Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

Universidad de La Laguna

Escuela de Doctorado y Estudios de Posgrado

Desarrollo de mejoras para el robot Multi-Well Organ Bath (MuWOB)

Trabajo Final de Máster

Autor: Salvador Domínguez Durante

Tutor: Cándido Caballero Gil

Cotutor: Silvia Alayón Miranda

Sección: Ingeniería de Sistemas y Automática

Máster: Máster en Ingeniería Industrial

Mayo 2023

La publicación de este Trabajo Fin de Máster solo implica que el estudiante ha obtenido al menos la nota mínima exigida para superar la asignatura correspondiente no presupone que su contenido sea correcto, aunque si aplicable. En este sentido, la ULL no posee ningún tipo de responsabilidad hacia terceros por la aplicación total o parcial de los resultados obtenidos en este trabajo. También pone en conocimiento del lector que, según la ley de protección intelectual, los resultados son propiedad intelectual del alumno, siempre y cuando se haya procedido a los registros de propiedad intelectual o solicitud de patentes correspondientes con fecha anterior a su publicación.

RESUMEN

El estudio de los efectos de diferentes químicos en tejidos es un aspecto importante de la investigación farmacológica. Sin embargo, realizar experimentos de forma manual requiere grandes cantidades de tiempo y un alto nivel de precisión. De cara a resolver este problema, un robot capaz de mejorar la precisión y eficiencia de estos procesos, denominado MuWOB (Multi-Well Organ Bath), ha sido diseñado por el Departamento de Farmacología de la Universidad de La Laguna (ULL). En este Trabajo Fin de Máster (TFM) se estudia el estado actual del prototipo del robot y se plantean y desarrollan mejoras para el mismo, concretamente, el diseño de nuevas piezas mecánicas, el control de la cámara del prototipo mediante una Raspberry Pi, y la propuesta de un nuevo método de procesamiento de imagen basado en software libre.

PALABRAS CLAVE

Robot, Farmacología, Automatización, Diseño mecánico, Procesamiento de imágenes

ABSTRACT

Studying the effects of different chemicals on tissues is an important aspect of pharmacological research. However, performing experiments manually requires large amounts of time and a high level of precision. In order to solve this problem, a robot capable of improving the precision and efficiency of these processes, called MuWOB (Multi-Well Organ Bath), has been designed by the Department of Pharmacology of the University of La Laguna (ULL). This Master's Thesis (TFM) studies the current state of the robot prototype and proposes and develops improvements for it, specifically, the design of new mechanical parts, the control of the prototype's camera using a Raspberry Pi, and the proposal of a new image processing method based on free software.

KEYWORDS

Robot, Pharmacology, Automation, Mechanical Design, Image Processing

ÍNDICE

LISTA DE FIGURAS

1. INTRODUCCIÓN

1.1 Motivación y objeto

El estudio de los efectos de fármacos en diversos tejidos es un campo importante en la investigación farmacológica. El baño de órganos de múltiples pocillos es un instrumento muy utilizado para estudiar las propiedades farmacológicas de diversos fármacos en distintos tejidos. Sin embargo, realizar los experimentos manualmente puede llevar mucho tiempo y requiere un alto nivel de precisión.

Para solucionar este problema, el Departamento de Farmacología de la Universidad de La Laguna (ULL) ha diseñado un robot de baño de órganos multipocillo para automatizar el proceso (MuWOB), aumentando así la eficacia y precisión de los ensayos con fármacos.

El objetivo de este Trabajo Fin de Máster (TFM) es el estudio del estado actual del prototipo del robot, para plantear y desarrollar mejoras en el mismo. Concretamente, el diseño de nuevas piezas mecánicas, el control de la cámara del prototipo mediante una Raspberry Pi, y la propuesta de un nuevo método de procesamiento de imagen basado en software libre.

1.2 Descripción del robot MuWOB y estado actual de desarrollo

El funcionamiento del robot MuWOB se basa en la inyección de fármacos mediante pipetas automatizadas en una bandeja de pocillos (figura 1). En estos pocillos se encuentran los tejidos a estudiar. A lo largo del experimento los tejidos de cada pocillo son fotografiados con el fin de medir los cambios que los fármacos producen en estos. Todo este mecanismo se encuentra situado en una caja hermética que posee regulación de temperatura, para mantener la temperatura del cuerpo humano en el interior durante todo el experimento.

