E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Desarrollo de un equipo para verificar funcionalmente tarjetas electrónicas

Grado en Ingeniería Eléctrica y Electrónica

Trabajo Fin de Grado

Wilmer De Avila Gutierrez
Ignacio Del Villar Fernández
Pamplona, de junio de 2017

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Dedicatoria

Dedicado a mi madre, la razón por la que he conseguido llegar donde estoy.

Especiales agradecimientos a:

Mi tutor del proyecto, *Ignacio Del Villar Fernández*, por su apoyo y guía durante todo el desarrollo.

A la empresa *Electrónica Falcón* y su plantilla, en especial a los departamentos de ingeniería de test e I+D.

A mis padres por el gran esfuerzo que supuso mantenerme estudiando.

A los compañeros y amigos de la universidad con quienes compartí todos estos años.

Las comunidades de Processing y Arduino, por la interminable cantidad de información que facilitan en la web.

Resumen

La fabricación de una tarjeta electrónica conlleva varios procesos; diseño del esquema eléctrico, diseño de la placa de circuito impreso y montaje de componentes. Una vez fabricada, se debe verificar el funcionamiento de la misma. El presente proyecto consiste en el desarrollo de un equipo para llevar a cabo esa tarea, utilizando para ello una tarjeta Arduino.

La tarjeta a verificar se fabrica y se monta en la empresa *Electrónica Falcón S.A*, para uno de sus clientes. Basado en las especificaciones del cliente, se diseña el software, hardware y la mecánica del verificador, todo ello teniendo en cuenta los procedimientos y normativas establecidas por la empresa.
El útil se conecta a un ordenador vía puerto serie con el fin de presentar una interfaz gráfica al operario encargado de su manejo.

La primera orden de fabricación de parte del cliente consta de diez mil unidades, por lo que, el grado de automatización y la velocidad de la verificación han sido factores tenidos muy en cuenta en el diseño.

Abstract

The manufacture of an electronic card involves several processes; electrical design, printed circuit board design and mounting components. Once manufactured, it must be functionally verified. This Project is about the developing of a system for that purpose, using an Arduino board.

The board that will be verified is fabricated and mounted in the company *Electrónica Falcón S.A*, for one of its clients. The software, hardware and mechanic, are designed based on the client specifications and the company regulations.
The tool is connected to the computer via serial bus, in order to show a graphic interface to the operator.

The first client's order consists of ten thousand units. Because of this, the automation degree and the speed of the verification had been highly important to the design.

Key word list

- Verification sequence.
- Device Under Test (DUT).
- Verifier software.
- Development environment.
- Client specifications.
- Automation degree.
- Printed circuit board (PCB).
- Test probes bed.

Lista de palabras clave

- Secuencia de verificación.
- Tarjeta a verificar (DUT).
- Software del verificador.
- Entorno de desarrollo.
- Especificaciones del cliente.
- Grado de automatización.
- Placa de circuito impreso (PCB).
- Cama de pinchos.

TABLA DE CONTENIDO

ANEXOS

Índice De Figuras

Índice DeTablas

1 Introducción

Después de la fabricación de una PCB y el montaje de sus componentes, se debe comprobar el correcto funcionamiento de la misma. Por ejemplo, es impensable que una empresa que se dedique a la automoción instale la electrónica del coche sin tener la total seguridad que esas tarjetas han sido testeadas y comprobadas con anterioridad.
Hay componentes que se pueden ver que están defectuosos con una inspección visual, pero no es posible asegurar que no fallos internos. Para asegurar que el 100% de las tarjetas fabricadas funcionen como deben, se diseñan equipos de verificación para cada tipo de tarjeta, en los cuales, con el uso de electrónica auxiliar se testea punto por punto el comportamiento del dispositivo en cuestión.

En Navarra, la empresa Electrónica Falcón (En adelante *E. Falcón*) fue la primera en realizar este tipo de trabajo, y actualmente es líder del sector en la provincia y una de las primeras en España. Es en esta empresa donde se realiza el montaje y desarrollo del equipo para la verificación de la tarjeta denominada TCTRL para uno de sus clientes. (Debido a la confidencialidad, no se puede nombrar al cliente en cuestión).

El funcionamiento de un verificador es relativamente sencillo; se diseña un soporte físico o caja donde se posiciona la tarjeta a testear (DUT, Device Under Test), luego, mediante accionamiento manual, se conecta la tarjeta a la electrónica auxiliar, la cual se encarga, mediante el software diseñado, de realizar el test y mostrar los pasos y resultados en la pantalla del ordenador.

Hay que tener en cuenta que el equipo se diseña para ser utilizado en el área de producción de la empresa por distintos operarios en función del día y la disponibilidad, por lo que, el verificador debe tener: una mecánica fácil de usar, un hardware robusto, un software rápido y con una interfaz gráfica simple y fácil de seguir.

2 Justificación y objetivos

En el mundo de la electrónica tiende cada vez más a la reducción del tamaño de los dispositivos, se busca introducir más componentes en menos espacio. Esto dificulta su manipulación y aumenta la fragilidad y complejidad del diseño.

Actualmente existen máquinas capaces de seleccionar los componentes de un lugar determinado y posicionarlos de manera automática en la PCB. Además, también pueden verificar el correcto posicionamiento mediante el uso cámaras y sensores. Sin embargo, estas máquinas pueden fallar, ya sean fallos mecánicos, fallos de programación o de alimentación de componentes. Por ejemplo, una tarjeta con resistencias de valor incorrecto; ópticamente, para la máquina es un montaje correcto, pero un error así podría llegar a dañar las tarjetas fabricadas.

Por otro lado, está el montaje por introducción o agujero pasante (THT Through-Hole Technology), el cual se realiza de manera manual por un operario. Puede haber fallos por mala soldadura, por mal posicionamiento (colocar componentes del revés) o por manipulación incorrecta del componente.

Entregar tarjetas defectuosas al cliente supondría pérdidas importantes no solo económicas sino también de la imagen corporativa de la empresa. Por este motivo, es necesario comprobar el correcto funcionamiento de las placas producidas mediante un equipo de verificación o verificador.

El objetivo principal de este proyecto es el de desarrollar un equipo que verifique el funcionamiento de una tarjeta dada por el cliente (TCTRL) y dejarlo instalado y funcionando en el área de verificación (producción) de la empresa. Para conseguir este objetivo se van desarrollando otras metas intermedias:

- Coordinar el equipo de trabajo, asignar y validar tareas.
- Realizar un sistema mecánico cómodo y fácil de usar.
- Diseñar una interfaz gráfica clara y sencilla.
- Conseguir un elevado grado de automatización.
- No sobrepasar un tiempo de verificación de un minuto por tarjeta.
- Documentar según las normas de la empresa; pruebas, tutoriales, informes, información del cliente, planos, y cualquier otro tipo de archivo.
- Dar soporte al responsable de verificación sobre el funcionamiento de la aplicación antes y durante el inicio de la verificación.

3 Estado del arte

Se realiza una descripción general de la empresa E. Falcón; historia, servicios, procedimientos y situación actual en el mercado. Acto seguido, se realiza un estudio sobre los distintos tipos de verificadores que se fabrican en la empresa y sus métodos de verificación.

3.1 Datos de la empresa

3.1.1 Historia de la empresa

[1], [2] E. Falcón es una empresa fundada 1976 como una de las primeras empresas españolas dedicadas a la subcontratación del montaje de circuitos impresos. En ese año, Jesús Falcón y su mujer, como autónomos, crearon la empresa en los bajos de su vivienda en Peralta, Navarra. En aquel momento no había ninguna empresa con experiencia en el montaje de circuitos impresos. Cuando la empresa Azkoyen, dedicada a la fabricación de máquinas de vending, pasa de la mecánica a la electrónica, E. Falcón se convierte en la encargada de suministrar las tarjetas electrónicas necesarias.

Hasta entonces las multinacionales españolas que necesitaban circuitos eléctricos, los integraban en su producción. Al aparecer Falcón, estas empresas empezaron a subcontratar esos montajes, lo que les permitía no solo liberar una parte del trabajo (a lo que no se dedica la empresa), sino también confiarlo a una empresa especializada. Así, una empresa que se dedica a realizar coches, pasó de tener una línea para crear las tarjetas a subcontratar esa parte, y dedicarse enteramente al montaje de coches, lo que les daba mayor rentabilidad.

En el 2002 la empresa realizo su primera exportación, precisamente a Francia, generando un cambio de mentalidad que a fin de cuentas desembocaría en lo que es la empresa ahora, con exportaciones a todo el mundo y a grandes empresas como General Electrics o Gamesa.

En la actualidad, la empresa cuenta con 150 empleados distribuidos en una planta de 8500m^2, donde se ubican todas las instalaciones necesarias para atender las diversas posibilidades de servicios que ofrece al cliente [2].

Figura 1 – Nave industrial Electrónica Falcón.
www.electronicafalcon.com

3.1.2 Servicios

Desde su creación, Electrónica Falcón ha permanecido en una continua evolución hasta llegar a ser la empresa que es en la actualidad. Esta transformación se ha llevado a cabo mediante la ampliación de sus instalaciones, la adquisición de nuevas tecnologías, la optimización de procesos y métodos de trabajo, y el mantenimiento de una plantilla cualificada la cual participa activamente en la consecución de una mejora constante asumiendo la calidad en su trabajo. En base a esto, la empresa ofrece los siguientes servicios.

- Diseño hardware y software de equipos electrónicos
- Fabricación de prototipos y preseries
- Desarrollo de equipos de test
- Montaje de tarjetas electrónicas
- Montaje de equipos electromecánicos

3.1.2.1 Investigación y desarrollo (I+D)

Formada por personal técnico altamente cualificado y con elevada experiencia en diversos sectores tecnológicos, la división de investigación y desarrollo de la empresa tiene como objetivo aportar al cliente la mejor solución posible en la ingeniería de proyectos de sistemas electrónicos y electromecánicos

El cliente puede seleccionar la amplitud que requiere para cada proyecto, puesto que la empresa ofrece la cobertura completa y continua de la vida de un producto; desde su nacimiento a partir de un concepto, pasando por fases de prototipos y certificaciones hasta su perfeccionamiento tras la fabricación de series.

Figura 2 - Fases de proyecto.
www.electronicafalcon.com

La gran diversidad de proyectos realizados hace que se tenga un alto conocimiento en muchas de las tecnologías más punteras tanto en hardware como software, pudiendo destacar entre todas:

- Programación de cualquier tipo de microcontrolador.
- Aplicaciones bajo Windows y Linux embebido.
- Desarrollo de drivers específicos.
- Comunicaciones inalámbricas (RF, Zigbee, bluetooth).
- Posicionamiento por GPS.
- Reproducción MP3.
- Sistemas de iluminación.
- Interfaces de usuarios (Pantallas táctiles, LCD).

3.1.2.2 Logística

La empresa, salvo excepciones solicitadas por el cliente, gestiona la totalidad de la materia prima tanto electrónica como mecánica que se precisa para la fabricación de las necesidades del cliente.

A día de hoy los departamentos de compras y logística a través de su propia herramienta de comercio electrónico manejan más de 15000 referencias, las cuales quedan ubicadas en los distintos almacenes distribuidos en la empresa.

Además, la empresa cuenta con almacenes de material electrónico con control de humedad según la norma STD-033 (normativa sobre el manejo, embalaje, control de humedad y temperatura de componentes de montaje superficial o SMD [4]).

3.1.2.3 Fabricación

La fabricación de tarjetas electrónicas en la empresa se lleva a cabo en tres secciones:

- SMT (Surface Mounting Technology, tecnología de montaje superficial): cuenta con una capacidad productiva de 210000 componentes a la hora en tres líneas de fabricación con las últimas tecnologías; maquinas pick&place (posicionamiento automático de componentes), hornos de refusión (soldadura), máquinas de serigrafía, máquinas de test óptico (verificación óptica), equipos de aguja móvil (testeo sin utillajes) y equipos de rayos X (inspección de soldaduras no visibles).

- Montaje convencional: Inserción de componentes (through hole) y soldadura mediante máquinas de ola. También consta de un área de acabados manuales para aquellas piezas que por su complejidad o volumen no pueden ser ensamblados en las líneas de montaje.

- Tropicalizado: Destinado a realizar tratamientos antihumedad a las tarjetas por revestimiento con productos químicos como barnices, siliconas o ceras. Consta e cabinas de luz ultravioleta, hornos de curado, tropicalizado por inmersión y una máquina automática de aplicación selectiva.

Todos los departamentos de la empresa trabajan con productos con normativas Rohs y Reach como con estaño-plomo (normativas para la eliminación de sustancias tóxicas como el plomo, mercurio o el cadmio [5]). Además, se instalan y se dispone de equipos para el control de la electricidad estática, esencial para el manejo de componentes electrónicos.

3.1.2.4 Ensamblaje

Con el fin de aportar un mayor servicio y simplificar el trabajo a sus clientes, E. Falcón creó la sección de montaje electromecánico, en la cual las tarjetas electrónicas pasan a ser materia prima junto a piezas metálicas, inyectados de plástico, mazos, etc. y son ensamblados formando un producto final mucho más completo para el cliente.

3.1.2.5 Calidad

La empresa, mediante la sección de verificación asegura el correcto funcionamiento del 100% de las tarjetas con los siguientes sistemas de test.

- Verificación funcional por cama de pinchos: tres máquinas que pueden realizar la verificación de la tarjeta aportando todo tipo de señales analógicas y digitales y realizando las consiguientes medidas que aseguren su correcto funcionamiento.

- Verificadores manuales funcionales: Diseñados y fabricados en Electrónica Falcón, a medida para cada circuito impreso.

- Cámara climática: permite la realización de pruebas en condiciones extremas de temperatura y humedad comprobando así su correcto funcionamiento. Utilizada tanto para comprobación de prototipos como para series.

- Test de aislamiento: como complemento a comprobación funcional de equipos mecánicos, se dispone de un módulo para test de rigidez dieléctrica.

Cabe destacar que la empresa se encuentra certificada desde hace más de 10 años conforme la norma UNE-ENISO9001:2008 y UNE-ENISO9001:2004.

Actualmente, Electrónica Falcón ocupa el tercer lugar en el ranking del sector "Fabricación de circuitos impresos y ensamblados", y la posición 214 en el ranking de empresas de Navarra, con una facturación de 16 millones de euros [3].

A continuación, se adjunta un enlace a un artículo del día 26 de noviembre de 2016, sobre la celebración de los 40 años de la empresa, realizado por el diario Noticias de Navarra.
http://www.noticiasdenavarra.com/2016/11/26/economia/electronica-falcon-en-peralta-factura-al-ano-16-millones-de-euros

3.2 Estructura general de un verificador

Como se explicó en el punto anterior, la empresa asegura el funcionamiento del 100% de las tarjetas desarrolladas. Para esto, utiliza herramientas llamadas, verificadores funcionales. La estructura del verificador varía, sobre todo, en función del tamaño y número de puntos de la tarjeta a testear. Según esto, la verificación puede realizarse de una manera más automatizada o manual.

3.2.1 Verificadores manuales funcionales

Están basados en conectores o mazos de cables. Se testea la tarjeta mediante la conexión directa entre esta y la electrónica auxiliar. La verificación puede ser semiautomática (mediante software, pero conectando y desconectando cables) o completamente manual (mediante polímetros, osciloscopios, etc.). Este tipo de verificación se usa sobre todo en tarjetas pequeñas, o por petición explicita del cliente (para reducir costes).

3.2.2 Verificadores funcionales por cama de pinchos.

La verificación por cama de pinchos consiste en una superficie con agujas sobre la que se deposita la tarjeta, de tal manera que dichas agujas coinciden con los puntos del circuito (de la tarjeta) que se quieren testear (Test Points). La punta de contacto del pincho se presiona contra la tarjeta, y sobre la otra punta se suelda o se enrolla (Wrapping, Grapinar) el cable que va a la electrónica del verificador.

Figura 3 – verificador. cama de pinchos. Exterior (izquierda), Interior (derecha).

Lo que varía de un verificador a otro es el montaje mecánico de la caja que alberga la electrónica. La elección se hace, sobre todo, en base al tamaño de la tarjeta a testear y de la electrónica auxiliar que haya que montar. A continuación, se muestran algunos ejemplos:

3.2.2.1 Verificador Wayne Kerr (WK)

Es una herramienta muy potente, la cual consta de multitud de módulos programables, como, por ejemplo, módulos de potencia, relés, osciloscopio, generador de señales, cargas, etc. Está pensado para aquellas tarjetas con muchos tipos distintos de entradas y salidas.

Figura 4 – Verificador Wayne Kerr.

La cama de pinchos se monta sobre una caja comercial, lo cual da la ventaja de poder montar o intercambiar varias camas en la misma caja. Dicha caja se conecta directamente al WK.

Figura 5 – Montaje para el Wayne Kerr.

3.2.2.2 Mesa neumática

Cuando se tiene una tarjeta grande y con una verificación no tan complicada (sino se haría para le WK), se utiliza la cama neumática. Lo que ofrece esta herramienta es que garantiza una buena conexión entre los pinchos y la tarjeta, ya que utiliza cilindros neumáticos para presionar la tarjeta, dándole más estabilidad y sobre todo uniformidad en la presión.

Figura 6 - Verificador en mesa neumática.

3.2.2.3 Verificadores con accionamiento manual

Se utiliza para tarjetas pequeñas que puedan ser presionadas uniformemente mediante una brida manual. Constan de una base de apoyo para la tarjeta, un actuador mecánico y una caja para albergar la electrónica auxiliar.

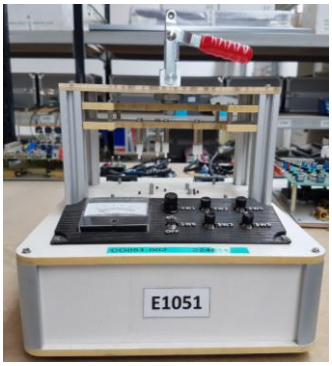


Figura 7 - Verificador en mesa neumática.

Cabe mencionar que es este tipo de configuración es la elegida para el desarrollo del proyecto.

4 Proceso de diseño y fabricación

A continuación, se enumeran los pasos a seguir para llevar a cabo el proceso de diseño y fabricación de los sistemas de test en E. Falcon [6]. Un sistema de test engloba los equipos utilizados para la comprobación de que el montaje de las tarjetas electrónicas se realiza de forma correcta y su funcionamiento es el adecuado. Las tarjetas a comprobar se denominan con el acrónimo DUT (Device Under Test, Dispositivo Bajo Prueba).

Existen tres tipos de trabajos principales diferentes en lo que a los sistemas de test se refiere:

- Diseño y fabricación de un sistema de test.
- Fabricación de un sistema de test diseñado por el cliente.
- Puesta en marcha de un sistema de test suministrado por el cliente.

4.1 Realización de ofertas

Se lleva a cabo el proceso de realización de una oferta para el diseño y fabricación de un sistema de Test. Esto incluye las especificaciones preliminares y la estimación de las horas por trabajador, hasta llegar a un precio final.

Una vez enviada la oferta, es el responsable de ventas el encargado de realizar el seguimiento del estado de la misma.

En caso de que la oferta sea aceptada, el responsable de ventas se encarga de que se introduzcan los datos del cliente en el sistema, así como los pertenecientes al pedido correspondiente.

4.2 Diseño de Sistemas de Test

4.2.1 Entradas para el diseño y fabricación de sistemas de test

Una vez aceptada por parte del cliente la oferta realizada, el siguiente paso es cerrar las especificaciones definitivas. Junto con el cliente se lleva a cabo una fase de análisis y definición detallada de los requisitos funcionales y de desempeño del proyecto. En caso de que estas difieran respecto de las especificaciones preliminares utilizadas para la realización de la oferta, se estos cambios se documentan y archivan. Adicionalmente, la oferta podrá ser revisada en caso necesario.

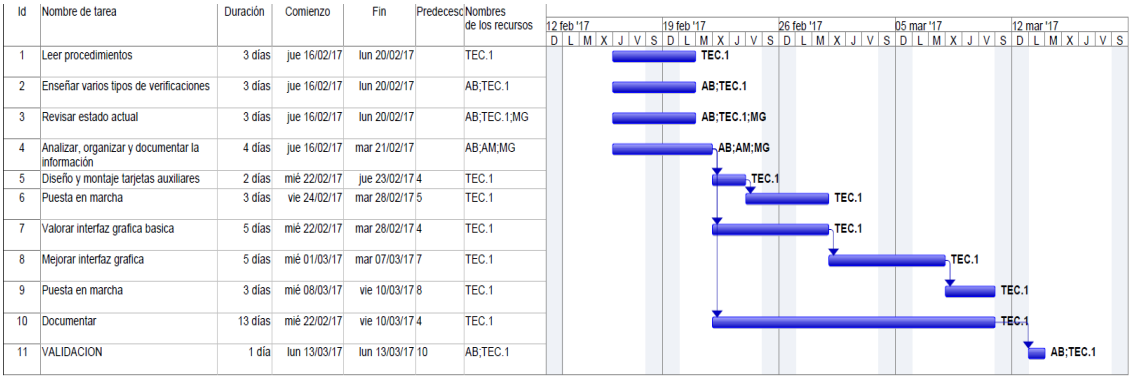
Se definen cuáles van a ser los criterios de aceptación y las pruebas de validación que van a realizarse a los sistemas de test fabricados para certificar que cumplen con los requisitos del cliente. En caso de que este no defina unos requerimientos adicionales concretos, se considerará que el sistema será válido siempre que sea capaz de medir los parámetros definidos previamente con una estabilidad y fiabilidad suficientes como para garantizar el correcto montaje de las DUT fabricadas en serie.

Se hace especial hincapié en que los requisitos o especificaciones definitivas acordadas con el cliente estén suficientemente detalladas y sean claras y sin ambigüedades que puedan dar pie a interpretaciones erróneas o contradictorias.

4.2.2 Planificación del diseño del sistema de test

Una vez se tiene claro y detallado el alcance y los requisitos a cumplir por el sistema de test, se procede a la planificación de los trabajos a realizar.
Primeramente, el responsable del departamento de Ingeniería nombra un líder de proyecto, que será el responsable de coordinar los aspectos técnicos del desarrollo.

En base a la estimación de las horas a invertir para cada tarea, realizada en la fase de oferta, y a los recursos disponibles, el responsable del departamento realiza la planificación, junto con el líder del proyecto, de las tareas a desarrollar, en función de la complejidad y duración de cada una de ellas. También se define el equipo de trabajo y se realiza la asignación de tareas a cada técnico. Como resultado de dicha planificación se confecciona un diagrama de Gant con el planning del proyecto y recursos asociados a cada tarea. Esto solo se hará para proyectos de una relevancia considerable, que o bien sean relativamente largos en el tiempo y/o en los que vayan a participar varios técnicos.



Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesc	Nombres de los recursos
1	Leer procedimientos	3 días	jue 16/02/17	lun 20/02/17		TEC.1
2	Enseñar varios tipos de verificaciones	3 días	jue 16/02/17	lun 20/02/17		AB;TEC.1
3	Revisar estado actual	3 días	jue 16/02/17	lun 20/02/17		AB;TEC.1;MG
4	Analizar, organizar y documentar la información	4 días	jue 16/02/17	mar 21/02/17		AB;AM;MG
5	Diseño y montaje tarjetas auxiliares	2 días	mié 22/02/17	jue 23/02/17	4	TEC.1
6	Puesta en marcha	3 días	vie 24/02/17	mar 28/02/17	5	TEC.1
7	Valorar interfaz grafica basica	5 días	mié 22/02/17	mar 28/02/17	4	TEC.1
8	Mejorar interfaz grafica	5 días	mié 01/03/17	mar 07/03/17	7	TEC.1
9	Puesta en marcha	3 días	mié 08/03/17	vie 10/03/17	8	TEC.1
10	Documentar	13 días	mié 22/02/17	vie 10/03/17	4	TEC.1
11	VALIDACION	1 día	lun 13/03/17	lun 13/03/17	10	AB;TEC.1

Figura 8 – Planning de un proyecto.

4.2.3 Diseño y fabricación del sistema de test

Dependiendo de las necesidades de cada proyecto, puede haber diferentes tipos de sistemas de test:

- Verificador automático basado en cama de pinchos para plataforma Wayne Kerr o plataforma ICT (SEICA).
- Verificador automático c/s cama de pinchos para PC (incluido Boundary Scan)
- Verificador manual autónomo basado en cama de pinchos.
- Verificador manual autónomo.

Cada tipo de sistema de test requiere siempre de un diseño eléctrico. Para esto, se usa el programa de diseño Altium Designer.

Según la complejidad de la electrónica de control, o de otros factores como el número de electrónicas a fabricar, el montaje de la esta se puede realizar o bien sobre un PCB de prototipado o bien se puede fabricar un circuito impreso con la fresadora Cyclone.

Adicionalmente, la mayoría de sistemas de test requerirá además de un diseño mecánico. Para ello se usa el programa Autodesk Inventor. Las piezas sencillas (2D) necesarias, se mecanizan en la fresadora de control numérico (MAPE), para lo cual será necesario generar previamente el correspondiente programa basado en el diseño mecánico realizado.

En aquellos momentos en los que se requiere la fabricación de piezas de geometrías más complejas (3D), se utiliza la impresora de adición de material plástico. De forma análoga al caso anterior, será necesaria la generación previa del programa de control de la impresora.

En el caso de sistemas de test basados en plataforma Wayne Kerr o SEICA (ICT), se requerirá además de un desarrollo del software de test. También puede haber otros casos en los que haya que desarrollar el firmware de test, el cual se grabará en la DUT.

4.3 Fabricación de un sistema de test diseñado por el cliente

Será un caso particular al descrito anteriormente, donde el diseño eléctrico es realizado y suministrado por el cliente. En caso de requerirse de un software de test, es el cliente quien normalmente lo proporciona. Así pues, será aplicable lo descrito en el apartado anterior para las tareas a desarrollar por Falcón.

4.4 Puesta en marcha de un sistema de test suministrado por el cliente

Al igual que antes, este será otro caso particular de lo descrito anteriormente, donde las únicas tareas a realizar por Falcón serán la de codificar los diferentes componentes del verificador y realizar la pauta de verificación y/o grabación.

4.5 Controles de diseño y fabricación del sistema de test

Periódicamente el responsable del departamento de ingeniería se reúne con los líderes de cada proyecto para controlar el estado de las diferentes tareas del desarrollo y fabricación de los diferentes sistemas de test. La periodicidad de dichas reuniones depende de diferentes factores como la complejidad del proyecto, la evolución del mismo o la criticidad de las tareas en desarrollo en ese momento.

En caso de detectar desviaciones respecto a la planificación realizada o de identificar riesgos potenciales de importancia, se establece un plan de actuación para prevenir o corregir las situaciones no deseadas. Dentro de dicho plan se podrá considerar ejecutar acciones como la asignación de nuevos recursos, bien sean internos o externos, o acordar con el cliente una modificación de los acuerdos tomados (como el plazo de desarrollo).

Con el fin de realizar un control de la mano de obra invertida en el proyecto, cada técnico involucrado introduce en el sistema informático un parte de trabajo semanal con las horas dedicadas a cada tarea.

4.6 Salidas del diseño y fabricación del equipo de test

Una vez fabricado el sistema de test, se da comienzo a la fase de depuración y verificación del funcionamiento del sistema, tanto software como del hardware.

En el momento en que todas las pruebas de verificación han sido superadas, se procede a codificar y dar de alta en el sistema informático, tanto el equipo de test como los elementos que lo forman, así como a realizar la pauta de verificación. La pauta de verificación es un documento en el que se detallan paso a paso, las acciones a realizar por el operario para realizar la verificación.

Posteriormente se entrega el equipo al responsable de verificación quien se encarga de revisar y validar el funcionamiento del sistema.

Habrá ocasiones en las que el cliente requiera realizar una validación del sistema de test. En estos casos, además de la validación interna mencionada anteriormente, también será necesario documentar el resultado de dicha validación, bien sea archivando la comunicación escrita remitida por el cliente o bien, en caso de que esta se realice por cualquier medio no escrito, generando un archivo donde se indique cómo se ha producido dicha validación (quién, cuándo, cómo).

Tras la finalización o validación del sistema de test, toda la información del diseño deberá quedar almacenada en el servidor.

4.7 Cambios de diseño y desarrollo de producto

Si tras el proceso de verificación o validación, o durante su uso normal, se detectan deficiencias cuya solución implique una modificación del hardware del comprobador, será necesario realizar un rediseño.
También puede ser el cliente quien solicite llevar a cabo una modificación a un sistema de test ya fabricado.
En ambos casos se generará una nueva revisión del diseño. Todas las revisiones generadas a lo largo de la vida del producto, deberán ser guardadas en la carpeta del proyecto. Los cambios realizados en cada una de las revisiones, así como la causa o solicitante de la modificación, serán recogidos y documentados en la carpeta del proyecto.

5 Documentación inicial

Todo proyecto parte de una etapa de negociación entre cliente y fabricante, en donde se definen los puntos a seguir y la forma de llevarlo a cabo. Después de esto, el cliente aportará la documentación necesaria para iniciar el desarrollo del proyecto.
Cabe mencionar que algunas de los puntos expuestos en la documentación inicial pueden variar a lo largo del desarrollo del verificador.

5.1 Oferta

Se recibe la petición por parte del cliente para el montaje y verificación de una tarjeta para el control de las luces de un remolque. En base a los esquemas eléctricos y a una descripción del funcionamiento de la tarjeta, se realiza un estudio económico por parte de la empresa para realizar una oferta al cliente.
Dicha oferta contiene una serie de especificaciones para la realización del verificador, las cuales se recogen a continuación.

5.1.1 Descripción del verificador

Se recoge en un documento el estudio previo inicial para el diseño y fabricación por parte de la empresa Electrónica Falcón (en adelante, Falcón) del equipo de test necesario para la comprobación de las tarjetas denominadas T-CTRL para el cliente.
Se trata de fabricar un verificador basado en una cama de pinchos, los cuales se conectan a los puntos de test de la tarjeta accionando una plataforma mecánica manual. Además, será necesario emplear una electrónica auxiliar adicional para cambiar los niveles de tensión y para aislar la electrónica del verificador de la propia tarjeta a testear.
La verificación estará controlada por el software creado por Falcón, el cual se comunicará con un ordenador con Windows, mediante el puerto serie.
El cliente facilitará a Falcón una tarjeta ya montada y verificada (maqueta) para utilizarla durante el desarrollo y puesta en marcha del verificador.

5.1.2 Plazos y costes

El plazo de desarrollo estimado para la fabricación del verificador es de diez semanas laborales a contar desde el momento de la formalización del correspondiente pedido y una vez cerradas las especificaciones definitivas del proyecto. Dicho plazo podrá variar en función de los plazos de suministro tanto del material (maqueta y componentes) como de la información requerida (ficheros definitivos de la PCB, secuencia definitiva).

Se fija el tiempo de la verificación de cada tarjeta en, como máximo un minuto. Esto incluye; retardos y tiempos de espera por el propio funcionamiento de la tarjeta, grabación, verificación funcional y tiempo de manipulación del operario.

A continuación, se muestran los aspectos tenidos en cuenta para el cálculo de la fabricación del verificador:

- Diseño: realización de planos mecánicos, esquemas eléctricos, software.
- Montaje eléctrico y mecánico.
- Materiales.
- Puesta en marcha.
- Documentación.

Los costes de fabricación se explicarán con más detalle en el presupuesto (apartado 13).

5.1.3 Resumen de tareas

Se resumen a continuación las funciones de cliente y fabricante en el proyecto.

El cliente se encarga de suministrar:

- Esquemas eléctricos: los esquemáticos de la tarjeta con todos los valores de los componentes y señalando los puntos que se van a testear.
- Archivos para la creación de la PCB (Gerbers): rutas, taladros, lista de materiales.
- Archivo con el programa a grabar (Firmware): fichero con extensión .hex
- Los componentes a montar en la tarjeta.
- que contiene el código del programa que se debe grabar en el microcontrolador de la tarjeta.
- Secuencia de verificación: describe los pasos a seguir para comprobar el funcionamiento de la tarjeta.

E. Falcón se encarga de:

- Gestionar la fabricación de la PCB.
- Montar los componentes.
- Grabar el firmware a las tarjetas.
- Verificar el funcionamiento de las tarjetas.
- Recoger los resultados de todas las tarjetas en un fichero informático.

Uno de los aspectos más importantes a la hora de diseñar el probador es la velocidad de la verificación, la cual no debe exceder el minuto. Se debe intentar reducir el tiempo lo más posible, ya que, teniendo en cuenta que son diez mil unidades, una reducción de cinco segundos por tarjeta supondría ahorrar catorce horas de trabajo de un operario.

5.2 Documentación de la tarjeta

El cliente facilita a Falcón los documentos necesarios para realizar el desarrollo de la tarjeta, tales como, esquemas eléctricos, componentes, distribución de componentes en la tarjeta, etc.

5.2.1 Esquemas eléctricos

Por motivos de confidencialidad, los esquemas eléctricos diseñados por el cliente no pueden ser enseñados, sin embargo, es posible realizar un esquema general de ellos (Figura 9). Se muestra a continuación un diagrama de bloques de la DUT.

El componente centrar de la tarjeta es el microcontrolador, donde se graba el programa que hace funcionar al conjunto. Desde este se gestionan todas las entras y salidas, las cuales se llevan al exterior mediante conectores o puertos.

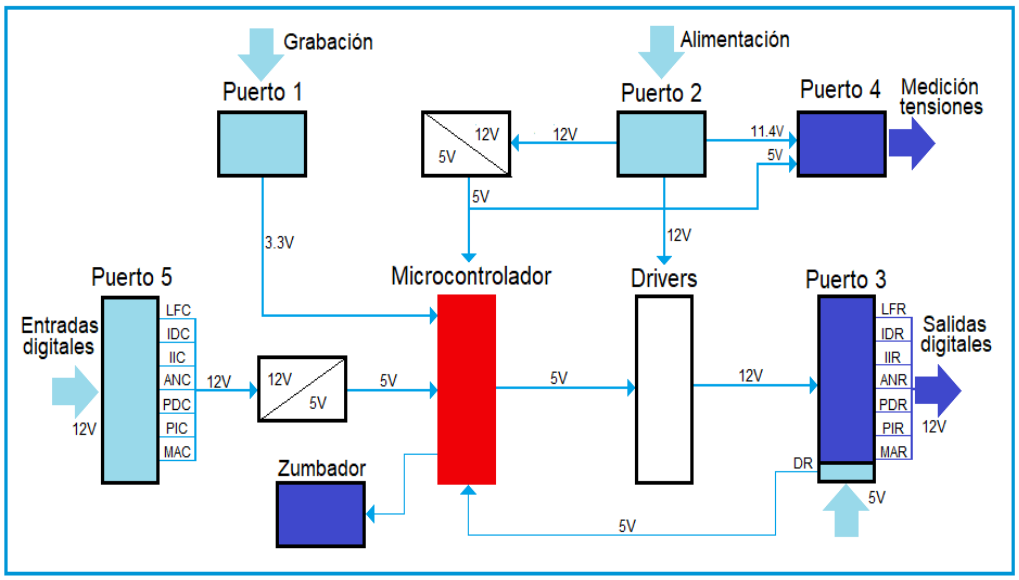


Figura 9 – Diagrama de bloques de la DUT.

La tarjeta consta de 5 puertos:

- Puerto 1: tiene cuatro pines, los cuales se usan para la grabación de la tarjeta. Entre estos se incluye el reset y la alimentación de 5 voltios.

- Puerto 2: por medio de estos pines se alimenta a la tarjeta a su tensión de trabajo, que es de 12 voltios. Se utiliza una fuente de alimentación de corriente continua de 12 voltios y 2.1 amperios. Esta tensión se convierte internamente en 5 voltios para alimentar al microcontrolador y a 11.4 voltios para alimentar los drivers.

- Puerto 3: consta de 7 salidas digitales y una entrada digital.
  Las salidas cuales varían entre 0 (nivel bajo) y 12 voltios (nivel alto). Estas tensiones las dan los drivers, los cuales se encargan de transformar la tensión que da el microcontrolador y amplificarlas, tanto en tensión como de corriente.
  Una de los pines de este puerto es una entrada digital que va directamente al microcontrolador, por lo que funciona a 5 voltios.

- Puerto 4: en este puerto se leen tensiones que utiliza internamente la tarjeta para su funcionamiento.

- Puerto 5: consta de 7 entradas digitales, las cuales se activan aplicando 12 voltios y se desactivan con 0 voltios. Estas entradas pasan por una reducción de tensión para poder adaptarse a los niveles del microcontrolador.

Además de los puertos, la tarjeta incorpora un zumbador, el cual se activa en momentos concretos y debe ser verificado no solo mediante medición de señal sino también por medición de sonido.

5.3 Secuencia de verificación

En la secuencia de verificación, se detalla la forma en la que tiene que funcionar la tarjeta para poder ser dada por válida. Se divide en pasos para una mejor organización.

Se colocarán cargas de 60Ω, y mínimo 2.5 W, en las salidas de la P3.2 a la P3.8. Estando estas permanentemente conectadas.

Se alimentará la tarjeta con una tensión de 12V entre las entradas P2.2, P2.4 (VDD) y P2.1, P2.3 (GND).

- **Paso 1: Tensión 5V.**
  Se medirá la tensión en la salida P1.1 (VDD). Esta deberá ser de 5V.

- **Paso 2: Tensión 11,4V.**
  Se medirá la tensión en la salida P4.4 (VR). Esta deberá ser de 11,4V.

- **Paso 3: Grabación.**
  Se grabará el firmware en la tarjeta.

- **Paso 4: Modo test.**

  Se aplicará nivel bajo a la entrada P3.1 (DR). Esto indica que el remolque no se ha conectado aún.

  Se aplicará nivel alto en la entrada TP32 (modoTest). Esto le indica a la tarjeta que debe entrar en configuración de testeo.

  Se introduce un tiempo de espera de 100 milisegundos.

- **Paso 5: Reset.**

  Se aplicará un nivel bajo a la entrada P1.4 (Reset) durante 500ms. Luego, se aplicará nivel alto durante todo el test.

  Se introduce un tiempo de espera de 200 milisegundos.

  (Debido a la configuración del hardware, para aplicar nivel bajo al reset, hay que aplicar nivel alto a la tarjeta auxiliar).

- **Paso 6: Detección de remolque (Cargas).**

  Se aplicará un nivel alto a la entrada P3.1 (DR) durante todo el test. La tarjeta activará el zumbador, emitiendo un pitido durante un segundo. Se debe comprobar el correcto funcionamiento del zumbador.

- **Paso 7: Tensión 0,4V.**

  Se medirá la tensión de la salida P4.3 (RAN). Esta deberá estar por debajo de 0,4V.

- **Paso 8: PWM On.**

  Se generará un tren de pulsos (PWM) con frecuencia de 100Hz y un ciclo de trabajo del 50%. Esta señal se aplicará simultáneamente a las entradas P5.2 (PIC), P5.3 (PDC), P5.4 (ANC). Pasados 150 milisegundos, se medirá la tensión de la salida P4.1 (ANP) durante un segundo. Esta deberá mostrar un PWM igual al de las entradas.

- **Paso 9: PWM Off.**

  Se aplicará un nivel bajo a las entradas P5.2 (PIC), P5.3 (PDC), P5.4 (ANC). La salida P4.1 (ANP) deberá pasar a nivel bajo.

  Se introduce un tiempo de espera de 200 milisegundos.

- **Paso 10: Entradas/Salidas digitales.**

  Se aplicará un nivel alto a la entrada P5.1 (MAC). Pasados 150 milisegundos, se comprobará que la salida P3.2 (MAR) se encuentre a nivel alto. Se introduce un tiempo de espera de 300 milisegundos.

  A continuación, se aplicará un nivel bajo a la entrada P5.1 (MAC). Pasados 150 milisegundos, se comprobará que la salida P3.2 (MAR) se encuentre a nivel bajo. Se introduce un tiempo de espera de 300 milisegundos.

  Se repite el paso 10 para cada una de las siguientes entradas y sus correspondientes salidas.

Entradas	Salidas	Descripción
P5.1 (MAC)	P3.2 (MAR)	Marcha atrás
P5.2 (PIC)	P3.3 (PIR)	Posición izquierda
P5.3 (PDC)	P3.4 (PDR)	Posición derecha
P5.4 (ANC)	P3.5 (ANR)	Antiniebla
P5.5 (IIC)	P3.6 (IIR)	Interior izquierdo
P5.6 (IDC)	P3.7 (IDR)	Interior derecho
P5.7 (LFC)	P3.8 (LFR)	Freno

Tabla 1 – Entradas/Salidas digitales de la DUT.

Durante la activación de P3.5 (ANR) se comprueba que P4.2 (ANR-R0) se encuentra al mismo nivel.

Para la realización del test harán falta las siguientes entradas:

- 9 entradas digitales.
- 11 salidas digitales.
- 3 entradas analógicas.

5.4 Verificación manual de la tarjeta

Se realiza de manera manual la verificación de la tarjeta según la secuencia de verificación. Las conexiones se realizan mediante cables conectados directamente las fuentes de alimentación de 12 y 5V.
Los principales resultados y conclusiones se recogen a continuación:

Durante la detección de cargas (paso 6), la tarjeta activa todas las salidas del puerto 3 (de P3.2 a p3.8) durante aproximadamente 2 segundos. El consumo por cada salida es de 0,24 amperios, por tanto, durante este tiempo, el consumo total será de 1,68A (7 salidas). Se elige la fuente alimentación en concordancia con este consumo.

Durante el paso 8, se aplica a las entradas P5.2, P5.3 y P5.4 un PWM de 100Hz y 50% de ciclo de trabajo, generado por Arduino y amplificado a 12V mediante un optoacoplador. Se observa la forma de onda en la salida P4.1 mediante un osciloscopio.

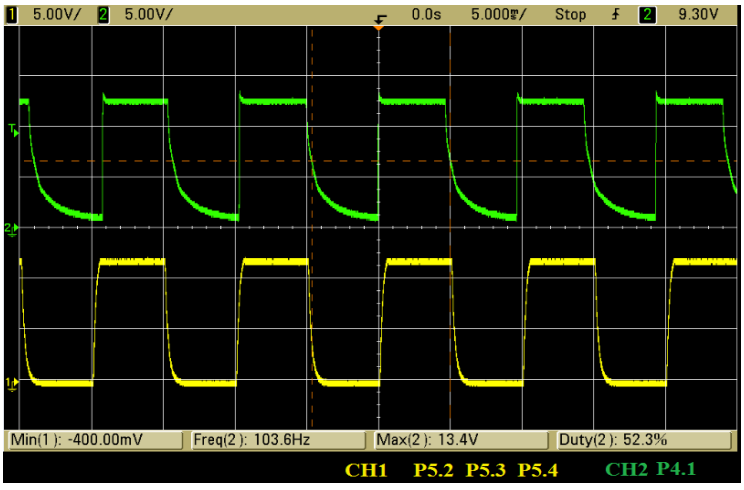


Figura 10 – Señales PWM. paso 8.

Se observa que el PWM de la tarjeta no es exacto, por esto, se amplía la tolerancia en la medida del tiempo en alto y en bajo de la señal a 5±0.5ms.
Cuando se desconecta una carga que estaba conectada en el momento de la detección, y se activa cualquier entrada del puerto 5, la tarjeta emite un pitido continuo, hasta que se desactiva la entrada, imposibilitando la realización del test.
Cuando se realiza la detección de cargas, aquellas salidas del puerto 3 que se encuentren sin carga no se activarán, aun activando su correspondiente entrada en el puerto 5 y aun conectando posteriormente una carga.

Esto implica que se deben tener conectadas las cargas a cada una de las salidas del puerto 3 en el momento de la detección de cargas (paso 6) para poder realizar la verificación. Contrario a lo que se pretendía realizar, cada salida deberá tener conectada su propia carga y estas deberán permanecer conectadas durante todo el test.

6 Selección del microcontrolador (tarjeta comercial)

Un microcontrolador es un dispositivo electrónico programable, compuesto básicamente por tres bloques funcionales: CPU, Memoria y entradas/salidas.
La CPU es el “cerebro” del microcomputador y actúa bajo del control del programa almacenado en la memoria. La CPU se ocupa básicamente de traer las instrucciones del programa desde la memoria, interpretarlas y hacer que se ejecuten [7].

Las tarjetas comerciales funcionan como nexo entre un microcontrolador y el usuario, facilitando su uso. Están compuestas por un microcontrolador y una electrónica auxiliar que da al usuario la posibilidad de manejar todas las características del microcontrolador de manera rápida y sencilla.

Otro motivo por el que se ha extendido el uso de estos dispositivos es que son de código abierto, es decir, los usuarios tienen total libertad para realizar y modificar programas y compartirlos. Gracias a esto, se crean comunidades en las que se comparte información, lo cual es de mucha ayuda ya que permite realizar una programación modular, es decir, usar librerías creadas por otros y adaptarlas al desarrollo propio.

6.1 Estudio del Mercado

Existe una amplia gama de tarjetas en el mercado, basadas en varios tipos de microcontroladores, cada una con sus ventajas y desventajas de cara al usuario. Para poder elegir la más conveniente para el proyecto, se realiza un estudio de algunas de ellas, tales como, Arduino, LaunchPad, Teensy, Raspberry, entre otras. Se puede ver el estudio completo en el apartado 1 de los anexos. (Estudio realizado con la colaboración de S. Nesterenco. Estudiante de ingeniería de la universidad pública de Navarra).
Los aspectos a tener en cuenta en el estudio son:

FUNCIONALIDADES DE LA TARJETA COMERCIAL		
Tensiones de trabajo		
Frecuencia de reloj		
Entrada/Salidas	PWM	
	ADC	
	DAC	
Comunicación	UART	WiFi
	SPI	Ethernet
	I2C	CAN
	USB	Bluetooth
Extensiones y módulos externos compatibles	Relés	Audio
	Multiplexado	Alimentación
	DAC/ADC	
	Comunicaciones	
Timers		
Precio		

Tabla 2 – funcionalidades de la tarjeta comercial. Estudio del mercado.

6.2 Selección y justificación

Debido a la gran comunidad que existe en internet, prestaciones y facilidad de soporte, la tarjeta elegida es la Arduino Mega2560.

Figura 11 – Arduino Mega.
https://www.arduino.cc/en/Main/arduinoBoardMega

Se elige esta tarjeta no solo por las prestaciones del hardware sino también por la gran comunidad de Arduino, una comunidad que no para de crecer y de desarrollar librerías y adaptadores a sus tarjetas.

ASPECTOS TÉCNICOS ARDUINO MEGA2560	
Microcontrolador	ATmega2560
Tensión de operación	5V
Tensión de alimentación	7-12V (máx. 20V)
Entradas/Salidas digitales	54
Salidas con PWM	15
Entradas analógicas	16
Corriente por pin	20mA
Frecuencia del reloj	16 MHz
Tamaño	101x53mm
Peso	37g
Comunicación	I2C, UART, SPI, USB

Tabla 3 – Aspectos técnicos Arduino Mega 2560.
https://www.arduino.cc/en/Main/arduinoBoardMega2560

Esta tarjeta reúne de sobra todos los requisitos necesarios para la realización no solo de este proyecto, sino también para futuras ampliaciones. El amplio número y versatilidad de sus entradas/salidas y el bajo precio en comparación con las demás fueron los factores claves que influyeron en la decisión.

En la Tabla 3 se puede ver un resumen con los aspectos técnicos básicos de la tarjeta.

Se puede consultar la documentación de la tarjeta en la página web de Arduino: https://www.arduino.cc/en/Main/arduinoBoardMega2560.

7 Diagrama de bloques del verificador

Para tener una visión global de la estructura del verificador se muestra a continuación un diagrama de bloques con el conexionado de los diferentes elementos que lo componen.

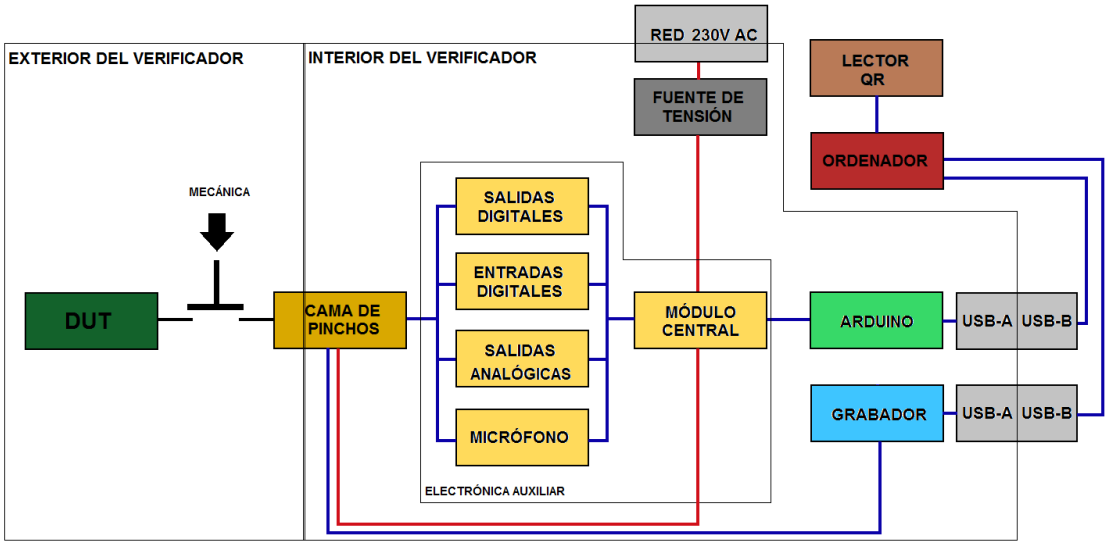


Figura 12 – Diagrama de bloques del verificador.

Según el color de los bloques, se diferencian varios elementos, o grupos de elementos que componen el verificador. La tarjeta a verificar (DUT) se coloca en el verificador, donde, accionando una brida manual, se conecta a la cama de pinchos. Los pinchos se encargan de conectar la DUT con la electrónica auxiliar, que, a su vez, lleva las señales a Arduino, que es el cerebro del verificador.

Se utiliza una herramienta para grabar el firmware en las tarjetas. Este, al igual que Arduino, se conecta al ordenador mediante un puerto USB. En el ordenador se ejecuta la aplicación del verificador (creada mediante Processing).

El verificador se conecta a la red para alimentar la fuente de tensión que da corriente continua al sistema.

A continuación, se muestran las imágenes reales del verificador.

