejecutando manualmente. El sistema actual que usa código heredado (paquete de Python desarrollado por el equipo) se puede identificar como una prueba de concepto que tiene ciertas desventajas como

interferencias eléctricas, una inconsistencia en las conexiones y lectura de datos para los dispositivos conectados, resultados inconclusos y un alcance limitado en los dispositivos que se pueden conectar.

El equipo de Rhythmia desea automatizar estas pruebas en un entorno fiable, por lo cual se requiere realizar una infraestructura formal mejorando al sistema actual, para esto se necesita el diseño de circuitos impresos para las conexiones esenciales, diseño de un accesorio que contenga las PCB desarrolladas para armar una interfaz formal y un evaluador del sistema para dar trazabilidad a fin de identificar si un problema (estado del sistema fuera de los parámetros operativos normales) es un caso conocido o desconocido, verificando el estado de las conexiones físicas y de los dispositivos antes y después de ejecutar cada prueba, en base al estado por defecto que se deba tener.

1.2.2 Justificación

Previo al lanzamiento de nuevas versiones del software se realizan en promedio de 2 a 3 sub-fases de pruebas y en cada sub-fase se deben realizar un máximo de 453 pruebas dependientes de “hardware” manualmente, lo cual representa una tercera parte de las pruebas realizadas. Cada prueba consume tiempo de trabajo en ambos equipos, que podría ser reducido en caso de desarrollar un completo procedimiento de automatización para este tipo de pruebas.

Dentro del desarrollo de este procedimiento, la ausencia de un sistema de verificación previo a las ejecuciones de las pruebas impide que se conozca el estado de los equipos y sistemas, resultando en fallas no rastreables que ocasionan falsos negativos. Esto puede denominarse un ambiente de testeo no formal, sin ningún tipo de estandarización que provee información de baja confiabilidad retrasando los procesos de pruebas y el tiempo de lanzamiento del software, repercutiendo económicamente a la empresa.

1.2.3 Síntesis del problema

La ausencia de una infraestructura (hardware y software) para el ambiente de pruebas dependientes de hardware impide el desarrollo de un sistema formal para la ejecución de estas, produciendo inestabilidad en el proceso de testeo y resultados inconclusos.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar parte de la infraestructura necesaria para la formalización del sistema de ejecución de pruebas dependientes de hardware.

1.3.2 Objetivos específicos

1. Proponer una estrategia de evaluación para los componentes del sistema de automatización de pruebas dependientes de hardware

2. Diseñar placas de circuitos impresos para la conmutación de señales y accesorio mecánico de resguardo de acuerdo con los requerimientos para la formalización de las conexiones.

3. Validar los diseños de hardware y software por medio de la simulación analítico-enfocada de las partes desarrolladas para la formalización del sistema de acuerdo con los requerimientos de funcionalidad.

1.4 Estructura del documento

En el presente trabajo se solucionó la carencia del diseño de las partes principales que requería el entorno automatizado de pruebas dependientes de hardware para ser formalizado mediante la aplicación de técnicas ingenieriles mecánicas, electrónicas y software. Este se encuentra estructurado de la siguiente manera:

- Capítulo 2. Sección que presenta los conceptos más importantes relacionados a la teoría y entorno del proyecto para facilitar la comprensión del lector.
- Capítulo 3. Una explicación de la metodología de diseño con la que el proyecto se plantea y se desarrolla.
- Capítulo 4. Los requerimientos obtenidos del estudio de campo y las necesidades de la empresa que son utilizados como base para las decisiones de diseño.
- Capítulo 5. Las propuesta y selección de los distintos conceptos.

- Capítulo 6. Integración de los conceptos seleccionados para el desarrollo del componente.
- Capítulo 7. Desarrollo de pruebas funcionales y simulaciones para validar el cumplimiento de los requerimientos de cada elemento.
- Capítulo 8. Conclusiones cualitativas y cuantitativas basadas en los resultados obtenidos.
- Capítulo 9. Recomendaciones en caso de que se desee tomar como base los diseños y resultados obtenidos en este proyecto para darle seguimiento y mejora en un proyecto futuro.

2 Marco Teórico

2.1 Generalidades del sistema de mapeo de Rhythmia

El sistema de mapeo “RHYTHMIA HDx” es un sistema de mapeo 3D y navegación usado en procedimientos electrofisiológicos (EP). Este sistema emplea 2 mecanismos principales para realizar el mapeo y la navegación:

- Mapeo continúo basado en las señales cardiacas del paciente provenientes de catéteres intracardiacos y electrodos ECG (Electrocardiograma) superficiales [1].
- Localización continua de catéteres rastreados magnéticamente y por impedancia [1].

Estos catéteres se dividen en 3 tipos los de mapeo, los de referencia y los de ablación, estos catéteres pueden incluir uno o ambos sistemas de rastreo. Los sistemas de rastreo utilizan un sensor magnético ubicado en las puntas de los catéteres para definir su posición respecto al campo magnético y las señales eléctricas intracardiacas que se propagan entre electrodos para definir los valores de impedancia que se utilizan construir la posición del catéter dentro del corazón.

2.1.1 Componentes Principales del sistema

El sistema de mapeo de Rhythmia requiere de distintos componentes para la recolección de los datos, el procesamiento de información, la generación de señales, la interfaz del usuario, entre otros. El conjunto de estos componentes es probado por un equipo especializado para asegurar su funcionamiento previo a la distribución, a continuación, se describirán detalladamente cada uno de los componentes principales del sistema.

En la figura 2.1 podemos observar una representación general de las partes del sistema, excluyendo componentes y accesorios relacionados a la estación de señales como catéteres, parches para electrocardiogramas y cajas de conexión.

Figura 2.1. Componentes principales de sistema de mapeo. Fuente: Propia

2.1.1.1 Estación de señales

La estación de señales acepta las señales provenientes de los catéteres intracardiacos y los electrodos ECG. Esta amplifica, digitaliza y transfiere las señales a la estación de trabajo en tiempo real para su procesamiento y despliegue. También soporta la localización de catéteres y estimulación para diagnóstico [1]. Como se observa en la figura 2.2 en la parte frontal encontramos los puertos de entradas y salidas de la estación de señales, los puertos del 1 al 5 son las entradas y del 6 al 9 las salidas. Los puertos 1, 2 y 3 son para la entrada de señales intracardiacas, es importarte aclarar que los catéteres de mapeo únicamente se pueden conectar al puerto 1. [11]

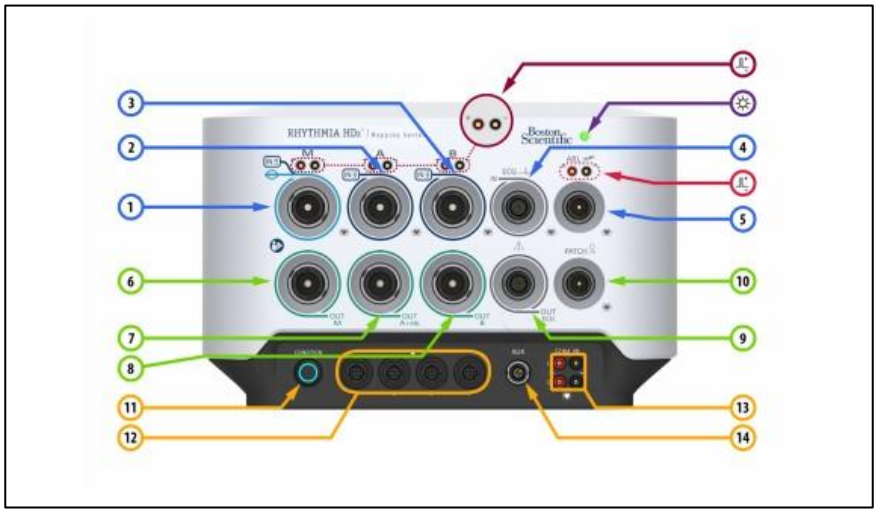


Figura 2.2. Puertos frontales de la estación de señales. [11]

2.1.1.2 Software del sistema

Este software se ejecuta en la estación de trabajo, se encarga de procesar los datos recibidos de la estación de señales y provee una interfaz de usuario. Además, realiza funciones primordiales como representación de las señales ECG e intracardiacas, localización y rastreo de catéteres, mapeo y visualización 3D y enrutamiento de las señales de estimulación para diagnóstico [1].

2.1.1.3 Estación de trabajo

La estación de trabajo unifica los componentes de hardware (computador, monitor, teclado, ratón y cables de alimentación) y el software del sistema. Adicionalmente también almacena, recupera y exporta datos.[1]

2.1.1.4 Generador de ablación.

Este generador permite los procedimientos de ablación con y sin irrigación abierta. La ablación es un procedimiento que se emplea en el tratamiento de arritmias cardiacas, las ablaciones por radiofrecuencia se originan por el calentamiento resistivo en zonas adyacentes a la punta del catéter y por la conducción pasiva del calor hacia zonas más profundas, para temperaturas mayores a 100°C en la punta del catéter de debe utilizar la irrigación de un líquido salino para evitar coágulos.[5] Este sistema combina una interfaz de usuario inteligente de diagnóstico integral en tiempo real para maximizar el éxito de los procedimientos. [2]

2.1.2 Dispositivos de entrada y salida de la estación de señales

Todos los componentes y accesorios que se conectan a los puertos frontales de la estación de señales para la entrada y salida de información.

2.1.2.1 Catéter de diagnóstico y mapeo.

Consiste en un conjunto de cables aislados unidos a electrodos. En el extremo próximo más cercano al final del catéter cada cable esta unido a un enchufe que se conecta a los puertos M, A o B de la estación de señales mostrados en la figura 2.2 [3]. Un catéter común tiene cuatro electrodos, 2 de estos son utilizados para la recolección de señales eléctricas y los otros 2 son utilizados para estimulación eléctrica, también existen catéteres, comúnmente usados para mapeo, con más electrodos para la recolección de más señales eléctricas de manera

simultánea, el más utilizado es el “IntellaMap Orion mapping catheter”. El tamaño de cada electrodo varia de 2mm a 8mm con un espacio entre ellos de 2mm a 10mm [3].

2.1.2.2 Catéter de ablación.

Son aquellos catéteres especializados con los que se pueden realizar ablaciones del tejido por medio de diferentes formas de energía (radiofrecuencia, frio, laser, etc.), estos catéteres tienen como objetivo producir una lesión controlada del tejido para eliminar la formación de distintos padecimientos cardiacos como las arritmias [4]. Los catéteres de ablación más comunes son los de radiofrecuencia, estos utilizan un generador externo que energía a la punta del catéter y este la convierte en energía térmica en la interfase electrodo tejido, la extensión y profundidad depende del catéter utilizado.

Dentro del equipo de Rhythmia para pruebas podemos encontrar los que contienen tecnología “StablePoint” y los que contienen tecnología “Force”, ambos tipos de conectan al mismo puerto, pero utilizan cajas de conexión distintas.

2.1.2.3 Caja de Conexión (Breakout Box).

La Breakout box y sus cables de conexión proveen una interfaz física entre la estación de señales y los catéteres de diagnóstico. Tiene 64 pines de entrada y un puerto con múltiples pines de salida como se observa en la figura 2.3, esta salida se puede conectar a los puertos M, A y B de la estación de señales [11]. La comunicación de este dispositivo y la estación de señales se realiza utilizando el protocolo I2C.

Figura 2.3. Breakout Box. [11]

2.1.2.4 Caja de conexión de ablación.

Esta caja de conexión es requerida para conectar los catéteres de ablación al sistema, su propósito es prevenir que la señal de ablación por radiofrecuencia interfiera con la señal de localización del catéter, presenta conexiones al generador de ablación, al catéter de ablación y una única salida que va conectada al puerto de ablación de la estación de señales [11]. Este elemento utiliza un protocolo de comunicación I2C a 100kHz para su conexión con la estación de señales. En la figura 2.4 se puede observar la caja de conexión.

Figura 2.4. Caja de conexión de ablación. [11]

2.1.2.5 Parche trasero de localización

También se conoce como “backpatch”. Se coloca en una posición estable a la mitad del dorso en la sección de la espalda, se utiliza como punto de referencia para la localización de elementos [11]. Este parche se conecta a la estación de señales por medio de un cable al puerto “Patch”, el conector presenta 6 pines por los cuales se transmite información de sensores y los datos de referencia utilizando como protocolo de transmisión I2C [12]. En la figura 2.5 se puede apreciar el cable de conexión usado.

Figura 2.5. Cable de parche trasero. [11]

2.1.2.6 Cables de conexión IC

Estos cables son utilizados en conjunto con el amplificador “ClearSign” y el sistema de grabación externo “CardioLab”, estos 2 sistemas utilizan distintos cables y modalidades de conexión.

- Cable de amplificador ClearSign: Se utiliza el “IC Orion” en el puerto M-OUT cuando se está mapeando con un “Intellamap” Orion Mapping Catéter” y el “IC A/B” cuando se conecta la “Breakout Box” a cualquier de los puertos de entrada. Cada canal de salida va asociado a una señal distinta como se observa en la tabla 2.1, los canales de salida utilizados dependerán de las conexiones realizadas en los puertos de entrada teniendo una capacidad de hasta 64 canales cuando se utilizan todas las conexiones del “Breakout Box”.
- Cable CardioLab: Este cable puede ser conectado a cualquiera de los 3 puertos de salida, dependiendo de la conexión de entrada cada canal de salida está asociado a una señal como se muestra en la tabla 2.1.

Tabla 2.1. Salidas y entradas de cables IC. [11]

IC Orion		IC A/B				Cable Cardiolab					
Puerto de entrada M		Puerto de entrada A+ABL		Puerto de entrada M y B		Puerto de salida M		Puerto de Salida A+ABL		Puerto de Salida B	
Electrodos	Canales de salida	Canales de entrada	Canales de salida	Canales de entrada	Canales de salida	Canales de entrada	Canales de salida	Canales de entrada	Canales de salida	Canales de entrada	Canales de salida
A1-A8	1-8	BB 1-64	1-64	BB 1-64	1-64	Orion A1-H8	1-64	BB 1-32	1-32	BB 1-64	1-64
B1-B8	11-18	ABL 1-8	65-72			BB 1-64		ABL 1-8	33-40		
C1-C8	21-28							BB 33-40	N/A		
D1-D8	31-38							BB 41-64	41-64		
E1-E8	41-48										
F1-F8	51-58										
G1-G8	61-68										
H1-H8	71-78										
Canales de entrada: BB = Breakout Box ABL = Ablation Connection Box											

2.2 Plataforma de módulos para conexiones periférica

A nivel académico, estas plataformas de medida han permitido la caracterización de todo tipo de señales electromagnéticas presentes en sistemas comerciales de múltiples campos (automatización, control, comunicaciones, bioingeniería) [7]. En el diseño también se han conseguido valiosos avances por parte de la academia; se tienen informes de soluciones a problemas de la industria apoyados en estas plataformas, y que fueron desarrolladas en los laboratorios de prestigiosas universidades de todo el mundo [8].

2.2.1 Estándar PCI extensiones para instrumentación (PXI) 101

Es un estándar introducido en 1998, desarrollado por “National Instruments (NI)”, esencialmente para pruebas y medidas automatizadas implementado en una plataforma modular, este estándar resulta muy versátil y abierto a recibir módulos de múltiples fabricantes en las carcasas asociadas en factor de forma de 3 unidades

de Rack (U), o sea, 100 mm x 160 mm o 6U, es decir, 233.35 mm x 160 mm. PXI está basado en compactPCI (cPCI) y sus aspectos mecánicos están establecidos por especificaciones Eurocard (ANSI 310-C, IEC-297, IEEE 1101.1, IEEE 1101.10, y P1101.11) [6].

La especificación del estándar incluye la descripción de los módulos y la descripción de hardware y software; constantemente se está sometiendo a revisiones para hacerlo más compatible con tecnologías ya existentes y con nuevos desarrollos [9].

2.2.2 Módulos

Son las tarjetas que ofrecen el hardware específico para la aplicación a desarrollar. Se acomodan de forma deslizante en las ranuras del chasis dejando, hacia la parte externa, los conectores que corresponden a entradas y salidas, dependiendo de cada módulo. Mecánicamente, los módulos pueden obtenerse con factor de forma de 3U o 6U (U: Unidad de rack), dependiendo de la plataforma a la que se vayan a asociar [6]. Pueden observarse en la figura 2.6.

Figura 2.6. Aspecto físico de modulo PXI. [6]

2.2.3 Chasis

El sub-bastidor o chasis, que aloja los módulos y el plano trasero que conforman el hardware de la plataforma, es parte fundamental del sistema y responde a las especificaciones Eurocard antes mencionadas (ANSI 310-C, IEC-297, IEEE 1101.1, IEEE 1101.10 y P1101.11), el plano trasero es la tarjeta que aloja los conectores donde se acoplarán los módulos al ser deslizados [6]. En la figura 2.7 se observa un típico chasis del fabricante NI.

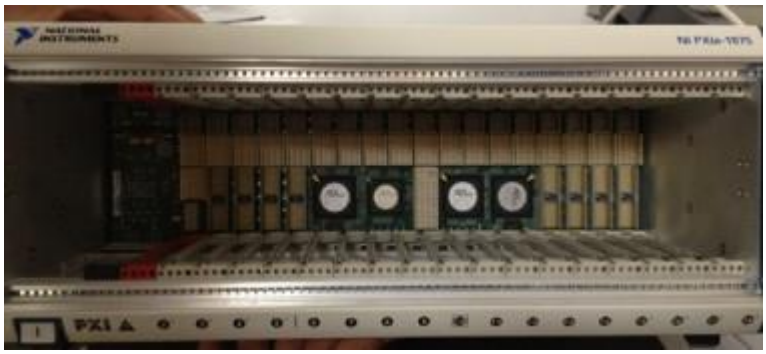


Figura 2.7. Chasis PXI. [6]

2.2.4 Controlador

La mayoría de los chasis PXI contienen una ranura de controlador de sistema, ésta es la más cercana a la orilla izquierda del chasis (ranura 1) [10]. Los tipos de controladores que se pueden seleccionar son:

- Controladores remotos
- Controladores embebidos de alto rendimiento.

2.3 Entorno de programación y código legado

Una de las plataformas más utilizadas comercialmente como interfaz gráfica de testeo es Squish, esta plataforma utiliza propiedades basadas en identificación de objetos, en ella se pueden escribir e interpretar códigos para test en lenguajes como JavaScript, Perl, Python, entre otros [13]. Actualmente el equipo de Rhythmia cuenta con 2 librerías para Python implementadas en esta plataforma que contienen las funciones base para la automatización de pruebas dependientes de hardware, basados en los datos promedio este permite que cada ejecución automática tenga una duración aproximada de 10 a 15 minutos.

2.3.1 Paquete RTMD para Hardware

Este paquete fue desarrollado por el equipo de hardware de Rhythmia y contiene las clases que hacen referencia a los principales puertos de la estación de señales, cada clase contiene distintos objetos para cada elemento que se conecte a dicho puerto, estos presentan una referencia a un archivo de tipo json en el cual se relaciona cada objeto con los relés del módulo PXI que manejan su conexión de esta manera se controla la conexión en el módulo de relés de las principales señales de entrada y salida, esto lo realiza utilizando funcionalidades de la librería de

National Instruments del Chasis PXI, además correlaciona las clases de cada puerto con las implementadas en el Software de Rhythmia “Solo”, ya que la interfaz gráfica de este programa se diseñó orientada a objetos.[14]

Este paquete únicamente se encuentra en las estaciones de testeo de Boston Scientific. Sus clases se encuentran directamente relacionadas con los puertos de la estación de señales mostrados en la figura 2.2 y el generador de ablación mostrado en la figura 2.1. Algunas de las clases implementadas dentro del paquete RTMD son:

- Puerto de entrada M (M_IN).
- Puerto de entrada A (A_IN).
- Puerto de entrada ECG.
- Señal de alimentación de la estación de señales (PIU).
- Puerto de Salida M (M_OUT).
- Puerto de salida A (A_OUT).
- Puerto de salida B (B_OUT).
- Puerto de salida ECG.
- Puerto de entrada Patch.
- Botones de control del generador de ablación (MAESTRO_4000).

2.3.2 Librería NI Switch

Las Bibliotecas de Clase .NET de NI-SWITCH son un software controlador que extiende la funcionalidad del controlador de instrumentos NI-SWITCH. Este software soporta comunicación entre aplicaciones .NET y módulos de multiplexor de conmutación RF PXI, módulos de relé de RF y módulos SwitchBlock compatibles. Además, el software proporciona una interfaz API .NET compatible con clase de instrumentos virtuales intercambiables (IVI) y presenta un conjunto de operaciones y propiedades que ejercen la funcionalidad de soporte .NET para conmutadores específicos. Las Bibliotecas de Clase .NET de NI-SWITCH requieren la instalación del controlador de instrumentos NI-SWITCH para funcionar.[15]

Una de las referencias API más utilizada es “Session”, esta es la funcionalidad que permite crear sesiones para los distintos modulo del Chasis PXI que permiten controlar los distintos relés dentro del módulo, también permite realizar

testeos al Chasis para corroborar la comunicación entre los drivers y los módulos, así como obtener información sobre estos que permiten llevar control para el debido mantenimiento.

2.4Métodos de Testing

El Testing en software es un conjunto de pruebas que se llevan a cabo con la intención de probar funcionalidades o encontrar errores en el código. [16]

Entre estos uno de los más comunes son:

- Pruebas funcionales: Nos permiten corroborar las características y comportamientos operacionales de un programa.
- Pruebas unitarias: Se realizan a las funciones del código de manera separada con el fin de verificar que cada una funciona de manera individual.
- Caja blanca: Basan su análisis en las funciones internas teniendo en cuenta su lógica y comportamiento dentro del código.
- Caja negra: No se contemplan las funciones internas sino las respuestas a una ejecución general del programa basándose totalmente en los requisitos.

2.5 Estándar de Restricción de Sustancias Peligrosas

Esta es una directiva creada por el parlamento europeo en febrero del 2003 y actualizada en el 2012, también conocida como Directiva 2011/65/UE y está orientada a la reducción del uso de sustancias peligrosas en aparatos electrónicos [13]. Las restricciones de las sustancias y su cantidad máxima permitida se muestran en la tabla 2.2.

Tabla 2.2. Sustancias restringidas por la norma RoHS. [13]

Sustancia	Cantidad máxima (ppm)
Plomo	1000
Mercurio	100
Cadmio	100
Cromo hexavalente	1000
Bifenilos poli bromados	1000
Éteres difenil poli bromados	1000

3. Metodología

La metodología de diseño y desarrollo de producto con la que se realizó este proyecto se basa en la metodología que se detalla en el libro Ulrich y Eppinger [17], a continuación, se explican las etapas en las cuales fue ordenada esta estrategia de trabajo.

3.1 Búsqueda de información

Para esta sección del proyecto se plantearon las siguientes actividades a desarrollar:

1. Recopilar información sobre las características de los principales dispositivos de Rhythmia Mapping System y su función.
2. Investigar los dispositivos y las conexiones que componen al actual entorno de automatización.
3. Estudiar el entorno de programación utilizado por el equipo de Rhythmia, así como las principales librerías implementadas.
4. Investigar sobre las normativas de seguridad requeridas para el diseño de elementos electrónicos.
5. Definir los principales componentes a evaluar.
6. Definir los dispositivos y las señales que deben incluir en el sistema de conexiones.
7. Elaborar una lista de requerimientos para el evaluador y la infraestructura del sistema.

Las actividades 1, 2 y 3 se contemplan en el desarrollo del marco teórico, la primera actividad tiene como objetivo conocer las características generales de los dispositivos que se encuentran dentro del alcance de las pruebas desarrolladas por el equipo. La actividad numero 2 busca obtener un entendimiento técnico de los dispositivos que se deben contemplar para la automatización de las pruebas dependientes de hardware, comprender la relación entre sus conexiones y la información que transmiten.

Con la actividad número 3 se pretende entender los actuales códigos del sistema y el método de programación que utilizan ya que estos contienen clases y

funciones que se pueden incluir dentro del desarrollo del código del evaluador para facilitar su comprensión y mantener la estructura del paradigma modular.

A pesar de que este proyecto se desarrolla en el ámbito médico, el sistema que se desea desarrollar se implementara únicamente para prueba de funcionamiento y ninguno de los elementos utilizados se incluirá en un entorno clínico por lo cual no se contemplaran normas como la ISO 13485, ISO 14971, entre otras que evalúa a los dispositivos médicos y únicamente dentro de la actividad 4 se tomaran en cuenta las normas sobre seguridad ambiental como el estándar RoHS.

Las actividades 5, 6 y 7 son indispensables para el proceso de diseño, primeramente, la actividad 5 y 6 permitirá delimitar el alcance de los dispositivos que se deben incluir en los diferentes bloques de la infraestructura y con base a esto se desarrollarán los requerimientos.

3.2 Generación y evaluación de conceptos

Para esta sección se plantearon las siguientes actividades a desarrollar:

1. Identificar posibles soluciones para el evaluador y generar los conceptos.
2. Identificar posibles soluciones para las conexiones y diseño de circuitos impresos y generar conceptos.
3. Identificar posibles soluciones para el diseño mecánico y generar conceptos.
4. Generar diagramas de flujo que represente el código de evaluación.
5. Seleccionar los mejores conceptos según los criterios establecidos.

En las primeras 3 actividades se deben identificar las posibles soluciones que cumplan con los criterios de funcionalidad según los requerimientos establecidos para cada diseño utilizando fuentes confiables como consejos de expertos, consulta a literatura publicada, comparación con códigos relacionados.

La actividad 4 permitirá establecer de manera detallada el proceso de evaluación, poder identificar el flujo de entradas y salidas de todas las funciones, también nos permite observar un panorama general del flujo de código para así ver detalladamente su coherencia y abstracción. Los diagramas de flujo contendrán la

estructura principal y el orden de las funciones, también representarán la estructura de cada función, los valores de interés y el manejo de las variables.

Para la actividad 5, se utilizara una matriz de selección con el formato observado en la tabla 3.1, en base a este formato se debe tener un concepto base de comparación, debido a que actualmente dentro del equipo de Rhythmia no se cuenta con diseños similares para el evaluador, los PCB y el accesorio mecánico se escogerá el primer concepto desarrollado como base y se compraran los demás respecto a este, cada concepto se calificara utilizando las puntuaciones observadas en la tabla 3.2, debido a que cada primer concepto no tiene base de comparación se calificara cada criterio con un desempeño relativo igual que la "referencia". La interpretación de la puntuación adecuada de los criterios para cada concepto será a juicio del equipo de diseño, así como los pesos para cada uno.

Tabla 3.1. Matriz de evaluación de conceptos. [17]

		Conceptos			
		A		B	
Criterio de Selección	**Peso**	**Calificación**	**Evaluación Ponderada**	**Calificación**	**Evaluación Ponderada**
Total					

Tabla 3.2. Escala de calificación de criterios. [17]

Desempeño Relativo	Calificación
Mucho Peor que la referencia	1
Peor que la referencia	2
Igual que la referencia	3
Mejor que la referencia	4
Mucho mejor que la referencia	5

3.3 Desarrollo y validación del producto

Para esta sección se plantearon las siguientes actividades a desarrollar:

1. Programar el evaluador en el entorno designado.
2. Desarrollar un prototipo analítico-enfocado de los circuitos de los PCB.

3. Desarrollar un prototipo analítico-enfocado del accesorio mecánico.
4. Validar el código por medio de pruebas de software.
5. Validar los PCB por medio de tablas lógicas y simulaciones de funcionamiento.
6. Validar accesorio mecánico por medio de simulaciones de ensamblaje y fijación de componentes.

En la primera actividad se integran todos los conceptos relacionados con el evaluador, se comenzará a escribir código en el entorno de programación Python. Se utiliza squish y Pycharm para la verificación de la sintaxis y se aplican prueba de las distintas partes del código de manera unitaria y en conjunto.

Para las actividades 2 y 3 se integrarán los conceptos ganadores para los circuitos y el diseño mecánico, en ambos casos se generarán prototipos analítico-enfocados utilizando las herramientas de simulación OrCAD y SolidWorks las cuales nos permiten seleccionar según los requerimientos de diseño los materiales, componentes y piezas estándar que se van a utilizar.

Para la actividad 4 se plantea el desarrollo de pruebas de tipo funcionales y unitarias de caja negra y caja blanca, estas se detallas en la sección 2.5.

En la actividad 5 se utilizarán tablas de lógica para verificar la funcionalidad de los conmutadores de señales, además del cálculo de corriente y tensión para asegurar la integridad de los elementos, así como simulaciones de los circuitos y su funcionamiento.

La actividad 6 incluye el diseño tridimensional de las distintas partes del accesorio, así como un diseño dimensional de los componentes que resguarda para la simulación del ensamble y desensamble con el fin de comprobar la facilidad de manejo del accesorio.

3.4 Generación de documentación

Para esta sección se plantearon las siguientes actividades a desarrollar:

1. Documentar los diagramas de flujo y manual de uso del evaluador.
2. Elaborar planos eléctricos y planos del PCB.
3. Elaborar planos mecánicos.
4. Generar el informe final de proyecto de graduación.

La actividad 1 consiste en la representación detallada de la funcionalidad del evaluador y sus partes por medio de diagramas de flujo, además de incluir instrucciones de uso para comprensión de los usuarios.

Las actividades 2 y 3 documentan a detalle las características requeridas para la construcción del accesorio mecánico y los PCB. Finalmente, en la actividad 4 se documentará todo el proceso de diseño y los resultados en el informe del proyecto.

4. Desarrollo de requerimientos

Inicialmente se procedió a recopilar las necesidades expuestas por el grupo de Rhythmia en relación con las distintas partes del proyecto, esto con el objetivo de tener claro el alcance que debe tener cada diseño para solventar todas las funcionalidades y características requeridas.

4.1 Requerimientos del software de evaluación

Tabla 4.1 Requerimientos del evaluador. Fuente: Propia.

Requerimiento	Valor (Unidad)
1. Cantidad de dispositivos a evaluar.	4
2. Cantidad de módulos a evaluar.	1
3. Cantidad de relés a evaluar.	Max 100
4. Generación de una hoja con los resultados de evaluación con distinto niveles de registro.	Cumple/No cumple
5. El software se ejecuta en la ventana de comandos mediante una sola instrucción.	Cumple/No cumple
6. El directorio de resultados puede ser elegido por el usuario.	Cumple/No cumple
7. Manejo de Errores y terminaciones forzadas	Cumple/No cumple
8. Verificación de conexión a internet de los dispositivos	Cumple/No cumple
9. Verificación de comunicación entre módulo y controlador.	Cumple/No cumple
10. Verificación de conmutación y estado de los relés.	Cumple/No cumple
11. Los parámetros de entrada presentan un orden fijo.	Cumple/No cumple

De la tabla 4.1 los primeros 2 requerimientos se contemplan en relación con los dispositivos principales que componen cada parte del sistema, dentro de estos podemos encontrar:

- La estación de señales que se comunica con la estación de trabajo.
- La estación de trabajo en la cual se encuentra el Software de Rhythmia.
- La estación de testeo donde se ejecutan las pruebas.

- El controlador del Chasis PXI, que maneja los relés.
- El módulo del PXI Chasis que contiene los relés.

La selección de estos dispositivos fue en base a su importancia tomando en cuenta su peso dentro de todo el sistema de automatización y sus funcionalidades según los textos [1][10][11]. Actualmente el Chassis PXI modelo (PXI2-2569) cuenta con un módulo 1E3BF44 el cual tiene 100 relés individuales para conexión/desconexión, actualmente se han utilizado 71 relés, pero debido a que dentro del alcance del proyecto se contempla un posible añadido de señales lo que implica la definición del requerimiento 3 de la tabla 4.1.

El evaluador además de verificar la funcionalidad de distintos equipos, elementos y conexiones también pretende devolver al usuario los resultados obtenidos en esta evaluación, la hoja de resultados debe ser coherente y fácil de interpretar por ello cada acción registrada se debe dividir según su importancia para que el usuario únicamente acceda a la información que se considera de interés en la evaluación por ello se aplica el requerimiento 4.

Como se define en el requerimiento 5 no es necesaria la implementación de una interfaz gráfica, el usuario únicamente debe ser capaz de ejecutar el programa mediante la consola, haciendo uso de una única instrucción la cual contenga todos los detalles para definir las características del proceso de evaluación, esto incluye como se especifica en el requerimiento 6 que el usuario pueda indicar la dirección de almacenamiento del registro de resultados.

Respecto al requerimiento 5 el usuario debe ser capaz de caracterizar el proceso de evaluación de la siguiente manera:

- Seleccionar la evaluación estándar.
- Seleccionar una evaluación únicamente del PXI Chassis.
- Para cada tipo de evaluación, seleccionar si los resultados deben contener o no el nivel detallado.
- Ser capaz de seleccionar si se desean evaluar los elementos de conexión de todos los puertos o únicamente los especificados por el usuario.

- En relación con el requerimiento 6 el usuario debe ser capaz de ingresar la dirección del archivo de resultados o utilizar la dirección estándar.
- Generación de mensaje de ayuda en caso de que el usuario lo solicite.

Durante la ejecución del evaluador se inician múltiples procesos de conexión y se cambian el estado de distintos elementos en relación con las acciones realizadas para los requerimientos 8, 9 y 10, es por ello se debe tener un mecanismo de control sobre el cierre de estos procesos en caso de un fallo inesperado y en base a esto surge el requerimiento 7.

4.2 Requerimientos de circuitos impresos

Tabla 4.2 Requerimientos de Circuitos impresos. Fuente: Propia

Requerimiento	Valor (Unidad)
1. Nivel de tensión de señales analógicas	1 (V) a 5 (V)
2. Alimentación del circuito	5 (V)
3. Corriente máxima en los puertos de entrada M, A y B	1 (A)
4. Corriente máxima en los puertos de salida M, A y B	1 (A)
5. Corriente máxima en los puertos de ablación	2 (A)
6. Cantidad de dispositivos en los puertos de entrada M, A y B	3 por puerto
7. Cantidad de dispositivos en el puerto de ablación	2
8. Cantidad de dispositivos en los puertos de salida M, A y B	3 por puerto
9. Cantidad de dispositivos con conexión a generador de ablación	2
10. Tiempos de conmutación	< 5ms
11. Frecuencia de las señales	100 kHz Max
12. LED para indicar conexión y desconexión de dispositivos	Cumple/No Cumple
13. Elaborar PCB de acuerdo con la normativa IPC-2221 B	Cumple/No Cumple

14. Controla la conexión y desconexión de los dispositivos en cada puerto (Entrada y Salida)	Cumple/No cumple

En esta sección todos los requerimientos que se mencionen pertenecen a la tabla 4.2 y describen las características más importantes que se esperan tener en todos los PCB diseñados, el comportamiento primordial de cada PCB y que se ve reflejado en el requerimiento 14 es que estos controlen la conexión y desconexión de los dispositivos que se conectan a los puertos de la estación de señales, esto con el objetivo de poder simular de manera automática este comportamiento que es deseable para el desarrollo de la mayoría de pruebas incluidas en el alcance del sistema de automatizado. Además, con el fin de poder llevar un control visual dentro de la configuración de hardware se estipula el requerimiento 14.

Las señales en cada uno de los puertos de entrada, salida y ablación provienen de las señales de distintos catéteres o del generador de señales, por medio del procesamiento interno de los distintos dispositivos estas alcanzan tensiones que varían entre 1 y 5 Voltios, tomando esto en cuenta y basándonos en las características de los dispositivos de conmutación se definen los requerimientos 1 y 2.

Los catéteres y parte del sistema del generador de ablación son considerados como circuitos de baja impedancia por lo cual manejan niveles de corriente de 1 a 2 amperios dependiendo del puerto, teniendo esto en cuenta se definen los requerimientos 3, 4 y 5.

Para los puertos de entrada y salida según el alcance de las pruebas que se desean automatizar no se deberán conectar más de 3 catéteres por puerto, teniendo esto en cuenta se definieron los requerimientos 6 y 8.

Dentro del equipo el mayor volumen de pruebas de ablación se realizan utilizando 2 tipos de catéteres el “MiFi” y el “StablePoint”, cada uno de estos catéteres utiliza una caja maestra de conexión distinta que se conectan al mismo puerto de las estación de señales y al mismo generador de ablación, excluyendo unas cuantas pruebas que se decidieron dejar fuera del alcance del sistema

automatizado únicamente se deben conectar estos 2 tipos de dispositivos es por ello que el requerimientos 7 y 9 se definen de esta manera.

Por las características técnicas de los puertos de la estación de señales la cual establece un tiempo mínimo de reconocimiento de un dispositivo asociado con la estructura de programación de las pruebas automatizadas se espera que entre la señal de control y la conmutación de la señal no existe un tiempo mayor a 5ms como lo establece el requerimiento 10.

Según la descripción técnica obtenida de las señales que se van a conmutar la frecuencia máxima de señales que recibirá el dispositivo de conmutación no superará los 100 KHz, esto en señales cuadradas por ello se define el requerimiento 11.

Los PCB diseñados en Boston para el sistema de automatización deben seguir como estándar mínimo las características definidas por la IPC2221-B, es por ello por lo que esta se incluye como parte de las características de diseño en el requerimiento 13.

4.3 Requerimientos del accesorio mecánico de resguardo

Tabla 4.3. Requerimientos para accesorio mecánico de resguardo. Fuente: Propia

Requerimiento	Valor (Unidad)
1. Cantidad de placas a resguardar	8
2. Tolerancia de dimensiones	0.5 mm
3. Cantidad de conectores de entrada	22
4. Cantidad de conectores de salida	8
5. Fijación de tapa por cejilla	Cumple/No cumple
6. Fijación de placas por unión desmontable de tornillo	Tipo M3
7. Sellado para polvo	Cumple/No Cumple
8. Aliviadores de estrés en los puertos de conexión	Cumple/No cumple

9. Disponer de ventilación	Cumple/No cumple
10. Configuración para facilitar el mantenimiento	Cumple/No cumple
11. Material ligero	Cumple/No cumple
12. Resistencia a temperatura	35(°C) max
13. Material fácilmente manejable y no deformable	Cumple/No Cumple

En esta sección todos los requerimientos mencionados son los incluidos en la tabla 4.3. Los PCB diseñados serán para la conmutación de los dispositivos que se conectan en los puertos de entrada, salida y ablación. A los puertos de entrada M, A y B se les conmutara la conexión de 3 dispositivos cada uno eso hace un total de 9 puertos, esto ocurre igual para los puertos M, A y B de salida lo cual genera 9 puertos más, y para la señal de ablación y del generador de señales se utilizarán 4 puertos más, obteniendo en total 22 puertos de entrada y 8 de salida. Así se definen los requerimientos 1, 3 y 4.

Debido a que la carcasa de protección no es un elemento que requiera alta precisión en sus diseños y contemplando que se desea que este sea un elemento fácilmente desmontable para posibles cambios o mantenimiento se selecciona en el requerimiento 2 una tolerancia de 0.5mm la cual además se ajusta con las características promedio que obedecen las impresoras 3D de gama media evitando un aumento de costo en la manufactura.

El sistema de automatización es un proyecto que se mantiene en constante crecimiento y sus elementos deben estar adecuados a fácil mantenimiento o posibles reajustes es debido a esto que tanto los elementos como PCB y la carcasa tendrán fijaciones desmontables con el cuidado de mantener un buen sellado para protección al polvo como lo definen los requerimientos 5, 6, 7 y 10.

Es común que los dispositivos probados estén siendo cambiados ya que los catéteres tienen fecha de expiración, esto conlleva a una continua conexión y

desconexión de los dispositivos para aumentar el tiempo de vida de los conectores y reducir el estrés mecánico en las áreas circundantes a los puertos se define el requerimiento 8.

Según estudios previos de las condiciones de operación continua del generador de señales las temperaturas máximas que alcanzan los 30°C internamente en las épocas de altas temperaturas, al ser este accesorio de resguardo un elemento de características similares y con el fin de evitar sobrecalentamientos se definen los requerimientos 9 y 12.

El elemento de resguardo como parte del sistema de automatización de hardware a pesar de que no debe estar en constante movimiento debe ser fácil de reubicar y además ser de un material que permita de fuertes sujeciones o traslados sin que sufra algún tipo de daño, además no debe ser frágil, pueda ser operado y abierto sin sufrir daños por ello se definen los requerimientos 11 y 13.

5. Elaboración y selección de conceptos

En esta sección de plantean las propuestas de distintos conceptos para la resolución de los problemas plantados, cada concepto se desarrolló tomando en cuenta los requerimientos planteados en la sección 4, es posible que para distintos problemas se puedan plantear múltiples soluciones por lo cual se hará uso de la tabla 3.1 para seleccionar el mejor candidato.

5.1 Conceptos para el diseño del software de evaluación

5.1.1 Paradigma de programación

El sistema automatizado de pruebas dependientes de hardware está planteado para poder ser actualizado conforme todos los productos del Rhythmia Mapping System mejoran y adquieren nuevas funcionalidades, todos los sistemas de testeo deben actualizarse en conjunto, sobre esta línea el paquete RTMD presenta un diseño orientado a objetos que contiene las clases que definen a los objetos que caracterizan a cada puerto y pueden ser fácilmente ampliables.

Teniendo en cuenta esta necesidad de dinamismo se seleccionó para el software de evaluación un paradigma de programación modular, basándonos en las siguientes consideraciones.

Primeramente, el software de evaluación interactúa con elementos que tienen distintas características y no necesariamente se relacionan entre ellos de manera directa, es por ello considerar cada elemento por separado dentro del diseño del código nos permite tener módulos auxiliares de evaluación para cada uno y ser invocados en el módulo principal del programa permitiéndonos agregar en un futuro nuevos elementos de evaluación o modificar fácilmente los actuales. En [18] recomiendan el paradigma modular para casos similares.

También existen características en común entre algunos elementos, estas pueden ser declaradas en el código a manera de variables globales, las cuales son una característica clave en el paradigma modular y pueden ser utilizadas en las distintas sub-evaluaciones sin necesidad de declararlas múltiples veces.

Esto genera una ventaja en la importación de los objetos desde el paquete RTMD ya que las características de ciertos objetos se plantea manejarlas de manera global dentro del software debido a que requieren ser utilizados por

múltiples módulos auxiliares. Otra facilidad es en el manejo de las banderas las cuales definen los procedimientos de evaluación.

Parte primordial del proceso de diseño es definir el orden de evaluación de los elementos como parte del procedimiento principal pero cada evaluación puede descomponerse en procedimientos más pequeños facilitando el proceso de depuración de cada procedimiento por separado utilizando pruebas de caja blanca entre otras, concluyendo en que si cada procedimiento funciona correctamente la integración de todos debe proporcionar un procedimiento principal funcional, como se plantea en [19].

5.1.2 Registro de eventos y resultados

Los resultados de cada elemento deben de estar disponibles para el usuario al final de cada ciclo de evaluación, pero la información se espera que presente un orden y formato. Como es requerido los resultados deben ser mostrados tanto en consola como en un archivo y que toda la información registrada contenga fecha, hora y nivel de registro. Para esto se evalúan 2 métodos de registro de datos basados en los módulos funcionales disponibles en Python.

Como concepto 1 se propone la configuración básica de la librería “logging” la cual es parte de las librerías estándar de Python y el uso de la librería “datetime” para el control de la fecha y hora. En la tabla 5.1 podemos observar la forma de implementación de cada función utilizada. Este tipo de implementación se puede observar en [20].