Al inicio de este TFM encontramos los diferentes componentes del robot, hardware y software, en estado avanzado de desarrollo. La parte de hardware se subdivide principalmente en dos componentes: el propio robot y una estufa hermética. El robot está compuesto de partes móviles que permiten inyectar el líquido por medio de pipetas capaces de moverse en el eje Y mediante motores paso a paso (figura 2). El robot se encuentra montado a falta de refinamiento y calibrado, y su placa de control está desarrollada sobre una protoboard (figura 3). En un futuro, esta electrónica de control se va a situar en la propia estufa, mediante la fabricación de un circuito impreso que contenga los diferentes circuitos. En la electrónica de control se encuentra una placa Arduino conectada a diferentes componentes del robot entre los que se encuentran los motores paso a paso que controlan el movimiento de las pipetas móviles y el

propio control de las bombas de líquido que inyectan el fármaco líquido en los pocillos.

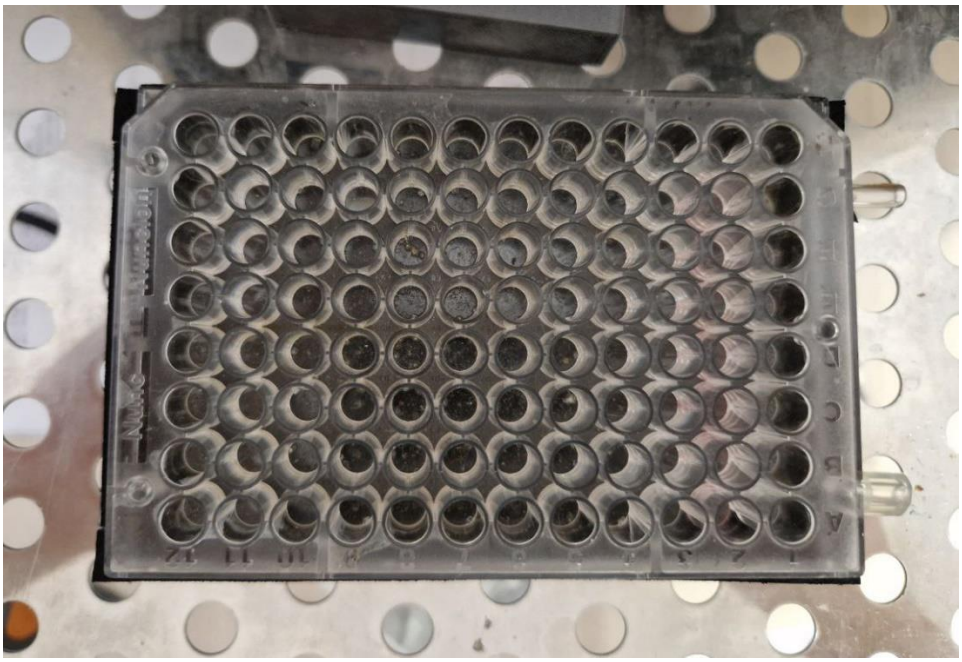


Figura 1: Bandeja de pocillos utilizada

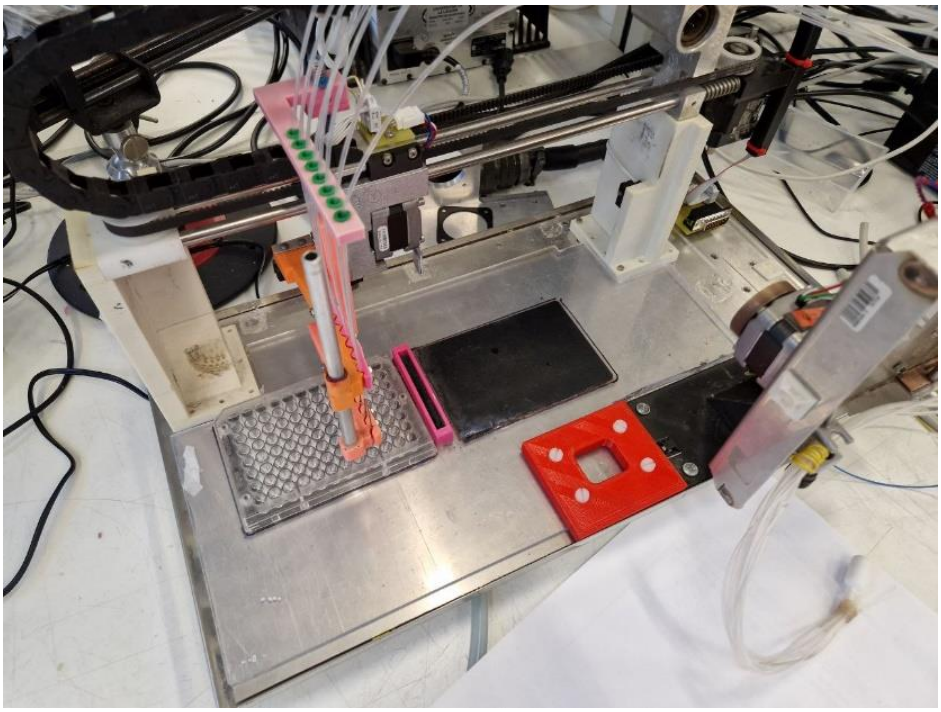


Figura 2: Robot de inyección de líquidos

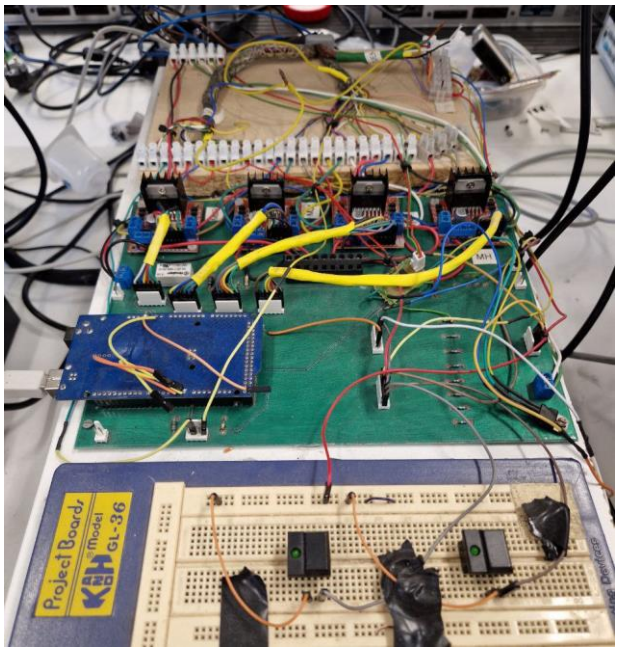


Figura 3: Electrónica de control del robot