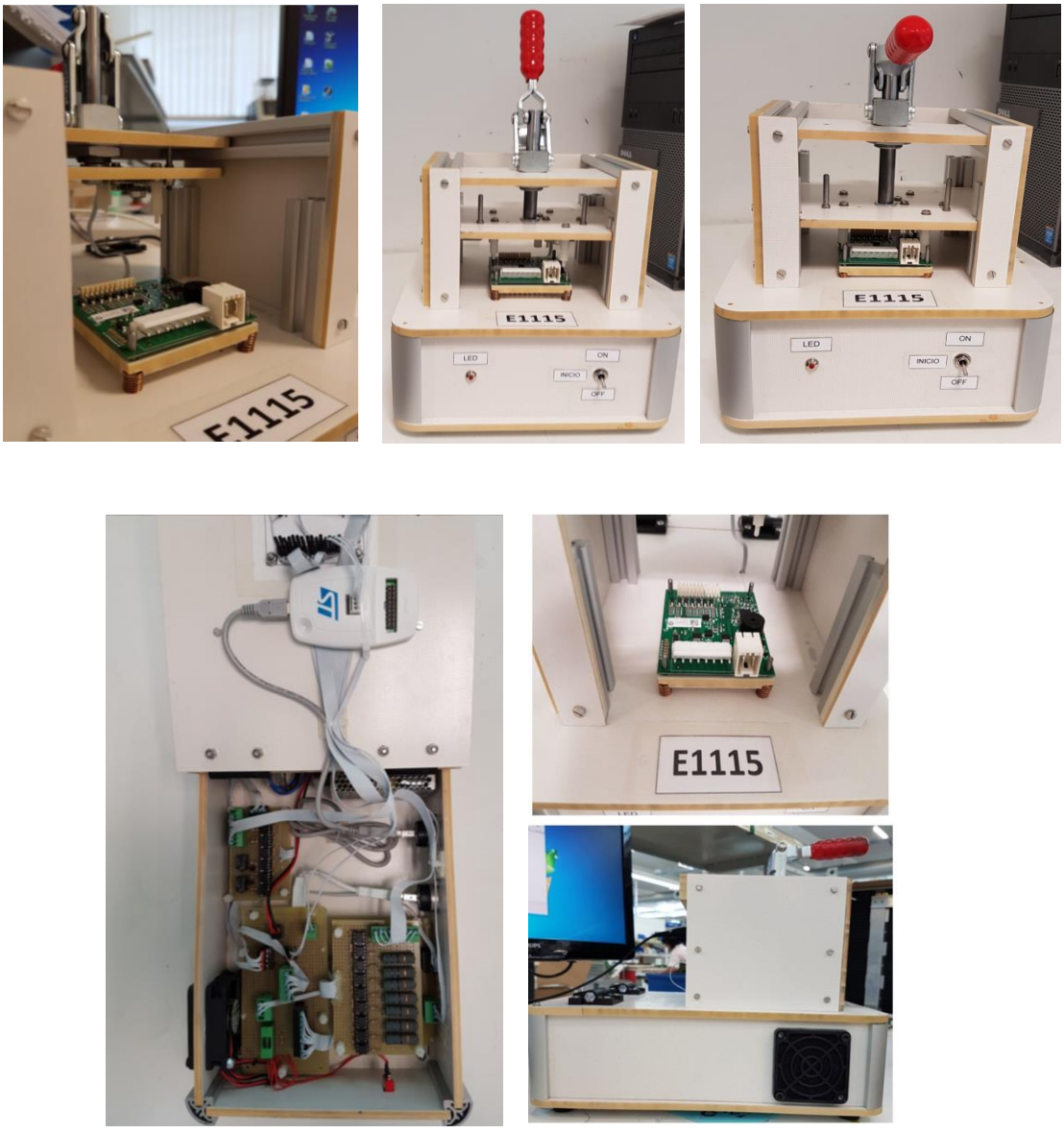


Figura 13 – Fotos del verificador.

7.1 Ordenador

Herramienta desde donde se ejecuta el programa. Ofrece la interfaz gráfica y comunica con los periféricos necesarios para la verificación.

7.2 DUT

La tarjeta a testear, proveída por el cliente, la cual trabaja a 12V y cuyo funcionamiento se explicó en el apartado 4.

7.3 Mecánica y cama de pinchos

Compuesta por una plataforma accionada manualmente, la cual hace presión sobre la tarjeta, conectándola a los pinchos.

7.4 Tarjeta Arduino

Mediante Arduino se testean las entradas y salidas de la tarjeta cliente.
Su tensión de funcionamiento es de 5 voltios, por lo que es necesaria la implementación de una electrónica auxiliar, tanto para aumentar la tensión de salida como para disminuir la tensión de entrada. Además, se asegura el aislamiento entre la DUT y Arduino.

7.5 Grabador ST-Link-v2

El hardware que se usa es el ST-Link-v2, el cual es específico para la grabación de microcontroladores de la familia STM8, que es el que lleva la tarjeta. El grabador se aloja dentro del verificador y se lleva al exterior mediante un conector USB-B, para poder conectarlo al ordenador.
Se adjunta la documentación en el apartado 1 de datasheets.

7.6 Fuente de alimentación

La tarjeta se alimenta con 12 voltios de continua, por lo que se usa una fuente de tensión de ese mismo valor, y en este caso, con una corriente máxima de 2.1 amperios, suficientes para poder realizar el test sin problemas.

Aparte del precio, uno de los aspectos tenidos en cuenta a la hora de elegir la fuente de alimentación es su tamaño, ya que esta deberá ir anclada en el interior del verificador.

Se adjunta la documentación en el apartado 2 de datasheets.

Figura 14 – Fuente de alimentación TXM 025-112.
http://es.farnell.com/b/tracopower

7.7 Lector QR

Para asegurar la trazabilidad de los productos, Falcón utiliza etiquetas con códigos QR (Quick Response Code, código de respuesta rápida). Cada tarjeta lleva asociada su etiqueta, la cual debe ser leída mediante el lector de códigos para guardar sus datos.

Figura 15 – Pegatina DUT.

La pistola para leer los códigos se conecta al ordenador mediante un puerto USB, y es detectada por Windows como si esta fuese un teclado.
La pegatina creada para las tarjetas de este proyecto contiene cuatro campos: código QR, Referencia de la tarjeta, Orden de fabricación y número de tarjeta. En el momento de leer la etiqueta, los valores alfanuméricos se decodifican de la siguiente manera:

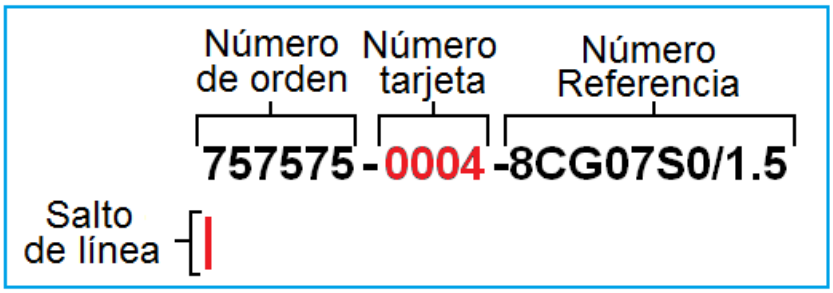


Figura 16 – Codificación de la pegatina

El número de orden o, número de orden de fabricación, es el campo elegido para nombrar los ficheros de resultados.
El lector está configurado para dar un salto de línea al final, esto es muy importante ya que es la manera en la que se detecta que ha finalizado la lectura del código.

7.8 Electrónica auxiliar

La función de este hardware adicional es la de adaptar los niveles de tensión de las señales, ya que la tarjeta a testear trabaja a 12 voltios, mientras que Arduino lo hace con 5 voltios. La forma de realiza esto es mediante relés y optoacopladores, consiguiendo así aislar eléctricamente los distintos elementos del verificador.
Se explica con más detalle en el apartado *9.2.2 Tarjetas auxiliares.*

7.9 Conectores

Se tienen dos conectores tipo USB-A a USB-B para conectar Arduino y el ST-Link al ordenador. Además, se coloca un conector de alimentación a red estándar.

8 Entornos de desarrollo

Se puede dividir el diseño del verificador en 3 grandes partes: hardware, software y mecánica. Se utiliza un programa de desarrollo específico para cada una de estas partes.

Para el diseño del software se utiliza el entorno de programación de *Arduino* y *Processing*, además de scripts en *MS-DOS*. Con Arduino trabaja sobre las entradas y salidas, en Processing se crea la interfaz gráfica y se gestionan ficheros, y en MS-DOS se ejecutan programas y directrices de Windows.

Para la parte hardware se utiliza el programa *Altium designar*, con el que se crean los esquemas eléctricos y la PCB.

La mecánica del verificador se diseña con *Autodesk Inventor 2013*.

8.1 Arduino

Arduino es una gama de circuitos electrónicos open source, basados la mayor parte en un microcontrolador del fabricante Atmel. Estos circuitos integran los componentes necesarios para permitir un uso rápido y sencillo del microcontrolador. Esta simplificación está orientada a hacer accesible a todos, la creación y la programación de objetos o dispositivos interactivos. Estos objetos pueden contener todo tipo de captadores, indicadores luminosos o interruptores que queramos [8].

Arduino es diferente a otras plataformas del mercado debido a las siguientes funcionalidades [9]:

- Es multiplataforma. Funciona en Windows, Macintosh y Linux
- El entorno de programación es sencillo y fácil de usar (está basado en Processing).

- Se programa por USB no por puerto serie. Ordenadores podernos no se fabrican con puerto serie.
- Es código abierto tanto en software como en hardware. Se pueden descargar los esquemas, comprar los componentes y montar cualquier tarjeta de Arduino de manera libre.
- Es barato.
- Cuenta con una comunidad en internet bastante extensa.

La facilidad para interactuar con el microcontrolador, y el hardware, diseñado para acceder de manera fácil y cómoda a las entradas/salidas, agilizan los procesos de diseño y montaje del proyecto a desarrollar. Todo apoyado por una gran comunidad que aporta soluciones y da consejos en los momentos de dificultad.

8.2 Processing

Processing es un lenguaje de programación con entorno de trabajo integrado, utilizado para crear programas visuales, por ejemplo, animaciones, interacciones, imágenes o interfaces gráficas. Al igual que Arduino, Processing es de código abierto y multiplataforma (Linux, MAC OS, Windows), con lo que cuenta con su propia comunidad de desarrolladores, librerías, aplicaciones y documentación [10].

Processing está montado sobre *Java*, un lenguaje de programación gratuito y orientado a objetos. Los objetos son un tipo de funciones especiales. Estas contienen datos y funciones propias, las cuales no pueden ser modificadas desde el exterior, y solo se pueden acceder a ellas mediante métodos definidos por el mismo objeto.
Para entender mejor lo que es un objeto se puede plantear el siguiente ejemplo:

"Pensemos en un objeto tipo 'coche'. Desde el código no podremos cambiar directamente su velocidad, ni el ángulo de sus ruedas. Si lo hiciéramos, podríamos generar saltos de un punto a otro que son en realidad imposibles. El modo de interactuar con este objeto 'coche' será utilizar su volante y su acelerador como elementos de su interfaz, y será el código interno del 'coche' el que se ocupe de modificar la velocidad y la dirección, siempre según un modelo consistente y comprobado" [11].

En lo que concierne a este proyecto, Processing se utiliza para realizar tres funciones: crear la interfaz gráfica de la aplicación, interactuar con el operario mediante el ratón y el teclado, y gestionar ficheros.

8.2.1 Interfaz gráfica

El programa se ejecuta en una ventana de 800x400 pixeles en donde se dan directrices al operario y se muestran mensajes sobre la situación del test.
Es de gran importancia informar al operario sobre el resultado final del test y en caso de fallo, comunicarlo con el debido código.
A la hora de instalar el verificador en la planta de producción, se realiza un documento con los pasos que debe seguir el operario para realizar la verificación (pauta de verificación). Aun así, es conveniente que el propio programa muestre los pasos que debe seguir el operario.

8.2.2 Interacción con el operador

El programa es capaz de detectar la introducción de datos mediante el teclado y el ratón. En varios momentos de la verificación, el operario debe elegir entre una de las opciones mostradas en pantalla. Se muestran las opciones a elegir y un botón o recuadro que se ilumina cuando es seleccionado (Figura 17).

La navegación entre opciones se puede realizar, o bien desplazando el ratón, o usando el tabulador o las teclas de navegación del teclado. Para elegir la opción seleccionada se debe presionar Intro o hacer clic sobre el recuadro correspondiente.

Además, al detectar pulsaciones del teclado, permite obtener los datos dados por el lector de códigos QR.

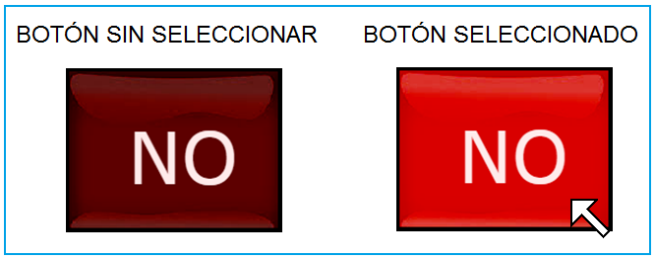


Figura 17 – Selección de opciones por el operador

8.2.3 Gestión de ficheros

En Processing se realiza la creación, modificación y lectura de ficheros de distintas extensiones. En este caso, se utiliza para crear los ficheros de texto (.txt) donde se guardan los resultados de los testes de las tarjetas.
Además, desde Processing se pueden ejecutar programas externos y lanzar aplicaciones. Gracias a esto es posible realizar la grabación de la tarjeta de manera automática, lanzando un fichero que ejecute el programa de grabación.

8.3 MS-DOS

Es un sistema operativo que controla las actividades del ordenador. Este funciona a partir de la introducción de comandos, con los que se consiguen varias funcionalidades, tales como [12]:

- administración de archivos y carpetas
- actualizaciones de disco
- configuración del hardware
- optimización de la memoria
- ejecución de programas

8.3.1 Ficheros Batch

Son archivos de texto guardados con extensión .BAT. En estos se escriben comandos de DOS, los cuales se ejecutarán línea a línea al lanzar el fichero, lo que permite una automatización de procesos.
En este proyecto se crea un archivo batch con los comandos necesarios para ejecutar el programa que se encarga de la grabación de la tarjeta (ST Visual Programmer, en modo CMD). Gracias a esto, la grabación de la tarjeta se hace con el mismo software del verificador y de manera totalmente automática, ahorrando mucho tiempo y evitando errores derivados de la manipulación del programa por parte del operario.

8.4 Autodesk Inventor 2013

Inventor es un software que ofrece herramientas profesionales para el diseño mecánico, la documentación y simulación de productos 3D [13].

Las funciones esenciales del software Autodesk Inventor incluyen: diseño mecánico 3D, diseño de piezas plásticas, diseño de ensambles. Comunicación de diseños, manejo de datos, visualización 3D, documentación y enlace a manufactura. Las herramientas de productividad CAD, integración e interoperabilidad DWG están incluidas [14].

En lo referente al proyecto, se utiliza este software para el diseño de y generación de los planos mecánicos de la caja del verificador.

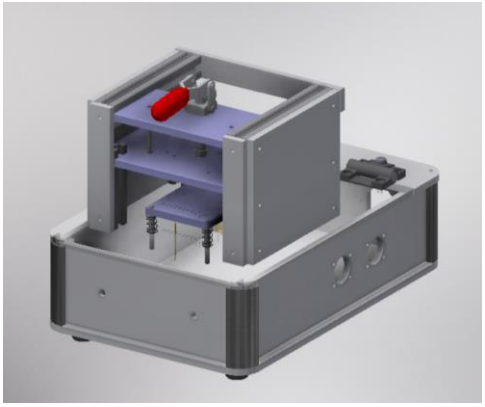


Figura 18 – Ensamblaje del verificador en Inventor.

8.5 Altium designer 9

Altium contiene todas las herramientas requeridas para el desarrollo de una PCB, desde el diseño hasta la fabricación. Consta de varios editores, anidados en un mismo entorno gráfico, facilitando el intercambio de datos [15].
En Altium se diseñan todos los planos eléctricos de la electrónica auxiliar, y también el rutado de la PCB.

9 Hardware

Se recoge a continuación todo lo referente a la electrónica necesaria para el funcionamiento del verificador, ya sea hardware comercial o creado en la propia empresa.

9.1 Planos del cliente modificados

En algunas ocasiones es necesario modificar los esquemas dados por el cliente para tener una mejor organización e interpretarlos de manera más sencilla.

9.1.1 Mapa de puntos numerados

Se modifica el esquema de la serigrafía de la tarjeta para agregar los testpoints. Cada punto a verificar de la tarjeta está conectado a un testpoint mediante una pista creada en PCB. Son estos puntos los que se conectarán con los pinchos y desde donde se accederá a la tarjeta. Se realiza de esta manera porque no es cómodo ni fiable medir directamente sobre la soldadura de un componente cuando hay muchos tamaños distintos.

Se presenta el esquema del posicionamiento de los testpoints en la tarjeta. Los círculos blancos se corresponden con la posición de los pines de los conectores y los grises a la posición de los testpoints. Partiendo de este esquema se realiza la cama de pinchos.

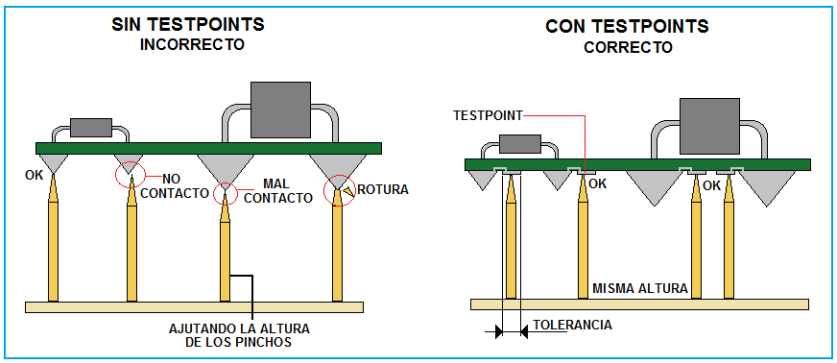


Figura 19 – Uso de testpoints.

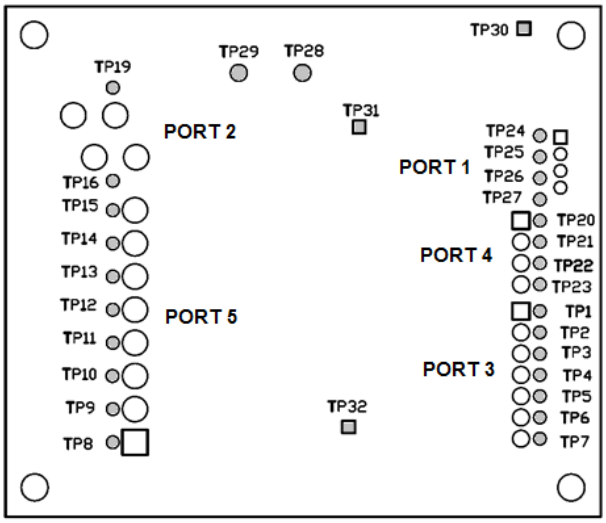


Figura 20 – Mapa numerado.

9.2 Esquemas del verificador

El elemento central del verificador es el microcontrolador, que gestiona el test, en este caso, Arduino. La electrónica auxiliar sirve de nexo entre este y la tarjeta cliente. Además, se incorpora un grabador para el firmware de la tarjeta y un micrófono para detectar la activación del zumbador.

9.2.1 Esquema general

Se presenta un esquema general del verificador, en donde se ven las conexiones entre cada una de las partes que lo componen.

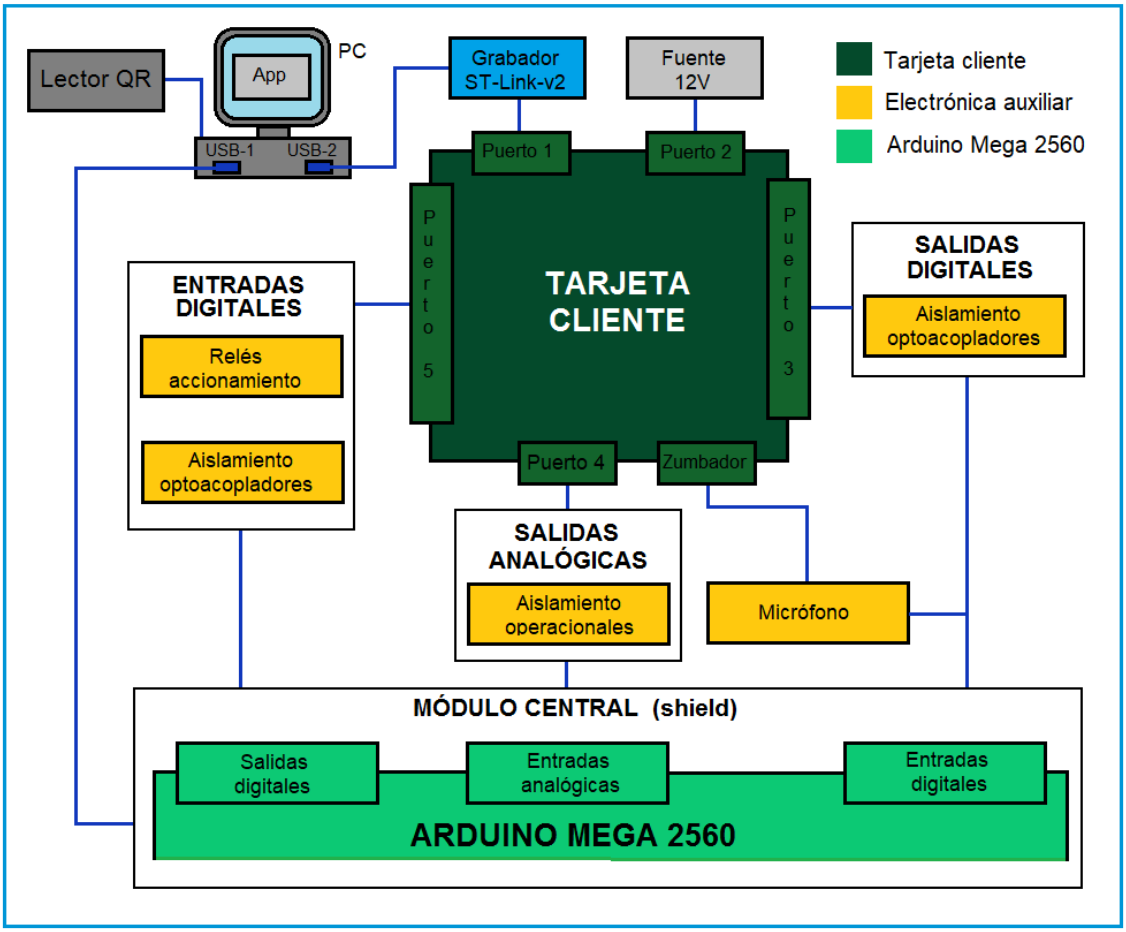


Figura 21 – Esquema general del Hardware.

El esquema completo se puede ver en el apartado 6 de los anexos.

9.2.2 Tarjetas auxiliares

Se dispone de cuatro tarjetas auxiliares para la adecuación de señales entre Arduino y la DUT. Estas son: el módulo central, salidas digitales, entradas digitales y salidas analógicas (las “salidas” y “entradas” están referidas a la DUT).

9.2.2.1 Módulo central (Shield)

Se conecta directamente a la tarjeta de Arduino a modo de shield. Un shield es una tarjeta hecha a medida para conectarse directamente sobre el hardware de Arduino. Los shields son módulos o extensiones que añaden nuevas funciones a la tarjeta de Arduino. En el mercado se pueden encontrar muchas de estas tarjetas, por ejemplo, módulos con relés, shields para comunicaciones (CAN, Ethernet, etc.), entre otros.

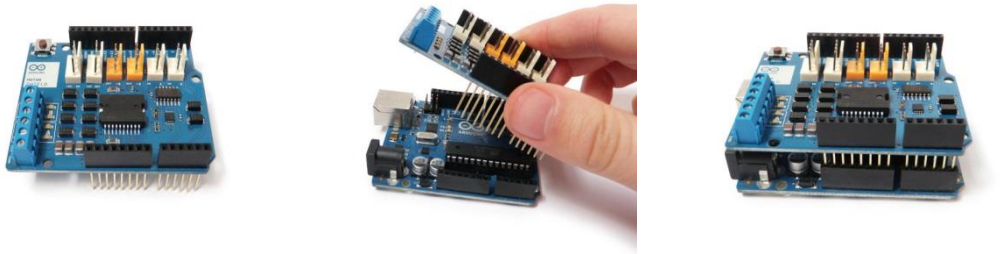


Figura 22 – Conexión de un shield de Arduino.
http://www.instructables.com/id/Arduino-Motor-Shield-Tutorial/

El módulo central básicamente se encarga tomar las señales de las entradas/salidas de Arduino y llevarlas, mediante distintos conectores, a la tarjeta auxiliar correspondiente. Además, también distribuye las tensiones de alimentación (Potencia, 12V y Control, 5V).

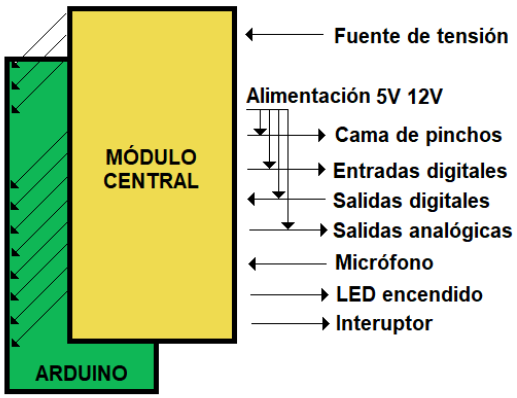


Figura 23 – Esquema módulo central.

El esquema completo se puede ver en el apartado 6 de los anexos.

9.2.2.2 Tarjeta de entradas digitales

En esta tarjeta se coloca la electrónica necesaria para activar las entradas de la DUT desde Arduino. Las entradas que se activan son las correspondientes al, P1.4 (reset), P3.1 (DR), TP32 (modoTest) y al Puerto 5. Todas las entradas se aíslan eléctricamente mediante el uso de optoacopladores.

Con el fin de ahorrar salidas en la tarjeta de Arduino, reducir la electrónica, ganar seguridad y, sobre todo, aumentar la velocidad de la verificación, la activación de las entradas del puerto 5 se hace de una manera distinta a la descrita en la secuencia de verificación (Vale la pena recordar que se trabaja con el puerto 5 en los pasos 8, 9 y 10 de la secuencia de verificación (5.3 Secuencia de verificación).

Durante el paso 8 y 9 se activan las entradas P5.2 P5.3 y P5.4 a la vez.
Durante el paso 10 se van activando todas las entradas del puerto 5 y leyendo su salida correspondiente del puerto 3. En lugar de ir una a una, se activan primero las entradas impares (P5.1, P5.3, P5.5 y P5.7) y luego las pares. En ambos casos, se lee el estado de todas las salidas del puerto 3.

Conexionado interno de la tarjeta: cuando se activa una entrada del puerto 5, esta, activa una salida del puerto 3.

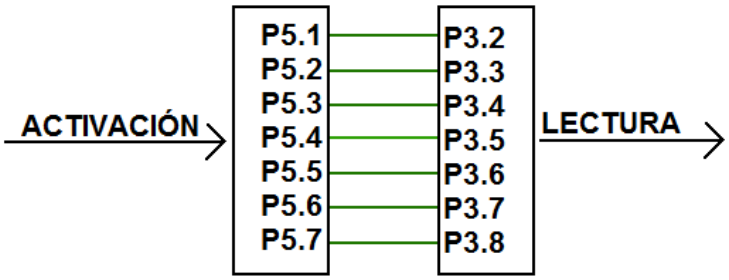


Figura 24 – Conexionado interno, puerto 5 y puerto 3. DUT

Secuencia indicada por el cliente: se activa una entrada y se lee su salida correspondiente, luego se desactiva la entrada y se vuelve a leer la salida. Se repite para las siete entradas.

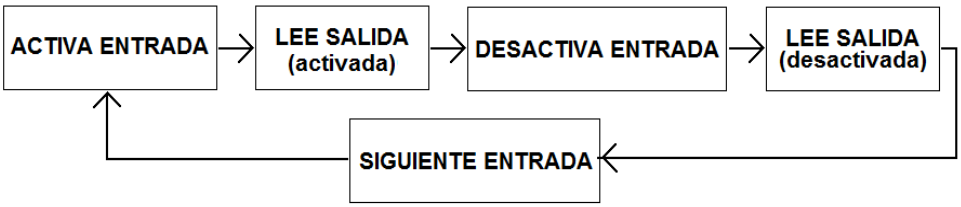


Figura 25 – Descripción del paso 10 de la secuencia de verificación.

Secuencia elegida: se realiza solo en dos pasos en lugar de siete. Al comprobar primero las pares y luego las impares, leyendo todas las salidas a la vez, se pueden detectar posibles cortocircuitos entre patillas consecutivas.

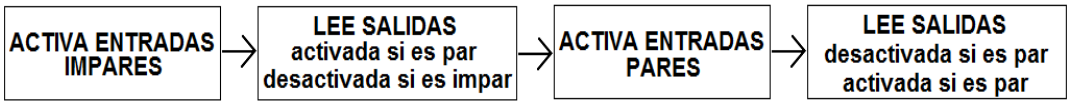


Figura 26 – Modo de realización del paso 10.

Para realizar esto se utilizan tres salidas de Arduino; una para la señal de activación/desactivación (VP5) y dos para los relés que configuran la conexión de las entradas. Se muestra a continuación una tabla de estados:

Relé		Puerto 5							
B	**A**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Paso**
0	0	L	VP5	VP5	VP5	L	L	L	8 y 9
0	1	VP5	L	VP5	L	VP5	L	VP5	10
1	0	L	VP5	L	VP5	L	VP5	L	10
1	1	NO SE USA							

Tabla 4 – Configuración de relés para activar el puerto 5.

La tarjeta cuenta tres conectores:

Conector J5: recibe las señales de activación/desactivación desde el módulo central.

Conector JC: transmite las señales a la DUT por la cama de pinchos.

Conector JA3: recibe las tensiones de alimentación desde el módulo central.

El esquema completo se puede ver en el apartado 6 de los anexos.

9.2.2.3 Tarjeta de salidas digitales

En esta tarjeta se coloca la electrónica necesaria para leer en Arduino las salidas de la DUT. Las salidas a leer son las correspondientes a los puntos P4.1 (ANP), P4.2 (ANR-0) y al puerto 3. Cabe destacar que a las salidas del puerto 3 se conectan las cargas descritas por el cliente en la secuencia de verificación (60Ω, y mínimo 2,5 W).

Todas las salidas se aíslan eléctricamente mediante el uso de optoacopladores.

La tarjeta cuenta tres conectores:
Conector J3: Recibe las señales de la DUT desde la cama de pinchos.
Conector JD: Transmite las señales al módulo central.
Tira de pines T2: Recibe las tensiones de alimentación desde la tarjeta de salidas analógicas.

El esquema completo se puede ver en el apartado 6 de los anexos.

9.2.2.4 Tarjeta de salidas analógicas

Lee aquellas salidas de la DUT que deben tener un valor concreto de tensión. Dichas salidas son: P1.1 (5V), P4.4 (12V) y P4.3 (0,4V). Esto se realiza mediante amplificadores operacionales de aislamiento, los cuales, como su nombre indica, aíslan eléctricamente y amplifican/reducen de manera lineal la tensión en su entrada.
El componente que se usa es el ADum3190, cuya hoja de características se adjunta en el apartado 3 de datasheets.

Este amplificador analógico transforma un rango de tensiones de entrada (máximo 20V) en otro rango de salida (máximo 5V). Entre más estrecho el rango a medir, más resolución se obtendrá; no es lo mismo transformar tensiones entre 0 y 2 voltios que transformar tensiones entre 0 y 12 voltios, ya que el rango de salida varía muy poco.

Se eligen los rangos de entrada en función del valor que se debería leer en la salida correspondiente:

Salida	Valor a medir	Rango de entrada (vIn)	Rango de salida (vOp)
P1.1	11,4V	0-15V	0-3,5V
P4.4	5V	0-15V	0-3,5V
P4.3	0,4V	0-2V	0-5V

Tabla 5 – Rangos elegidos en el ADum3190.

Se deja algo de margen a la hora de elegir los rangos de entrada para poder medir también tensiones fuera de rango (detección de errores).

El esquema completo se puede ver en el apartado 6 de los anexos.

Se realizan pruebas para comprobar empíricamente los valores de cada amplificador. Estas pruebas se realizan con la tarjeta montada y conectada a la DUT para asegurar el buen comportamiento de estos en condiciones de operación.
También se realiza una gráfica para comprobar la linealidad de los amplificadores. Los datos medidos se ajustan a una recta, y estos valores se usan para calcular, mediante el programa de Arduino, el valor real medido.

$$y = m.X + b \quad \rightarrow \quad vOp = m.vIn + b$$

Se puede ver el estudio completo en el apartado 2 de los anexos.

Se obtuvieron los siguientes resultados:

Salida	Ecuación Lineal y=mx+b
P4.4	vOp = -0,2197vIn + 3,4803
P4.3	vOp = -1,2489vIn + 4,8539
P1.1	vOp = -0,22212vIn + 3,2999

Tabla 6 – Resultados pruebas ADum3190

La tarjeta cuenta tres conectores:
Conector J4: Recibe las señales de la DUT desde la cama de pinchos.
Conector JB: Transmite las señales al módulo central.
Conector JA2: recibe las tensiones de alimentación desde el módulo central.
Tira de pines T2: conecta con la tarjeta de salidas digitales para darle las señales de alimentación.

9.2.3 Sensor de sonido (micrófono)

En el paso 6 de la secuencia de verificación se ha de detectar la activación del zumbador de la DUT. La manera de realizar la comprobación es mediante un sensor de sonido, el cual entrega una señal de cero a cinco voltios en función de la cantidad de sonido que detecte.
Se adjunta la documentación en el apartado 4 de datasheets.

El micrófono entrega una tensión de entre 0 y 5V en función de la cantidad de sonido que detecte.
Se conecta a una entrada analógica de Arduino, la cual tiene una resolución de:

$$\frac{5V}{2^{10}bits} = 4{,}88\frac{mV}{bit}$$

Se elige un valor límite para la detección (umbral). Una lectura por encima de este límite indica que se está detectando el pitido del zumbador.

Cada vez que haya una lectura por encima del umbral, se incrementa un contador. Pasado el tiempo de detección (1 segundo) se comprueba el valor de la variable contador. Este valor debe estar comprendido entre un límite inferior (limInf) y uno superior (limSup).
El límite inferior asegura que el pitido no dura menos de lo que debería.
El límite superior asegura que solo se detecta el pitido y no otros ruidos externos.

Las pruebas realizadas al sensor y el posicionamiento en el verificador se comentarán más adelante en el apartado 10.3.1.

9.2.4 ST-Link v2

El ST-Link es un depurador/grabador en circuito (ICD, In Circuit Device), es decir, que permite la grabación del microcontrolador sin necesidad de extraerlo o desconectarlo de su tarjeta. Es una herramienta específica para los microcontroladores de la familia STM8 y STM32 de STMicroelectronic. Utiliza una interfaz USB de alta velocidad para comunicarse con los softwares de STVD (ST Visual Develop) y STVP (ST Visual Programmer), este último es el utilizado en este proyecto para grabar el firmware en la DUT [16].

Figura 27 – ST-Link-v2.
http://www.st.com/en/development-tools/st-link-v2.html

9.3 Diseño de la PCB

Se diseña una PCB para albergar toda la electrónica auxiliar necesaria para el proyecto. Debido al tamaño, se decidió realizar dos tarjetas, las cuales se conectarían entre sí mediante un mazo de cables. Las pistas se realizarían mecánicamente (taladros) en lugar de químicamente, lo cual complicaba el diseño al tener que realizar pistas más grandes y separadas entre sí. Además, se realizaban cambios en la electrónica continuamente por parte del cliente, obligando a modificaciones sustanciales en el diseño e la PCB.

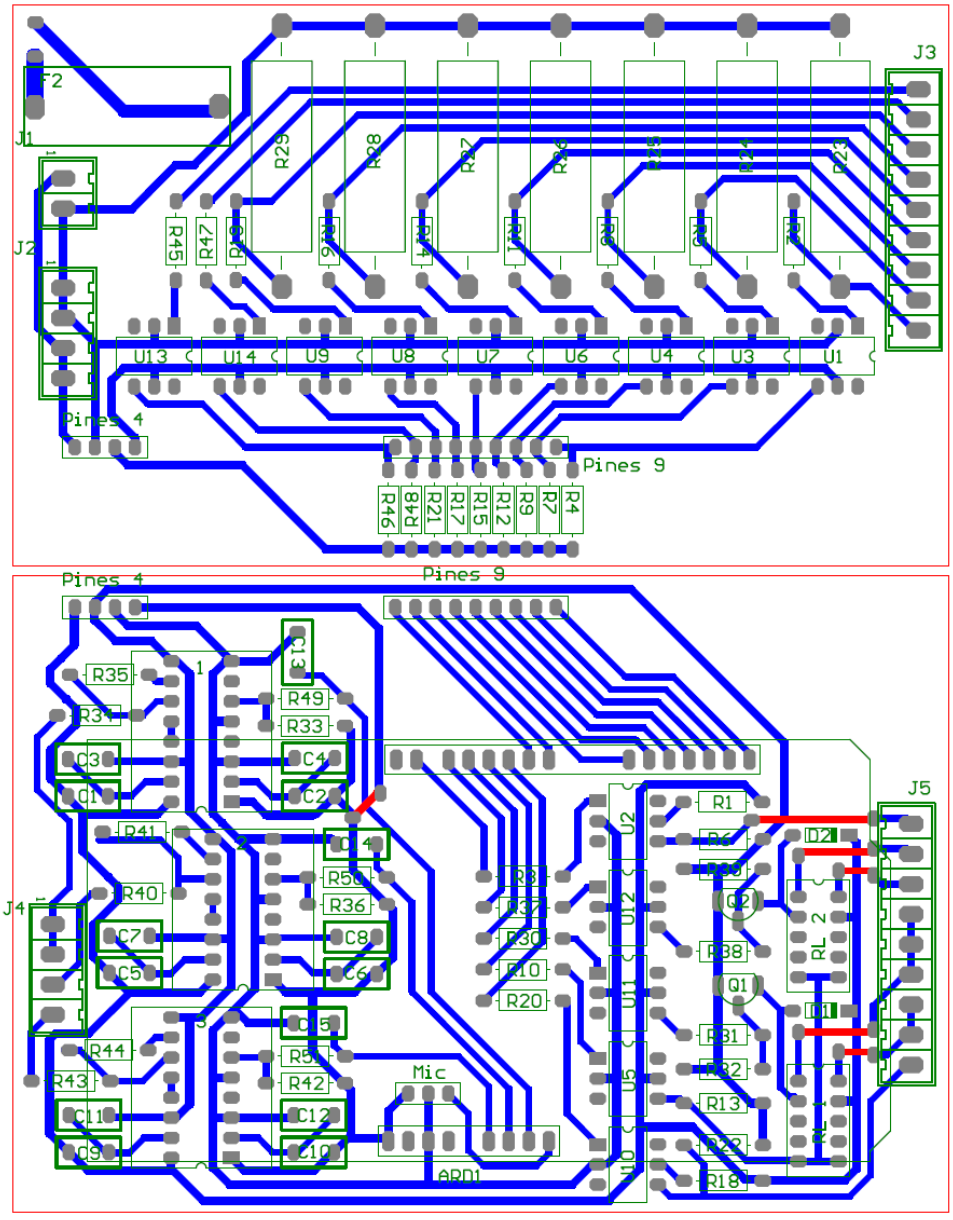


Figura 28 – PCB de la electrónica auxiliar

Al finalizar el diseño de la PCB se procedió a enviarla a mecanizar. Sin embargo, problemas de precisión con la taladradora obligaron a abandonar la vía de la PCB. Como se puede ver a continuación, las pistas no eran lo suficientemente buenas, algunas incluso estaban obstruidas, y el principal problema era que muchos de los paths (donde se suelda el componente) desaparecerían a la hora de taladrar el agujero. Solo se llegó a mecanizar una de las dos placas.

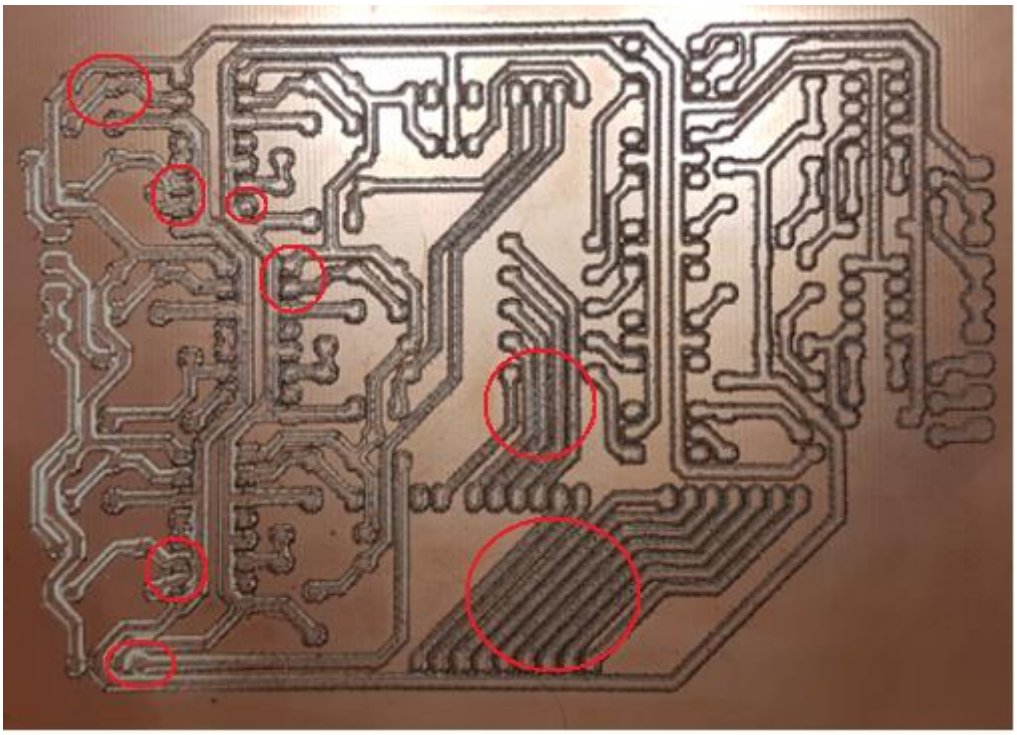


Figura 29 – PCB mecanizada.

Debido a estas dificultades se decidió realizar manualmente todas las tarjetas en placas de prototipado. Los cambios en el hardware siguieron sucediéndose, por lo que, al final, resultó siendo más cómodo hacerlo a mano.

9.4 Conexionado y distribución interna

Se muestra la distribución de las tarjetas auxiliares y el hardware en el interior del verificador, y sus conexiones.

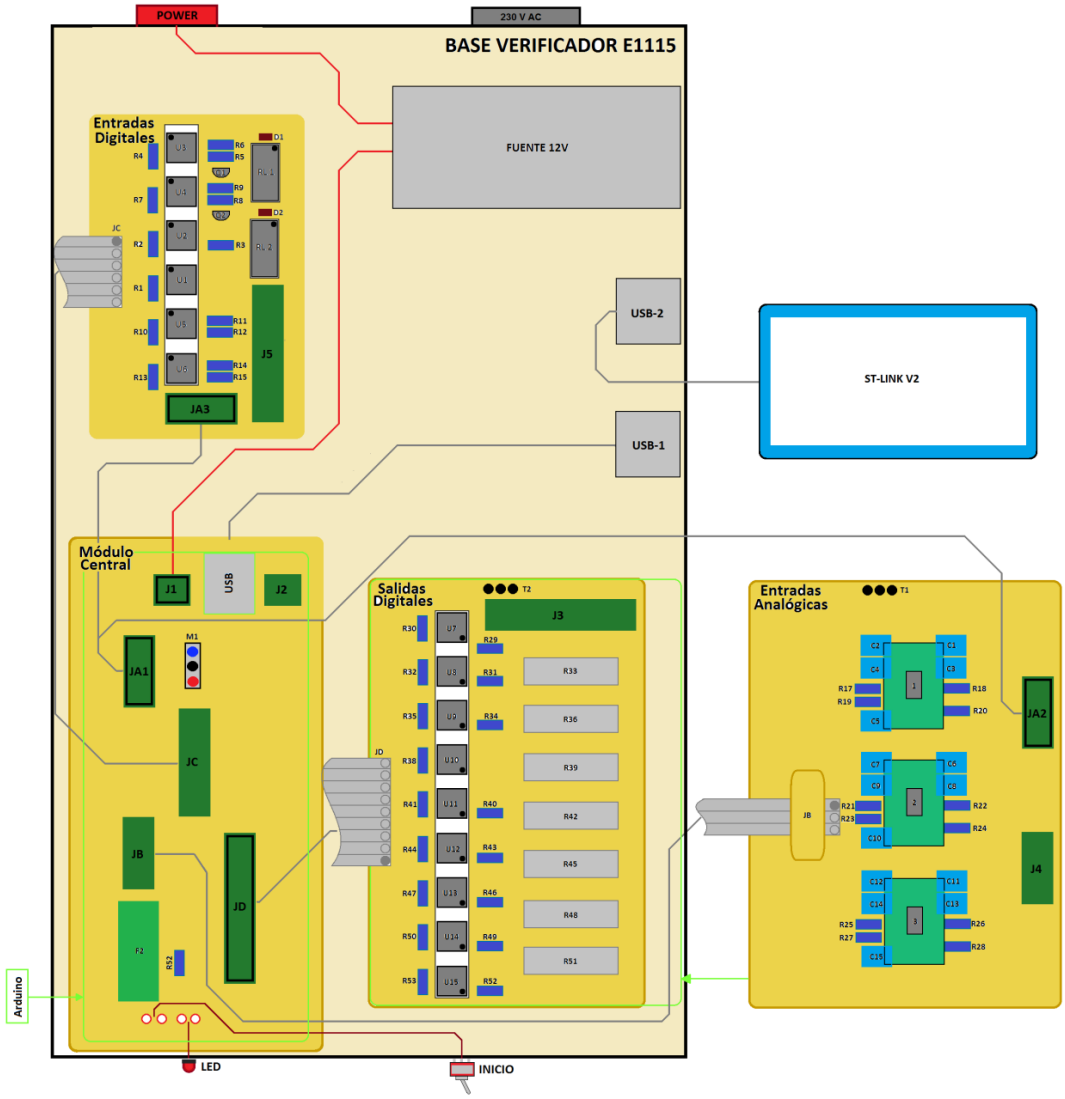


Figura 30 – posicionamiento y conexionado interno del verificador.

La tarjeta de entradas digitales se coloca en la parte superior izquierda junto a la fuente de alimentación.

Arduino se coloca en la parte inferior izquierda, y sobre este se conecta el módulo central.

La tarjeta de entradas analógicas se coloca en la parte inferior derecha, y sobre esta se conecta la tarjeta de salidas digitales.

El ST-Link se sujeta a la parte superior de la tapa, junto a los pinchos.

10 Mecánica

La parte mecánica del verificador está compuesta por la cama de pinchos, el mecanismo de conexión de la tarjeta y la caja que une todos los componentes y alberga la electrónica en su interior.
La caja se realiza con el material EP 105, el cual consiste en láminas a alta presión con un tratamiento superficial anti electricidad estática [17].

10.1 Planos mecánicos

10.1.1 Base

Se mecaniza una pieza de 300x200x6mm. Se realiza un redondeo con un radio de 19mm en las esquinas.
Se realizan cuatro agujeros pasantes en cada esquina para alojar las patas de goma donde se apoyará el verificador.
Se realizan tres agujeros pasantes en cada esquina para atornillar los perfiles metálicos en donde se encajarán las distintas partes del verificador.
El esquema completo se puede ver en el apartado 7 de los anexos.

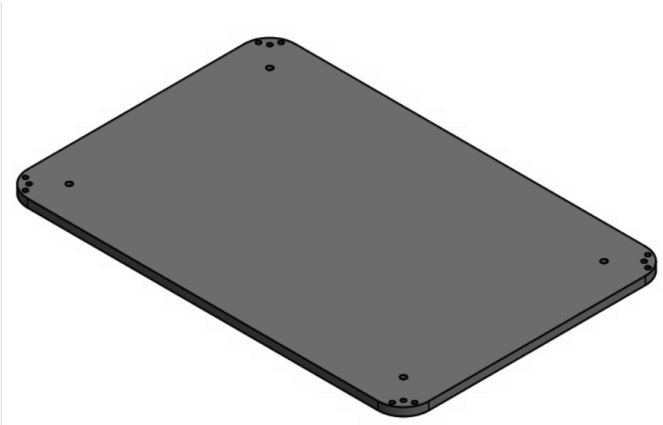


Figura 31 – Base.

10.1.2 Trasera

Tiene unas dimensiones de 170x70x6mm. Se mecanizan los alojamientos para el conector de red y el interruptor de encendido.
El esquema completo se puede ver en el apartado 7 de los anexos.

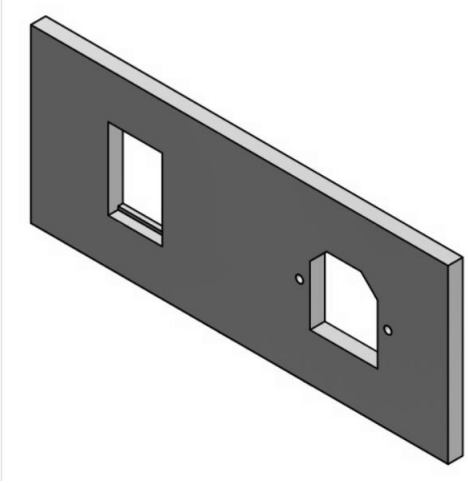


Figura 32 – Trasera.

10.1.3 Laterales

Tienen unas dimensiones de 270x70x6mm.
En el lateral derecho se mecanizan los dos alojamientos para los puertos USB.
El esquema completo se puede ver en el apartado 7 de los anexos.

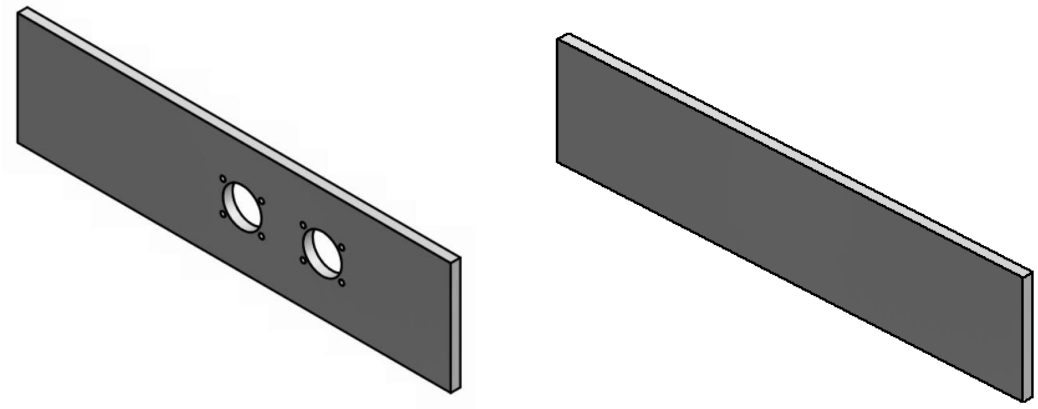


Figura 33 – Laterales

10.1.4 Frontal

Tiene unas dimensiones de 170x70x6mm. Se mecanizan dos agujeros pasantes, uno de 6mm para alojar un LED, y otro de 10mm para alojar un interruptor.

El esquema completo se puede ver en el apartado 7 de los anexos.

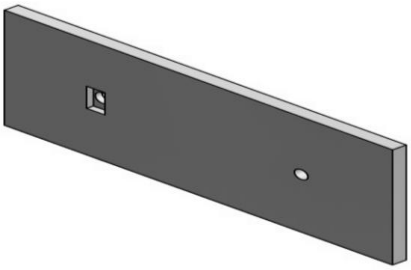


Figura 34 – Frontal.