Tabla 5.1. Funciones para logging básico. Fuente: Propia

Función	Descripción
basicConfig ()	Retorna módulo de registro único
strftime()	Retorna la hora y fecha según el formato que se indique.

Para el concepto 2 se propone el uso de las clases “Fomatter()”, “FileHandler” y “StreamHandler()” las cuales pertenecen a la librería “logging” y se pueden ver implementadas en [21], acá se explica que el uso de estas clases permite configurar múltiples módulos de registro, a los cuales se les configura previamente el formato

y el o los destinos donde se registraran los mensajes. Para este concepto la configuración de la librería y las clases varia como se observa en la tabla 5.2.

Tabla 5.2. Funciones para logging con Handler. Fuente: Propia

Función	Descripción
getLogger()	Retorna módulo de registro, permite crear múltiples módulos.
setLevel ()	Configura el nivel máximo de mensajes en los módulos de registro.
FileHandler ()	Retorna clase que controla la salida de registro del archivo de resultados.
StreamHandler ()	Retorna clase que controla la salida de registro de consola.
Formatter ()	Retorna clase que contiene el formato para el registro.
setFormatter ()	Configura el formato de las salidas de registro.
addHandler ()	Agrega las clases creadas al módulo de registro.

A razón de escoger entre ambos conceptos de ha decidido utilizar la matriz de selección que se observa en la tabla 5.3, además el concepto 1 se tomará como referencia obteniendo calificación de 3 para todos los criterios, la calificación del segundo concepto será evaluada tomando en cuenta lo detallado en [21], acá describe las implementaciones de ambos conceptos junto con ejemplos de uso y características de las funciones.

Tabla 5.3. Matriz de selección para concepto de registro de eventos. Fuente: Propia

		Conceptos			
		Logging Básico		Logging con Handler	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Modularidad	45%	3	1.35	4	1.80
Actualizable	30%	3	0.90	5	1.50
Facilidad de implementación	25%	3	0.75	2	0.50
Total		3.00		3.80	

Como podemos observar de las tablas 5.1 y 5.2, para definir los módulos de registro y sus características como el nivel máximo de los registros de evento, el concepto 1 lo realiza todo por medio de la invocación de una única función y los parámetros dispuestos para esta función afectan al registro de consola y de archivo por igual, mientras que el concepto 2 nos permite definir múltiples módulos de registro en caso de ser requerido, con formatos que se pueden definir de manera individual para los eventos de consola y de archivo que se parametrizan individualmente, teniendo esto en cuenta se califica el segundo concepto con un 4 proporcionando mejor modularidad al separar la parametrización de los módulos de registro.

En caso de ser requeridas modificaciones individuales a los formatos de registro, el concepto 2 nos permite actualizar luego de creado el módulo los niveles de registro de eventos de manera separada o crear nuevos futuros módulos de registro con distinta dirección del anterior sin afectarlo por lo cual a diferencia del concepto 1, este presenta más capacidad para futuras actualizaciones siendo calificado con 5, mucho mejor que el concepto de referencia.

Finalmente, el concepto 1 debido a que una única función define toda la configuración del módulo de registro, su implementación dentro del código es menos compleja que la del concepto 2 teniendo este una calificación peor que la referencia para este criterio, a pesar de ello el total define que el concepto 2 se adapta mejor a los requerimientos de diseño y por ello es seleccionado para su implementación.

5.1.3 Lectura de parámetros de entrada

Como requerimiento estándar del software definen que la interfaz de usuario debe ser mínima y que las instrucciones al software de evaluación deben ser dadas por una única línea de comando que pueda resumir las características que el usuario requiere, por lo tanto, se debe definir una estructura de comando inicial que pueda incluir lo necesario para identificar los parámetros del evaluador como se especifican en la sección 4.1, los parámetros de entrada según análisis de los requerimientos se propone que tengan el formato mostrado en la figura 5.1.

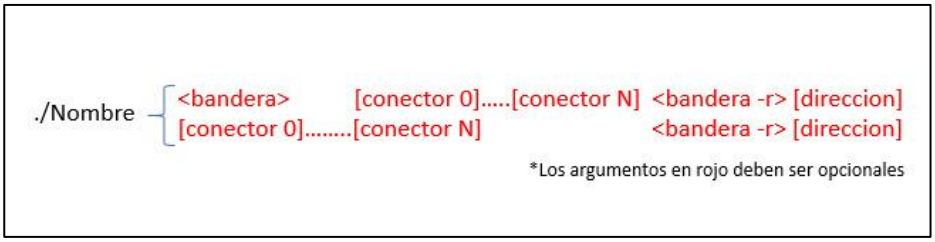


Figura 5.1 Formato de argumentos de entrada. Fuente: Propia

Como concepto 1 se evaluó el uso de la librería “argparse ()” la cual como analizador sintáctico para opciones, argumentos y subcomandos es utilizado en otros códigos dentro del equipo de trabajo, la sintaxis de definición de estos argumentos se realiza por medio del método “add_arguments” el cual crea objetos de cada argumento añadido con la sintaxis observada en la figura 5.2, esta librería identifica automáticamente a los argumentos y permite consultar los valores ingresados para cada uno, además se caracteriza por:

1. Permite al usuario definir conjuntos de argumentos de tipo posicional y opcional.
2. Para los posicionales cada argumento se le define su bandera identificadora, cantidad de valores del argumento y el tipo de valores.
3. Los argumentos posicionales no pueden ser omitidos al invocar.
4. Los valores evaluados siempre deben ir acompañados de un argumento, y la cantidad de valores para un argumento deben ser definidos al inicio.

```
ArgumentParser.add_argument(name or flags...[, action][, nargs][, const][, default][, type][,
choices][, required][, help][, metavar][, dest])
```

Figura 5.2 Sintaxis de implementación del método add_argument.[22]

Para el concepto 2 se plantea generar un analizador de argumentos personalizado utilizando la función específica del sistema “sys.argv” la cual retorna una lista con todos los argumentos con los cuales se invocó el código. Por medio del manejo manual de la lista de argumentos se propone un análisis de cada uno haciendo uso de condicionales, este estará caracterizado de la siguiente manera:

1. Todos los argumentos serán opcionales y presentan de 0 a N argumentos válidos.
2. Los parámetros de entrada deben presentar un orden especifico.

3. Los valores de entrada pueden ser omitidos y no necesariamente deben ir acompañador de un argumento.
4. Los argumentos son tratados como una lista y se analizan manualmente en distintas funciones según sea conveniente.

Para seleccionar entre ambos conceptos se utilizó la matriz de selección que se puede observar en la tabla 5.4, para esta selección se tomará como referencia el concepto 1 el cual se le dará una calificación de 3 en todos los ámbitos, el segundo se califica con base en discutido con el equipo sobre las características de cada concepto y como estos se acoplan mejor con los requerimientos observados en la sección 4.1 tomando en cuenta las funcionalidades presentadas en [22].

Tabla 5.4. Matriz de selección de método de lectura de parámetros de entrada. Fuente: Propia

		Conceptos			
		Implementación argparse		Implementación personalizada	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Modularidad	40%	3	1.20	2	0.80
Actualizable	30%	3	0.90	3	0.90
Nivel de abstracción	30%	3	0.90	5	1.50
Total		3.00		3.20	

El primer criterio es el de modularidad, como se ejemplifica en [23] y [24] la función “argparse” crea un objeto por cada argumento que se define, esto proporciona una referencia directa a los valores de cada argumento permitiendo ser invocados de manera separada a diferencia del concepto 2 en el cual se trabaja con una lista en donde cada argumento y valor relacionado están juntos, esto hace que a diferencia del concepto 1 la subdivisión de elementos en la implementación personalizada es peor.

Ambos permiten añadir nuevos argumentos y modificar los antiguos con nuevas características en caso de ser requerido, estos trabajan de manera similar a la hora de actualizar los argumentos, por un lado el concepto 1 como se muestra en [24] se le pueden añadir la cantidad de argumentos deseados y la cantidad de

valores que acompañan al argumento puede ser fácilmente modificable mientras se conozca el dato, de igual manera al trabajar directamente con la lista de argumentos en el concepto 2 en caso de ser modificado solo se deben reflejar estos cambios en el código.

Como se explica en [24] el verdadero inconveniente del concepto 1 es que se debe trabajar bajo los parámetros de la librería, esto nos impide manipular en su totalidad los elementos de entrada, un ejemplo de esto es que los valores en la lista de entradas deben siempre ir acompañados de un argumento (bandera) para que puedan ser identificados o que el número de valores para cada argumento debe ser especificado previamente a diferencia del concepto 2 en el cual se trabajaría directamente con la lista de entradas lo cual nos permite manejar un nivel de abstracción menor que nos facilita la manipulación de los parámetros.

Finalmente, ambos conceptos como denota su clasificación son casi igual de buenos, pero por criterio experto se decide que una implementación personalizada se adapta mejor a lo deseado, permitiendo ingresar valores sin necesidad de ir acompañados de un argumento y trabajando todos los argumentos como opcionales.

5.1.4 Método de verificación de conexión

Como parte del funcionamiento del sistema de testeo, la estación de señales debe ser capaz de comunicarse con la estación de trabajo administradora del testeo, ambos elementos deben tener una conexión a la red por lo tanto como parte de la evaluación se debe comprobar que dicha conexión se encuentra habilitada para ello se plantea el uso del protocolo de conexión “ssh” como herramienta de verificación.

Como concepto 1 se propone utilizar la clase “RemoteSystem”, que se encuentra implementada en el entorno de trabajo de squish del equipo de Rhythmia, como parte de la librería propia de squish, únicamente requiere ser invocada especificando la dirección de conexión, y esta misma devuelve la sesión ssh que nos permite la ejecución de comandos en consola y manejar las salidas de un Shell tradicional.

Por otro lado, se propone como concepto 2 el uso de la librería “Pexpect” la cual contiene la clase “pxshh” que está especializada en control de sesiones ssh,

esta clase permite crear sesiones remotas con parámetros para dirección, usuario y contraseña del servidor, ingresar comandos directamente a la consola y buscar patrones de respuesta.

En la tabla 5.5 se observa la matriz de selección con la cual se evaluaron ambos conceptos, el primer concepto se toma como referencia ya que previamente se ha implementado en otros códigos del equipo de Rhythmia.

Tabla 5.5. Matriz de selección para método de verificación de conexión. Fuente: Propia

		Conceptos			
		RemoteSystem		Pexpect	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Nivel de Dependencia	45%	3	1.35	5	2.25
Facilidad de implementación	25%	3	0.75	1	0.25
Nivel de abstracción	30%	3	0.90	3	0.90
Total		3.00		3.40	

El primer criterio de selección es el nivel de dependencia en paquetes externos a Python, el concepto 1 tiene un alto grado de dependencia el cual es una característica no deseable, sus funciones pertenecen al paquete de squish y no es descargable por lo tanto si se utiliza este método se debe asegurar que las estaciones de evaluación contengan squish con el paquete instalado y debidamente configurado ya que esta realiza la conexión al servidor de squish que tenga la estación de trabajo, requisito que no es estrictamente necesario para la estación de evaluación mientras que el concepto 2 es una librería publica que comúnmente como se especifica en [25] esta es parte de las librerías estándares de Python por lo cual se le clasifica con 5, como un criterio que cumple mucho mejor que la referencia.

Por otro lado, la implementación del concepto 1 ya que esta solo debe ser invocada y no requiere ninguna configuración, sus clases y funciones ya contienen una configuración estándar que únicamente requiere las direcciones IP para

verificar la conexión. El concepto 2 como se ve en los ejemplos mostrados en [26] requiere más detalle en su implementación, la clase “pxshh” tiene funciones más específicas por lo que se requieren más líneas de código e interpretación de los datos de retorno es por ello por lo que se clasifica con 1, mucho peor que la referencia.

Como ultimo criterio se considera el nivel de abstracción, para esta clasificación se considera el concepto 2 igual a la referencia ya que a pesar de que este requiere más líneas de código ambos son implementaciones del protocolo SSHv2 utilizando el indicador de Shell para sincronizar las salidas con sus hosts remotos por lo tanto a ambos conceptos se les da la misma clasificación. Finalmente, con base en la clasificación de los criterios se selecciona el uso de la librería Pexpect.

5.1.5 Manejo de errores

Durante el uso de un software nunca se está exentó de encontrar errores durante su ejecución, comúnmente estos ocurren como parte de la invocación incorrecta de funciones o valores de entrada que no concuerdan con la lógica del programa, como parte de una correcta funcionalidad dentro del software de evaluación se requiere lidiar y capturar estas excepciones es por ello por lo que se proponen 3 métodos distintos para el manejo es estas.

Como concepto 1 se plantea el flujo de manejo de excepciones previstas ejemplificado en la figura 5.3, acá como se detalla en [27] existen múltiples tipos de excepciones estándar tales como NameError, TypeError, entre otras. El objetivo de este manejo es realizar un estudio detallado de las funciones invocadas y las operaciones dentro del código para definir el tipo de errores que se pueden producir y generar una excepción estándar por cada uno de ellos, cuando la excepción ocurra se realiza el manejo según el tipo de excepción y se termina la ejecución.

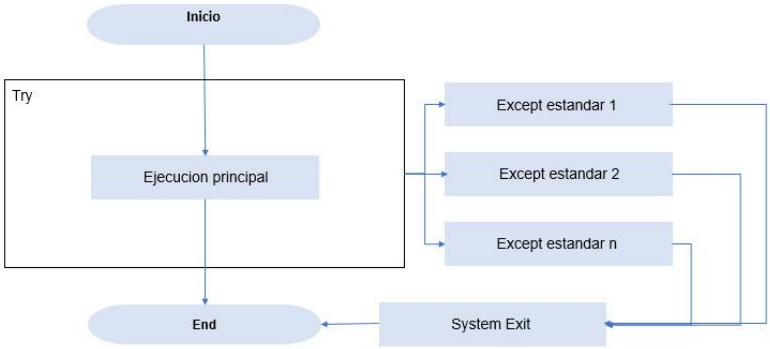


Figura 5.3. Diagrama de flujo de manejo de excepción del concepto 1. Fuente: propia

Para el concepto 2 se propone un manejo general de excepciones inesperadas de manera que se generaliza la invocación de la excepción para cualquier tipo de error que se produzca como se observa en la figura 5.4.

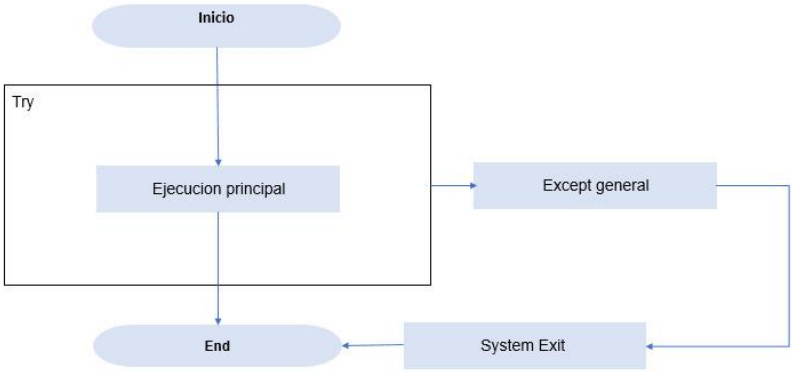


Figura 5.4. Diagrama de flujo de manejo de excepción del concepto 2. Fuente: propia

Para el concepto 3 se plantea una ampliación del concepto 2 y el cual sería un manejo general de excepciones con control y registro de error, este se caracteriza por tratar todas las excepciones como inesperadas y cuando estas ocurren realizar un registro del error por medio del parámetro “str(e)” y finalmente realizar un control de la terminación por error del programa, cerrando las sesiones de conexión y finalizando todas las ejecuciones de manera segura, esto se muestra en la figura 5.5.

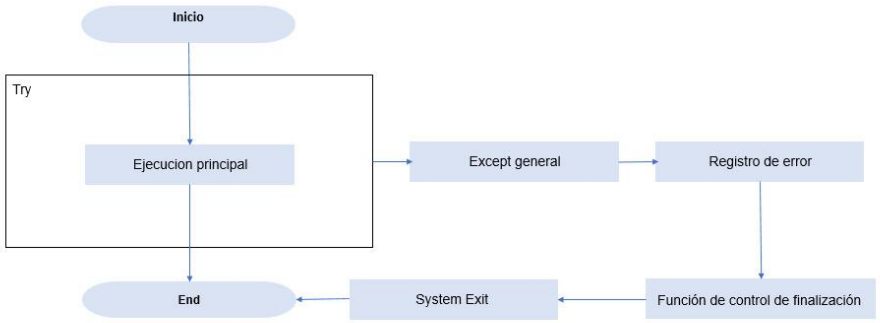


Figura 5.5. Diagrama de flujo de manejo de excepción del concepto 3. Fuente: propia

En la tabla 5.6 podemos observar la matriz para la selección de conceptos, debido a que no hay implementaciones previas se toma el concepto 1 como referencia, para la calificación de los conceptos de tomo en cuenta opiniones expertas discutidas dentro del equipo de Rhythmia tomando en cuenta las características técnicas de los métodos de manejo de error expuestas en [28].

Tabla 5.6. Matriz de selección para concepto de manejo de excepciones. Fuente: Propia

		Conceptos					
		Excepciones esperadas		Excepciones inesperadas		Con registro y control	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Interpretación de resultados	35%	3	1.05	3	1.05	4	1.40
Facilidad de implementación	20%	3	0.60	4	0.80	4	0.80
Robustez	45%	3	1.35	4	1.80	5	2.25
Total		3.00		3.65		4.45	

Dentro del registro de resultados se debe llevar como lo presentan los requerimientos de la sección 4.1 un control de los eventos ocurridos dentro del código, el concepto 1 y concepto 2 finaliza el programa presentando el error únicamente en consola, pero no implementa ningún método de registro del error por lo cual el concepto 2 se califica igual a la referencia, mientras que en el concepto 3 se le incluye un registro de estos errores por lo tanto se le clasifica mejor que la referencia.

Para la implementación del concepto 1 se debe realizar un análisis detallado del código y de los tipos de excepciones presentadas en [27] de manera que el tiempo de implementación y las líneas de código serán mucho mayores por otro lado los conceptos 2 y 3 tratan a todas las excepciones como inesperadas, permitiendo una implementación general del método para todas las funciones siendo así mejor que la referencia para el segundo criterio.

Finalmente el criterio 3 define la robustez la cual es la capacidad del método de responder a todas las posibilidades, mientras que el concepto 1 puede tender a omitir algún tipo de excepción no contemplada en el diseño los conceptos 2 y 3 abarcan todas las excepciones posibles, además de esto el concepto 3 implementa una función extra para el control de la finalización del programa por error, considerando todos los elementos con los que el programa de evaluación interactúo y deben ser regresados a un estado estándar por lo tanto hace que este concepto sea mucho mejor a la referencia.

Finalmente basándonos en la clasificación final se selecciona el concepto 3 para ser implementado.

5.2 Conceptos para el diseño de circuitos impresos

5.2.1 Conmutación de señales

El objetivo de la conmutación de señales es poder simular la conexión y desconexión de múltiples dispositivos en los distintos puertos de conexión para aumentar el alcance de las pruebas que se pueden automatizar.

En esta sección se toma en cuenta que todas las señales que deben ser conmutadas son de tipo analógicas, estas pueden provenir de los distintos tipos de catéteres que se deben conectar o del generador de ablación, para el alcance de este proyecto se contemplaran las características más importantes de cada señal con el fin de desarrollar una configuración del sistema de conmutación que pueda ser utilizada de manera general en todos los puertos, ya que un manejo individual de cada señal conlleva una complejidad que excede el alcance de este proyecto.

El concepto 1 es una configuración utilizando multiplexores de tipo 4:1 o 2:1 dependiendo del puerto, esta configuración igual que las características promedio de los multiplexores como se describe en [29] tendrá tiempos de conmutación en

términos de nanosegundos, impedancias de salida en sus canales que van de los 4 a los 10 mΩ y robustez a corrientes de máximo 600mA. La tecnología propuesta para los multiplexores en esta configuración es CMOS ya que son las más comunes en el mercado. Debido a la baja robustez para altos amperajes esta configuración requiere de un acondicionamiento previo de cada señal por medio de un divisor de corriente.

Para el concepto 2 se propone una configuración de múltiples niveles utilizando relés DPDT de baja señal como se muestra en la figura 5.6, esta configuración se caracteriza por que sus dispositivos tienen robustez a altas tensiones y corrientes sus tiempos de conmutación se encuentran en los milisegundos, conmutados por señales analógicas de alimentación en sus bobinas e impedancias de salida entre los 50 a 100 mΩ. Cada nivel de relés conecta a un dispositivo y no se genera ningún estado abierto ya que cada terminal siempre se conecta.

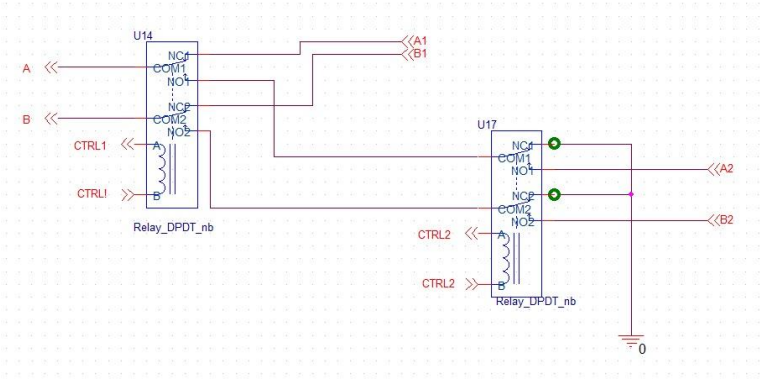


Figura 5.6. Representación simple de la configuración utilizando 2 relés DPDT. Fuente: Propia

El concepto 3 tiene características similares al concepto 2 pero como se muestra en la figura 5.7 este se diferencia ya que en el último nivel de conexión se utilizan relés 2PST disminuyendo así -una línea en cada conexión de dispositivos y teniendo un estado de abierto en los relés.

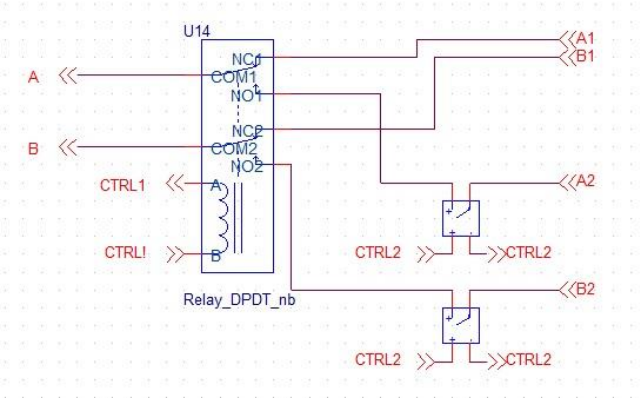


Figura 5.7.Representacion simple de configuración combinando relés DPDT y 2PST

En la tabla 5.7 se encuentra la matriz de selección para el concepto de mejor configuración en el sistema de conmutación, el concepto utilizando multiplexores será tomado como referencia para calificar los 2 conceptos restantes, las calificaciones se realizaron en base a 3 puntos, lectura de hojas de datos y documentaciones, estudios previos realizados por el equipo de trabajo y criterio profesional.

Tabla 5.7. Matriz de selección para concepto de conmutación de señales. Fuente: Propia

		Conceptos					
		Multiplexores		Relés DPDT		Relés DPDT y 2PST	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Robustez	45%	3	1.35	5	2.25	5	2.25
Implementación	25%	3	0.75	5	1.25	4	1
Tiempo de conmutación	15%	3	0.45	2	0.3	2	0.3
Impedancia de salida	15%	3	0.45	1	0.15	1	0.15
Total		3.00		3.95		3.7	

El primer criterio evaluado es la robustez, para este criterio se tomó en cuenta inicialmente la capacidad máxima de tensión y corriente que en promedio resisten los dispositivos de cada configuración, debido a que en prometo los multiplexores presentan una capacidad de tensión similar a la de alimentación y que la resistencia

a la corriente se encuentra en el valor de las centenas en mA, los relés comerciales de baja señal superan con creces estas características manejando tensiones de hasta 40V y corrientes de hasta 3A. Características como protección a sobre voltajes o protecciones a enganches no se toman en cuenta ya que ambos tipos de conmutadores cuentan con ellos.

Para el criterio de implementación tomamos en consideración la facilidad de construcción del PCB, la cantidad de elementos a utilizar y su posible efectividad acoplándose con los demás elementos, como el generador de señales y el PXI Chassis. Teniendo en cuenta que para la configuración con los multiplexores se debe realizar un ajuste a la señal eso conlleva mayor cantidad de dispositivos volviendo así ambos conceptos mejores que la referencia ya que la complejidad del PCB aumentaría, además en el concepto 3 al conllevar un estado abierto según pruebas realizadas puede ocasionar que por corrientes filtradas no se detecte la desconexión total de algunas señales siendo finalmente el concepto 2 el mejor calificado.

Los multiplexores tienen tiempos de conmutación en rangos de nanosegundos por lo cual debido a que los relés de baja señal comercialmente presentan tiempos en rangos de milisegundos estos son calificados peor que la referencia con un 2, a pesar de esto según pruebas realizadas la velocidad de desconexión permitida en los puertos de la estación de señales puede tener un retraso de hasta 5 milisegundos es por ello que a pesar de no ser tan rápidos como el concepto 1 estos tiempos son aceptables y únicamente se consideran un punto más bajo de la referencia.

La impedancia de salida promedio de los multiplexores se encuentra en el rango de decenas de mΩ mientras que para los relés de baja señal se alcanzan en algunos casos las centenas de mΩ es por ello por lo que se califican mucho peor que la referencia, aun así, ambos dispositivos mantienen características de baja impedancia.

Finalmente, de los resultados de la tabla 5.7, podemos decir que el concepto 2 pese a requerir más relés en su configuración, estos se conectan de manera

simple entre ellos y se acoplan de manera óptima a los demás sistemas, presentan alta robustez y características aceptables dentro de los requerimientos.

5.2.2 Selección de relé

En esta sección se tomarán en cuentas las características que se detallan en las hojas de datos de cada uno de los dispositivos seleccionados teniendo en cuenta los requerimientos del sistema. Todos los dispositivos considerados se encuentras comercialmente disponibles y sus características pueden observarse en la tabla 5.8, estos serán calificados siguiendo la misma línea que para la selección de concepto con la matriz observada en la tabla 5.9.

Tabla 5.8. Características técnicas de relés DPDT. Fuente: Propia

Dispositivo	Tensión de bobina	Tiempo de conmutación	Máxima corriente	Impedancia de salida
G6K-2F DC12	12V	2ms	2A	100mΩ
TQ2SA 5V	5V	3ms	2A	75mΩ
TQ2SA 3V	3V	3ms	2A	75mΩ

Para las calificaciones se tomará el primer dispositivo como referencia y los demás serán evaluados respecto a este, todas las puntaciones recibidas serán tomando en cuenta cual se ajusta más a los requerimientos del sistema además de criterios profesionales.

Tabla 5.9. Matriz de selección de relé. Fuente: Propia

		Dispositivos					
		G6K-2F DC12		TQ2SA 5V		TQ2SA 3V	
Criterio de Selección	**Peso**	**Calificación**	**Evaluación Ponderada**	**Calificación**	**Evaluación Ponderada**	**Calificación**	**Evaluación Ponderada**
Robustez	**45%**	**3**	**1.35**	**3**	**1.35**	**3**	**1.35**
Implementación	**25%**	**3**	**0.75**	**4**	**1**	**3**	**0.75**
Tiempo de conmutación	**15%**	**3**	**0.45**	**2**	**0.15**	**2**	**0.3**
Impedancia de salida	**15%**	**3**	**0.45**	**4**	**0.6**	**4**	**0.6**
Total		**3.00**		**3.25**		**3.00**	

Como podemos observar de la tabla 5.9 todos los dispositivos presentan una robustez similar, es por ello por lo que son calificados con un valor de 3 igual a la referencia. Por otro lado, cuando se habla de implementación nos referimos a la simplicidad que conlleva utilizarlos, debido a que todos son relés DPDT su implementación en la configuración seleccionada no será muy distinta, pero debido a que para el dispositivo referencia y el tercero en la tabla se requiere un adaptador de tensión para la alimentación de su bobina, se cataloga al relé TQ2SA 5V con un calor de 4 mejor que la referencia ya que este puede ser alimentado en su bobina con la alimentación del circuito.

Como se puede observar en la tabla 15 en relé G6K presenta mejores tiempos de conmutación que los demás, es por ellos que los otros 2 relés tienen una calificación más baja, a pesar de esto todos se mantienen en el rango aceptado según los requerimientos del sistema.

Para la impedancia de salida ambos relés TQ2 presentan una impedancia de salida más baja que la referencia lo cual los favorece siendo calificados con un 4. Finalmente se selecciona el relé TQ2SA 5V para ser utilizado en la configuración escogida en la sección 5.2.1.

5.2.3 Alimentación de bobina

La alimentación de las bobinas se realizará a través de las tensiones que provienen de los distintos canales del PXI Chassis que funcionaran como señales de control para alimentar las bobinas que conmutan los relés, por ello en esta sección se escogerá el circuito de alimentación óptimo para nuestra aplicación basados en los requerimientos y criterio profesional.

Como concepto 1 se define un circuito para activación de relé simple el cual únicamente por medio de una tensión de alimentación se energiza la bobina para procurar la conmutación del relé y en paralelo presenta una resistencia de protección y un led para indicar que la conexión se realizó, esto se puede observar en la figura 5.8.

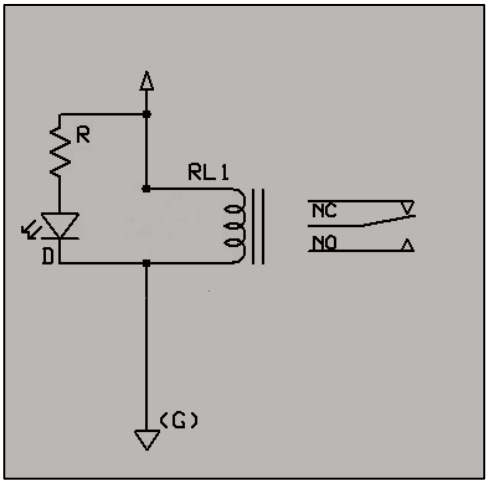


Figura 5.8. Configuración de alimentación simple. Fuente: Propia

Como concepto 2 se selecciona el circuito de alimentación estándar como se muestra en la figura 5.9, el cual presenta características similares al anterior, pero se le agrega un diodo para protección debido a que al cortar la señal de alimentación se puede producir una polarización inversa por la carga de la bobina que podría terminar afectando otros dispositivos del circuito o conectados a este como se explica en [30], el diodo colocado es de tipo flyback con una rango de corriente mayor a 28mA ya que esta es la corriente promedio del embobinado.

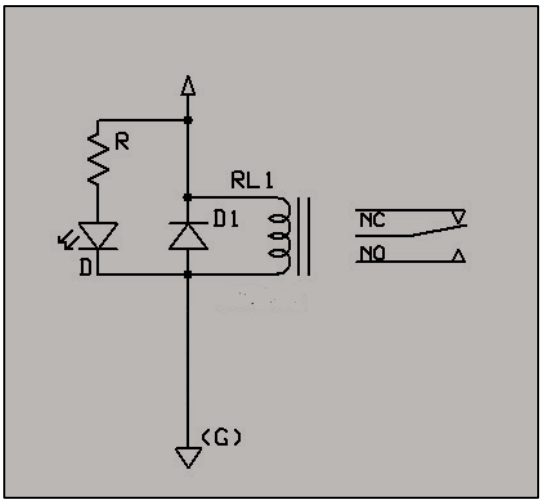


Figura 5.9. Configuración de alimentación estándar. Fuente: Propia

El concepto 3 es una versión modificada del concepto 2 como se observa en la figura 5.10 ya que acá se incluye un transistor NPN teniendo por un lado la fuente de alimentación del circuito y una señal de activación del transistor que puede ser

de carácter digital proveniente de un posible controlador, el transistor le proporciona mayor estabilidad a la conexión y desconexión del circuito permitiendo separar la señal de control y de alimentación como se menciona en [31].

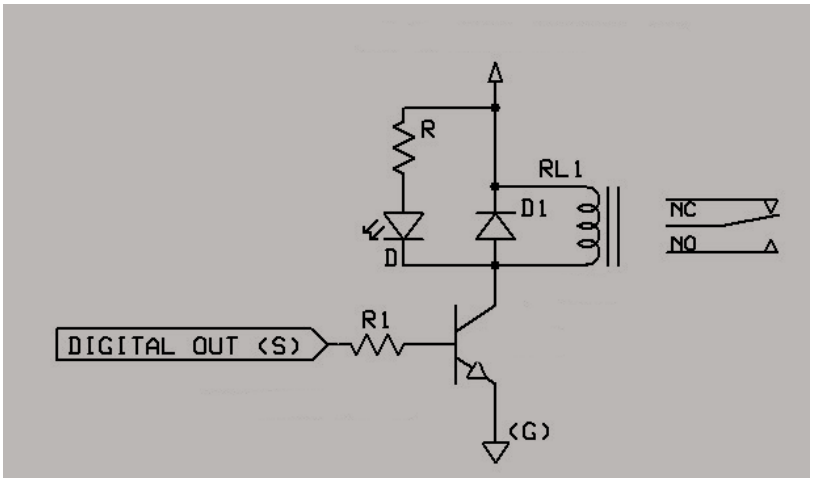


Figura 5.10. Configuración de alimentación modificada. Fuente: Propia

Tabla 5.10. Matriz de selección del concepto de circuito de alimentación de bobina. Fuente: Propia

		Dispositivos					
		Simple		Estándar		Modificado	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Robustez	45%	3	1.35	5	2.25	5	2.25
Implementación	35%	3	1.05	2	0.7	1	0.35
Estabilidad	20%	3	0.60	4	0.80	5	1
Total		3.00		3.75		3.60	

Debido a que el concepto 1 no tiene ningún tipo de protección para posibles polarizaciones inversas las cuales ocurren conforme la bobina se mantenga alimentada por más tiempo, lo cual en esta aplicación en altamente plausible que

ocurra se puede considerar a los otros 2 conceptos mucho mejores que la referencia en este criterio siendo ambos clasificados con 5.

El criterio de implementación se basa en con que facilidad se puede incluir este circuito al PCB y cuantos dispositivos incluye, es por ello por lo que siendo el concepto 1 un circuito más simple se considera mejor en este criterio, por otro lado, entre el concepto 2 y 3 este último al incluir una señal de control extra y agregar un transistor se le califica con 1 a diferencia del concepto 2 que se califica únicamente como peor que la referencia.

El criterio 3 puede ser considerado más estable ya que separa las señales de alimentación de la bobina y de control del circuito, mientras que el concepto 2 las mantiene como la misma señal, ambas inicialmente se consideran más estables que el concepto 1 ya que tienen mejor control y disminución de ruido, sin embargo, como se menciona en [31] esta estabilidad es deseable cuando se controlan cargas como motores, en este caso al ser un relé la importancia del criterio disminuye.

Finalmente seleccionando el concepto 2 según los resultados de calificación de la tabla 5.10, además para el diodo se seleccionará un flyback SM4007PL-TP el cual soporta una corriente máxima de 30A la cual supera a la corriente de la bobina por lo que el factor de seguridad de protección será mucho mayor que 1 y en paralelo a la bobina se colocará un LED (DIP).

5.3 Conceptos para el diseño del accesorio de resguardo.

La carcasa como sistema de resguardo de los circuitos de conmutación debe contar con características de diseño que proporcionen una larga vida útil de la carcasa, faciliten la manipulación para su mantenimiento y aseguren una protección fiable a los circuitos

5.3.1 Estructura base de la carcasa

El diseño de la estructura y forma de la carcasa se realizará tomando en cuenta la forma y configuración en el acomodo de los PCB, así también como en el posicionamiento y orientación de los pines de conexión. Debido a que los PCB son cuadrados y de dimensiones similares entre los 100 y 150 mm de ancho y largo se propone una distribución como la observada en la figura 5.11, en la cual estos se acomodan en 2 hileras con el objetivo de distribuir los elementos para que las

dimensiones de anchura y longitud mantengan proporciones dimensionales aceptables.

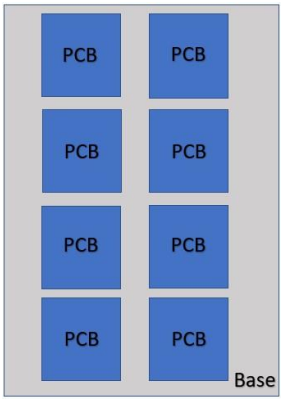


Figura 5.11. Acomodo de circuitos en base de carcasa. Fuente: Propia

Teniendo esto en cuenta se proponen 3 conceptos sobre el diseño base de la estructura, todos ellos con forma de prisma rectangular, es importante denotar que estos conceptos definen la cantidad de partes y el acomodo de las entradas, detalles de mayor complejidad como cejillas de sellado y acabados de diseño no se ven representados.

Como primer concepto tenemos la estructura observada en la figura 5.12, acá podemos denotar que está compuesta de 2 partes, ambas partes de la estructura pueden ser manufacturadas por medio de tecnología de impresión 3D.

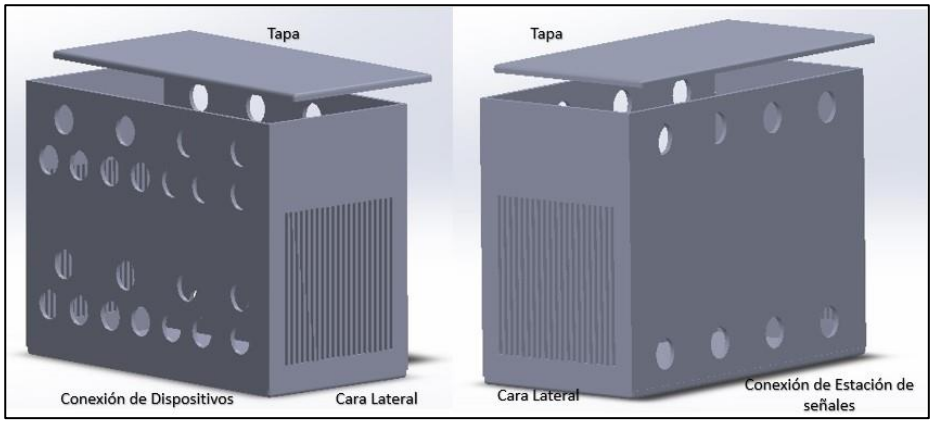


Figura 5.12. Concepto de ensamble de tipo 1. Fuente: Propia

Por un lado, tenemos la tapa la cual tiene como función el sellado de la estructura, fácil de retirar y a través de la apertura superior permite acceder a todo el espacio interno en la carcasa, la otra parte está compuesta por el cuerpo de la

carcasa unida a la base de montura de los circuitos impresos, esta parte tiene como características:

- En su cara frontal tiene dispuestos los agujeros en los cuales se colocarán los conectores de los distintos dispositivos y las entradas de las señales de control.
- En la cara trasera tiene dispuestos los agujeros en los cuales se realizarán las conexiones con la estación de señales y la caja maestra.
- Las conexiones se realizan en las caras verticales y los circuitos impresos de acomodan internamente de manera horizontal.
- La base y cuerpo de la carcasa se encuentran unidas como una sola pieza, esta estructura debe ser manufacturada en 2 partes.
- En sus costados laterales encontramos las aperturas que funcionaran como ventilación.

Para el concepto 2 se propone un diseño en 3 partes como se observa en la figura 5.13, en el cual las conexiones se realizan por medio del cuerpo de la carcasa, este concepto permite manipular la base de montura de los circuitos impresos de manera ajena al resto de la estructura, todas las partes pueden ser diseñadas utilizando tecnología de impresión 3D.

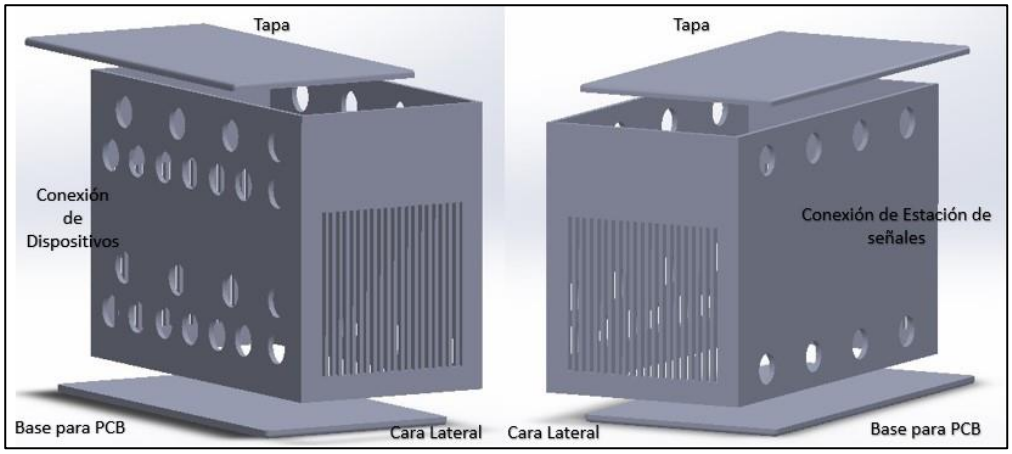


Figura 5.13. Concepto de ensamble de tipo 2. Fuente: Propia

Las características más destacables de este concepto son:

- Los agujeros de conexión con los múltiples dispositivos se encuentran dispuestos en toda la cara frontal, incluyendo las entradas de las señales de control.

- Los agujeros de conexión con la estación de señales y la caja maestra se encuentran dispuestos en la cara trasera.
- Las conexiones de realizan a través de las caras verticales y los circuitos impresos están dispuestos horizontalmente en la base.
- La tapa y la base son elementos separados del cuerpo de la carcasa, ambas partes se unen al cuerpo de la carcasa por medio de cejillas.
- El cuerpo de la carcasa presenta aperturas en dirección vertical en sus caras laterales que funcionan como agujeros de ventilación.

Para el concepto 3 se propone un diseño en 5 partes como se observa en la figura 5.14, este permite manipular la base de montura de los circuitos impresos de manera ajena al resto de la estructura, permite una manipulación interna de la estructura sin necesidad de retirar la tapa y la base, además todas las partes pueden ser manufacturadas por medio de tecnología de impresión 3D.

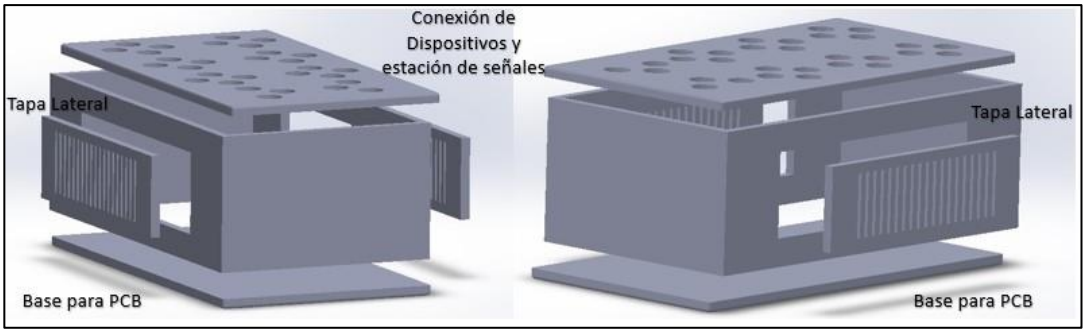


Figura 5.14. Concepto de ensamble de tipo 3. Fuente: Propia

Las características más destacables de este concepto son:

- Todos los agujeros de conexión con los dispositivos y la estación de señales se encuentran en la tapa superior de la carcasa.
- Las conexiones de realizan a través de la cara horizontal superior y los circuitos impresos se distribuyen de manera horizontal en la base de montura.
- La tapa y la base son elementos separados del cuerpo de la carcasa, ambas partes se unen al cuerpo de la carcasa por medio de cejillas.
- Las caras laterales frontal y trasera presentan unas tapas con agujeros para ventilación que se encuentran fijadas al cuerpo de la carcasa por cejillas y pueden ser fácilmente retirables.