El segundo componente hardware es una estufa hermética capaz de controlar la temperatura y humedad en su interior. El robot en su estado final se introducirá en la estufa, para controlar las variables ambientales durante los experimentos, y replicar el interior del cuerpo humano. En esta estufa se encuentra instalada una cámara en la parte superior, con el fin de obtener imágenes de los tejidos presentes en los pocillos. La estructura principal de la estufa ya está realizada, sólo falta solucionar problemas de cableado y diseñar una carcasa externa. Como se observa en la figura 4, se encuentra recubierta de paneles de cartón pegados con cinta adhesiva.

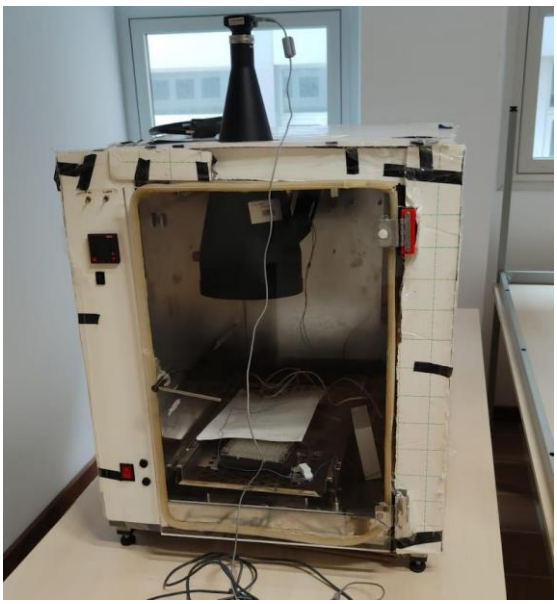


Figura 4: Prototipo de estufa

La parte software está compuesta por un programa de Matlab, que se ejecuta en un ordenador, y se encarga del procesamiento de las imágenes de los pocillos adquiridas por la cámara, un programa en C que controla el Arduino, cuya función es controlar los motores y bombas del robot, y una aplicación móvil para Android que permite controlar funciones básicas del robot como por ejemplo movimiento de las pipetas o el llenado y vaciado de las mismas. Tanto la aplicación en Android/Arduino como la de Matlab fueron desarrolladas previamente por otros alumnos. [1] [2]