10.1.5 Tapas

Se divide en dos partes, las cuales se unen mediante dos bisagras, con el fin de poder acceder al interior de la caja.

La parte fija tiene unas dimensiones de 200x50x6mm y se conecta a la base mediante los perfiles de las esquinas.

La parte móvil es una de las más importantes ya que en este se realizan los agujeros donde se alojarán los pinchos. Además, se realizan los agujeros donde se colocarán los elementos que sujetarán la brida.

El esquema completo se puede ver en el apartado 7 de los anexos.

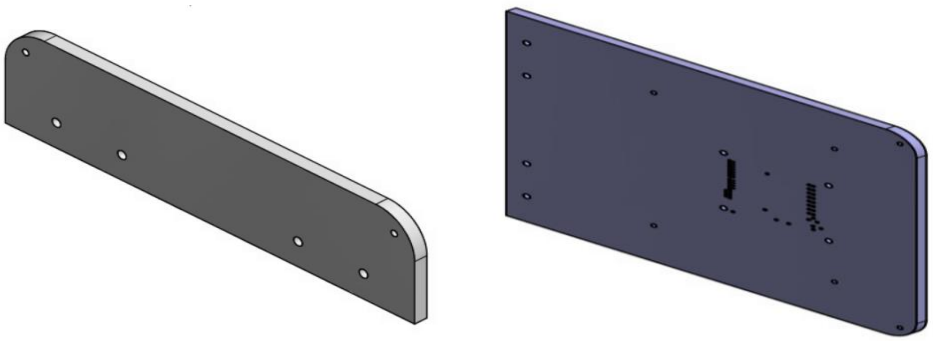


Figura 35 – Tapas.

10.1.6 Soporte tarjeta

Sobre esta se colocará la tarjeta. Al igual que en la tapa, los pinchos deben atravesarla para hacer contacto con la tarjeta. Se colocan unas guías para ajustar esta junto con la tapa.

La pieza descansa sobre cuatro muelles, los cuales son necesarios a la hora de presionar la tarjeta contra los pinchos.

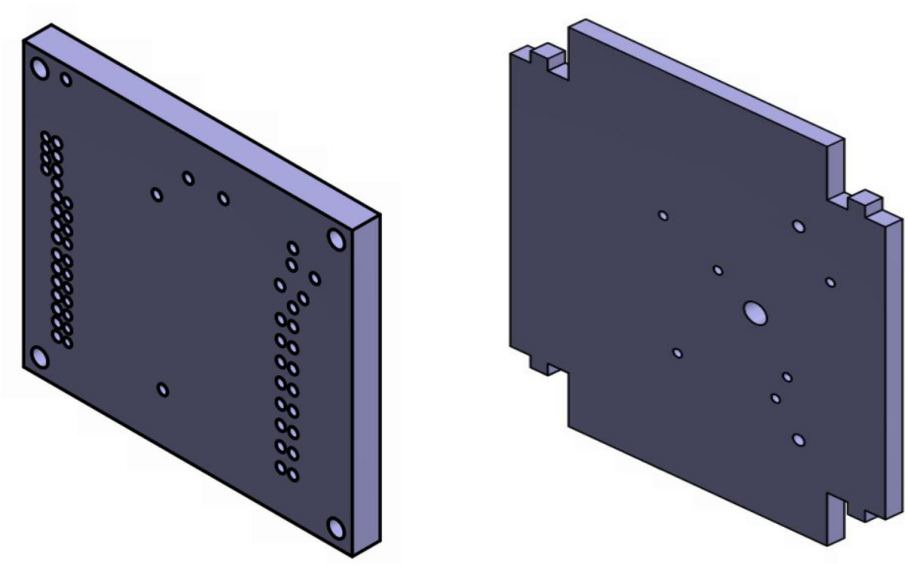
Tiene unas dimensiones de 61x71x6mm.

El esquema completo se puede ver en el apartado 7 de los anexos.

10.1.7 Empujador

Es el encargado de presionar la tarjeta contra los pinchos. Se realiza de tal forma que baje guiada por los perfiles que soportan la brida y para que haga tope, evitando que se desacople del verificador.

El esquema completo se puede ver en el apartado 7 de los anexos.



(a) (b)

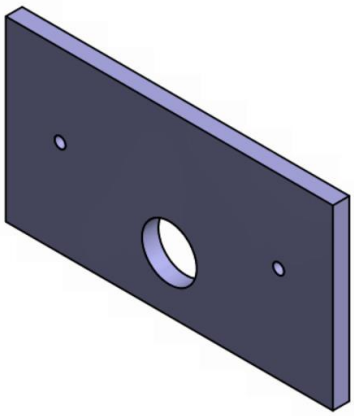
Figura 36 – Soporte tarjeta (a). Empujador (b).

10.1.8 Soporte brida

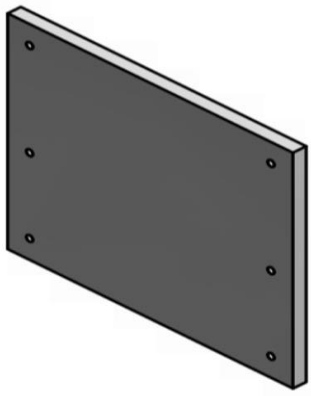
Consta de cuatro partes diseñadas para permitir el movimiento del empujador mediante la brida, de tal manera que el operario pueda introducir la mano para retirar la tarjeta después de una verificación.
La placa donde se atornilla la brida debe poder deslizar fácilmente por las guías horizontales, por lo que su tamaño es un poco menor.

Dimensiones:

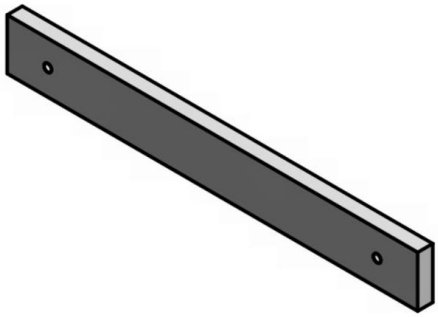
- Soporte lateral: 110x130x6 (dos unidades).
- Soporte frontal: 110x20x6 (dos unidades).
- Soporte trasero: 162x20x6.
- Soporte brida: 119x65x6.



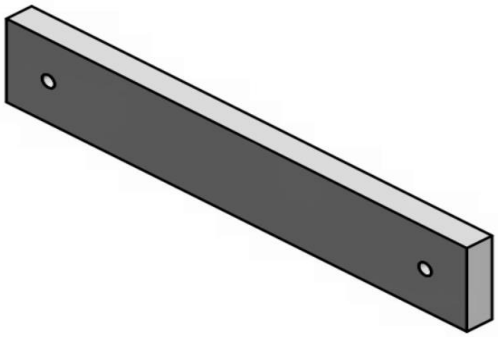
(a)



(b)



(c)



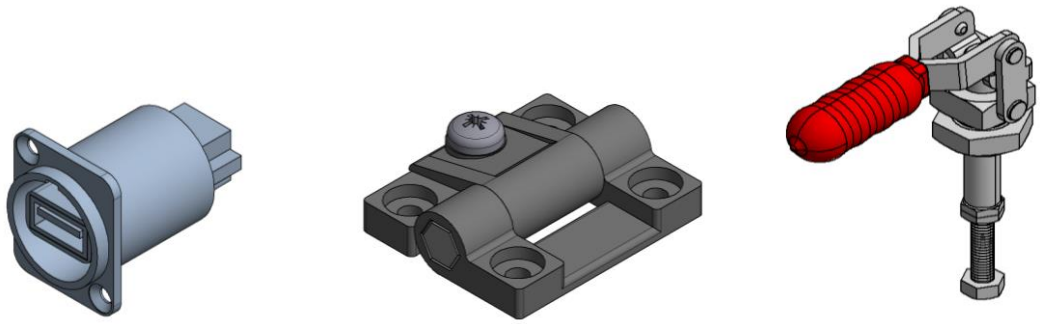
(d)

Figura 37 – Soporte brida.
Soporte brida (a). Soporte lateral (b). Soporte trasero (c). Soporte frontal (d).

10.1.9 Componentes

Se utilizan una serie de componentes prefabricados, los cuales son necesarios para el ensamblaje y posterior uso del verificador.

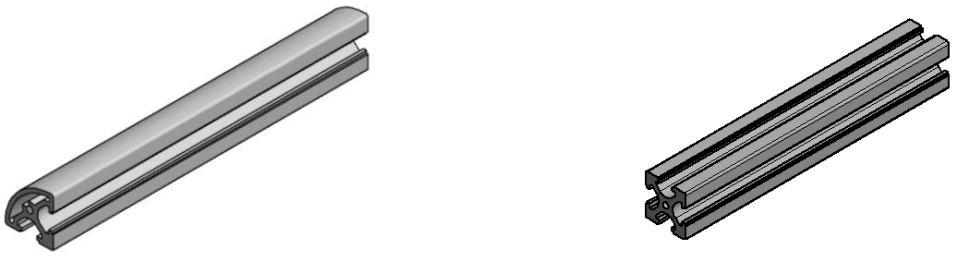
- Adaptadores USB: adaptadores tipo A-B. En el interior del verificador se coloca el tipo A y en el exterior el tipo B.
- Bisagras: necesarias para levantar la tapa y acceder al interior de la caja.
- Brida manual: utilizada para empujar la tarjeta contra los pinchos.



(a) (b) (c)

Figura 38 – Adaptador USB (a). Bisagra (b). Brida manual (c).

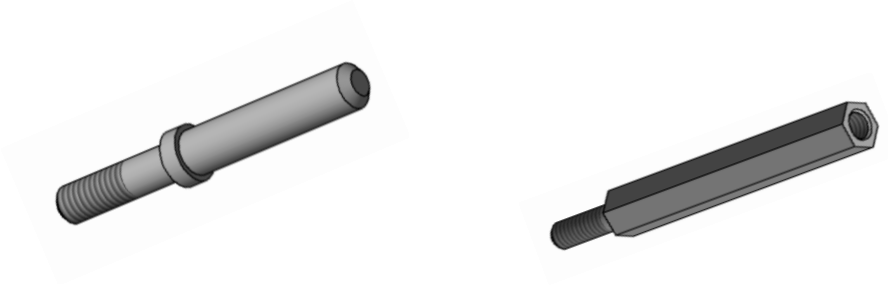
- Patas de goma: se utilizan para apoyar el verificador.
- Muelles: devuelven la tarjeta a su posición después de levantar la brida.
- Perfiles de canto: son perfiles con dos caras planas y el resto redondeado. Se usan para unir las partes exteriores del verificador (los redondeos se colocan hacia fuera).



(a) (b)

Figura 39 – Perfiles.
Perfiles de canto (a). Perfiles cuadrados (b).

- Perfiles cuadrados: unen las partes que sirven como soporte para la brida. Además, por ellos se deslizan las partes móviles.
- Posicionadores: sirven como guía para el correcto posicionamiento entre piezas móviles del verificador.



(a) (b)

Figura 40 – Posicionador (a). Separador (b).

- Separadores: son perfiles hexagonales que se sujetan al empujador. Estos son los que entran en contacto con la tarjeta. Se colocan de tal manera que pisen en sitios libres de la DUT:
- Tornillería: se utilizan tornillos, arandelas y tuercas de métrica 3 y distintos tamaños para unir las piezas del verificador.

10.2 Proceso de instalación de la cama de pinchos

Primero que todo se realiza una descripción sobre los pinchos utilizados en las camas.

Cuando se habla de pinchos de test, test probes o spring contact probes, se habla de agujas conductoras cuya misión, por lo general, es hacer contacto en partes metálicas de la PCB a testear [18].

Figura 41 – pincho / test probe. [15]

Existen diferentes tipos de “cabezas” en función de donde hará contacto esta: En un pad (de punta), en la pata de un componente de inserción (de corona), en un taladro metalizado (piramidal), etc.

Figura 42 – tipos de pinchos.
https://www.peaktest.co.uk/

Primeramente, se instala en la cama una camisa exterior o receptáculo, la cual será la parte fija y donde se conectará el cable que irá a la electrónica. Estos pueden tener una ranura para soldar el terminal o ser cilíndricos para ‘crimpar’ el cable en él (Enrollar). Los receptáculos se fijan a la cama por apriete, utilizando una herramienta especial.

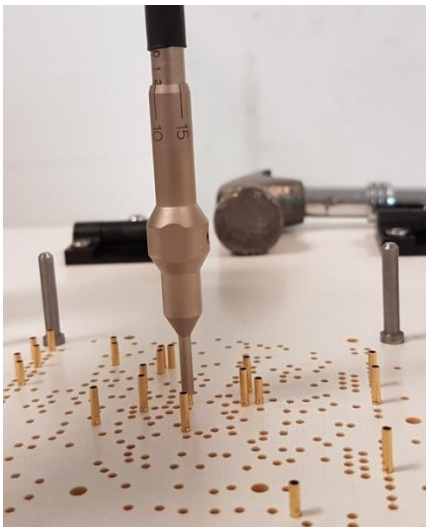


Figura 43 – Montaje de camisas para pinchos.

A continuación, se inserta el pincho en la camisa, este se encaja a presión.

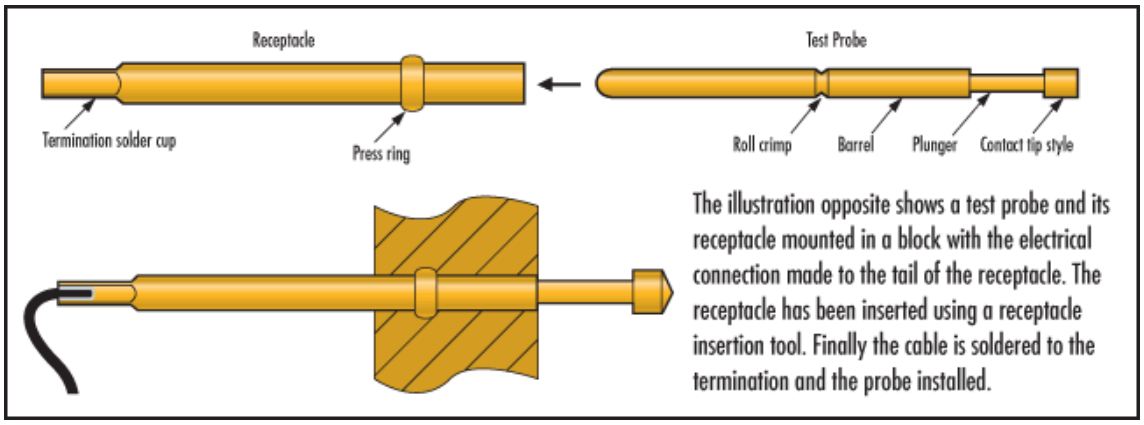


Figura 44 – Montaje del pincho.
http://www.coda-systems.co.uk/catalog/What_are_spring_contact_test_probes.html

10.3 Modificaciones posteriores

10.3.1 Sensor de sonido

Se opta por utilizar un sensor de sonido para detectar automáticamente la activación del zumbador. Por comodidad y estética se decidió colocarlo a un lateral del verificador (cuadro rojo).

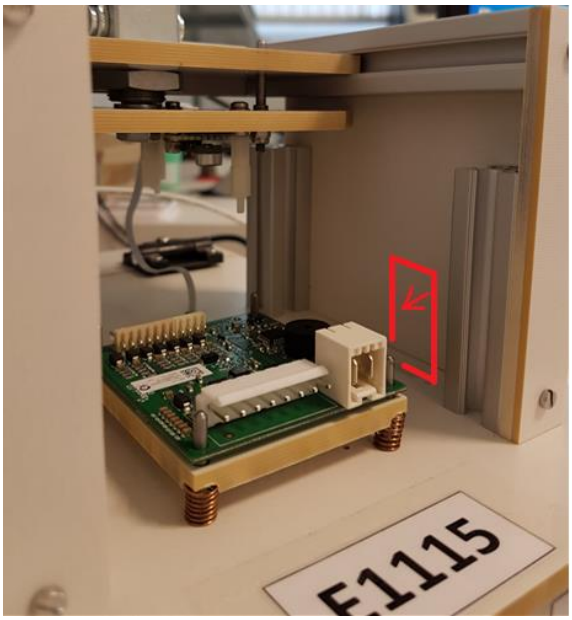


Figura 45 - Posicionamiento anterior del micrófono.

Sin embargo, el comportamiento de este no era el que se esperaba. Como se observa en la siguiente gráfica, los valores leídos en la entrada analógica de Arduino no eran muy constantes.

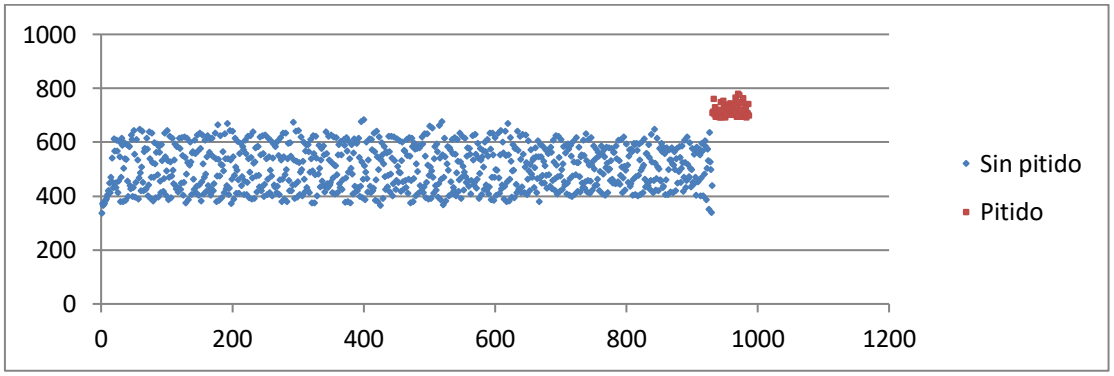


Figura 46 – Medidas micrófono posicionamiento anterior.

Pero se puede apreciar la diferencia de medidas entre la detección y la no detección. En base a esa diferencia se elige un valor de umbral de 690bits. (para el funcionamiento del sensor, ver apartado 9.2.3).

El problema radica en el número de veces que se obtenían lecturas por encima del umbral (contador) ya que era muy variable, por tanto, no se podían establecer correctamente los límites del contador.

Se realizaron medidas distintos días y en distintas zonas, y se recogieron los valores medios de la variable *contador*:

Umbral	690 bits

	Contador				
I+D	60	90	100	25	80
Producción	90	80	120	190	230

Tabla 7 – Pruebas micrófono posicionamiento anterior.

Como se ve, los datos son muy dispares, por lo que se hizo imposible elegir los valores límites. Además, en función de la inclinación que se le diera al micrófono, se detectaba más o menos.

Se decidió colocar el sensor en la parte superior del verificador, de tal manera que quedaba justo encima del zumbador.

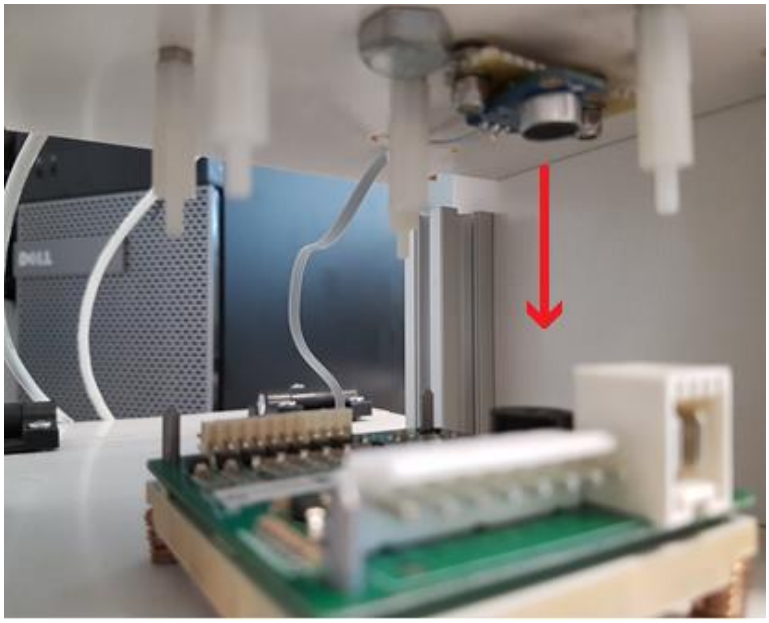


Figura 47 - Posicionamiento actual del micrófono.

Se volvieron a realizar las pruebas, obteniendo los siguientes resultados:

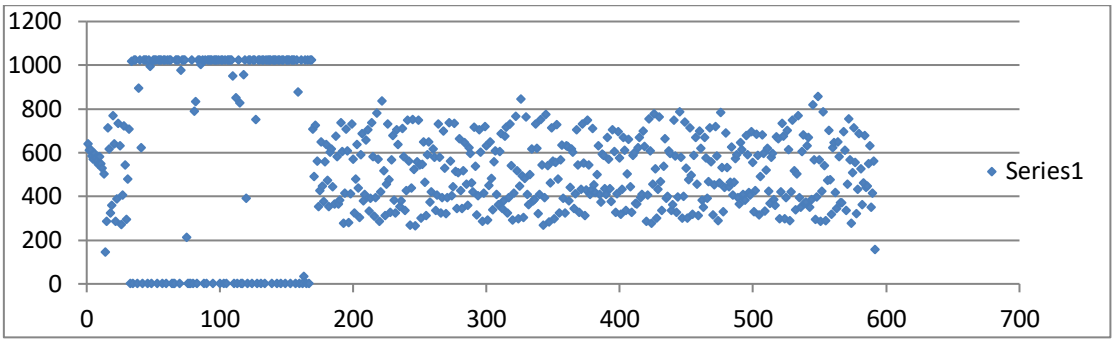


Figura 48 - Medidas micrófono posicionamiento actual.

Se distingue claramente la zona del pitido, oscila entre el máximo y el mínimo (señal PWM que genera el zumbador).

Como el sensor da un valor máximo (1024 bits) en cada detección, se opta por poner un umbral de 1000 bits.

Se repiten las pruebas en los distintos ordenadores:

umbral	1000 bits

	contador				
I+D	2494	2531	2543	2517	2525
Producción	2719	2689	2693	2657	2664

Tabla 8 - Pruebas micrófono posicionamiento actual.

El comportamiento es similar entre las medidas en I+D y en Producción, pero lo importante es que los valores en cada caso son prácticamente constantes.
Se establece la nueva configuración en el verificador.

La nueva configuración es menos estética debido a que el cable de conexión queda visible en mitad del verificador, pero se compensa con la automatización de un test que de no ser así podía dar muchos problemas, por ejemplo, si el operario se distrae y no escucha el pitido.

10.3.2 Ventilador

Se decide colocar un pequeño ventilador en el lateral izquierdo del verificador, con el fin de refrigerar el interior de este, debido al calentamiento de las resistencias.

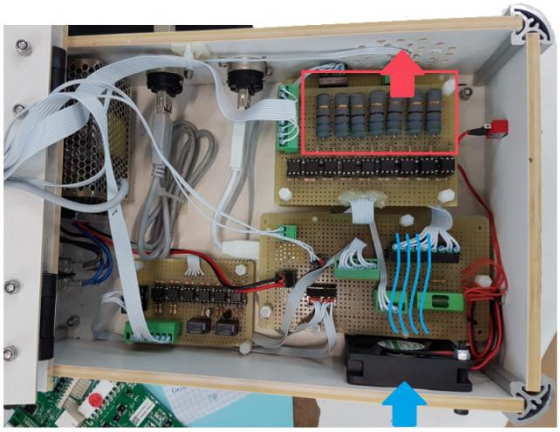


Figura 49 – Flujo de aire dentro del verificador.

No se previó este problema porque en realidad el tiempo en que se activarían las cargas era muy pequeño y no supondría un problema. Sin embargo, como se verá más adelante, una modificación del firmware de la tarjeta por parte del cliente, invirtió la señal de activación de las resistencias (con 0V estaban activas), calentando en exceso el interior del verificador.

Además de colocar el ventilador en el lateral izquierdo, también se mecanizan agujeros de ventilación en el lateral derecho.

Figura 50 – Vista lateral del verificador. Agujeros de ventilación.

11 Software

La función principal del software del verificador es la de comprobar el funcionamiento de la DUT según la secuencia de verificación. Sin embargo, también debe cumplir ciertos requisitos internos de la empresa.
Se comprobará que los directorios donde se albergan los ficheros necesarios para la verificación sean accesibles.
Se generarán los correspondientes ficheros de resultados en las rutas especificadas.

Por petición del cliente, se generan en la empresa dos referencias para la misma tarjeta: una para grabar y verificar las tarjetas, y otra solo para grabarlas. Se prevé también que en un futuro habrá una tercera orden que consistirá en solo verificar las tarjetas, sin grabar. El programa deberá ser capaz de ejecutarse en cualquiera de las tres versiones.

En caso de modificaciones de rutas (donde se guardan los ficheros), tiempos de ejecución (ordenadores más lentos) o alguna otra funcionalidad (detección de sonido), se deberá poder modificar el comportamiento del programa de manera rápida y sencilla.
El programa se adaptará a la forma de trabajo de los operarios (visualización y manejo).
Se reducirá, en medida de lo posible, el tiempo de verificación.

En base a esto, se diseña el programa para realizar la verificación funcional de la DUT mediante la tarjeta de Arduino [19]-[25]. Se dota al software de las siguientes funcionalidades:

- Interacción entre el programa y el operador, mediante el teclado y/o el ratón.

- Generación/Modificación de ficheros de resultados en las rutas especificadas.

- Ficheros .txt de configuración para cambiar funcionalidades.
- Testeo de las entradas y salidas de la DUT.
- Grabación manual o automática de la tarjeta.
- Detección de sonido de manera manual o automática.
- Detección de fallos de red (directorios incorrectos).
- Lectura de códigos de barras (mediante el útil correspondiente).
- Presentación de resultados del test en pantalla.
- Tiempo de grabación modificable en función del ordenador en el que se ejecute el programa.
- Funcionamiento en cualquier Windows.

11.1 Diagrama de flujo general del software

El programa se divide en dos partes: la parte de Arduino, la cual se encarga de testear la DUT, y la parte de Processing que gestiona las funcionalidades del software. La comunicación entre estos se realiza mediante el puerto serie.
En la figura 51 se muestra el diagrama de flujo del programa. En el lado izquierdo se describe el código creado para Arduino, y en el derecho generado para Processing. Las flechas centrales muestran las partes de envió/recepción de datos por el puerto serie.

Más adelante, en el apartado 11.3 se detallará el funcionamiento del código y se expandirá el flujograma.

Se puede ver el código completo, tanto de Arduino como de Processing en el apartado 3 de los anexos.

11.2 Ficheros de resultados

Como se indicó con anterioridad, los resultados del test se guardarán en ficheros .txt. los cuales se convertirán en ficheros .csv para ser consultados mediante Excel.

11.2.1 Nombre y rutas

Los nombres de los ficheros vienen dados por el número de orden leído en la pegatina de la tarjeta y por el resultado del test realizado.

- numerodeorden.txt Tarjetas que hayan superado el test.
- numerodeorden-FAIL.txt Tarjetas que no hayan superado el test

el número de orden corresponde a una serie de seis números identificativos del proceso de producción. Las rutas donde se guardan los ficheros dependen del número de referencia de la tarjeta. Se tienen dos referencias:

- 8CG07S0 Grabar el firmware A y verificar las tarjetas.
- 8CG07S0-INV Grabar el firmware B.

Así entonces, una posible ruta sería:
Y: \195\FICHEROS DE RESULTADOS\8CG07S0\757575.txt

Dentro de este fichero se irían guardando los resultados de todos los test realizados a las tarjetas con ese número de orden y referencia.

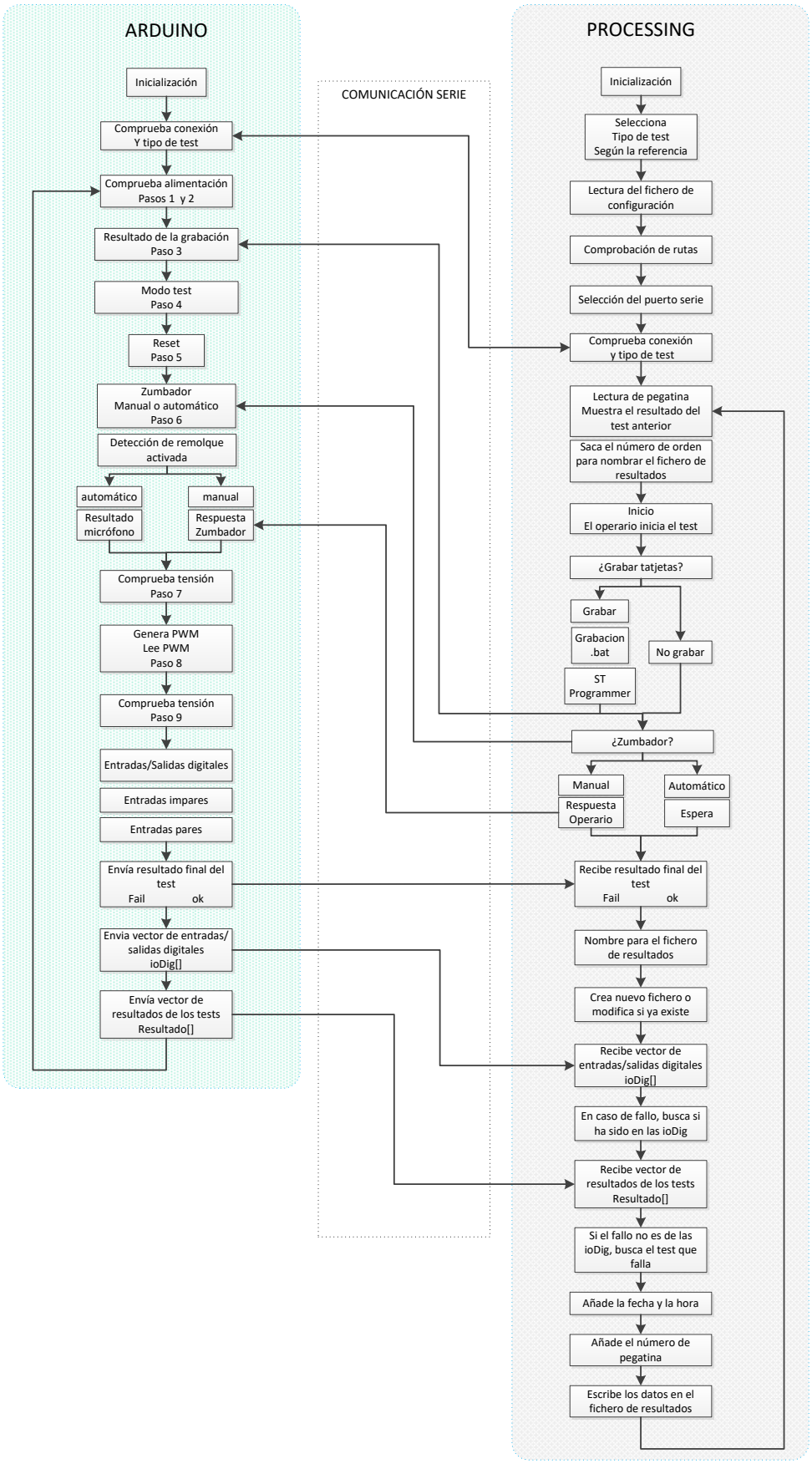


Figura 51 – Diagrama de flujo del software

11.2.2 Formato interno del fichero

Los resultados del test se guardarán con el siguiente formato:

Dos filas para el encabezado con la descripción de la columna y 18 columnas separadas por ';' con los valores del test. Utilizar ';' permite la separación por columnas al pasar el fichero de .txt a .csv para poder abrirlo con Excel y visualizarlo mejor.

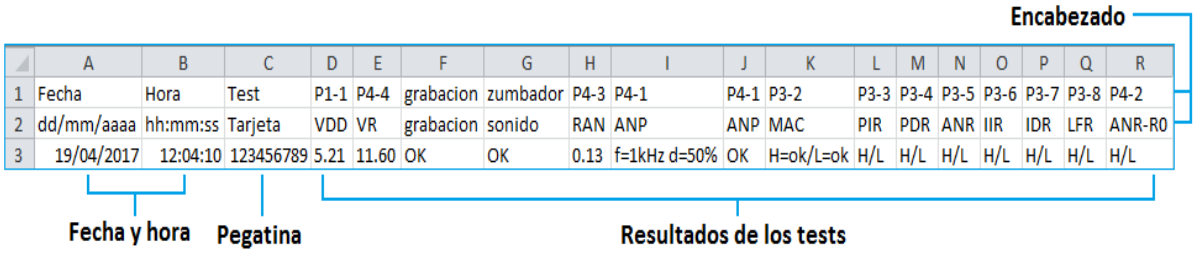


Figura 52 – Formato del fichero de resultados.

Los resultados de cada test son distintos en función de si se ha superado el test o no.

Test	Test superado	Test fallido
P1.1 VDD	valor de tensión	valor de tensión
P4.4 VR	valor de tensión	valor de tensión
grabación	OK	FAIL
Zumbador	OK	FAIL
P4.3 RAN	valor de tensión	valor de tensión
P4.1 ANP	f=1kHz d=50%	FAIL
P4.1 ANP	OK	FAIL
P3.2 (MAC)	H=ok/L=ok	H=fail/L=ok
P3.3 (PIR)		
P3.4 (PDR)		H=ok/L=fail
P3.5 (ANR)		
P3.6 (IIR)		H=fail /L=fail
P3.7 (IDR)		
P3.8 (LFR)		
P4.2 (ANR-R0)		

Tabla 9 – Resultados de los test.

Si la tarjeta no pasa el test, los resultados se guardarán en el fichero 'numerodeorden-FAIL.txt.' En este caso, se mostrarán los resultados hasta el punto de fallo, es decir, el programa detiene el test al encontrar un fallo. El resto de las columnas se rellenan con 'x'.

Los datos del test se guardarán en el correspondiente fichero una vez terminado el test para poder diferenciar entre tarjetas buenas y malas.

Después de leer la pegatina de la tarjeta, el programa comprobará la existencia del fichero con el mismo número de orden. Si el fichero ya ha sido creado, se empezarán a guardar los datos desde el final del fichero con el fin de evitar sobrescribirlo. En caso contrario, se generará un nuevo fichero.

En el caso de la referencia 8CG07S0-INV en la que solo se grabarán las tarjetas, se realizará el test hasta la grabación, el resto de columnas se rellenarán con 'x'.

Si se utiliza la tercera opción, es decir, solo verificar las tarjetas (sin grabar), se realizarán todos los test, y la columna de grabación se rellenará con "no grabar".

11.2.3 Ficheros de configuración

Se generan dos ficheros de configuración para modificar de manera sencilla una serie de parámetros previo al inicio del programa, tales como, rutas o tiempos. Los ficheros creados son: config.txt para la referencia 8CG07S0 y config_INV para la referencia 8CG07S0-INV. Se puede ver el contenido de estos en la tabla 10.

Dentro del propio fichero se hace una descripción de cada línea (Figura 53), como modificarlas y valores aconsejables. Es de gran importancia modificar de manera correcta el fichero, prestando especial atención en no cambiar líneas de posición, ya que, si no, el programa no funcionará.

Línea	Contenido
1	Directorio donde se guardan los ficheros de las tarjetas malas
2	Directorio donde se guardan los ficheros de las tarjetas buenas
3	Ruta del archivo .bat con las instrucciones para grabar las tarjetas
4	Ruta del fichero de resultado de la grabación generado por el ST Visual Programmer
5	Opción para grabar o no las tarjetas
6	Opción para detectar el zumbador automática o manualmente
7	Cada cuanto tiempo se comprobará si ha finalizado la grabación
8	Tiempo máximo de espera para la grabación

Tabla 10 – Líneas del fichero de configuración.

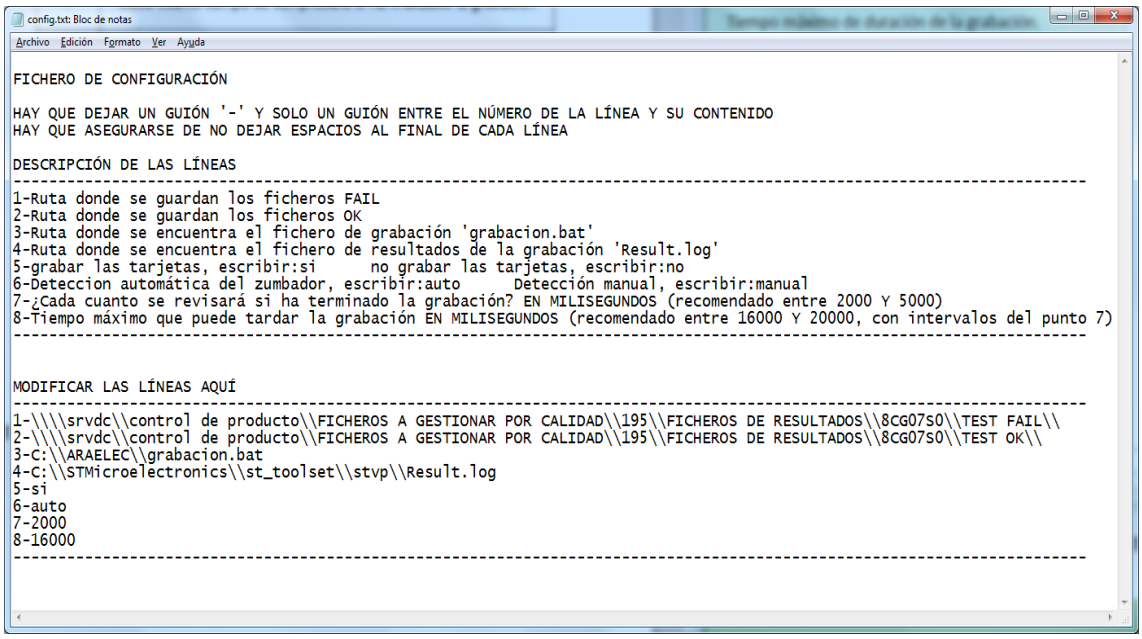


Figura 53 – Fichero de configuración.

11.3 Funcionamiento del programa de verificación

Se describe paso a paso el funcionamiento del código del programa desde el momento de su ejecución.

11.3.1 Selección del tipo de test

Se muestra una ventana con las tres opciones para realizar el test, cada una corresponde a las distintas referencias.

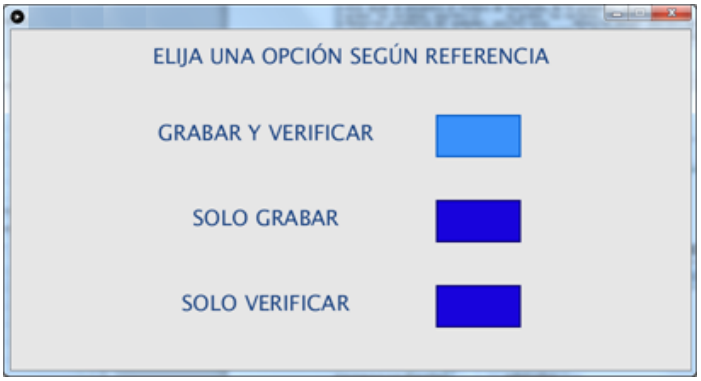


Figura 54 – Pantalla de selección del tipo de test.

En función de la selección se da valor a dos variables, por medio de las cuales se elegirá el fichero de configuración a usar para el test.

Opción	Referencia	Ficheros de configuración	Variables	
			grabar	soloGrabar
Grabar y verificar	8CG07S0	config.txt	true	false
Solo grabar	8CG07S0-INV	config_INV.txt	true	true
Solo verificar	Sin especificar.	config.txt	false	false

Tabla 11 – Variables de control para el tipo de test.

11.3.2 Lectura del fichero de configuración

Mediante la función 'leerConfig' se accede al fichero correspondiente (punto anterior) y se cargan las líneas. Se toman las líneas que interesan (de la 21 a la 28) y se guardan en la variable 'linesConfig'.
Las líneas se cargan como tipo String (cadena de caracteres), por lo que aquellas líneas numéricas se convierten a variables enteras.

Con el tiempo máximo de grabación (Línea 8) y el tiempo en el que se comprobará si ha finalizado o no (Línea 7), se calcula las veces que se realizará esa comprobación.
Se comprueba cada X segundos, durante un máximo de Y segundos, por tanto, se realizarán como mucho Y/X comprobaciones.

11.3.3 Comprobación de directorios

Se comprueba que los directorios escritos en el fichero de configuración son accesibles. Se leen las líneas de la 1 a la 4, y se comprueba si existen. En caso de fallo de red el programa mostrará una ventana de error, indicando la(s) línea(s) que falla(n). El programa se cierra automáticamente en 10 segundos.

Figura 55 – Pantalla de fallo de red.

Si no hay ningún fallo, se pasa a la selección de puertos.

11.3.4 Selección del puerto serie

Processing guarda el número de puertos conectados en ese momento y muestra un mensaje para que el operario conecte el cable USB correspondiente al Arduino.

Figura 56 – Pantalla de selección de puerto.

Una vez conectado, se detecta que hay un puerto más (corresponde a Arduino) y se inicia la comunicación por el puerto serie.

Para comprobar la conexión entre Processing y Arduino se envía desde Arduino el carácter 'k' durante 1,5 segundos.

Figura 57 – Pantalla de verificación de la conexión.

Si pasados 1500ms, Processing no ha detectado el carácter, quiere decir que no está comunicando con Arduino; esto podría ser porque el programa cargado en Arduino no es el adecuado (esto no debería suceder). Este tiempo ha sido elegido según las pruebas realizadas en los ordenadores de producción.

Si todo es correcto, Processing envía el carácter 'v' a Arduino si el tipo de test es el de solo grabar, en caso contrario, envía el carácter 'c'.
Arduino recibe el carácter que le indica cómo va a ser el test y se mantiene a la espera del carácter 'a' para iniciar los test.

Todas las configuraciones realizadas hasta este punto (tipo de test, directorios, puertos) se realizan solo una vez, al ejecutar el programa. A partir de aquí se empieza a trabajar con cada DUT por individual. No es necesario volverá ejecutar el programa para verificar más tarjetas.

11.3.5 Lectura de la pegatina

Se muestra la pantalla para leer la pegatina de la tarjeta, indicando que es el primer test que se realiza, ya que posteriormente se mostrará el resultado del test anterior.

El lector de códigos de barra hace las veces de teclado. Processing detecta cada vez que se pulsa una tecla y guarda el valor leído (el carácter final es INTRO).
El lector debe estar configurado para introducir 'intro' al final de la lectura, si no, se deberá pulsar la tecla manualmente.
Es importante que la ventana del programa esté seleccionada en este momento ya que si no lo está no se detectarán pulsaciones del teclado. El programa muestra una pantalla con el mensaje para seleccionar la ventana en caso que no esta no esté activa.

Figura 58 – Pantalla de lectura de la pegatina.

Una vez leída la pegatina, el programa extrae el número de orden, con el que se nombrarán los ficheros de resultados, este corresponde con los primeros 6 dígitos de la pegatina.

11.3.6 Inicio del test - Tensiones de alimentación

Se muestra la pantalla para dar inicio al test. Esta muestra el número leído en la pegatina. En este punto el operario introducirá la tarjeta en el verificador y conectará la tensión.

Para iniciar, se puede dar clic sobre el botón 'START' o presionar 'intro'. Processing envía el carácter 'a' a Arduino. Arduino inicia el test.

Figura 59 – Pantalla de inicio del test.

Paso 1

Lectura analógica de la entrada P11. Se usa la función "LecturaAnalogica" (lecAnalog). Se recibe, el número de la entrada, el límite inferior, el límite superior, la escala, los valores para el escalado (*m* y *b* de la recta obtenida con el Excel. Apartado 9.2.2.4) y el índice del vector de resultados.

Se lee el valor de la entrada 10 veces y se realiza la media.

Arduino convierte la tensión de la entrada en bits, con una resolución de:

$$\frac{5V}{2^{10}bits} = 0.0048828\frac{V}{bits} \rightarrow \text{resArd}$$

Si el valor leído es de:

bitsIn $bits$

La tensión de salida del operacional es:

$$vOp\ [V] = \ (\text{bitsIn}\ bits)\left(\text{resArd}\ \frac{V}{bits}\right)$$

Utilizando la relación entre la tensión de entrada y de salida de los operacionales se obtiene el valor real medido vIn:

$$\text{vOp}[V] = (\text{m})\ (\text{vIn}\ ) + \text{b}$$

$$\text{vIn}\ [V] = \frac{vOp\ - b}{m}$$

Si el valor obtenido *vIn* está dentro de los límites, el test es correcto, se guarda el valor en el vector de resultados (resultado[índice]).
Si el valor está fuera de los límites, se envía "fail*" a Processing (resultado del test), luego se envía el vector de las I/O digitales (ioDig) y por último se envía el vector de resultados (resultado[]). Finaliza el programa sin realizar los otros test.

En este paso, si el verificador no está encendido (el operario olvida conectar la tensión), el valor leído será negativo (debido a la configuración de los operacionales). En este caso, el resultado será 'x' y el test fallará y se indicará en pantalla un mensaje de fallo de alimentación.

Los valores de las rectas (m y b) han sido modificados para adaptarlos a los valores medidos una vez montado el verificador.

Paso 2 Lectura analógica

Se procede de la misma manera que en el paso 1, pero con la entrada P44.

11.3.7 Grabación de tarjeta

Paso 3

Arduino envía “vOK*” a Processing para indicar que las tensiones de alimentación son correctas y espera a que Processing le envíe una respuesta sobre la grabación (‘s’, ‘n’ o ‘d’)

Processing recibe “vOk*” y mira si tiene que grabar o no las tarjetas.

La grabación de la tarjeta se lleva a cabo mediante un archivo .bat con instrucciones para ejecutar el software grabador del ST-link. Este software es el “ST Visual Programmer”, el cual tiene una extensión llamada “STVP_CmdLine” para poder ser ejecutado desde líneas de comando en la consola de MS-DOS.
Al finalizar la grabación, el STVP genera automáticamente un fichero de resultados “Result.log”. En este fichero se recoge la evolución el proceso de grabación y sus resultados, por lo que es de gran importancia, ya que, accediendo a él es como se comprueba si la grabación ha sido correcta o no.

A petición del cliente, no solo deberá grabarse el firmware suministrado, sino que también se grabará la opción para proteger la tarjeta ante lectura/escritura (optionbyte)._Por esto, aparte de tener el fichero del firmware, se crean dos ficheros .hex, uno para activar la protección después de grabar y otro para desactivarla antes de grabar.

Processing lee la línea 4 de la configuración, que corresponde a la ruta donde se encuentra el fichero de resultados de la grabación “Result.log”. Se borra el contenido del fichero antes de cada grabación para guardar solo los resultados de la grabación actual (en caso contrario se irían acumulando los resultados de distintas tarjetas).

Se lee línea 5 del fichero de configuración, que corresponde a la ruta donde se encuentra el fichero .bat (fichero con las instrucciones para la grabación) y se ejecuta.

11.3.7.1 Fichero grabacion.bat

Contiene las instrucciones necesarias para ejecutar el software del grabador y grabar el firmware de la DUT.
Cabe mencionar que se crean dos ficheros de grabación para las dos referencias de tarjetas, ya que le firmware en cada caso es distinto.

Se muestra a continuación las líneas del código del fichero (color azul) y se explica su funcionamiento (color negro).

Nombre del firmware (.hex).

```
set nombre_fw=firmware.hex
```

Ruta donde se encuentra el firmware.

```
set ruta_fw=\\srvdc\Software\8CG07S0\
```

Nombre archivo, protección activada (.hex).

```
set nombre_obon=optionbyte_ON.hex
```

Nombre archivo, protección desactivada (.hex).

```
set nombre_oboff=optionbyte_OFF.hex
```

ruta donde se encuentran los archivos de protección.

```
set ruta_ob=\\srvdc\Software\8CG07S0\
```

Ruta del directorio local donde se copiarán los archivos .hex (para que el programador pueda acceder a los archivos .hex, estos deben estar en un directorio local).

```
set ruta_local=C:\LOCAL\
```

Copia los archivos .hex al directorio local

```
xcopy %ruta_fw%%nombre_fw% %ruta_local% /y
xcopy %ruta_ob%%nombre_obon% %ruta_local% /y
xcopy %ruta_ob%%nombre_oboff% %ruta_local% /y
```

Ingresa en la carpeta del programa para ejecutarlo

cd\

cd C:\STMicroelectronics\st_toolset\stvp

Ejecución del programa. Desactivación de la protección.

STVP_CmdLine

Nombre del grabador.

-BoardName=ST-LINK

Tipo de comunicación.

-Port=USB

Modo de programación (Single Wire Interface Module. Cable único.)

-ProgMode=SWIM

Microprocesador .

-Device=STM8S003K3

Habilita la creación del fichero de resultados.

-log

Deshabilita la grabación en cadena (solo se ejecuta una vez).

-no_loop

Guarda el progreso de la grabación.

-progress

Verifica al terminar.

-verif

Archivo a grabar: optionbyte_OFF.hex (desactiva la protección).

-FileOption=%ruta_local%%nombre_oboff%

Ejecución del programa. Grabación del firmware y protección.

Programa ejecutable.

STVP_CmdLine

-BoardName=ST-LINK

-Port=USB -ProgMode=SWIM

-Device=STM8S003K3

-log

-no_loop

-progress

Deshabilita el mensaje de advertencia al grabar la protección.

-no_warn_protect

-verif

Archivo a grabar: firmware.hex (firmware del cliente).

-FileProg=%ruta_local%%nombre_fw%

Archivo a grabar: optionbyte_ON.hex (activa la protección).

-FileOption=%ruta_local%%nombre_obon%

Se muestra la pantalla de espera mientras se lleva a cabo la grabación.

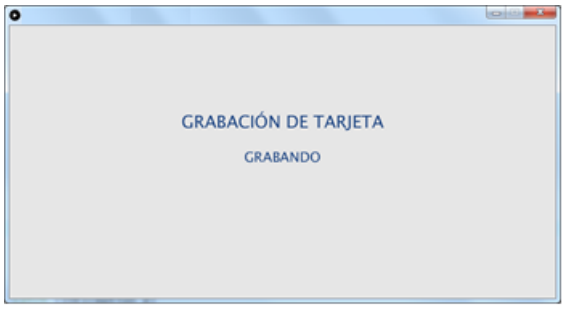


Figura 60 – Pantalla de grabación.

Pasados **X** segundos (línea 7 del fichero de configuración), se lee el fichero de resultados de la grabación 'Result.log'.

Se comprueban dos líneas específicas que indican el final exitoso de la grabación. Si estas líneas no coinciden con lo que se espera ("<<< Verifying PROGRAM MEMORY succeeds" , "<<< Verify OPTION BYTE succeeds"), se aumenta un contador (lecturasGrab) y se repite el proceso X segundos después.

En el momento en el que las líneas coincidan, habrá finalizado correctamente la grabación y continuaría con el siguiente paso.

Si pasados **Y** segundos (línea 8 del fichero de configuración) se comprueba que el resultado de la grabación sigue sin ser correcto, se da la grabación por fallida. Realizando la comprobación de esta manera, se consigue finalizar la grabación en el momento justo, en lugar de esperar un tiempo fijo.