- El cuerpo de la carcasa presenta aperturas en dirección vertical en sus caras laterales que funcionan como agujeros de ventilación.

Para la selección del mejor concepto se utilizó la matriz de selección de la tabla 5.11.

Tabla 5.11 Matriz de selección de concepto para ensamble. Fuente: Propia

		Dispositivos					
		Concepto 1		Concepto 2		Concepto 3	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Manufactura	30%	3	0.9	4	1.2	5	1.5
Mantenimiento	15%	3	0.45	4	0.6	5	0.75
Robustez	15%	3	0.45	3	0.45	3	0.45
Transportable	5%	3	0.15	3	0.15	3	0.15
Desensamble	25%	3	0.75	5	1.25	3	0.75
Facilidad de diseño	10%	3	0.3	2	0.2	2	0.2
Total		3.00		3.85		3.80	

Para el criterio de manufactura se toma en consideración que esta será realizada por medio de tecnología de impresión 3D de tipo aditivo, el concepto 3 por la disposición de los agujeros en sus distintas partes brinda mayor facilidad de manufactura en las piezas siendo calificado con 5, mientras que el concepto 2 al separar la base y la tapa del cuerpo de la carcasa presenta una mejoría respecto a la base en la calidad de la manufactura siento calificada con 4.

Tanto el concepto 2 y 3 presentan piezas que individualmente son más fáciles de mantener o sustituir en caso de que sean dañadas, por ello se califican con 4 y 5 respectivamente.

En los criterios de robustez y facilidad de ser transportada todos los conceptos por su diseño morfológico similar tiene la misma valoración por lo cual se calificarán todos con un 3.

El criterio de desensamble se refiere a la facilidad con que los circuitos impresos pueden ser manipulados, desde las distintas aristas del diseño, para el concepto 2 este únicamente requiere que la tapa superior sea removida para poder acceder a su interior además que al tener la facilidad de poder retirar la base se le califica con un 4, por otro lado por la disposición de los agujeros en el concepto 3 para poder remover la tapa superior primero se deben desconectar las señales utilizando los agujeros laterales lo cual vuelve el proceso complejo por ello se califica con 3.

El criterio en la facilidad de diseño contempla la cantidad de detalles que lleva cada estructura y que deben ser implementador en el diseño, debido a que los conceptos 2 y 3 involucran más partes, requieren de más cejillas y detalles para el cierre y sellado de los componentes, por ello se califican con 2.

Finalmente, según los resultados de la evaluación de criterios se selecciona el concepto 2 como el mejor concepto a ser implementado.

5.3.2 Sujeción de circuitos PCB

Para sujetar placas de circuitos impresos en la mayoría de los diseños comerciales vemos primordialmente 2 tipos, por pestañas en las cuales por medio de la fricción los circuitos PCB quedan comprimidos y sujetos a la carcasa o por tornillería, debido a los requerimientos se seleccionará la segunda opción.

Para el diseño del elemento de sujeción con tornillos se proponen 2 conceptos.

El concepto 1, como se observa en la figura 5.15 se trata de un cilindro hueco el cual por medio de la impresión 3D en el momento de la manufactura o por medio de un proceso consecuente a la impresión 3D utilizando una terraja macho, se le realiza en el material de impresión la rosca en la cual el tornillo se acopla directamente.

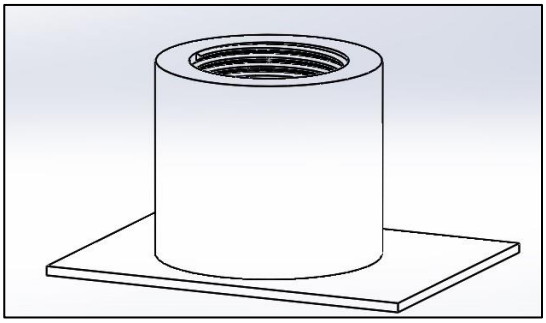


Figura 5.15. Concepto de sujeción de circuito tipo 1. Fuente: Propia

Como concepto 2, que se observa en la figura 5.16, propone un cilindro hueco diseñado con tecnológica de impresión 3D, en el cual se le realiza un añadido de un inserto con rosca interna, este inserto de acopla al cilindro utilizando calor y presenta unas estrías que por medio de la fricción y un ajuste tallado quedan fijados y sobre la rosca interna de estos insertos se acopla el tornillo.

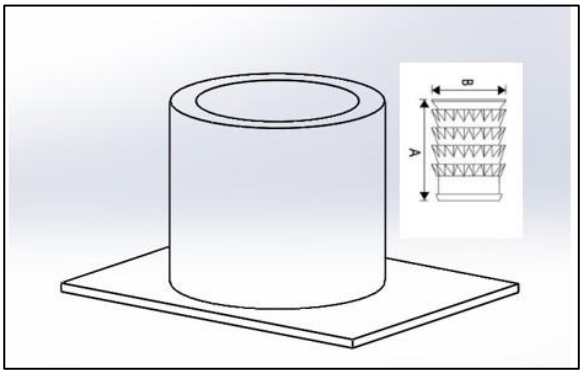


Figura 5.16. Concepto de circuito de sujeción tipo 2. Fuente: Propia

Para la selección de los conceptos se utilizó la matriz observada en la tabla 5.12.

Tabla 5.12. Matriz de selección de concepto para sujeción de circuito. Fuente: Propia

		Conceptos			
		Tipo 1		Tipo 2	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Facilidad de manufactura	30%	3	0.9	5	1.5
Robustez	35%	3	1.05	5	1.75

Facilidad de implementación	**15%**	**3**	**0.45**	**2**	**0.3**
Facilidad de Mantenimiento	**20%**	**3**	**0.6**	**5**	**1**
Total		**3.00**		**4.55**	

El primer criterio de selección es la facilidad de manufactura, tomando en cuenta que esta se realizará por medio de impresión 3D aditiva, generar una rosca interna como se define en el concepto 1 requiere de una precisión que genera mucha más complejidad en comparación al concepto 2 el cual solo es un cilindro hueco y liso, además que el añadido del inserto se realiza en menos de 1 minuto sin requerir de ningún equipo especial por lo cual se califica con 5.

Para el criterio 2 se considera que las bajas resistencias de los plásticos impresos no son muy fiables específicamente en su resistencia y durabilidad para las roscas, en comparación a los insertos que son de materiales como aluminio o latón los cuales presentaran menos desgaste y una vida útil más larga, por ello el concepto 2 se califica con 5.

En su implementación el concepto 2 requiere de más trabajo al requerir un proceso extra de manufacturación es por ello por lo que en este criterio se califica con 2.

Para el mantenimiento de elemento de sujeción, en el concepto 1 en caso de que la rosca se dañe no hay manera de cambiarla y se tendría que rehacer la pieza completa, mientras que para el concepto 2 por medio de calor los insertos pueden ser retirados, esto permite mayor flexibilidad en el mantenimiento siento calificado con un 2. Basados en los resultados de la matriz de calificación finalmente el concepto 2 fue seleccionado.

5.3.3 Cejillas para cierre de carcasa

El cierre de la carcasa se implementará a través del diseño de cejillas para la unión de la tapa y la base con el cuerpo de la carcasa, los 2 conceptos propuestos están enfocados en facilitar el proceso de cierre y apertura de la carcasa.

Como concepto 1 se propone una cejilla que va unida a la parte lateral de la tapa y la base, esta tiene forma de “C” como se puede observar en la figura 5.17, esta cejilla se ajusta en el cuerpo de la carcasa uniéndose a su elemento contraparte

en el cuerpo de la carcasa de manera que el movimiento de esta quede fijado, esta cejilla permite que la tapa se retire cuando se genera una presión sobre la misma de forma perpendicular a la “C” de manera que se flecte y se separe la unión.

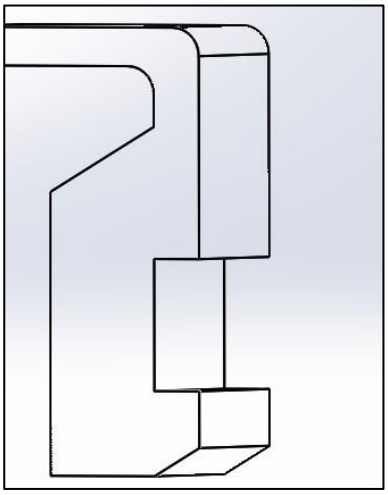


Figura 5.17. Concepto de cejilla de sujeción de tipo 1. Fuente: Propia

Como concepto 2 se propone una cejilla que como se observa en la figura 5.18, va unida a la sección horizontal de la tapa y la base, esta presenta una forma triangular la cual se une a su contraparte en el cuerpo de la carcasa para realizar el cierre y sellado, con este concepto la apertura de la carcasa se realiza generando una fuerza perpendicular al largo de la tapa o base generando que por la forma que presenta la cejilla esta se deslice sobre su contraparte y se separe la unión.

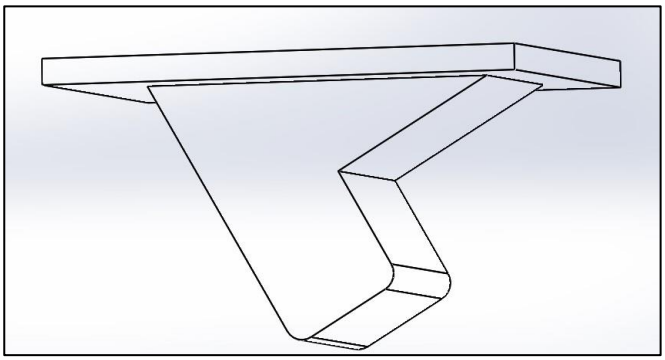


Figura 5.18. Concepto de cejilla de sujeción de tipo 2. Fuente: Propia

Para la selección de los conceptos se utilizó la matriz de la tabla 5.13.

Tabla 5.13 Matriz de selección de concepto para cejilla de cierre. Fuente: Propia.

		Conceptos			
		Tipo 1		Tipo 2	
Criterio de Selección	Peso	Calificación	Evaluación Ponderada	Calificación	Evaluación Ponderada
Facilidad de manufactura	30%	3	0.9	4	1.2
Robustez	35%	3	1.05	5	1.75
Facilidad de implementación	15%	3	0.45	3	0.3
Facilidad de Diseño	20%	3	0.6	4	0.8
Total		3.00		4.05	

El criterio en facilidad de manufactura lo gana el concepto 2 ya que por su forma y posicionamiento respecto a la tapa es más sencillo de ser desarrollado mediante la tecnológica de impresión 3D aditiva ya que requiere menos material de apoyo y las formas son menos complejas, también se debe considerar que los ángulos no superan los 45 grados lo que significa menos material de aporte, por ello el concepto 2 se califica con 4.

En la impresión 3D, los elementos manufacturados son más frágiles cuando ante fuerzas perpendiculares a la dirección de adición de material, por el tipo de implementación del concepto 1 esto lo afecta, además el concepto 2 tiene más material por área de esfuerzo lo que presenta una mayor robustez siendo así calificado con un 5. Mientras que en el criterio sobre facilidad de implementación ambos presentan un nivel de complejidad similar por lo cual se califican ambos con 3.

Como se observa en la figura 5.17, el concepto 1 presenta formas más complejas en su diseño, por la cantidad de ángulos presentes y secciones a 90 grados que generan puntos críticos en su diseño, además el concepto 1 requiere que la cejilla quede en voladizo respecto a la cara lateral de la pieza (tapa o base) lo que aumenta el nivel de complejidad de estas. Mientras que la forma del concepto 2 es mucho más simple siendo básicamente un triángulo menos una sección a pesar

de que su contraparte también requiere una sección en voladizo esta seria interna en el cuerpo de la carcasa dándole mayor seguridad para evitar fracturas, por lo tanto, el concepto 2 se califica con un 4, finalmente siendo este el seleccionado.

6. Desarrollo, integración y validación de conceptos

6.1 Diseño de Software de evaluación

Como se menciona en la sección 5.1.1 el paradigma para la implementación del código será modular, dicho esto inicialmente se plantea comenzar con la inicialización de las variables globales, las cuales nos permitirán enlazar información entre los distintos módulos del código, las variables globales creadas son aquellas que por medio de un análisis previo de la estructura de evaluación se considera que serán utilizadas por distintos módulos o es beneficioso incluirlas dentro del bloque de variables globales para futuras implementaciones.

Las de tipo booleano se utilizan para de llevar control sobre el inicio de sesión dentro del módulo PXI, este inicio de sesión es requerido para controlar los relés de los conectores, la importancia de llevar control sobre el inicio de la sesión es que esta debe cerrarse siempre al finalizar el uso de los módulos y en caso de un error dentro del código se debe conocer el estado de la sesión, las otras variables de tipo booleano serán utilizadas para controlar las banderas seleccionadas por el usuario y caracterizar el flujo de evaluación que deberá tener el software.

Las variables de tipo string contendrán información sobre los elementos que el software deberá evaluar tales como direcciones IP de las maquinas a las cuales se debe tener acceso, nombre de usuario y contraseña para la conexión ssh, nombre del módulo para el inicio de sesión del PXI y también contiene información utilizada para la creación del archivo de registro de eventos tales como la dirección donde se guardara el archivo y nombre del archivo.

Las listas globales servirán para llevar control durante la evaluación de los conectores, registrando cuales conectores se deben evaluar según lo caracterice el usuario, almacenando los conectores o relés que presenten fallas para poder ser registrados dentro de los resultados al final de la evaluación. Y los diccionarios globales permitirán invocar valores asociados a llaves fácilmente interpretables dentro del código.

6.1.1 Función principal del código

En el diagrama de flujo de la función “main” en la figura 6.1, se observa la función principal del código del software de evaluación, se realizó tomando en cuenta las características de implementación de los conceptos seleccionados para el paradigma de programación, registro de eventos y resultados y la lectura de parámetros además de los requerimientos dispuestos en la sección 4.1, acá se denota el alcance del programa de evaluación, el cual se encargará principalmente de probar el funcionamiento y evaluar el estado de los relés del módulo PXI y realizar verificaciones de conexión y comunicación a los demás elementos.

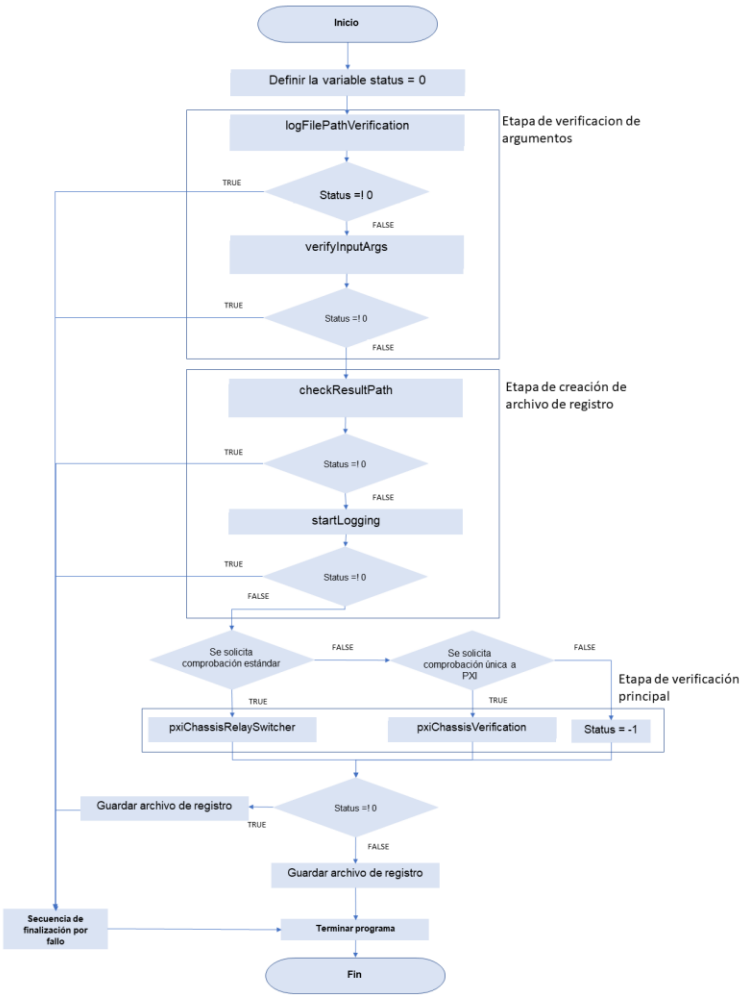


Figura 6.1. Diagrama de la función main. Fuente: Propia

La etapa 1 “verificación de argumentos” define todas las características posteriores que deberá tener el proceso de evaluación, luego la etapa 2 “creación

del archivo de registro" utilizando el concepto seleccionado en la sección 5.1.2, configura y crea los módulos y el archivo para el registro de eventos y resultados, más adelante la etapa 3 "verificación principal" realiza la evaluación de los 4 dispositivos incluidos en el requerimiento 1 de la tabla 4.1.

Cada una de estas etapas se realizarán en funciones separadas respetando la modularidad del código, finalmente luego de que las evaluaciones concluyan se guardará el archivo de resultados y se procederá con la finalización de la evaluación.

Para cada una de las funciones implementadas se llevara control de una variable denominada "status" esta variable tendrá 2 valores enteros, 0 en caso de que la función sea ejecutada sin ningún problema y -1 en caso de que al ejecutar la función se produzca un error, al finalizar cada etapa del programa principal se corroborará el valor de esta variable, solo en caso de que la variable indique algún error el programa procederá a ejecutar la secuencia de finalización debido a fallo de otra manera el flujo del programa procederá como se describió sin ser afectado.

6.1.2 Etapa de verificación de argumentos

Esta etapa se realizará tomando en cuenta el concepto seleccionado en la sección 5.1.3.

La instrucción en consola para la ejecución del programa definirá los argumentos de entrada como se puede observar en la tabla 6.1, se debe tener en cuenta que el nombre del programa es "post.py" y los conectores se denominarán igual a las clases del paquete RTMD en la sección 2.3.1.

Tabla 6.1 Ejemplo de instrucciones para ejecución del programa. Fuente: Propia

Instrucción en consola	Argumentos
./post.py -v -r /home/user/Desktop	[-v, -r, /home/user/Desktop]
./post.py -v A_OUT -r /home/user/Desktop	[-v, A_OUT, -r, /home/user/Desktop]
./post.py A_OUT M_IN -r /home/user/Desktop	[A_OUT, M_IN, -r, /home/user/Desktop]

En la etapa "verificación de argumentos" el bloque "logFilePathVerification" invoca la función de mismo nombre que evaluará si es requerido modificar la dirección del archivo de registro buscando la bandera "-r" en la lista de argumentos

de entrada, al finalizar retornará los demás argumentos y el valor de “status”, esto se observa la figura 6.2.

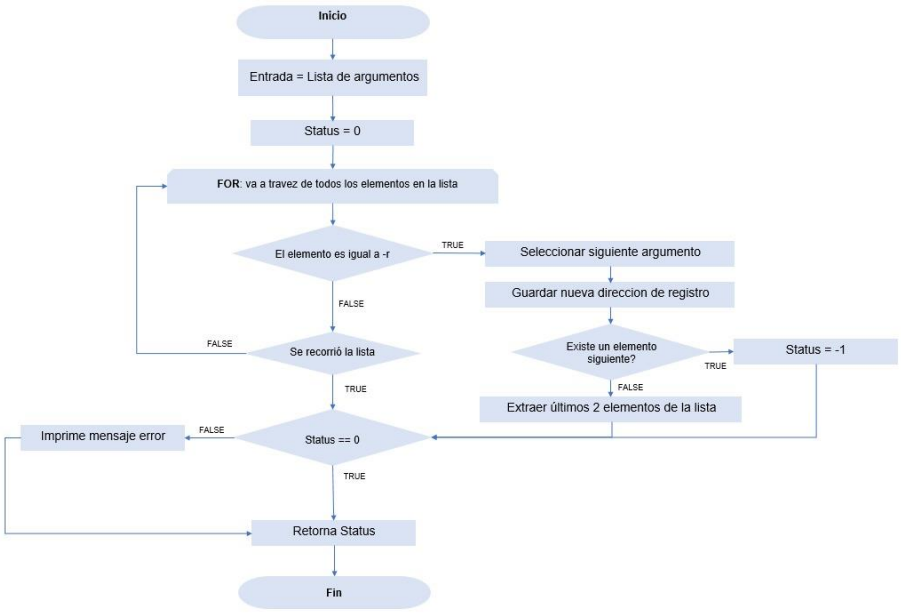


Figura 6.2. Diagrama de flujo de función logFilePathVerification. Fuente: Propia

En la etapa “verificación de argumentos” el bloque “verifyInputArgs” verificará los demás argumentos como se muestra en la figura 6.3, este bloque caracterizará el flujo de evaluación que tendrá el programa, modificando las banderas de tipo booleano e invoca a la función “connectorVerification”.

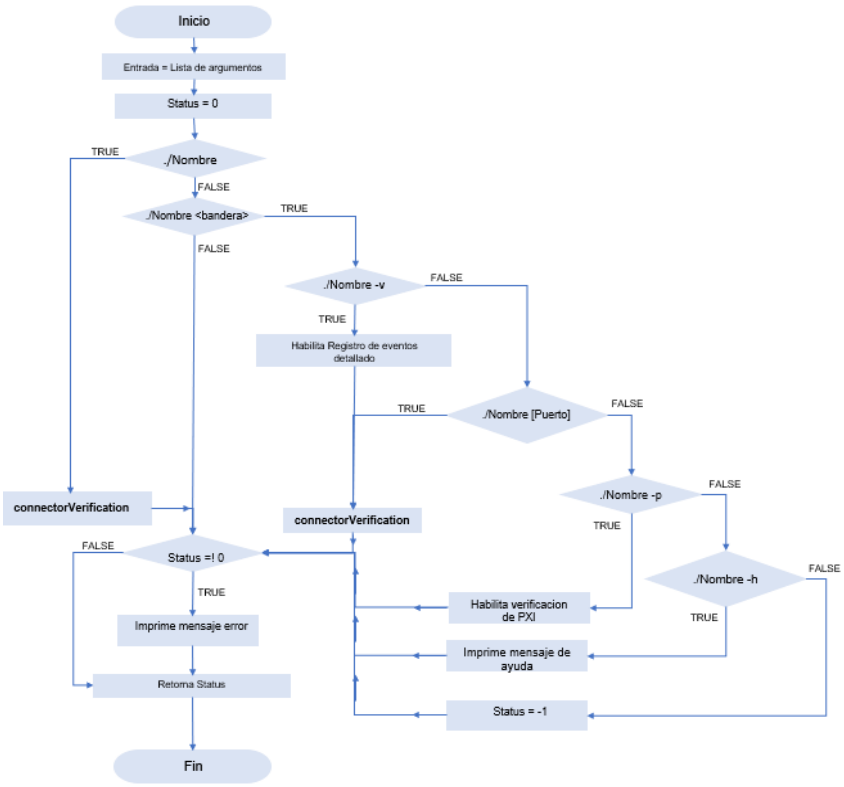


Figura 6.3. Diagrama de flujo de función verifyInputArgs Fuente: Propia

La función “connectorVerification” se encargará de validar los conectores ingresados por el usuario como se observa en la figura 6.4, en caso de que la lista de entrada este vacía se evaluarán todos los conectores. Los conectores repetidos se desecharán esto asegurará que no se evalué 2 veces el mismo conector, en caso de que alguno de los conectores digitados no se encuentre dentro de los disponibles, se registrará esto como un error en los argumentos de entrada y se procederá a retornar -1.

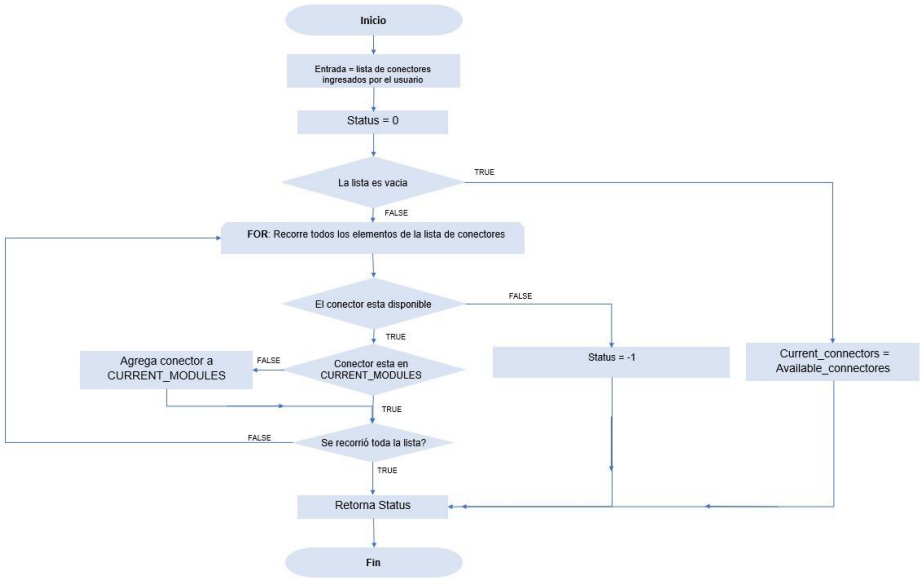


Figura 6.4. Diagrama de flujo de función connectorVerification. Fuente: Propia

6.1.3 Etapa de creación de archivo de registro

La segunda etapa del diagrama de la figura 6.1 toma en cuenta el concepto escogido en la sección 5.1.2, el bloque “checkResultPath” corroborará la existencia de la dirección en la cual se deberá guardar el archivo de registro de eventos, en caso de que no exista se retornara el “status” igual a -1.

La caracterización y creación del archivo de registro se hará en el bloque “startLogging” por medio de los métodos mostrados en la tabla 5.2, siguiendo la lógica del ejemplo mostrado en [21]. No requiere de un manejo de error ya que los parámetros están previamente definidos en el código o se han verificado en otras funciones.

6.1.4 Bloque de evaluación estándar

En la etapa “verificación principal” de la figura 6.1 el bloque “pxiChassisRelaySwitcher” incluye las funciones que serán parte de la evaluación estándar, como se muestra en la figura 6.5, dentro de las funciones de este bloque se aplicarán los conceptos seleccionados en las secciones 5.1.4 y 5.1.5. En este bloque se usarán los relés del módulo PXI por lo tanto se realizará inicialmente la verificación del PXI Chasis en el bloque “pxiChassisVerification” que se explica con detalle en la sección 6.1.6.

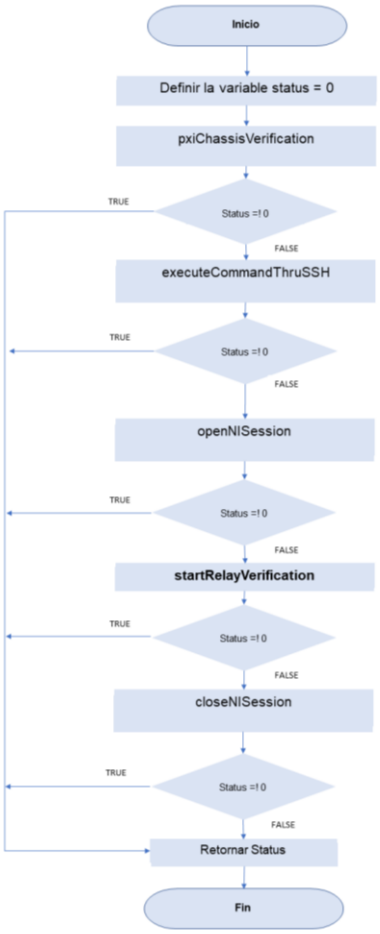


Figura 6.5. Diagrama de flujo del bloque pxiChassisRelaySwitcher . Fuente: Propia

El bloque “executeCommandThruSSH” de la figura 6.5 seguirá el flujo observado en la figura 6.6, utilizará el método “pxssh” escogido en la sección 5.1.4 para inicializar y validar la conexión con los dispositivos.

Todo el bloque de tendrá un manejo de error con el concepto seleccionado en la sección 5.1.5 de manera que, si ocurre alguna excepción, se registrará el mensaje de error y se retornará -1.

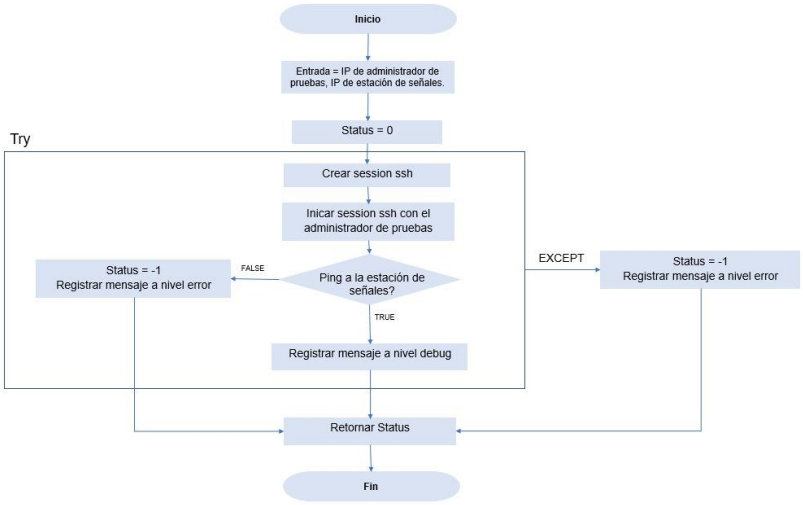


Figura 6.6. Diagrama de flujo de la función executeCommandThruSSH . Fuente: Propia

Los bloques “openNISession” y “closeNISession” de la figura 6.5 se encargarán de iniciar y cerrar sesión con el PXI respectivamente, ambas incorporarán manejo de error ya que utilizan funciones del paquete RTMD explicado en la sección 2.3.1. Ambas registraran un mensaje con el resultado del proceso y el bloque “openNISession” marcará de forma global el éxito de la conexión mediante una variable global.

6.1.5 Función para la verificación de conectores

El bloque “startRelayVerification” de la figura 6.5 verificará de manera secuencial los relés asociados a cada conector, al inicio registrará un mensaje de nivel “Info” con todos los conectores que se evaluarán y al final se retornara el valor de “status” para indicar el éxito de la verificación, como se observa en la figura 6.7.

Esta función llevara un registro de los relés que fallaron y el conector al cual están asociados de manera que al final del proceso de evaluación se registraran los resultados de cada uno de ellos en 2 secciones separadas para fallos y éxitos.

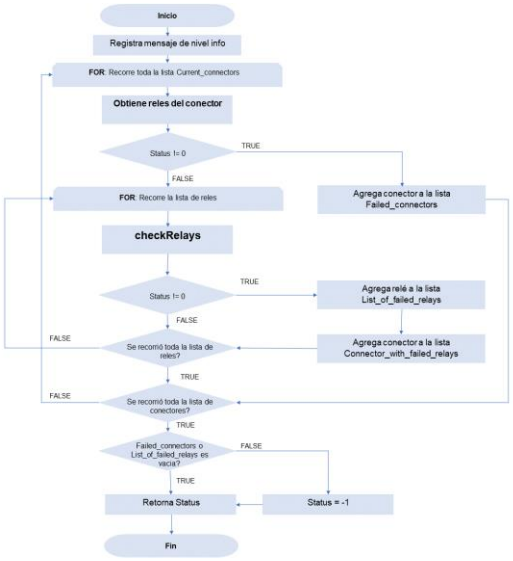


Figura 6.7. Diagrama de flujo de función startRelayVerification . Fuente: Propia

El bloque “checkRelays” de la figura 6.7 verificará la conmutación del relé siguiendo el flujo de la figura 6.8, inicialmente registrara un mensaje de advertencia si el estado del relé no es el estándar. En caso de que no cambie el estado, se registrara un mensaje de nivel error y se retornará un “status” igual a -1 indicando al programa que no es posible conmutar dicho relé.

Al final del proceso se registrarán todos los resultados obtenidos dentro del archivo a un nivel “info” y se incluirá una sección con los estados estándar de todos los conectores evaluados, con el objetivo que el usuario los tenga presentes de manera explícita en caso de no conocerlos.

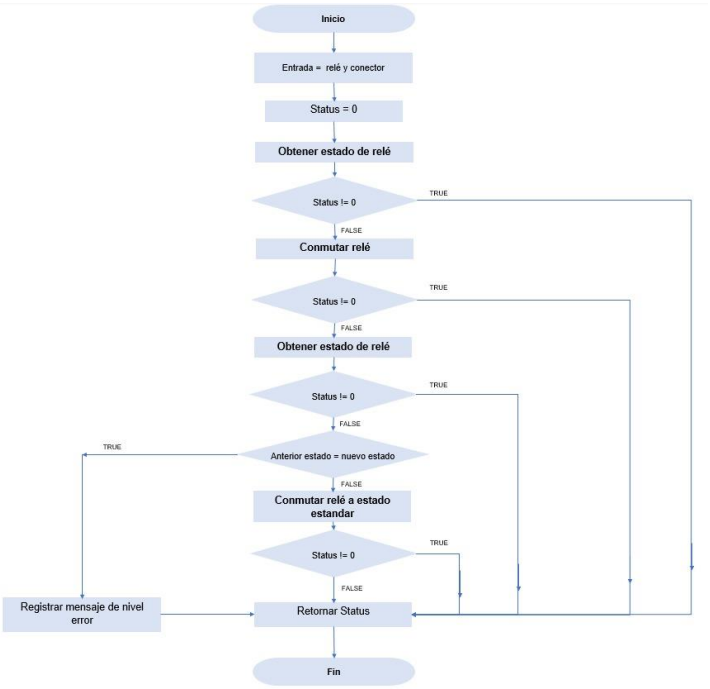


Figura 6.8. . Diagrama de flujo de la función checkRelays. Fuente: propia

6.1.6 Verificación del PXI Chassis

Esta sección pertenece al bloque “pxiChassisVerification” observado en la figura 6.1. La librería Ni_switch contiene la API Session como se explica en la sección 4.2.2, haciendo uso de esta se ejecutará el método “_self.test” el cual nos retornará un valore entero indicando si está habilitada la comunicación entre el driver del PXI y sus módulos, en caso de que esta conexión no se encuentre habilitada se retornará un valor de “status” -1 que le indicará al programa que el control del módulo se encuentra en estado no operativo. En caso de que la comunicación no presente problema se utilizaran distintos métodos de la API Session para obtener información de importancia del PXI como, por ejemplo:

- Nombre del dispositivo
- Modelo del dispositivo
- Descripción del controlador
- Numero serial del controlador.
- Numero de relés disponibles en el módulo.

Esta información se registrará a nivel “info” dentro del archivo en una sección especifica de manera que se encuentre explicita para el usuario. Todo el proceso

se encuentra dentro de un bloque “try” que en caso de que falle la ejecución de la API se registrara un mensaje de nivel error y se retornará un “status” igual a -1 indicando que ocurrió un problema.

6.2 Diseño de circuitos impresos

En esta sección veremos la implementación de los conceptos seleccionados en la sección 5.2 para el desarrollo de múltiples circuitos de conmutación, basándonos en el alcance de este proyecto se diseñarán circuitos de conmutación en placas PCB para 7 puertos distintos y un octavo circuito para las señales provenientes del generador de ablación. Estos circuitos de conmutación se dividen en:

- Puertos de entrada: Este circuito de conmutación será replicado para los 3 puertos de entrada M, A y B, cada puerto de entrada conmutará entre 3 dispositivos.
- Puertos de salida: Este circuito de conmutación será replicado para los 3 puertos de salida M, A y B, cada puerto de salida conmutará entre 3 dispositivos.
- Conexión Maestro a estación de señales: Este circuito de conmutación se utilizará para conmutar la conexión a la estación de señales entre 2 dispositivos distintos, la caja Maestra y la caja Maestra SP.
- Conexión generador de ablación a maestro: Este circuito de conmutación se utilizará para conmutar la conexión al generador de ablación de 2 dispositivos distintos, la caja Maestra y la caja Maestra SP.

6.2.1 Puertos de entrada

El circuito de conmutación desarrollado para los puertos de entrada conmutará entre 3 dispositivos distintos en cada puerto, los puertos de entrada como se describe en la sección 2.1.2 se utilizan para la conexión de múltiples catéteres estos se pueden conectar de manera directa o utilizando la caja de conexión mostrada en la sección 2.1.2.3.

En la tabla 6.2 podemos observar las señales de los 3 dispositivos distintos que se deben conectar a los puertos M, A y B de entrada, estas señales son

nombradas así en la documentación técnica del hardware [11]. Estos dispositivos pueden ser 2 tipos “Intellamap Orion Catheter” o Caja de conexión (ver sección 2.1.2.3) . Estas señales tienen un límite máximo de tensión de 5V y límite máximo de corriente de 1A. A excepción de las señales 1 y 2 las cuales presentan frecuencias máximas de 100KHz, las demás señales son continuas.

Tabla 6.2. Señales de los puertos de entrada de la estación de señales. Fuente: Propia.

Señal	Dispositivo A	Dispositivo B	Dispositivo C
1	ID 1	ID 1	ID 1
2	ID 2	ID 2	ID 2
3	A	IC21	IC21
4	B	IC22	IC22
5	C	IC23	IC23
6	D	IC24	IC24
7	Nav 1A	IC25	IC25
8	Nav 1B	IC26	IC26
9	Nav 2A	IC27	IC27
10	Nav 2B	IC28	IC28

Para este circuito de conmutación se utilizarán 5 grupos de 3 relés DPDT en cascada conectados de manera tal que cuando estos se encuentren en estado no conmutado se conecte el puerto de la estación de señales a tierra para simular la desconexión de cualquier dispositivo, en cada nivel de grupos de relés se conectaran 2 señales del mismo dispositivo, la conexión de cada dispositivo será controlado por medio de 3 señales provenientes del PXI Chassis, cada señal de control cuando se encuentre en alto conmutará un nivel del relés distinto siguiendo la lógica observada en la tabla 6.3, como se puede observar en esta la lógica de control presenta 4 casos no contemplados ya que estos serán filtrados por medio del código de control del PXI Chassis.

Para las entradas y salidas se utilizarán conectores AMP CT con resistencia de hasta 2.5 Amperios de conducción, los conectores utilizados serán de 2 terminales ya que por el acomodo de los pines que cada relé DPDT esto facilitara el posicionamiento del conector en la placa simplificando de esta manera las líneas

de conducción en el PCB. En la figura 6.9 podemos observar una versión simplificada del esquemático de conexión en la cual se muestran 3 grupos de relés DPDT en cascada, para todo el circuito se colocarán únicamente 3 leds uno por cada señal de control, estos no irán directamente soldados a la placa, sino que estarán conectador por medio de 2 terminales para así poder colocarlo de manera que sean visibles desde el exterior de la carcasa.

Tabla 6.3 Señales de control para conexión a los puertos de entrada de la estación de señales. Fuente: Propia

Control A	Control B	Control C	Conecta
Abierto	Abierto	Abierto	N/A
Abierto	Abierto	Cerrado	Dispositivo C
Abierto	Cerrado	Abierto	Dispositivo B
Cerrado	Abierto	Abierto	Dispositivo A
Casos no aplicables			
Abierto	Cerrado	Cerrado	Dispositivo C
Cerrado	Abierto	Cerrado	Dispositivo C
Cerrado	Cerrado	Abierto	Dispositivo B
Cerrado	Cerrado	Cerrado	Dispositivo C

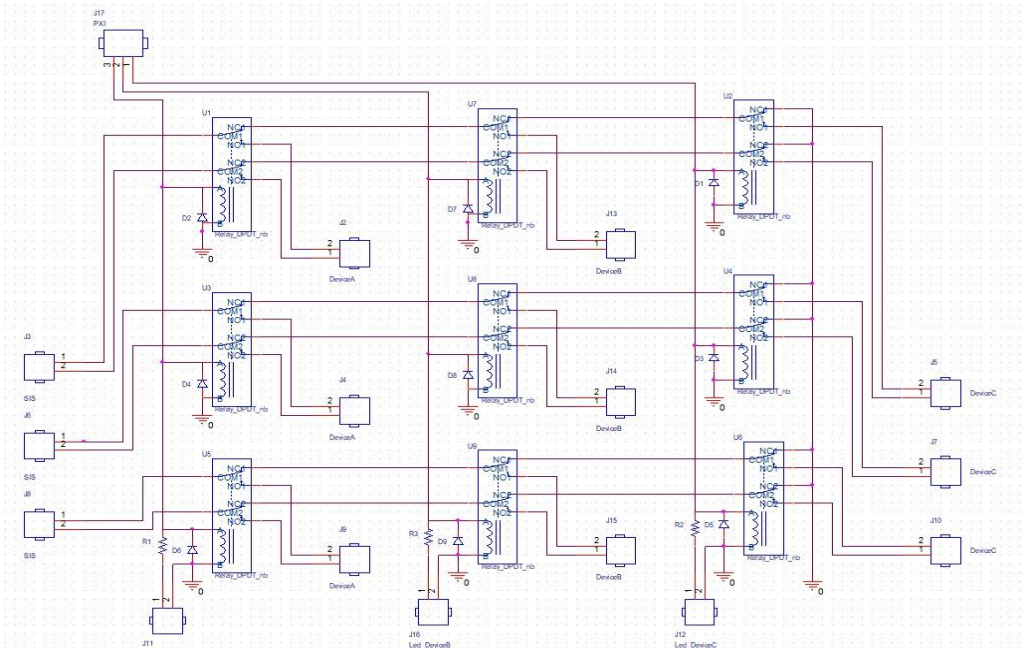


Figura 6.9. Esquemático simplificado de conexión a los puertos de entrada. Fuente: Propia

6.2.2 Puertos de salida

El circuito desarrollado para los puertos de salida conmutara las señales provenientes de la estación de señales, hacia 3 dispositivos distintos, los puertos de salida trasmiten las mismas señales recibidas en los puertos de entrada como se puede observar en la tabla 6.4 por lo cual para el alcance del sistema de automatización se utilizarán únicamente 10 canales de transmisión. Al ser las mismas señales que las de entrada estas presentan las mismas características de tensión, amperaje y frecuencia.

Tabla 6.4. Señales de los puertos de salida de la estación de señales. Fuente: Propia

Señal	Dispositivo A	Dispositivo B	Dispositivo C
1	ID 1	ID 1	ID 1
2	ID 2	ID 2	A/B1
3	Orion1	IC1	A/B2
4	Orion2	IC2	A/B3
5	Orion3	IC3	A/B4
6	Orion4	IC4	A/B5
7	Orion5	IC5	A/B6
8	Orion6	IC6	A/B7
9	Orion7	IC7	A/B8
10	Orion8	IC8	A/B9

En la figura 6.10 podemos observar una versión simplificada del esquemático de conexión en la cual se muestra 3 grupos de 3 relés DPDT, teniendo en cuenta la similitud de las señales y que como se especifica en la hoja de datos del relé [32] la transmisión funciona en ambas vías podemos concluir que este circuito presenta la misma forma que el desarrollado en la sección 6.2.1, es por ello que para los puertos de salida se utilizará el mismo diseño de placa únicamente cambiando las entradas y salidas en su implementación.