El funcionamiento actual del robot es el siguiente: la aplicación móvil [1] se conecta vía Bluetooth al Arduino, el cual está preparado para ejecutar órdenes preprogramadas (figura 5). Estas órdenes controlan el movimiento de las pipetas móviles, así como la dosis del fármaco a inyectar. Por otro lado, se dispone de un ordenador con el programa desarrollado en Matlab instalado (figura 6). Con este programa se controla la cámara, pudiendo tomar fotografías de las muestras según los parámetros introducidos por el usuario. Este programa de Matlab fue desarrollado en una tesis [2]. El funcionamiento del prototipo está descrito en [3].

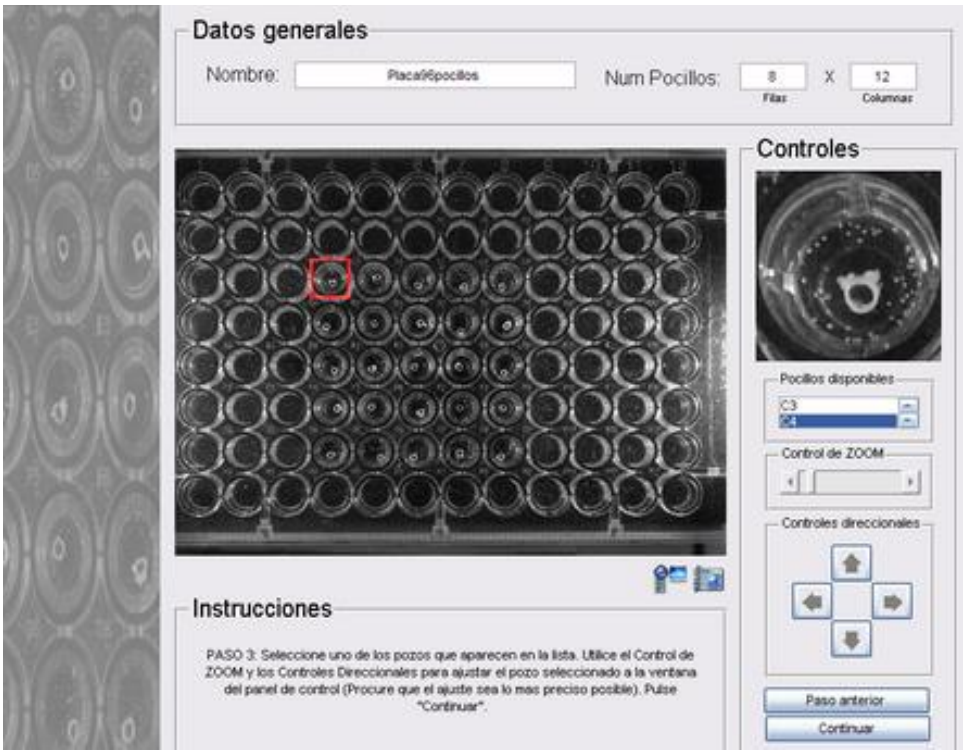


Figura 5: Programa en Matlab para la adquisición y procesamiento de imágenes de pocillos [1]