Es importante que **Y** sea lo suficientemente grande como para permitir la grabación en un ordenador menos potente. Realizando pruebas en distintos ordenadores, se comprobó que en el peor de los casos la grabación tardaba 12 segundos. Se recomienda que este valor esté entre 15 y 20 segundos, y X varíe entre 2 y 5 segundos (Apartado 12.6).

En función del resultado de la grabación, Processing envía un carácter u otro a Arduino:

- Si no hay que grabar las tarjetas, envía ‘d’.
- Si la tarjeta se ha grabado correctamente, envía ‘s’.
- Si ha habido un fallo en la grabación, envía ‘n’.

Arduino recibe el resultado de la programación. Si es ‘n’ envía “fail*” a Processing, envía los resultados y finaliza el programa sin realizar los otros test.
Processing vuelve a la pantalla de leer pegatina pero con el correspondiente mensaje de error.

Figura 61 – Pantalla de fallo en la grabación.

Si Arduino recibe ‘s’ guarda “OK” en la casilla correspondiente del vector de resultados (resultados[3]=”OK”). Si Arduino recibe ‘d’ guarda “No grabar” Continua con el paso 4.

Processing muestra la pantalla de espera indicando que se está realizando el test.

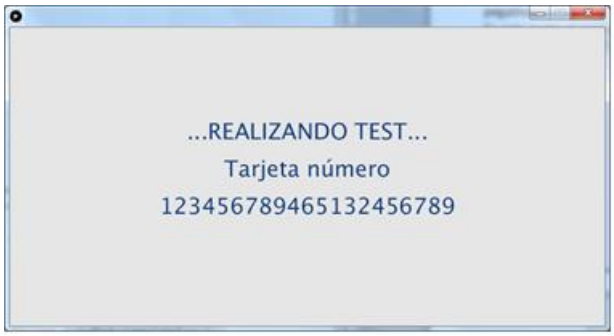


Figura 62 – Pantalla de espera.

Paso 4

P31 (DR) a nivel bajo.

modoTest a nivel alto para habilitar la realización del test.

Paso 5

Se realiza el reset a la tarjeta.

11.3.8 Comprobación del zumbador

Paso 6

P31 (DR) a nivel alto. Se comprueba que se activa el zumbador de la tarjeta (emite un pitido).

Arduino envía “zumbador*” a Processing y se queda esperando a recibir respuesta sobre si la comprobación del zumbador se hará de manera automática (recibe ‘z’) o manual (recibe ’x’).

Processing lee la línea 6 del fichero de configuración. Si es “auto” envía ‘z’ a Arduino (comprobación automática), si no, envía ‘x’ (comprobación manual).

11.3.8.1 Comprobación automática

Después de activar la entrada P3.1 (es cuando emite el pitido), se llama a la función “Zumbador”. Se lee la entrada del detector de audio durante un segundo.

Cuando detecta el pitido, la lectura de Arduino pasa de aproximadamente 400bits (2V) a más de 1000bits (5V). Cada vez que ocurre esto se incrementa un contador.

Realizando pruebas tanto en los ordenadores de producción como en I+D, se vio que pasaba de 1000bits entre 2400 y 2800 veces cada vez que pitaba (ver apartado 10.3.1).

Si pasado ese tiempo, el contador está entre 2000 y 3000, el test se da por bueno y continua con el paso 7. Si no, envía los resultados a Processing y finaliza el programa. Processing muestra el mensaje de error correspondiente.

11.3.8.2 Comprobación manual

Activa la entrada P31 y espera a que Processing le envíe respuesta sobre la activación el zumbador.

Processing muestra una pantalla para que el operario indique si ha escuchado el pitido o no.

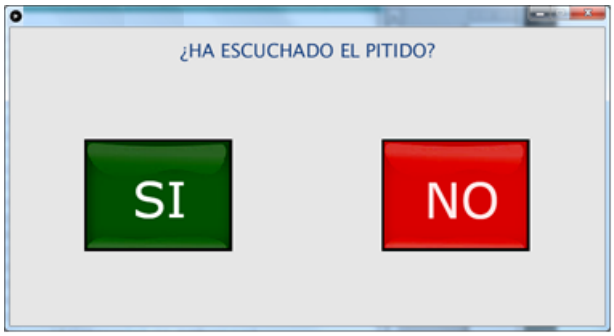


Figura 63 – Pantalla de selección Sí/No.

La elección se puede hacer mediante el ratón y el teclado (‘Tab’, flechas de navegación e ‘intro’).

Si la respuesta es 'SI', Processing envía 's' a Arduino para que este continúe con el paso 7. Se vuelve a mostrar la pantalla8 (pantalla de espera).
Si la respuesta es 'NO', Processing envía 'n' a Arduino. Arduino envía los resultados a Processing y finaliza el test. Processing muestra el mensaje de error correspondiente.

Paso 7

Igual que el paso 1 pero con la entrada P43 (lectura analógica).

11.3.9 Comprobación PWM

Paso8

En Arduino, se configuran los relés para conectar VP5 a las entradas P5.2 P5.3 y P5.4. Se genera el PWM (100Hz y duty=50%) en la salida VP5, mediante el uso de la librería TimerThree.

Se activa la interrupción de la entrada P4.1, donde se debe leer un PWM igual que el generado. Cada vez que la entrada cambie de nivel, se ejecutará la interrupción.

Dentro de la función de interrupción, se mide el tiempo que está en alto/bajo, este tiempo debe ser de 5±0,8ms (corresponde a un semiperiodo de una señal de 100Hz). Se realizan un total de 20 mediciones (corresponde aproximadamente a 1 segundo, tiempo indicado por el fabricante en la secuencia de verificación).
Si hay más de 3 mediciones fuera de límites, se considera que el PWM leído no es correcto. se envían los resultados a Processing y finaliza el programa. En caso contrario, continúa con el siguiente paso.

Paso 9

Con la misma configuración de los relés, se desactiva la salida VP5, lo que escribe un nivel bajo en las salidas P5.2, P5.3 y P5.4.

P4.1 debe quedar pasar a nivel bajo. Si no lo hace, envía los resultados a Processing y finaliza el programa. En caso contrario, continúa con el siguiente paso.

11.3.10 Activación de entradas/salidas digitales

Paso 10

Se realiza en dos partes:

Entradas impares activadas (P5.1, P5.3, P5.5, P5.7).

Las salidas pares (P3.2, P3.4, P3.6, P3.8) deben quedar a nivel alto, mientras las impares deben quedar a nivel bajo.

Entradas pares activadas (P5.2, P5.4, P5.6).

Las salidas impares (P33, P35, P37) deben quedar a nivel alto, mientras las pares deben quedar a nivel bajo.

En ambos casos, se comprueba el estado de la salida P42, la cual está conectada a la P35.

Se utiliza la función “lecturaDigital”. Se configuran los relés para conectar las entradas P5.1, P5.3, P5.5 y P5.7 de la tarjeta, a la salida VP5 de Arduino (releB=0, releA=1) y se aplica nivel alto a VP5.

Se leen las entradas del puerto 3. Si la medición es correcta, guarda “H=ok” (impares) o “L=ok” (pares), en la posición correspondiente del vector de resultados. Además, guarda “1” en la posición correspondiente en el vector de las I/O digitales (ioDig). Si la medición es incorrecta, “H=fail” o “L=fail” en el vector de resultados y “0” en el vector de I/O digitales.

Configura los relés para conectar P5.2, P5.4 y P5.6 a la salida VP5 de Arduino. Aplica nivel alto a VP5.

Se leen las entradas del puerto 3. Si la medición es correcta, añade “H=ok” (pares) o “L=ok” (impares) al resultado del paso anterior. Si la medición es incorrecta, añade “H=fail” o “L=fail”.

Si cualquiera de las entradas, en cualquiera de los pasos da una lectura errónea, la tarjeta no pasa el test. Envía “fail*” al Processing, envía el vector de resultados y vuelve al inicio del programa. En caso contrario, envía “ok*” al Processing, envía el vector de resultados y vuelve al inicio del programa.

11.3.11 Resultados del test

Processing recibe primero el resultado del test.
Si es “ok*” el nombre asignado al fichero será “numeroDeOrden.txt”
Si es “fail*” el nombre asignado al fichero será “numeroDeOrden-FAIL.txt”

Recibe el resultado de las I/O digitales y revisa si ha habido fallo en este paso (paso 10). Comprueba entrada por entrada y guarda las que hayan fallado para mostrarlas en pantalla.

Figura 64 - Pantalla de fallo.

Si el valor de la primera I/O es igual a ‘x‘, quiere decir que el fallo se produjo en un paso anterior (no han llegado a testearse las I/O digitales).

Processing recibe el vector de resultados. Busca en qué paso se detuvo el test (columna con valor ‘x‘) y elige el mensaje de fallo a mostrar (tabla 12).

Carga la fecha y la hora de realización del test, los adjunta a los resultados y los escribe en el fichero correspondiente.

Fallo	Descripción
Fallo 1	No hay alimentación
Fallo 2	Hay X V en P1.1 en lugar de 5 +-0.5 V
Fallo 3	Hay X V en P4.4 en lugar de11.4 +-0.5 V
Fallo 4	Fallo en la grabación
Fallo 5	No se activó el zumbador
Fallo 6	Hay X V en P4.3 en lugar de <0.4V
Fallo 7	PWM incorrecto en P4.1
Fallo 8	No se desactivó P4.1

Tabla 12 – Códigos de fallo.

11.3.12 Creación/Modificación de ficheros

Se crea un objeto o clase, para realizar la gestión de ficheros en Processing.

Se comprueba si ya existe un fichero con el nombre dado. Para esto, se abre el directorio donde se va a guardar el archivo y se carga una lista que con los nombres de los ficheros que se encuentran en esa carpeta.
Se comparan uno a uno los nombres de los ficheros con el nombre del que se quiere crear.
Si no hay coincidencia, se crea un nuevo fichero, se escribe la cabecera (Apartado 11.2.2) y se escriben los resultados.
Si hay coincidencia, se lee el fichero existente, se copia su contenido y se reescribe el fichero con lo que ya tenía, más los nuevos resultados.

Una vez finalizado el test, Processing vuelve a la pantalla de lectura de pegatina, mostrando el resultado del test que se acaba de realizar.

11.3.12.1 Tiempo de lectura/escritura del fichero

Se realiza una prueba para comprobar el tiempo que el programa tarda en modificar el fichero de resultados, teniendo en cuenta este llegará a tener más de 10000 líneas. Es importante que el programa no tenga retrasos.

Además, los ficheros de texto deberán cambiar a ficheros *.csv* para su visualización en Excel. Se ha de comprobar que este cambio se realiza sin ningún problema.

Se realiza una verificación (mediante Arduino). Se genera el fichero .txt.

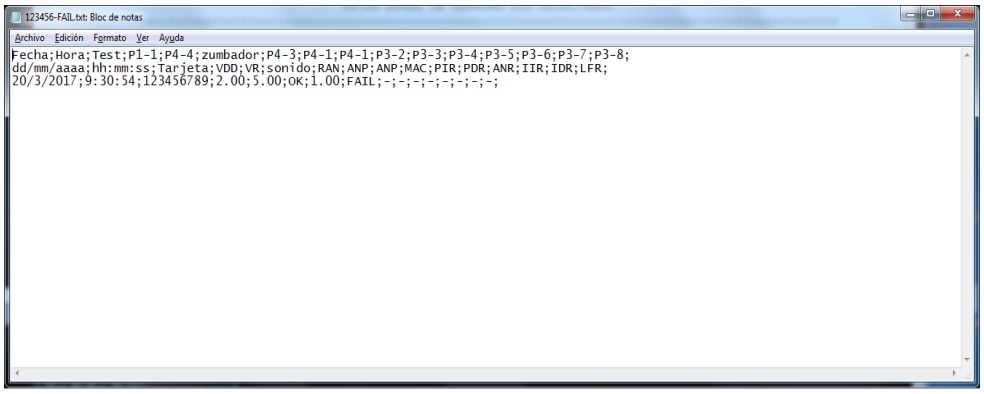


Figura 65 – Creación del .txt. Una verificación.

Se cambia la extensión del fichero a .csv.

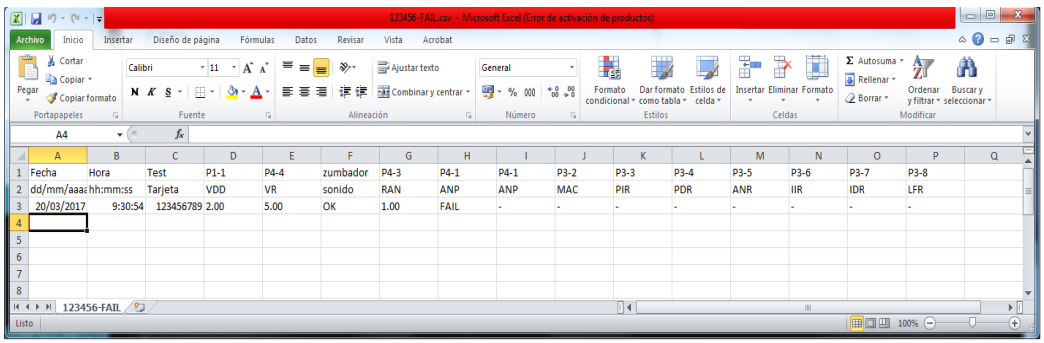
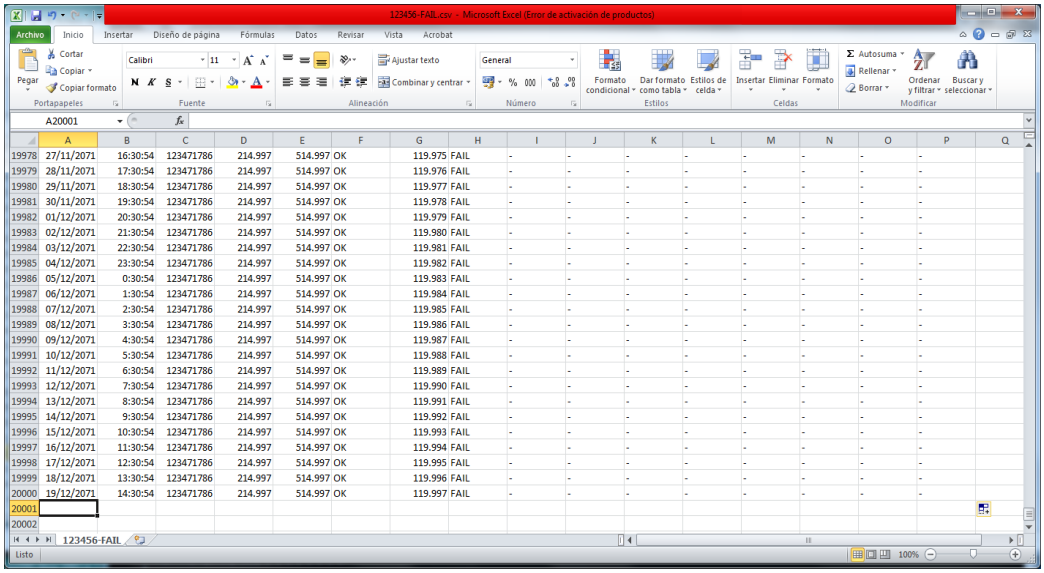


Figura 66 – Conversión a csv. Una verificación.

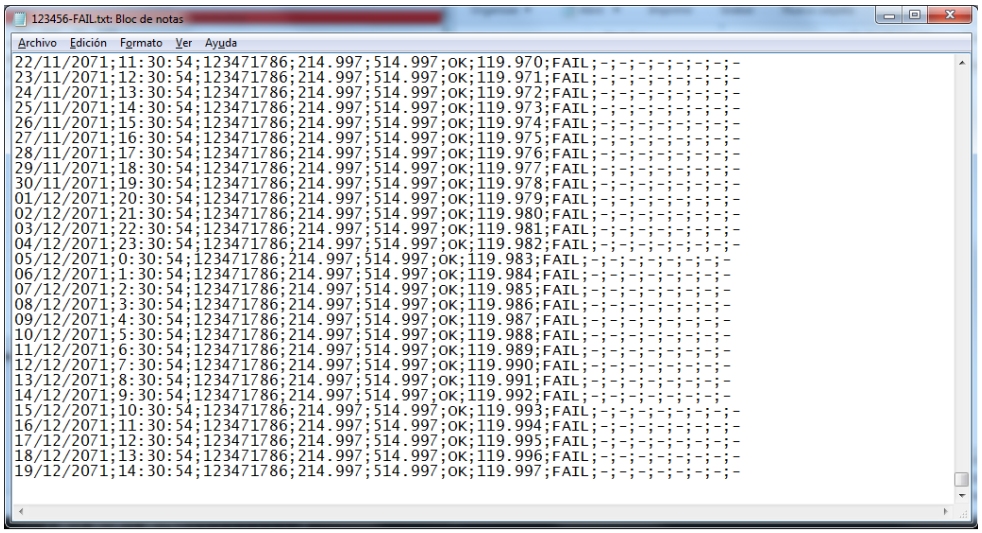
Se copia la última línea hasta llegar a la fila 20000. Esto simula un fichero con 20000 verificaciones realizadas.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
19978	27/11/2071	16:30:54	123471786	214.997	514.997	OK	119.975	FAIL	-	-	-	-	-	-	-	-	
19979	28/11/2071	17:30:54	123471786	214.997	514.997	OK	119.976	FAIL	-	-	-	-	-	-	-	-	
19980	29/11/2071	18:30:54	123471786	214.997	514.997	OK	119.977	FAIL	-	-	-	-	-	-	-	-	
19981	30/11/2071	19:30:54	123471786	214.997	514.997	OK	119.978	FAIL	-	-	-	-	-	-	-	-	
19982	01/12/2071	20:30:54	123471786	214.997	514.997	OK	119.979	FAIL	-	-	-	-	-	-	-	-	
19983	02/12/2071	21:30:54	123471786	214.997	514.997	OK	119.980	FAIL	-	-	-	-	-	-	-	-	
19984	03/12/2071	22:30:54	123471786	214.997	514.997	OK	119.981	FAIL	-	-	-	-	-	-	-	-	
19985	04/12/2071	23:30:54	123471786	214.997	514.997	OK	119.982	FAIL	-	-	-	-	-	-	-	-	
19986	05/12/2071	0:30:54	123471786	214.997	514.997	OK	119.983	FAIL	-	-	-	-	-	-	-	-	
19987	06/12/2071	1:30:54	123471786	214.997	514.997	OK	119.984	FAIL	-	-	-	-	-	-	-	-	
19988	07/12/2071	2:30:54	123471786	214.997	514.997	OK	119.985	FAIL	-	-	-	-	-	-	-	-	
19989	08/12/2071	3:30:54	123471786	214.997	514.997	OK	119.986	FAIL	-	-	-	-	-	-	-	-	
19990	09/12/2071	4:30:54	123471786	214.997	514.997	OK	119.987	FAIL	-	-	-	-	-	-	-	-	
19991	10/12/2071	5:30:54	123471786	214.997	514.997	OK	119.988	FAIL	-	-	-	-	-	-	-	-	
19992	11/12/2071	6:30:54	123471786	214.997	514.997	OK	119.989	FAIL	-	-	-	-	-	-	-	-	
19993	12/12/2071	7:30:54	123471786	214.997	514.997	OK	119.990	FAIL	-	-	-	-	-	-	-	-	
19994	13/12/2071	8:30:54	123471786	214.997	514.997	OK	119.991	FAIL	-	-	-	-	-	-	-	-	
19995	14/12/2071	9:30:54	123471786	214.997	514.997	OK	119.992	FAIL	-	-	-	-	-	-	-	-	
19996	15/12/2071	10:30:54	123471786	214.997	514.997	OK	119.993	FAIL	-	-	-	-	-	-	-	-	
19997	16/12/2071	11:30:54	123471786	214.997	514.997	OK	119.994	FAIL	-	-	-	-	-	-	-	-	
19998	17/12/2071	12:30:54	123471786	214.997	514.997	OK	119.995	FAIL	-	-	-	-	-	-	-	-	
19999	18/12/2071	13:30:54	123471786	214.997	514.997	OK	119.996	FAIL	-	-	-	-	-	-	-	-	
20000	19/12/2071	14:30:54	123471786	214.997	514.997	OK	119.997	FAIL	-	-	-	-	-	-	-	-	
20001																	
20002																	

Figura 67 – Fichero csv. 20000 líneas.

Se cambia la extensión a .txt



```
22/11/2071;11:30:54;123471786;214.997;514.997;OK;119.970;FAIL;-;-;-;-;-;-;-;-
23/11/2071;12:30:54;123471786;214.997;514.997;OK;119.971;FAIL;-;-;-;-;-;-;-;-
24/11/2071;13:30:54;123471786;214.997;514.997;OK;119.972;FAIL;-;-;-;-;-;-;-;-
25/11/2071;14:30:54;123471786;214.997;514.997;OK;119.973;FAIL;-;-;-;-;-;-;-;-
26/11/2071;15:30:54;123471786;214.997;514.997;OK;119.974;FAIL;-;-;-;-;-;-;-;-
27/11/2071;16:30:54;123471786;214.997;514.997;OK;119.975;FAIL;-;-;-;-;-;-;-;-
28/11/2071;17:30:54;123471786;214.997;514.997;OK;119.976;FAIL;-;-;-;-;-;-;-;-
29/11/2071;18:30:54;123471786;214.997;514.997;OK;119.977;FAIL;-;-;-;-;-;-;-;-
30/11/2071;19:30:54;123471786;214.997;514.997;OK;119.978;FAIL;-;-;-;-;-;-;-;-
01/12/2071;20:30:54;123471786;214.997;514.997;OK;119.979;FAIL;-;-;-;-;-;-;-;-
02/12/2071;21:30:54;123471786;214.997;514.997;OK;119.980;FAIL;-;-;-;-;-;-;-;-
03/12/2071;22:30:54;123471786;214.997;514.997;OK;119.981;FAIL;-;-;-;-;-;-;-;-
04/12/2071;23:30:54;123471786;214.997;514.997;OK;119.982;FAIL;-;-;-;-;-;-;-;-
05/12/2071;0:30:54;123471786;214.997;514.997;OK;119.983;FAIL;-;-;-;-;-;-;-;-
06/12/2071;1:30:54;123471786;214.997;514.997;OK;119.984;FAIL;-;-;-;-;-;-;-;-
07/12/2071;2:30:54;123471786;214.997;514.997;OK;119.985;FAIL;-;-;-;-;-;-;-;-
08/12/2071;3:30:54;123471786;214.997;514.997;OK;119.986;FAIL;-;-;-;-;-;-;-;-
09/12/2071;4:30:54;123471786;214.997;514.997;OK;119.987;FAIL;-;-;-;-;-;-;-;-
10/12/2071;5:30:54;123471786;214.997;514.997;OK;119.988;FAIL;-;-;-;-;-;-;-;-
11/12/2071;6:30:54;123471786;214.997;514.997;OK;119.989;FAIL;-;-;-;-;-;-;-;-
12/12/2071;7:30:54;123471786;214.997;514.997;OK;119.990;FAIL;-;-;-;-;-;-;-;-
13/12/2071;8:30:54;123471786;214.997;514.997;OK;119.991;FAIL;-;-;-;-;-;-;-;-
14/12/2071;9:30:54;123471786;214.997;514.997;OK;119.992;FAIL;-;-;-;-;-;-;-;-
15/12/2071;10:30:54;123471786;214.997;514.997;OK;119.993;FAIL;-;-;-;-;-;-;-;-
16/12/2071;11:30:54;123471786;214.997;514.997;OK;119.994;FAIL;-;-;-;-;-;-;-;-
17/12/2071;12:30:54;123471786;214.997;514.997;OK;119.995;FAIL;-;-;-;-;-;-;-;-
18/12/2071;13:30:54;123471786;214.997;514.997;OK;119.996;FAIL;-;-;-;-;-;-;-;-
19/12/2071;14:30:54;123471786;214.997;514.997;OK;119.997;FAIL;-;-;-;-;-;-;-;-
```

Figura 68 – Conversión a .txt. 20000 líneas.

Se realiza una nueva verificación (mediante Arduino). Se comprueba el tiempo de ejecución de la aplicación.

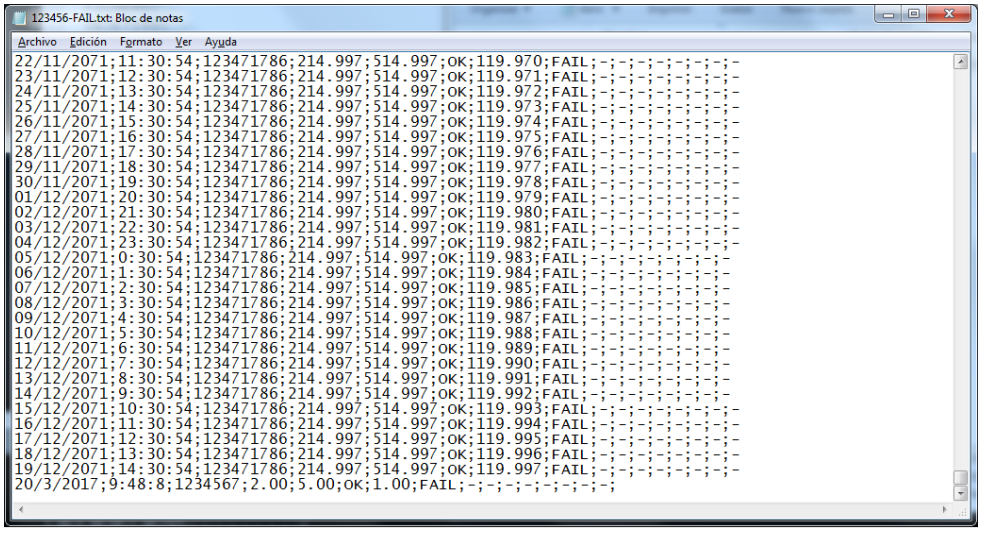


Figura 69 – Fichero txt. 20000 + 1 líneas.

Se cambia la extensión a .csv. Se observa que se ha añadido una nueva fila.

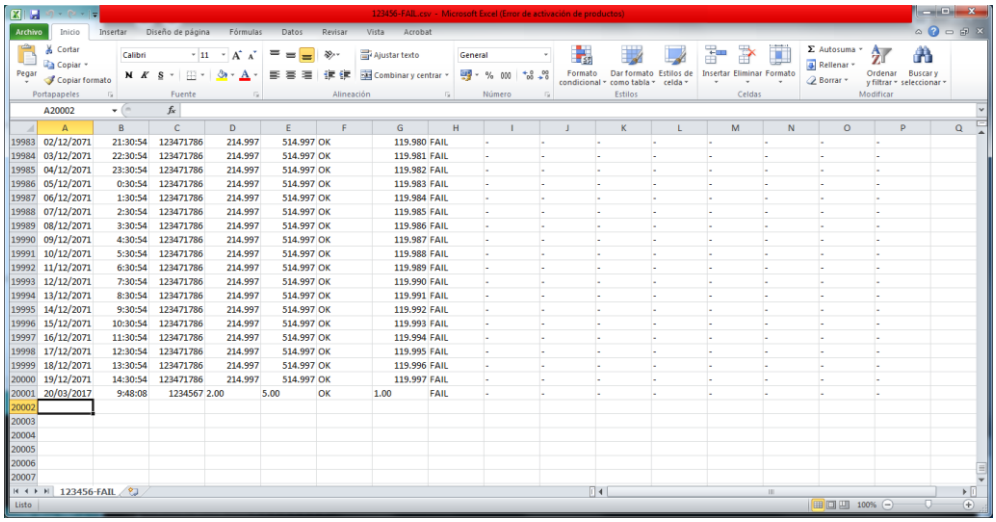


Figura 70 – Conversión a .csv. 20000 + 1 líneas.

Mediante esta prueba, se comprobó que el tiempo de verificación es el mismo, tanto en la primera verificación (creación del fichero), como en la última (lectura y escritura de un fichero con 20000 líneas). También se comprobó que el fichero no sufría daños al cambiar la extensión del mismo.

11.3.13 Diagrama de flujo

Se muestra el diagrama de flujo completo del programa junto con las rutinas de Arduino y Processing (figura 71). Para una mejor visualización, se adjunta también en el apartado 4 de los anexos.

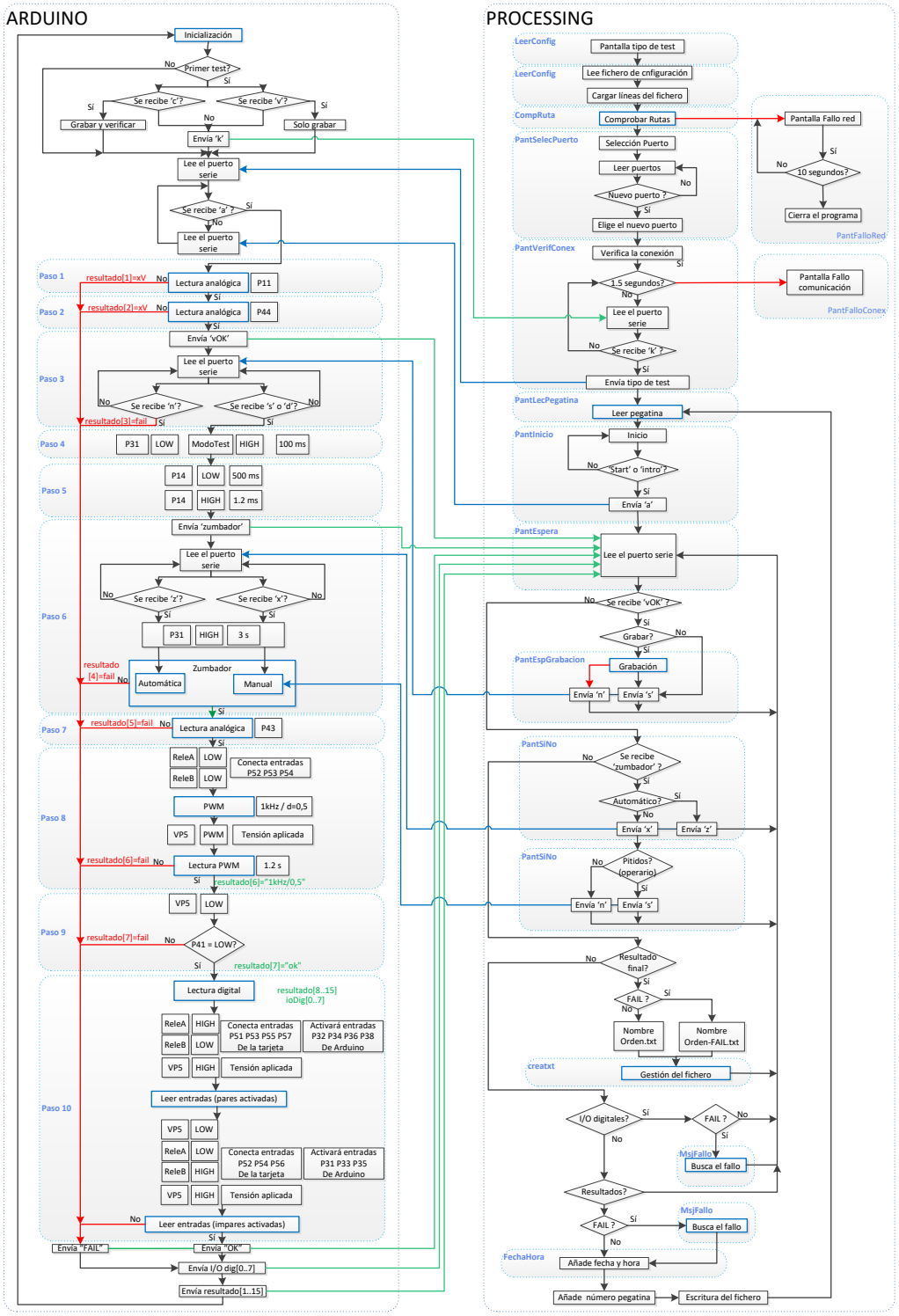


Figura 71 – Diagrama de flujo completo.

FUNCIONES/FICHEROS ARDUINO

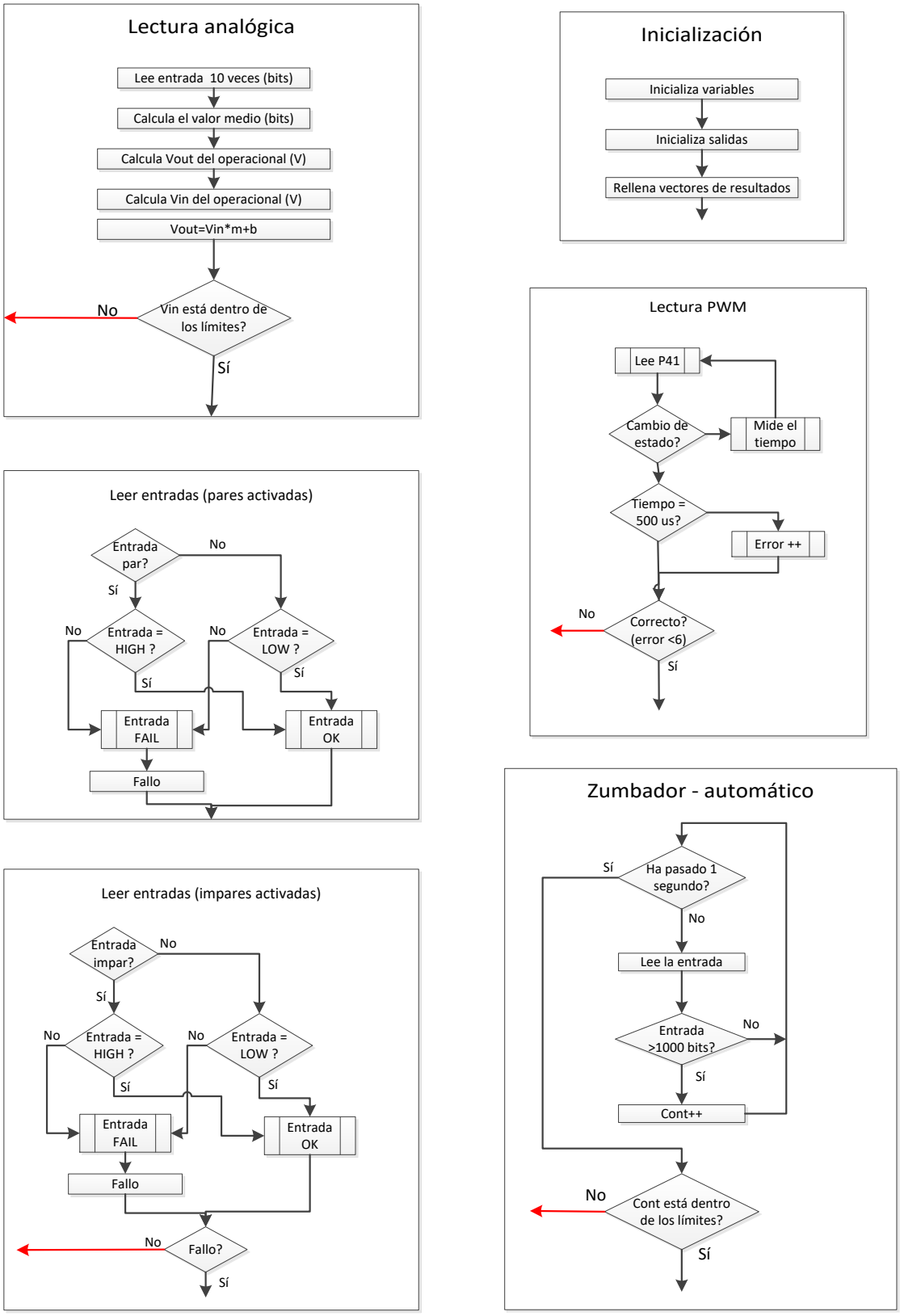
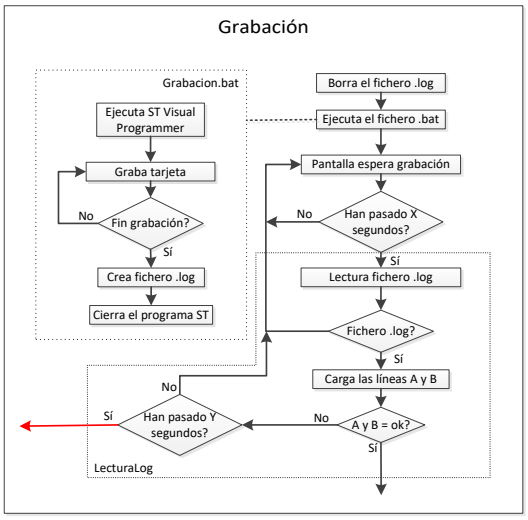
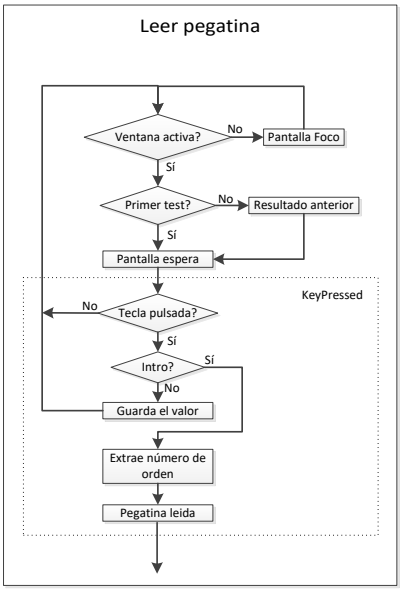
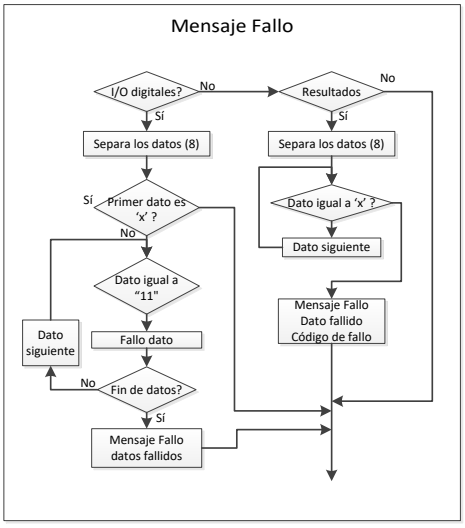


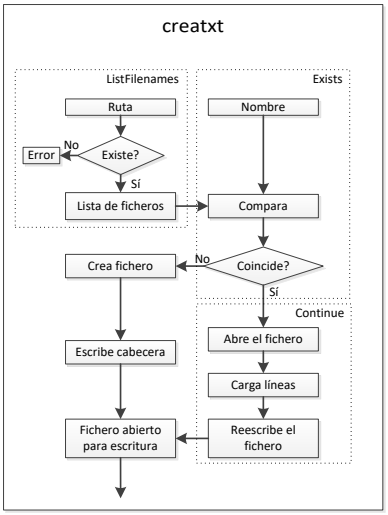
Figura 72 – Diagramas de flujo. Funciones en Arduino.

FUNCIONES/FICHEROS PROCESSING









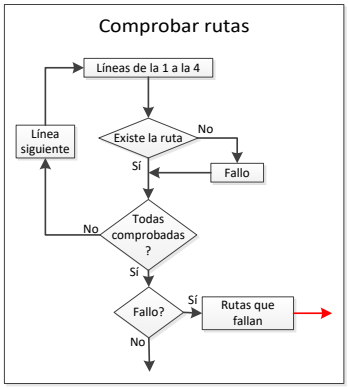


Figura 73 – Diagrama de flujo. Funciones en Processing.

11.4 Versiones anteriores del software

A lo largo de todo el proceso se fueron sucediendo cambios debido a peticiones del cliente. Además, se añadían nuevas funcionalidades al software para hacerlo más robusto y estético. A continuación, se enuncian las revisiones hechas a los códigos de Arduino y Processing.

11.4.1 Arduino

- Revisión 1: Se crea la primera versión del software.

- Revisión 2: no es necesario leer los valores de tensión en el paso 10. Basta con saber si están en alto o en bajo.

- Revisión 3: se añade la salida P4.2 al test y se crea el vector de entradas/salidas digitales (ioDig).

- Revisión 4: Se corrigen fallos en la asignación de entradas/salidas.
  La lectura de las salidas analógicas se realiza de manera inversa.

- Revisión 5: se conecta una carga individual a cada salida del puerto 3. Hasta entonces, se colocaba una sola carga que se conmutaba y se activaba con relés. Esto implica retirar dichos relés y ampliar el diseño de la PCB.
  Se adapta el programa al funcionamiento de los amplificadores ADum3190.
  Se separan las funciones en distintos ficheros.

- Revisión 6: se cambia el orden de algunas entradas para facilitar el ruteado de la PCB.
  La comprobación de la conexión con Processing se realiza una sola vez.
  Se ordena el código.

- Revisión 7: Se añade el sensor de sonido y se adapta el programa para realizar la comprobación del zumbador de manera manual o automática.

Se realizan modificaciones de los tiempos debido a cambios en la secuencia de verificación.

Se añada la entrada modoTest. La tarjeta ahora tiene distinto funcionamiento en función de si está en modo de prueba o no.

- Revisión 8: Arduino y Processing interactúan más. Comunicaciones bilaterales para la comprobación de tensiones de alimentación, detección del zumbador y grabación.

- Revisión 9: versión actual: Se diferencias todas las posibles opciones; grabar y verificar, solo grabar, solo verificar.
  Se reducen tiempos de ejecución.
  Se ajustan los parámetros del sensor de sonido según su nueva posición en el verificador.

11.4.2 Processing

- Revisión 1: Se crea la primera versión del software. La ventana de ejecución del programa es de 400x200 px.

- Revisión 2: Se divide le programa por ficheros
  Se añade la opción para seleccionar el puerto del Arduino

- Revisión 3: Se añaden textos de fallo a la pantalla.
  Se añade la función de grabación.
  Detecta fallos de red.
  Se crean paths para las rutas de resultados y se separan en carpetas OK y FAIL.

- Revisión 4: se agranda la ventana de ejecución del programa, pasa a 800x400.
  Se cambian las imágenes de los botones (más grandes y mejores).

Se mejora el código para la detección del Arduino.
Se pueden visualizar hasta seis puertos al ejecutar el programa.
Mejora la visualización del programa.

- Revisión 5: se añade una pantalla para que el operario seleccione si se van a grabar o no las tarjetas.
  Se agrega el nuevo test de la salida P4.2
  Se cambian los nombres de las variables para hacerlos más intuitivos
  Se organiza el código.

- Revisión 6: se arreglan errores de ejecución. (fallos con el zumbador, mal detección de fallos).
  Se cambia el orden de las peticiones; primero se elige el puerto y luego si grabar o no.

- Revisión 7: se modifica para realizar el test del zumbador automáticamente.
  Se incluye la verificación de la grabación con el optionByte.
  Se cambia el orden de las peticiones; Lectura de pegatina antes que grabación (la lectura de la pegatina se hace con la tarjeta fuera del verificador).
  Se añade una pantalla de resultado del test con dos opciones en caso de fallo: repetir el test, o continuar con otra tarjeta.
  Si hay fallo de grabación el programa da la opción de intentar grabar nuevamente.
  Se suprime el fichero con el que se detectaba la acción del ratón. Se sustituye por una manera más fácil, ahorrando muchas líneas de código.

- Revisión 8: La opción de grabar o no tarjetas aparece una sola vez al principio del programa.
  Se suprime la opción de repetir test en caso de fallo (no es conveniente).
  Se puede realizar el test con el teclado y con el ratón: \t para cambiar de opción y \n para elegir opción.

Se suprime la pantalla de resultado del test. Ahora el resultado aparece en la pantalla de leer la pegatina (se ahorra tiempo).

Se crea un fichero de configuración 'config.txt' con las rutas OK y FAIL, la opción de grabar o no las tarjetas, la opción del zumbador y las rutas de grabación.

Se elimina la pantalla de selección de puertos. Ahora se debe arrancar el programa con el USB de Arduino desconectado. Al conectarlo, el programa lo detectará automáticamente.

- Revisión 9: Cada X segundos se revisa el estado de la grabación. Si ha terminado, continúa con el programa, si no, vuelve a preguntar pasados otros X segundos (tiempo máximo Y segundos). X e Y se modifican desde el fichero de configuración.

  Se cambia el fichero de configuración para explicar todo. Este se lee solo una vez al principio del programa.
  Comprueba la accesibilidad de todas las rutas del fichero de configuración.
  Cambia las rutas en función de la versión de Windows del ordenador.

  Ahora se puede navegar por el programa con las flechas del teclado.
  El foco siempre está en la ventana del programa (era un problema, sobre todo al leer la pegatina. Si el foco no está en la ventana, no detecta pulsaciones en el teclado).

- Revisión 10: No hace falta cambiar las rutas en función de si es Windows XP o Windows 7 porque los ficheros de configuración, grabación y los ejecutables se guardarán en el disco local.

  Se crea un .bat para hacer la copia a los ordenadores automáticamente.

Se cambia el tiempo de verificación de la conexión de Arduino con Processing; pasa de 1 segundo a una variable inicializada en 1.5 segundos (debido a la velocidad de los ordenadores).

Se muestra una pantalla de fallo de rutas, indicando las rutas que fallan.

Cada error tiene asignado su código para facilitar la labor de los reparadores.

Se muestra una pantalla roja si en el momento de leer la pegatina no está activada la ventana del programa (no tiene el foco).

- Revisión 11 – Versión actual: Viene una nueva orden con una nueva referencia, un nuevo firmware del cliente y con la opción de solo grabar tarjetas.

  El número de orden la pegatina ahora está en la primera posición de la lectura (antes estaba en posiciones intermedias).

  Se crea un desplegable al principio con las opciones de: solo grabar, grabar y verificar, solo verificar.

  Se crean 2 ficheros batch y dos de configuración distintos, según la referencia que se vaya a verificar.

  Se arregla un error que colgaba el programa si no se finalizaba la grabación.

12 Puesta en marcha y validación

Una vez diseñado el verificador, se procede a realizar la instalación en los ordenadores de producción y probar su funcionamiento.

12.1 Denominación del verificador

El equipo debe darse de alta en la base de datos de la empresa, en donde se le asignará un código identificativo, el cual deberá colocarse en un lugar visible del mismo. Este código es único e inmodificable.

Después de acceder al sistema de la empresa, se le otorgó al verificador el código E1115, donde la letra ‘E’ indica que se trata de un verificador funcional.

12.2 Etiquetas

Se archiva un documento que contiene las etiquetas que se colocarán en la caja del verificador, tales como, el código del equipo, puertos y todos los puntos que necesiten una guía visual.

Figura 74 – Etiquetas del verificador.

12.3 Instalación en producción

Se crea un fichero batch para realizar automáticamente la instalación en los ordenadores de producción. Ejecutando este archivo se copian los ficheros necesarios para el funcionamiento del verificador en el ordenador.

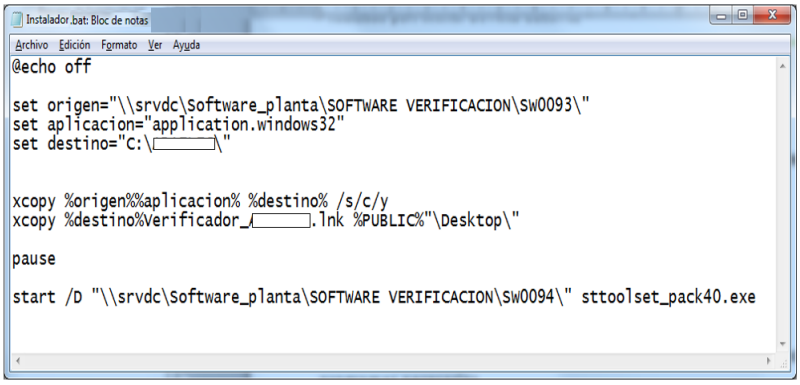


Figura 75 – Fichero de instalación automática.

Con este fichero, se copia la carpeta donde está el ejecutable del verificador, los archivos de grabación (grabación.bat y grabación_INV.bat) y los archivos de configuración (config.txt y config_INV.txt) desde la ruta origen (software_planta) a la ruta destino (C:\195\).

También se copia un acceso directo en el escritorio (este paso no funciona en los ordenadores con Windows XP. Se debe hacer manualmente).
Después de copiar los archivos, ejecuta el fichero de instalación del programa de grabación ST-Toolset.

12.4 Pauta de verificación

La pauta de verificación es un documento en el que se explica con detalle los pasos a seguir por el operario a la hora de realizar la verificación, así como los útiles necesarios para llevarla a cabo. Este documento se imprime y se mantiene con el verificador, ya que el puesto de trabajo y el operario seleccionado varían según el día.
La pauta se realiza y se nombra con un formato definido por la empresa.
Se adjunta el archivo en el apartado 5 de los anexos.

12.5 Medición de tiempos

En las primeras versiones del software del verificador se fijaba un tiempo para la grabación y para la comprobación de la comunicación entre Arduino y Processing. Al instalar este programa en los ordenadores de producción se observó que en algunos ordenadores fallaba la comunicación y en otros fallaba la grabación, incluso algunos llegaban a funcionar aleatoriamente.
Se realizó la medición de los tiempos en los distintos tipos de ordenadores de producción y conectando en distintos puertos.

Como se puede observar en las tablas 13 y 14, el tiempo de grabación varía mucho de un ordenador a otro, mientras que el de verificación es constante.
La zona de verificación cuenta con 8 ordenadores normales y 2 industriales. Con la ayuda del servicio informático de la empresa, se intentó solucionar el problema de los equipos industriales, sin embargo, esto no pudo ser posible.

Ordenador industrial						
	USB 3.0			USB 2.0		
firmware	grabación	verificación	total (seg)	grabación	verificación	total (seg)
8CG07S0-INV	11	13	**24**	11	13	**24**
8CG07S0	11	13	**24**	11	13	**24**

Tabla 13 – Tiempos de grabación. Ordenador industrial.

Ordenador normal						
	USB 3.0			USB 2.0		
firmware	grabación	verificación	total	grabación	verificación	total
8CG07S0-INV	6	13	**19**	7	13	**20**
8CG07S0	8	13	**21**	6	13	**19**

Tabla 14 – Tiempos de grabación. Ordenador normal.

Para ampliar los datos, se decidió medir el tiempo de grabación en ordenadores del departamento de I+D, ya que uno de ellos es el más potente de la empresa. Se obtuvieron los siguientes datos:

Equipo	Grabación
Última generación - En I+D	4,5 segundos
Ordenador personal - En I+D	6 segundos

Tabla 15 – Tiempos de grabación. Ordenador potente.

Se observa la gran diferencia entre el ordenador potente y el resto.
Obviamente es impensable renovar los equipos del área de producción, pero se pueden dar por aceptables los tiempos de grabación de alrededor de 6 segundos.

Se llegó a la conclusión de que esta verificación siempre se realizaría en ordenadores normales.

Además, esto llevó a realizar una gran mejora al software del verificador, esta fue, el paso de un tiempo fijo para la grabación a un tiempo configurable. En lugar de dar 10 segundos al programa para grabar, ahora se revisa cada 2 segundos si ha finalizado la grabación un número máximo de veces. Esto quiere decir que, si un ordenador tarda menos de 10 segundos en grabar, se ahorrará la diferencia de tiempo, y si tarda más de 10 segundos, el programa no fallará y esperará a que termine la grabación (siempre que no exceda un tiempo límite. Fijado en 16 segundos. puede modificarse).