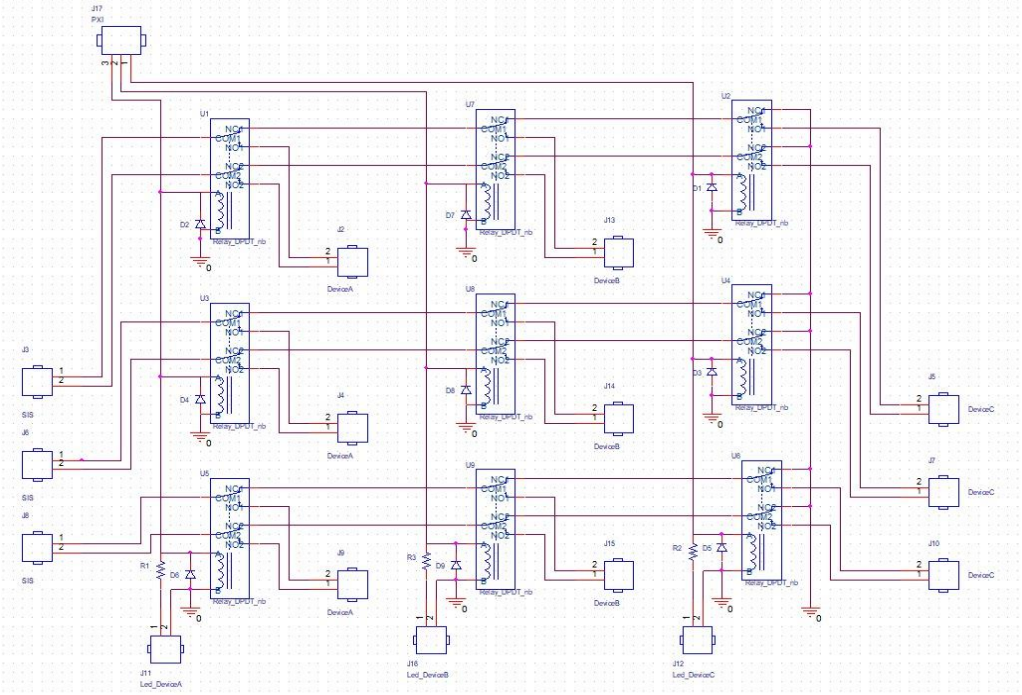


Figura 6.10. Esquemático simplificado de circuito de conexión a los puertos de salida. Fuente: Propia

6.2.3 Conexión Maestro-Estación de señales

La funcionalidad de este circuito es conectar la salida de 2 maestros de ablación (ver sección 2.1.2.4) distintos a un mismo puerto en la estación de señales, los maestros que se deben conectar se dividen en Maestro Intellanav y Maestro SP, y transmiten las señales observadas en la tabla 6.5 en la cual se detallan las características más relevantes de cada señal.

Tabla 6.5. Señales de la caja Maestro de ablación hacia el puerto de ablación. Fuente: Propia

Señal	Maestro Intellanav	Maestro SP	Característica
1	ABL ID	SP ID	100 kHz
2	ABL ID	SP ID	100 kHz
3	ABL1	R1	Tensión 5V Max Corriente 1A Max
4	ABL2	R2	
5	ABL3	R3	
6	ABL4	SP1	
7	ABL5	SP2	

8	ABL6	SP3	
9	ABL Sensor 1A	ABL Sensor 1A	Corriente 1.5A Max
10	ABL Sensor 2A	ABL Sensor 1A	
11	ABL Sensor 1B	ABL Sensor 1B	
12	ABL Sensor 2B	ABL Sensor 1B	
13	AUX1	CPLD AUX1	Tensión 5V Max
14	AUX2	CPLD AUX2	Tensión 5V Max
15	TIP	TIP	2A Max
16	SHIELD	SHIELD	Tierra

Para este circuito, utilizando los conceptos seleccionados en la sección 5.2, se utilizarán 8 grupos de 2 relés DPDT cada uno en cascada. Como observamos en la figura 6.11, la cual es una versión simplificada del esquemático de conexión completo, en el cada puerto “NO” de los relés se conectará una señal distinta con el objetivo que cuando cada relé se encuentre en un estado no conmutado el puerto de la estación de señales se encuentre conectado a tierra, simulando una desconexión.

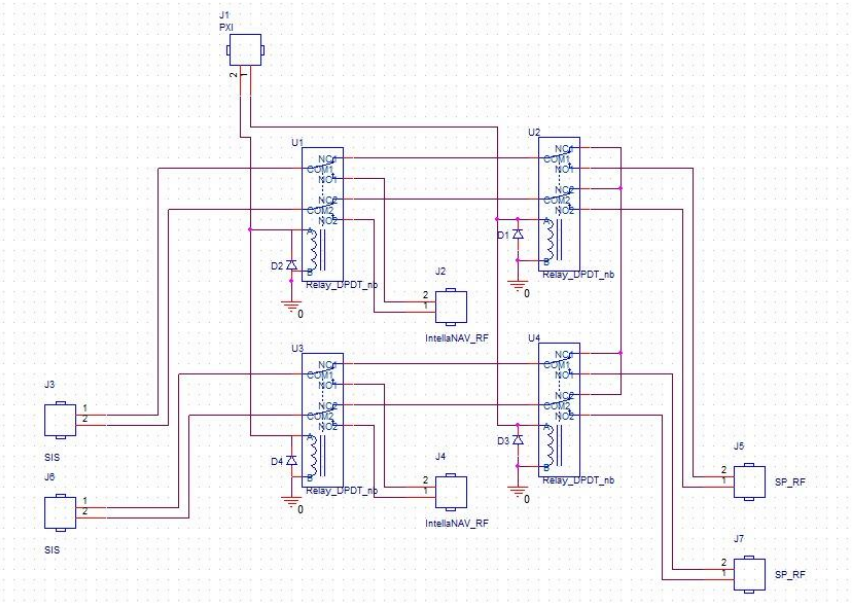


Figura 6.11. Esquemático simplificado de conexión maestro a puerto de ablación. Fuente: Propia

Para el control de las conexiones se utilizarán 2 señales provenientes del PXI Chassis las cuales conmutaran los relés y realizaran la conexión siguiendo la lógica mostrada en la tabla 6.6. Como se puede observar en la tabla 6.6 la lógica de control

presenta 1 caso no aplicable ya que este será filtrado por medio del código de control del PXI Chassis.

Para las entradas y salidas se utilizarán conectores AMP CT con resistencia de hasta 2.5 Amperios de conducción, los conectores utilizados serán de 2 terminales ya que por el acomodo de los pines que cada relé DPDT esto facilitara el posicionamiento del conector en la placa simplificando de esta manera las líneas de conducción en el PCB. Al igual que en los diseños anteriores se utilizarán en todo el circuito únicamente 2 leds, uno por cada señal de control.

Tabla 6.6. Lógica de control de conexión de maestro a estación de señales. Fuente: Propia

Control A	Control B	Conecta
Abierto	Abierto	No conecta
Abierto	Cerrado	SP
Cerrado	Abierto	IntellaNav
Caso no aplicable		
Cerrado	Cerrado	SP

6.2.4 Conexión Maestro-Generador de ablación

La funcionalidad de este circuito es conmutar la conexión de las entradas RF de los maestros de ablación IntellaNav y SP (ver sección 2.1.2.4) con el generador de ablación, este transmite a los maestros las señales observadas en la tabla 6.7.

Tabla 6.7 Señales del generador de ablación a maestro de ablación. Fuente: Propia

Señal	Generador de ablación	Característica
1	TIP	2A MAX
2	RID	3.3V Max
3	GND	Tierra común
4	TC1	Corriente 1.5A Max
5	TC2	
6	THERM	

Para este circuito, utilizando los conceptos seleccionados en la sección 5.2, se utilizarán 3 grupos de 2 relés DPDT cada uno en cascada. En figura 6.12 se

observan las señales entran por el puerto “COM” del primer nivel de relés, cuando estos se encuentren en estado no conmutado estas señales serán transferidas a tierra de manera que desde la perspectiva de la estación de señales no se verá ningún tipo de conexión y cada “Maestro” se tomará como desconectado.

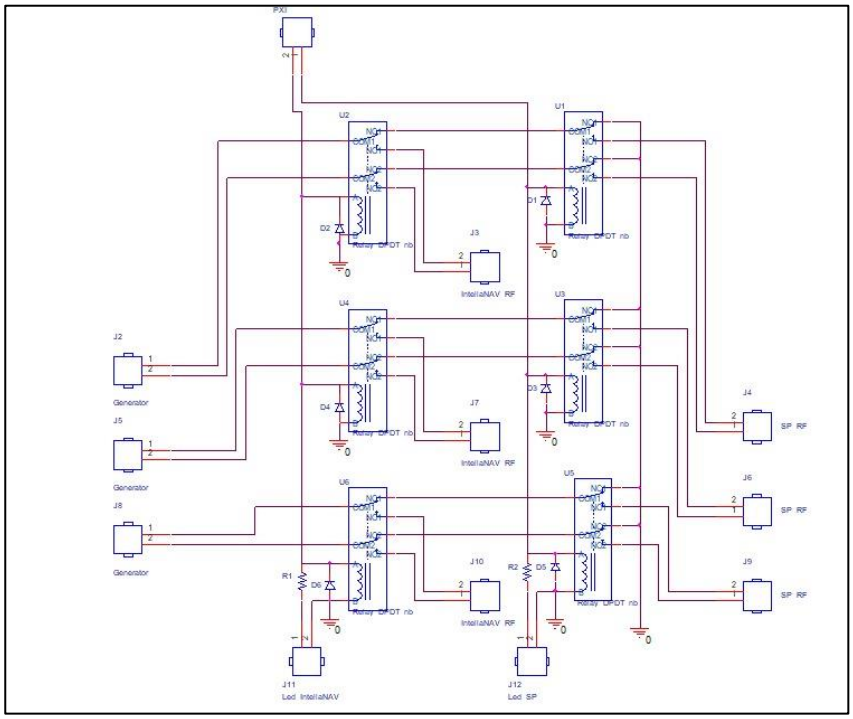


Figura 6.12. Esquemático de conexión generador de ablación a maestro de ablación. Fuente: Propia

Para el control de las conexiones se utilizarán 2 señales provenientes del PXI Chassis las cuales conmutaran los relés y realizaran la conexión siguiendo la lógica mostrada en la tabla 6.8. Como se puede observar en la tabla 6.8 la lógica de control presenta 1 caso no aplicable ya que este será filtrado por medio del código de control del PXI Chassis.

Tabla 6.8 Lógica de control de conexión de generador de ablación a maestro de ablación. Fuente: Propia

Control A	Control B	Conecta
Abierto	Abierto	No conecta
Abierto	Cerrado	SP
Cerrado	Abierto	IntellaNav
Caso no aplicable		
Cerrado	Cerrado	SP

Para las entradas y salidas se utilizarán conectores AMP CT con resistencia de hasta 2.5 Amperios de conducción, los conectores utilizados serán de 2 terminales ya que por el acomodo de los pines que cada relé DPDT esto facilitara el posicionamiento del conector en la placa simplificando de esta manera las líneas de conducción en el PCB. Al igual que en los diseños anteriores se utilizarán en todo el circuito únicamente 2 leds, uno por cada señal de control.

6.3 Diseño de accesorio de resguardo

6.3.1 Selección de material

Para el desarrollo de este accesorio de resguardo, se requiere que todas sus partes sean manufacturadas por medio de la tecnología de impresión 3D, dentro de la empresa Boston Scientific como equipo disponible utilizado por distintos departamentos se tiene la Big60 V3 de la compañía Modix la cual tiene una variedad de tipos de materiales soportados entre ellos el Nylon, el cual es uno de los materiales más utilizados en la industria.

El Nylon como material en impresión 3D en forma de filamentos ofrece un buen equilibrio en sus características químicas y mecánicas ya que presenta una buena rigidez, flexibilidad y resistencia a los golpes superando incluso a las características del ABS, comúnmente es utilizado en las industria aeroespacial, robótica y automotriz ideal para piezas que requieren alta resistencia y larga vida útil [33].

Entre las características más destacables de este material son:

- Un módulo de elasticidad de 494MPa
- Esfuerzo cortante ultimo 220MPa
- Resistencia a flexión 38MPa
- Resistencia a compresión 51MPa
- Limite elástico 28MPa
- Fácil de maquinar
- Resistencia a altas temperaturas, mayor a 100°C
- Liviano.

Entre estas anteriores podemos destacar la resistencia a altas temperaturas, con la cual se cumple el requerimiento 12 de la tabla 4.3, además de sus

características mecánicas lo cual nos asegura que es un material fácilmente manejable como es requerido según el requerimiento 13 y ligero como lo especifica el requerimiento 11 ambos de la tabla 4.3.

6.3.2 Desarrollo de partes

El diseño mecánico de la carcasa de resguardo se diseñará a partir de los conceptos seleccionados de estructura de carcasa, sujeción de circuitos impresos y cejillas para el sellado. Inicialmente se decide iniciar con el diseño de la base, esta presenta unas dimensiones de 388x156x10.5mm de manera general presenta una forma de prisma rectangular.

Desde cada extremo se dejarán 5mm de espacio para que se ajusta con el espesor del cuerpo de la carcasa, además de un espacio de 6mm en el cual de colocaran las cejillas para el sellado las cuales irán únicamente en los segmentos laterales para facilitar la manufactura, en los segmentos frontal y trasero se añadirá un elemento rectangular que cumple la función de sellado y garantiza que la base se mantenga estática, estos detalles se pueden observar en las figuras 6.13 y 6.14.

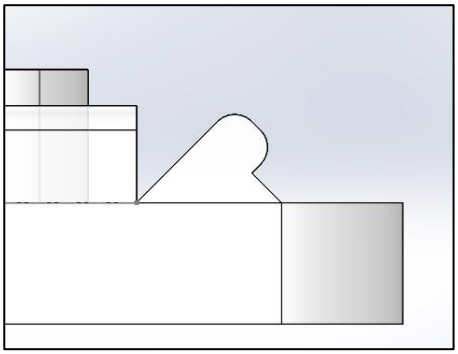


Figura 6.13. Cejilla Lateral de la base. Fuente: Propia

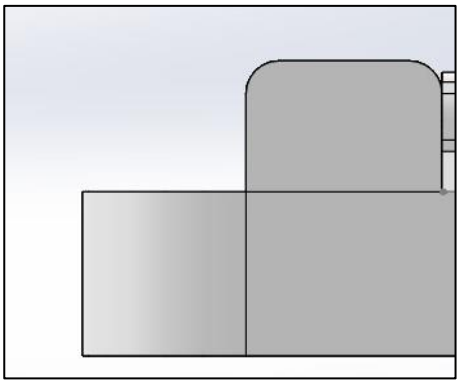


Figura 6.14. Cejilla Frontal/Trasera de la Base. Fuente: Propia

Como se observa en la figura 6.15, se distribuyeron los elementos de sujeción PCB de acuerdo con el acomodo propuesto, con una separación de 10mm a lo ancho y 5mm a lo largo entre cada circuito impreso, entre cada par de elementos de sujeción de añadió a lo largo unos nervios rectangulares los cuales incrementaran la resistencia de los cilindros y además proporcionaran mayor rigidez a la base para evitar pandeos, en la figura 6.16 se puede observar de manera general el resultado, debido al material y sus dimensiones reducidas en espesor la tapa tiene una masa de aproximadamente 450 gramos lo cual la hace una parte liviana que puede ser retirada y sujetada con facilidad, como se espera según los requerimientos del componente.

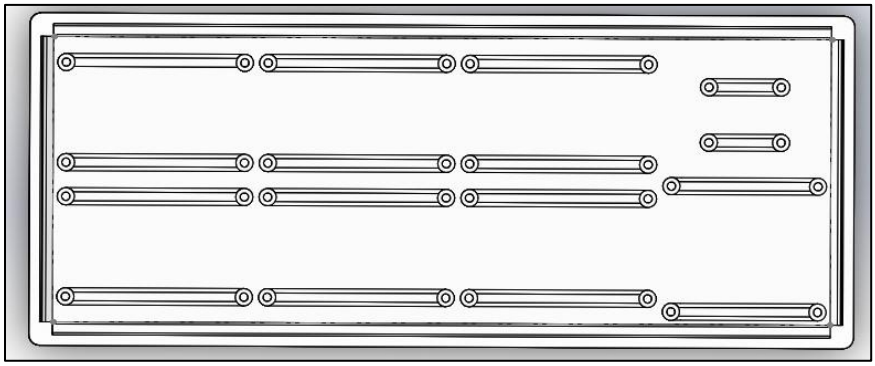


Figura 6.15. Vista superior de la base. Fuente: Propia

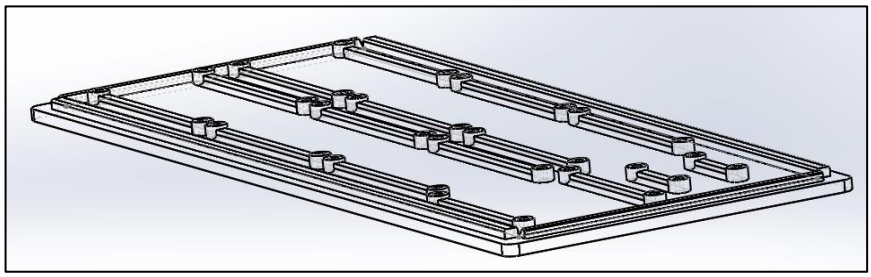


Figura 6.16. Base de la carcasa de resguardo. Fuente: Propia

Una vez diseñada la base se procedió a desarrollar la tapa de la estructura la cual presenta una forma de prisma rectangular con medidas de 388x156x9mm, utilizando los mismos conceptos de sellado por medio de las cejillas y basándonos en los criterios de diseño anteriores se colocaron las cejillas para cierre a lo ancho de la carcasa, y utilizando a lo largo una sección rectangular que asegure el sellado y mantenga estática la tapa en el eje Y.

Por otro lado, como se observa en la figura 6.17, se colocaron varios nervios a lo largo y ancho de la cara interna de la tapa con el fin de aumentar su rigidez y resistencia al pandeo, debido al material y sus dimensiones reducidas en espesor la tapa tiene una masa de aproximadamente 450 gramos lo cual la hace una parte liviana que puede ser retirada y sujetada con facilidad, como se espera según los requerimientos del componente.

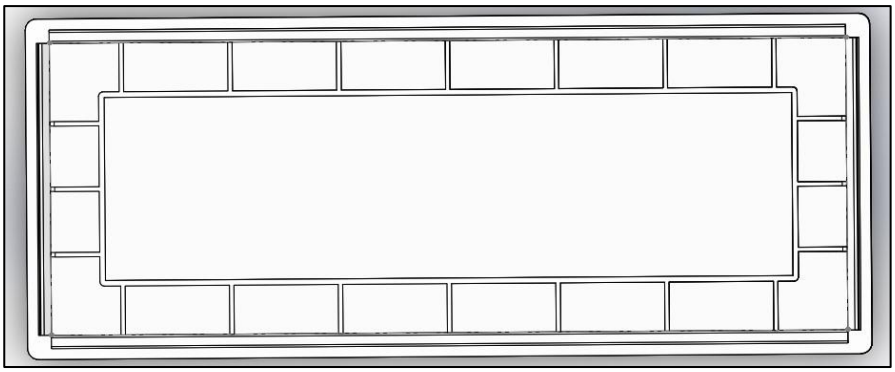


Figura 6.17. Vista superior de la cara interna de la tapa. Fuente: Propia

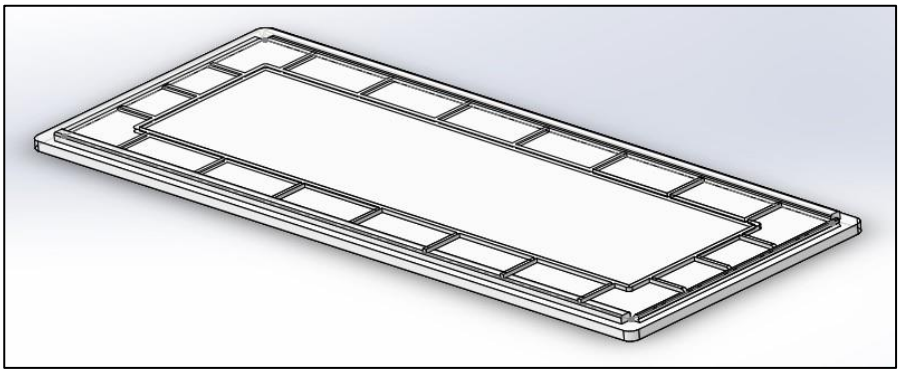


Figura 6.18. Tapa de la estructura de resguardo. Fuente: Propia

Finalmente se procedió a diseñar el cuerpo principal de la estructura de resguardo, este presenta una forma de prisma rectangular con dimensiones máximas de 388x156x200mm con una masa de 1.2Kg debido a sus dimensiones y material.

Como se observa en la figura 6.19, en la cara frontal de la carcasa de resguardo como se especifica en el requerimiento 3 de la tabla 4.3 se realizaron 22 agujeros para la conexión de dispositivos, estos agujeros presentan dimensiones de 20mm y 25mm, además de un corte circular extra de 5mm de diámetro y 3mm de profundidad en el perímetro de la cara exterior para el ajuste de cada conector

hembra de tipo PL-900 los cuales cuentan con un aro para fijado exterior y una rosca para fijado interior.

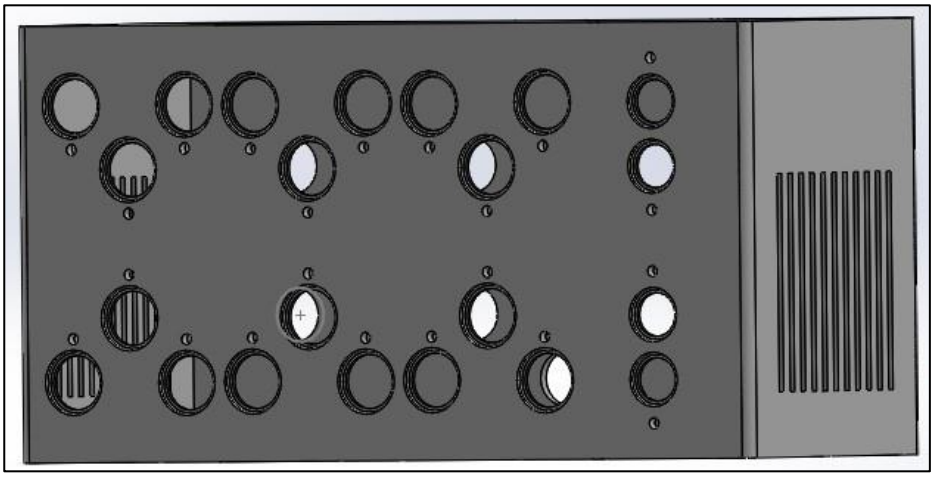


Figura 6.19. Cara frontal de carcasa de resguardo. Fuente: Propia

Para la cara trasera que se observa en la figura 6.20, como se indica en el requerimiento 4 se realizaron 8 agujeros con las mismas especificaciones de la cara frontal, además se realizó un agujero extra de 25mm de tamaño en el cual se colocara el conector con las señales de control provenientes del PXI Chassis.

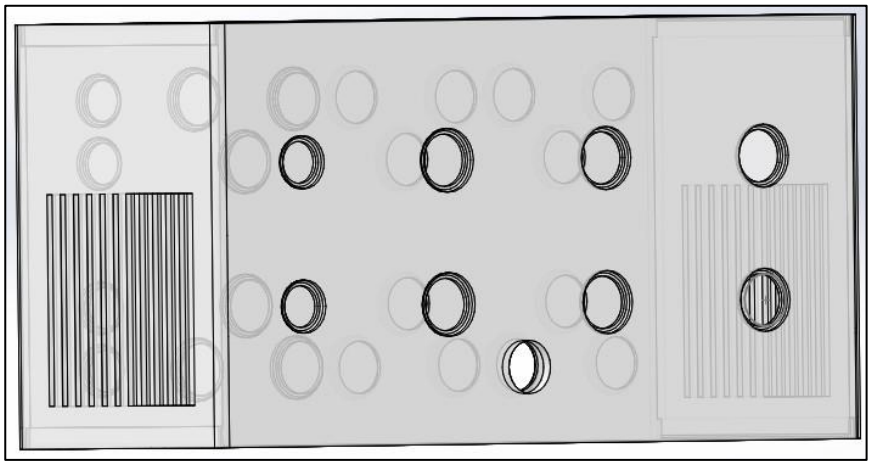


Figura 6.20. Cara trasera de carcasa de resguardo. Fuente: Propia

En ambas caras laterales se realizaron 11 cortes rectangulares de 4mm de ancho y 100mm de largo con la funcionalidad de ventilación para evitar el sobrecalentamiento el ambiente interno, como lo especifica el requerimiento 9 de la tabla 4.3. Cada corte está separado por 6mm esto para evitar la fragilidad del elemento entre cada rejilla y asegurar su integridad, como se observa en la figura 6.21.

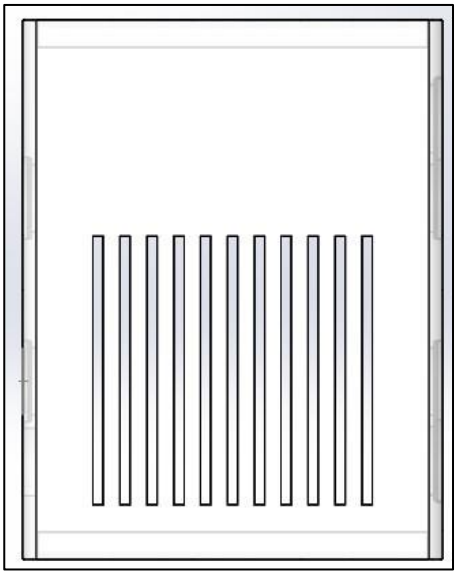


Figura 6.21. Cara lateral de carcasa de resguardo. Fuente: Propia

Finalmente, como contraparte de la cejilla de sellado en las caras laterales internas se desarrolló un elemento de perfil triangular, pegado a una sección rectangular de 1.5mm de espesor en voladizo con un espacio para flexión de 1.5mm suficiente para facilitar la dinámica de apertura y cierre de la carcasa sin recargar de esfuerzo las cejillas, para el desarrollo de esta sección se realizaron de esfuerzos de tensión y cortantes siguiendo la teoría de vigas en voladizo, esto se puede observar en la figura 6.22.

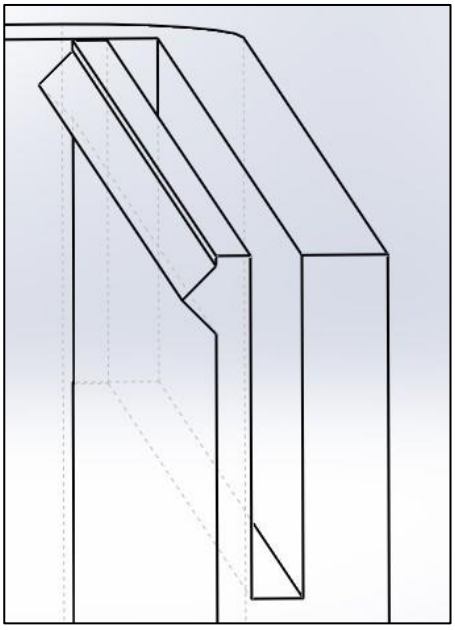


Figura 6.22. Contraparte de cejilla para cierre de carcasa. Fuente: Propia

7. Validación de los diseños.

7.1 Pruebas de funcionalidad del software de evaluación

Para esta sección se contempla que los elementos no se encuentran conectados de manera formal, pero si se encuentran configurados para pruebas informales, de esta manera se tiene en cuenta que el PXI Chassis y sus módulos funcionan íntegramente, existe una conexión entre el administrados de pruebas y la estación de señales, el paquete RTMD esta implementado en la estación de evaluación la cual se utilizó para realizar estas pruebas.

Todos los requerimientos mencionados en esta sección pertenecen a la tabla 4.1. Además, todas las pruebas fueron realizadas por el programador y un usuario ajeno al código en momentos distintos, para comprobar su fiabilidad.

Todas las pruebas que involucran la verificación o conmutación manual de los relés se realizaron con ayuda de un programa secundario llamado “relay_flipper”, como se muestra en la figura 7.1 este programa permite consultar y modificar el estado de un relé por medio de su número.

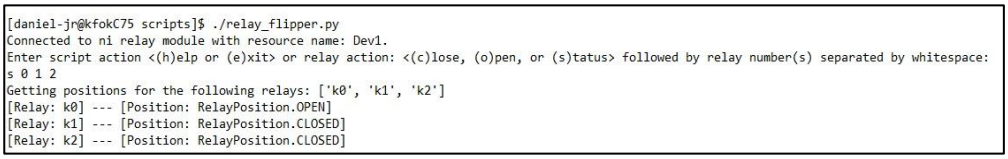


Figura 7.1. Programa relay_flipper. Fuente: propia

7.1.1 Pruebas unitarias de caja blanca

La función “logFilePathVerification” modifica el directorio como se explica en la sección 6.1.2. La función retorna el valor de “status” y los argumentos excluyendo la bandera y la dirección, en caso de que la instrucción no cumpla con el formato de la figura 5.1 el valor de “status” retornado será -1 y se detendrá la ejecución. Con el fin de cumplir el requerimiento 6 que dictamina que el directorio de resultados puede ser elegido por el usuario, se realizaron las pruebas de la tabla 7.1 para asegurar que la función cumple con las siguientes características:

- Identifica errores de formato.

- Modifica el directorio únicamente si es requerido e indicado de manera correcta.
- Retorna los argumentos restantes.
- Elimina los argumentos correctos.
- Reporta el fallo en caso de que ocurra.

Los resultados obtenidos fueron corroborados utilizando instrucciones de impresión colocadas en secciones estratégicas del código.

Tabla 7.1 Pruebas para función logFilePathVerification. Fuente: Propia

Pruebas	Resultados Esperado	Resultado Obtenido	Requerimiento Evaluado
Ejecuta el programa sin especificar nueva dirección Entrada: [post.py]	La variable Full_path mantiene la dirección estándar. **Salida**: [post.py]	La variable Full_path mantiene la dirección estándar. **Salida**: [post.py]	**6**
Ejecuta el programa sin especificar nueva dirección, especificando conectores Entrada: [post.py, conector$_1$, conector$_2$, conector$_3$]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, conector$_1$, conector$_2$, conector$_3$]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, conector$_1$, conector$_2$, conector$_3$]	
Ejecuta el programa haciendo uso de la bandera -v sin especificar nueva dirección Entrada: [post.py, -v]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, -v]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, -v]	
Ejecuta el programa haciendo uso de la bandera -v sin especificar nueva dirección, especificando conectores Entrada: [post.py, -v, conector$_1$, conector$_2$, conector$_3$]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, -v, conector$_1$, conector$_2$, conector$_3$]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, -v, conector$_1$, conector$_2$, conector$_3$]	
Ejecuta el programa haciendo uso de la bandera -p sin especificar nueva dirección Entrada: [post.py, -p]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, -p]	La variable Full_path mantiene la dirección estándar. **Salida:** [post.py, -p]	

Ejecuta el programa especificando nueva dirección correctamente **Entrada: [post.py, -r, dirección]**	La variable Full_path cambia la dirección. **Salida**: [post.py]	La variable Full_path cambia la dirección. **Salida**: [post.py]	
Ejecuta el programa especificando nueva dirección correctamente y especificando conectores **Entrada: [post.py, conector$_1$, conector$_2$, conector$_3$, -r, dirección]**	La variable Full_path cambia la dirección. **Salida:** [post.py, conector$_1$, conector$_2$, conector$_3$]	La variable Full_path cambia la dirección **Salida:** [post.py, conector$_1$, conector$_2$, conector$_3$]	
Ejecuta el programa haciendo uso de la bandera -v especificando nueva dirección correctamente **Entrada: [post.py, -v, -r, dirección]**	La variable Full_path cambia la dirección. **Salida:** [post.py, -v]	La variable Full_path cambia la dirección. **Salida:** [post.py, -v]	
Ejecuta el programa haciendo uso de la bandera -v especificando nueva dirección correctamente y especificando conectores **Entrada: [post.py, -v, conector$_1$, conector$_2$, conector$_3$, -r, dirección]**	La variable Full_path cambia la dirección. **Salida:** [post.py, -v, conector$_1$, conector$_2$, conector$_3$]	La variable Full_path cambia la dirección. **Salida:** [post.py, -v, conector$_1$, conector$_2$, conector$_3$]	
Ejecuta el programa haciendo uso de la bandera -p especificando nueva dirección **Entrada: [post.py, -p, -r, dirección]**	La variable Full_path cambia la dirección. **Salida:** [post.py, -p]	La variable Full_path cambia la dirección. **Salida:** [post.py, -p]	
Ejecuta el programa ingresando nueva dirección de manera incorrecta **Entrada: [post.py, -r, conector$_1$, conector$_2$, conector$_3$, dirección]** **Entrada: [post.py, conector$_1$, -r , dirección, conector$_2$, conector$_3$, dirección]** **Entrada: [post.py, -r , dirección, -v]**	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	

Entrada: [post.py, -r , dirección, -v, conector$_1$, conector$_2$, conector$_3$,]			

La función “verifyInputArgs” se encarga de comprobar los demás argumentos de la instrucción de ejecución y la función “connectorVerification” invocada dentro de la primera mencionada corrobora que todos los conectores validos indicados por el usuario, por lo tanto con el fin de comprobar que los parámetros de entrada presenten un orden fijo y que el software se puede ejecutar mediante una única instrucción como estipulan los requerimientos 5 y 11, se realizaron las pruebas de la tabla 7.2 corroborando las siguientes características:

- Ejecuta sin error para las instrucciones que siguen el formato de la figura 5.1 con la correcta sintaxis.
- Modifica variables globales cuando ocurra.
- Identifica errores formato y sintaxis en la instrucción.
- Indica al usuario en caso de error (ver figura 7.2).
- Reporta el fallo en caso de que ocurra.
- Crea lista de conectores a evaluar según el usuario lo indique.
- No repite conectores en la lista.

Los resultados obtenidos fueron corroborados utilizando instrucciones de impresión colocadas en secciones estratégicas del código.

```
[daniel-jr@kfokC75 scripts]$ ./post.py -v A_UT
------>Invalid flag or invalid arguments detected. Select flag -h for more information<------
User input was: ['./post.py', '-v', 'A_UT']
```

Figura 7.2. Error debido a argumento invalido. Fuente: propia

Tabla 7.2 Pruebas para función verifyInputArgs. Fuente: Propia

Pruebas	Resultados esperados	Resultados obtenidos	Requerimientos Evaluados
Se ejecuta el programa sin especificar ninguna bandera o conector. **Entrada: [post.py]**	Lista Current_connectors contiene todos los conectores disponibles. Log_detailed_message_enable: False. Test_pxi_chassis_enable: False	Lista Current_connectors contiene todos los conectores disponibles.	**5 y 11**

		Log_detailed_message_enable: False. Test_pxi_chassis_enable: False	
Se ejecuta el programa sin especificar ninguna bandera o conector, con nombre incorrecto. **Entrada: [past.py]**	**Salida:** No existe tal programa	**Salida:** No existe tal programa	
Se ejecuta el programa con banderas incorrectas. **Entrada: [post.py, g]** **Entrada: [post.py, v]** **Entrada: [post.py, 5]** **Entrada: [post.py, #]** **Entrada: [post.py, -v, t]**	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	
Se ejecuta el programa con bandera correctas. **Entrada: [post.py, -v]**	Lista Current_connectors contiene todos los conectores disponibles. Log_detailed_message_enable: True Test_pxi_chassis_enable: False	Lista Current_connectors contiene todos los conectores disponibles. Log_detailed_message_enable: True Test_pxi_chassis_enable: False	
Se ejecuta el programa con bandera correctas. **Entrada: [post.py, -p]**	Lista Current_connectors vacía. Log_detailed_message_enable: False Test_pxi_chassis_enable: True	Lista Current_connectors vacía. Log_detailed_message_enable: False Test_pxi_chassis_enable: True	
Se ejecuta el programa con bandera correctas. **Entrada: [post.py, -h]**	Imprime en consola mensaje de ayuda Finalización del programa	Imprime en consola	

		mensaje de ayuda Finalización del programa	
Se ejecuta el programa con banderas incorrectas y argumentos correctos. **Entrada: [post.py, v, conector$_1$, conector$_2$, conector$_3$]** **Entrada: [post.py, 5, conector$_1$, conector$_2$, conector$_3$]**	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	
Se ejecuta el programa con bandera correctas y argumentos incorrectos. **Entrada: [post.py, -v, hola]** **Entrada: [post.py, -v, $]** **Entrada: [post.py, -h, conector$_1$, conector$_2$, conector$_3$]** **Entrada: [post.py, -p, conector$_1$, conector$_2$, conector$_3$]**	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	
Se ejecuta el programa con bandera y argumentos correctos. **Entrada: [post.py, -v, conector$_1$, conector$_2$, conector$_3$]**	Lista Current_connectors contiene conector$_1$, conector$_2$, conector$_3$. Log_detailed_message_enable: True Test_pxi_chassis_enable: False	Lista Current_connectors contiene conector$_1$, conector$_2$, conector$_3$. Log_detailed_message_enable: True Test_pxi_chassis_enable: False	
Se ejecuta el programa sin bandera y argumentos correctos. **Entrada: [post.py, conector$_1$, conector$_2$, conector$_3$]**	Lista Current_connectors contiene conector$_1$, conector$_2$, conector$_3$. Log_detailed_message_enable: False Test_pxi_chassis_enable: False	Lista Current_connectors contiene conector$_1$, conector$_2$, conector$_3$.	

		Log_detailed_message_enable: False Test_pxi_chassis_enable: False	
Se ejecuta el programa sin bandera y argumentos incorrectos. **Entrada: [post.py, hola, $, 5]**	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	**Salida:** Imprime en consola-Error, bandera invalida o argumento invalido detectado Finalización del programa	

Para asegurar la creación de la hoja de resultados como lo dispone el requerimiento 4 se debe verificar que el directorio donde se guardará existe, esto a través de la función “checkResultPath”, para ello se realizaron las pruebas de la tabla 7.3 evaluando las siguientes características:

- Si el directorio existe retorna únicamente 0.
- Si el directorio no existe reporta el error (ver figura 7.3).

Las pruebas positivas y negativas se realizaron utilizando múltiples directorios y los resultados fueron corroborados mediante instrucciones de impresión colocadas en secciones estratégicas del código.

```
[daniel-jr@kfokC75 scripts]$ ./post.py -r /no/existe
Out directory /no/existe does not exists.
```

Figura 7.3. Error debido a directorio inexistente. Fuente: propia

Tabla 7.3 Pruebas para función checkResultPath. Fuente: Propia

Pruebas	Resultados esperados	Resultados Obtenidos	Requerimientos Evaluados
Se ejecuta utilizando el directorio estándar (existe) **Entrada: /.squish/TestResults/PostResults**	**Salida:** Se retorna un 0	**Salida:** Se retorna un 0	**4**
Se ejecuta utilizando un nuevo directorio (existe) **Entrada: /home/user/Documents**	**Salida:** Se retorna un 0	**Salida:** Se retorna un 0	

Se ejecuta utilizando un nuevo directorio (no existe) **Entrada: /home/user/Pruebaerror**	**Salida:** Directorio de salida /home/user/Pruebaerror no existe **Termina ejecución del programa**	**Salida:** Directorio de salida /home/user/Pruebaerror no existe **Termina ejecución del programa**	

El Controlador del PXI Chassis es uno de los 4 dispositivos que se deben evaluar como lo indica el requerimiento 1, para asegurar que tiene habilitada la comunicación con el módulo de relés como se estipula en el requerimiento 9 y que se encuentra conectado a internet como lo define el requerimiento 8. Esto se realiza ejecutando la función “pxiChassisVerification” a la cual por medio de las pruebas de la tabla 7.4 se le evalúan las siguientes características:

- Verifica la comunicación entre el controlador y modulo del PXI Chassis.
- Lee la información del dispositivo, la registra en consola y en el archivo de resultados.
- Presenta manejo de errores reportándolos y registrándolos, según el requerimiento 4. (ver figura 7.4).
- Retorna el valor de “status” correcto según como ocurra la verificación.

Las pruebas se realizaron utilizando múltiples estaciones de trabajo y los resultados fueron corroborados mediante instrucciones de impresión colocadas en secciones estratégicas del código.

```
[daniel-jr@kfokC75 scripts]$ ./post.py -p
2020-12-08 23:39:39,512 [ERROR] The PXI Switch Self Test Failed. Refer to the following error message: -1074118655: Another process has already opened a session to this switch

{"error_constant":"niapalerr_resourceNotFound","file":"/P/sa/ss/xlatormgr/trunk/19.5/source/nixlatormgr/main.cpp","line":263,"component":"nixlator","std_exception_what":"basic
Traceback (most recent call last):
  File "./post.py", line 946, in pxiChassisVerification
    self_Test = niswitch.Session(RESOURCE_NAME)._self_test()
  File "/usr/lib/python2.7/site-packages/niswitch/session.py", line 1486, in __init__
    self._vi = self._init_with_topology(resource_name, topology, simulate, reset_device)
  File "/usr/lib/python2.7/site-packages/niswitch/session.py", line 2193, in _init_with_topology
    errors.handle_error(self, error_code, ignore_warnings=False, is_error_handling=False)
  File "/usr/lib/python2.7/site-packages/niswitch/errors.py", line 95, in handle_error
    raise DriverError(code, description)
DriverError: -1074118655: Another process has already opened a session to this switch module.

{"error_constant":"niapalerr_resourceNotFound","file":"/P/sa/ss/xlatormgr/trunk/19.5/source/nixlatormgr/main.cpp","line":263,"component":"nixlator","std_exception_what":"basic
2020-12-08 23:39:39,513 [INFO] The PXI CHASSIS Verification: FAIL
2020-12-08 23:39:39,513 [CRITICAL] Finish session due to error.
2020-12-08 23:39:39,513 [INFO]
2020-12-08 23:39:39,513 [INFO] Script execution: DONE.
2020-12-08 23:39:39,514 [INFO] The results are logged in the file: </home/daniel-jr/.squish/TestResults/postResults_08-Dec-2020_23:39:37.log>
```

Figura 7.4. Error en la verificación del PXI. Fuente: propia

Tabla 7.4 Pruebas para función pxiChassisVerification. Fuente: Propia

Pruebas	Resultados esperados	Resultados Obtenidos	Requerimientos Evaluados
Se ejecuta con el PXI conectado y el nombre correcto del módulo.	**Salida:** Retorna 0, imprime información de los dispositivos.	**Salida:** Retorna 0, imprime información de los dispositivos.	**1, 4, 8 y 9**
Se ejecuta con el PXI desconectado y el nombre correcto del módulo	**Salida:** Retorna -1 **Registra mensaje:** El controlador no se pudo comunicar con el módulo**.**	**Salida:** Retorna -1 **Registra mensaje:** El controlador no se pudo comunicar con el módulo**.**	
Se ejecuta con el PXI conectado a otro dispositivo y el nombre correcto del módulo	**Salida:** Retorna -1 **Registra mensaje:** El controlador está siendo utilizado por otro dispositivo**.**	**Salida:** Retorna -1 **Registra mensaje:** El controlador está siendo utilizado por otro dispositivo**.**	
Se ejecuta con el PXI conectado y el nombre incorrecto del módulo	**Salida:** Retorna -1 **Registra mensaje:** El controlador no se pudo comunicar con el módulo**.**	**Salida:** Retorna -1 **Registra mensaje:** El controlador no se pudo comunicar con el módulo**.**	

El administrador de pruebas y la estación de señales son 2 de los 4 dispositivos a evaluar mencionados en el requerimiento 1, se les debe evaluar la conexión a internet como lo estipula el requerimiento 8 a través de la función

“executeCommandThruSSH”. Para asegurar su funcionalidad se utilizan las pruebas de la tabla 7.5 evaluando las siguientes características:

- Presenta manejo de errores reportándolos y registrándolos, según el requerimiento 4.
- Registra mensajes de éxito y error según sea adecuado, según el requerimiento 4.
- Verifica que ambos dispositivos tengan conexión a internet y se comuniquen entre ellos.

Las pruebas se realizaron desde múltiples estaciones de trabajo y los resultados fueron corroborados mediante instrucciones de impresión colocadas en secciones estratégicas del código.

Tabla 7.5 Pruebas para función executeCommandThruSSH. Fuente: Propia

Pruebas	Resultados esperados	Resultados Obtenidos	Requerimientos Evaluados
Se ejecuta con las IP del administrador de pruebas y la estación de señales correctas	**Salida:** Retorna 0, Registra mensaje a nivel “debug” que la verificación fue completada	**Salida:** Retorna 0, Registra mensaje a nivel “debug” que la verificación fue completada	**1, 4 y 8**
Se ejecuta con la IP del administrador de pruebas incorrecta y la de la estación de señales correcta.	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	
Se ejecuta con el administrador de pruebas apagado y la IP de la estación de señales correcta.	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	
Se ejecuta con las IP del administrador de pruebas y la estación de señales correctas, con usuario incorrecto.	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	
Se ejecuta con las IP del administrador de pruebas y la estación de señales correctas, con contraseña incorrecta.	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	**Salida:** Retorna -1 **Registra mensaje:** No se pudo establecer conexión, ERROR	
Se ejecuta con las IP del administrador de pruebas correcta	**Salida:** Retorna -1	**Salida:** Retorna -1	

y la IP de la estación de señales correctas incorrecta.	**Registra mensaje:** Maquina <IP> no es accesible.	**Registra mensaje:** Maquina <IP> no es accesible.	
Se ejecuta con las IP del administrador de pruebas correcta y la estación de señales apagada.	**Salida:** Retorna -1 **Registra mensaje:** Maquina <IP> no es accesible	**Salida:** Retorna -1 **Registra mensaje:** Maquina <IP> no es accesible	

La función “openNISession” se encarga de inicializar sesión con el controlador, que se utiliza para conmutar los relés del módulo PXI y cumplir con el requerimiento 10 por ello para asegurar su funcionalidad se le realizaron las pruebas de la tabla 7.6, evaluando las siguientes características:

- Presenta manejo de errores reportándolos y registrándolos, según el requerimiento 4.
- Registra los mensajes pertinentes, según el requerimiento 4.
- Marca el inicio de sesión a través de una variable global.
- Retorna el valor correcto de “status”.

Las pruebas se realizaron desde múltiples estaciones de trabajo y los resultados fueron corroborados mediante instrucciones de impresión colocadas en secciones estratégicas del código.

Tabla 7.6 Pruebas para función openNISession. Fuente: Propia

Pruebas	Resultados esperados	Resultados Obtenidos	Requerimientos Evaluados
Se ejecuta con el PXI conectado.	**Salida:** Retorna 0. Registra mensajes de nivel “Debug” Ni_session = True	**Salida:** Retorna 0, Registra mensajes de nivel “Debug” Ni_session = True.	**4 y 10**
Se ejecuta con el PXI desconectado.	**Salida:** Retorna -1. Registra mensajes de nivel “Error” Ni_session = False	**Salida:** Retorna -1. Registra mensajes de nivel “Error” Ni_session = False**.**	
Se ejecuta cuando otro dispositivo ya inicio sesión.	**Salida:** Retorna -1. Registra mensajes de nivel “Error”: Otra maquina ya está conectada Ni_session = False	**Salida:** Retorna -1. Registra mensajes de nivel “Error”: Otra maquina ya está conectada Ni_session = False	

A la función “getRelayFromConnector” se le realizaron pruebas como las ejemplificadas en tabla 7.7 (versión completa tabla 11.1), acá se evaluó que para cada conector los relés retornados fueran los correctos, esto con el objetivo de verificar que todos se estén contemplando en la evaluación como lo define el requerimiento 3.

Tabla 7.7 Ejemplo de pruebas para función getRelayFromConnector. Fuente: Propia

Pruebas	Resultados esperados	Resultados Obtenidos	Requerimientos Evaluados
Se ejecuta para el conector PIU.	**Salida:** Retorna 0. **Relés:** [<Relays.K00: 'k0'>]	**Salida:** Retorna 0. **Relés:** [<Relays.K00: 'k0'>]	**3**
Se ejecuta para el conector Conditioning.	**Salida:** Retorna 0. **Relés:** [<Relays.K50: 'k50'>]	**Salida:** Retorna 0. **Relés:** [<Relays.K50: 'k50'>]	
Se ejecuta para el conector Conditioning. Se ejecuta para el conector Maestro_4000	**Salida:** Retorna -1. **Salida:** Retorna 0. **Relés:** [<Relays.K85: 'k85'>, <Relays.K99: 'k99'>, <Relays.K83: 'k83'>, <Relays.K91: 'k91'>, <Relays.K89: 'k89'>, <Relays.K93: 'k93'>, <Relays.K95: 'k95'>, <Relays.K87: 'k87'>, <Relays.K97: 'k97'>, <Relays.K02: 'k2'>]	**Salida:** Retorna -1. **Salida:** Retorna 0. **Relés:** [<Relays.K85: 'k85'>, <Relays.K99: 'k99'>, <Relays.K83: 'k83'>, <Relays.K91: 'k91'>, <Relays.K89: 'k89'>, <Relays.K93: 'k93'>, <Relays.K95: 'k95'>, <Relays.K87: 'k87'>, <Relays.K97: 'k97'>, <Relays.K02: 'k2'>]	

7.1.2 Pruebas del sistema

Estas pruebas fueron acompañadas de un programa ajeno al evaluador denominado “relay_flipper.py” (ver figura 7.1) que permite observar el estado de los relés ya que estas pruebas están enfocadas en verificar que las funciones relacionadas con la comprobación del módulo de relés funcionan y que los resultados del evaluador son los esperados.

Para las pruebas de la tabla 7.8 todos los elementos a evaluar se encontraban en estado óptimo, acá se verificaron las posibles combinaciones de entradas que puede tener el evaluador y que el proceso de evaluación, el registro de eventos en consola y en el archivo fuera el esperado.

Tabla 7.8 Pruebas positivas de funcionalidad. Fuente: Propia

#	Prueba	Instrucción	Resultados esperados	Resultados Obtenidos	Requerimientos Evaluados
1	Entrada simple, sin resultados detallados.	./post.py	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo sin detallar. • Evalúa todos los conectores. • Sección de resultados • Directorio del archivo de resultados: el estándar	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo sin detallar. • Evalúa todos los conectores. • Sección de resultados • Directorio del archivo de resultados: el estándar	**2, 10 y 4**
2	Entrada simple, con resultados detallados.	./post.py -v	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo detallados. • Evalúa todos los conectores. • Sección de resultados • Directorio del archivo de resultados: el estándar	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo detallados. • Evalúa todos los conectores. • Sección de resultados	**1, 2 y 10**

				• Directorio del archivo de resultados: el estándar	
3	Entrada indicando conectores, con resultados sin detallar.	./post.py PIU	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo sin detallar. • Evalúa únicamente conectores indicados. • Sección de resultados • Directorio del archivo de resultados: el estándar	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo sin detallar. • Evalúa únicamente conectores indicados. • Sección de resultados • Directorio del archivo de resultados: el estándar	**1, 2 y 10**
4	Entrada indicando conectores, con resultados detallados.	./post.py -v PIU	• El código de salida: 0 • Eventos y resultados en consola y archivo detallados. • Evalúa únicamente conectores indicados. • Sección de resultados • Directorio del archivo de resultados: el estándar	• El código de salida: 0 • Eventos y resultados en consola y archivo detallados. • Evalúa únicamente conectores indicados. • Sección de resultados Directorio del archivo de	**1, 2 y 10**

				resultados: el estándar	
5	Entrada solicitando evaluación únicamente del controlador del PXI Chassis.	./post.py -p	• El código de salida: 0 • En consola y en el archivo únicamente se registra información de la evaluación del PXI. • Evalúa únicamente la conexión entre el controlador y los módulos. • Directorio donde del archivo de resultados: el estándar.	• El código de salida: 0 • En consola y en el archivo únicamente se registra información de la evaluación del PXI. • Evalúa únicamente la conexión entre el controlador y los módulos. • Directorio donde del archivo de resultados: el estándar.	**5**
6	Entrada solicitando ayuda.	./post.py -h	• El código de salida: Ninguno • En consola se registra mensaje de ayuda. • No se crea archivo de resultados.	• El código de salida: Ninguno • En consola se registra mensaje de ayuda. • No se crea archivo de resultados	**5**
7	Entrada simple, sin resultados detallados, indicando nuevo directorio.	./post.py -r /home/daniel-jr/Desktop	• El código de salida: 0 • Eventos y resultados en consola y archivo sin detallar.	• El código de salida: 0 • Eventos y resultados en consola	**6**

			• Evalúa todos los conectores. • Sección de resultados • Directorio del archivo de resultados: indicado	y archivo sin detallar. • Evalúa todos los conectores. • Sección de resultados • Directorio del archivo de resultados: indicado	
8	Entrada simple, con resultados detallados, indicando nuevo directorio.	./post.py -v -r /home/daniel-jr/Desktop	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo detallados. • Evalúa todos los conectores. • Sección de resultados • Directorio del archivo de resultados: indicado	**Salida:** • El código de salida: 0 • Eventos y resultados en consola y archivo detallados. • Evalúa todos los conectores. • Sección de resultados • Directorio del archivo de resultados: indicado	**6**

Todas las pruebas del flujo de verificación estándar se realizaron para asegurar el cumplimiento del requerimiento 2 y 10 que tienen relación directa con la evaluación del módulo PXI y sus relés. También se evalúa que la integración de las demás funciones se realizó con éxito y con cada una se cumplan los requerimientos con los que se relacionan.

La prueba 1 es una comprobación de la verificación estándar en la cual se revisó el estado de todos los relés y se obtuvieron resultados como los observados

en la figura 7.5, acá se puede denotar la estructura de los resultados, según el requerimiento 4:

- Inicialmente la verificación del controlador PXI.
- Se indican los conectores que se evaluará.
- La sección de resultados agrupa los conectores evaluados con éxito y los conectores que fallaron (ver figura 7.11).
- Se indica que la ejecución concluyo con éxito.
- El archivo de resultados presenta una sección con el estado estándar de los relés de cada conector evaluado. (ver figura 7.6)

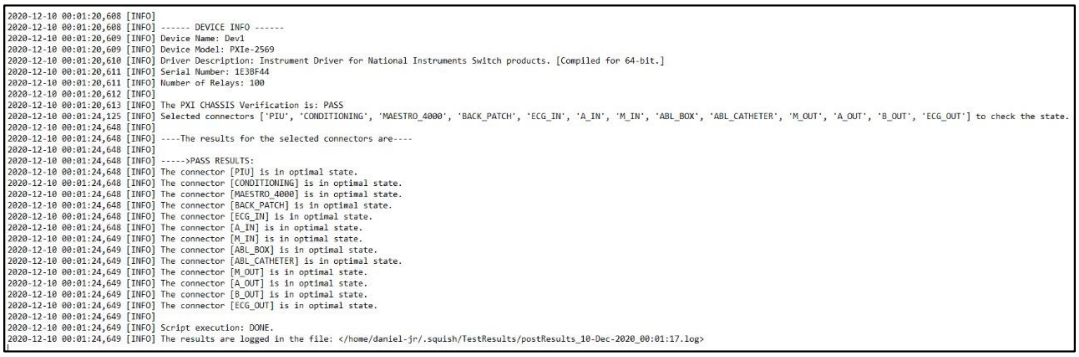


Figura 7.5. Resultados de evaluación estándar. Fuente: Propia

```
------The default state for the verified connectors is:------

        DEFAULT STATE OF PIU RELAYS ARE: OPENED
        DEFAULT STATE OF CONDITIONING RELAYS ARE: OPENED
        DEFAULT STATE OF MAESTRO_4000 BUTTONS RELAYS ARE:OPENED
        DEFAULT STATE OF MAESTRO_4000 POWER RELAY IS:CLOSED
        DEFAULT STATE OF BACK_PATCH RELAYS ARE: CLOSED
        DEFAULT STATE OF ECG_IN RELAYS ARE: CLOSED
        DEFAULT STATE OF A_IN RELAYS ARE: OPENED
        DEFAULT STATE OF M_IN RELAYS ARE: CLOSED
        DEFAULT STATE OF ABL_BOX RELAYS ARE: OPENED
        DEFAULT STATE OF ABL_CATHETER RELAYS ARE: OPENED
        DEFAULT STATE OF M_OUT RELAYS ARE: OPENED
        DEFAULT STATE OF A_OUT RELAYS ARE: OPENED
        DEFAULT STATE OF B_OUT RELAYS ARE: OPENED
        DEFAULT STATE OF ECG_OUT RELAYS ARE: OPENED
```

Figura 7.6. Indicación del estado por defecto de los conectores evaluados. Fuente: Propia

Las pruebas 2, 3 y 4 fueron variaciones de la verificación estándar, cuando se utilizó la bandera “-v” se registraron todos los eventos, con esto se verificaron las siguientes características:

- La ejecución completa del código incluye la evaluación los 4 dispositivos mencionados en el requerimiento 1. (Ver figura 7.7, incluye resultados de evaluación del controlador PXI y de la conexión

del administrador de pruebas y la estación de señales.)(Ver figura 7.8, incluye resultados detallados de la evaluación del módulo PXI)

- El archivo de resultados y la consola presentan los eventos detallados.
- Los relés conmutados son únicamente los indicados, corroborándolo con ayuda del programa “relay_flipper”. (ver figura 7.1)

```
2020-12-10 00:02:37,049 [INFO]
2020-12-10 00:02:37,049 [INFO] ------ DEVICE INFO ------
2020-12-10 00:02:37,050 [INFO] Device Name: Dev1
2020-12-10 00:02:37,050 [INFO] Device Model: PXIe-2569
2020-12-10 00:02:37,051 [INFO] Driver Description: Instrument Driver for National Instruments Switch products. [Compiled for 64-bit.]
2020-12-10 00:02:37,052 [INFO] Serial Number: 1E3BF44
2020-12-10 00:02:37,053 [INFO] Number of Relays: 100
2020-12-10 00:02:37,053 [INFO]
2020-12-10 00:02:37,054 [INFO] The PXI CHASSIS Verification is: PASS
2020-12-10 00:02:39,490 [DEBUG] Machine 10.0.2.15 is reachable and this ensures dhcpv6-client service is running
2020-12-10 00:02:39,891 [DEBUG] The verification between the Test Manager and the Signal Station is: PASS
2020-12-10 00:02:39,892 [DEBUG] Start session to NI Relay Module
2020-12-10 00:02:39,892 [DEBUG] The Session start with the resource name: Dev1.
2020-12-10 00:02:40,557 [DEBUG] The NI Session initialization is: PASS
2020-12-10 00:02:40,557 [DEBUG] Start Relay Verification
```

Figura 7.7. Resultados detallados de la función executeCommandThruSSH. Fuente: Propia

```
2020-12-10 00:07:27,726 [DEBUG] Start Relay Verification
2020-12-10 00:07:27,726 [INFO] Selected connectors ['PIU'] to check the state.
2020-12-10 00:07:27,726 [DEBUG]
2020-12-10 00:07:27,726 [DEBUG] -----Getting list of relays from the connector PIU.-----
2020-12-10 00:07:27,726 [DEBUG]
2020-12-10 00:07:27,727 [DEBUG] Getting state of the NI Relay Number: Relays.K00.
2020-12-10 00:07:27,727 [DEBUG] NI Relay Number: Relays.K00 state is OPENED.
2020-12-10 00:07:27,728 [DEBUG] Closing the NI Relays: Relays.K00.
2020-12-10 00:07:27,731 [DEBUG] Getting state of the NI Relay Number: Relays.K00.
2020-12-10 00:07:27,732 [DEBUG] NI Relay Number: Relays.K00 state is CLOSED.
2020-12-10 00:07:27,732 [DEBUG] Returning the NI Relay Number: Relays.K00 to default state: OPENED, the actual statue is CLOSED.
2020-12-10 00:07:27,732 [DEBUG] Opening the NI Relays: Relays.K00.
2020-12-10 00:07:27,736 [DEBUG] Getting state of the NI Relay Number: Relays.K00.
2020-12-10 00:07:27,736 [DEBUG] NI Relay Number: Relays.K00 state is OPENED.
2020-12-10 00:07:27,737 [DEBUG]
2020-12-10 00:07:27,737 [DEBUG] Finish session to NI Relay Module
2020-12-10 00:07:27,738 [DEBUG] The NI Session close is: PASS
2020-12-10 00:07:27,738 [INFO]
2020-12-10 00:07:27,738 [INFO] ----The results for the selected connectors are----
2020-12-10 00:07:27,738 [INFO]
2020-12-10 00:07:27,738 [INFO] ----->PASS RESULTS:
2020-12-10 00:07:27,739 [INFO] The connector [PIU] is in optimal state.
2020-12-10 00:07:27,739 [INFO]
2020-12-10 00:07:27,739 [INFO] Script execution: DONE.
2020-12-10 00:07:27,739 [INFO] The results are logged in the file: </home/daniel-jr/.squish/TestResults/postResults_10-Dec-2020_00:07:21.log>
```

Figura 7.8. Mensajes detallados en la verificación de los relés. Fuente: Propia

La prueba 5 y 6 permitieron verificar que cuando se utilizan estas banderas no se ejecuta ninguna función relacionada los módulos de relés y que el programa se ejecuta adecuadamente siguiendo los argumentos de una única instrucción según lo estipula el requerimiento 5.

En la figura 7.9 se observa cómo se ve el archivo de resultados cuando únicamente se verifica el PXI y en la figura 7.10 se observa la impresión en consola que funciona de ayuda para el usuario en caso de que este lo solicite con la bandera “-h”.

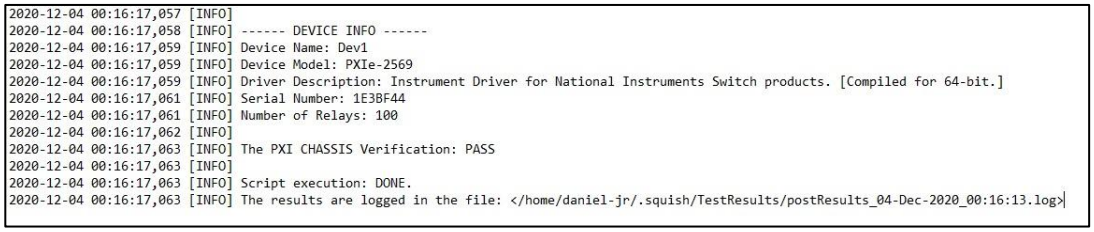


Figura 7.9. Resultados de evaluación individual del PXI. Fuente: Propia

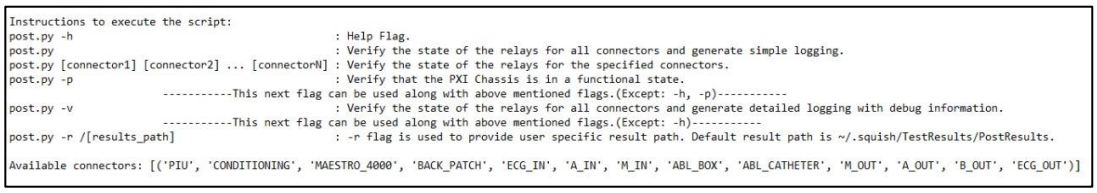


Figura 7.10. Mensaje de ayuda para el usuario. Fuente: Propia

Las pruebas 7 y 8 también se realizaron indicando 1 o múltiples conectores y siempre corroborando que al final se registrara el correcto directorio y que en dicho directorio realmente existiera el archivo de resultado, cumpliendo así con el requerimiento 6.

Las pruebas de la tabla 7.9 se replicaron fallando distintos conectores o colocando fuera de estado en cada ejecución distintos relés, el objetivo es comprobar el manejo y registro de errores como lo estipula el requerimiento 7.

Tabla 7.9 Pruebas negativas de funcionalidad. Fuente: Propia

#	Prueba	Resultados esperados	Resultados Obtenidos	Requerimientos evaluados
1	Se ejecuta el programa evaluador forzando el fallo en 1 o múltiples conectores.	**Salida:** • El código de salida: -1 • Luego de la verificación de los conectores se finaliza el programa. • En la hoja de resultados se indican los conectores y relés fallidos. • El programa finaliza la	**Salida:** • El código de salida: -1 • Luego de la verificación de los conectores se finaliza el programa. • En la hoja de resultados se indican los conectores y relés fallidos. • El programa finaliza la ejecución debido a error con el proceso esperado.	**7**

		ejecución debido a error con el proceso esperado.		
2	Se ejecuta el programa con relés aleatorios fuera de estado por defecto.	**Salida:** • El código de salida: 0 • Se registra mensaje de advertencia para cada relé fuera del estado por defecto. • Todos los relés regresan a su estado por defecto.	**Salida:** • El código de salida: 0 • Se registra mensaje de advertencia para cada relé fuera del estado por defecto. • Todos los relés regresan a su estado por defecto.	

En la figura 7.11 se pueden ver el registro de las 2 secciones de resultados de la conmutación de relés, los fallos se dividen en los conectores a los cuales no se les pudo obtener la lista de relés y los relés asociados a los conectores evaluados que no pudieron ser conmutados. También ejemplifica el registro de evento critico indicando que la ejecución finalizará debido a error y que previo a finalizar la ejecución se cierran las sesiones abiertas, este comportamiento se repite para los errores en cualquier sección del programa.

```
2020-11-30 13:14:01,464 [INFO] ----The results for the selected modules are----

2020-11-30 13:14:01,464 [INFO] ----->PASS RESULTS:
2020-11-30 13:14:01,464 [INFO] The module [CONDITIONING] is in optimal state.
2020-11-30 13:14:01,464 [INFO] The module [MAESTRO_4000] is in optimal state.
2020-11-30 13:14:01,464 [INFO] The module [BACK_PATCH] is in optimal state.
2020-11-30 13:14:01,464 [INFO] The module [ECG_IN] is in optimal state.
2020-11-30 13:14:01,465 [INFO] The module [A_IN] is in optimal state.
2020-11-30 13:14:01,465 [INFO] The module [M_IN] is in optimal state.
2020-11-30 13:14:01,465 [INFO] The module [ABL_BOX] is in optimal state.
2020-11-30 13:14:01,465 [INFO] The module [ABL_CATHETER] is in optimal state.
2020-11-30 13:14:01,465 [INFO] The module [A_OUT] is in optimal state.
2020-11-30 13:14:01,465 [INFO] The module [ECG_OUT] is in optimal state.
2020-11-30 13:14:01,465 [INFO]
2020-11-30 13:14:01,465 [WARNING] ----->FAIL RESULTS:
2020-11-30 13:14:01,465 [WARNING]
2020-11-30 13:14:01,465 [WARNING] Could not get the relays of the following modules:
2020-11-30 13:14:01,465 [WARNING] FAILED MODULE: [PIU].
2020-11-30 13:14:01,465 [WARNING] FAILED MODULE: [B_OUT].
2020-11-30 13:14:01,465 [WARNING]
2020-11-30 13:14:01,465 [WARNING] From module M_OUT, the following relays failed to switch:
2020-11-30 13:14:01,465 [WARNING] FAILED RELAY: [Relays.K90].
2020-11-30 13:14:01,466 [WARNING] FAILED RELAY: [Relays.K94].
2020-11-30 13:14:01,466 [WARNING] FAILED RELAY: [Relays.K92].
2020-11-30 13:14:01,466 [WARNING]
2020-11-30 13:14:01,466 [CRITICAL] Finish session due to error.
2020-11-30 13:14:01,466 [INFO] Finish session to NI Relay Module
2020-11-30 13:14:01,466 [INFO] The NI Session close is: COMPLETE
2020-11-30 13:14:01,466 [INFO] Script execution: DONE.
```

Figura 7.11. Fallos registrados en el archivo de resultados. Fuente: propia

En las figuras 7.12 y 7.13 se muestran ejemplos de los resultados de las pruebas de tipo 2 en el cual utilizando de apoyo el programa “relay_flipper” se modifica el estado de relés aleatorios de un mismo conector de manera manual previo a la ejecución del programa de evaluación y después de la ejecución del programa se consulta el estado de los relés para asegurar que fue modificado (ver figura 7.14), en estas figuras se observa el registro de los mensajes de advertencia de manera detallada y no detallada cuando el evaluador encuentra un relé fuera del estado estándar, como parte del requerimiento 4.

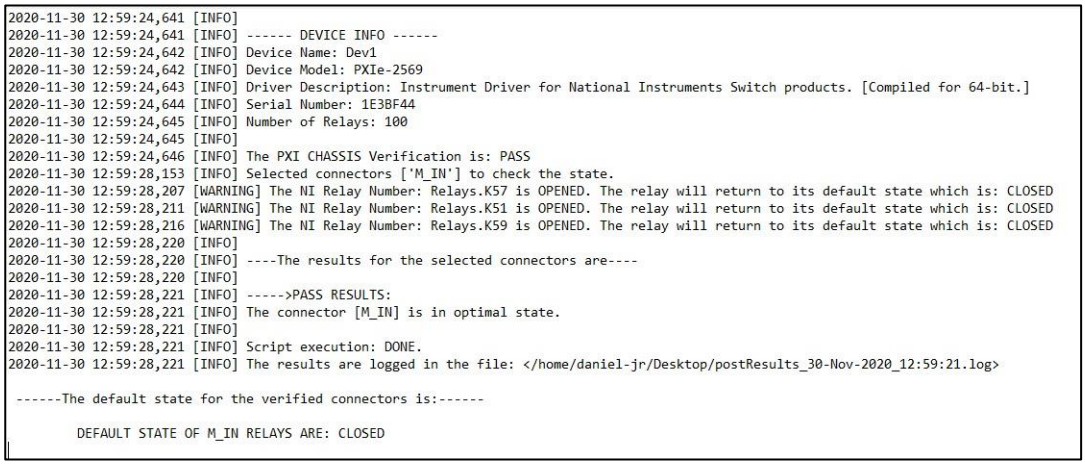


Figura 7.12. Advertencia en registro de resultados sin detallar. Fuente: Propia

```
2020-11-30 13:08:16,924 [DEBUG] -----Getting list of relays from the connector BACK_PATCH.-----
2020-11-30 13:08:16,924 [DEBUG]
2020-11-30 13:08:16,924 [DEBUG] Getting state of the NI Relay Number: Relays.K75.
2020-11-30 13:08:16,925 [DEBUG] NI Relay Number: Relays.K75 state is CLOSED.
2020-11-30 13:08:16,925 [DEBUG] Opening the NI Relays: Relays.K75.
2020-11-30 13:08:16,929 [DEBUG] Getting state of the NI Relay Number: Relays.K75.
2020-11-30 13:08:16,929 [DEBUG] NI Relay Number: Relays.K75 state is OPENED.
2020-11-30 13:08:16,929 [DEBUG] Returning the NI Relay Number: Relays.K75 to default state: CLOSED, the actual statue is OPENED.
2020-11-30 13:08:16,930 [DEBUG] Closing the NI Relays: Relays.K75.
2020-11-30 13:08:16,934 [DEBUG] Getting state of the NI Relay Number: Relays.K75.
2020-11-30 13:08:16,934 [DEBUG] NI Relay Number: Relays.K75 state is CLOSED.
2020-11-30 13:08:16,934 [DEBUG]
2020-11-30 13:08:16,934 [DEBUG] Getting state of the NI Relay Number: Relays.K71.
2020-11-30 13:08:16,935 [DEBUG] NI Relay Number: Relays.K71 state is OPENED.
2020-11-30 13:08:16,935 [WARNING] The NI Relay Number: Relays.K71 is OPENED. The relay will return to its default state which is: CLOSED
2020-11-30 13:08:16,935 [DEBUG] Closing the NI Relays: Relays.K71.
2020-11-30 13:08:16,939 [DEBUG] Getting state of the NI Relay Number: Relays.K71.
2020-11-30 13:08:16,939 [DEBUG] NI Relay Number: Relays.K71 state is CLOSED.
2020-11-30 13:08:16,940 [DEBUG] Getting state of the NI Relay Number: Relays.K71.
2020-11-30 13:08:16,940 [DEBUG] NI Relay Number: Relays.K71 state is CLOSED.
2020-11-30 13:08:16,940 [DEBUG]
2020-11-30 13:08:16,940 [DEBUG] Getting state of the NI Relay Number: Relays.K73.
2020-11-30 13:08:16,941 [DEBUG] NI Relay Number: Relays.K73 state is OPENED.
2020-11-30 13:08:16,941 [WARNING] The NI Relay Number: Relays.K73 is OPENED. The relay will return to its default state which is: CLOSED
2020-11-30 13:08:16,941 [DEBUG] Closing the NI Relays: Relays.K73.
2020-11-30 13:08:16,945 [DEBUG] Getting state of the NI Relay Number: Relays.K73.
2020-11-30 13:08:16,945 [DEBUG] NI Relay Number: Relays.K73 state is CLOSED.
2020-11-30 13:08:16,945 [DEBUG] Getting state of the NI Relay Number: Relays.K73.
2020-11-30 13:08:16,946 [DEBUG] NI Relay Number: Relays.K73 state is CLOSED.
2020-11-30 13:08:16,946 [DEBUG]
2020-11-30 13:08:16,946 [DEBUG] Getting state of the NI Relay Number: Relays.K80.
2020-11-30 13:08:16,946 [DEBUG] NI Relay Number: Relays.K80 state is OPENED.
2020-11-30 13:08:16,947 [WARNING] The NI Relay Number: Relays.K80 is OPENED. The relay will return to its default state which is: CLOSED
2020-11-30 13:08:16,947 [DEBUG] Closing the NI Relays: Relays.K80.
2020-11-30 13:08:16,951 [DEBUG] Getting state of the NI Relay Number: Relays.K80.
2020-11-30 13:08:16,951 [DEBUG] NI Relay Number: Relays.K80 state is CLOSED.
2020-11-30 13:08:16,951 [DEBUG] Getting state of the NI Relay Number: Relays.K80.
2020-11-30 13:08:16,951 [DEBUG] NI Relay Number: Relays.K80 state is CLOSED.
2020-11-30 13:08:16,952 [DEBUG]
2020-11-30 13:08:16,952 [DEBUG] Getting state of the NI Relay Number: Relays.K82.
2020-11-30 13:08:16,952 [DEBUG] NI Relay Number: Relays.K82 state is CLOSED.
2020-11-30 13:08:16,952 [DEBUG] Opening the NI Relays: Relays.K82.
2020-11-30 13:08:16,956 [DEBUG] Getting state of the NI Relay Number: Relays.K82.
2020-11-30 13:08:16,956 [DEBUG] NI Relay Number: Relays.K82 state is OPENED.
2020-11-30 13:08:16,957 [DEBUG] Returning the NI Relay Number: Relays.K82 to default state: CLOSED, the actual statue is OPENED.
2020-11-30 13:08:16,957 [DEBUG] Closing the NI Relays: Relays.K82.
2020-11-30 13:08:16,961 [DEBUG] Getting state of the NI Relay Number: Relays.K82.
2020-11-30 13:08:16,961 [DEBUG] NI Relay Number: Relays.K82 state is CLOSED.
2020-11-30 13:08:16,961 [DEBUG]
2020-11-30 13:08:16,961 [DEBUG] Finish session to NI Relay Module
2020-11-30 13:08:16,962 [DEBUG] The NI Session close is: PASS
```

Figura 7.13. Advertencia en registro de resultados detallado. Fuente: Propia

```
-----------------------------------------Previo a la ejecucion del evaluador-----------------------------------------------------------

Enter script action <(h)elp or (e)xit> or relay action: <(c)lose, (o)pen, or (s)tatus> followed by relay number(s) separated by whitespace:
o 57 51 59
Performing action for the following relays: ['k57', 'k51', 'k59']


Enter script action <(h)elp or (e)xit> or relay action: <(c)lose, (o)pen, or (s)tatus> followed by relay number(s) separated by whitespace:
s 57 51 59
Getting positions for the following relays: ['k57', 'k51', 'k59']
[Relay: k57] --- [Position: RelayPosition.OPEN]
[Relay: k51] --- [Position: RelayPosition.OPEN]
[Relay: k59] --- [Position: RelayPosition.OPEN]

-----------------------------------------Despues de la ejecucion del evaluador---------------------------------------------------------

Enter script action <(h)elp or (e)xit> or relay action: <(c)lose, (o)pen, or (s)tatus> followed by relay number(s) separated by whitespace:
s 57 51 59
Getting positions for the following relays: ['k57', 'k51', 'k59']
[Relay: k57] --- [Position: RelayPosition.CLOSED]
[Relay: k51] --- [Position: RelayPosition.CLOSED]
[Relay: k59] --- [Position: RelayPosition.CLOSED]
```

Figura 7.14. Resultados de verificación del programa relay_flipper. Fuente: Propia

Con este conjunto de pruebas se verifica que el programa evalúa 4 dispositivos distintos que son el administrador de pruebas, la estación de señales, el módulo PXI y el controlador PXI, al módulo se le verifican todos los relés que se encuentran actualmente en uso y todo esto realizado mediante una única instrucción de formato fijo en la ventana de comando, dándole posibilidad al usuario de modificar el directorio por defecto donde se guarda el archivo con los resultados.

7.2 Verificación de circuitos de conmutación

7.2.1 Simulaciones de circuito de conmutación

Para esta sección se utilizó la herramienta OrCAD PSpice la cual está catalogada como uno de los programas de simulación más eficaces para reproducir el comportamiento de los circuitos eléctricos.

En esta sección se utilizaron distintas configuraciones de conexión con los elementos seleccionados en la sección 5.2 con el fin de corroborar que los elementos soportan la potencia, tensión y corriente de las señales según se describe en los requerimientos y además cumple con los requisitos en tiempos de conmutación.

Para la simulación de la señales se utilizó la configuración observada en la figura 7.15, esta utiliza una fuente de pulso programable la cual nos permite modificar las características de la señal de salida en conjunto con un potenciómetro, ya que reproducir con exactitud la forma y nivel de cada una de las señales no es posible en el entorno de trabajo, con esta configuración se buscó poder aproximar los límites de tensión, corriente y frecuencia para poder asegurar que la configuración de diseño seleccionada es capaz de manejar cualquier punto intermedio.

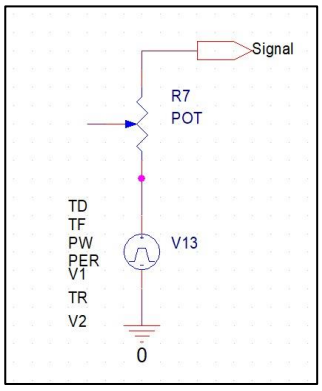


Figura 7.15. Generador de señales. Fuente: Propia

Para la simulación de las señales de control se utilizó la configuración observada en la figura 7.16, como se puede observar se utilizaron relés para simular la transmisión de las señales de control al circuito en un tiempo determinado, la señal de control proviene de una fuente DC de 5V para la tensión de alimentación de las bobinas de los relés, teniendo en cuenta que cada simulación se realizó con

esta estructura, se puede concluir que la alimentación es de 5V cumpliendo así con el requerimiento 2 de la tabla 4.2.

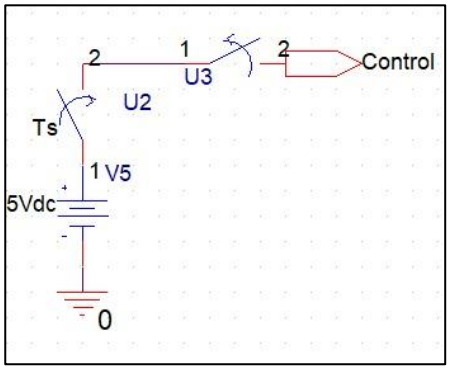


Figura 7.16. Simulación de señales de control. Fuente: Propia

Para las primeras pruebas se evaluaron múltiples señales constantes de 1V hasta 5V enviadas a través de 2 y 3 niveles de relés DPDT, estas pruebas se realizaron en un margen de tiempo de 5 minutos en los cuales se conectó y se simulo la conexión y desconexión de múltiples dispositivos, en la figura 7.17 podemos observar un ejemplo general del tipo de grafica obtenida en el cual se muestran la señal original de entrada y las resultantes de salida. De manera más detallada en la figura 7.18 se observa como en los 60s se activa la señal de control A y la señal del dispositivo 1 en rojo adquiere una tensión de 5V constantes al igual que la señal se entrada para este caso específico, luego a los 120 segundos la señal de control A se desactiva y se activa la de control B con una tensión de 5V constantes conmutando así la señal de entrada entre dispositivos.

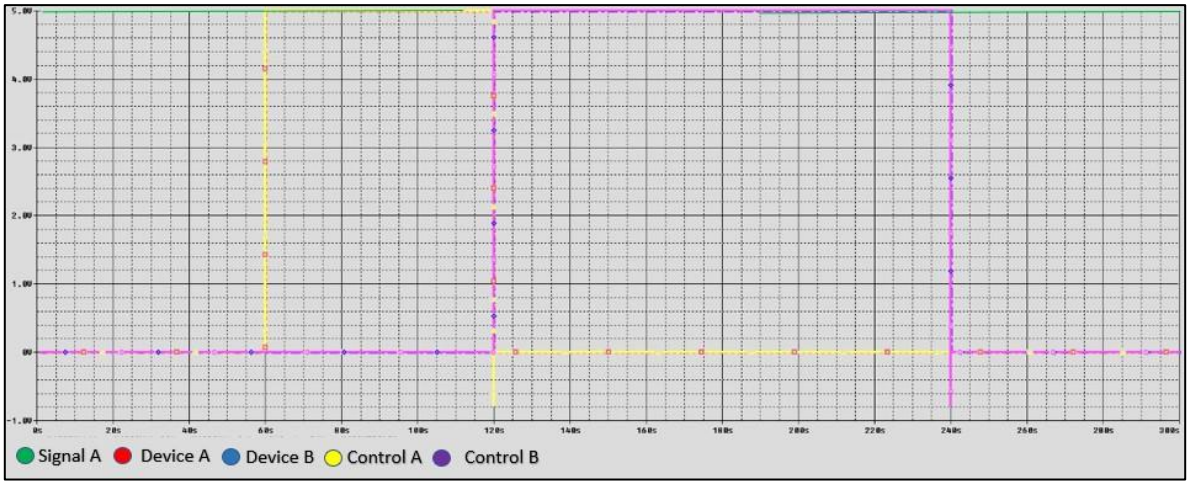


Figura 7.17. Gráfico completo de señales. Fuente: Propia

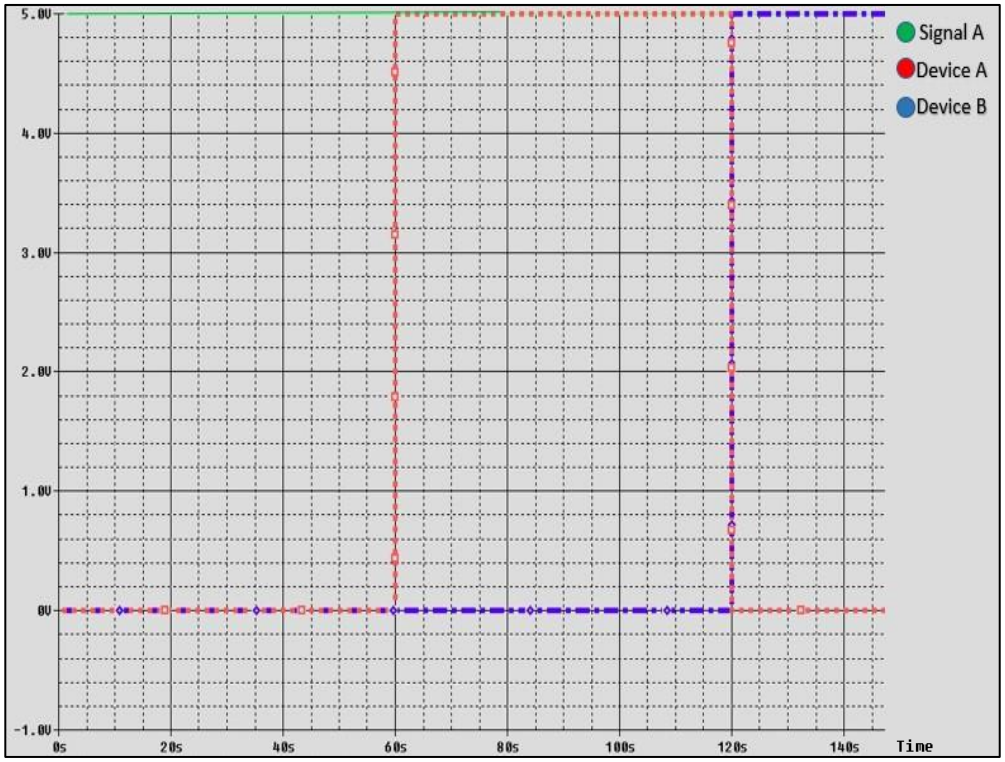


Figura 7.18.Ejemplo de grafico de conmutación entre dispositivos. Fuente: Propia

La intención de estas pruebas es verificar la caída de tensión de la señal al pasar por cada relé DPDT y el consumo de corriente que estos generan para corroborar que la perdida de amperaje se mantiene en los niveles aceptables y estos dispositivos son capaces de conmutar las señales sin modificar de manera significativa su potencia, en la figura 7.19 podemos observar como la señal verde luego de pasar por el primer relé tiene una caída representada en señal roja, esto se repite en cada nivel de relés como se observa en la figura 7.20 que luego del segundo nivel de relés la señal verde tiene otro porcentaje de perdida denotándose así en la señal azul y esto ocurre una tercera vez para el ultimo nivel de relés, calculando estas diferencias y teniendo en cuenta que la impedancia del relé es de 75mOhms podemos calcular la perdida de amperaje, replicando esto para las demás señales se obtuvieron los resultados observados en la tabla 7.10. Es importante recalcar que este comportamiento se evaluó para las señales transmitidas de “COM” a “tiro” y de “tiro” a “COM” concluyendo que sin importar la dirección los resultados de perdida mantenían concordancia.