13 Presupuesto

Se realiza el presupuesto para el proyecto teniendo en cuenta las diferentes etapas que conlleva: definición, ejecución, conclusión. Primeramente, se detallan los gastos en material. Luego, se realiza el presupuesto general del desarrollo que incluye también las horas de trabajo empleadas.

Coste materiales

Electrónica auxiliar		Uds	Precio Ud.	Total
Optoacoplador	4N35	15	0,67	10,05
Relé	G5V-1 DIP10 (6)	2	2,07	4,14
Diodo LED	LED 5mm	1	0,064	0,064
Conector	Aéreo 9 vías	2	1,95	3,9
	PCB 9 vías	2	0,594	1,188
	Aéreo 8 vías	2	1,87	3,74
	PCB 8 vías	2	0,567	1,134
	Aéreo 4 vías	2	1,1	2,2
	PCB 4 vías	5	0,177	0,885
	Aéreo 2 vías	2	0,84	1,68
	PCB 2 vías	2	0,118	0,236
Transistor	BC 547C TO-92A	2	0,037	0,074
Diodo	1N4148	2	0,052	0,104
Resistencia	RCONV - 2.5x10mm 0,5W	44	0,05	2,2
	RCONV - 7.5x24mm 5W	7	1,364	9,548

Condensador	Con disc 5mm	15	0,574	8,61
	macho x20	2	0,74	1,48
Tira de pines	hembra x32	1	1,2	1,2
	DIL 14	10	0,636	6,36
Zócalos	DIL 8	20	0,358	7,16
Amplificador analógico	Adum 3190	3	2,53	7,59
Fusible	FuseT4A	2	0,3	0,6
Placa de prototipado	RE200-LF 100x160x1,5. 1mm Holes	6	6,01	36,06
Total				**110,203**

Componentes externos		Uds	Precio Ud.	Total
Arduino	Mega2560	1	43,51	43,51
Sensor Audio	MAX 9814	1	11,16	11,16
Grabador	ST-Link	1	20,42	20,42
Fuente de tensión	12V/2,1A	1	27,5	27,5
Total				**102,59**

Cama de pinchos y caja		Uds	Precio Ud.	Total
Cama de pinchos				
Maderas		1	25	25
Pinchos		32	1,05	33,6
Brida manual		1	11,89	11,89
Caja				
Maderas		2	25	50
Interruptor iluminable		1	4	4
Interruptor SPST		1	1,22	1,22
Perfiles		10	0,96	9,6
Patas goma		4	2,4	9,6
Conectores				
USB-A USB-B		2	5,25	10,5
Conector de red		1	3,24	3,24
Total				**158,65**

Presupuesto general				
Coste por hora: 50 €				
Operación	**Comentario**	**horas**	**material**	**coste**
Definición y diseño				
Estudio y definición test funcional		4		200
Selección de componentes externos	Arduino, grabador, otros	2	102,59	202,59
Esquemas eléctricos	Esquemas en Altium	30		1500
Definición de mecanizados	Diseño en Inventor y Altium	24		1200
Obtener puntos de test		1		50
Subtotal		**61**	**102,59**	**3152,59**

Caja y cama de pinchos	**Comentario**	**horas**	**material**	**coste**
Materiales			158,65	158,65
Hacer programa mecanizado	Programa para la máquina	1		50
Mecanizado	Tiempo que tarda la máquina	3		150
Montar pinchos		2		100
Cablear cama		3		150
Subtotal		**9**		**608,65**

Utillaje	**Comentario**	**horas**	**material**	**coste**
Montaje electrónica auxiliar		18	110,2	1010,2
Montaje mecánico		3		150
Conexionados eléctricos		25		1250
Subtotal		**46**	**110,2**	**2410,2**

Software del sistema de test	**Comentario**	**horas**	**material**	**coste**
Estudio del micro a programar	Selección tarjeta comercial	5		250
Estudio software del cliente	Estudio del firmware	2		100
Desarrollo fuentes programa		30		1500
Depuración SW de test		10		500
Subtotal		**47**		**2350**

Documentación	**Comentario**	**horas**	**material**	**coste**
Documentar en servidor		10		500
Hacer pauta de verificación		1		50
Subtotal		11		550

TOTAL DESARROLLO	**9071,44**

El coste del desarrollo es de NUEVE MIL SETENTA Y UN EUROS CON DIEZ CÉNTIMOS.

14 Resultados y conclusiones

Se combinaron los conocimientos derivados de la formación universitaria, junto con los adquiridos en la estancia en la empresa para llevar a cabo el desarrollo del proyecto.

Se realizó de manera satisfactoria el verificador funcional. Actualmente se han verificado un total de 6300 de las primeras 10000 tarjetas.

La aplicación es clara y los pasos a seguir, explicados en la pauta de verificación son sencillos y se realizan de forma cómoda.

Se consiguió un tiempo total (grabación más verificación) de alrededor de 20 segundos, dando margen más que suficiente al operario para la manipulación de la tarjeta. En total, la verificación de una tarjeta, paso a paso tarda entre 35 y 40 segundos. Esta reducción de tiempo es debida al alto grado de automatización conseguido.

La integración y la coordinación en el equipo de trabajo arrojaron resultados muy satisfactorios y permitieron el crecimiento en lo que a competencias derivadas de la dirección de proyectos se refiere.

Cada proceso realizado fue correctamente documentado en la base de datos de la empresa. Se realizó una reunión informativa sobre este tema al final de la puesta en marcha.

15 Líneas futuras

Después de haber observado el comportamiento del verificador y estudiando más detalladamente sus componentes, se encuentran puntos que se pueden mejorar con vistas a futuros pedidos.

Es conveniente añadir las librerías utilizadas en Arduino al propio código del programa, de tal manera que no sea necesario descargarlas e instalarlas en cada ordenador donde se requiera cargar el programa a la tarjeta de Arduino.

Mejorar el código de Arduino manipulando directamente los puertos del microcontrolador, dando más rapidez a algunas de las funciones, sobre todo en las lecturas digitales.

Mejorar el código de Processing y su integración con Arduino. Hay operaciones que se pueden realizar de forma más sencilla. Los constantes cambios a lo largo del desarrollo no ayudaron a tener un código completamente limpio y ordenado.

Se estudian por parte de la empresa varias opciones con el fin de agilizar el proceso de verificación debido a que el cliente ha comentado que realizará alrededor de 50000 tarjetas este año.

- Realizar una copia del verificador: no supondría mucho tiempo ni dinero, pero debido a la saturación del área de producción, no es posible tener a dos personas trabajando para un solo pedido.

- Realizar un nuevo verificador para verificar varias tarjetas a la vez: es la opción más viable para la empresa. Las tarjetas se fabrican en paneles de seis. Se pretende realizar un verificador para paneles, las tarjetas se verifican de 6 en 6, ganando mucha velocidad, que compensa el tiempo invertido en el diseño.

- Realizar la verificación mediante el Wayne Kerr: sería más rápido debido a la potencia de la máquina, pero esta se saturaría, estaría en constante uso para estas tarjetas y no podría ser usada para otras que de verdad lo necesiten.

Todas las opciones se analizan en el departamento de compras. Hasta la fecha, no se ha tomado una decisión al respecto.

Sustituir en interruptor de inicio por un pincho interruptor. Hasta el momento, el interruptor de inicio se usa para alimentar la tarjeta después de colocar la tarjeta en la cama y bajar la brida. Con el pincho, se detectaría mecánicamente que la tarjeta está lista para iniciar la verificación, y esta se podría iniciar automáticamente (es muy común iniciar el test olvidando conectar la tensión).

Se estudia la posibilidad de traspasar el código de Processing a Visual Studio, ya que este último es el que comúnmente se utiliza en la empresa.

16 Bibliografía

[1] Carlos Dominguez Pacheco, “ELECTRÓNICA FALCÓN Tarjetas electrónicas”, 24-nov-2012. [Vídeo]. Disponible en: https://www.youtube.com/watch?v=hEPShxZHI4c

[2] “Electrónica Falcón SA”, 2017. [En línea]. Disponible en: http://www.electronicafalcon.com/

[3] Empresite, elEconomista. “Electrónica Falcon Sa – Peralta”, 2017. [En línea]. Disponible en: http://empresite.eleconomista.es/ELECTRONICA-FALCON.html

[4] JEDEC – Standars & documents, “JOINT IPC/JEDEC STANDARD FOR HANDLING, PACKING, SHIPPING, AND USE OF MOISTURE/REFLOW SENSITIVE SURFACE-MOUNT DEVICES”, 2012. [En línea]. Disponible en: ttps://www.jedec.org/standards-documents/results/j-std-033c

[5] Schneider Electric, “Normativa REACh y RoHS”. [En línea]. Disponible en: http://www.schneiderelectric.es/sites/spain/es/productos-servicios/normativa-rohs/normativa-rohs.page

[6] Electrónica Falcón. (2017, mayo). Documentación interna de la empresa.

[7] F. Valdés Pérez, R. Pallás Areny, *Microcontroladores: fundamentos y aplicaciones con PIC.* España: Marcombo, 2007.

[8] N. Goilav, G. Loi, *Arduino: Aprender a desarrollar para crear objetos inteligentes*. Barcelona: ediciones eni, 2016.
Disponible en: goo.gl/iPPDxK

[9] M. Banzi, *Getting Started with Arduino (2 ed.).* USA: O'Reilly, 2011. Disponible en: goo.gl/B7svMo

[10] ADRIAN MB, "Que es Processing??", 9-jul-2014. [En línea]. Disponible en: http://blog.make-a-tronik.com/que-es-processing/

[11] Jaime Munarriz, "Processing. Una breve introducción." [En línea]. Disponible en: http://avantspace.org/processing/p5history.htm

[12] "Introducción a DOS", Jun-2017. [En línea]. Disponible en: http://es.ccm.net/contents/170-introduccion-a-dos

[13] Autodesk Inventor, [En línea]. Disponible en: https://www.autodesk.es/products/inventor/overview

[14] 3D CAD portal, "Autodesk inventor". [En línea]. Disponible en: http://www.3dcadportal.com/autodesk-inventor.html

[15] Altium, "Altium Designer". [En línea]. Disponible en: http://www.altium.com/altium-designer/overview

[16] ST electronics, "ST-LINK/V2", 2017. [En línea]. Disponible en: http://www.st.com/en/development-tools/st-link-v2.html

[17] Vonroll, "Adaptonit 105 AS". [En línea]. Disponible en: http://www.vonroll.com/en/product-detail/ep-105-as/?id=product7620

[18] Javier vizcarguenaga, "Pinchos de test", 09-may-2012. [En línea]. Disponible: https://vizcarguenagajavier.wordpress.com/2012/05/09/pinchos-de-test/

[19] Oracle, Java documentation, “Manipulating Characters in a String”. [En línea]. Disponible en: https://docs.oracle.com/javase/tutorial/java/data/manipstrings.html

[20] P. Reverte Gomez, “GESTIÓN DE CADENAS”, [En línea]. Disponible en: http://dis.um.es/~lopezquesada/documentos/IES_1213/IAW/curso/UT3/ActividadesAlumnos/15/index.html

[21] ADRIAN MB, “[Tutorial Processing] crear archivos TXT”, 2-jul-2014. [En línea]. Disponible en: http://blog.make-a-tronik.com/tutorial-processing-crear-archivos-txt/

[22] ADRIAN MB, “[Tutorial Processing] leer archivos TXT”, 2-jul-2014. [En línea]. Disponible en: http://blog.make-a-tronik.com/tutorial-processing-leer-archivos-txt/

[23] Arduino, “:: Timer1 ::”. [En línea]. Disponible en: http://playground.arduino.cc/Code/Timer1

[24] Sami Mughal, “Creating a variable frequency PWM output on Arduino UNO”, 7-abr-2011. [En línea]. Disponible en: http://www.oxgadgets.com/2011/04/creating-a-variable-frequency-pwm-output-on-arduino-uno.html

[25] Prometec, “Arduino y los timers”. [En línea]. Disponible en: http://www.prometec.net/timers/

Desarrollo de un equipo para verificar funcionalmente tarjetas electrónicas

ANEXOS

ÍNDICE DE ANEXOS

1 Estudio de tarjetas comerciales

1.1 Módulos

Fabricante	Modelo	Alimentación	Micro	Reloj MHz	I/O	ADC	PWM	DAC	Timer	Flash Mem	UART	SPI	I2C	USB	CAN	BT	WiFi	Ethernet	Tamaño	Peso	Precio
Arduino																					
	Uno	5 V	ATmega328P (8bit)	16 MHz	14	6 (10 bit)	6 (8bit)	-	3*	32KB(0,5KB)	1	1	1	-	-	-	-	-	68.6 x 53,4	25 g	22,19 €
	Leonardo	5 V	ATmega32u4 (8bit)	16 MHz	14	12 (10 bit)	7 (8 bit)	-	5	32KB(4KB)	1	1	1	-	-	-	-	-	68.6 x 53,3	20 g	18,00 €
	101	3,3 V	Intel Curie (32bit)	32 MHz	14 (5V)*	6()	4(8bit)	-		196KB	1	1	1	-	-	1	-	-	68.6 x 53,4	34 g	28,65 €
	MKRZero	3,3 V	SAMD ARM (32bit)	48MHz	22	7(8/10/12bit)	12(8bit)	1(10bit)		256KB	1	1	1	1	-	-	-	-			20.90 €
	Industrial 101	5 V	ATmega32u4 (8bit)	16 MHz	7	4(10bit)	7(8bit)	-	5	32KB	1	1	1	1	-	-	1	100	51 x42	12g	35,00 €
	Mega	5 V	ATmega2560/16U2(8bit)	16 MHz	54	16(10 bit)	15(8bit)	-	5	256KB(8KB)	4	1	1	1	-	-	-	-	101,52 x 53,3	37 g	35,00 €
	Zero	3,3 V	ATSAMD21G18(32)	48 MHz	20	6(12 bit)	10(8bit)	1(10bit)	5+	256KB(8KB)	2	1		2*	-	-	-	-	68 x 30	12 g	42,90 €
	Zero Pro	3,3 V	ATSAMD21G18(32)	48 MHz	20	6(12 bit)	11(8bit)	1(10bit)	5+	256KB(4KB)	2	1	2	2*	-	-	-	-	68 x 30		46,89 €
	Due	3,3 V	AT91SAM3X8E (32bit)	84 MHz	54	12(12bit)	12(8bit)	2(12bit)(2,20Vpp)	3	512KB	4	1	2	2	1*	-	-	-	101,52 x53,3	36 g	36,00 €
	Mega ACK	5 V	ATmega2560/16U2(8bit)	16 MHz	54	16(10bit)	15(8bit)	-	5	256KB(8KB)	4	1	1	1	-	-	-	-	101,52 x53,3	36 g	44,00 €
	M0	3,3 V	ATSAMD21G18	48 MHz	20	6(12 bit)	12(8bit)	1(10bit)	RTC	256KB	3	1	1	-	-	-	-	-	68,5 x 53	21 g	28,98 €
	M0 Pro	3,3 V	ATSAMD21G18	48 MHz	20	6(12 bit)	12(8bit)	1(10bit)	RTC	256KB	3	1	1	1	-	-	-	-	68,58 x 53,34	22 g	44,32 €
	TIAN	3,3 V		48 MHz	20	12(10bit)	12(8bit)	1(10bit)		256KB	1	1	1	1	-	1	1	1	68,5 x 53	36 g	87,00 €
	Primo*	3,3 V	Nortdc nRF52832	64 MHz	20	6	3	1		512KB				1		1	1			34 g	-
	Star OTTO*	3,3 V	STM32F469BI	180 MHz	54	14	12	2									1		101,52 x53,3	34 g	-
LaunchPad					GPIO																
	MSP-EXP430FR5994	3,3 V	MSP430FR5994(16)	16 MHz	68	20(12bit)	5	-	6(16)	256KB	4	8	4	-	-	-	-	-			15,99 $
	MSP-EXP432P401R	3,3 V	MSP432P401R(32)	48 MHz	84	24(14bit)	5	-	4(16)+2(32)	256KB	4	8	4	-	-	-	-	-			12,99 $
	EK-TM4C123GXL	3,3 V	TM4C123GH6PM(32)	80 MHz	43	12(12bit)	16	-	6(32)+6(64)	256KB	8	4	4	-	2	-	-	-			12,99 $
	EK-TM4C129EXL	3,3 V	TM4C129ENCPDT(32)	120 MHz	90	20(12bit)	8	-	8(16/32)	1024KB	8	4	10	1	2	-	-	10/100			24,99 $
Piccolo	LaunchXL-F28069M	3,3 V	TMS320F28069M(32)	90 MHz	54	12bit-16ch(dual)	16(8high)	-	3	256KB	2	2	1	1	1	-	-	-			24,99 $
Delfino	LaunchXL-F28377S	3,3 V	TMS320F28377S(32)(2)	200 MHz	84	6(16/12bit-24ch)	24(16high)	3(12bit)	3(+3)	1024KB	4	3	2	1	2	-	-	-			29,99 $
Hercules	LaunchXL2-570LC43	3,3 V	TMS570LC4357(32)	300 MHz	168	2(12bit-41ch)	14	-	2	4096KB		5	2	-	4			10/100			29,99 $
Hercules	LaunchXL2-RM57L	3,3 V	RM57L843(32)	320 MHz	168	2(12bit-41ch)	14	-	2	4096KB	4	5	2	-	4	-	-	10/100			29,99 $
Net Duino																					
	Netduino 3		STMicro ST32F4	168 MHz	22	6(12bit)	6	-	-	-	4	1	1	-	-	-	-	-	-	-	43,15 €
Teensy																					
	Teensy 3.2	3,3 V	MK20DX256VLH7	72 MHz	34 (5v)*	21(13bit)	12	1(12bit)	7	-	-	1	2	1	1	-	-	-	-	-	19,80 $
	Teensy 3.6	3,3 V	MK66FX1M0VMD18	180 MHz	62	25(13bit)	22	2(12bit)	14	1024KB		3	4	6	2	-	-	10/100	-	-	29,99 $
Ruggeduino	Iguales que arduino, disponen de dos modelos (UNO,MEGA).Protecciones para las entradas, corrientes mas elevadas .Considerablemente mas caros																				
	UNO																				54,95$
	MEGA																				109,95$

1.2 Control

		GPU	RAM	Memoria	GPIO	ADC	PWM	Timer	UART	SPI	I2C	CAN	USB	WiFi	Bluetooth	Ethernet	HDMI	Extra	Precio
Raspberry PI 3																			
	PI 3	1,2GHz 64bit quad-core ARMv8	1 GB	SD	40	-	7	4	1	2	1	-	4	802.11n	4.1/BLE	10/100	1.4/1080/Full		39,90 € +
BeagleBone/Board																			
	Black	AM335x 1 GHz ARM Cortex-A8	512 MB	2GB/SD	92(65)	7(1,8V)	8	4	4	2	2	2	1	si	si	10/100	micro*	-	61,11 €
	SunCloud Enhanced	AM335x 1 GHz ARM Cortex-A8	1 GB	4GB/SD	92(65)	7(1,8V)	8	4	3	2	2	2	4	si	si	10/100/1000		-	Crowfunding
	xM (board)	AM37x 1GHz ARM Cortex-A8	512 MB	MMC/SD					si	si	si		4	-	-	10/100	-	DB9 RS232	149,00 $
	X-15 (board)	AM5728 2x1,5 ARM Cortex-A15	2GB	4GB/SD	157				7	1	1	2	1+3(3.0)	-	-	2x(10/100/1000)	1.4/1080/Full	-	240,11 €
Odroid																			
	C1	1,5GHz Amlogic ARM quad-core 32 bits ARMv7	1 GB	8-16GB*/SD	40	2 x 10bit	si		si	si	si	-	4	-	-	10/100/1000	1.4/1080/Micro	-	35,00 $
	C2	1,5GHz Amlogic S905 64-bit quad-core 64 bits ARMv8	2 GB	8-32GB*/SD	40	2 x 10bit	si		si	-	2	-	4	-	-	10/100/1000	2.0/4K/Full	-	46,00 $
	XU4	2GHz Samsung Exynos5422 64-bit Octa core	2 GB	8-64GB *	30 (1,8V)	si	si		si	si	si	-	2+1(3.0)	si	si	10/100/1000	1.4/1080/Micro	-	59,00 $
Arduino																			
	Samsung ARTIK 10	1,5GHz quad core ARM CortexA15	2 GB	16GB*/SD	95	6 x12bit	si	si	si	1	4	-	1+1(3.0)	802.11n/ac	4.1/BLE	-	1.4	ZigBee 802.15.4	149,00 $
	Intel GalileoGen 2	SoC Quark X1000 32-bit	256 MB	SD	20	6	6		2	1	1	-	1	-	-	-	-	-	79,90 €
	Intel Edison	Dual core Intel Atom CPU/ Intel Quark 32bit	1 GB	4GB */SD		6	4		2	1	1	-	1	-	-	-	-	-	82,64 €
Udoo																			
	Neo Basic	ARM Cortex-A9 core Cortex-M4	521MB	SD	32		8		3	1	3	2	2	-	-	10/100	micro	Acel/Giro/Mag/Arduino comp	49,90 $+VAT
	Neo Expanded	ARM Cortex-A9 core Cortex-M4	1GB	SD	32		8		3	1	3	2	2	802.11b/g/n	4.1/BLE	-	micro	Acel/Giro/Mag/Arduino comp	59,90 $+VAT
	Neo Full	ARM Cortex-A9 core Cortex-M4	1GB	SD	32		8		3	1	3	2	2	802.11b/g/n	4.1/BLE	10/100	micro	Acel/Giro/Mag/Arduino comp	64,90 $+VAT
	Quad	1GHz Freescalei.MX 6 ARM Cortex-A9 quad core	1GB	Sata/SD	76	14						-	4	si	-	10/100/1000	micro		135;00 $+VAT
	X 86 Basic	2GHz Intel Atom X5-E8000 quad core	2GB	Sata/SSD/SD	+20	6 x 10bit	14		2	1	2	-	2+3(3.0)	*	4.1/BLE	10/100/1000	Full(Intel HD Graphics)	Acel/Giro/Mag/Arduino comp	125,00 $+VAT
	X 86 Advanced Plus	2,24GHz Intel Celerion N3160 quad core	4GB	Sata/SSD/MMC/SD	+20	6 x 10bit	14		2	1	2	-	2+3(3.0)	*	4.1/BLE	10/100/1000	Full(Intel HD Graphics)	Acel/Giro/Mag/Arduino comp	165,00 $+VAT

1.3 Extensiones

Relés	Fabricante	Modelo	V out max	I out max	P max	V con	I con	Pin control	in opto*	V in opto	Max reles	Compatibilidad	Precio
	ELECTAN-RELAYC	Control reles con 4 optoacopladores	50 V	500mA	1W(2,25W)	5,Vin, Vext		5, 6, 7 (A2,A3)	2, 3, 8, 9	5-24V	200 (25 mod)	UNO	22,39 €
	ELECTAN-RELAYS	Control reles	50 V	500mA	1W(2,25W)	5,Vin, Vext		5, 6, 7 (A2,A3)		-	200 (25 mod)	UNO	16,34 €
	Arduino	Extensión con 4 reles	48 V	2A	60W	3,3/5 V	35mA(140)	4, 7, 8, 12(A2, A3, 5, 6, I2C)	-	-	-	5 V/ 3,3 V	23,76 €
	Arduino	Modulo con 8 reles	30 V	10A		5V		-	-			-	19,97 €
	OEM- RELE4C	Modulo con 4 reles optoacoplados				12V(5V)	15-20mA	-	-	-	-	-	9,95 €
	OEM- RELE8C	Modulo con 8 reles optoacoplados	30 V	10A		12V(5V)	15-20mA	-	-	-	-	-	11,95 €
		Modulo con 16 reles optoacoplados	30 V	10A		5/12V		-	-	-	-	-	10 - 25 €
	Modulos reles contro serie												
	USB-RLY8	8 reles usb	30 V	1A		USB							55,15 €
	USB-RLY16	8 reles alta potencia usb	24 V (220AC)	16A		12 V	500mA						68,45 €
	WIFI8020	20 reles 8 entradas analogicas	24 V (220AC)	16A		12 V	1A						137,15 €
	+ Arduino UNO												22,19 €

Multiplexado			in/out analog/dig					Pins control			Max shield		
	Mux Shield	De/Multiplexor	48	3 x 16				2, 3, 4, 5 (A0, A1, A2)				UNO	24,90+VAT
	Mux Shield II	De/Multiplexor	48	3 x 16				2, 3, 4, 5 (A0, A1, A2)			10	UNO/MEGA	24,90+VAT
	CD74HC4067	De/Multiplexor	16			5 V							3 - 5 €
	CD4067B	De/Multiplexor	16			5/10/15 V			14MHz				
	+ Arduino UNO												22,19 €

DAC			Resolucion	In V+	In V-	Out Vpp	Out I protec.	Frecuencia					
	Visgence	Power DAC Shield MCP4921	12 bit	5/15 V	-5/-15 V	0-30 V	0,25/0,10 mA	220 KHz				UNO/MEGA	32,35 €
	Visgence	Power DAC Module MCP4921	12 bit	3,3/12 V	-3,3/-12 V	0-24 V	0,25/0,10 mA					5V/ 3,3V	32,35 €
	SparkFun	MCP4725	12 bit	2,7/5,5 V									4,95 $
		CJMCU-MCP4725	12 bit	2,7/5,5 V									1,91 €
	+ Arduino UNO												22,19 €
	Arduino DUE		2 x 12bit	0,55/2,75 V	-	2,20 V							36,00 €

ADC/DAC			Resolucion	In V+	In V-	Rango	ADC	DAC					
	Digilent & TI	Analog Shield	16 bit	5 V	-5 V	5,5 mA/V	4 Ch	4 Ch				UNO	49,99 $
	+ Arduino UNO												22,19 €
	Arduino DUE		2 x 12bit	0,55/2,75 V	-	2,20 V	12x12 bit	2x12 bit					36,00 €

Generador Señal			Vin	Resolucion DAC	Vout pp	Registros	SPI	Generador	Hz				

	SparkFun	MiniGen	3,3/5 V	10 bit	1 V	2 fase / 2 frec.	40MHz	Seno/Cuadrada/Tri	3MHz				29,95 $
	+Arduino Pro Mini												9,95 €

CAN		CAN Controller / Transceiver	CAN		SPI		Buffer	Conexión					
	Shield V1.2	MCP2515 / MCP2551	V2.0 B	1 Mb/s	10 MHz	11/29 bit	2	DB9/ pins	-	-	-	UNO	29,98 €
	Sparkfun DEV-13262	MCP2515 / MCP2551	V2.0 B	1 Mb/s	10 MHz		2	DB9	SD	conex. LCD y GPS	Joystick	UNO	31,55 €
	Chuangzhuo	MCP2515 / MCP2551	V2.0 B	1 Mb/s	10 MHz	11/29 bit	2	DB9/ pins	-	-	-	UNO	18,71 €
	+ Arduino UNO												22,19 €

Audio		Analog out	Digital out								
	max4466	si	-								2,02 €
	KY-038	si	si								1,42 €

Alimentación		V in (V)	V out (V)	P out	I out (A)						
DFRobot	Power shield	4,5-35	L-3,3-5-9-12	15 W	2A (3A pico)					UNO/MEGA	23,60 €
Control de cargas con MOSFET											
SparkFun	Power Driver Shield (control)	3 x 5V	3 x 12V							UNO/MEGA	19,95 $
+ Arduino UNO											22,19 €

Comunicación	Estandar		Conexión	Tipologia Red	Velocidad						
Multoprotocol Radio Shield v2.0											40,00 €
+CAN Bus Module	ISO 11898	Par trenzado	DB9	Multimaestro	125-1000Kbps	Half-duplex	Diferencial			Arduino/ Pi	66,00 €
+RS-485/Modbus	EIA RS-485	Par trenzado	DB9	Punto a punto/Multipunto/Multo-drop	-	Half/Full - duplex				Arduino/ Pi	60,00 €
+RS-232/Modbus	TIA-232-F		DB9	Punto a punto/Multipunto/Multo-drop	115200 bps	Ful-duplex				Arduino/ Pi	21,00 €
RS232 Shield V2			DB9								11,90 $
RS485 Shield V2											11,99 $
+ Arduino UNO											22,19 €

Ehernet		UART	SPI	I2C	CAN	USB	WiFi	Bluetooth	Ethernet	HDMI	Precio
BeagleBone	SunCloud Enhanced	3	2	2	2	4	si	si	10/100/1000		Crowfunding
Odroid	C1	si	si	si	-	4	-	-	10/100/1000	1.4/1080/Micro	35,00 $
Arduino TIAN	TIAN	1	1	1	1	-	1	1	10/100/1000		87,00 €

2 Pruebas Amplificador analógico ADum3190

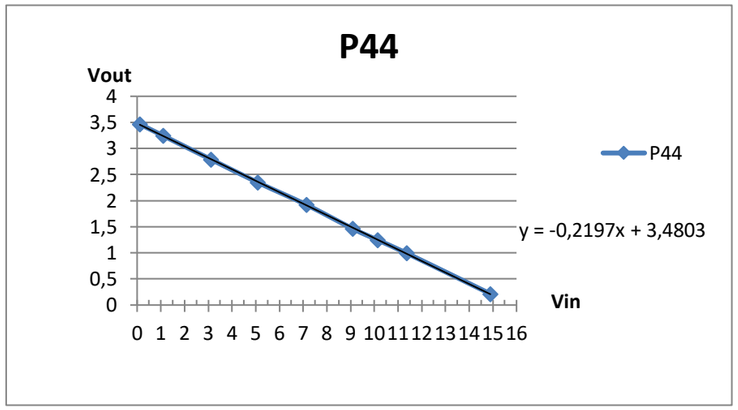
MEDIDAS DE TENSIÓN EN LOS AMPLIFICADORES ADuM 3190 (Entradas analógicas de Arduino)

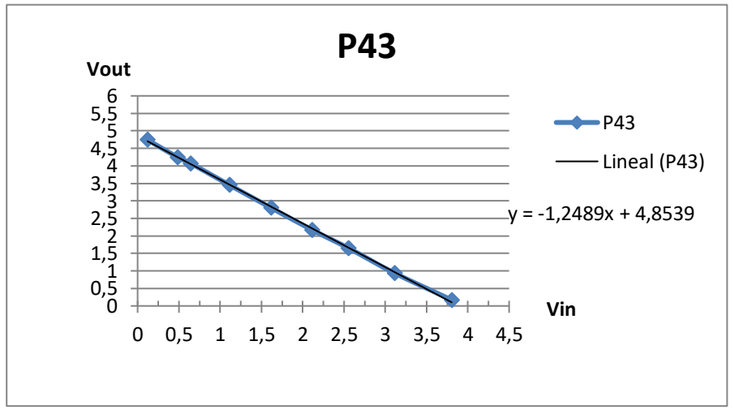
Debido a que la señal no es perfectamente estable, se miden los valores máximos y mínimos, y se hace la media

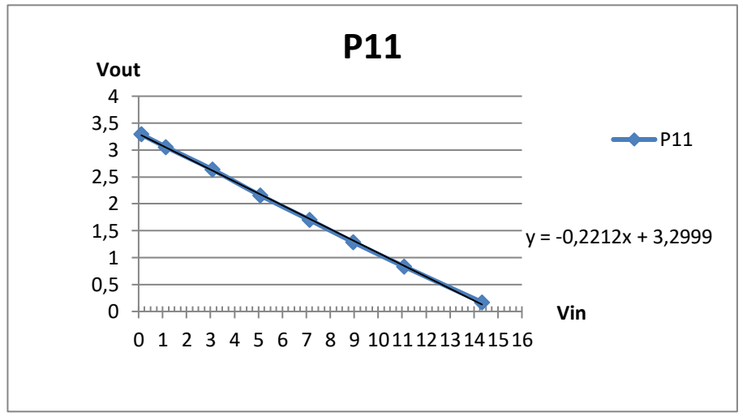
P44					
vIn(max)	**vIn(min)**	**vIn**	**vOp(max)**	**vOp(min)**	**vOp**
-0,01	0,24	**0,115**	3,37	3,56	**3,465**
0,99	1,2	1,095	3,15	3,34	3,245
2,93	3,3	3,115	2,68	2,9	2,79
4,93	5,24	5,085	2,24	2,46	2,35
6,99	7,3	7,145	1,81	2,03	1,92
8,93	9,24	9,085	1,37	1,56	1,465
9,99	10,3	10,145	1,15	1,34	1,245
11,18	11,55	11,365	0,9	1,09	0,995
14,74	15,05	**14,895**	0,12	0,31	**0,215**

P43					
vIn(max)	**vIn(min)**	**vIn**	**vOp(max)**	**vOp(min)**	**vOp**
0	0,24	**0,12**	4,65	4,84	**4,745**
0,36	0,61	0,485	4,15	4,34	4,245
0,49	0,8	0,645	3,96	4,18	4,07
0,93	1,3	1,115	3,34	3,56	3,45
1,43	1,8	1,615	2,68	2,93	2,805
1,93	2,3	2,115	2,06	2,28	2,17
2,43	2,68	2,555	1,56	1,74	1,65
2,93	3,3	3,115	0,84	1,03	0,935
3,68	3,93	**3,805**	0,06	0,28	**0,17**

P11					
vIn(max)	**vIn(min)**	**vIn**	**vOp(max)**	**vOp(min)**	**vOp**
-0,01	0,24	**0,115**	3,18	3,4	**3,29**
0,99	1,3	1,145	2,96	3,15	3,055
2,93	3,24	3,085	2,53	2,74	2,635
4,93	5,24	5,085	2,03	2,28	2,155
6,99	7,3	7,145	1,59	1,81	1,7
8,8	9,11	8,955	1,18	1,4	1,29
10,93	11,24	11,085	0,71	0,96	0,835
14,18	14,49	14,335	0,06	0,28	0,17







Ecuacion Lineal y=mx+b	vOp = -0,2197vIn + 3,4803

Ecuacion Lineal y=mx+b	vOp = -1,2489vIn + 4,8539

Ecuacion Lineal y=mx+b	vOp = -0,22212vIn + 3,2999

3 Códigos del programa

3.1 Processing

El código se divide en varios ficheros para mejor organización.

3.1.1 Variables

```
//=============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  Variables
//  OBJETIVO: Definicion de variables, clases y librerias
//=============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=============================================================================

//=============================================================================
// Librerias
//=============================================================================
import java.io.FileReader.*;
import java.io.BufferedReader.*;
import processing.serial.*;
import java.io.Writer;


//=============================================================================
// Definición de variables
//=============================================================================
//verificación conexión con Arduino
boolean falloVerif=true;
long    tiempoVerifConex=1500;

//Guarda los datos que llegan del puerto serie
String datoSerie;

//Guarda el valor de la pegatina
String pegatina="";
//Modificaciones al valor de la pegatina
String lecPeg="";

//Flag Lectura del fichero de configuración
boolean config=true;
//Ruta de los archivos de configuración
String archivoConfig="";
//Líneas del fichero de configuración
String linesConfig[];

//Imágenes
PImage inicio1;
PImage inicio2;
PImage si1;
PImage si2;
PImage no1;
PImage no2;
PImage puerto1;
PImage puerto2;
PImage lector;

//Asociadas a la grabación de la tarjeta
boolean tipoTest=false;
boolean soloGrabar=false;
boolean grabar=true;
long    tiempoGrab=0;
int     lecturasGrab=0;
long    lecGrabMax=0;
int     tabGrab=0;

//Flags grabación
boolean grabacion=true;
```

```
boolean espGrabacion=true;
boolean respGrabacion=false;
boolean falloGrabacion=false;

//Comprobación de rutas
boolean compRutas=false;

//Inicio del test
boolean inicio=true;

//Se ha recibido el resultado del test
boolean resultado=true;

//Se ha leido la pegatina
boolean respLecPegatina=false;

//Nombre del archivo a crear
String name;

//Se ha recibido el resultado de las I/O uno a uno
boolean ioDig=true;

//Asociadaas al test del zumbador
boolean zumbador=false;
boolean respZumbador=false;

//Es el primer test realizado
boolean primerTest=true;

//Asociadas a la seleccion del puerto
boolean respSelecPuerto=false;
boolean Puerto=false;
int     puertoElegido=0;
int     numeroPuertos=0;
int     puertosInicio=0;
boolean nuevoPuerto=false;

//Guarda el mensaje de fallo a mostrar
String menFallo="";
//Guarda el resultado del test
String resultadoTest="";

//Para medir tiempos
long tiempoP=0;

//Guarda si se ha pulsado la tecla 'Tab'
boolean tabulacion=true;


//================================================================================
// Clases
//================================================================================
//Guarda el puerto a usar
Serial puertoSerie;
//Objeto para crear archivos txt
creatxt archivotxt;
```

3.1.2 Verificador 195

```
//================================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  Verificador_195
//  OBJETIVO: Setup y Main (Draw, rutina bucle)
//================================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//================================================================================


//================================================================================
//  Nombre :   setup
//--------------------------------------------------------------------------------
//  Proposito: Inicialización del programa
//================================================================================
void setup()
{
```

```
    //Colocar el foco en la ventana
    surface.setVisible(true);
    //Tamaño de la ventana
    size(800, 400);
    //Carga las imágenes de los botones
    inicio1  = loadImage("inicio1.png");
    inicio2  = loadImage("inicio2.png");
    si1      = loadImage("si1.png");
    si2      = loadImage("si2.png");
    no1      = loadImage("no1.png");
    no2      = loadImage("no2.png");
    puerto1  = loadImage("puerto1.png");
    puerto2  = loadImage("puerto2.png");
    lector   = loadImage("lector1.jpg");
}


//============================================================================
//  Nombre :   draw
//----------------------------------------------------------------------------
//  Propósito: Rutina bucle
//============================================================================
void draw()
{
//Selección del tipo de test
if(!tipoTest)
{
    pantTipoTest();
}

//Lectura del fichero de configuración
else if (config)
{
    config = false;
    //Función para leer el fichero
    leerConfig();
    //variables numéricas
    tiempoGrab = int(linesConfig[27]);
    lecGrabMax = int(linesConfig[28])/tiempoGrab;
}

//Comprueba si las rutas son accesibles
else if (!compRutas)
{
    comprobarRuta();
}



//Selección del puerto
else if (!respSelecPuerto)
{
    numeroPuertos=Serial.list().length;
    pantSelecPuerto();
}

//Se ha seleccionado el puerto
else if (!Puerto)
{
    Puerto=true;
    //Inicia la comunicación serie
    String portName = Serial.list()[puertoElegido];
    puertoSerie = new Serial(this, portName, 9600);
}

//Comprueba la comunicación con Arduino
else if (((millis()-tiempoP) < tiempoVerifConex) && (respSelecPuerto) &&
(!respLecPegatina))
{
    background(230);
    pantVerifConex();
}

//Si no se ha podido comunicar con Arduino
else if (falloVerif)
{
```

```
    pantFalloConex();
}

//Espera a que se lea la pegatina
else if (!respLecPegatina)
{
    pantLecPegatina();
}

//Grabacion ok. Inicia el test
else if (inicio)
{
    //Inicializa variables
    resultado = true;
    zumbador = false;
    puertoSerie.clear();
    pantInicio();
}

//Se va a grabar la tarjeta
else if ((!grabacion) && (grabar))
{
    grabacion=true;
    //Carga la ruta del .log
    String rutaLog = linesConfig[24];
    //Borra el contenido del .log
    PrintWriter salidaLog;
    salidaLog = createWriter(rutaLog);
    salidaLog.close();
    String rutaBat = linesConfig[23];
    //Lanza el .bat
    launch(rutaBat);
    //Tiempo de inicio de la grabación
    tiempoP = millis();
    lecturasGrab = 0;
}

//Espera mientras se graba la tarjeta
else if ((!espGrabacion) && (grabar))
{
    pantEspGrabacion();
}
else
{
    //Espera a que se realice el test del zumbador
    if (!respZumbador)
    {
        pantEspera();
        zumbador=false;
        tabulacion=true;
    }

    //Test del zumbador - manual
else if (zumbador)
{
    background(230);
    pantSiNo();
}

//Recepción de datos de arduino
if (puertoSerie.available() > 0)
{
    //Lee el puerto serie hasta detectar '*'
    datoSerie = puertoSerie.readStringUntil('*');
    delay(50);
    //Descarta datos nulos
    if (datoSerie!=null)
    {
        //Recibe que se está realizando test del zumbador
        if (datoSerie.equals("zumbador*") && (!zumbador ))
        {
            //Zumbador. automático o manual
            if(linesConfig[26].equals("auto"))
            {
                //Envía a Arduino, zumbador automático
                puertoSerie.write('z');
```

```
            respZumbador = false;
            zumbador = false;
        }
        else
        {
            //Envía a Arduino, zumbador manual
            puertoSerie.write('x');
            zumbador = true;
            respZumbador = true;
        }
    }

    //Recibe que las tensiones de alimentación son correctas
    else if (datoSerie.equals("vOK*"))
    {
        //no se grabarán las tarjetas
        if(!grabar)
        {
            //Envía a Arduino, solo verificar
            puertoSerie.write('d');
        }
        else
        {
            grabacion = false;
            espGrabacion = false;
        }
    }

    //Recibe el resultado final del test
    else if (resultado)
    {
         respZumbador = true;
        //Test fallido
        if (datoSerie.equals("fail*"))
        {
            //carga la ruta FAIL
            String pathFail2 = linesConfig[21];
            //genera archivo de texto
            archivotxt = new creatxt(pathFail2+name+"-FAIL.txt", pathFail2);
            resultadoTest = "FAIL";
            menFallo = "si";
        }
        else
        {
            //carga la ruta OK
            String pathOk2 = linesConfig[22];
            archivotxt = new creatxt(pathOk2+name+".txt", pathOk2);
            resultadoTest = "OK";
            menFallo = "";
        }
        //Desactiva para no entrar nuevamente
        resultado = false;
    }

    //Recibe los resultados de las I/O digitales
    else if (ioDig)
    {
        //Si ha habido fallo
        if (resultadoTest.equals("FAIL"))
        {
            //Busca cual ha sido el fallo
            mensajeFallo();
        }
        //Desactiva para no entrar nuevamente
        ioDig=false;
    }

    //Recibe los resultados de cada test
    else
    {
        //Si el fallo ha sido en las ioDig
        if (menFallo.equals("si"))
        {
            //Busca cual ha sido el fallo
            mensajeFallo();
        }
```

```
//Guarda el valor leido en la pegatina
if (!(pegatina.equals("")))
{
    lecPeg=pegatina;
}

//Carga la fecha y la hora
String tiempo= FechaHora();

//Concatena: fecha,hora,lecturapegatina y vectorderesultados
datoSerie=tiempo+pegatina+";"+datoSerie;

//Borra el '*'
archivotxt.write(datoSerie.substring(0, datoSerie.indexOf('*', 0)));

//Cierra el fichero
archivotxt.close();

//vuelve a inicio
inicio = true;
respLecPegatina = false;
ioDig = true;
grabacion = true;
                    espGrabacion = true;
                    pegatina = "";
                }//recepción de datos
            }//datoSerie!=null
        }//puerto serie available
    }//inicia el test
}//draw


//============================================================================================
//  Nombre :       leerConfig
//--------------------------------------------------------------------------------------------
//  Propósito:     lee el fichero de configruación y guarda las líneas
//============================================================================================
void leerConfig()
{
    //Lee el fichero de configuracion correspondiente al tipo de test seleccionado
    if(soloGrabar)
    {
        archivoConfig = "C:\\ARAELEC\\config_INV.txt";
    }
    else
    {
        archivoConfig = "C:\\ARAELEC\\config.txt";
    }

    //Leer archivo
    try
    {
        //Abre el fichero
        BufferedReader br= new BufferedReader (new java.io.FileReader(archivoConfig));
        //Carga las lineas del fichero
        linesConfig = loadStrings(archivoConfig);
        br.close();
    }
    catch (IOException e)
    {
        println(e);
    }
    //Elimina el número de la línea y la guarda
    for(int m=21;m<=28;m++)
    {
        linesConfig[m]=linesConfig[m].substring(2,linesConfig[m].length());
    }
}
```

```
//=============================================================================================
//  Nombre :   FechaHora
//---------------------------------------------------------------------------------------------
//  Propósito: concatena en una sola cadena, la fecha y la hora
//             con los correspondientes caracteres de separación
//=============================================================================================
String FechaHora()
{
    String tiempo = day()+"/"+month()+"/"+year()+";"+hour()+":"+minute()+":"+second()+";";
    return tiempo;
}
```

3.1.3 CompRuta

```
//=============================================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  CompRuta
//  OBJETIVO: Revisión de rutas
//=============================================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=============================================================================================

//=============================================================================================
//  Nombre :   comprobarRuta
//---------------------------------------------------------------------------------------------
//  Propósito: Comprobar si los paths existen o son accesibles
//=============================================================================================
void comprobarRuta()
{
    //se comprueban 4 rutas
    int rutas[]={0,0,0,0};
    boolean fallo=false;

    //Separa ruta y nombre del .log
    String rutaLog=linesConfig[24];
    linesConfig[24]=linesConfig[24].substring(0, linesConfig[24].lastIndexOf('\\')+1);

    //comprueba las rutas
    for(int n=0;n<=3;n++)
    {
          File directorio = new File(linesConfig[n+21]);
          //Si existe
          if(directorio.exists())
          {
              rutas[n]=1;
              if(n==3)
              {
                  //Vuelve a juntar ruta+nombre del .log
                  linesConfig[24]=rutaLog;
              }
          }
          else
          {
              rutas[n]=0;
              fallo=true;
          }
    }//fin For

    //Si hay fallo muestra la pantalla de error
    if(fallo==true)
    {
        compRutas=false;
        pantFalloRed(rutas);
    }
    else
    {
        compRutas=true;
    }
}
```

3.1.4 LecturaLog

```
//=============================================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  LecturaLog
//  OBJETIVO: Lectura del fichero de grabación .log
//=============================================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=============================================================================================


//=============================================================================================
//  Nombre :       lecturaLog
//---------------------------------------------------------------------------------------------
//  Propósito:     Lee el fichero Result.log generado por en la programacion
//                 de la tarjeta para saber si se ha grabado o no
//=============================================================================================
void lecturaLog()
{
    //Mensaje de espera
    background(230);
    textSize(25);
    textAlign(CENTER);
    fill(23,64,128);
    text("GRABACIÓN DE TARJETA",400,150);
    textSize(20);
    text("GRABANDO",400,200);
    text("la tarjeta emitirá un pitido al terminar la grabación", 400,300);

    //Si no ha excedido el número de comprobaciones
    if(lecturasGrab<=lecGrabMax)
    {
        //Lee el archivo .log
        try
        {
            //Carga la ruta
            String rutaLog=linesConfig[24];
            //Abre el fichero
            BufferedReader br2= new BufferedReader (new java.io.FileReader(rutaLog));
            //Carga las líneas del fichero
            String[] lines = loadStrings(rutaLog);
            //Si el fichero de resultados no está vacío
            if(lines.length>10)
            {
                  //Lee las líneas donde están los resultados de la grabación
                  //Si son las correctas, ha terminado la grabación sin problemas
                  if((lines[lines.length-1].equals("<<< Verify OPTION BYTE succeeds")) &&
                    (lines[lines.length-9].equals("<<< Verifying PROGRAM MEMORY succeeds")) )
                  {
                      //Finaliza la grabación
                      espGrabacion=true;
                      falloGrabacion=false;
                      background(230);
                      //Envía el resutlado de la grabación a arduino
                      puertoSerie.write('s');
                      pantEspera();
                  }
                  //Reinicia para realizar otra comprobación
                  else
                  {
                      tiempoP=millis();
                  }
            }
            //Si no son las correctas
            //Espera nuevamente y vuelve a leer el fichero (lee hasta 'lecGrbMax' veces)
            else
            {
                tiempoP=millis();
            }
            br2.close();
        }
        catch (IOException e)
        {
            println(e);
        }
    }
```

```
    //Si despues de 'lecGrbMax' lecturas, las líneas no coinciden, hay fallo
    //Cierra el fichero
    else
    {
       falloGrabacion=true;
       espGrabacion=true;
       falloGrabacion=false;
       //Envía el resutlado de la grabación a arduino (fallo)
       puertoSerie.write('n');
    }
}
```

3.1.5 MsjFallo

```
//============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  MsjFallo
//  OBJETIVO: Selección del mensaje de fallo
//============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//============================================================================
//============================================================================
//  Nombre :   mensajeFallo
//----------------------------------------------------------------------------
//  Propósito: Busca el test que ha fallado para elegir el mensaje a mostrar
//============================================================================
void mensajeFallo()
{
    int j=0;
    boolean indice = true;

    //Revisa si el error está en las I/O uno a uno
    if(ioDig)
    {
        //Inicializa las variables
        menFallo="";
        ioDig=false;

        //Separa los resultados del test en strings individuales
        String[] datoArray = split(datoSerie,';');

        //Solo si error ha ocurrido en las I/O uno a uno
        if((datoArray[0].charAt(0))!='x')
        {
            //Busca la salida con error
            for(int i=0; i<=7; i++)
            {
                //"11" indica que no hay error en la salida
                if(!datoArray[i].equals("11"))
                {
                    //En la pantalla se mostrarán las salidas con error
                    if(i==7)
                        menFallo=menFallo+" P4.2";
                    else
                        menFallo=menFallo+" P3."+(i+2);
                }
            }
        }
        else
        {
            menFallo="si";
        }
    }
    //El error está en un test anterior
    else
    {
        //Separa los resultados del test en strings individuales
        String[] datoArray = split(datoSerie,';');
        //Cuando un test falla, los test siguientes muestran 'x'
        //Busca ese caracter y guarda la posicion anterior
        while(indice)
        {
            if((datoArray[j].charAt(0))=='x')
```

```
        {
            indice=false;
        }
        else
        {
            j++;
        }
    }

    //Mensaje de fallo
    switch(j)
    {
        case 0:
          menFallo="Fallo 1 - no hay alimentación";
          break;
        case 1:
          menFallo="Fallo 2 - hay "+datoArray[j-1]+"V en P1.1 en lugar de 5 +-0.5 V";
          break;
        case 2:
          menFallo="Fallo 3 - hay "+datoArray[j-1]+"V en P4.4 en lugar de11.4 +-0.5 V";
          break;
        case 3:
          menFallo="Fallo 4 - Fallo en la grabación";
          break;
        case 4:
          menFallo="Fallo 5 - no se activó el zumbador";
          break;
        case 5:
          menFallo="Fallo 6 - hay "+datoArray[j-1]+"V en P4.3 en lugar de <0.4V";
          break;
        case 6:
          menFallo="Fallo 7 - PWM incorrecto en P4.1";
          break;
        case 7:
          menFallo="Fallo 8 - no se desactivó P4.1";
          break;
      }//switch
  }//else
}//void
```