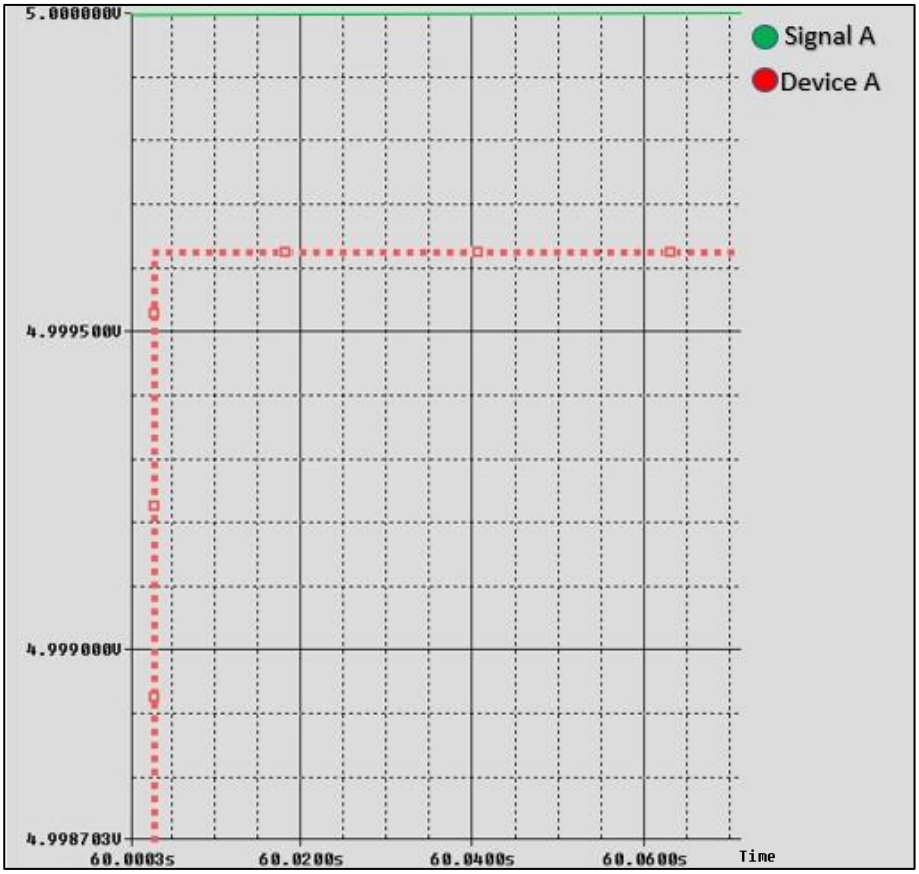


Figura 7.19. Ejemplo de perdida de tensión por un relé. Fuente: Propia

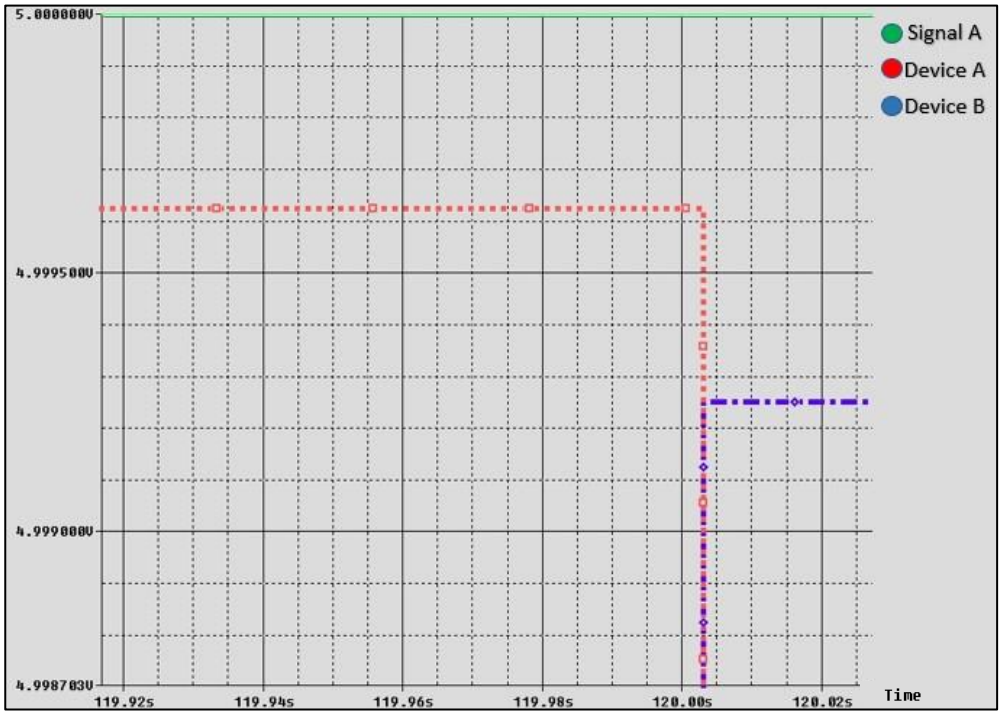


Figura 7.20. Ejemplo de perdida de tensión por un segundo relé. Fuente: Propia

Tabla 7.10 Datos de pérdidas de tensión y amperaje en la conmutación de señales. Fuente: Propia

Vin ± 0.0001 (V)	VoutA ± 0.0001 (V)	VoutB ± 0.0001 (V)	VoutC ± 0.0001 (V)	PerdidaA ± 0.0001 (mA)	PerdidaB ± 0.0001 (mA)	PerdidaC ± 0.0001 (mA)
5.0000	4.9996	4.9992	4.9988	5.3333	10.6667	16.000
4.5000	4.4996	4.4992	4.4988	5.3333	10.6667	16.000
4.0000	3.9996	3.9992	3.9988	5.3333	10.6667	16.000
3.5000	3.4996	3.4992	3.4988	5.3333	10.6667	16.000
3.0000	2.9996	2.9992	2.9988	5.3333	10.6667	16.000
2.5000	2.4996	2.4992	2.4988	5.3333	10.6667	16.000
2.0000	1.9996	1.9992	1.9988	5.3333	10.6667	16.000
1.5000	1.4996	1.4992	1.4988	5.3333	10.6667	16.000
1.0000	0.9996	0.9992	0.9988	5.3333	10.6667	16.000

De la tabla anterior podemos observar que la máxima perdida de corriente es de 16mA y la mayor pérdida de tensión es de 1.2mV, por lo que se puede concluir que se está perdiendo un 0.12% de tensión en las señales lo que para valores de no menos de 1V se puede considerar despreciable, concluyendo de esta manera que la perdida aproximada para señales constantes es aceptable en el diseño de los circuitos empleados, cumpliendo de esta manera con el requerimiento 1 de la tabla 4.2.

Por otro lado, en la figura 7.21 se muestra un ejemplo de la gráfica utilizada para corroborar los tiempos de conmutación entre la activación de la señal de control (amarilla) y la respuesta del relé DPDT (roja) el cual tiene sus propiedades configuradas para que sean equivalentes con los datos técnicos del relé seleccionado así en la figura 7.21 se observa que el retraso es de 3ms.

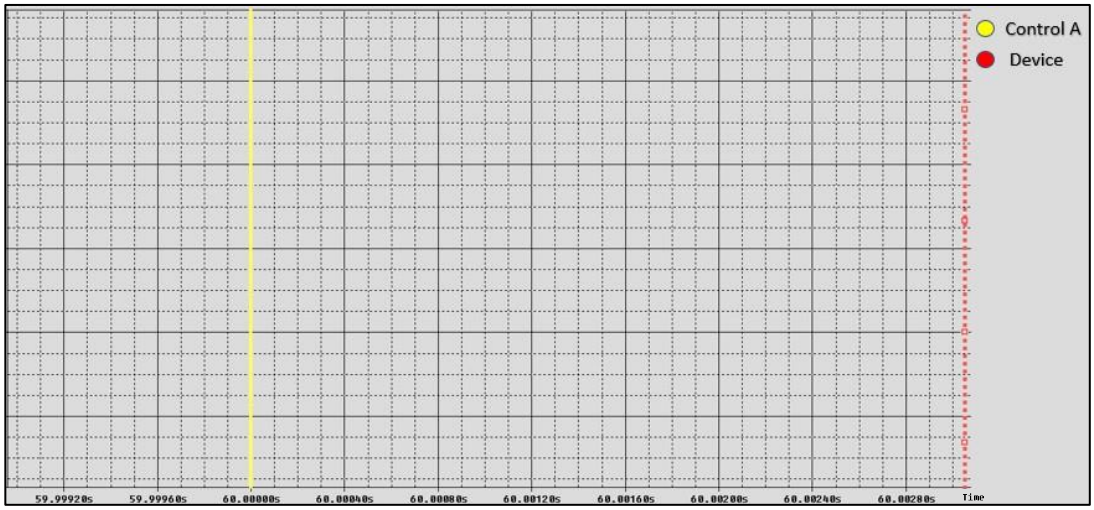


Figura 7.21. Ejemplo de grafica para evaluar la respuesta del relé ante la señal de activación. Fuente: Propia

En la figura 7.22 podemos observar una gráfica ejemplo del instante de desconexión de la señal de control y la respuesta del relé que debido a que para la simulación se está utilizando un relé estándar debido a que la librería del software no presenta un relé compatible con las características de los relés del PXI Chassis los cuales son los encargados de la conexión y desconexión de la señal de control, sin embargo a pesar de que estos generan un rebote de señal de aproximadamente 5uS, la propiedad de interés en esta simulación es el tiempo de conmutación del relé DPDT desde la activación de la señal de control. Esta señal de rebote es despreciable ya que no afecta en el comportamiento de la activación del relé DPDT, como se observa en la figura 7.22 los 3ms son medidos a partir de la primera activación de la señal.

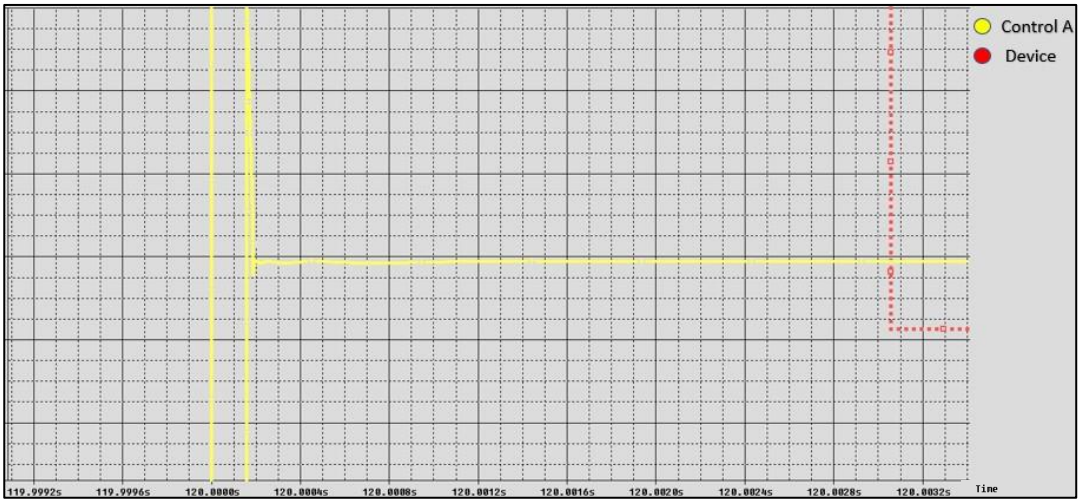


Figura 7.22. Ejemplo de grafica de la respuesta del relé ante la desactivación de la señal de control. Fuente: Propia

También es rescatable ver que luego de la desconexión de cada señal de control, se presenta una tensión negativa en las bobinas, esto confirma lo que teóricamente se esperaba debido a la polarización inversa de un inductor, por lo que manteniendo el diodo polarizado inversamente se protege principalmente a los leds conectados a esta misma señal alargando su tiempo de vida, ya que la resistencia de la bobina es de 178ohms y el voltaje inverso detectado es de aproximadamente 800mV la corriente por polarización inversa que puede llegar a observar ronda en los valores de los mA y teniendo en cuenta que los diodos leds estándar encontrados comercialmente soportan corrientes inversas en los valores de uA en caso de no tener el flyback este se dañaría, así aseguramos que el cumplimiento del requerimiento 12 de tabla 4.2.

Parte de las características de algunas señales de los dispositivos es presentar frecuencias de 100kHZ, por lo que se debe verificar para los distintos niveles de conmutación en los relés que las señales pueden ser conmutadas y transmitidas a dicha frecuencia, es por ello por lo que se simuló para configuraciones de varios niveles la conmutación de este tipo de señales.

Inicialmente se evaluó si a esta frecuencia las pérdidas de tensión presentaban un comportamiento igual a las pruebas con señales constantes, por lo que como resultado podemos observar la figura 7.23 en la cual como se observa las pérdidas son equivalentes y como se podía asumir no dependen de la velocidad

de la señal, por otro lado en esta figura podemos observar que casi alcanzando el nivel máximo de tensión ocurre una distorsión en la señal, sin embargo la frecuencia de la señal se mantiene, lo cual es el factor de mayor importancia y dicha distorsión puede despreciada ya que para los demás niveles de tensión la señal no se distorsiona como se observa en la figura 7.24.

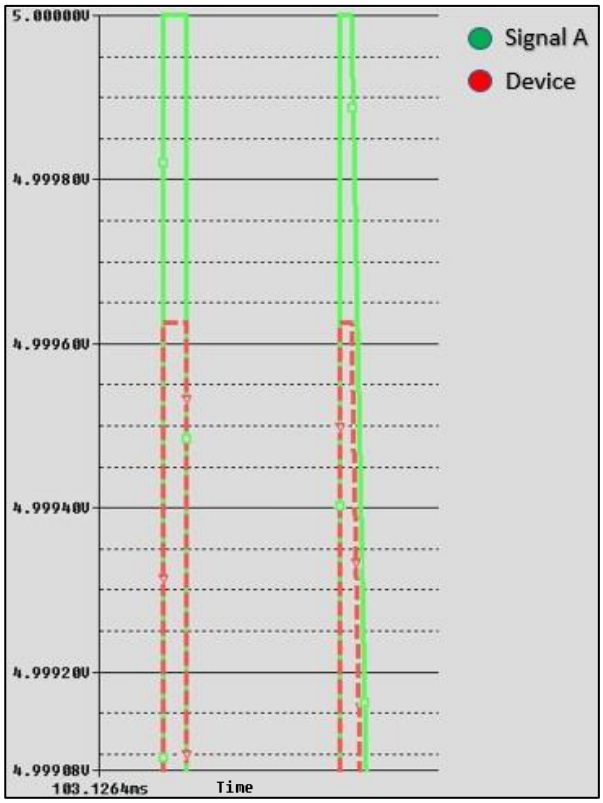


Figura 7.23. Ejemplos de pérdidas para señales de alta frecuencia. Fuente: Propia

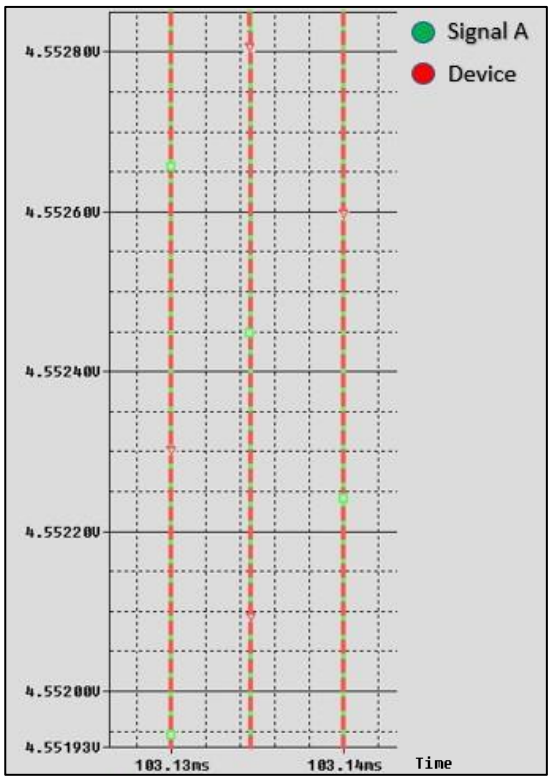


Figura 7.24. Ejemplo de sincronización de la señal de salida con la de entrada. Fuente: Propia

Otra sección relevante a observar es el instante de conmutación, para ello se puede observar en la figura 7.25, la gráfica A muestra la señal de control (azul) se activa aproximadamente a los 100ms, mientras que en la gráfica B podemos denotar que el tiempo de activación de la señal roja es a los 103ms lo cual nos indica que el periodo de conmutación de 3ms se continua respetando, por otro lado al ser el periodo de la señal transmitida menor que el tiempo de conmutación al instante de la activación se genera un desfase de 0.001ms, el cual se corrige 0.005ms luego cuando comienza el siguiente periodo de la señal transmitida, y a partir de este momento continúan manteniendo el mismo periodo, siendo este desfase no significativo.

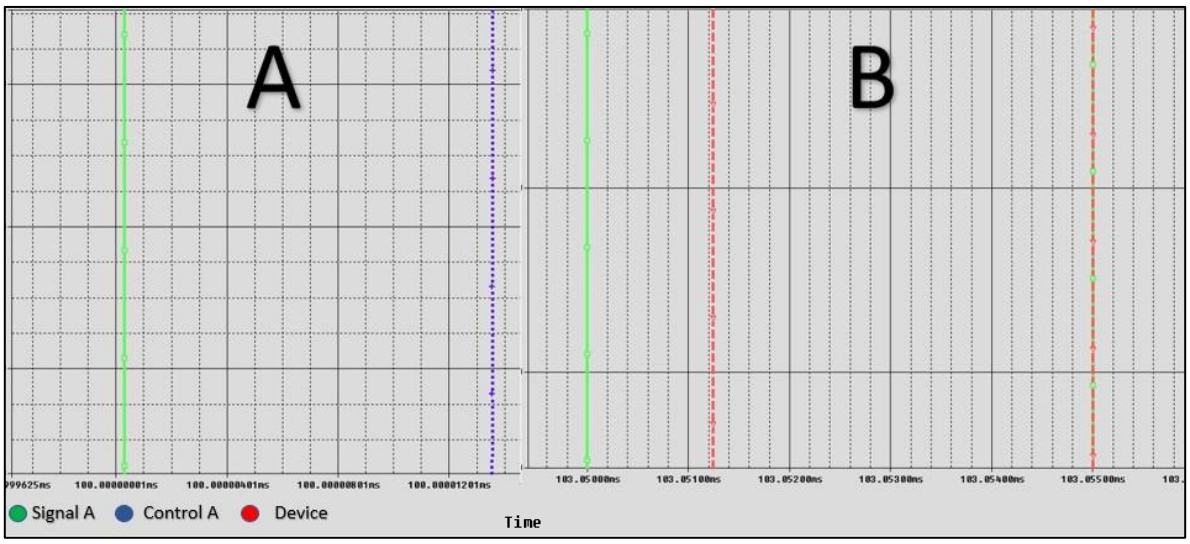


Figura 7.25. Ejemplo de respuesta de relé ante la señal de control a altas frecuencias. Fuente: Propia

Concluyendo de esta manera que el dispositivo utilizado para conmutar es capaz de transmitir este tipo de señales. Este análisis se realizó para las señales después de ser transmitidas hasta por 3 relés y el comportamiento es consistente, incluyendo las pérdidas de tensión y amperaje de esta manera la configuración que permite la conexión de hasta 3 dispositivos como se define en los requerimientos 6 y 8 de la tabla 4.2 queda validada.

Por otro lado, no se puede asumir que todas las señales presentan una forma lineal es por ello por lo que también se realizaron pruebas de señales no lineales a distintas frecuencias, un ejemplo de las gráficas obtenidas en estas pruebas se puede observar en la figura 7.26 la cual representa la gráfica de una señal exponencial y el comportamiento de la señal de salida para un dispositivo ante la activación y desactivación de la señal de control.

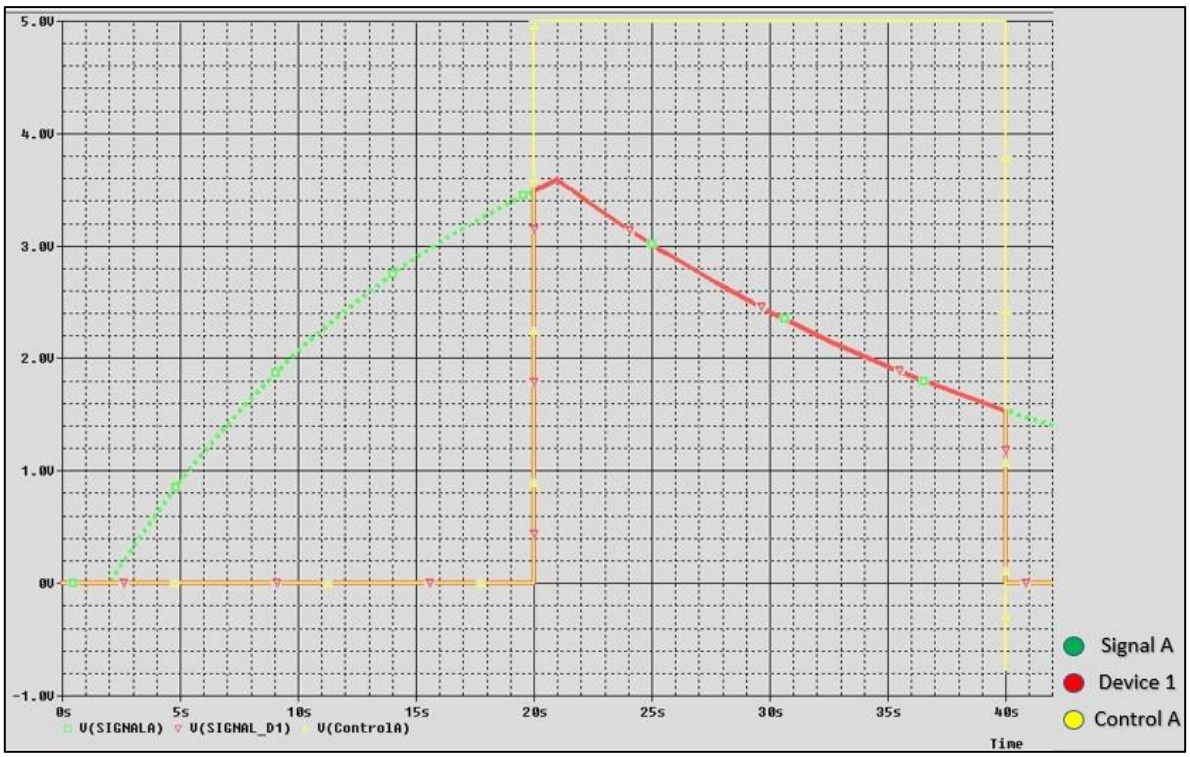


Figura 7.26. Ejemplo de conmutación de señal no lineal. Fuente: Propia

Las pruebas explicadas anteriormente fueron repetidas utilizando diferentes características en simulaciones con un máximo en periodos de máximo 5 minutos, en la tabla 7.11 se observa un resumen de estos resultados en los cuales se verifico si las señales sin presentar perdidas ni distorsiones significativas pudo ser conmutada en hasta 3 niveles de relés, en dirección A (COM a tiro) o B (de tiro a COM), la tabla completa se puede encontrar en la sección de anexos.

Con estas pruebas se puede concluir que con los dispositivos seleccionados y la configuración escogida el sistema es capaz de cumplir con los requerimientos 1, 2, 10 y 11 de la tabla 4.2. Por otro lado, para el cumplimiento de los requerimientos 3, 4 y 5 de la tabla 4.2 nos basamos en las características técnicas de la hoja de datos del modelo de relé seleccionado en el cual nos indica que es capaz de soportar los niveles de corriente indicados.

Tabla 7.11 Segmento de pruebas de verificación para requerimientos de conmutación. Fuente: Propia

Señal	Tensión Max (V)	Frecuencia (kHz)	Relés	Tiempo de Conmutación (ms)	Distorsión	Perdida
Constante	5	N/A	1	3	Aceptable	Aceptable
Constante	5	N/A	2	3	Aceptable	Aceptable
Constante	5	N/A	3	3	Aceptable	Aceptable
Cuadrada	1	60	3	3	Aceptable	Aceptable
Cuadrada	1	80	3	3	Aceptable	Aceptable
Cuadrada	1	100	3	3	Aceptable	Aceptable
No lineal	5	20	1	3	Aceptable	Aceptable
No lineal	5	40	1	3	Aceptable	Aceptable
No lineal	5	60	1	3	Aceptable	Aceptable

7.2.2 Desarrollo de circuito impreso

Una vez desarrollado los esquemáticos de los 3 tipos de circuitos de procedió a confeccionar utilizando la herramienta Allegro PCB la cual es la herramienta asociada a OrCAD para el diseño de circuitos impresos, estos circuitos fueron diseñados teniendo en cuenta las características que indica la norma IPC-2221A que se observan en la tabla 7.12, se esta manera cumpliendo con el requerimiento 13 de la tabla 4.2.

Tabla 7.12 Características de diseño de circuitos impresos. Fuente: Propia

Característica	Valor (unidad)
Numero de capas	2
Tamaño de agujeros de montaje	3.4 (mm)
Grosor de pistas	35 (µm)
Ancho de pista	0.5 (mm)
Grosor de núcleo	1.6 (mm)

En la figura 7.27 se observa el diseño del PCB para las conexiones de los puertos de entrada y salida, como se observa este cuenta con los conectores para distribuir las señales de entrada o salida de hasta 3 dispositivos distintos y en serie con las resistencias y por cada señal de control tenemos los pines donde se conectarán los led cumpliendo así para este circuito impreso con los requerimientos 6, 8, 12 y 14 de la tabla 4.2.

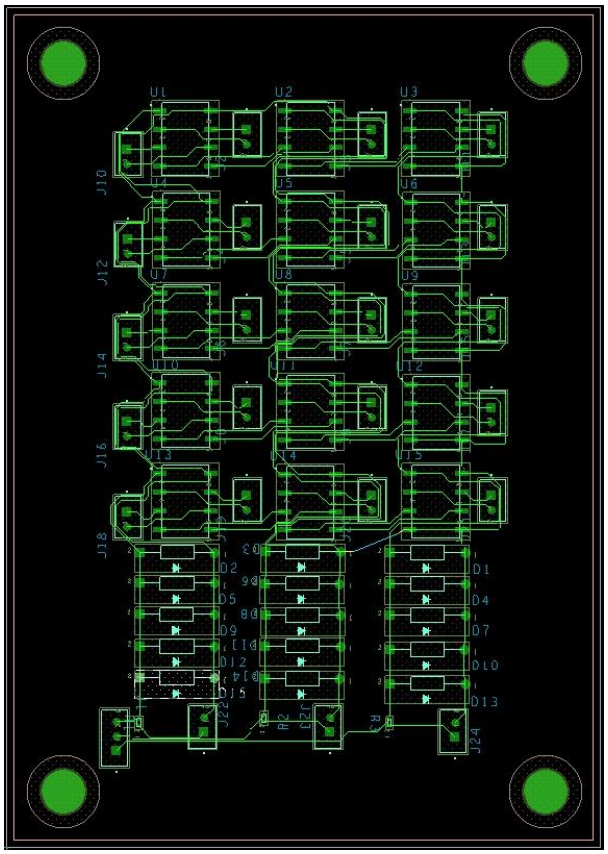


Figura 7.27. Circuito impreso para los puertos de entrada y salida M, A y B. Fuente: Propia

En la figura 7.28 se muestra el diseño PCB para la conmutación de señales al puerto de ablación, contando con la capacidad de conexión de máximo 2 dispositivos distintos y con los pines requeridos para la conexión de 1 led por cada señal de control cumpliendo así con los requerimientos 7, 12 y 14 de la tabla 4.2.

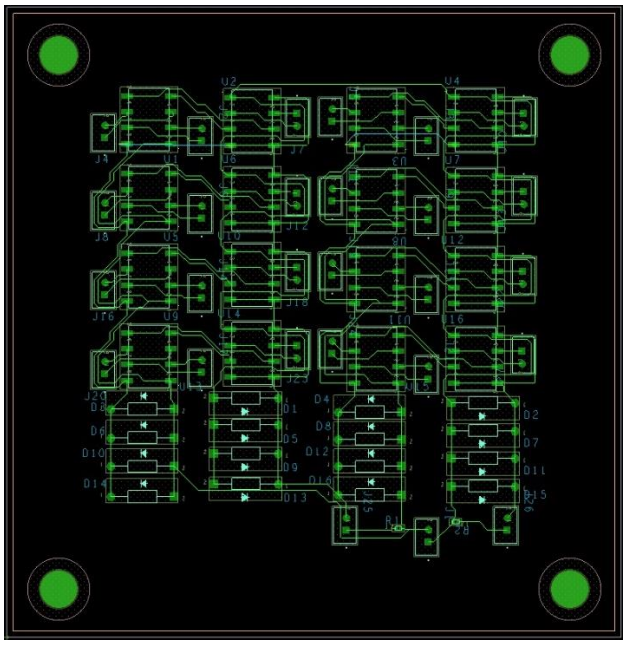


Figura 7.28. Circuito impreso para el puerto de ablación. Fuente: Propia

En la figura 7.29 se muestra el diseño PCB para la conmutación de señales provenientes del generador de ablación, contando con la capacidad de conexión de máximo 2 dispositivos distintos de salida y con los pines requeridos para la conexión de 1 led por cada señal de control cumpliendo así con los requerimientos 9, 12 y 14 de la tabla 4.2.

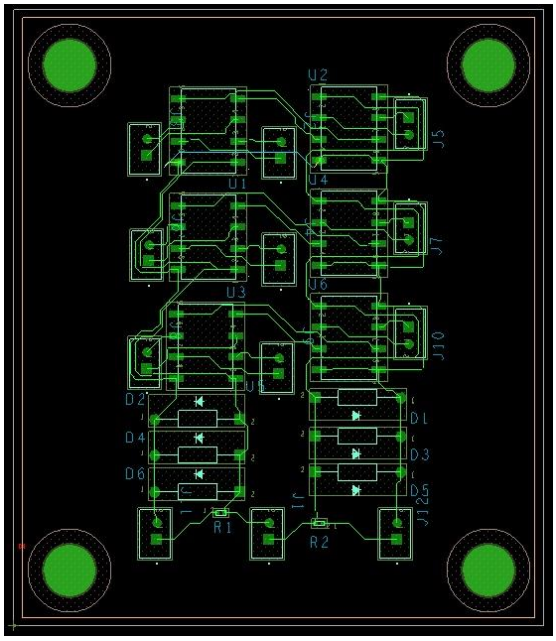


Figura 7.29. Circuito impreso para señales del generador de ablación. Fuente: Propia

7.3 Simulación de accesorio mecánico

7.3.1 Análisis matemático y simulación de estados críticos.

Con el fin de verificar que el diseño desarrollado sea funcional se realizará un análisis matemático de lo que se considera como el punto donde críticamente el diseño podría fallar. Para el diseño seleccionado la contraparte de la cejilla de cierre seleccionada en el diseño del cuerpo de la carcasa, observada en la figura 6.22, es el punto que podría fallar debido a que experimenta fuerzas de tensión y compresión en los momentos de cierre y apertura de la carcasa.

Para su análisis de esfuerzos se considerará este elemento como una viga en voladizo sometida a una carga puntual en el extremo libre, el perfil que presenta es de 147mm de base y 2mm de altura y una longitud de 20mm, además que se deflactará como se observa en la figura 7.30.

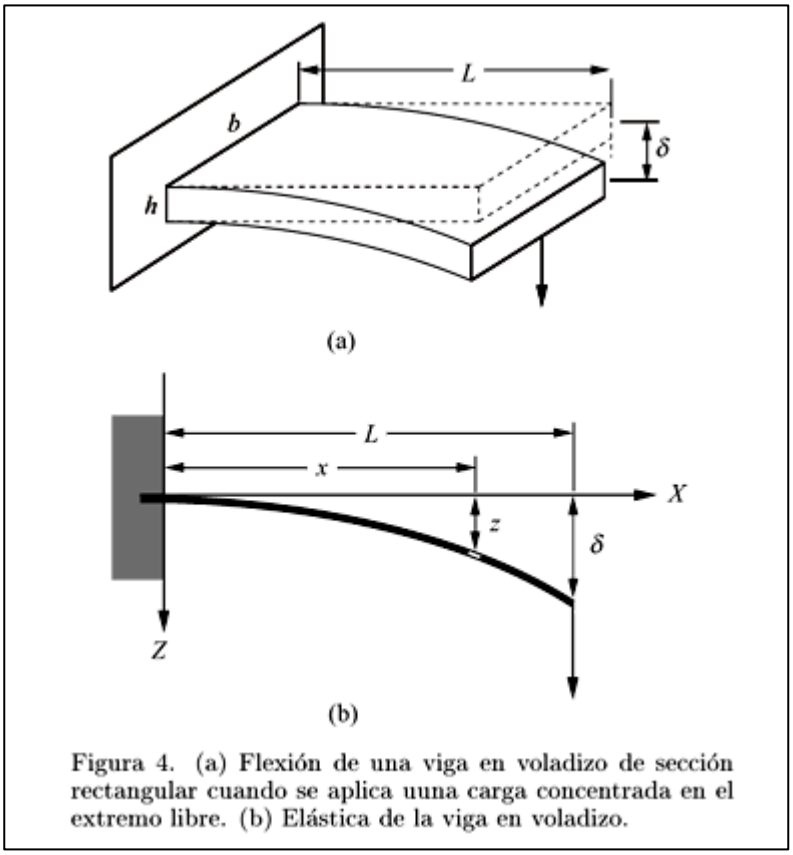


Figura 4. (a) Flexión de una viga en voladizo de sección rectangular cuando se aplica uuna carga concentrada en el extremo libre. (b) Elástica de la viga en voladizo.

Figura 7.30. Deflexión viga en voladizo. [34]

Primeramente, se calculará cuanta es la fuerza requerida para que la pieza en su extremo libre se flexione 1mm el cual es la distancia máxima que debe recorrer la pieza en cada apertura y cierre de la carcasa, como se demuestra en [34] para vías en voladizo con cargas puntuales en sus extremos libres la ecuación que describe la distancia de deflexión es la (7.2) y la ecuación del momento de inercia para vigas cuadradas se observa en la (7.1).

$$I = \frac{base * altura^3}{12} \qquad (7.1)$$

$$y_f = \frac{L^3}{3YI} \qquad (7.2)$$

Donde:

- Y_f: Distancia de flexión
- L: Longitud de la barra
- Y: Modulo de elasticidad
- I: Momento de inercia

Despejando la ecuación y sustituyendo los valores obtenemos que la fuerza requerida para abrir y cerrar la carcasa en la tapa o la base es de 72N debajo de la fuerza promedio en las manos de una persona adulta la cual es de 94N [35], es importante destacar que el módulo de elasticidad fue obtenido de un análisis de resistencia de polímeros para impresión 3D [36]. Asumiendo que el peso de la tapa y la base será de aproximadamente 5N podemos asegurar que no se verá afectado el sellado y transporte por el propio peso de los componentes.

Para el cálculo de fallo por esfuerzos en tensión se utilizará la ecuación (7.3) sustituyendo da como resultado un esfuerzo de 1.05MPa con un factor de seguridad de 2 en los cálculos como se observa en la ecuación (7.5), para el cálculo del esfuerzo a compresión se utilizó la ecuación (7.4) la cual da como resultado un esfuerzo de 1.40MPa con un factor de seguridad de 2 como lo demuestra la ecuación (7.6), ambos esfuerzo se encuentran muy por debajo del límite elástico y esfuerzo ultimo por lo cual se puede concluir que la pieza no se deformará ni fallará cumpliendo de esta manera con una fijación por cejilla de manera segura como lo define el requerimiento 6 de la tabla 4.3.

$$Esf.Tensión = \frac{M * c}{I} * FS \quad (7.3)$$

$$Esf.Compresión = \left(\frac{M * c}{I} + \frac{F}{A}\right) * FS \quad (7.4)$$

$$Esf.Tensión = \frac{(72 * \cos(45) * 0.002) * 0.0005 * 2}{\frac{0.145 * 0.002^3}{12}} = 1.05MPa \quad (7.5)$$

$$Esf.\ Compresión = \frac{(72 * \cos(45) * 0.002) * 0.0005 * 2}{\frac{0.145 * 0.002^3}{12}} + \frac{72 * sen(45) * 2}{0.145 * 0.002} = 1.40MPa \ (7.6)$$

Para darle una mayor fiabilidad al diseño se realizaron múltiples simulaciones de elementos finitos en las cuales se muestran la distribución de esfuerzos y desplazamiento de las partes del componente cuando se ven afectados por las cargas en el instante de apertura y cierre de la carcasa.

Inicialmente de simularon 2 escenarios de esfuerzos para las contrapartes de cierre en el cuerpo de la carcasa, como se observa en la figura 7.31, los máximos esfuerzos que perciben las partes críticas de la estructura son de 4MPa para las fuerzas de apertura, para el cierre de la carcasa los esfuerzos máximos calculados fueron de 4.2MPa en ambos escenarios los esfuerzos se mantienen por debajo del límite elástico. En la figura también se puede observar una flexión máxima de la sección libre de 1.5mm, estos resultados son equivalentes para todas las cejillas del cuerpo de la carcasa, tanto para las fuerzas de cierre como de apertura.

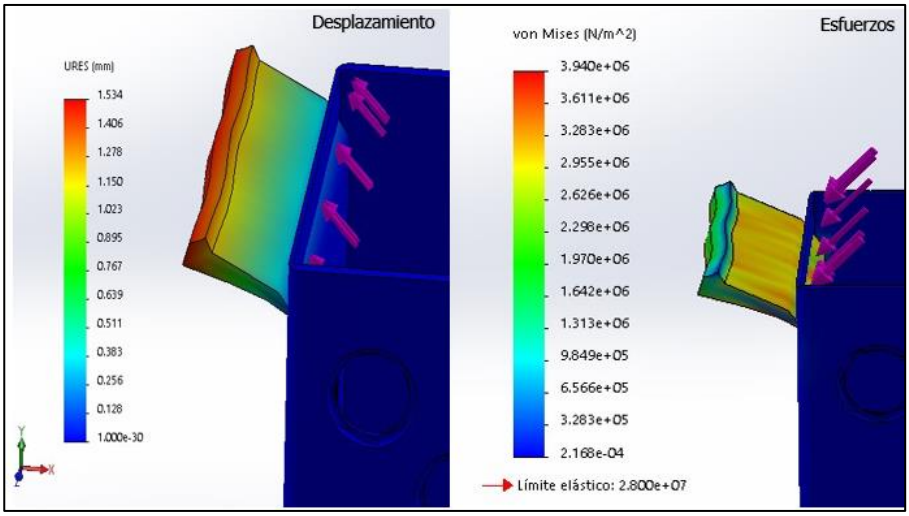


Figura 7.31. Simulación de esfuerzos y desplazamiento en cejilla. Fuente: Propia

También se realizaron simulaciones de la tapa y la base ante las fuerzas de cierre y apertura, en ambos casos se obtuvo que la deformación de las cejillas no superaba los 0.005mm en ambos casos, este valor al ser tan pequeño se considera despreciable además se calculó el factor de seguridad mínimo con von Misses para estas piezas del cual se obtuvo que para las zonas más críticas podría resistir esfuerzos hasta 47 veces más grandes lo que nos da un margen de seguridad muy alto en el diseño, un ejemplo de estos resultados se puede observar en la figura 7.32.

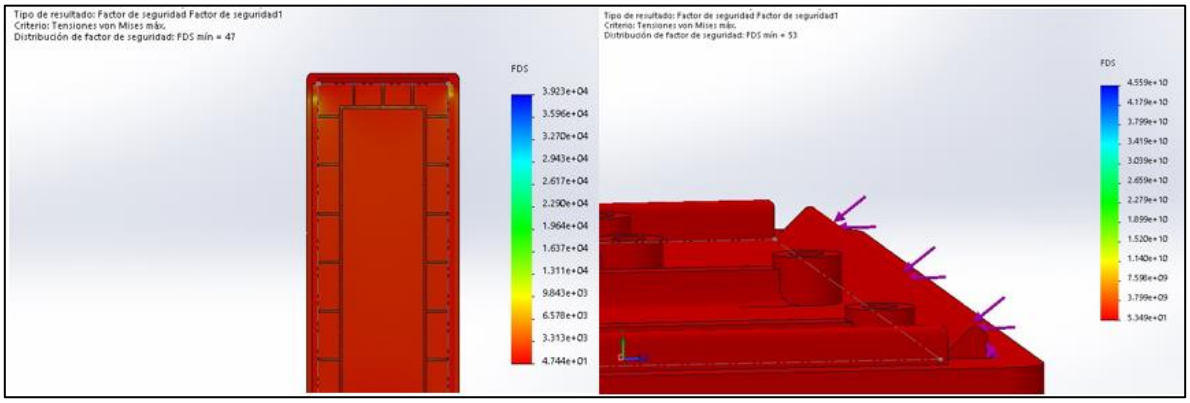


Figura 7.32. Factores de seguridad mínimos aceptables para esfuerzos de apertura y cierre. Fuente: Propia

Para asegurar que la deformaciones de la tapa y la carcasa ante su propio peso pueden ser despreciables y que los esfuerzos en estado estático no afectan al diseño se realizó una simulación de pandeo fijando los extremos y se obtuvo como se muestra en la figura 7.33 que la deformación máxima para la tapa es de 0.6mm

y para la base es de 0.5mm, con estos resultados se debe considerar que los extremos delanteros no fueron fijados para considerar el escenario más crítico de esfuerzos y que la deformación es mínima y practicante despreciable, además los factores de seguridad calculados con von Misses superaban las diez veces concluyendo así que el diseño es funcional.

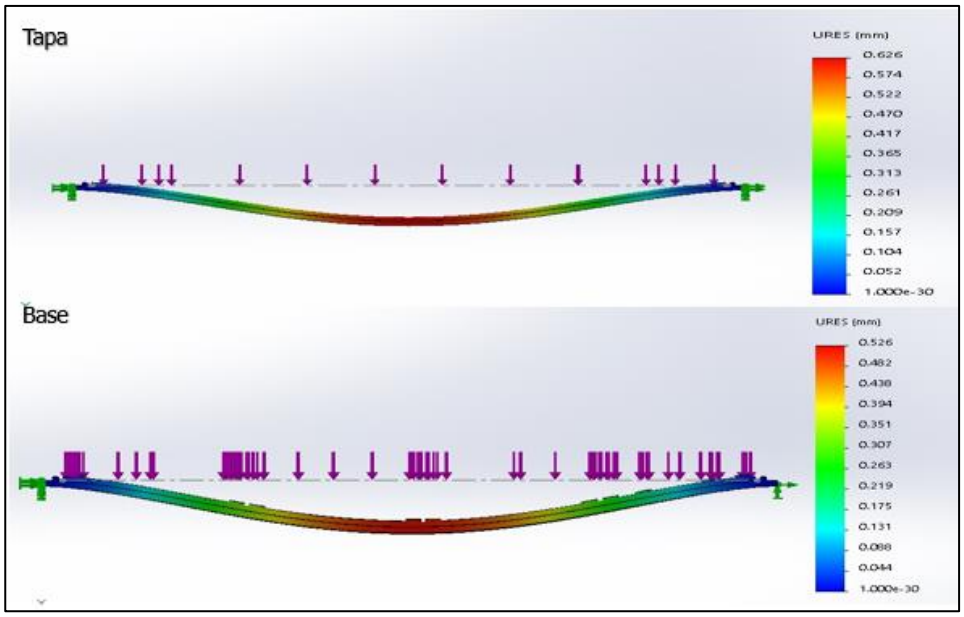


Figura 7.33. Pandeo en estructura debido al propio peso. Fuente: Propia

Finalmente teniendo en consideración que esta es una carcasa de protección se realizaron múltiples simulaciones de impacto desde distintos ángulos, se tomó en cuenta que por la altura promedio de una mesa la estructura la máxima distancia que podría caerse es de 1 metro, tomando la gravedad como 9.8m/s y considerando que la superficie de impacto es rígida los resultados mostraron que el máximo esfuerzo ante los impactos no supera los 7MPa muy por debajo de los esfuerzos limites, cumpliendo así con el requisito primordial de la estructura el cual es de resguardo, en la figura 7.34 se pueden observar ejemplos de estos resultados.

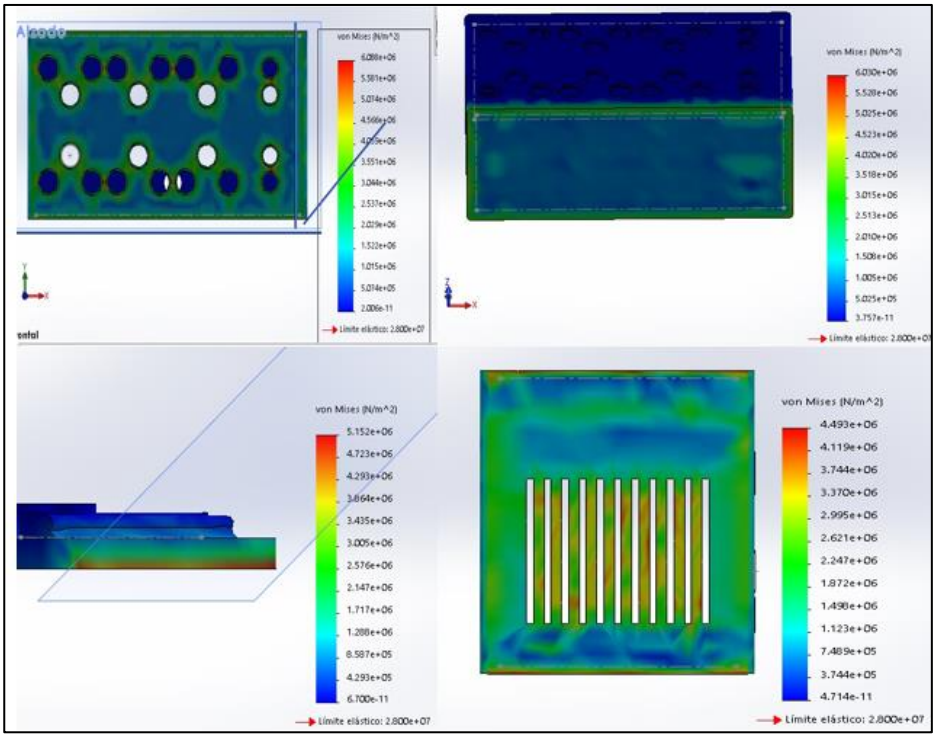


Figura 7.34. Simulaciones de impacto. Fuente: Propia

Unificándolo en un solo ensamble con los demás elementos para verificación del acomodo y distribución de las partes obtenemos el ensamble observado en la figura 7.35 y 7.36 en los cuales podemos observar que todas las partes calzan de manera ideal.

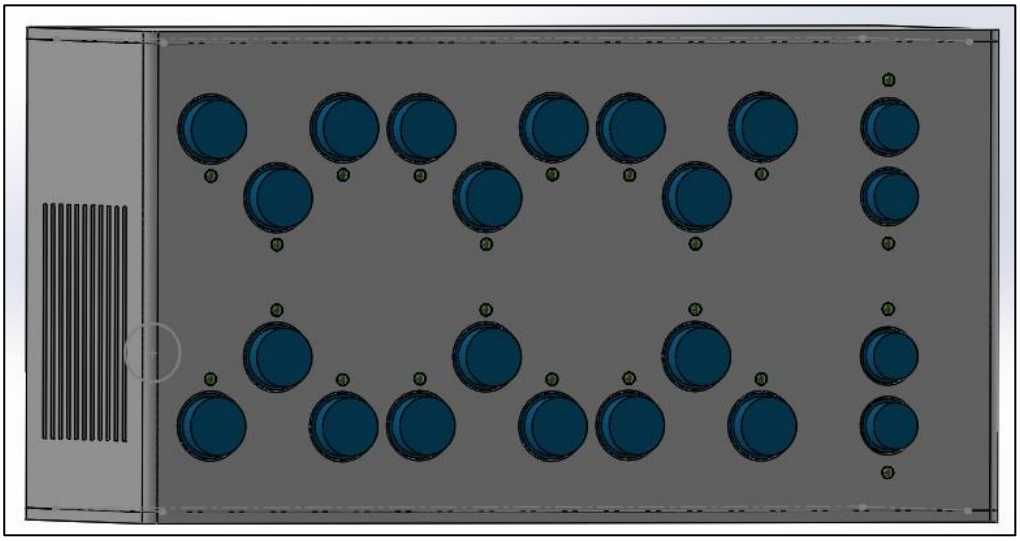


Figura 7.35. Carcasa ensamblada. Fuente: Propia

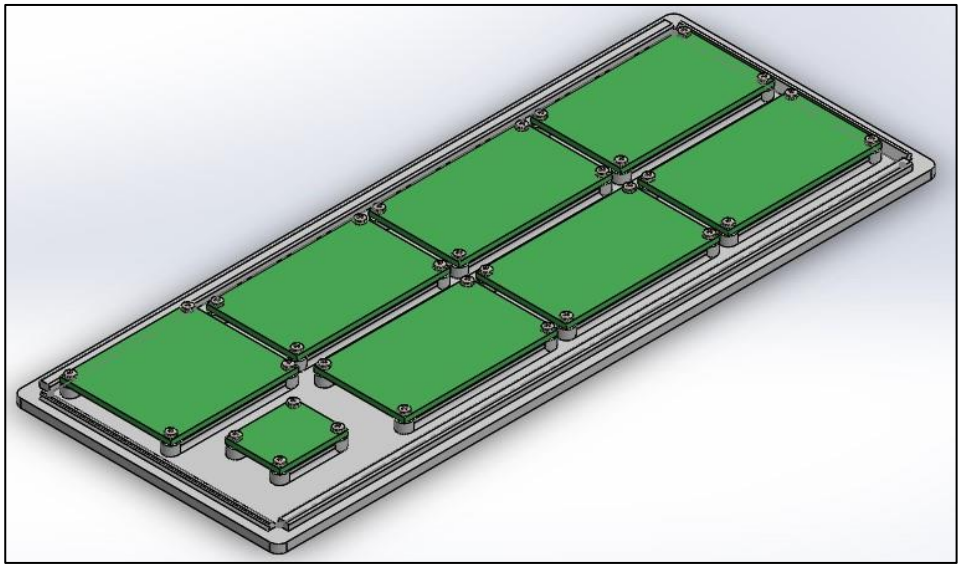


Figura 7.36. Base de ensamble. Fuente: Propia

Por último, se puede considerar que, con los detalles de diseño implementados, los cálculos y simulaciones se obtiene como resultado un diseño de carcasa funcional, fácil de abrir y cerrar para un operador, pero con características mecánicas que aseguran el resguardo y sellado, con todas los puertos requeridos y la ventilación adecuada, los requerimientos de su diseño se cumplen en totalidad como se resume en la tabla 7.13.

Tabla 7.13 Características de diseño asociadas a los requerimientos. Fuente: Propia

Característica	Requerimiento
Base de accesorio con 32 elementos de sujeción con insertos para tornillos M3, distanciados entre si tomando en cuenta las características dimensionales de cada circuito impreso.	1 y 6
Tecnología de manufactura con presión de 4 micras, ajustes deslizantes y holgados en todo el diseño y medidas mínimas mayores a 0.5mm.	2
22 agujeros en cara frontal y 9 en cara trasera, acomodados simétricamente y con espacio para aro de fijación.	3, 4 y 8
Cejilla de cierre en tapa y base con contraparte de perfil negativo en cuerpo de la carcasa, sección rectangular para evitar deslizamiento lateral y trasero. Facilidad de cierre y apertura mediante fuerzas menores a las promedio de una persona adulta, pero garantizando una rigidez suficiente para evitar aperturas no intencionales.	5, 7 y 10.

Rejillas laterales para garantizar el intercambio de calor entre el interior y exterior de la carcasa. Selección de material con resistencia a temperaturas de hasta 100°C.	9 y 12.
Peso menor a los 3kg en el ensamble.	11

7.4 Análisis Económico

En esta sección se describe el presupuesto enfocado a la construcción oficial del entorno de pruebas formalizado por lo que a pesar de que este proyecto lo plantea en un estado de desarrollo y diseño previo a la manufactura se contemplaran materiales y métodos de manufactura para obtener el producto final de diseño, luego se describirá en términos generales las retribuciones que representa la formalización de este entorno para el equipo.

Primeramente, se desglosan los materiales y precios para la manufacturación de los circuitos impresos, estos datos se especifican en la tabla 7.14.

Tabla 7.14 Presupuesto de circuitos impresos. Fuente: Propia

Pieza	Cantidad	Precio Unitario ($)	Precio Total ($)	Proveedor
Relé TQ2SA	112	3.15	353	Mouser Electronics
AMP CT Header 2 ways	177	0.15	27	Mouser Electronics
AMP CT Header 3 ways	6	0.20	1.2	Mouser Electronics
Diodo SM4007PL-TP	112	0.19	22	Mouser Electronics
Resistencia	22	0.50	11	Mouser Electronics
Placa de circuito impreso	8	50	400	-
Total				$ 814

Para el presupuesto del accesorio mecánico se debe considerar que la empresa cuenta con el equipo para impresión 3D, por lo cual se contemplara el costo del material y un 20% de este como cobro por uso de equipos según las políticas de la empresa, además en esta incluirán gastos de los componentes de

ensamblaje como conectores, leds, insertos y tornillería, esto se puede observar en la tabla 7.15.

Tabla 7.15 Presupuesto de accesorio mecánico y ensamble. Fuente: Propia

Pieza	Cantidad	Precio Unitario ($)	Precio Total ($)	Proveedor
Conector Amphenol PL Series	29	60	1740	Amphenol
Inserto SPREDSERT 2	32	0.15	5	AMTEC
Tornillo M3X0.5 L5	32	0.2	6.5	Fastenal
Led DIP verde 5mm	22	0.1	2.2	Mouser Electronics
Filamento de nylon 12-500g	7	50	350	StrataSys
Costo de impresión	-	-	70	-
Total				$ 2180

Finalmente es importante recalcar que para el desarrollo total del proyecto se utilizaron múltiples recursos computacionales de software no libres, estos no se incluyen dentro del presupuesto ya que la empresa contaba con ellos previo al proyecto y fueron facilitados sin coste alguno.

En términos de retribución al equipo y a la empresa el entorno formalizado agilizará las sub-fases de pruebas reduciendo los periodos de testeo, como se mencionó en el apartado 1.2.2 en total se realizan de 2 a 3 sub-fases de pruebas.

Del 100% de las pruebas dependientes de hardware el entorno de ejecución automatizado presentará la infraestructura y soporte para un 45% de estas pruebas, ahorrando en todo el proceso de testeo formal un total de 15225 minutos que equivalen a 31 días de trabajo, haciendo la ejecución de las pruebas dependientes de hardware 35% más rápidas, en la tabla 7.16 podemos encontrar una segmentación de los tiempos de ejecución de las pruebas dependientes de hardware soportadas por el entorno de automatización con y sin ejecución automatizada para cada fase y en la tabla 7.17 podemos encontrar el ahorro en días laborales debido al entorno de automatización, en esta sección de debe considerar

que manualmente las horas laborales son 8 mientras que de manera automatizada se pueden ejecutar pruebas durante las 24 horas del día.

Tabla 7.16 Desglosé de tiempos por fase de pruebas dependientes de hardware. Fuente: Propia

Pruebas dependientes de hardware (%)	Tiempo de ejecución manual total (min)	Tiempo de ejecución automatizado total (min)	Periodo	Tiempo ahorrado (min)
45	16 240	4060	Sub-fase 1	12180
20	3248	812	Sub-fase 2	2436
5	812	203	Sub-fase 3	609
Total, de tiempo ahorrado				15225

Tabla 7.17 Tiempo de pruebas total ahorrado con entorno automatizado. Fuente: Propia

Tiempo total de testeo sin automatización (días laborales)	Tiempo total de testeo con automatización (días laborales)	Tiempo ahorrado (días laborales)
94	56	39

Este ahorro se refleja en términos monetarios para el equipo como se muestra en la tabla 7.18, para estos cálculos se debe tomar en cuenta que el trabajo de pruebas se realiza 50% en Costa Rica y 50% en Estados Unidos por lo que se calculará utilizando el costo de horas ingeniero de ambos países.

Tabla 7.18 Ahorro monetario por automatización de pruebas dependientes de Hardware. Fuente: Propia

Tiempo ahorrado anual (Horas)	**254**
Costo hora ingeniero EU ($)	80
Costo de hora Ingeniero CR ($)	40
Dinero anual ahorrado ($)	15 240

8 Conclusiones

- Se elaboró un código de paradigma modular que evalúa el estado y la conexión de 4 dispositivos y 71 relés en un tiempo de ejecución de 30 segundos, eliminando en un 100% la ejecución de pruebas invalidas por fallos del entorno y ahorrando un promedio de 10 minutos, por prueba, de recursos de sistema cuando el entorno de ejecución presenta algún desperfecto.
- Se diseñó el sistema de evaluación para que el proceso de verificación en todas sus etapas sea documentado y retorne una hoja resumen con todos los resultados obtenidos ordenados por orden de proceso ejecutado e indicando el nivel de importancia de cada resultado para garantizar una comprensión intuitiva del operador.
- Se diseñaron 8 placas de circuitos impresos, bajo la norma IPC-2221A, basadas en una configuración cascada de relés DPDT que permiten las transmisión y conmutación de señales con pérdidas menores al 0.12% para señales constantes, escalonadas y no lineales con valores de máximos de 5V y 100kHz de frecuencia en tiempos máximos de 3ms, 160% más veloces que los requeridos.
- Se diseñó un accesorio mecánico para el resguardo de 8 placas de circuitos impresos cumpliendo con todos los requerimientos de diseño, con un factor de seguridad mayor a 3 para las temperaturas esperadas, resistencia a los impactos por caídas menores a 1 metro con un factor de seguridad de 4 y resistencia a esfuerzos internos con un factor de seguridad de 7.
- Se consideró la resistencia, ergonomía y portabilidad como las características más importantes en el diseño del accesorio de resguardo lo que resulto en una carcasa con 2 configuraciones para mantenimiento interno, sellado al polvo y facilidad de apertura requiriendo una fuerza 26% menor que la máxima promedio de una persona entre los 17 y 35 años, de dimensiones máximas 388x156x210mm con un peso de 2.5kg.

9 Recomendaciones

- El sistema de evaluación actual evalúa las condiciones del hardware y sus interacciones, para el futuro se recomienda evaluar la posibilidad de la implementación de un código que tenga comunicación con la interfaz de programación de la aplicación para incluir en la evaluación la respuesta del programa ante las pruebas del entorno de ejecución.
- La configuración y diseño del sistema de conmutación actual fue validada por medio de la simulación, se recomienda que previo a su manufactura se construya un prototipo físico al cual se le puedan conectar las señales correspondientes y se reevalúe el comportamiento del sistema.
- Consecutivo a la manufacturación de las partes se recomienda que estas se unifiquen en un sistema sobre el cual se pueda desarrollar un nuevo protocolo de testeo que tome en cuenta las realizadas individualmente e incluya un periodo de probatorio en el cual se estén ejecutando continuamente pruebas dependientes de hardware para evaluar el comportamiento del entorno y se pueda asegurar que mantiene una estabilidad suficiente para ser formalizado.
- También se sugiere implementar un método de evaluación del sistema de conmutación de los circuitos impresos para poder rastrear algún posible desperfecto en los componentes.
- Una vez formalizado el entorno de pruebas se recomienda continuar con una ampliación de este para agregar soporte a los componentes de hardware que no fueron incluidos como el pedal para mapeo, generador de campo magnético, entre otros, esto repercutiría en la creación de nuevos circuitos impresos y una modificación del accesorio de resguardo, para lo cual se debe hacer una reevaluación de las partes para asegurar que mantiene su resistencia, portabilidad y ergonomía.

10 Referencias Bibliográficas

[1] *Rhythmia HDx Mapping System, Software Direction for Use*, 1st ed. Boston Scientific, 2018, pp. 4-45.

[2] *Workbook 1 module 1 Cardiac Functions*, 2nd ed. Endocardial Solutions, Inc, 2005, p. 7.

[3] *Workbook 2 module 2 Electrophysiologic Studies*, 2nd ed. Endocardial Solutions, Inc, 2005, p. 2-4.

[4] M. Rodríguez Morales, M. Cabrerizo Sanz and M. Matas Avellà, *Manual de enfermería en arritmias y electrofisiología*. [Madrid]: Asociación Española de Enfermería en Cardiología, 2013, pp. 218-236.

[5] J. García and J. Rodríguez, "Ablación con catéter de punta irrigada", vol. 14, 2006. [Accessed 5 September 2021].

[6] P. Meléndez and A. Farid, *Plataforma NI PXI*, 1st ed. Barranquilla: Editorial Universitaria de la Costa EDUCOSTA, 2015, pp. 11-17.

[7] D. Cheij, *AN INSTRUMENT OBSOLESCENCE REPLACEMENT SOLUTION USING A PXI-SASED SYSTEM*. Austin: National Instruments, 2004.

[8] T. Maxwell, K. Patil, S. Bayne and R. Gale, *Hardware-in-the-loop Testing of GM Two-Mode Hybrid Electric Vehicle*. Lubbock.

[9] "PXI Systems Alliance - Home", *Pxisa.org*, 2021. [Online]. Available: http://www.pxisa.org/Default.aspx. [Accessed: 27- Sep- 2020].

[10]*¿Qué es PXI? - National Instruments*, 1st ed. National Instruments, 2017.

[11]*Rhythmia HDx Mapping System, Hardware Direction for Use*, 1st ed. Boston Scientific, 2018, pp. 4-45.

[12] H. Lee, *EPD PCA SIS INTERCONNECT FRONT,* Boston Scientific, Inc, 2016, pp.1-4.

[13]A. Leithold, *Structured testing in practice*, 1st ed. 2008, p. 53.

[14] K. Fok, *RTMD Hardware Package Manual Description*.

[15] *NI Modular Instruments Python API Documentation*, 3rd ed. National Instruments, 2020, p. 214.

[16]A. Álvarez and J. Calle, *Metodologías de Testing de Software y su Aplicación en el Centro de Informática de la Universidad EAFIT*. Medellín: Universidad EAFIT, 2009, pp. 43-70.

[17] K. Ulrich, S. Epinger and R. Madrigal Álvarez, *Diseño y desarrollo de productos*. México: McGraw-Hill, 2004.

[18] G. Pantaleo and L. Rinaudo, *Ingeniería de software*. Buenos Aires, Argentina: Alfaomega Grupo Editor Argentino, 2015.

[19] C. Rodríguez, *Paradigmas de Programación*. Departamento de Informática Universidad de Valladolid, 2011.

[20] "Logging Cookbook — Python 3.9.2 documentation", *Docs.python.org*, 2021. [Online]. Available: https://docs.python.org/3/howto/logging-cookbook.html. [Accessed: 01- Oct- 2020].

[21]*Scicomp.ethz.ch*, 2021. [Online]. Available: https://scicomp.ethz.ch/public/manual/Python/3.4.3/howto-logging-cookbook.pdf. [Accessed: 03- Oct- 2020].

[22] "argparse — Analizador sintáctico (Parser) para las opciones, argumentos y sub-comandos de la línea de comandos — documentación de Python - 3.9.2", *Docs.python.org*, 2021. [Online]. Available: https://docs.python.org/es/3/library/argparse.html. [Accessed: 05- Oct- 2020].

[23] "Tutorial de Argparse — documentación de Python - 3.8.8", *Docs.python.org*, 2021. [Online]. Available: https://docs.python.org/es/3.8/howto/argparse.html. [Accessed: 07- Oct- 2020].

[24] E. Bahit, *Análisis de argumentos enviados por línea de comandos mediante python con argparse*, 1st ed. Creative Commons BY-NC-SA, 2013.

[25]"pxssh - control an SSH session — Pexpect 4.8 documentation", *Pexpect.readthedocs.io*, 2021. [Online]. Available: https://pexpect.readthedocs.io/en/stable/api/pxssh.html. [Accessed: 11- Oct- 2020].

[26] "Examples — Pexpect 4.8 documentation", *Pexpect.readthedocs.io*, 2021. [Online]. Available: https://pexpect.readthedocs.io/en/stable/examples.html. [Accessed: 11- Oct- 2020].

[27] C. Guagliano, *Programación en Python II: Programación orientada a objetos*. RedUsers, 2019.

[28] "8. Errores y excepciones — documentación de Python - 3.9.2", *Docs.python.org*, 2021. [Online]. Available: https://docs.python.org/es/3/tutorial/errors.html. [Accessed: 14- Oct- 2020].

[29] A. Delgado and J. Mira, *Conmutadores Analógicos, Multiplexores y Circuitos de Muestreo y Retención*, 1st ed. Barcelona, 2017.

[30] H. Ott, *Noise Reduction Techniques in Electronic Systems*. New York: John Wiley & Sons, 1988.

[31] A. Estrada, *Convertidor tipo Flyback CFP para Iluminación Led*. Cantabria: Universidad de Cantabria, 2018, pp. 15-18.

[32] *TQ2SA-5V datasheet*. Panasonic, 2014.

[33] P. Caravaca, *Influencia de las condiciones de trabajo en piezas generadas por 3D*, 1st ed. Cartagena: Universidad Politécnica de Cartagena, 2019, p. 40.

[34] T. Belendez, C. Neipp and A. Belendez, "Estudio de la Flexión de una Viga de Material Elástico no Lineal", *Revista Brasileira de Ensino de Física*, vol. 24, no. 4, 2002. [Accessed 13 January 2021].

[35] P. Escalona, J. Naranjo, V. Lagos and F. Solís, "Parámetros de Normalidad en Fuerzas de Prensión de Mano en Sujetos de Ambos Sexos de 7 a 17 Años", *Revista Chilena de Pediatría*, no. 5, 2009.

[36] A. Zolfagharian, M. Khosravani and A. Kaynak, "Fracture Resistance Analysis of 3D-Printed Polymers", *Polymers*, vol. 12, no. 2, p. 302, 2020. Available: 10.3390/polym12020302 [Accessed 7 March 2021].

11 Apéndices

11.1 Información de Proyecto

Información del estudiante

- Nombre: Daniel Aguirre Sosa
- Cédula: 6 04330173
- Carné: 2014067945
- Dirección de su residencia en época lectiva: Entra Costado Este de la entrada Sur del Tecnológico de Cartago.
- Dirección de su residencia en época no lectiva: Esparza, Puntarenas distrito del espíritu santo.
- Email: danieldj.aguirre@gmail.com
- Celular: 8776-5303

Información de la empresa

- Nombre: Boston Scientific.
- Dirección: -
- Teléfono: -

Información del asesor

- Nombre: Ing. Rodolfo Piedra Camacho.
- Rango: Profesora del Área académica de Ingeniería en Mecatrónica.
- Teléfono: +506 8874-8809
- Email: fofo.piedra@gmail.com

Información del proyecto

- Nombre: Desarrollo de un evaluador previo a la aplicación de pruebas automatizadas, diseño de circuitos impresos y accesorio mecánico para el montaje de una interfaz formal
- Ubicación: -

11.2 Tabla de pruebas de funcionalidad

Tabla 11.1 Pruebas para función getRelayFromConnector. Fuente: Propia

Descripción	Pruebas	Resultados esperados	Resultados Obtenidos
Esta función se encarga de obtener la lista de relés de cada conector.	Se ejecuta para el conector PIU.	**Salida:** Retorna 0. **Relés:** [<Relays.K00: 'k0'>]	**Salida:** Retorna 0. **Relés:** [<Relays.K00: 'k0'>]
	Se ejecuta para el conector Conditioning.	**Salida:** Retorna 0. **Relés:** [<Relays.K50: 'k50'>]	**Salida:** Retorna 0. **Relés:** [<Relays.K50: 'k50'>]
	Se ejecuta para el conector Conditioning. Se ejecuta para el conector Maestro_4000	**Salida:** Retorna -1. **Salida:** Retorna 0. **Relés:** [<Relays.K85: 'k85'>, <Relays.K99: 'k99'>, <Relays.K83: 'k83'>, <Relays.K91: 'k91'>, <Relays.K89: 'k89'>, <Relays.K93: 'k93'>, <Relays.K95: 'k95'>, <Relays.K87: 'k87'>, <Relays.K97: 'k97'>, <Relays.K02: 'k2'>]	**Salida:** Retorna -1. **Salida:** Retorna 0. **Relés:** [<Relays.K85: 'k85'>, <Relays.K99: 'k99'>, <Relays.K83: 'k83'>, <Relays.K91: 'k91'>, <Relays.K89: 'k89'>, <Relays.K93: 'k93'>, <Relays.K95: 'k95'>, <Relays.K87: 'k87'>, <Relays.K97: 'k97'>, <Relays.K02: 'k2'>]
	Se ejecuta para el conector Back_patch	**Salida:** Retorna 0. **Relés:** [<Relays.K75: 'k75'>, <Relays.K71: 'k71'>, <Relays.K73: 'k73'>, <Relays.K80: 'k80'>, <Relays.K82: 'k82'>]	**Salida:** Retorna 0. **Relés:** [<Relays.K75: 'k75'>, <Relays.K71: 'k71'>, <Relays.K73: 'k73'>, <Relays.K80: 'k80'>, <Relays.K82: 'k82'>]
	Se ejecuta para el conector ECG_IN	**Salida:** Retorna 0. **Relés:** [<Relays.K70: 'k70'>, <Relays.K72: 'k72'>, <Relays.K68: 'k68'>, <Relays.K66: 'k66'>, <Relays.K64: 'k64'>, <Relays.K62: 'k62'>,	**Salida:** Retorna 0. **Relés:** [<Relays.K70: 'k70'>, <Relays.K72: 'k72'>, <Relays.K68: 'k68'>, <Relays.K66: 'k66'>, <Relays.K64: 'k64'>, <Relays.K62: 'k62'>,

		<Relays.K74: 'k74'>, <Relays.K58: 'k58'>, <Relays.K78: 'k78'>, <Relays.K76: 'k76'>, <Relays.K60: 'k60'>]	<Relays.K74: 'k74'>, <Relays.K58: 'k58'>, <Relays.K78: 'k78'>, <Relays.K76: 'k76'>, <Relays.K60: 'k60'>]
	Se ejecuta para el conector A_IN	**Salida:** Retorna 0. **Relés**: [<Relays.K19: 'k19'>, <Relays.K39: 'k39'>, <Relays.K21: 'k21'>, <Relays.K23: 'k23'>, <Relays.K27: 'k27'>, <Relays.K29: 'k29'>, <Relays.K31: 'k31'>, <Relays.K41: 'k41'>, <Relays.K33: 'k33'>, <Relays.K15: 'k15'>, <Relays.K17: 'k17'>, <Relays.K37: 'k37'>, <Relays.K35: 'k35'>, <Relays.K07: 'k7'>, <Relays.K09: 'k9'>, <Relays.K11: 'k11'>, <Relays.K13: 'k13'>, <Relays.K25: 'k25'>, <Relays.K01: 'k1'>, <Relays.K03: 'k3'>, <Relays.K05: 'k5'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K19: 'k19'>, <Relays.K39: 'k39'>, <Relays.K21: 'k21'>, <Relays.K23: 'k23'>, <Relays.K27: 'k27'>, <Relays.K29: 'k29'>, <Relays.K31: 'k31'>, <Relays.K41: 'k41'>, <Relays.K33: 'k33'>, <Relays.K15: 'k15'>, <Relays.K17: 'k17'>, <Relays.K37: 'k37'>, <Relays.K35: 'k35'>, <Relays.K07: 'k7'>, <Relays.K09: 'k9'>, <Relays.K11: 'k11'>, <Relays.K13: 'k13'>, <Relays.K25: 'k25'>, <Relays.K01: 'k1'>, <Relays.K03: 'k3'>, <Relays.K05: 'k5'>]
	Se ejecuta para el conector M_IN	**Salida:** Retorna 0. **Relés**: [<Relays.K61: 'k61'>, <Relays.K63: 'k63'>, <Relays.K65: 'k65'>, <Relays.K67: 'k67'>, <Relays.K69: 'k69'>, <Relays.K55: 'k55'>, <Relays.K53: 'k53'>, <Relays.K57: 'k57'>, <Relays.K51: 'k51'>, <Relays.K59: 'k59'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K61: 'k61'>, <Relays.K63: 'k63'>, <Relays.K65: 'k65'>, <Relays.K67: 'k67'>, <Relays.K69: 'k69'>, <Relays.K55: 'k55'>, <Relays.K53: 'k53'>, <Relays.K57: 'k57'>, <Relays.K51: 'k51'>, <Relays.K59: 'k59'>]

	Se ejecuta para el conector: ABL_BOX	**Salida:** Retorna 0. **Relés**: [<Relays.K08: 'k8'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K08: 'k8'>]
	Se ejecuta para el conector: ABL_CATHETER	**Salida:** Retorna 0. **Relés**: [<Relays.K10: 'k10'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K10: 'k10'>]
	Se ejecuta para el conector: M_OUT	**Salida:** Retorna 0. **Relés**: [<Relays.K90: 'k90'>, <Relays.K94: 'k94'>, <Relays.K92: 'k92'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K90: 'k90'>, <Relays.K94: 'k94'>, <Relays.K92: 'k92'>]
	Se ejecuta para el conector: A_OUT	**Salida:** Retorna 0. **Relés**: [<Relays.K88: 'k88'>, <Relays.K84: 'k84'>, <Relays.K86: 'k86'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K88: 'k88'>, <Relays.K84: 'k84'>, <Relays.K86: 'k86'>]
	Se ejecuta para el conector: B_OUT	**Salida:** Retorna 0. **Relés**: [<Relays.K52: 'k52'>, <Relays.K56: 'k56'>, <Relays.K54: 'k54'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K52: 'k52'>, <Relays.K56: 'k56'>, <Relays.K54: 'k54'>]
	Se ejecuta para el conector: ECG_OUT	**Salida:** Retorna 0. **Relés**: [<Relays.K98: 'k98'>]	**Salida:** Retorna 0. **Relés**: [<Relays.K98: 'k98'>]

11.3 Hoja de resultados de software de evaluación

[fecha] [hora] [Tipo de información] ---Información de PXI Chassis---

[fecha] [hora] [Tipo de información] ---Resultados de verificación de conexión---

[fecha] [hora] [Tipo de información] --- Lista de módulos a evaluar---

[fecha] [hora] [Tipo de información] ---Relé evaluado de cada puerto---

[fecha] [hora] [Tipo de información] ---Estado del relé y acciones de evaluación---

[fecha] [hora] [Tipo de información] ---Resumen de Resultados---

[fecha] [hora] [Tipo de información] ---Dirección de hoja de resultados---

[fecha] [hora] [Tipo de información] ---Resumen del estado final de cada módulo---

11. 4 Tabla de verificación de conmutador de señales

Tabla 11.2 Pruebas para verificación de requerimientos de conmutación. Fuente: Propia

Señal	Tensión Max (V)	Frecuencia (kHz)	Relés	Tiempo de Conmutación (ms)	Distorsión	Perdida
Constante	5	N/A	1	3	Aceptable	Aceptable
Constante	5	N/A	2	3	Aceptable	Aceptable
Constante	5	N/A	3	3	Aceptable	Aceptable
Constante	4	N/A	1	3	Aceptable	Aceptable
Constante	4	N/A	2	3	Aceptable	Aceptable
Constante	4	N/A	3	3	Aceptable	Aceptable
Constante	3	N/A	1	3	Aceptable	Aceptable
Constante	3	N/A	2	3	Aceptable	Aceptable
Constante	3	N/A	3	3	Aceptable	Aceptable
Constante	2	N/A	1	3	Aceptable	Aceptable
Constante	2	N/A	2	3	Aceptable	Aceptable
Constante	2	N/A	3	3	Aceptable	Aceptable
Constante	1	N/A	1	3	Aceptable	Aceptable
Constante	1	N/A	2	3	Aceptable	Aceptable
Constante	1	N/A	3	3	Aceptable	Aceptable
Cuadrada	5	20	1	3	Aceptable	Aceptable
Cuadrada	5	40	1	3	Aceptable	Aceptable
Cuadrada	5	60	1	3	Aceptable	Aceptable
Cuadrada	5	80	1	3	Aceptable	Aceptable
Cuadrada	5	100	1	3	Aceptable	Aceptable
Cuadrada	5	20	2	3	Aceptable	Aceptable
Cuadrada	5	40	2	3	Aceptable	Aceptable

Cuadrada	5	60	2	3	Aceptable	Aceptable
Cuadrada	5	80	2	3	Aceptable	Aceptable
Cuadrada	5	100	2	3	Aceptable	Aceptable
Cuadrada	5	20	3	3	Aceptable	Aceptable
Cuadrada	5	40	3	3	Aceptable	Aceptable
Cuadrada	5	60	3	3	Aceptable	Aceptable
Cuadrada	5	80	3	3	Aceptable	Aceptable
Cuadrada	5	100	3	3	Aceptable	Aceptable
Cuadrada	4	20	1	3	Aceptable	Aceptable
Cuadrada	4	40	1	3	Aceptable	Aceptable
Cuadrada	4	60	1	3	Aceptable	Aceptable
Cuadrada	4	80	1	3	Aceptable	Aceptable
Cuadrada	4	100	1	3	Aceptable	Aceptable
Cuadrada	4	20	2	3	Aceptable	Aceptable
Cuadrada	4	40	2	3	Aceptable	Aceptable
Cuadrada	4	60	2	3	Aceptable	Aceptable
Cuadrada	4	80	2	3	Aceptable	Aceptable
Cuadrada	4	100	2	3	Aceptable	Aceptable
Cuadrada	4	20	3	3	Aceptable	Aceptable
Cuadrada	4	40	3	3	Aceptable	Aceptable
Cuadrada	4	60	3	3	Aceptable	Aceptable
Cuadrada	4	80	3	3	Aceptable	Aceptable
Cuadrada	4	100	3	3	Aceptable	Aceptable
Cuadrada	3	20	1	3	Aceptable	Aceptable
Cuadrada	3	40	1	3	Aceptable	Aceptable
Cuadrada	3	60	1	3	Aceptable	Aceptable
Cuadrada	3	80	1	3	Aceptable	Aceptable

Cuadrada	3	100	1	3	Aceptable	Aceptable
Cuadrada	3	20	2	3	Aceptable	Aceptable
Cuadrada	3	40	2	3	Aceptable	Aceptable
Cuadrada	3	60	2	3	Aceptable	Aceptable
Cuadrada	3	80	2	3	Aceptable	Aceptable
Cuadrada	3	100	2	3	Aceptable	Aceptable
Cuadrada	3	20	3	3	Aceptable	Aceptable
Cuadrada	3	40	3	3	Aceptable	Aceptable
Cuadrada	3	60	3	3	Aceptable	Aceptable
Cuadrada	3	80	3	3	Aceptable	Aceptable
Cuadrada	3	100	3	3	Aceptable	Aceptable
Cuadrada	2	20	1	3	Aceptable	Aceptable
Cuadrada	2	40	1	3	Aceptable	Aceptable
Cuadrada	2	60	1	3	Aceptable	Aceptable
Cuadrada	2	80	1	3	Aceptable	Aceptable
Cuadrada	2	100	1	3	Aceptable	Aceptable
Cuadrada	2	20	2	3	Aceptable	Aceptable
Cuadrada	2	40	2	3	Aceptable	Aceptable
Cuadrada	2	60	2	3	Aceptable	Aceptable
Cuadrada	2	80	2	3	Aceptable	Aceptable
Cuadrada	2	100	2	3	Aceptable	Aceptable
Cuadrada	2	20	3	3	Aceptable	Aceptable
Cuadrada	2	40	3	3	Aceptable	Aceptable
Cuadrada	2	60	3	3	Aceptable	Aceptable
Cuadrada	2	80	3	3	Aceptable	Aceptable
Cuadrada	2	100	3	3	Aceptable	Aceptable
Cuadrada	1	20	1	3	Aceptable	Aceptable

Cuadrada	1	40	1	3	Aceptable	Aceptable
Cuadrada	1	60	1	3	Aceptable	Aceptable
Cuadrada	1	80	1	3	Aceptable	Aceptable
Cuadrada	1	100	1	3	Aceptable	Aceptable
Cuadrada	1	20	2	3	Aceptable	Aceptable
Cuadrada	1	40	2	3	Aceptable	Aceptable
Cuadrada	1	60	2	3	Aceptable	Aceptable
Cuadrada	1	80	2	3	Aceptable	Aceptable
Cuadrada	1	100	2	3	Aceptable	Aceptable
Cuadrada	1	20	3	3	Aceptable	Aceptable
Cuadrada	1	40	3	3	Aceptable	Aceptable
Cuadrada	1	60	3	3	Aceptable	Aceptable
Cuadrada	1	80	3	3	Aceptable	Aceptable
Cuadrada	1	100	3	3	Aceptable	Aceptable
No lineal	5	20	1	3	Aceptable	Aceptable
No lineal	5	40	1	3	Aceptable	Aceptable
No lineal	5	60	1	3	Aceptable	Aceptable
No lineal	5	80	1	3	Aceptable	Aceptable
No lineal	5	100	1	3	Aceptable	Aceptable
No lineal	5	20	2	3	Aceptable	Aceptable
No lineal	5	40	2	3	Aceptable	Aceptable
No lineal	5	60	2	3	Aceptable	Aceptable
No lineal	5	80	2	3	Aceptable	Aceptable
No lineal	5	100	2	3	Aceptable	Aceptable
No lineal	5	20	3	3	Aceptable	Aceptable
No lineal	5	40	3	3	Aceptable	Aceptable
No lineal	5	60	3	3	Aceptable	Aceptable

No lineal	5	80	3	3	Aceptable	Aceptable
No lineal	5	100	3	3	Aceptable	Aceptable
No lineal	4	20	1	3	Aceptable	Aceptable
No lineal	4	40	1	3	Aceptable	Aceptable
No lineal	4	60	1	3	Aceptable	Aceptable
No lineal	4	80	1	3	Aceptable	Aceptable
No lineal	4	100	1	3	Aceptable	Aceptable
No lineal	4	20	2	3	Aceptable	Aceptable
No lineal	4	40	2	3	Aceptable	Aceptable
No lineal	4	60	2	3	Aceptable	Aceptable
No lineal	4	80	2	3	Aceptable	Aceptable
No lineal	4	100	2	3	Aceptable	Aceptable
No lineal	4	20	3	3	Aceptable	Aceptable
No lineal	4	40	3	3	Aceptable	Aceptable
No lineal	4	60	3	3	Aceptable	Aceptable
No lineal	4	80	3	3	Aceptable	Aceptable
No lineal	4	100	3	3	Aceptable	Aceptable
No lineal	3	20	1	3	Aceptable	Aceptable
No lineal	3	40	1	3	Aceptable	Aceptable
No lineal	3	60	1	3	Aceptable	Aceptable
No lineal	3	80	1	3	Aceptable	Aceptable
No lineal	3	100	1	3	Aceptable	Aceptable
No lineal	3	20	2	3	Aceptable	Aceptable
No lineal	3	40	2	3	Aceptable	Aceptable
No lineal	3	60	2	3	Aceptable	Aceptable
No lineal	3	80	2	3	Aceptable	Aceptable
No lineal	3	100	2	3	Aceptable	Aceptable

No lineal	3	20	3	3	Aceptable	Aceptable
No lineal	3	40	3	3	Aceptable	Aceptable
No lineal	3	60	3	3	Aceptable	Aceptable
No lineal	3	80	3	3	Aceptable	Aceptable
No lineal	3	100	3	3	Aceptable	Aceptable
No lineal	2	20	1	3	Aceptable	Aceptable
No lineal	2	40	1	3	Aceptable	Aceptable
No lineal	2	60	1	3	Aceptable	Aceptable
No lineal	2	80	1	3	Aceptable	Aceptable
No lineal	2	100	1	3	Aceptable	Aceptable
No lineal	2	20	2	3	Aceptable	Aceptable
No lineal	2	40	2	3	Aceptable	Aceptable
No lineal	2	60	2	3	Aceptable	Aceptable
No lineal	2	80	2	3	Aceptable	Aceptable
No lineal	2	100	2	3	Aceptable	Aceptable
No lineal	2	20	3	3	Aceptable	Aceptable
No lineal	2	40	3	3	Aceptable	Aceptable
No lineal	2	60	3	3	Aceptable	Aceptable
No lineal	2	80	3	3	Aceptable	Aceptable
No lineal	2	100	3	3	Aceptable	Aceptable
No lineal	1	20	1	3	Aceptable	Aceptable
No lineal	1	40	1	3	Aceptable	Aceptable
No lineal	1	60	1	3	Aceptable	Aceptable
No lineal	1	80	1	3	Aceptable	Aceptable
No lineal	1	100	1	3	Aceptable	Aceptable
No lineal	1	20	2	3	Aceptable	Aceptable
No lineal	1	40	2	3	Aceptable	Aceptable

No lineal	1	60	2	3	Aceptable	Aceptable
No lineal	1	80	2	3	Aceptable	Aceptable
No lineal	1	100	2	3	Aceptable	Aceptable
No lineal	1	20	3	3	Aceptable	Aceptable
No lineal	1	40	3	3	Aceptable	Aceptable
No lineal	1	60	3	3	Aceptable	Aceptable
No lineal	1	80	3	3	Aceptable	Aceptable
No lineal	1	100	3	3	Aceptable	Aceptable

11.5 Pasos para ensamblaje

Tabla 11.3 Instrucciones generales de ensamblaje. Fuente: Propia

Base de accesorio	
1.	Tome la base de la carcasa y colóquela sobre una superficie plana, lisa y estable.
2.	Coloque el inserto sobre el agujero de soporte correspondiente, utilizando un cautín comience a calentar el inserto, ejerciendo presión moderada hasta que el agujero del soporte comience a ceder y el inserto se pueda fijar internamente.
3.	Repita el paso 2 para todos los insertos.
4.	Coloque las placas de circuitos impresos como se visualiza en el plano de ensamble y ajústelas con los tornillos al inserto previamente colocado en el agujero del soporte.
Cuerpo del accesorio	
5.	De adentro hacia afuera coloque cada Led en un agujero de la cara frontal ejerciendo presión moderada para la inserción.
6.	De afuera hacia adentro y teniendo precaución con los cables de cada pin, coloque los conectores PL-Series correspondientes en cada agujero y ajústelos internamente con la tuerca, asegúrese que queden fijados.