3.1.6 Pantallas

```
//=============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  Pantallas
//  OBJETIVO: Pantallas de la aplicación
//=============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=============================================================================


//-----------------------------------------------------------------------------
//  Nombre :    panTipoTest
//-----------------------------------------------------------------------------
//  Propósito:  Seleccionar tipo de test: grabar y verificar, solo grabar, solo verificar
//=============================================================================
void pantTipoTest()
{
    //Coordenadas de botones
    int xb1=500;
    int anchox=100;
    int yb1=100;
    int anchoy=50;
    int sepy=100;

    //Mensajes
    background(230);
    textSize(25);
    textAlign(CENTER);
    fill(23,64,128);
    text("ELIJA UNA OPCIÓN SEGÚN REFERENCIA", 400, 40);
    text("GRABAR Y VERIFICAR", 300, 130);
    text("SOLO GRABAR",        300, 230);
    text("SOLO VERIFICAR",     300, 330);
    textAlign(CENTER);
```

```
//Detección posicionamiento del cursor
//Opcion 1 - grabar y verificar
if( (mouseX > xb1) && (mouseX < xb1+anchox) && (mouseY > yb1) &&  (mouseY < yb1+anchoy) )
{
      //Ubicacion de botones
      image(puerto2, xb1, yb1);
      image(puerto1, xb1, yb1+sepy);
      image(puerto1, xb1, yb1+2*sepy);
      //Se ha pulsado el ratón
      if(mousePressed==true)
      {
          grabar=true;
          soloGrabar=false;
          tipoTest=true;
      }
      //Selección mediante el teclado
      tabGrab=0;
 }
 //Opcion 2 - solo grabar
  else if((mouseX > xb1) && (mouseX < xb1+anchox) && (mouseY > yb1+sepy)   && (mouseY <
  yb1+sepy+anchoy))
 {
       //Ubicacion de botones
       image(puerto1, xb1, yb1);
       image(puerto2, xb1, yb1+sepy);
       image(puerto1, xb1, yb1+2*sepy);
       if(mousePressed)
       {
           grabar=true;
           soloGrabar=true;
           tipoTest=true;
       }
       //Selección mediante el teclado
       tabGrab=1;
 }
 //Opcion 3 - solo verificar
else if((mouseX > xb1) && (mouseX < xb1+anchox) && (mouseY > yb1+2*sepy)   && (mouseY <
yb1+2*sepy+anchoy))
 {
   //Ubicacion de botones
     image(puerto1, xb1, yb1);
     image(puerto1, xb1, yb1+sepy);
     image(puerto2, xb1, yb1+2*sepy);
     if(mousePressed)
     {
         grabar=false;
         soloGrabar=false;
         tipoTest=true;
     }
     //Selección mediante el teclado
     tabGrab=2;
 }
      else
      {
            //Navega por las opciones con el teclado (tab y arrows)
            switch (tabGrab)
            {
                case 0:
                    image(puerto2, xb1, yb1);
                    image(puerto1, xb1, yb1+sepy);
                    image(puerto1, xb1, yb1+2*sepy);
                    break;
                case 1:
                    image(puerto1, xb1, yb1);
                    image(puerto2, xb1, yb1+sepy);
                    image(puerto1, xb1, yb1+2*sepy);
                    break;
                case 2:
                    image(puerto1, xb1, yb1);
                    image(puerto1, xb1, yb1+sepy);
                    image(puerto2, xb1, yb1+2*sepy);
                    break;
            }//switch
      }//else
}//void
```

```
//=============================================================================================
//  Nombre :    pantFalloRed
//---------------------------------------------------------------------------------------------
//  Propósito:  Muestra fallo de red y conteo atras de 10 segundos para cerrar el programa
//=============================================================================================
void pantFalloRed(int[] rutasFail)
{
    //Mensajes
    background(230);
    textSize(30);
    textAlign(CENTER);
    //Color rojo
    fill(200,0,0);
    text("FALLO DE RED",400,50);
    textSize(25);
    //Color normal
    fill(23,64,128);
    text("NO SE PUEDE ACCEDER A LA RUTA(S)",400,100);
    text("FALLO EN EL FICHERO DE CONFIGURACIÓN",400,130);

    //Imprime las líneas que fallan
    textSize(20);
    for(int i=0; i<=3; i++)
    {
        if(rutasFail[i]==0)
        {
            text("LÍNEA"+(i+1), 400 , i*25+180);
        }
    }

    //Para cerrar el programa en 10 segundos en caso de error
    textSize(25);
    text("EL PROGRAMA SE CERRARÁ EN: ",400,300);
    textSize(20);
    int tiempo=millis()/1000;
    tiempo=10-tiempo;
    text(tiempo, 400,350);
    if (tiempo==0)
    {
        exit();
    }
}


//=============================================================================================
//  Nombre :        pantSelecPuerto
//---------------------------------------------------------------------------------------------
//  Propósito:      Espera a detectar la conexión de un nuevo puerto
//=============================================================================================
void pantSelecPuerto()
{
    //Mensajes
    background(230);
    textSize(30);
    textAlign(CENTER);
    fill(23,64,128);
    text("CONECTE EL CABLE USB-1",400,200);

    //Guarda el número de puertos conectados al principio
    if(!nuevoPuerto)
    {
        puertosInicio=numeroPuertos;
        nuevoPuerto=true;
    }

    //Se ha conectado el USB de Arduino (hay un nuevo puerto)
    if(numeroPuertos!=puertosInicio)
    {
        //Para verificar el puerto durante 1 segundo
        tiempoP=millis();
        //Elige el último puerto conectado (la lista empieza desde el cero)
        puertoElegido=puertosInicio;
        //Para no volver a entrar
        respSelecPuerto=true;
    }
}
```

```
//=============================================================================
//  Nombre :    pantVerifConex
//-----------------------------------------------------------------------------
//  Propósito:  Verifica si el puerto elegido es el correcto
//=============================================================================
void pantVerifConex()
{
    //Mensajes
    textSize(30);
    textAlign(CENTER);
    fill(23,64,128);
    text("...VERIFICANDO PUERTO...",400,150);
    textSize(30);
    text("MEGA2560",400,200);
    //Número de pegatina
    text(pegatina,400, 250);

    //Espera a que arduino le envíe un carácter
    if ( puertoSerie.available() > 0)
    {
        int dato=puertoSerie.read();
        if(dato=='k')
        {
            //Se ha recibido dato de arduino
            //Envía respuesta
            if(soloGrabar)
            {
                puertoSerie.write('v');
            }
            else
            {
                puertoSerie.write('c');
            }
            falloVerif=false;
        }
    }
}


//=============================================================================
//  Nombre :    pantFalloConex
//-----------------------------------------------------------------------------
//  Propósito:  mensaje de fallo si no hay conexión con arduino
//=============================================================================
void pantFalloConex()
{
    //Mensajes
    background(230);
    textSize(30);
    textAlign(CENTER);
    fill(250,0,0);
    text("NO SE PUEDE CONECTAR CON ARDUINO",400,200);
    textSize(25);
    fill(23,64,128);
    text("compruebe que el cable esté conectado \n y reinicie el programa",400,250);
    //Número de pegatina
    text(pegatina,400, 250);
}


//=============================================================================
//  Nombre :    pantInicio
//-----------------------------------------------------------------------------
//  Propósito:  Muestra el botón para iniciar el test
//=============================================================================
void pantInicio()
{
    //Mensajes
    background(230);
    textSize(25);
    textAlign(CENTER);
    fill(23,64,128);
    text("NÚMERO DE PEGATINA",400,100);
    textSize(22);
```

```
    text(pegatina, 400, 135);
    textSize(20);
    text("PRESIONE EL BOTÓN O LA TECLA 'INTRO' PARA INICIAR",400,210);

    //El cursor está sobre el botón START
    if((mouseX > 325) && (mouseX < 475) && (mouseY > 230)  && (mouseY < 380))
    {
        //Imagen botón pulsado
        image(inicio2, 325, 230);
        //se ha pulsado el botón
        if(mousePressed)
        {
            inicio=false;
            primerTest=false;
            //Envía la señal a Arduino para medir tensiones
            puertoSerie.write('a');
        }
    }
    else
    {
        //Imagen botón no pulsado
        image(inicio1, 325, 230);
    }
}

//================================================================================
//  Nombre :     pantLecPegatina
//--------------------------------------------------------------------------------
//  Propósito:   Espera hasta que se lea la pegatina - muestra el resultado del test
//================================================================================
void pantLecPegatina()
{
    //Mensajes
    background(230);
    textSize(25);
    fill(23, 64, 128);
    textAlign(CENTER);

    //Si el foco no está en la aplicación
    if(!focused)
    {
        textSize(30);
        fill(0);
        text("ACTIVE LA VENTANA PARA \n PODER LEER LA PEGATINA",400 ,200);
    }

    //Si es el primer test solo muestra para leer la pegatina
    else if(primerTest)
    {
        text("PRIMER TEST",400 ,60);
        text("LEER PEGATINA",400 ,140);
        image(lector,250 ,140);
    }
    //Si hay fallo en la grabación
    else if(falloGrabacion)
    {
        fill(250, 0, 0);
        text("GRABACION FALLIDA" , 400, 30);
        text("COMPRUEBE QUE EL VERIFICADOR ESTÁ ENCENDIDO",400,70);
        fill(23,64,128);
        text("VERIFICAR NUEVA TARJETA - LEER PEGATINA", 400, 135);
        image(lector, 250, 140);
    }
    //Si no es el primer test, muestra el resultado del test anterior
    else
    {
        if(resultadoTest=="OK")
        {
            //OK color verde
            textSize(20);
            text(lecPeg, 400, 50);
            textSize(25);
            textAlign(CENTER);
            fill(0, 200, 0);
            text("TEST OK", 400, 80);
```

```
        }
        else if(resultadoTest=="FAIL")
        {
            //FAIL y mensaje de error en color rojo
            textSize(20);
            text(lecPeg, 400, 30);
            textSize(25);
            fill(250, 0, 0);
            text("TEST FALLIDO" , 400, 60);
            //Imprime el código del fallo
            text(menFallo, 400, 90);
        }
        fill(23,64,128);
        text("VERIFICAR NUEVA TARJETA - LEER PEGATINA", 400, 135);
        image(lector, 250, 140);
    }
}


//=============================================================================
//  Nombre :    pantEspera
//-----------------------------------------------------------------------------
//  Propósito:  Espera mientras Arduino va realizando el test
//=============================================================================
void pantEspera()
{
    //Mensajes
    background(230);
    textSize(30);
    textAlign(CENTER);
    fill(23,64,128);
    text("...REALIZANDO TEST...",400,150);
    textSize(30);
    text("Tarjeta número",400,200);
    text(pegatina,400, 250);
}


//=============================================================================
//  Nombre :    pantSiNo
//-----------------------------------------------------------------------------
//  Propósito:  Muestra botones 'SI' o 'NO' cuando la comprobacion del zumbador
//              se realiza de forma manual
//=============================================================================
void pantSiNo()
{
    //Mensajes
    textSize(25);
    textAlign(CENTER);
    fill(23,64,128);
    //Deteccion manual del zumbador
    text("¿HA ESCUCHADO PITIDOS?", 400, 40);

    //Seleccion de respuesta con el teclado
    if(tabulacion)
    {
        image(si2, 100, 150);
        image(no1, 500, 150);
    }
    else
    {
        image(si1, 100, 150);
        image(no2, 500, 150);
    }

    //Cursor sobre el boton 'si'
    if((mouseX > 100) && (mouseX < 300) && (mouseY > 150)  && (mouseY < 300))
    {
        //No hace caso al teclado
        tabulacion=true;
        //Se hace clic en 'SI'
        if(mousePressed)
        {
            respZumbador=false;
            zumbador=false;
            //Envía la respuesta a Arduino.
```

```
            puertoSerie.write('s');
          }
       }

      //Cursor sobre el boton 'no'
      else if((mouseX > 500) && (mouseX < 700) && (mouseY > 150)  && (mouseY < 300))
      {
          //No hace caso al teclado
          tabulacion=false;
          //Se hace clic en 'NO'
          if(mousePressed)
          {
              respZumbador=false;
              zumbador=false;
              //Envía la respuesta a Arduino
              puertoSerie.write('n');
          }
      }
}


//============================================================================
//  Nombre :    pantEspGrabacion
//----------------------------------------------------------------------------
//  Propósito:  Espera mientras se graba la tarjeta
//============================================================================
void pantEspGrabacion()
{
    //Mensajes
    background(230);
    textSize(25);
    textAlign(CENTER);
    fill(23,64,128);
    text("GRABACIÓN DE TARJETA",400,150);
    textSize(20);
    text("GRABANDO",400,200);
    text("la tarjeta emitirá un pitido al terminar la grabación", 400,300);

    //Lee el resultado de la grabación cada 'tiempoGrab'segundos.
    if(millis()-tiempoP >tiempoGrab)
    {
        background(230);
        //Número de veces que ha leido el fichero
        lecturasGrab++;
        //Función para leer el fichero
        lecturaLog();
    }
}
```

3.1.7 Creatxt

```
//============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  creatxt
//  OBJETIVO: Objeto para la gestión de ficheros de texto
//============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//============================================================================

//============================================================================
//  Nombre (clase): creatxt
//----------------------------------------------------------------------------
//  Propósito: Define las funciones para la creación y modificacion de ficheros .txt
//============================================================================
class creatxt
{
      private PrintWriter output;      //Permite la creación de archivos
      private String      fileName;    //Nombre del archivo

      //Constructor
      //------------------------------------------------------------------------

      creatxt(String fileName, String ruta)
```

```
	{
		//Nombre del archivo
		this.fileName=fileName;
		//Comprueba si existe el archivo
		if (exist(this.fileName, ruta))
		{
			//Si ya existe, abre el archivo
			Continue();
		}
		else
		{
			//Si no existe, se crea un nuevo archivo
			output= createWriter(this.fileName);
			//Imprime la cabecera del fichero de resultados
			output.println("Fecha;Hora;Test;P1-1;P4-4;grabacion;zumbador;P4-3;P4-1;P4-
1;P3-2;P3-3;P3-4;P3-5;P3-6;P3-7;P3-8;P4-2");

	output.println("dd/mm/aaaa;hh:mm:ss;Tarjeta;VDD;VR;grabacion;sonido;RAN;ANP;ANP;MAC;PI
R;PDR;ANR;IIR;IDR;LFR;ANR-R0");
		}
	}

	//=========================================================================
	//  Nombre:    exist
	//-------------------------------------------------------------------------
	//  Propósito: Comprueba si ya existe un archivo con el nombre especificado
	//=========================================================================
	private boolean exist(String fileName, String Ruta)
	{
		//Carga la lista de archivos
		String[] filenames = listFileNames(Ruta);
		//Comprueba si el nombre coincide
		for (int x=0; x<=filenames.length-1;x++)
		{
			if ((fileName.equals(Ruta+filenames[x])) )
			{
				return true;
			}
		}
		return false;
	}

	//=========================================================================
	//  Nombre: listFileNames
	//-------------------------------------------------------------------------
	//  Propósito: Crea una lista de los archivos en el directorio
	//=========================================================================
	private String[] listFileNames(String dir)
	{
		//Directorios OK y FAIL
		File file1 = new File(dir);

		//Comprueba que sea un directorio y no un archivo
		if (file1.isDirectory())
		{
			//Carga la lista de archivos de ambos directorios
			String names1[] = file1.list();

			//Regresa names
			return names1;
		}
		else
		{
			return null;
		}
	}

	//=========================================================================
	//  Nombre: Continue
	//-------------------------------------------------------------------------
	//  Propósito: Lee el fichero existente para poder continuar escribiendo en él
	//=========================================================================
	private void  Continue()
	{
		try
		{
```

```
                //Abre el fichero
                BufferedReader br= new BufferedReader (new java.io.FileReader(fileName));
                //Carga las lineas del fichero
                String[] lines = loadStrings(fileName);
                //Crea nuevamente el fichero par reescribirlo
                output= createWriter(this.fileName);

                //Reescribe el fichero
                for (int j = 0; j < lines.length; j++)
                {
                        output.println(lines[j]);
                }

        //El fichero se deja abierto para poder seguir escribiendo en él
        }
        catch (IOException e)
        {
                println(e);
        }
}

//============================================================================
//  Nombre: close
//----------------------------------------------------------------------------
//  Propósito: Cierra el archivo para que pueda ejecutarse
//============================================================================
public void close()
{
        //Vacía buffer de escritura
        output.flush();
        //Cierra el archivo
        output.close();
}

//============================================================================
//  Nombre: write
//----------------------------------------------------------------------------
//  Propósito: Escribe los datos en el archivo
//  Recibe:    Los datos a escribir
//============================================================================
public void write(String data)
{
        output.print(data);
}

}//fin  clase creatxt
```

3.1.8 keyPressed

```
//============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  keyPressed
//  OBJETIVO: Detección del teclado
//============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//============================================================================


void keyPressed()
{
    //Tipo de grabación. grabar y verificar, solo grabar, solo verificar
    if(!tipoTest)
    {
        //Navega entre las opciones
        //Cambia a opcion siguiente con: 'tab', 'arrow down', 'arrow right'
        if ((key=='\t') || (keyCode==40) || (keyCode==39))
        {
            //Cambia de puerto
            if(tabGrab<2)
            {
                tabGrab++;
            }
            else
            {
                tabGrab=0;
            }
        }
        //Cambia a opción anterior con: 'arrow up', 'arrow left'
```

```
else if ((keyCode==38) || (keyCode==37))
{
    if(tabGrab>0)
    {
        tabGrab--;
    }
    else
    {
        tabGrab=2;
    }
}
//Si se pulsa 'intro'
else if (key=='\n')
{
    //Selección de la opción
    switch (tabGrab)
    {
        //Grabar y verificar
        case 0:
            grabar=true;
            soloGrabar=false;
            break;
        //Solo grabar
        case 1:
            grabar=true;
            soloGrabar=true;
            break;
        //Solo verificar
        case 2:
            grabar=false;
            soloGrabar=false;
        }
        tipoTest=true;
    }
}

//Lectura pegatina
else if((!respLecPegatina) && (tipoTest))
{
    //'Intro' es el último caracter que envía la pistola
    if ((key=='\n') && (pegatina.length()>10))
    {
        respLecPegatina=true;
        //Selecciona el número de orden de la pegatina
        name= pegatina.substring(0,  6 );
    }
    else if (key!='\n')
    {
        //Evita caracteres indeseados
        if(key!=CODED)
        {
            //Va guardando el valor de la pegatina
            pegatina=pegatina+key;
        }
    }
}

//Seleccion de inicio con la tecla 'intro'
else if((inicio) && (respLecPegatina))
{
    //Selección con intro
    if (key=='\n')
    {
        inicio=false;
        primerTest=false;
        //Envío a Arduino
        puertoSerie.write('a');
    }
}

//Seleccion zumbador
else if( (zumbador) && (tipoTest) )
{
    //Si se pulsa 'Tab', 'flecha izq' o 'flecha der'
    if ((key=='\t') || (keyCode==39) || (keyCode==37))
    {
        //Cambia de opción
        tabulacion=!tabulacion;
    }
    //Opción 'SI'
    if(tabulacion)
    {
        //Botón 'SI' iluminado
        image(si2, 100, 150);
        image(no1, 500, 150);
        //Se elige la opción con 'intro'
        if (key=='\n')
        {
            zumbador=false;
            respZumbador=false;
            //Envía respuesta a Arduino
            puertoSerie.write('s');
```

```
            }
        }
        //Opción 'NO'
        else
        {
            //Botón 'NO' iluminado
            image(si1, 100, 150);
            image(no2, 500, 150);
            //Se elige la opción con 'intro'
            if (key=='\n')
            {
                zumbador=false;
                respZumbador=false;
                //Envía respuesta a Arduino
                puertoSerie.write('n');
            }
        }
    }
}
```

3.2 Arduino

Se divide el código de Arduino en ficheros.

3.2.1 Verificador_195_arduino

```
//=============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  Verificador_195_arduino
//  OBJETIVO: Setup y Main (Loop, rutina bucle)
//=============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=============================================================================

//=============================================================================
// Librerias
//=============================================================================
#include <TimerThree.h>
#include <TimerOne.h>

//=============================================================================
// Definición de constantes
//=============================================================================

//PXY -> Puerto X pin Y
//SALIDAS DE LA TARJETA     ENTRADAS DE ARDUINO

//PUERTO 3 TARJETA  (PIN 14 - PIN 20)
const int P32 = 20;     //MAR   Marcha atrás
const int P33 = 19;     //PIR   Posición izq
const int P34 = 18;     //PDR   Posición der
const int P35 = 17;     //ANR   Antiniebla
const int P36 = 16;     //IIR   Interior izq
const int P37 = 15;     //IDR   Interior der
const int P38 = 14;     //LFR   Freno

//PUERTO 1 TARJETA
const int P11 = 'A2';   //VDD   Alimentación del micro (5V)

//PUERTO 4 TARJETA
const int P41 = 2;      //ANP    Interrupción 0
const int P42 = 3;      //ANR-R0
const int P43 = 'A1';   //RAN    (0,4V)
const int P44 = 'A0';   //VR     (11,4V)

const int audio = 'A3'; //entrada del micrófono

//ENTRADAS DE LA TARJETA    SALIDAS DE ARDUINO

const int modoTest = 9;
//PUERTO 3 TARJETA
const int P31 = 8;      //DR    Detección de remolque
//PUERTO 1 TARJETA
```

```
const int P14 = 4;       //Reset
//CONTROL DEL PUERTO 5
const int VP5 =   5;     //Tensión que se aplicará al puerto 5   Timer 3
const int releA = 6;     //Relé de control A (ver tabla de estados Puerto 5)
const int releB = 7;     //Relé de control B (ver tabla de estados Puerto 5)

/* Tabla de estados Puerto 5

  ---------------------------------------------------------------
     rele       ||            Puerto 5                         |
  ---------------------------------------------------------------
    A       B    ||  1   |  2  |  3  |  4  |  5  |  6   |  7   |
  ---------------------------------------------------------------
    0       0    ||  L   | VP5 | VP5 | VP5 |  L  |  L   |  L   |
  ---------------------------------------------------------------
    0       1    ||  L   | VP5 |  L  | VP5 |  L  | VP5  |  L   |
  ---------------------------------------------------------------
    1       0    || VP5  |  L  | VP5 |  L  | VP5 |  L   | VP5  |
  ---------------------------------------------------------------
    1       1    ||                NO SE USA                    |
  ---------------------------------------------------------------
*/
```

```
//=============================================================================
// Definición de variables
//=============================================================================
//se almacenan los resultados de los tests
String  resultado[16];

//se almacenan los resultados de las I/O uno a uno
String  ioDig[8];

//Asociadas al PWM
long    tiempoSemiciclo;          //guarda el tiempo en alto/bajo del PWM
int     numSemiciclo    = 0;      //guarda el numero de semicilos del PWM
int     errorSemiciclo  = 0;      //guarda el número de semiciclos malos del PWM
boolean falloPwm        = false;  //guada el resultado del test PWM
boolean primerTest      = true;
boolean zumb            = false;
boolean soloGrabar = false;

unsigned long tiempo    = 0;
//Límites de tensión
const float limiteInf5v   = 4.5;  //4.5
const float limiteSup5v   = 5.5;  //5.5
const float limiteInf12v  = 10.9;  //10.9
const float limiteSup12v  = 11.9; //11.9
const float limiteSup0v   = 0.4;  //0.4


//tensiones de referencia de los operacionales
//P44   vOp=-0.2197*vIn+3.4803    con vIn=(0.115 , 14.895)
//P43   vOp=-1.2481*vIn+4.8519    con vIn=(0.115 , 3.805)
//P11   vOp=-0.2212*vIn+3.2999    con vIn=(0.115 , 14.335)

const float P11mRecta = -0.2212;
const float P11bRecta = 3.2999;
const float P43mRecta = -1.2489;
const float P43bRecta = 4.8519;
const float P44mRecta = -0.2197;
const float P44bRecta = 3.4803;


//=============================================================================
//  Nombre :     setup
//-----------------------------------------------------------------------------
//  Proposito:   Inicia la comunicación serie, configura entradas y salidas.
//=============================================================================
void setup()
{
  //Inicio comunicación serie
  Serial.begin(9600);

  //Configura entradas digitales
  pinMode(P41, INPUT);
```

```
  pinMode(P42, INPUT);
  for (int i = 14; i <= 20; i++)
  {
    pinMode(i, INPUT);
  }

  //Configura salidas digitales. 4-9
  for (int i = 4; i <= 9; i++)
  {
    pinMode(i, OUTPUT);
  }
}


//================================================================================
//  Nombre :     loop
//--------------------------------------------------------------------------------
//  Proposito:   Rutina bucle. realiza los test.
//================================================================================
void loop()
{
  //variables a su valor inicial
  inicializacion();
  //detiene el timer
  Timer3.stop();

  //Comprueba conexión con processing la primera vez
  char intro = Serial.read();
  if (primerTest)
  {
    //Tipo de grabación
    while (intro != 'c' && intro != 'v')
    {
      delay(25);
      intro = Serial.read();
      Serial.print('k');
    }
    primerTest = false;
    if (intro == 'v')
    {
      soloGrabar = true;
    }
    else
    {
      soloGrabar = false;
    }
  }

  //Espera para iniciar el test (cuando se pulsa START)
  while (intro != 'a' && intro != 'q')
  {
    delay(25);
    intro = Serial.read();
  }

  //Se realiaza el test paso a paso
  //paso 1
  if ( lecturaAnalogica(P11, limiteInf5v, limiteSup5v, P11mRecta, P11bRecta, 1))
  {
    //paso 2
    if (lecturaAnalogica(P44, limiteInf12v, limiteSup12v, P44mRecta, P44bRecta, 2))
    {
      //tensiones de alimentación Ok a processing
      Serial.print("vOK*");

      //paso 3
      //Espera el resultado de la grabación
      while (intro != 's' && intro != 'n' && intro != 'd')
      {
        delay(25);
        intro = Serial.read();
      }

      //Grabación OK
      if (intro == 'd')
```

```
{
  resultado[3] = "no grabar";
}
else if (intro == 's')
{
  resultado[3] = "OK";
}

//grabación correcta
if (intro == 's' || intro == 'd')
{
  if (!soloGrabar)
  {
    //paso 4
    digitalWrite(P31, LOW);
    digitalWrite(modoTest, HIGH);
    delay(100);

    //paso 5
    digitalWrite(P14, HIGH);
    delay(500);
    digitalWrite(P14, LOW);
    delay(200);

    //paso 6
    //envía señal de test de zumbador
    Serial.print("zumbador*");

    //espera respuesta sobre comprobación manual o automática
    while (intro != 'z' && intro != 'x')
    {
      delay(25);
      intro = Serial.read();
    }
    //activa P31 (se activa el zumbador)
    digitalWrite(P31, HIGH);

    //comprobación automática
    if (intro == 'z')
    {
      tiempo = millis();
      zumb = zumbador();
    }

    //comprobación manual
    else
    {
      //respuesta del operario
      while (intro != 's' && intro != 'n')
      {
        delay(25);
        intro = Serial.read();
      }
    }

    //el zumbador se ha activado
    if (intro == 's' || zumb == true)
    {
      resultado[4] = "OK";
      //paso 7
      if (lecturaAnalogica(P43, 0, limiteSup0v, P43mRecta, P43bRecta, 5))
      {
        //paso 8
        while (numSemiciclo < 20)
        {
          //Rutina para generar y leer PWM
          testPwm();
        }
        //Desactiva interrupción
        detachInterrupt(0);
        Timer3.stop();
        if (!falloPwm)
        {
          resultado[6] = ("f=1kHz d=50%");

          //paso9
```

```
      digitalWrite(VP5, LOW);
      //retardos del cliente
      delay(500);
      if (digitalRead(P41) == LOW)
      {
        resultado[7] = "OK";

        //paso 10
        if (lecturaDigital())
        {
          Serial.print("ok*");
        }
        else
        {
          Serial.print("fail*");
        }
      }
      //falla paso 9
      else
      {
        resultado[7] = "FAIL";
        Serial.print("fail*");
      }
    }
          //Falla paso 8
          else
          {
            detachInterrupt(0);
            Timer3.disablePwm(VP5);
            resultado[6] = ("FAIL");
            Serial.print("fail*");
          }
        }
        //Falla paso 7
        else
        {
          Serial.print("fail*");
        }
      }
      //Falla paso 6
      else
      {
        resultado[4] = "FAIL";
        Serial.print("fail*");
      }
    }
    else
    {
      Serial.print("ok*");
    }
  }
  //falla el paso 3
  else if (intro == 'n')
      {
        resultado[3] = "FAIL";
        Serial.print("fail*");
      }
    }
    //Falla paso 2
    else
    {
      Serial.print("fail*");
    }
    //Falla paso 1
  }
  else
  {
    Serial.print("fail*");
  }

  //Envía los resultados de las I/O uno a uno
  for (int i = 0; i <= 7; i++)
  {
    Serial.print(ioDig[i]);
    Serial.print(";");
  }
```

```
  Serial.print("*");

  //Envía el vector de resultados de todos los test
  for (int i = 1; i <= 15; i++)
  {
    Serial.print(resultado[i]);
    Serial.print(";");
  }
  Serial.print("*");

}//fin loop
```

3.2.2 Inicializacion

```
//=====================================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  Inicializacion
//  OBJETIVO: Inicializa variables
//=====================================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=====================================================================================

void inicializacion()
{
    numSemiciclo    = 0;
    errorSemiciclo  = 0;
    falloPwm        = false;

    //Salidas digitales
    //deteccion remolque a cero (DR)
    digitalWrite(P31, LOW);
    digitalWrite(modoTest, LOW);
    //reset bajo-activo
    digitalWrite(P14, LOW);
    //VP5, ReleA, ReleB a cero
    digitalWrite(VP5, LOW);
    digitalWrite(releA, LOW);
    digitalWrite(releB, LOW);

    //rellena vector de resultados de los tests
    for (int i=0; i<=15; i++)
    {
        resultado[i]='x';
    }

    //rellena vector de resultados de las I/O uno a uno
    for (int i=0; i<=7; i++)
    {
        ioDig[i]='x';
    }

    //apaga el timer
    detachInterrupt(0);
    Timer3.stop();
}
```

3.2.3 lecAnalog

```
//=====================================================================================
//  Nombre: LecturaAnalogica
//-------------------------------------------------------------------------------------
//  Propósito: Leer la tensión en la entrada análogica
//  Recibe:    entrada        número de la entrada
//             limiteinf      límite de tensión inferior
//             limitesup      límite de tensión superior
//             escala         escala de tensión (5V, 12V)
//             indice         posición del vector 'resultado'
//
//  Devuelve:  true           el bitsIn medido está entre los límites
//             false          el bitsIn medido está fuera de los límites
//=====================================================================================

boolean lecturaAnalogica(int ent, float liminf, float limsup, float mRecta, float bRecta, int
indice)
{
    const float resArd=0.0048828; //  5/1024
    float bitsIn=0;
    float media=0;
    float vIn=0;
    float vOp=0;

    //realiza 10 lecturas y acumula el resultado
    for(int i=1; i<=10; i++)
    {
        bitsIn = analogRead(ent);
        media = media + bitsIn;
    }

    //realiza la media y la guarda en el vector de resultados
    bitsIn = media/10;
    //tensión que está dando el operacional
    vOp=bitsIn*resArd;
    //tensión de entrada al operacional
    vIn=(vOp-bRecta)/mRecta;

    //guarda el valor medido en el vector de resultados
    //si el resultado es negativo es porque no hay alimentación
    if(vIn<-2)
    {
        resultado[indice]='x';
    }
    else
    {
        resultado[indice]=vIn;
    }

    //resultado del test.
    if((vIn<=limsup) && (vIn>=liminf))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

3.2.4 Zumbador

```
//=====================================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  Zumbador
//  OBJETIVO: test del zumbador (detección de sonido)
//=====================================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=====================================================================================
boolean zumbador()
{
    int cont=0;
    int lec=0;
    //espera de 1 segundo (cliente)
```

```
    while((millis()-tiempo) < 1100)
    {
              //lee señal del micrófono
              lec=analogRead(audio);
              //cuando el microfono detecta el pitido
              if(lec>1000)
              {
                  //aumenta el contador
                  cont++;
              }
    }
    //el contador aumenta entre 2400 y 2600. se amplía el rango
    if((cont>=2000) && (cont<3000))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

3.2.5 PWM

```
//=============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  PWM
//  OBJETIVO: test del PWM
//=============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=============================================================================

void testPwm()
{
    //Conecta salida VP5 a entradas (de la tarjeta) P5.2  P5.3 y P5.4
    digitalWrite(releA, LOW);
    digitalWrite(releB, LOW);

    //Inicializa el timer y genera PWM 100hz, duty=0,5 en VP5
    Timer3.initialize();
    Timer3.pwm(VP5, 512, 10000);

    //Retardos del cliente
    delay(150);

    //si hay fallo o ha pasado 1 segundo
    if((numSemiciclo>=25) || (falloPwm==true))
    {
        //desactiva la interrupcion
        detachInterrupt(0);
        Timer3.stop();
    }
    else
    {
        //Activa la interrupcion
        attachInterrupt(0, interrupPwm, LOW);
    }
}


//=============================================================================
//  Nombre: interrupPwm
//-----------------------------------------------------------------------------
//  Propósito: leer PWM de entrada
//  Recibe:    nada
//  Devuelve:  nada
//=============================================================================
void interrupPwm()
{
    //Actualiza el contador
    numSemiciclo++;

    //Mide el tiempo que tarda en cambiar de estado
    if(digitalRead(P41)==HIGH)
    {
```

```
    tiempoSemiciclo = pulseIn(P41, HIGH);
}
else
{
    tiempoSemiciclo = pulseIn(P41, LOW);
}

//Medida correcta 5 +- 0.5 ms
if((tiempoSemiciclo<4200) || (tiempoSemiciclo>5800))
{
    errorSemiciclo++;
    //si hay más e 5 lecturas incorrectas, da error
    if(errorSemiciclo>5)
    {
        falloPwm=true;
    }
    //cuando no ha cambiado de estado, obliga error.
    if(tiempoSemiciclo==0)
    {
        errorSemiciclo=6;
    }
}
//Finaliza el test cuando ha pasado 1s o cuando hay fallo
if((numSemiciclo>=25) || (falloPwm==true))
    {
        numSemiciclo=25;
        detachInterrupt(0);
        Timer3.disablePwm(VP5);
        Timer3.stop();
    }
}
```

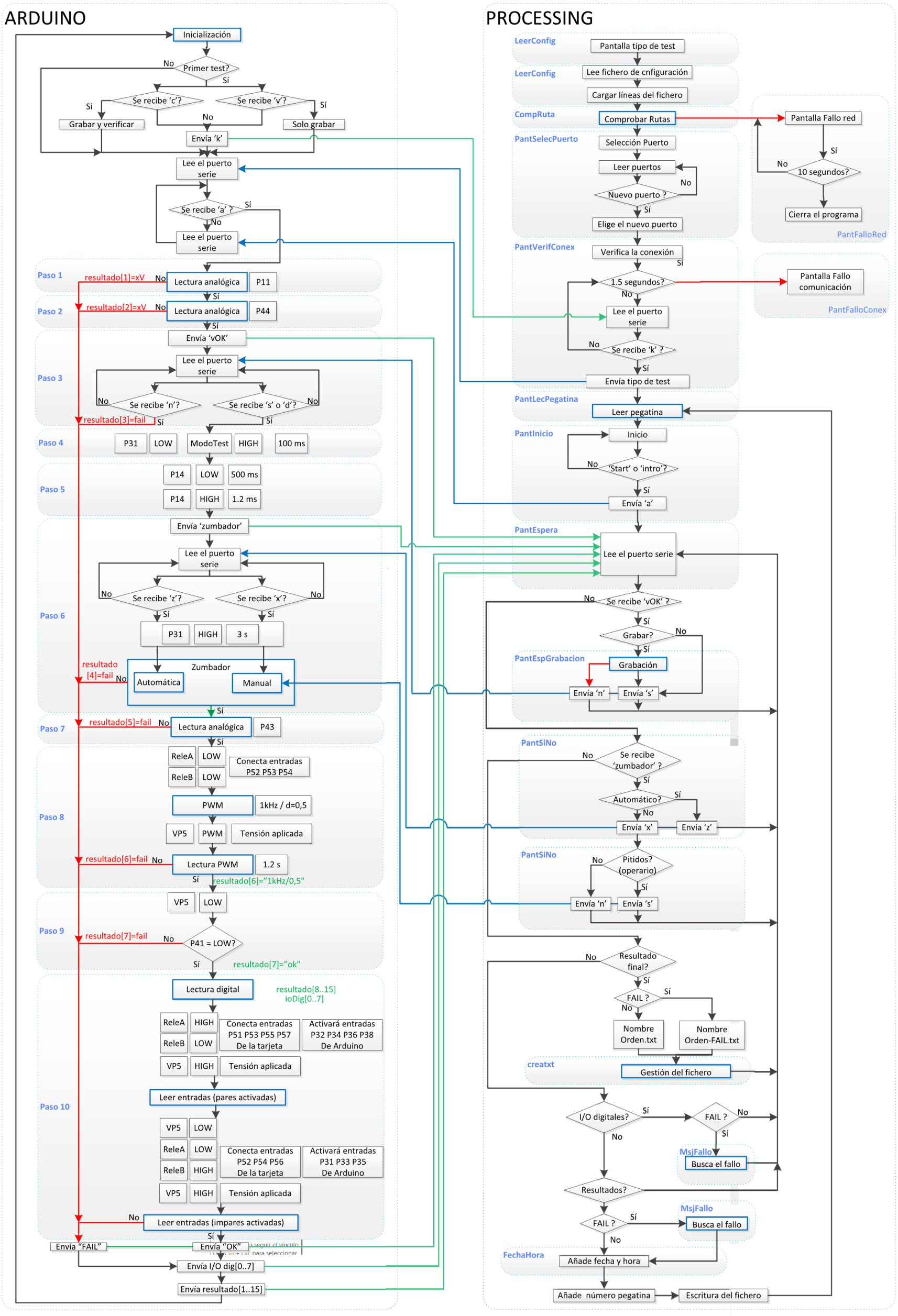
3.2.6 LecDigital

```
//=============================================================================
//  PROYECTO: Verificador Funcional
//  FICHERO:  LecDigital
//  OBJETIVO: manejo de las ioDig. paso 10
//=============================================================================
//  REALIZADO:   Mayo 2017,  Wilmer De Avila Gutierrez
//=============================================================================

boolean lecturaDigital()
{
    int     estado;
    int     estadoP42;
    boolean falloDig  = false;
    int j=0, i=0, k=0;
    //int i=0;

//Conecta salida VP5 a entradas (de la tarjeta) P5.1  P5.3 P5.5 y P5.7
digitalWrite(releB, LOW);
digitalWrite(releA, HIGH);
delay(50);
//Activa VP5
digitalWrite(VP5, HIGH);

//Mide salidas del P32..P38 de la tarjeta. entradas 14..20 de arduino
for(i=14; i<=20; i++)
{
    //índice del vector de resultados
    j++;
    k=(i-(j*2-1))+1;

    //retardos del cliente
    delay(150);
    //lectura y escalado
    estado = digitalRead(i);

    if(i==17)
    {
      estadoP42=digitalRead(P42);
    }
```

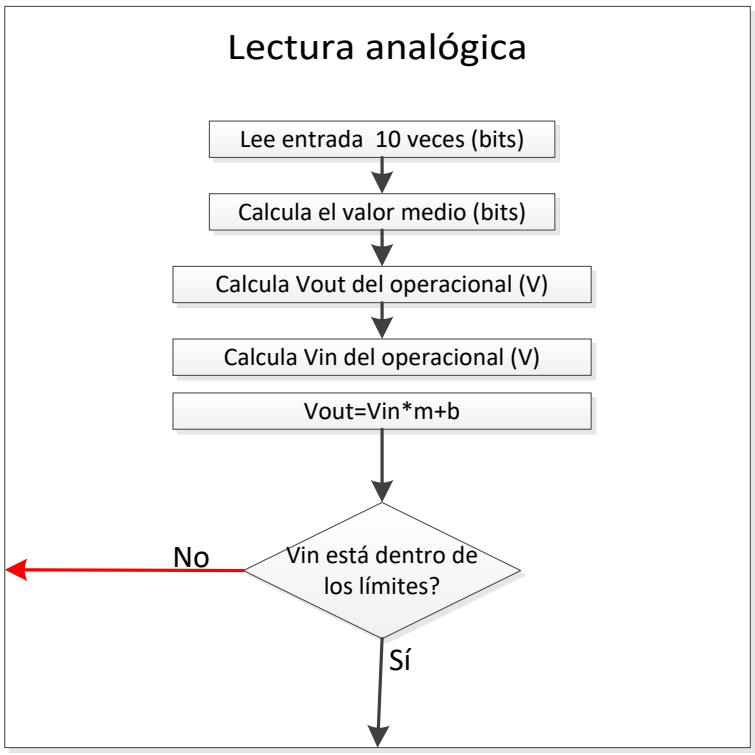
```
//retardos del cliente
delay(300);

//cuando se activa P35 hay que mirar lo mismo en el P42
if(i==17)
{
    if(estadoP42==LOW)
    {
        resultado[15]= "L=ok";
             ioDig[7]= "1";
    }
    else
    {
        resultado[15]=  "L=fail";
             ioDig[7]=  "0";
             falloDig= true;
    }
}
//entradas pares deben estar a nivel alto, impares a nivel bajo
if (i%2==0)
{
    if(estado==HIGH)
    {
        resultado[k]= "H=ok";
         ioDig[k-8]= "1";
    }
    else
    {
        resultado[k]= "H=fail";
         ioDig[k-8]= "0";
            falloDig= true;
    }
}
else
{
    if(estado==HIGH)
    {
        resultado[k]= "L=fail";
        ioDig[k-8]= "0";
        falloDig= true;
    }
    else
    {
        resultado[k]= "L=ok";
         ioDig[k-8]= "1";
    }
}
}// fin for
j=0;
//desactiva la salida
digitalWrite(VP5, LOW);
//Conecta salida VP5 a entradas (de la tarjeta) P5.2  P5.4 y P5.6
digitalWrite(releA, LOW);
digitalWrite(releB, HIGH);
delay(50);
//Activa VP5
digitalWrite(VP5, HIGH);

//mide entradas impares
for(i=14; i<=20; i++)
{
    //índice del vector de resultados
    j++;
    k=(i-(j*2-1))+1;

    //retardos del cliente
    delay(150);
    //lectura y escalado
    estado = digitalRead(i);

    if(i==17)
    {
      estadoP42=digitalRead(P42);
    }
```

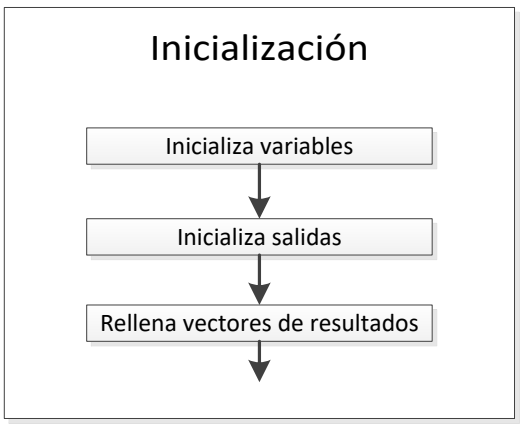
```
//retardos del cliente
delay(300);

if(i==17)
{
    if(estadoP42==HIGH)
    {
        resultado[15]= "H=ok/"+resultado[15];
            ioDig[7]= "1"+ioDig[7];
    }
    else
    {
        resultado[15]= "H=fail/"+resultado[15];
            ioDig[7]= "0"+ioDig[7];
            falloDig= true;
    }
}

//entradas impares deben estar a nivel alto, pares a nivel bajo
if (i%2==0)
{
    if(estado==LOW)
    {
        resultado[k]= resultado[k]+"/L=ok";
         ioDig[k-8]= ioDig[k-8]+"1";
    }
    else
    {
        resultado[k]= resultado[k]+"/L=fail";
         ioDig[k-8]= ioDig[k-8]+"0";
            falloDig= true;
    }
}
else
{
    if(estado==HIGH)
    {
        resultado[k]= "H=ok/"+resultado[k];
         ioDig[k-8]= "1"+ioDig[k-8];
    }
    else
    {
        resultado[k]="H=fail/"+resultado[k];
         ioDig[k-8]= "0"+ioDig[k-8];
            falloDig= true;
    }
}
}// fin for

digitalWrite(VP5,   LOW);

if(falloDig==true)
{
  return false;
}
else
{
  return true;
}
}
```