7.	Cada par cables de pines de los conectores PL-Series ajústelos a un puerto de los conectores AMP CT hembras.
8.	Conecte cada Led a un conector AMP CT Hembra.
9.	Coloque la base con los circuitos impresos en el cuerpo del accesorio.
10.	Conecte cada par de señales a los canales de los circuitos impresos, manteniendo el cuidado de asociar cada señal de entrada con la señal de salida correspondiente.
11.	Selle el accesorio colocando la tapa superior.

12 Anexos

12.1 Información del inserto

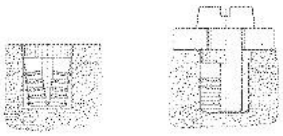
Insertos roscados **SPREDSERT® 1** *por expansión*

Ventajas

- Para termoplásticos
- El moleteado y anillos de anclaje aportan un alto grado de seguridad en la resistencia al giro y al arrancamiento
- Expansión por tornillo

Material: Cu Zn 38 Pb 2 (de acuerdo con la norma EU 2000/53)

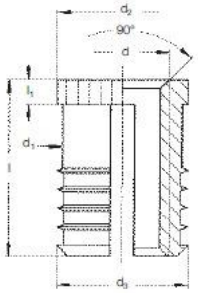
Principio



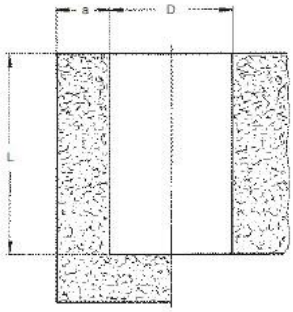
El inserto **SPREDSERT® 1** se introduce en el agujero hasta que el moleteado de la cabeza está totalmente anclado en el plástico. Al mismo tiempo la parte con anillos se cierra. Al introducir el tornillo se produce una expansión radial al **SPREDSERT® 1** causando el anclaje de los anillos en el plástico.
Se produce también en este proceso el bloqueo del tornillo. El par de apriete se debe incrementar en un 10 %.

Información técnica

Tipo 0831 – 0833

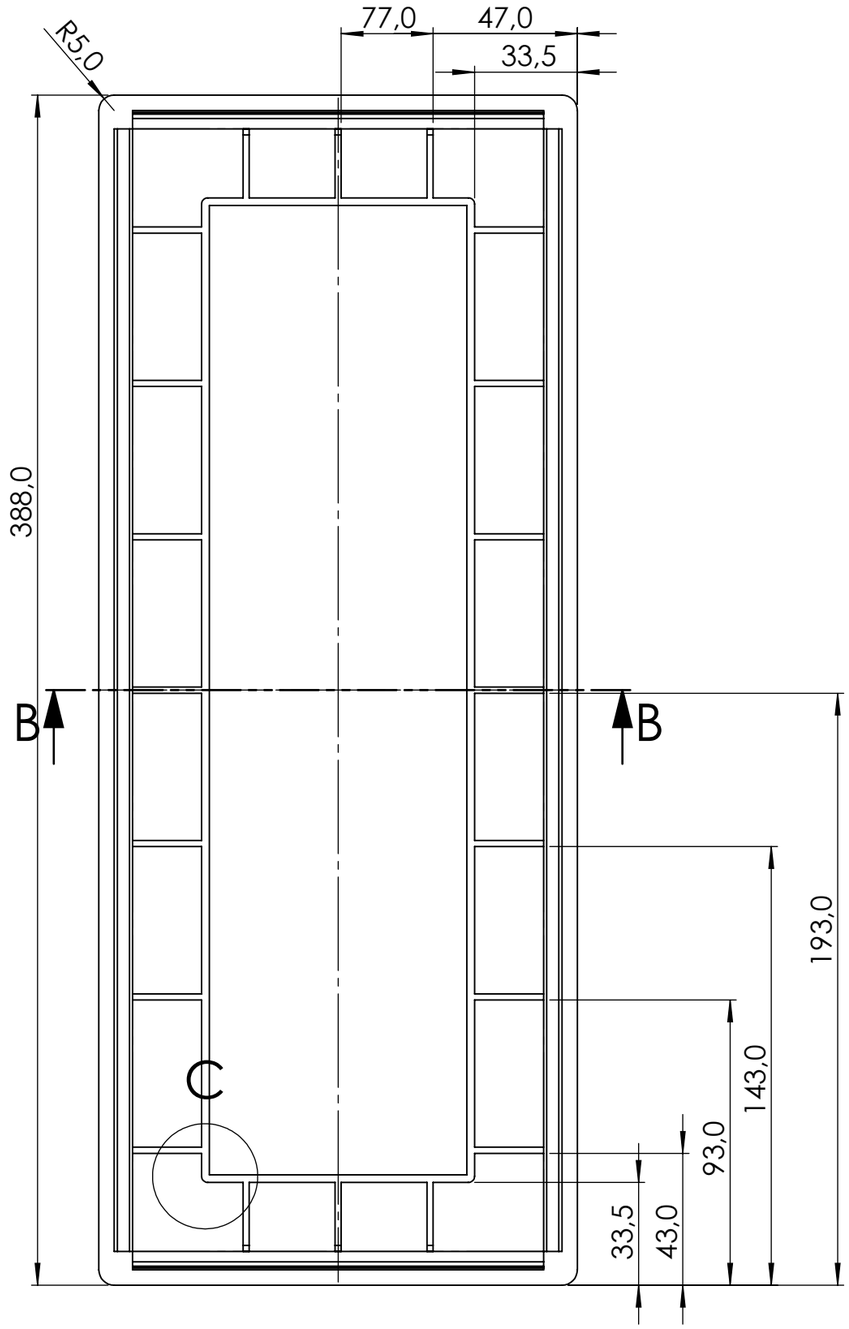


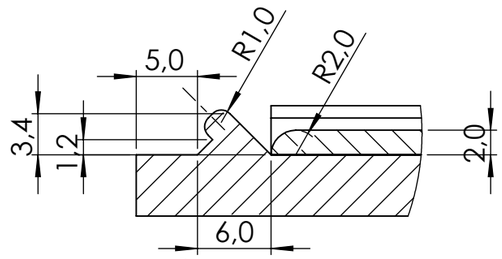
Dimensiones del alojamiento①



Para útiles y máquinas de colocación, ver páginas 39 – 40

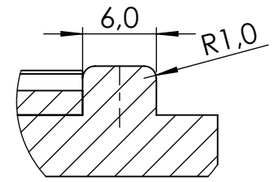
12.2 Planos





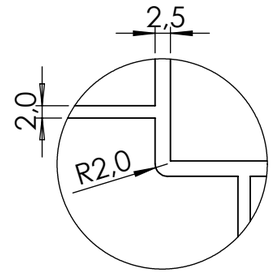
DETALLE A

ESCALA 2 : 1



DETALLE B

ESCALA 2 : 1



DETALLE C

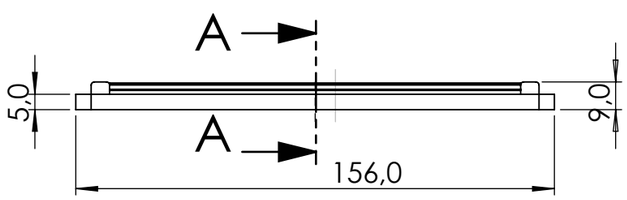
ESCALA 1 : 1

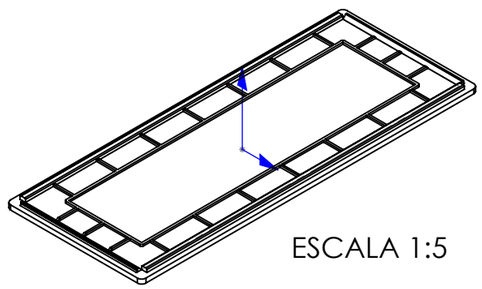
SECCIÓN B-B

ESCALA 1 : 1

SECCIÓN A-A

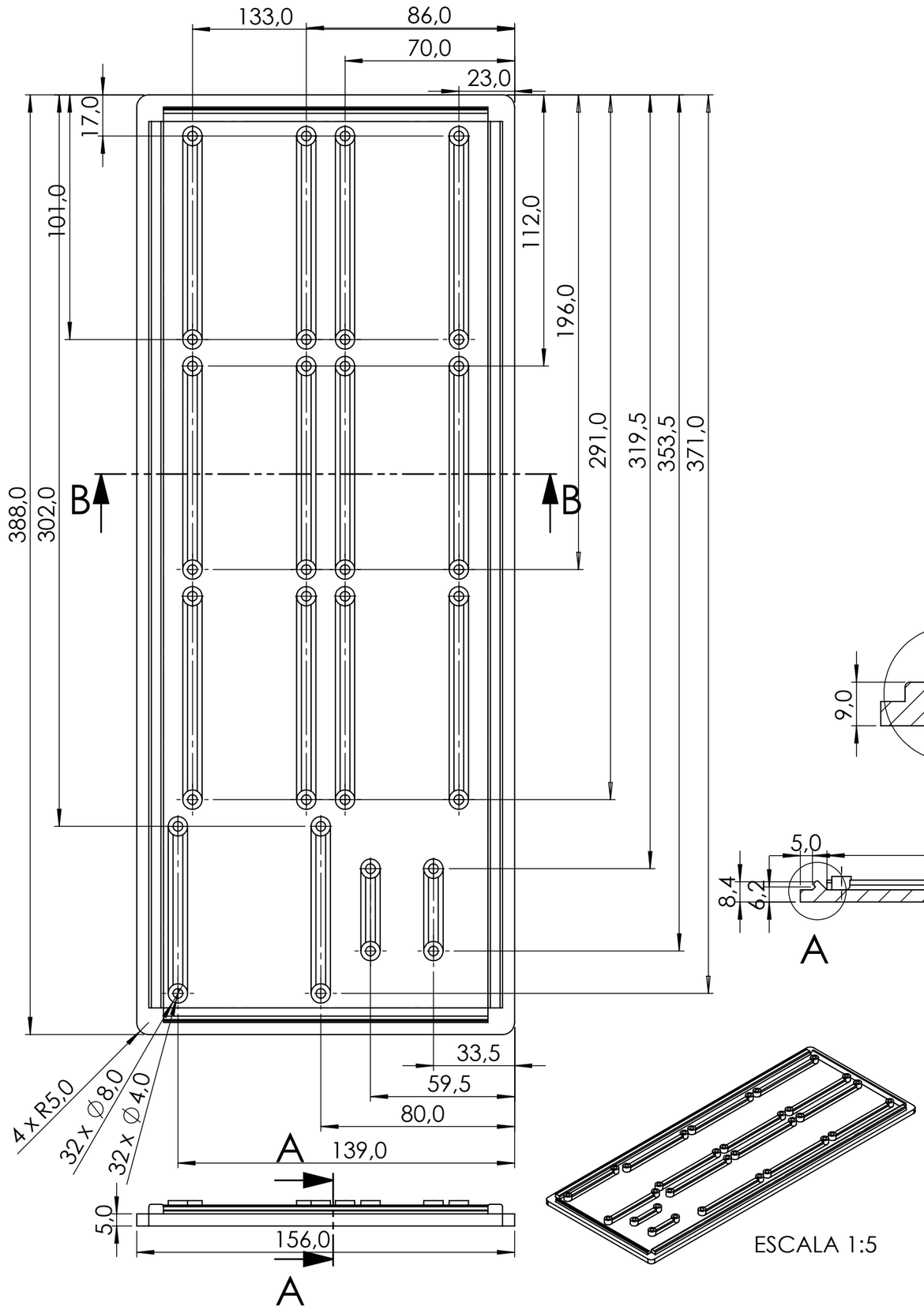
ESCALA 1 : 2





ESCALA 1:5

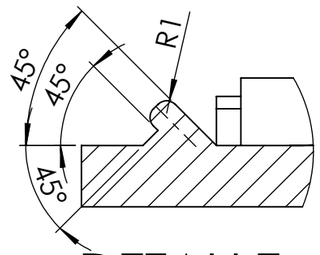
TEC | Tecnológico de Costa Rica				ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA		REVISIÓN:
				CURSO: Proyecto Final de Gradiación		A
	NOMBRE	FIRMA	FECHA	ACABADO GENERAL:	PARTE:	
DIBUJ.	D. Aguirre		25/03/21	N11	Tapa	
				MATERIAL: Nylon 12		
SISTEMA: FIRST ANGLE PROJECTION		TOLERANCIAS: LINEAL: X ± 1, .X ± 0.5, .XX ± 0.05, .XXX± 0.001 ANGULAR: ± °0.5		TOLERANCIA EN: mm	PROYECTO: Accesorio de resguardo	ESCALA: 1:2 / FORMATO A3
				PESO: -- gramos	UNIDADES: mm	HOJA 1 DE 4



ESCALA 1:5

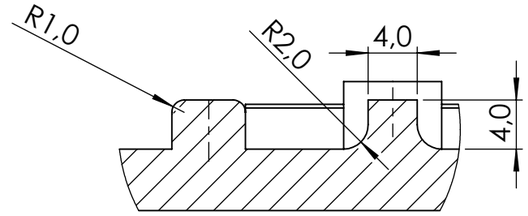
Notas:

- Las medidas del nervio presentadas en el detalle B, son iguales para todos los nervios.
- Los sujetadores cilindricos se encuentran alineados entre ellos



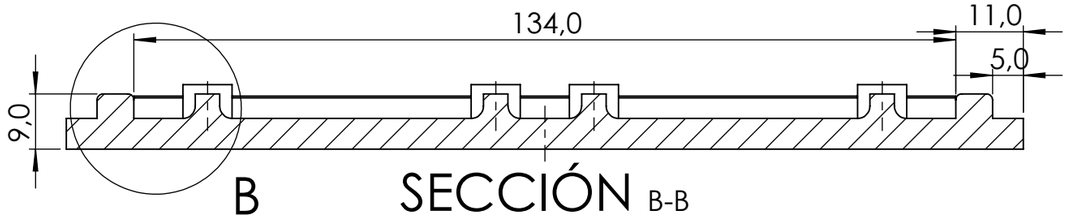
DETALLE A

ESCALA 2 : 1



DETALLE B

ESCALA 2 : 1



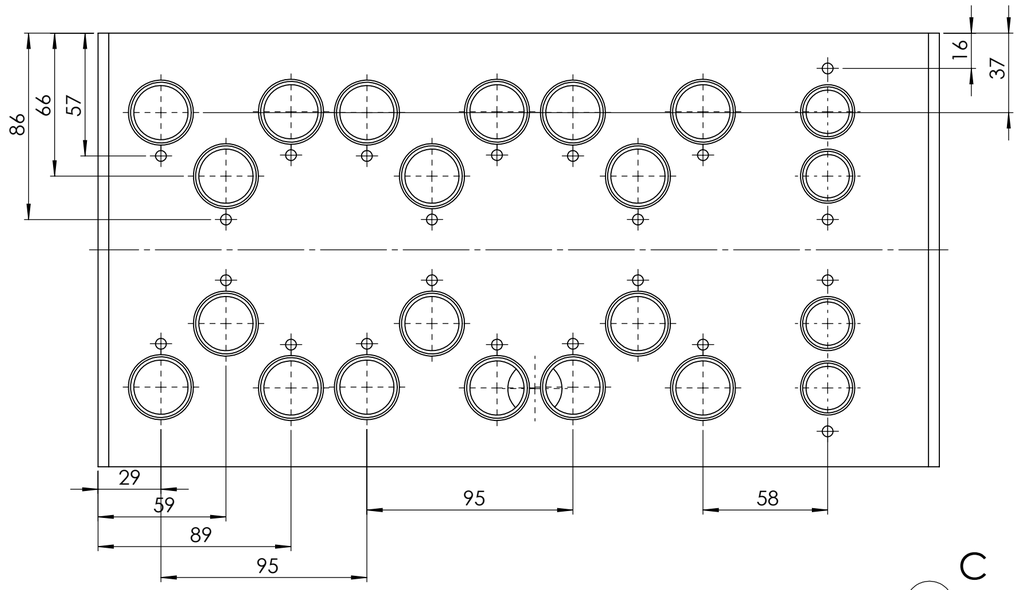
SECCIÓN B-B

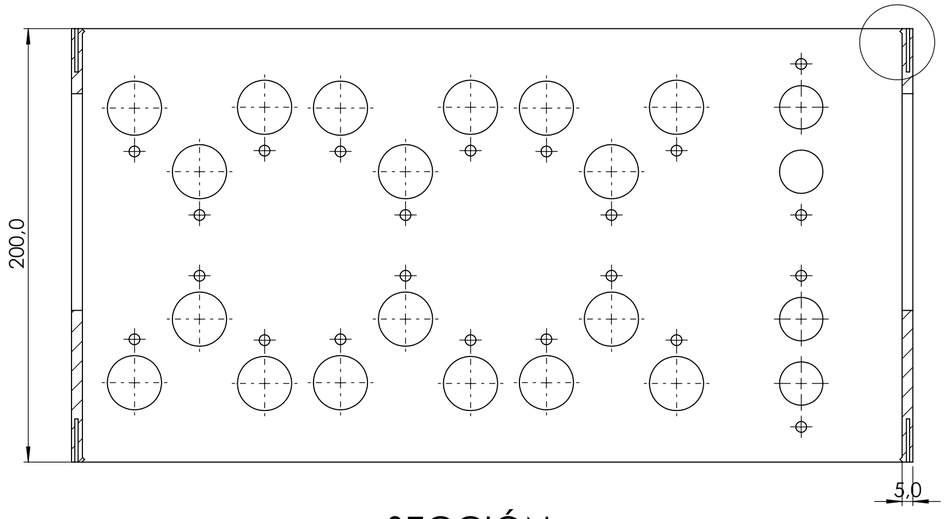
ESCALA 1 : 1

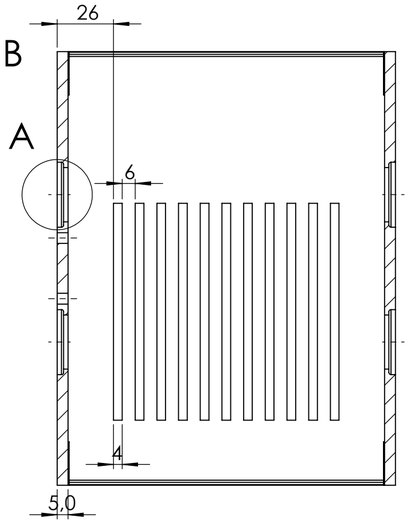
SECCIÓN A-A

ESCALA 1 : 2

TEC | Tecnológico de Costa Rica				ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA		REVISIÓN:
				CURSO: PROYECTO FINAL DE GRADIACIÓN		A
	NOMBRE	FIRMA	FECHA	ACABADO GENERAL:	PARTE:	
DIBUJ.	D. AGUIRRE		25/03/21	N11	BASE	
				MATERIAL: NYLON 12		
SISTEMA: FIRST ANGLE PROJECTION		TOLERANCIAS: LINEAL: X ± 1, . X ± 0.5, . XX ± 0.05, . XXX± 0.001, ANGULAR: ± º0.5		TOLERANCIA EN: mm	PROYECTO: ACCESORIO DE RESGUARDO	ESCALA: 1:2 / FORMATO A3
				PESO: -- GRAMOS	UNIDADES: mm	HOJA 2 DE 4





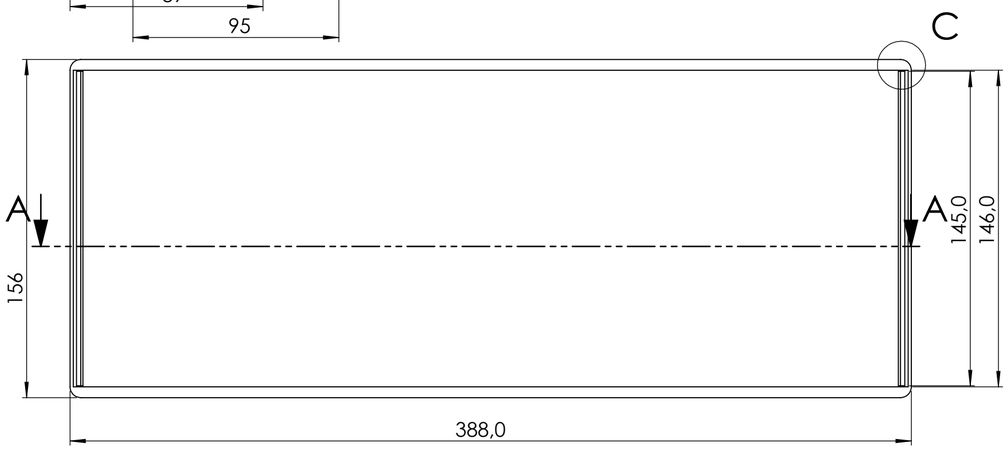


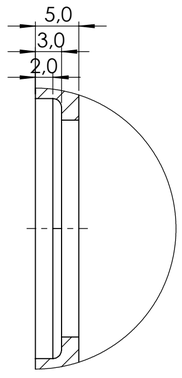
SECCIÓN A-A

ESCALA 1 : 2

SECCIÓN B-B

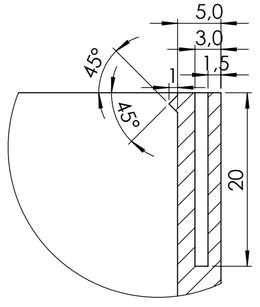
ESCALA 1 : 2





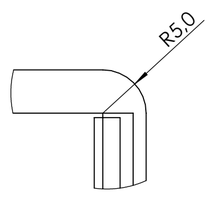
DETALLE A

ESCALA 2 : 1



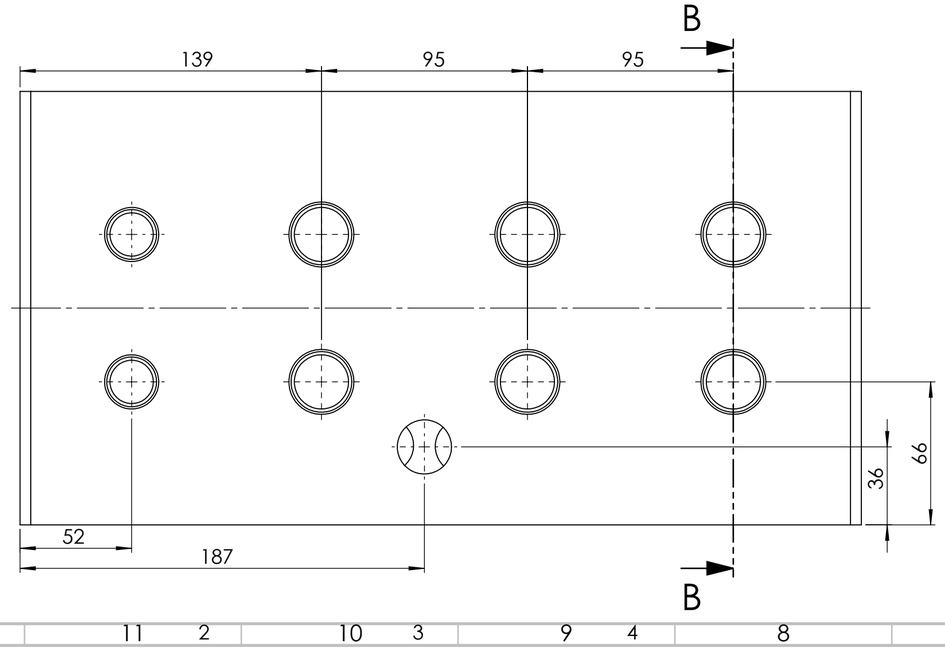
DETALLE B

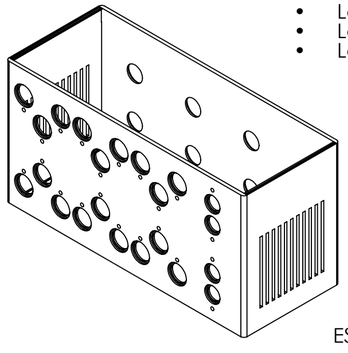
ESCALA 2 : 1



DETALLE C

ESCALA 2 : 1





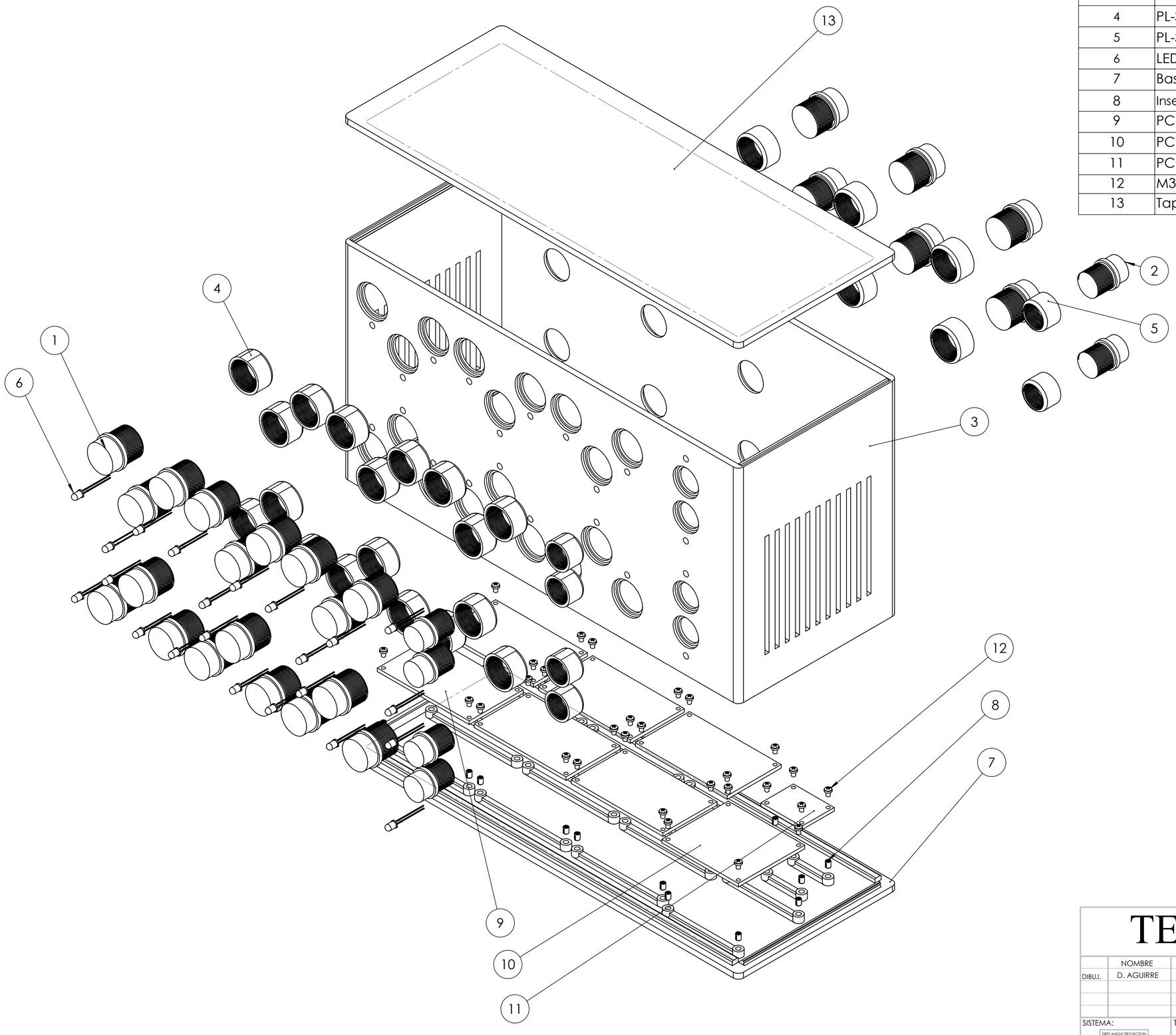
ESCALA 1:5

NOTAS:

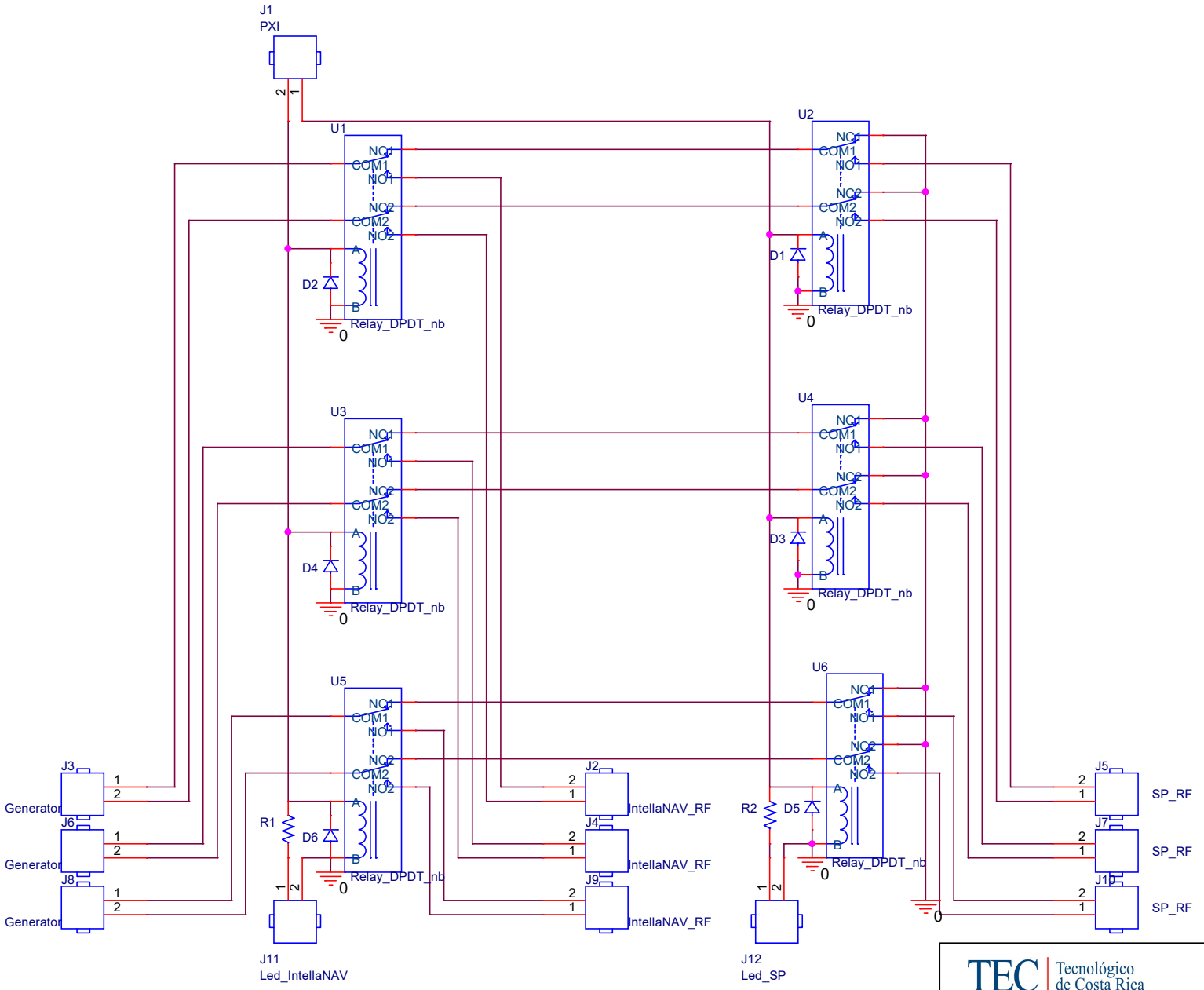
- Las medidas del detalle A se repiten para los agujeros grandes.
- Las medidas del detalle B se repiten para todos las 4 cejillas.
- Los agujeros pequeños se encuentran alineados verticalmente con los grandes.
- Las ranuras laterales observadas en la seccion B-B se encuentras espaciadas con la misma medida.
- Los agujeros se encuentran alineados horizontalmente entre ellos.

TEC | Tecnológico de Costa Rica				ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA	REVISIÓN: A
				CURSO: Proyecto Final de Gradiación	
	NOMBRE	FIRMA	FECHA	ACABADO GENERAL: N11	PARTE: Cuerpo
DIBUJ.	D. Aguirre		25/03/21		
				MATERIAL: Nylon 12	
SISTEMA: FIRST ANGLE PROJECTION	TOLERANCIAS: LINEAL: X ± 1, .X ± 0.5, .XX ± 0.05, .XXX± 0.001 ANGULAR: ± 0.5			TOLERANCIA EN: mm	PROYECTO: Accesorio de resguardo / ESCALA: 1:2 / FORMATO A2
				PESO: -- gramos	UNIDADES: mm / HOJA 3 DE 4

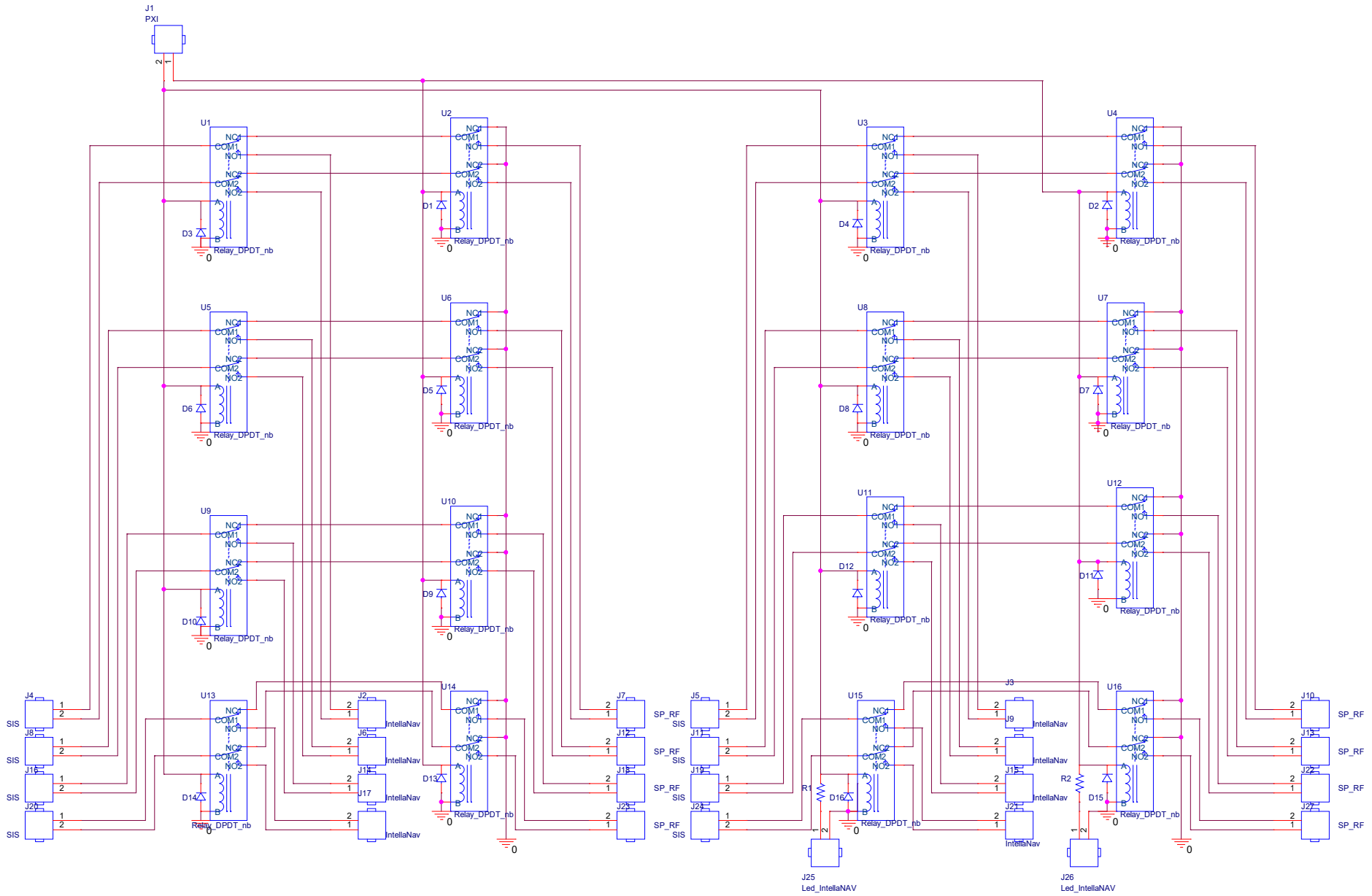
N.° DE ELEMENTO	N.° DE PIEZA	MATERIAL	CANTIDAD
1	PL-Serie 800	Plastico	24
2	PL-Serie 900	Plastico	6
3	Cuerpo	Nylon 12	1
4	PL-Serie 800 Rosca	Plastico	24
5	PL-Serie 900 Rosca	Plastico	6
6	LED	N/A	22
7	Base	Nylon 12	1
8	Inserto	Cu Zn 38 Pb 2	32
9	PCB Tipo A	N/A	6
10	PCB Tipo B	N/A	1
11	PCB Tipo C	N/A	1
12	M3 x 0.5 x 4 Type I	Acero	32
13	Tapa	Nylon 12	1



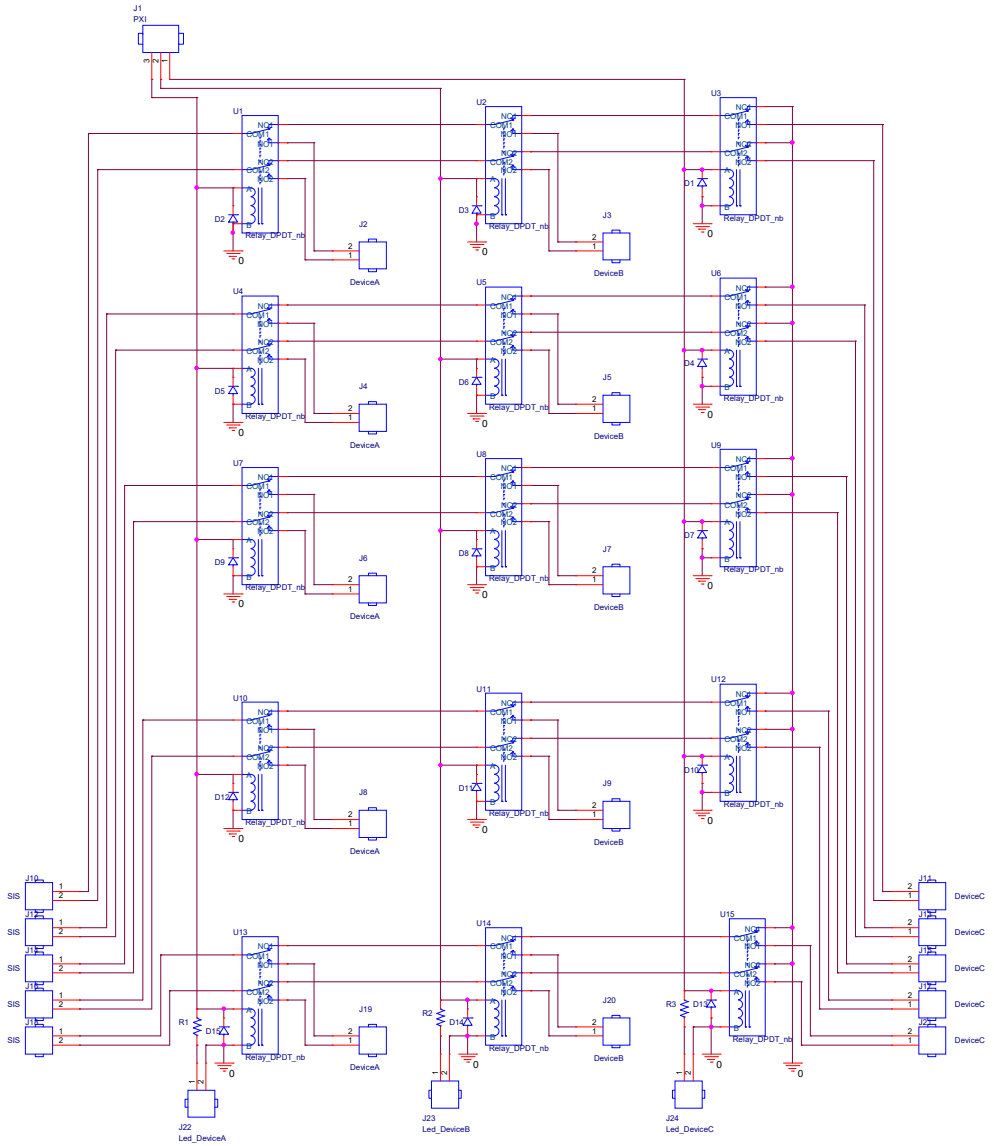
TEC | Tecnológico de Costa Rica				ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA	REVISIÓN: A
				CURSO: PROYECTO FINAL DE GRADIACIÓN	
	NOMBRE	FIRMA	FECHA	ACABADO GENERAL: N/A	PARTE: VISTA DE EXPLOSIÓN
DIBUJ.	D. AGUIRRE		25/03/21		
				MATERIAL: NYLON 12	
SISTEMA: FIRST ANGLE PROJECTION		TOLERANCIAS: LINEAL: X ± 1, .X ± 0.5, .XX ± 0.05, .XXX ± 0.001, ANGULAR: ± °0.5		TOLERANCIA EN: mm	PROYECTO: ACCESORIO DE RESGUARDO / ESCALA: 1:2 / FORMATO A2
				PESO: -- GRAMOS	UNIDADES: mm / HOJA 4 DE 4



TEC Tecnológico de Costa Rica				Ingenieria Mecatronica	Revision: A
				Proyecto Final de Graduacion	
				PROYECTO: Circuito impreso para pruebas automatizadas	
	Nombre	Firma	Fecha	ACABADO GENERAL: N/A	
DIBUJ.	D. Aguirre				
VERIF	N/A		N/A		
APROF	N/A		N/A	MATERIAL: N/A	
FABR	N/A		N/A		
SISTEMA	TOLERANCIA GENERAL			DESIGNACION: N/A	PARTE: Conmutador de señales del generador de ablacion; ESCALA: N/A; FORMATO: A4
				CANTIDAD: N/A	UNIDADES: mm; HOJA: 1 de 3



TEC Tecnológico de Costa Rica		Ingeniería Mecatrónica	Revision: A
		Proyecto Final de Graduacion	
		PROYECTO: Circuito impreso para pruebas automatizadas	
DIBUJ. D. Aguirre		ACABADO GENERAL: N/A	
VERIF N/A		MATERIAL: N/A	
APROF N/A			
FABR N/A			
SISTEMA	TOLERANCIA GENERAL	DESIGNACION: N/A	PARTE: Conmutador de señales del maestro de ablacion
		CANTIDAD: N/A	ESCALA: N/A
		UNIDADES: mm	FORMATO: A3
			HOJA: 2 de 3



TEC Tecnológico de Costa Rica		Ingeniería Mecatrónica	Revisión: A
		Proyecto Final de Graduacion	
		PROYECTO: Circuito impreso para pruebas automatizadas	
DIBUJ. D. Aguirre		ACABADO GENERAL: N/A	
VERIF N/A		N/A	
APROF N/A		MATERIAL: N/A	
FABR N/A		N/A	
SISTEMA:	TOLERANCIA GENERAL:	DESIGNACION: N/A	PARTE: Conmutador de señales de los puertos de entrada y salida
		CANTIDAD: N/A	ESCALA: N/A
		UNIDADES: mm	FORMATO: A2
			HOJA: 3 de 3