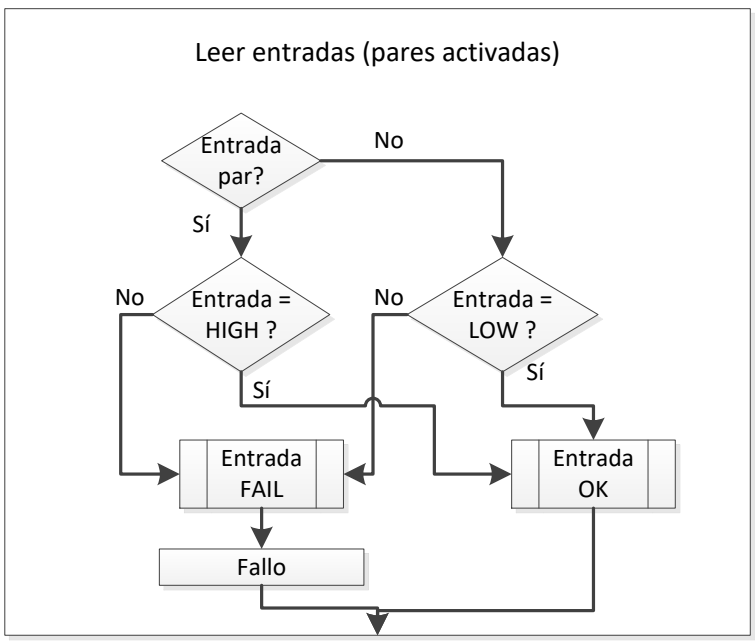
4 Diagrama de flujo

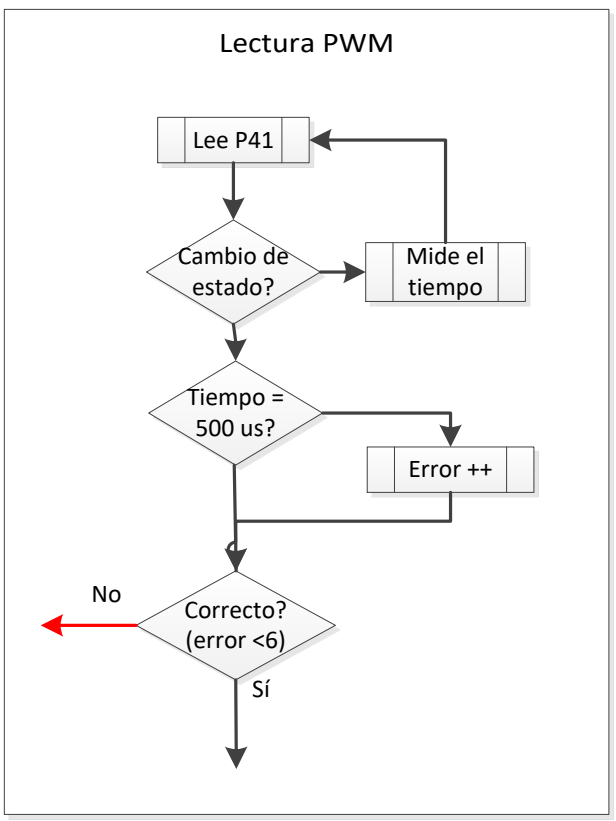


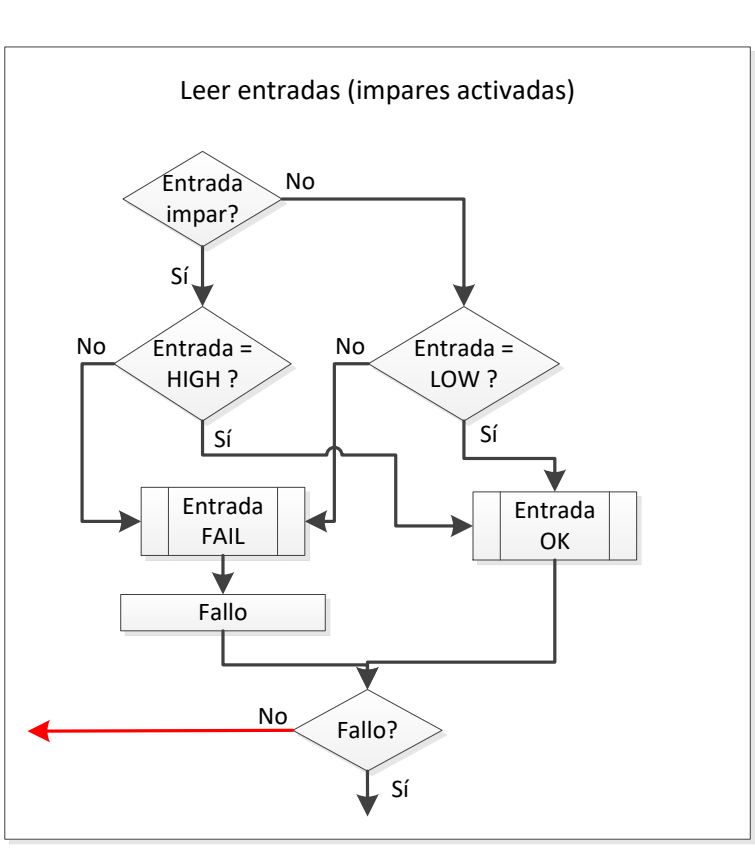
FUNCIONES/FICHEROS ARDUINO

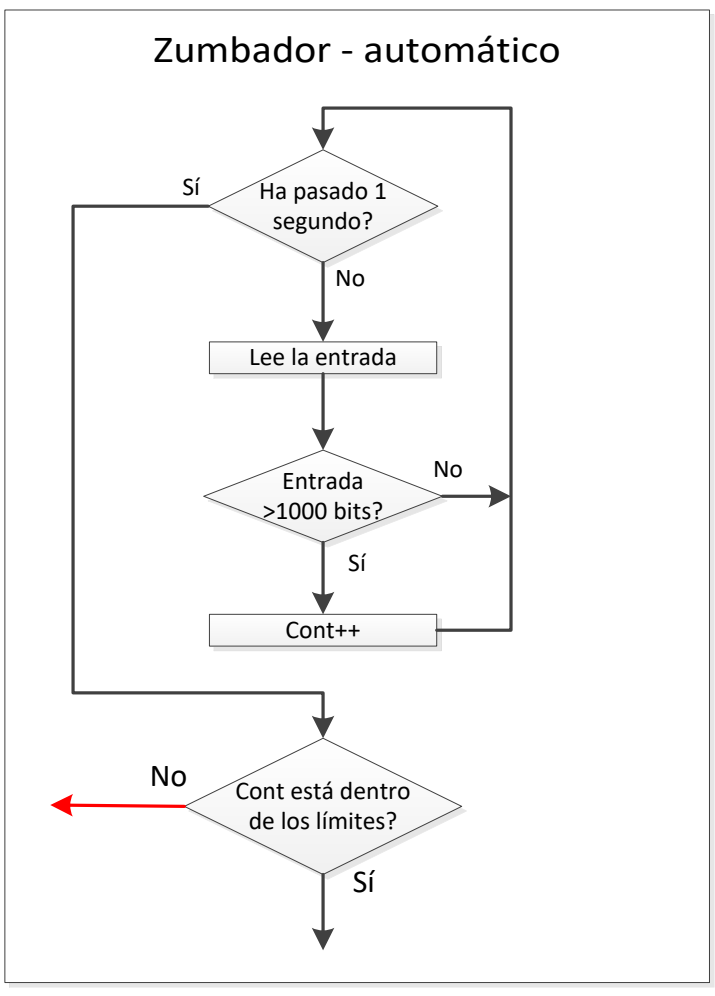




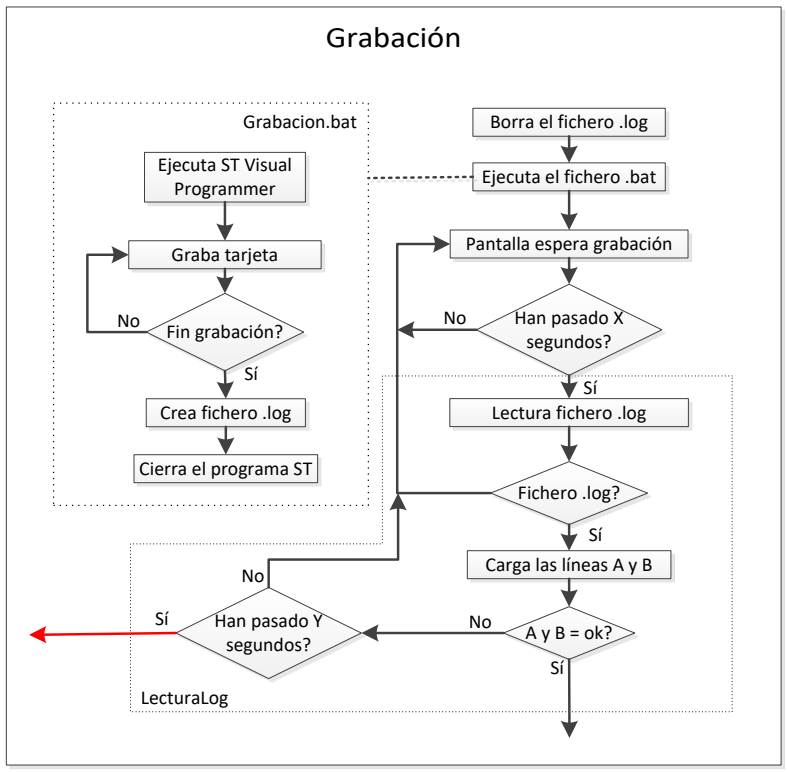


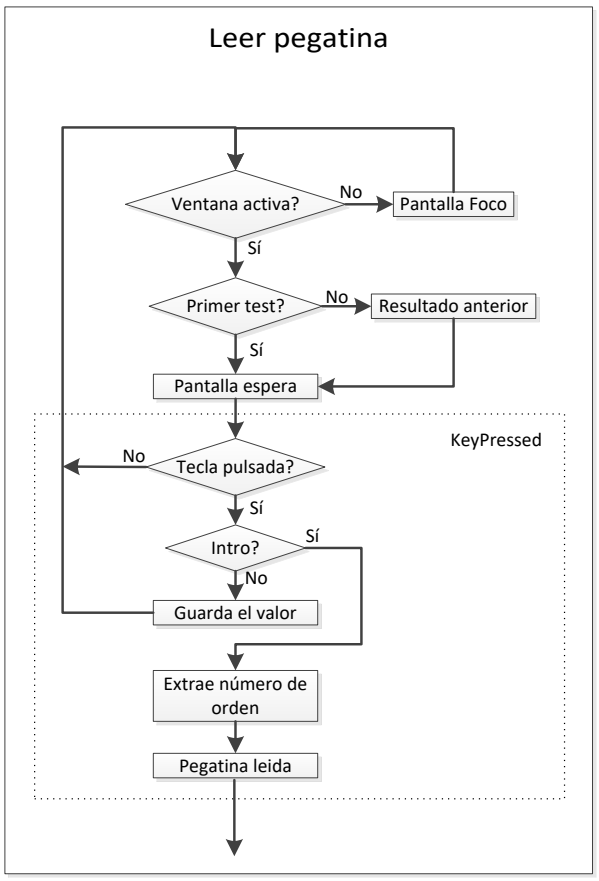


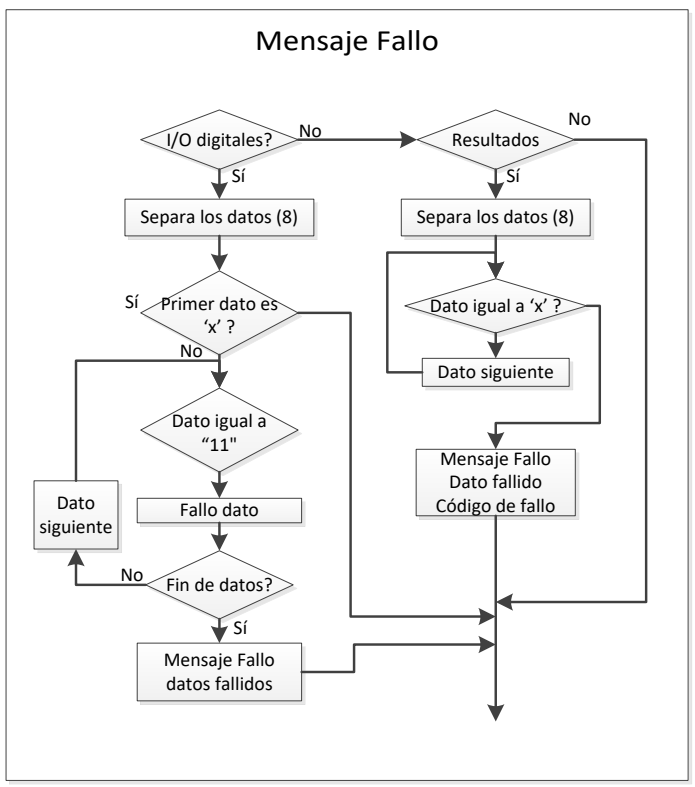


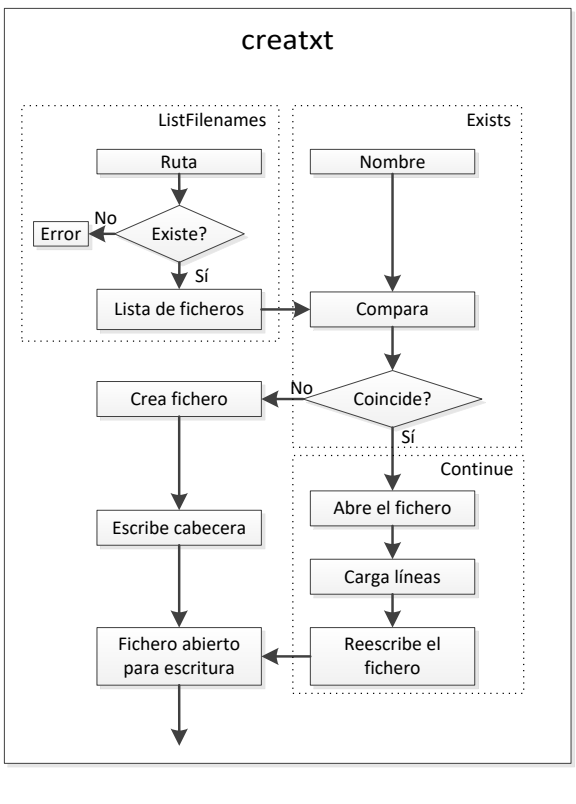


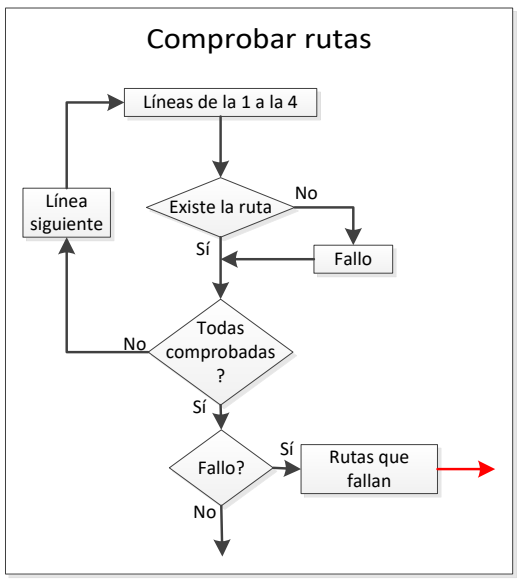
FUNCIONES/FICHEROS PROCESSING











5 Pauta de verificación

FALCÓN	PAUTA DE VERIFICACION	PV-1705081-0 Mayo 2017 REV: 00 Pág 1 de 4

Modelo: ARA-TB_Control-V2.00	**Equipos /Útiles a emplear:** -**U752** Lector 2D **U500** Mazo USB tipo A-B x 2 **E1115** Equipo funcional
Referencia: 8CG07S0 8CG07S0-INV	
Documento Técnico Aplicado: 170411 test	

Nota

Antes de verificar la tarjeta, hay que asegurarse que el lector 2D **U752** está configurado para dar un salto de línea al finalizar la lectura.

Nº	Proceso	Control de proceso
Preparación		
1	Conectar 2 cables **U500** al PC.	
2	Poner el interruptor "**POWER**" en "OFF" y conectar **la manguera de red** en el conector "**230Vac**", (Parte trasera de **E1115**)	
3	Poner el interruptor **"INICIO"** en **"OFF"** (Parte delantera de **E1115**)	
4	Conectar **U500** al conector "**USB-2**", (Lateral de **E1115**)	

REG.7/036 Rev 1

	PAUTA DE VERIFICACION	PV-1705081-0 Mayo 2017 REV: 00 Pág 2 de 4

5	Poner el interruptor "**POWER**" en "**ON**" (Parte trasera de **E1115**)	Comprobar que se ilumina el interruptor "**POWER**".
6	Ejecutar "**verificador_araelec**" de la carpeta **195** en el escritorio.	Debe aparecer esta ventana. 
7	Elegir la opción adecuada según la referencia de las tarjetas: 8CG07S0-INV SOLO GRABAR 8CG07S0 GRABAR Y VERIFICAR	Debe aparecer esta ventana.  Si aparece esta otra ventana, comprobar que el cable de red está debidamente conectado. 
8	Conectar **U500** al conector "**USB-1**", (Lateral de **E1115**)	

REG.7/036 Rev 1

FALCON ELECTRÓNICA	PAUTA DE VERIFICACION	PV-1705081-0 Mayo 2017 REV: 00 Pág 3 de 4

		Debe aparecer esta ventana 
9	La ventana del programa debe estar activa al momento de leer la pegatina. Con **U752**, leer la etiqueta QR de la tarjeta a verificar.	Después de leer la etiqueta debe aparecer esta ventana. 
10	Colocar la tarjeta a verificar en la cama de pinchos como se muestra en la imagen y bajar los empujadores.	
11	Poner el interruptor "**INICIO**" en "**ON**", (parte delantera de **E1115**)	Observar que se enciende el "LED" de la parte delantera.
12	Clic sobre el botón '**START**' o pulsar 'intro' para iniciar el test.	

REG.7/036 Rev 1

	PAUTA DE VERIFICACION	PV-1705081-0 Mayo 2017 REV: 00 Pág 4 de 4

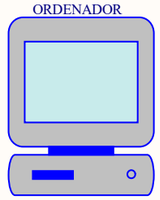
13	Esperar a que finalice el test.	Test OK.  En caso de fallo, anotar el número del fallo. 
14	Poner el interruptor "**INICIO**" en "**OFF**",	
15	Subir los empujadores y retirar la tarjeta.	
16	Para la verificación de otras tarjetas, repetir los pasos del **9** al **15**.	

Nº Rev	Modificación aplicada	Responsable	Aprobado	Fecha
00	Se crea la pauta.	W.De Avila	A. Bueno	16-05-2017

REG.7/036 Rev 1

6 Planos eléctricos

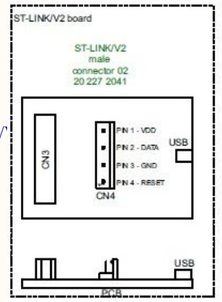
- **Esquema general.**
- **Módulo central.**
- **Entradas digitales.**
- **Salidas digitales.**
- **Salidas analógicas.**



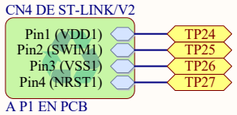
Ref. 2491563

PROGRAMACIÓN ST-LINK/

Cable USB A a mictoUSB

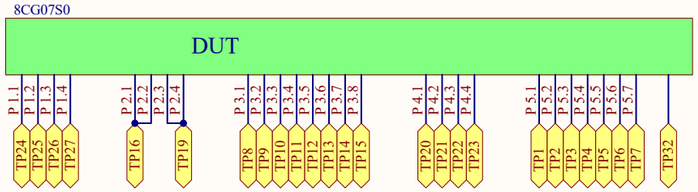


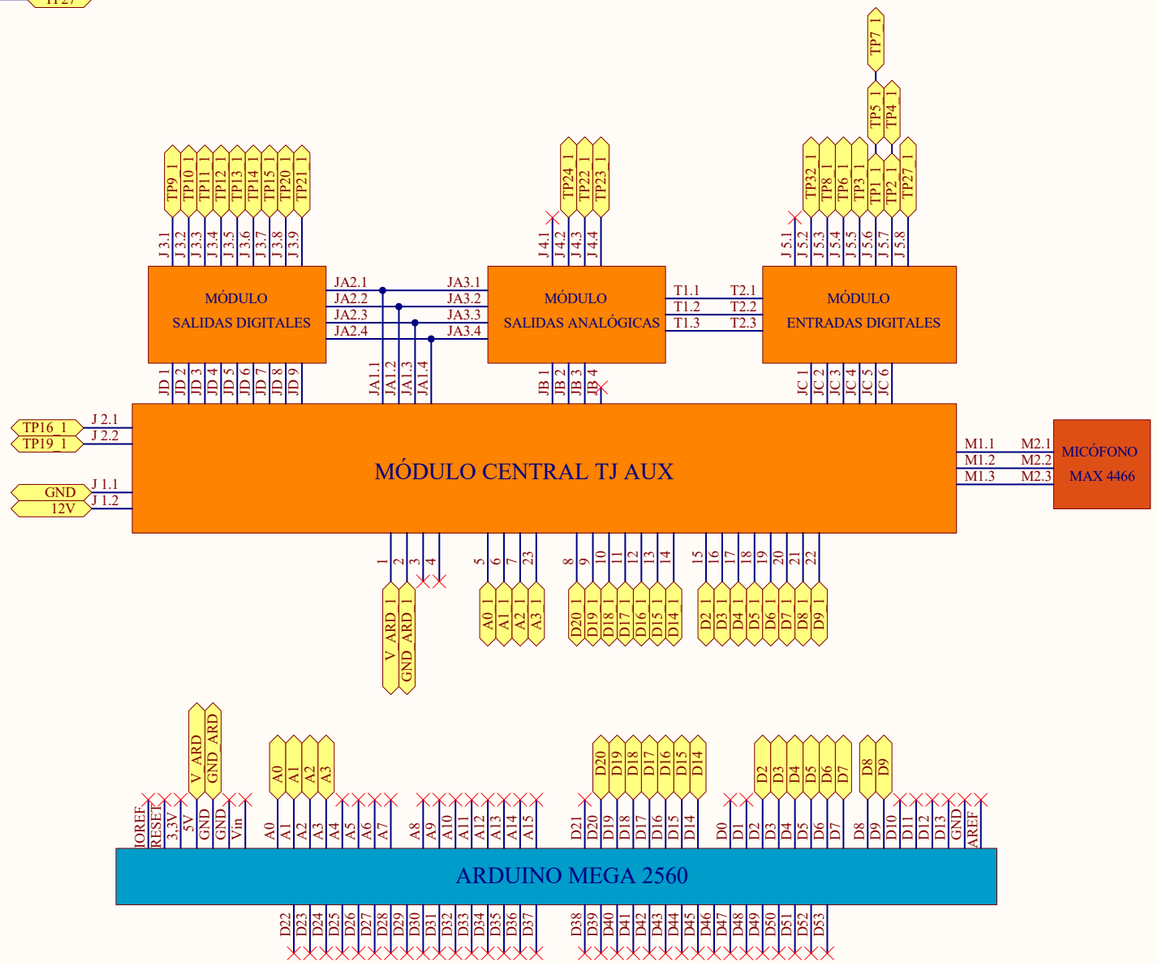
CN4 DE ST-LINK/V2

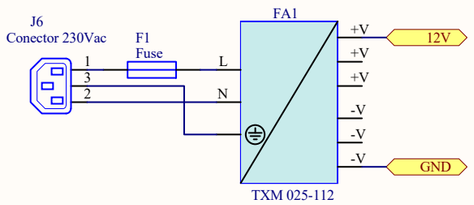


A P1 EN PCB

8CG07S0







TXM 025-112

ALIMENTACIÓN ARDUINO POR USB

Cable USB A-B

Ref. 2491563

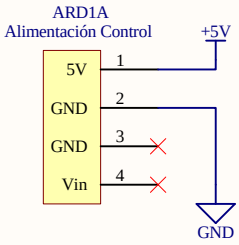
Rev.	Fecha	Descripción
0	20/Abr/2017	Esquema general del hardware del verificador.

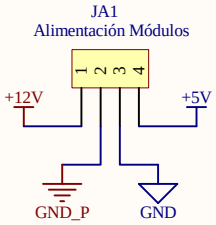
Titulo: ESQUEMA GENERAL

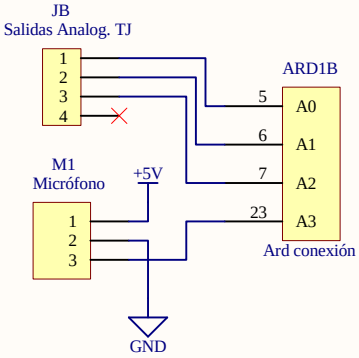
Tamaño:	Referencia:		Nombre	Fecha
A3	8CG07S0 8CG07S0-INV	Dibujado por:	S. Nesterenco	11/Abr/2017
Ver: 0	Archivo: EsquemaVerificador.SchDoc	Revisado:	W. De Avila	11/Abr/2017
	Hoja 1 de 5	Aprobado:	A. Bueno	12/Abr/2017

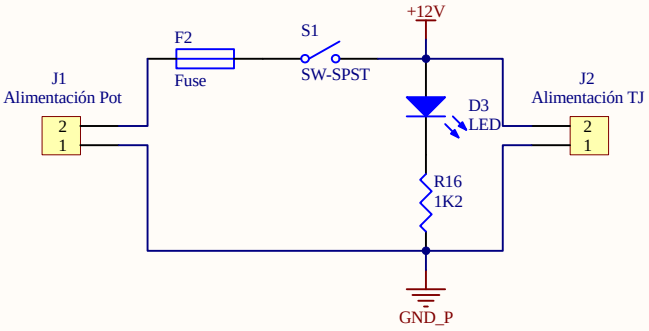
FALCÒN electrónica
Plgn. El Escopar Calle E, nº1
31350 - Peralta Navarra
www.electronicafalcon.com

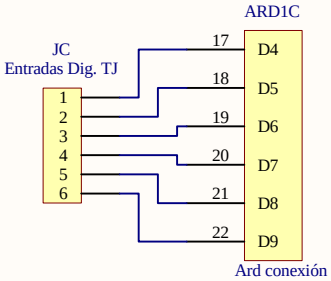
MÓDULO CENTRAL

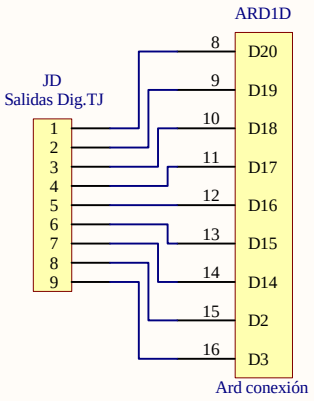












0	20/Abr/2017	Shield de conectores para Arduino
Rev.	Fecha	Descripción

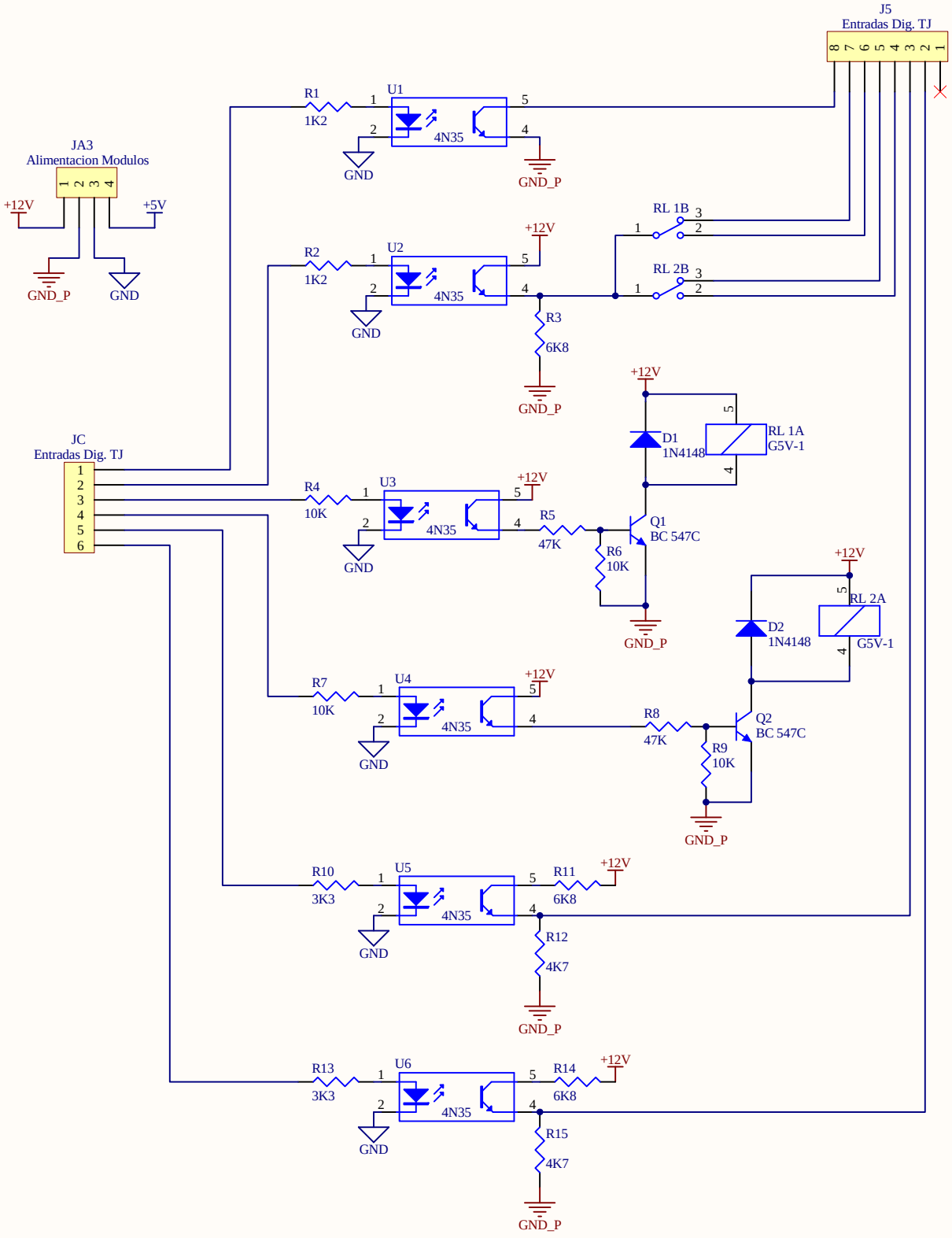
Titulo:

MÓDULO CENTRAL

Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha	
		Dibujado por:	S. Nesterenco	11/Abr/2017	
Ver: 0	Archivo: Modulo Central.SchDoc	Revisado:	W. De Avila	11/Abr/2017	
	Hoja 5 de 5	Aprobado:	A. Bueno	12/Abr/2017	

FALCÓN electrónica
Plgn. El Escopar Calle E, nº1
31350 - Peralta Navarra
www.electronicafalcon.com

TARJETA ENTRADAS DIGITALES



Rev.	Fecha	Descripción
0	20/Abr/2017	Entradas digitales de la UUT - Puerto 5. Salidas de Arduino

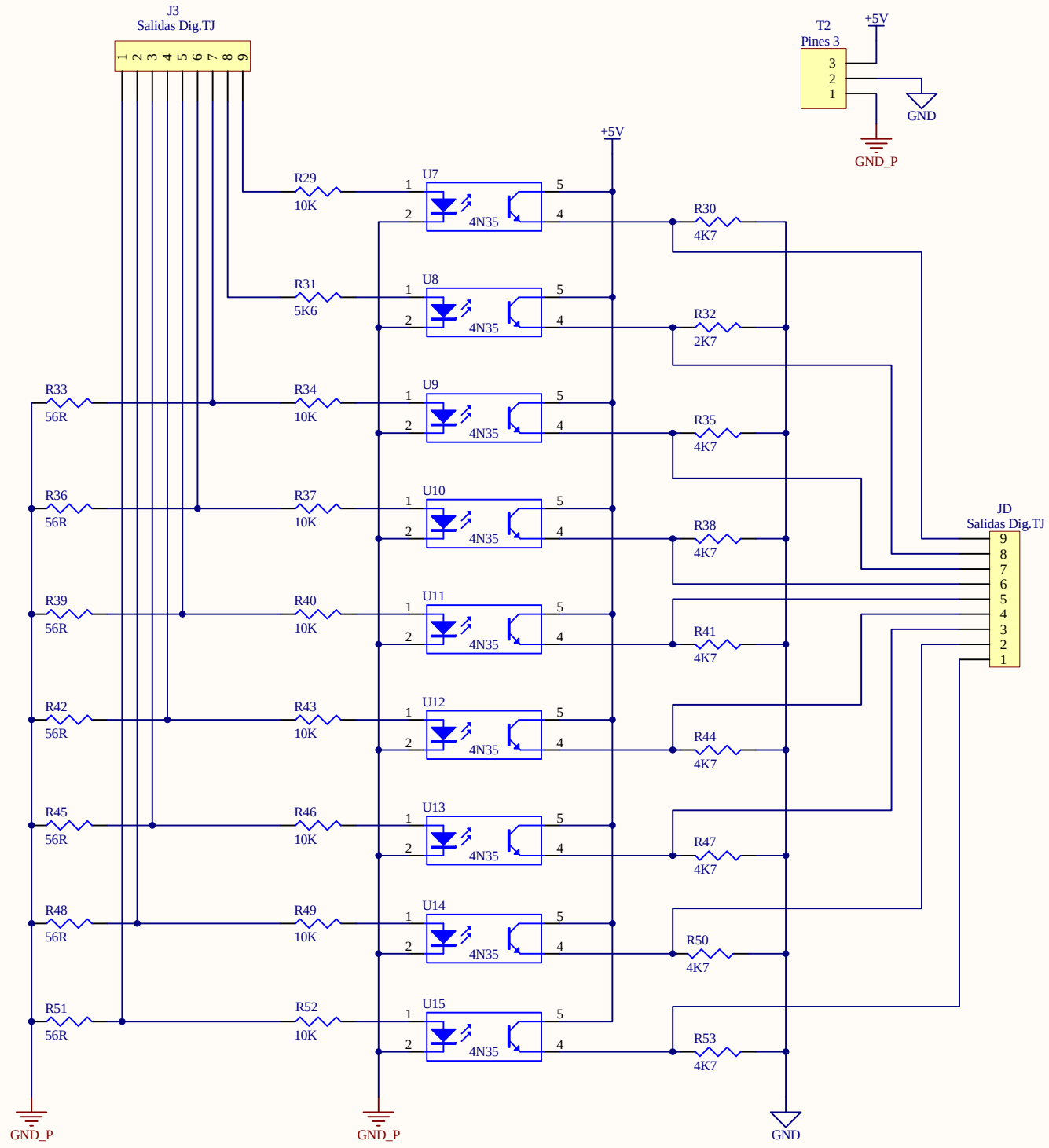
Titulo:

ENTRADAS DIGITALES

			Nombre	Fecha
Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	S. Nesterenco	11/Abr/2017
Ver: 0	Archivo: Entradas Digitales.SchDoc	Revisado:	W. De Avila	11/Abr/2017
	Hoja 2 de 5	Aprobado:	A. Bueno	12/Abr/2017

FALCON electrónica
Plgn. El Escopar Calle E, nº1
31350 - Peralta Navarra
www.electronicafalcon.com

TARJETA SALIDAS DIGITALES



Rev.	Fecha	Descripción
0	20/Abr/2017	Salidas digitales de la UUT - Puerto 3. Entradas de Arduino

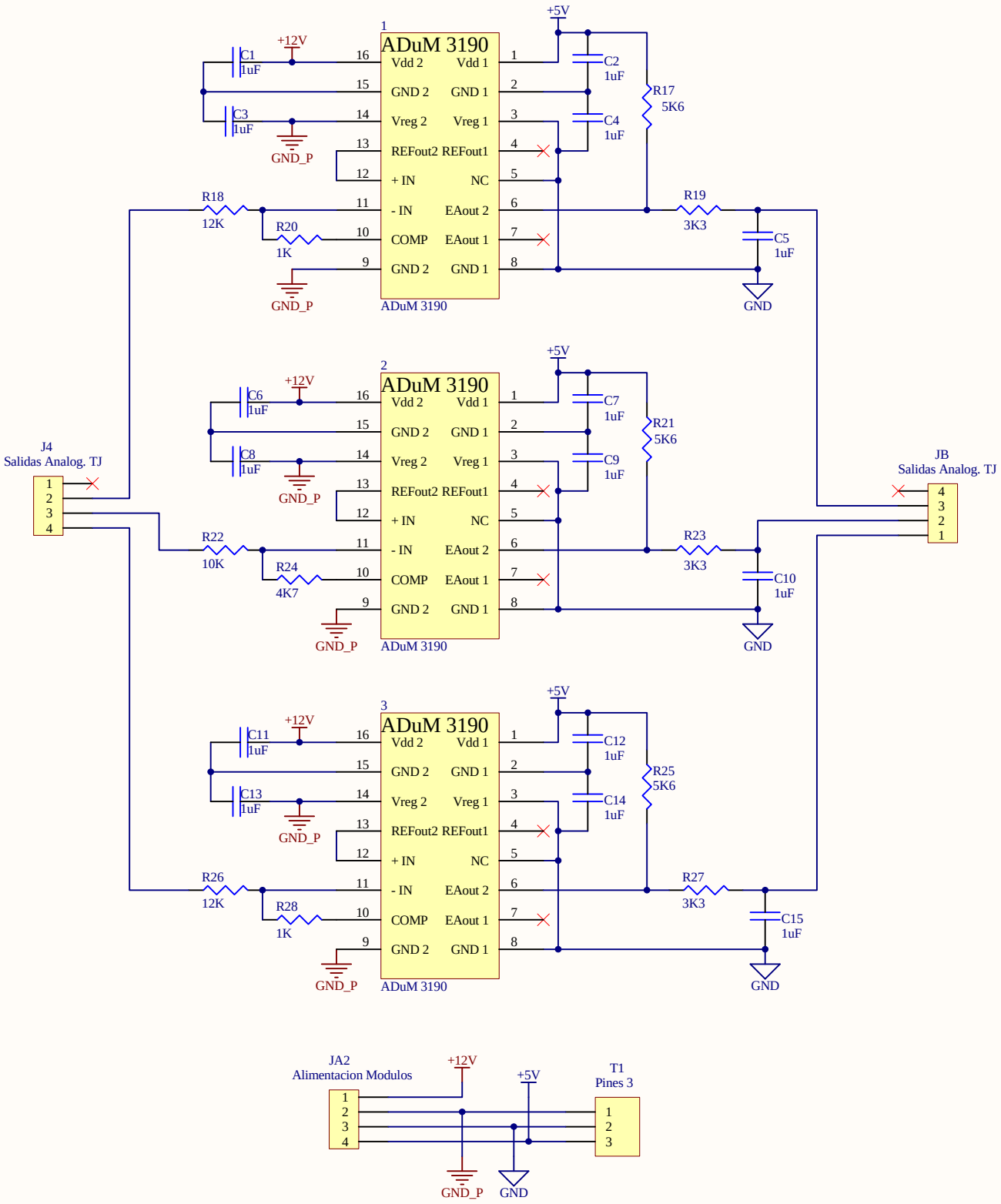
Titulo:

SALIDAS DIGITALES

			Nombre	Fecha
Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	S. Nesterenco	11/Abr/2017
Ver: 0	Archivo: Salidas Digitales.SchDoc	Revisado:	W. De Avila	11/Abr/2017
	Hoja 3 de 5	Aprobado:	A. Bueno	12/Abr/2017

FALCÓN electrónica
Plgn. El Escopar Calle E, nº1
31350 - Peralta Navarra
www.electronicafalcon.com

TARJETA SALIDAS ANALÓGICAS



Rev.	Fecha	Descripción
0	20/Abr/2017	Salidas analógicas de la UUT - Puertos 1 y 4. Entradas de Arduino

Titulo:

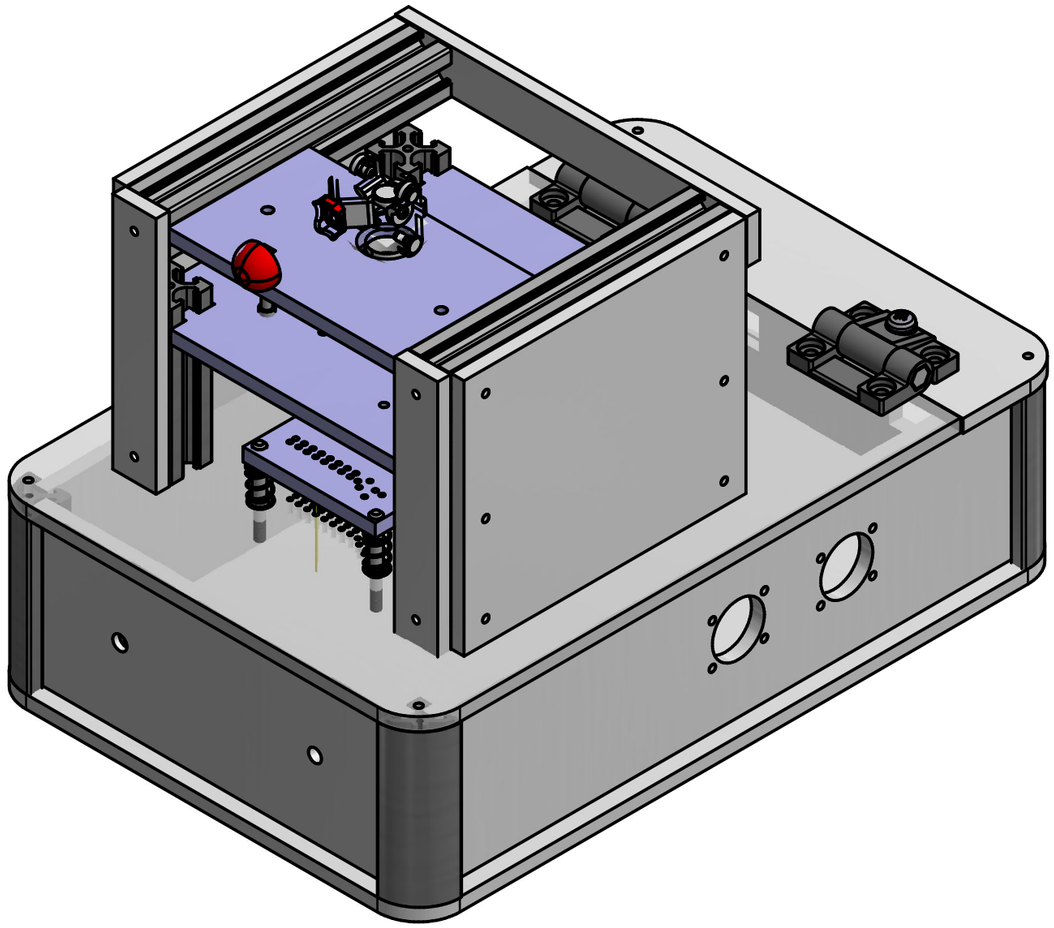
SALIDAS ANALÓGICAS

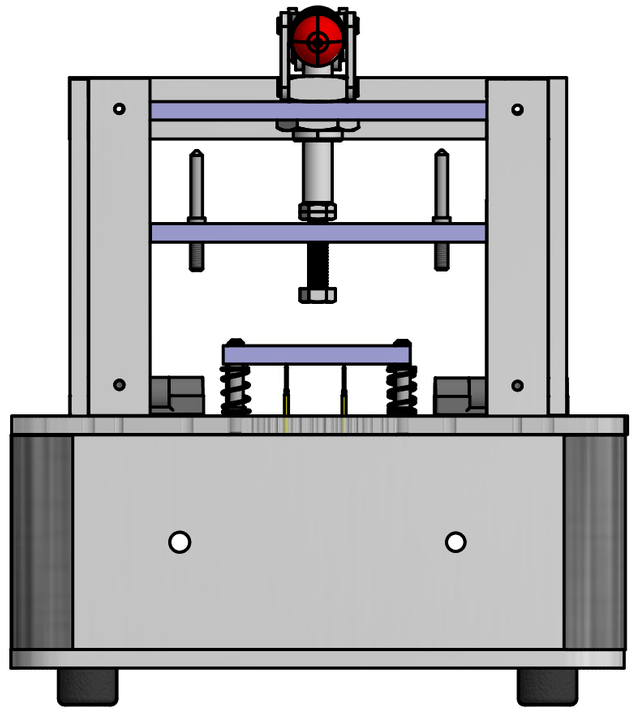
Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha
		Dibujado por:	S. Nesterenco	11/Abr/2017
Ver: 0	Archivo: Salidas Analogicas.SchDoc	Revisado:	W. De Avila	11/Abr/2017
	Hoja 4 de 5	Aprobado:	A. Bueno	12/Abr/2017

FALCON electrónica
Plgn. El Escopar Calle E, nº1
31350 - Peralta Navarra
www.electronicafalcon.com

7 Planos mecánicos

- **Verificador.**
- **Base.**
- **Trasera.**
- **Lateral izquierda.**
- **Lateral derecha.**
- **Frontal.**
- **Bisagras.**
- **Tapa superior.**
- **Soporte tarjeta.**
- **Empujador.**
- **Soporte brida.**
- **Soporte lateral.**
- **Soporte trasero.**
- **Soporte frontal.**



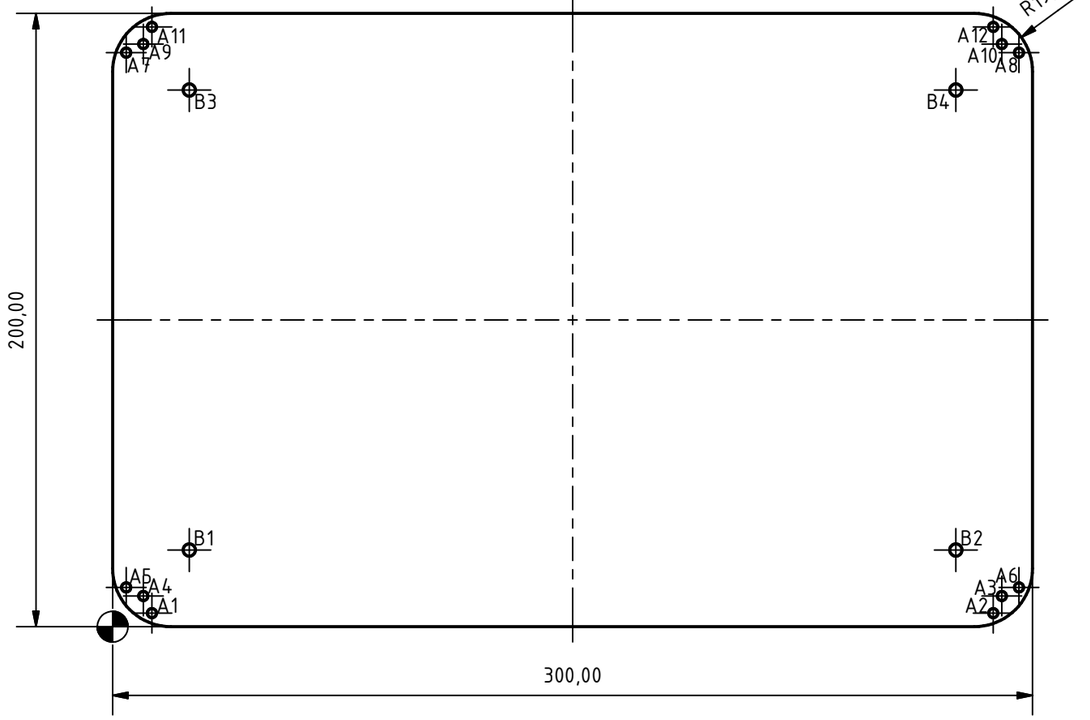


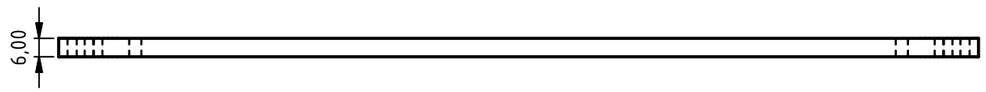
Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

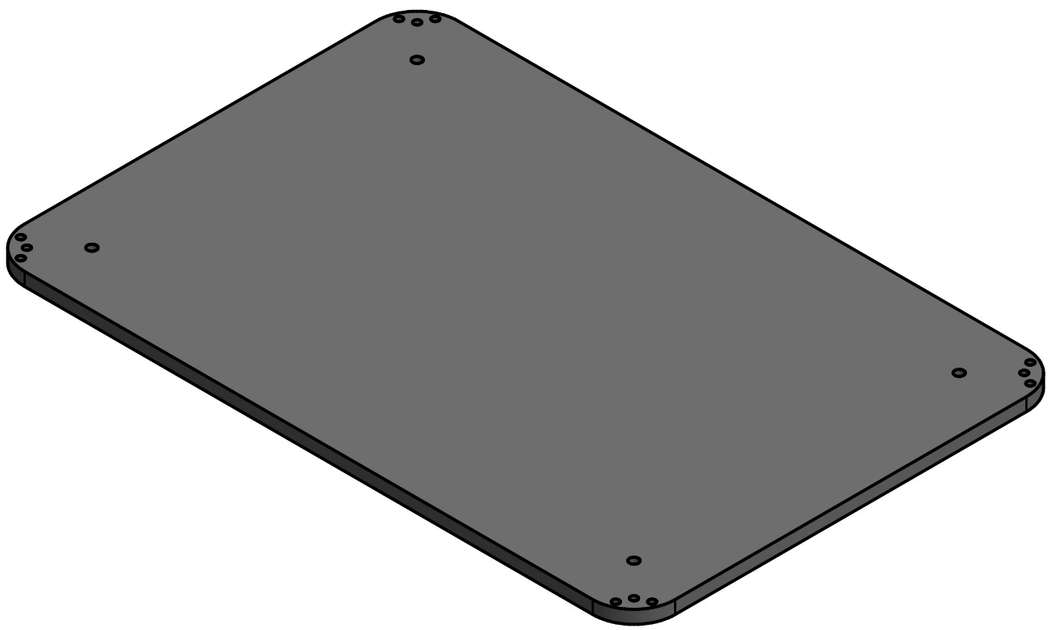
Título: Verificador

			Nombre	Fecha
Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24011_Base.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:2	Hoja 1 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com







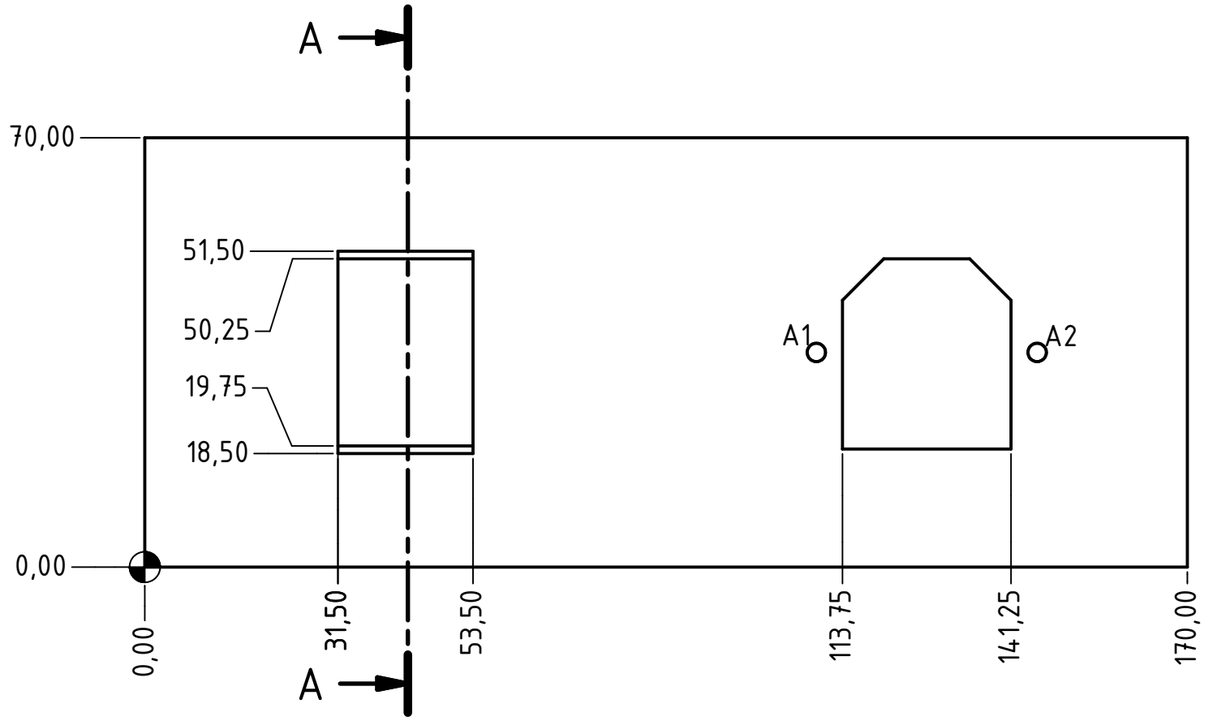
HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	12,80	4,41	Ø3,00 -6,00 DEEP
A2	287,20	4,41	Ø3,00 -6,00 DEEP
A3	290,00	10,00	Ø3,00 -6,00 DEEP
A4	10,00	10,00	Ø3,00 -6,00 DEEP
A5	4,41	12,80	Ø3,00 -6,00 DEEP
A6	295,59	12,80	Ø3,00 -6,00 DEEP
A7	4,41	187,20	Ø3,00 -6,00 DEEP
A8	295,59	187,20	Ø3,00 -6,00 DEEP
A9	10,00	190,00	Ø3,00 -6,00 DEEP
A10	290,00	190,00	Ø3,00 -6,00 DEEP
A11	12,80	195,59	Ø3,00 -6,00 DEEP
A12	287,20	195,59	Ø3,00 -6,00 DEEP
B1	25,00	25,00	Ø4,00 -6,00 DEEP
B2	275,00	25,00	Ø4,00 -6,00 DEEP
B3	25,00	175,00	Ø4,00 -6,00 DEEP
B4	275,00	175,00	Ø4,00 -6,00 DEEP

Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

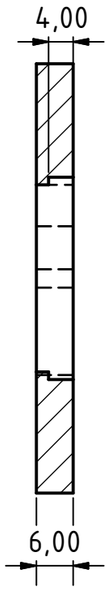
Título: Base

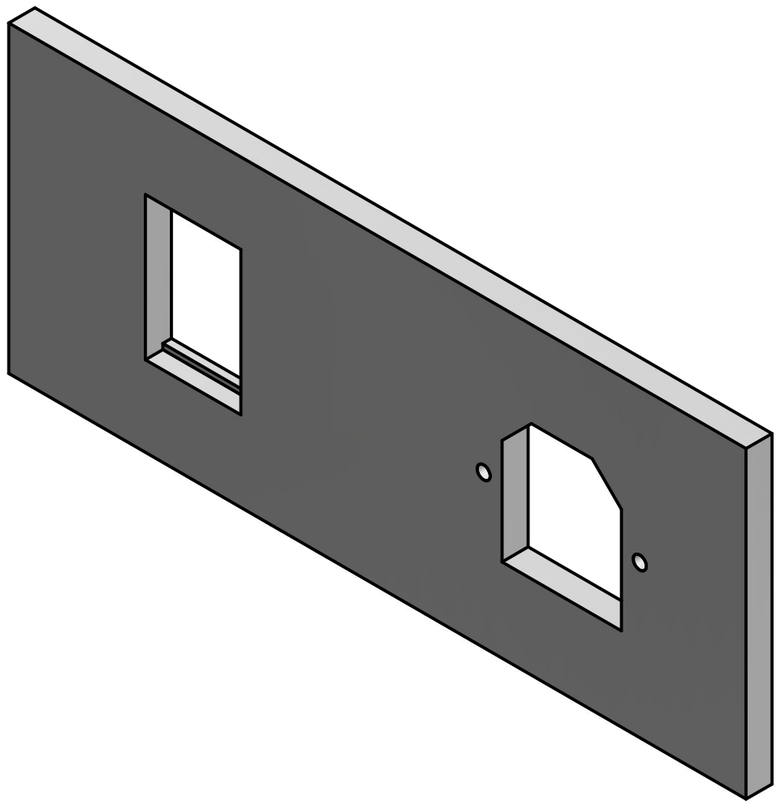
			Nombre	Fecha
Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24011_Base.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:2	Hoja 2 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com



A-A (1 : 1)





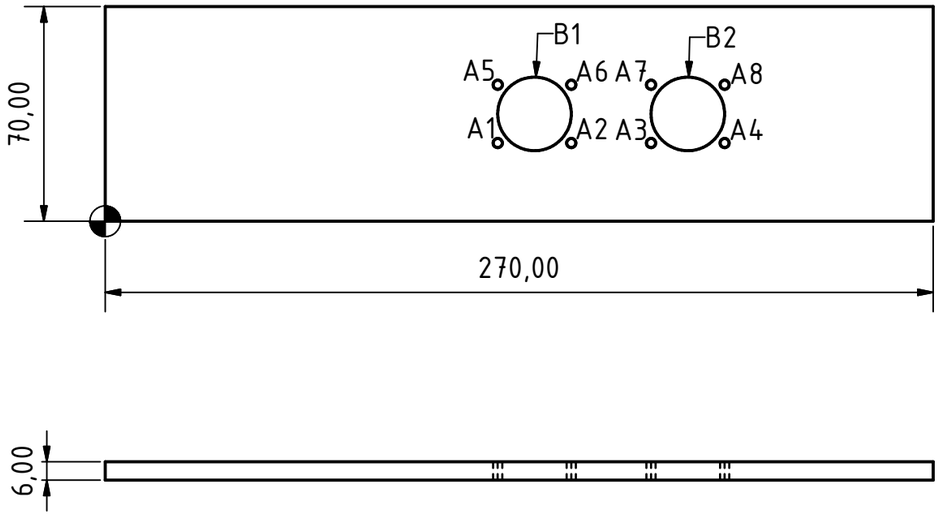
HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	109,50	35,00	Ø3,00 -6,00 DEEP
A2	145,50	35,00	Ø3,00 -6,00 DEEP

Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

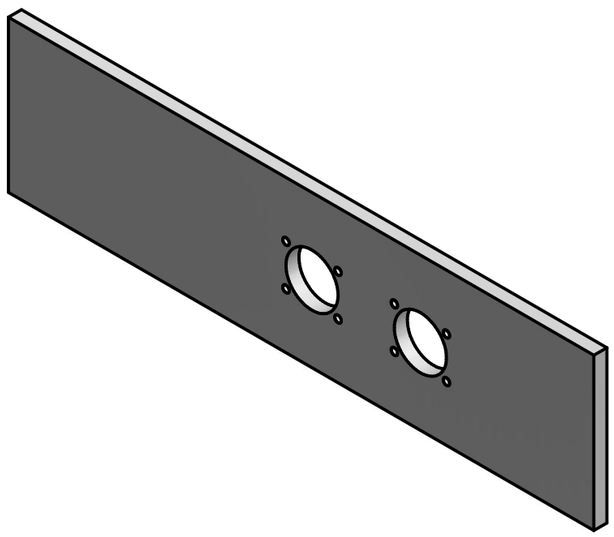
Título: Parte trasera

			Nombre	Fecha
Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24021_Parte Trasera.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:1	Hoja 3 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com



HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	128,00	25,50	Ø3,00 -6,00 DEEP
A2	152,00	25,50	Ø3,00 -6,00 DEEP
A3	178,00	25,50	Ø3,00 -6,00 DEEP
A4	202,00	25,50	Ø3,00 -6,00 DEEP
A5	128,00	44,50	Ø3,00 -6,00 DEEP
A6	152,00	44,50	Ø3,00 -6,00 DEEP
A7	178,00	44,50	Ø3,00 -6,00 DEEP
A8	202,00	44,50	Ø3,00 -6,00 DEEP
B1	140,00	35,00	Ø24,00 -6,00 DEEP
B2	190,00	35,00	Ø24,00 -6,00 DEEP

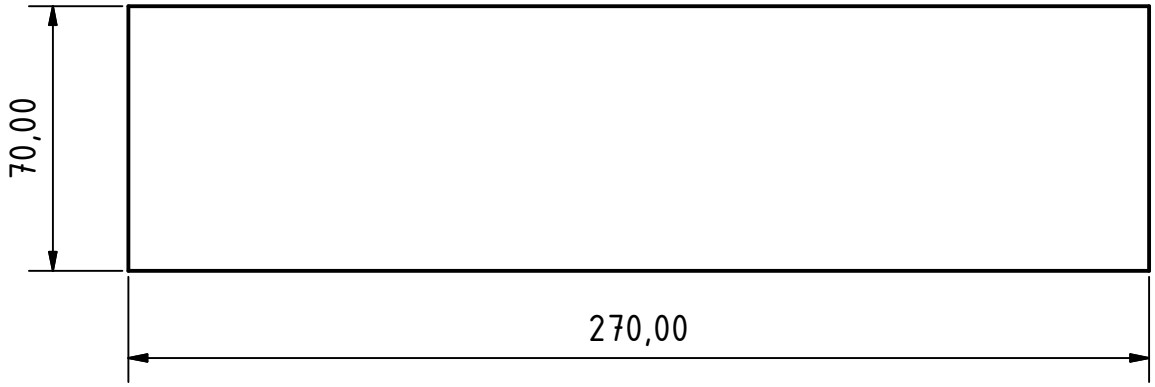


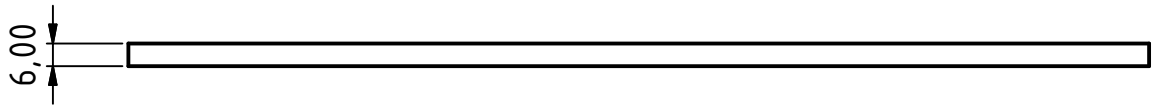
Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

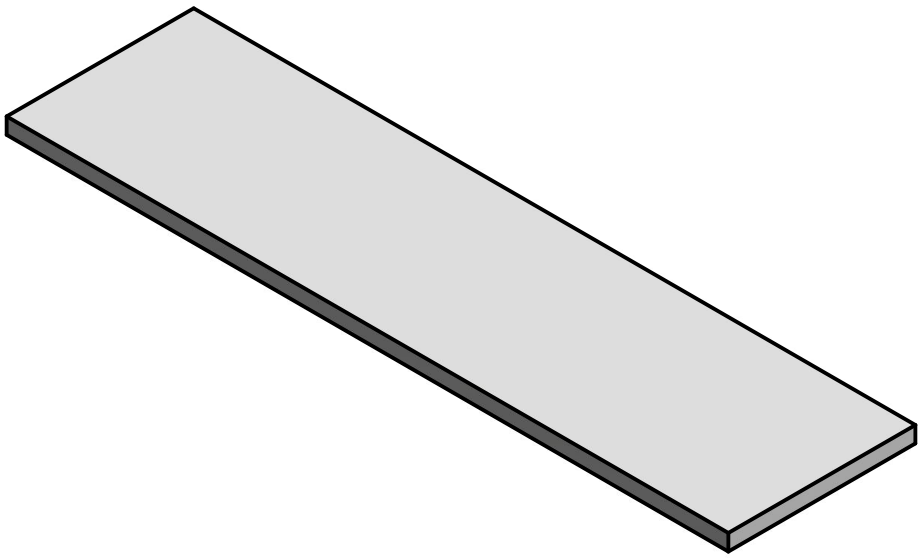
Título: Lateral izquierdo

			Nombre	Fecha
Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24031_Lateral_izq	Revisado:	A. Bueno	02/06/2017
Esc: 1:2	Hoja 4 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com





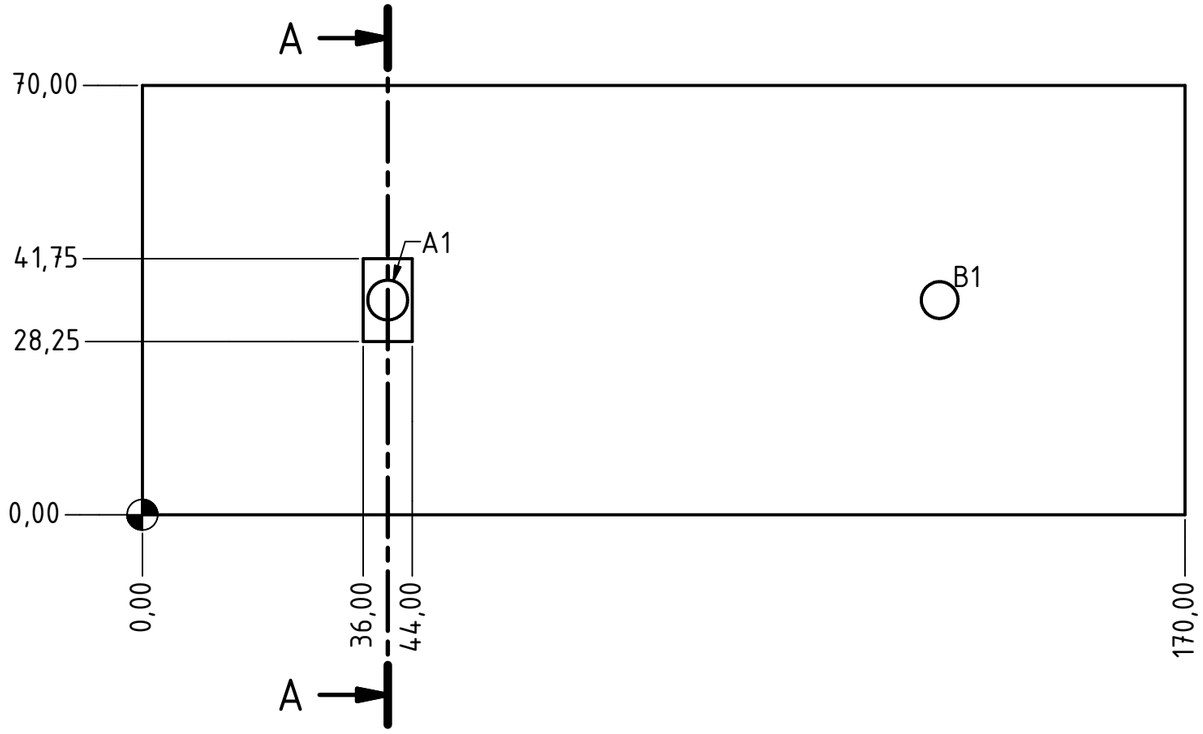


Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

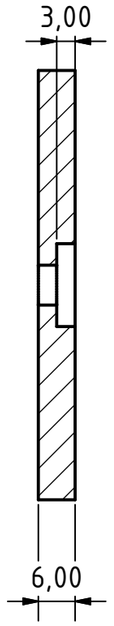
Título: Lateral derecho

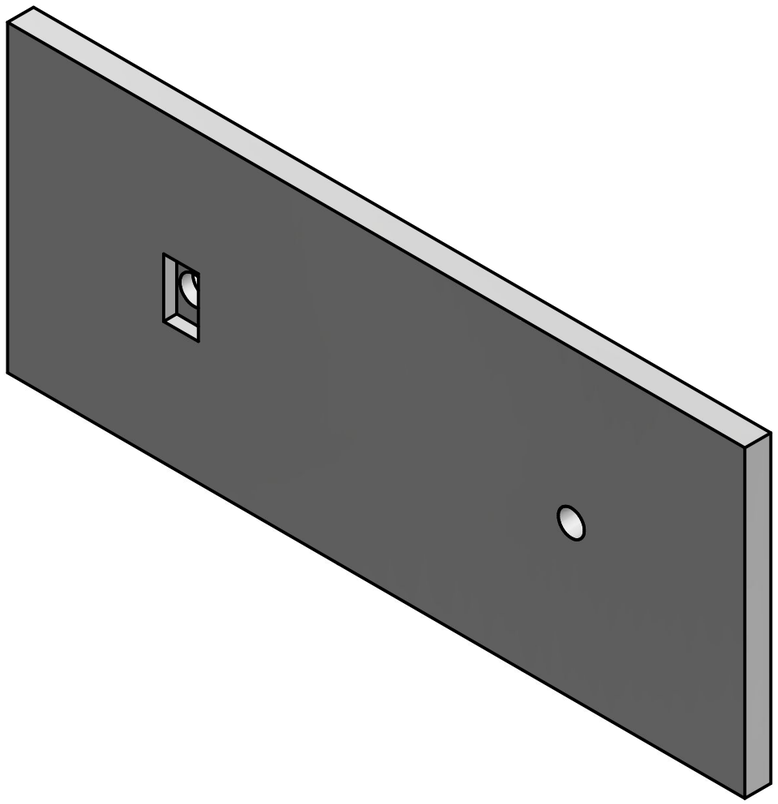
Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha	
		Dibujado por:	W. De Avila	01/06/2017	
Ver: 1	Archivo: 24041_Lateral_dcha.ipt	Revisado:	A. Bueno	02/06/2017	
Esc: 1:2	Hoja 5 de 14	Aprobado:	A. Bueno	02/06/2017	

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com



A-A (1 : 1)





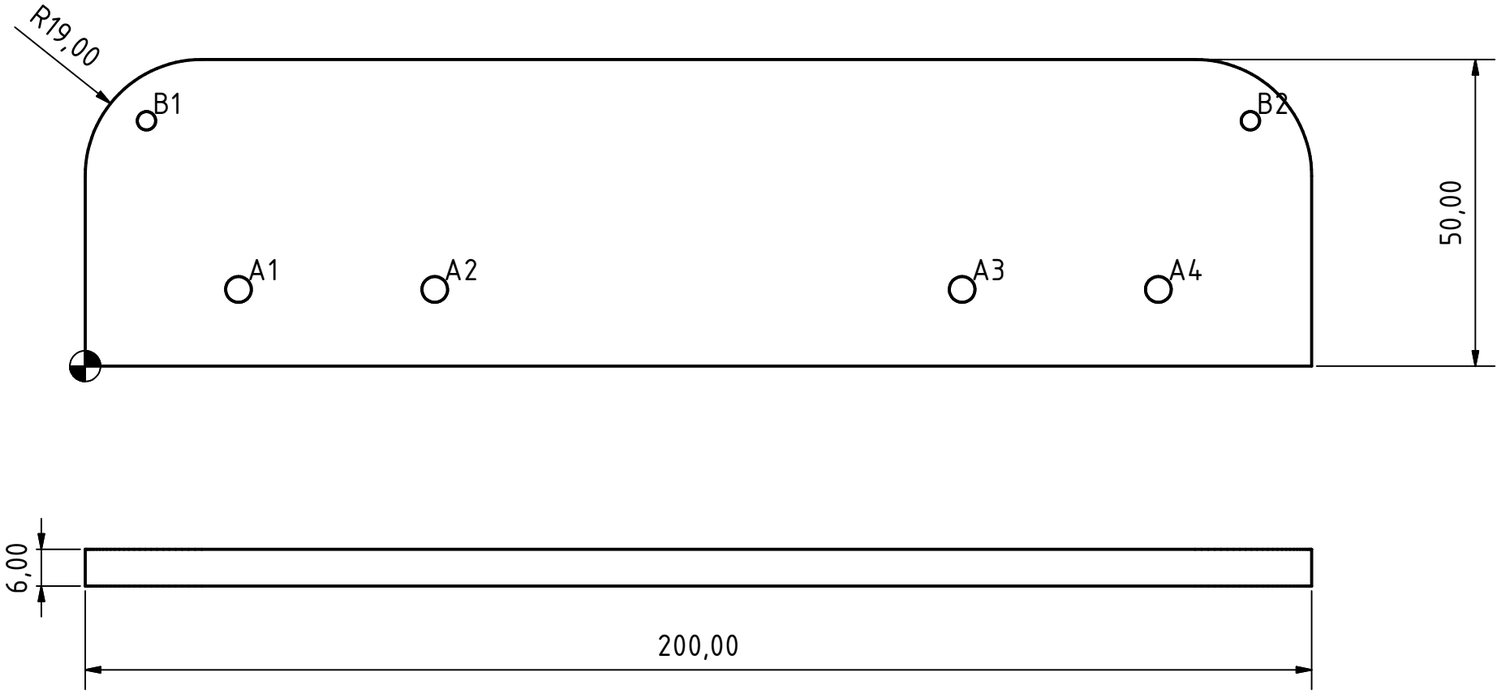
HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	40,00	35,00	Ø6,50 -6,00 DEEP
B1	130,00	35,00	Ø6,00 -6,00 DEEP

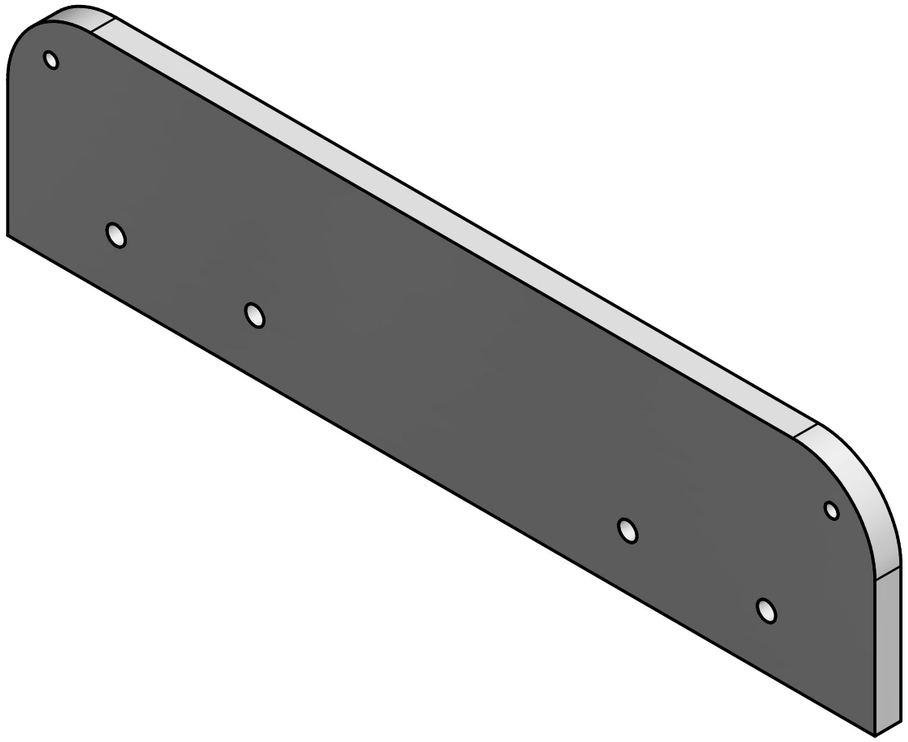
Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

Título: Frontal

			Nombre	Fecha
Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24051_Frontal.dwg	Revisado:	A. Bueno	02/06/2017
Esc: 1:1	Hoja 6 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com





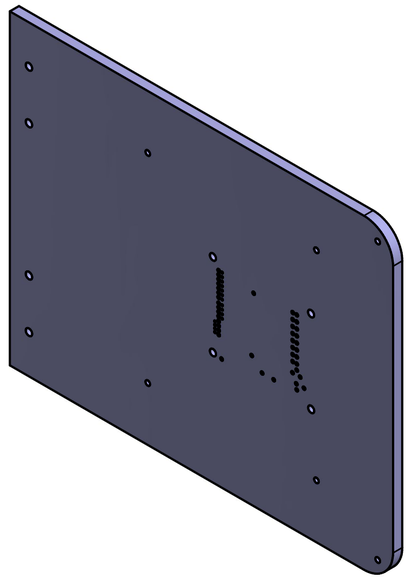
HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	25,00	12,50	Ø4,20 -6,00 DEEP
A2	57,00	12,50	Ø4,20 -6,00 DEEP
A3	143,00	12,50	Ø4,20 -6,00 DEEP
A4	175,00	12,50	Ø4,20 -6,00 DEEP
B1	10,00	40,00	Ø3,00 -6,00 DEEP
B2	190,00	40,00	Ø3,00 -6,00 DEEP

Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

Título: Bisagras

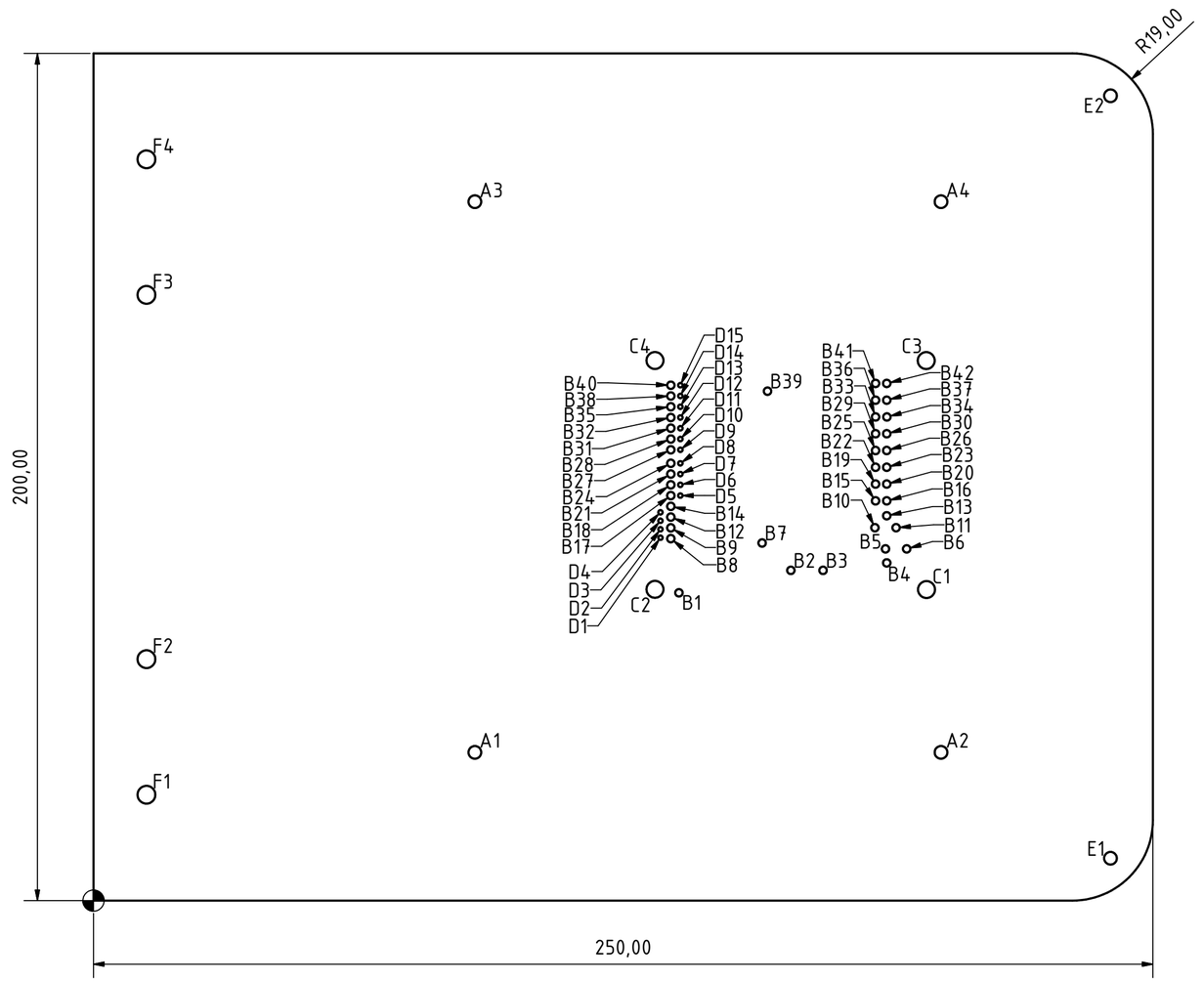
			Nombre	Fecha
Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24061_Bisagras.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:1	Hoja 7 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com



HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	90,00	35,00	Ø3,00 THRU
A2	200,00	35,00	Ø3,00 THRU
A3	90,00	165,00	Ø3,00 THRU
A4	200,00	165,00	Ø3,00 THRU
B1	138,15	72,67	Ø1,75 THRU
B2	164,57	77,96	Ø1,75 THRU
B3	172,16	77,96	Ø1,75 THRU
B4	187,19	79,73	Ø1,75 THRU
B5	186,90	83,04	Ø1,75 THRU
B6	191,90	83,04	Ø1,75 THRU
B7	157,79	84,43	Ø1,75 THRU
B8	136,24	85,46	Ø1,75 THRU
B9	136,24	88,00	Ø1,75 THRU
B10	184,40	88,04	Ø1,75 THRU
B11	189,40	88,04	Ø1,75 THRU
B12	136,24	90,54	Ø1,75 THRU
B13	187,19	90,86	Ø1,75 THRU
B14	136,24	93,08	Ø1,75 THRU
B15	184,55	94,38	Ø1,75 THRU
B16	187,19	94,38	Ø1,75 THRU
B17	136,24	95,62	Ø1,75 THRU
B18	136,24	98,16	Ø1,75 THRU
B19	184,55	98,34	Ø1,75 THRU
B20	187,19	98,34	Ø1,75 THRU
B21	136,24	100,70	Ø1,75 THRU
B22	184,55	102,30	Ø1,75 THRU
B23	187,19	102,30	Ø1,75 THRU
B24	136,24	103,24	Ø1,75 THRU
B25	184,55	106,26	Ø1,75 THRU
B26	187,19	106,26	Ø1,75 THRU
B27	136,24	106,43	Ø1,75 THRU
B28	136,24	108,97	Ø1,75 THRU
B29	184,55	110,22	Ø1,75 THRU
B30	187,19	110,22	Ø1,75 THRU
B31	136,24	111,53	Ø1,75 THRU
B32	136,24	114,05	Ø1,75 THRU
B33	184,55	114,18	Ø1,75 THRU
B34	187,23	114,18	Ø1,75 THRU
B35	136,24	116,59	Ø1,75 THRU
B36	184,55	118,14	Ø1,75 THRU
B37	187,19	118,14	Ø1,75 THRU
B38	136,24	119,13	Ø1,75 THRU
B39	159,06	120,27	Ø1,75 THRU
B40	136,24	121,67	Ø1,75 THRU
B41	184,55	122,10	Ø1,75 THRU
B42	187,19	122,10	Ø1,75 THRU

HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
C1	196,58	73,44	Ø4,00 THRU
C2	132,50	73,49	Ø4,00 THRU
C3	196,51	127,49	Ø4,00 THRU
C4	132,50	127,51	Ø4,00 THRU
D1	133,79	85,67	Ø1,05 THRU
D2	133,79	87,67	Ø1,05 THRU
D3	133,79	89,67	Ø1,05 THRU
D4	133,79	91,67	Ø1,05 THRU
D5	138,49	95,62	Ø1,05 THRU
D6	138,49	98,16	Ø1,05 THRU
D7	138,49	100,70	Ø1,05 THRU
D8	138,49	103,24	Ø1,05 THRU
D9	138,49	106,43	Ø1,05 THRU
D10	138,49	108,97	Ø1,05 THRU
D11	138,49	111,51	Ø1,05 THRU
D12	138,49	114,05	Ø1,05 THRU
D13	138,49	116,59	Ø1,05 THRU
D14	138,49	119,13	Ø1,05 THRU
D15	138,49	121,67	Ø1,05 THRU
E1	240,00	10,00	Ø3,00 -6,00 DEEP
E2	240,00	190,00	Ø3,00 -6,00 DEEP
F1	12,50	25,00	Ø4,20 -6,00 DEEP
F2	12,50	57,00	Ø4,20 -6,00 DEEP
F3	12,50	143,00	Ø4,20 -6,00 DEEP
F4	12,50	175,00	Ø4,20 -6,00 DEEP



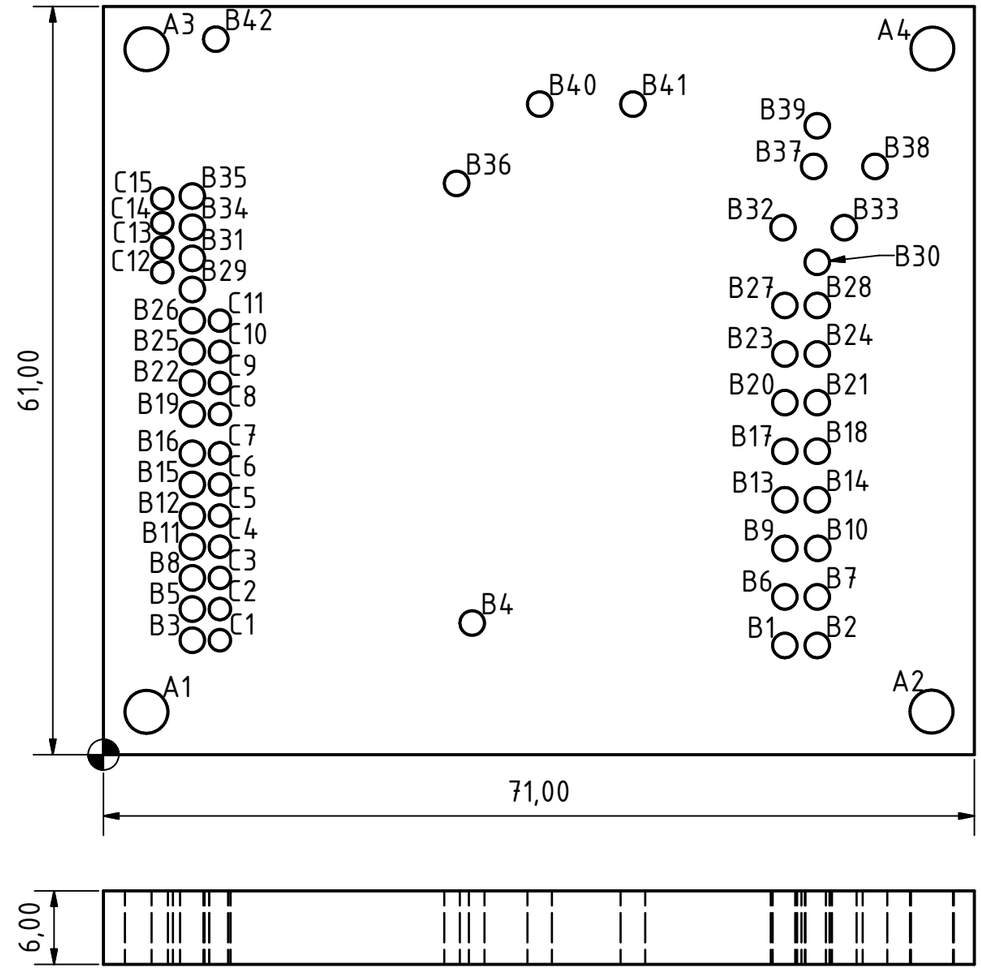
Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

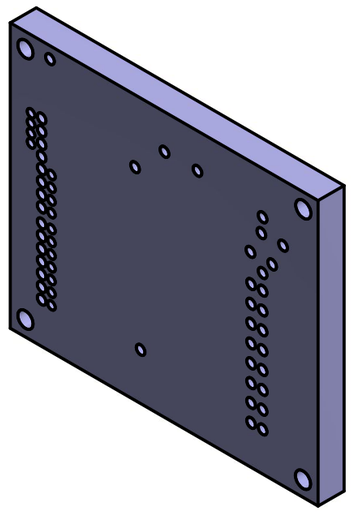
Título: Tapa superior

			Nombre	Fecha
Tamaño: A2	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24071_Tapa_Superior.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:1	Hoja 8 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº5
31350-Peralta Navarra
www.electronicafalcon.com

HOLE TABLE				HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION	HOLE	XDIM	YDIM	DESCRIPTION
A1	3,50	3,49	Ø3,50 -6,00 DEEP	B33	60,40	42,96	Ø2,00 -6,00 DEEP
A2	67,51	3,51	Ø3,50 -6,00 DEEP	B34	7,24	43,00	Ø2,00 -6,00 DEEP
A3	3,50	57,51	Ø3,50 -6,00 DEEP	B35	7,24	45,54	Ø2,00 -6,00 DEEP
A4	67,58	57,56	Ø3,50 -6,00 DEEP	B36	28,79	46,57	Ø2,00 -6,00 DEEP
B1	55,55	8,90	Ø2,00 -6,00 DEEP	B37	57,90	47,96	Ø2,00 -6,00 DEEP
B2	58,19	8,90	Ø2,00 -6,00 DEEP	B38	62,90	47,96	Ø2,00 -6,00 DEEP
B3	7,24	9,33	Ø2,00 -6,00 DEEP	B39	58,19	51,27	Ø2,00 -6,00 DEEP
B4	30,06	10,73	Ø2,00 -6,00 DEEP	B40	35,57	53,04	Ø2,00 -6,00 DEEP
B5	7,24	11,87	Ø2,00 -6,00 DEEP	B41	43,16	53,04	Ø2,00 -6,00 DEEP
B6	55,55	12,86	Ø2,00 -6,00 DEEP	B42	9,15	58,33	Ø2,00 -6,00 DEEP
B7	58,19	12,86	Ø2,00 -6,00 DEEP	C1	9,49	9,33	Ø1,75 -6,00 DEEP
B8	7,24	14,41	Ø2,00 -6,00 DEEP	C2	9,49	11,87	Ø1,75 -6,00 DEEP
B9	55,55	16,82	Ø2,00 -6,00 DEEP	C3	9,49	14,41	Ø1,75 -6,00 DEEP
B10	58,23	16,82	Ø2,00 -6,00 DEEP	C4	9,49	16,95	Ø1,75 -6,00 DEEP
B11	7,24	16,95	Ø2,00 -6,00 DEEP	C5	9,49	19,49	Ø1,75 -6,00 DEEP
B12	7,24	19,47	Ø2,00 -6,00 DEEP	C6	9,49	22,03	Ø1,75 -6,00 DEEP
B13	55,55	20,78	Ø2,00 -6,00 DEEP	C7	9,49	24,57	Ø1,75 -6,00 DEEP
B14	58,19	20,78	Ø2,00 -6,00 DEEP	C8	9,49	27,76	Ø1,75 -6,00 DEEP
B15	7,24	22,03	Ø2,00 -6,00 DEEP	C9	9,49	30,30	Ø1,75 -6,00 DEEP
B16	7,24	24,57	Ø2,00 -6,00 DEEP	C10	9,49	32,84	Ø1,75 -6,00 DEEP
B17	55,55	24,74	Ø2,00 -6,00 DEEP	C11	9,49	35,38	Ø1,75 -6,00 DEEP
B18	58,19	24,74	Ø2,00 -6,00 DEEP	C12	4,79	39,33	Ø1,75 -6,00 DEEP
B19	7,24	27,76	Ø2,00 -6,00 DEEP	C13	4,79	41,33	Ø1,75 -6,00 DEEP
B20	55,55	28,70	Ø2,00 -6,00 DEEP	C14	4,79	43,33	Ø1,75 -6,00 DEEP
B21	58,19	28,70	Ø2,00 -6,00 DEEP	C15	4,79	45,33	Ø1,75 -6,00 DEEP
B22	7,24	30,30	Ø2,00 -6,00 DEEP				
B23	55,55	32,66	Ø2,00 -6,00 DEEP				
B24	58,19	32,66	Ø2,00 -6,00 DEEP				
B25	7,24	32,84	Ø2,00 -6,00 DEEP				
B26	7,24	35,38	Ø2,00 -6,00 DEEP				
B27	55,55	36,62	Ø2,00 -6,00 DEEP				
B28	58,19	36,62	Ø2,00 -6,00 DEEP				
B29	7,24	37,92	Ø2,00 -6,00 DEEP				
B30	58,19	40,14	Ø2,00 -6,00 DEEP				
B31	7,24	40,46	Ø2,00 -6,00 DEEP				
B32	55,40	42,96	Ø2,00 -6,00 DEEP				



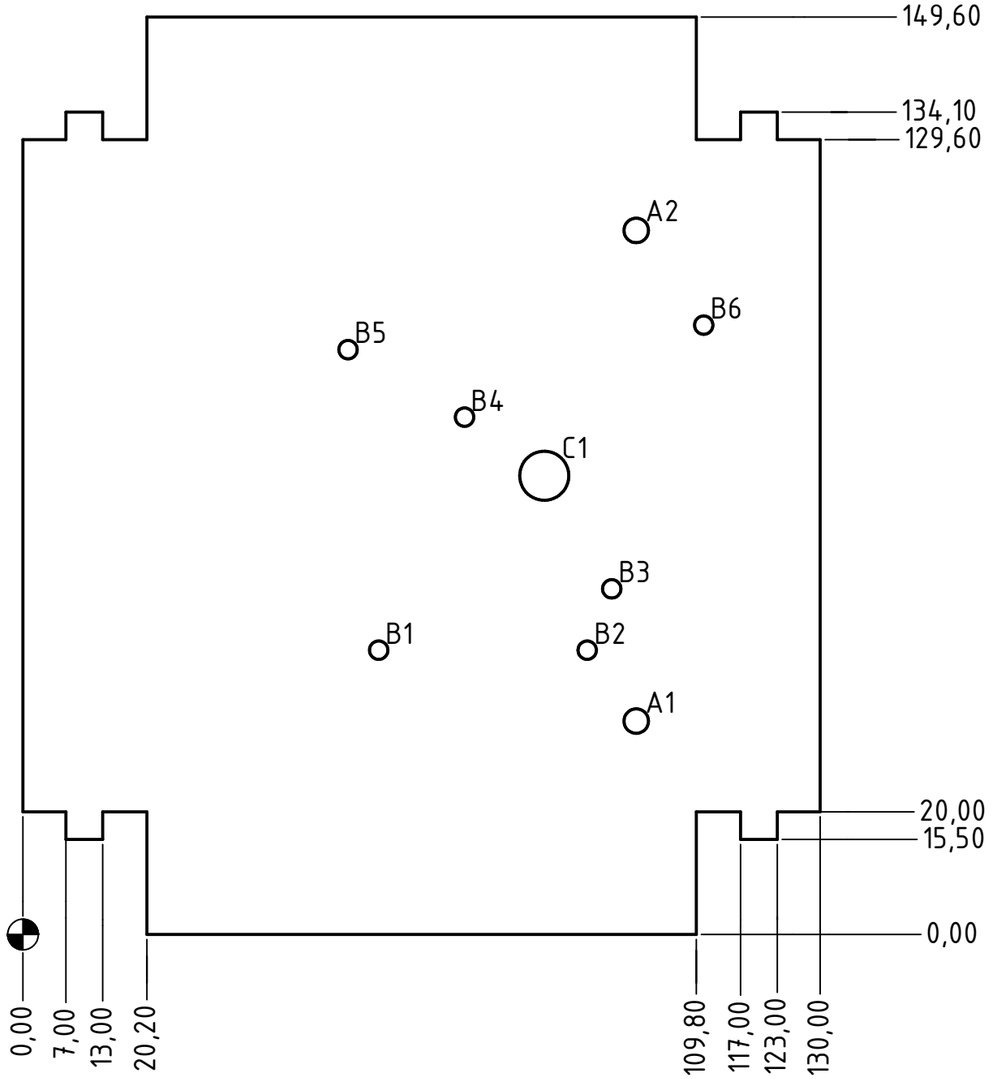


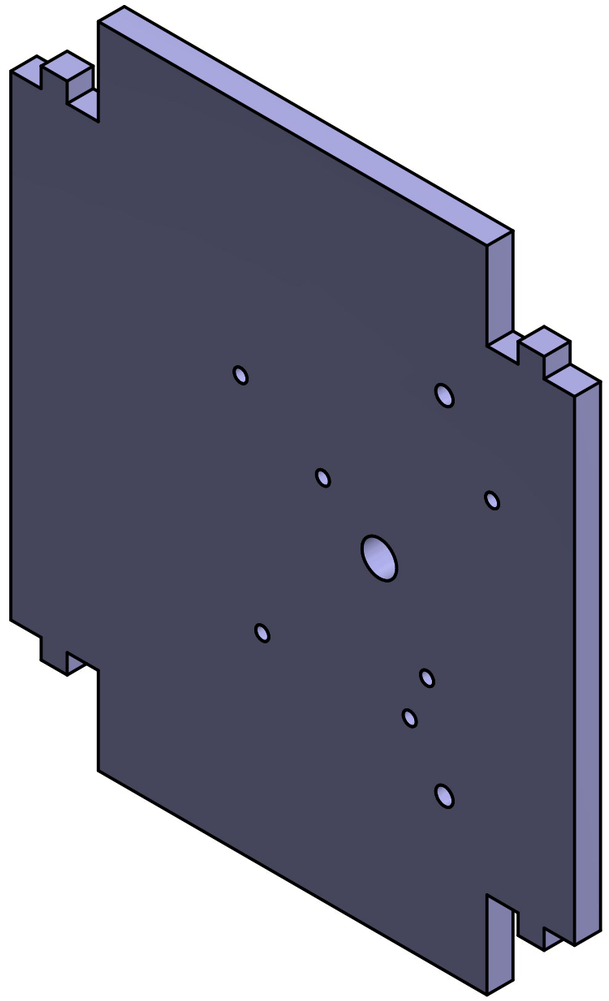
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano
Ver.	Fecha	Descripción

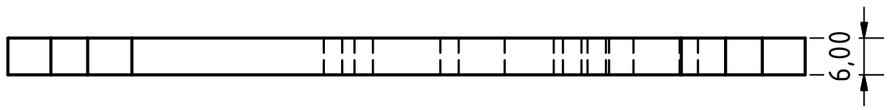
Título: Soporte tarjeta

Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha	FALCÓN electrónica
		Dibujado por:	W. De Avila	01/06/2017	
Ver: 1	Archivo: 24081_Soporte_Tarjeta.ipt	Revisado:	A. Bueno	02/06/2017	
Esc: 2:1	Hoja 9 de 14	Aprobado:	A. Bueno	02/06/2017	

Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com







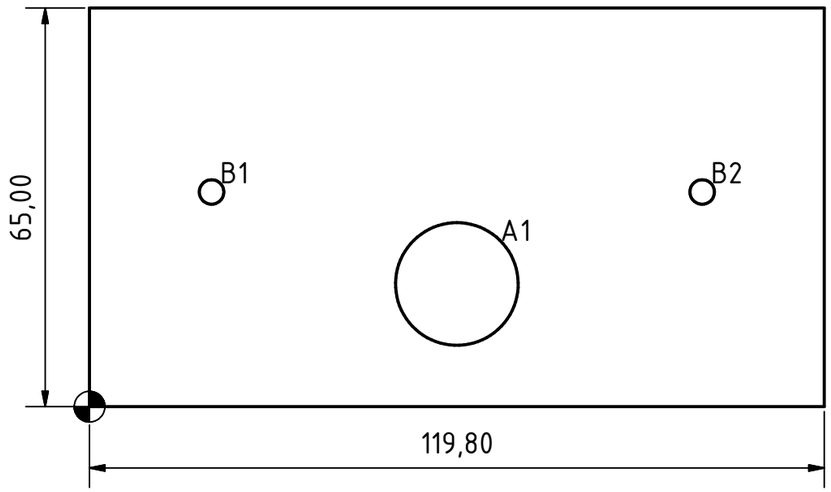
HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	100,00	34,80	Ø4,00 -6,00 DEEP
A2	100,00	114,80	Ø4,00 -6,00 DEEP
B1	58,00	46,36	Ø3,00 -6,00 DEEP
B2	92,00	46,36	Ø3,00 -6,00 DEEP
B3	96,00	56,36	Ø3,00 -6,00 DEEP
B4	72,00	84,36	Ø3,00 -6,00 DEEP
B5	53,00	95,36	Ø3,00 -6,00 DEEP
B6	111,00	99,36	Ø3,00 -6,00 DEEP
C1	85,00	74,80	Ø8,00 -6,00 DEEP

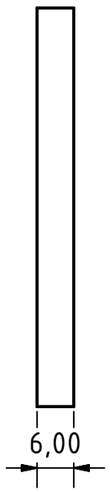
Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

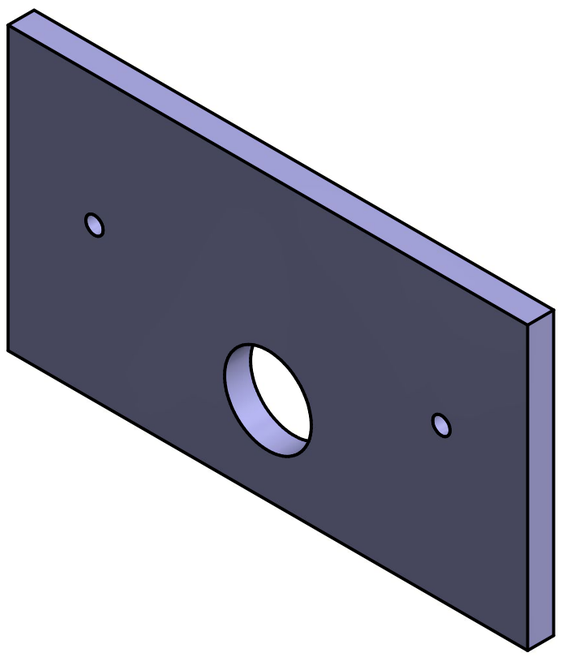
Título: Empujador

			Nombre	Fecha
Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV	Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24092_Empujador_Tarjeta.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:1	Hoja 10 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com





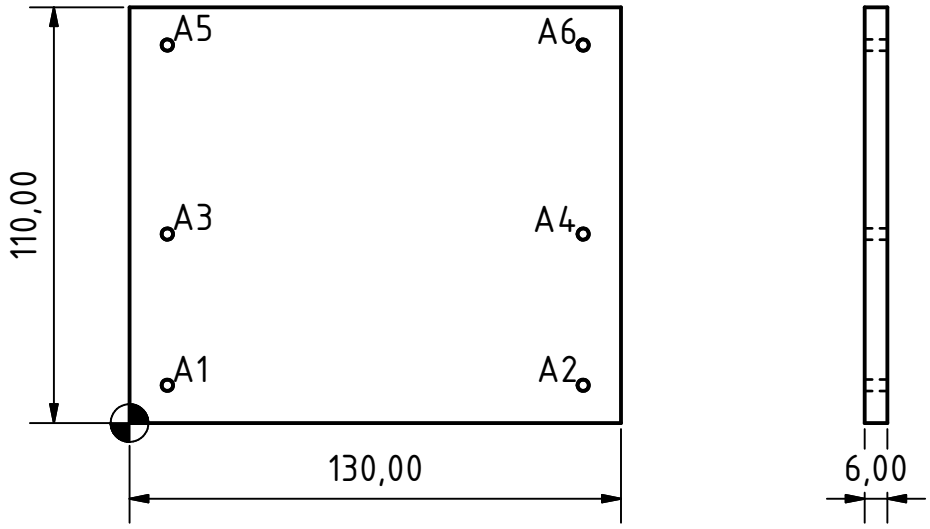


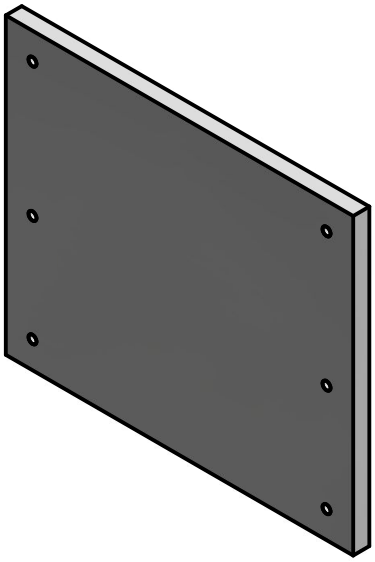
HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	59,90	20,00	Ø20,00 -6,00 DEEP
B1	19,90	35,00	Ø4,00 -6,00 DEEP
B2	99,90	35,00	Ø4,00 -6,00 DEEP

Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

Título: Soporte brida

Tamaño: A3	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha	FALCON electrónica
		Dibujado por:	W. De Avila	01/06/2017	Plgn. El Escopar Calle E nº1
Ver: 1	Archivo: 24101_Soporte_Brida.ipt	Revisado:	A. Bueno	02/06/2017	31350-Peralta Navarra
Esc: 1:1	Hoja 11 de 14	Aprobado:	A. Bueno	02/06/2017	www.electronicafalcon.com





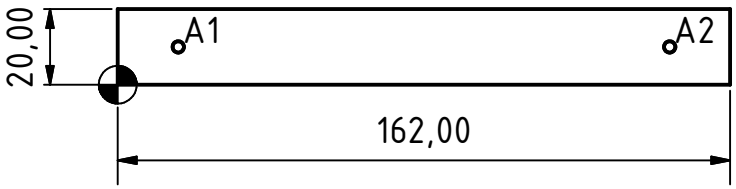
HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	10,00	10,00	Ø3,00 -6,00 DEEP
A2	120,00	10,00	Ø3,00 -6,00 DEEP
A3	10,00	50,00	Ø3,00 -6,00 DEEP
A4	120,00	50,00	Ø3,00 -6,00 DEEP
A5	10,00	100,00	Ø3,00 -6,00 DEEP
A6	120,00	100,00	Ø3,00 -6,00 DEEP

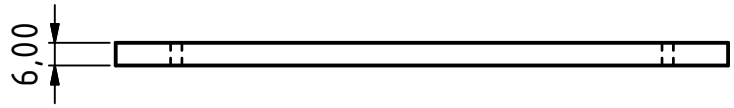
Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

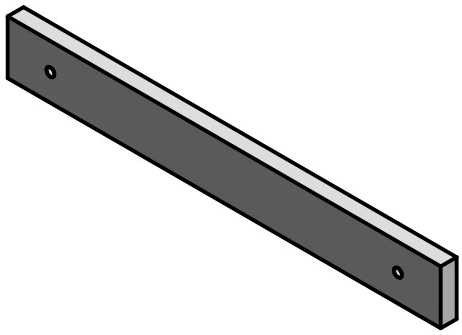
Título: Soporte lateral

Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha
		Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24111_Soporte_Lateral	Revisado:	A. Bueno	02/06/2017
Esc: 1:2	Hoja 12 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com







HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	16,00	10,00	Ø3,00 -6,00 DEEP
A2	146,00	10,00	Ø3,00 -6,00 DEEP

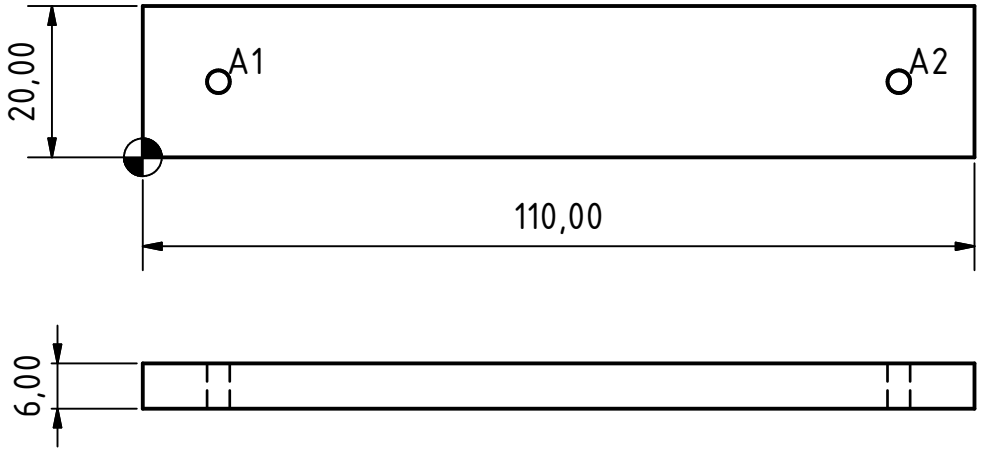
Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

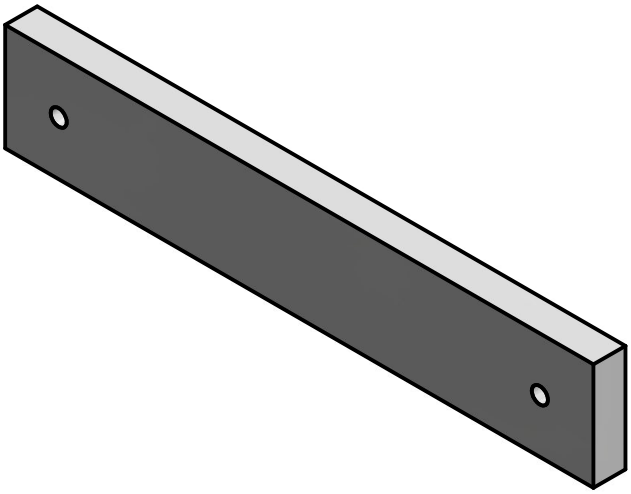
Título:

Soporte trasero

Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha
		Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24121_Soporte_Trasero.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:2	Hoja 13 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com





HOLE TABLE			
HOLE	XDIM	YDIM	DESCRIPTION
A1	10,00	10,00	Ø3,00 -6,00 DEEP
A2	100,00	10,00	Ø3,00 -6,00 DEEP

Ver.	Fecha	Descripción
1	01/06/2017	Se mejora la visualización
0	24/02/2017	Se crea el plano

Título: Base

Tamaño: A4	Referencia: 8CG07S0 8CG07S0-INV		Nombre	Fecha
		Dibujado por:	W. De Avila	01/06/2017
Ver: 1	Archivo: 24122_Soporte_Frontal.ipt	Revisado:	A. Bueno	02/06/2017
Esc: 1:1	Hoja 14 de 14	Aprobado:	A. Bueno	02/06/2017

FALCÓN electrónica
Plgn. El Escopar Calle E nº1
31350-Peralta Navarra
www.electronicafalcon.com