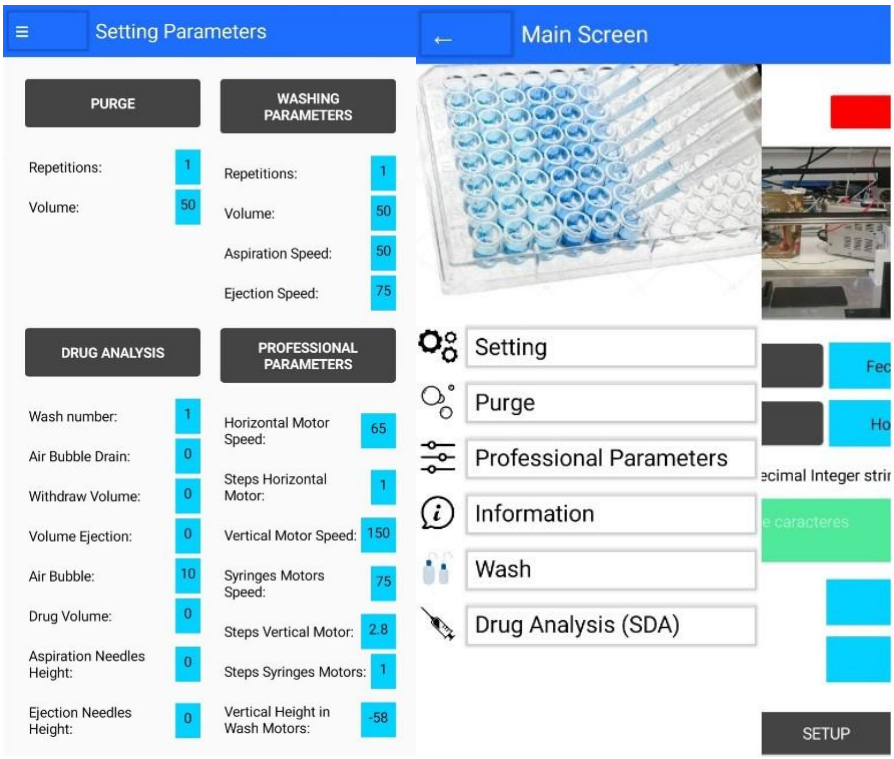


Figura 6: Aplicación Android conectada al Arduino para el control del robot [1]

Tras enumerar los diferentes componentes de hardware y software, podemos esquematizar el funcionamiento actual del robot como se muestra en la figura 7.

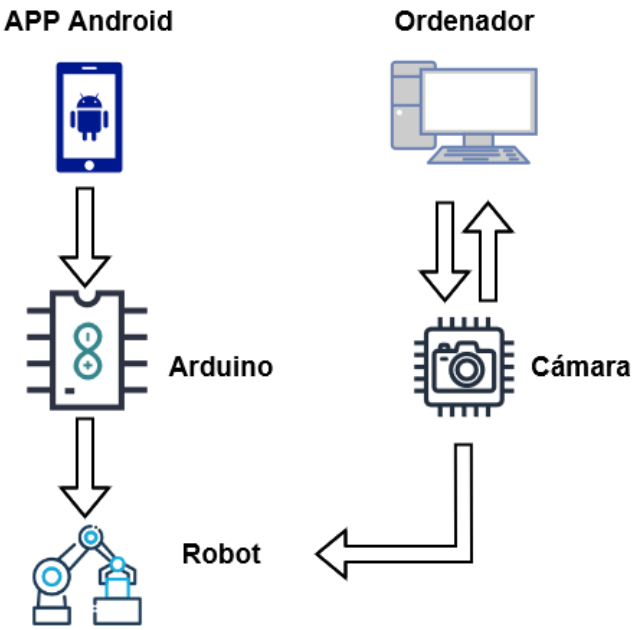


Figura 7: Esquema de funcionamiento actual del robot MuWOB.

Como se observa en la figura anterior, el ordenador es encargado de la toma de imágenes y su posterior procesamiento. Los inconvenientes de usar Matlab son los propios del uso de un software propietario, es necesario el pago de una licencia anual para su uso. También hay que destacar en nuestro caso los problemas de compatibilidad de programas desarrollados en versiones anteriores a la actual.

De cara a resolver este problema lo ideal sería desarrollar un software en código no propietario capaz de realizar la toma y procesamiento de las imágenes, así como la integración de un pequeño ordenador en el robot con el fin de no depender de uno externo y que el usuario pueda controlarlo si es necesario desde otro dispositivo.

1.3 Funcionamiento deseado del prototipo MuWOB

Para la toma de fotografías de los diferentes estados del tejido en distintos momentos se ha de desarrollar un software capaz de permitir al usuario decidir qué muestras necesita estudiar, así como el número de fotos necesarias y el tiempo entre la primera y la última.

El funcionamiento para la toma de imágenes consiste en el uso de una aplicación móvil para Android previamente programada que se comunica mediante Bluetooth con un Arduino que a su vez mediante conexión serial manda la orden a un ordenador, en nuestro caso una Raspberry Pi. Este ordenador se conecta a la cámara mediante conexión USB y se encarga tanto de mandar la orden a la cámara como del procesamiento posterior de las imágenes tomadas. El diagrama de funcionamiento descrito se puede observar en la siguiente figura:

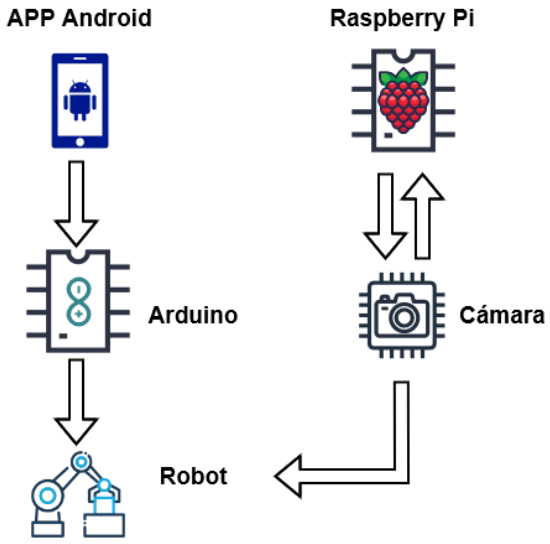


Figura 8: Diagrama de funcionamiento para la toma de fotografías

En este trabajo se desarrolla la implementación a partir de la Raspberry Pi, quedando como trabajo futuro la modificación del código ya realizado previamente en el Arduino y la aplicación Android para añadir la opción de toma de imágenes.

1.4 Metodología, objetivos y planificación del TFM

El objetivo principal de este TFM es la mejora del funcionamiento del robot MuWOB. Este objetivo se dividió en diferentes subobjetivos:

- Diseño de una carcasa externa para la estufa previamente descrita, con el fin de aislar completamente el interior. Además, el interior está recubierto de un material aislante potencialmente peligroso si se respira por lo que es necesario mejorar el aislamiento.

- Diseño de un soporte para la placa de pocillos. A la hora de adquirir fotografías de los pocillos, la posición de la bandeja de pocillos ha de ser la misma, con el fin de obtener la máxima precisión en las medidas realizadas. Mediante el diseño de un soporte físico se asegura que la placa de pocillos siempre se encuentre en la misma posición.

- Sustitución del ordenador de sobremesa por una Raspberry Pi 4, integrable en el prototipo. Con esta mejora hacemos que el prototipo no dependa del uso de un PC externo. El programa desarrollado en el PC, programado en Matlab, se utiliza para adquirir y procesar las fotografías de los pocillos. La idea es que estas funciones de adquisición y procesado las asuma ahora la Raspberry Pi 4.

- Desarrollo de un programa para la toma de fotografías usando lenguaje de programación abierto. Con el fin de no depender de un software propietario como es Matlab, se propone desarrollar un programa en Python capaz de conectarse a la cámara y tomar fotografías.

- Desarrollo de un programa para el procesamiento de imágenes de los pocillos y la consecuente medición de áreas de los tejidos contenidos en ellos, usando lenguaje de programación abierto. Usando Python, se programa un script capaz de analizar las imágenes previamente obtenidas y calcular la evolución de los tejidos analizados.

Tras la definición de metas a lograr, se estudiaron las diferentes publicaciones y trabajos realizados en este campo [3], destacando el trabajo realizado previamente por otros alumnos. [1]

Para alcanzar los diferentes objetivos propuestos, fue necesario estructurar y planificar las diferentes etapas del proyecto. Esta planificación debía ser flexible, ya que las tareas a realizar estaban sujetas a cambios y disponibilidad de recursos. Las diferentes etapas del TFM han sido las siguientes:

1. Estudio del estado del robot y del estado del arte.
2. Desarrollo e implementación de las mejoras mecánicas.
3. Desarrollo e implementación de las mejoras electrónicas y de programación.
4. Análisis de los resultados obtenidos con la implementación de las mejoras y conclusiones.
5. A partir de las conclusiones, definición de tareas propuestas para el futuro.

2. ESTADO DEL ARTE

Los robots de baño de órganos con múltiples pocillos son herramientas de laboratorio avanzadas que han revolucionado el estudio de la fisiología, la farmacología y la toxicología de los órganos. Estos robots están diseñados para automatizar los experimentos tradicionales de baño de órganos, lo que permite a los investigadores realizar estudios a gran escala y de alto rendimiento en múltiples muestras biológicas simultáneamente. [4]

En este apartado se hará un repaso de la evolución y del estado actual de los robots de baño de órganos con múltiples pocillos. Comenzaremos hablando de la historia de la técnica de baño de órganos, seguida de una visión general del diseño, la funcionalidad y las características clave de los robots de baño de órganos con múltiples pocillos. A continuación, examinaremos las aplicaciones actuales de estos robots en diversos campos de investigación, como el descubrimiento de fármacos, la toxicología y la farmacología. Por último, destacaremos algunos de los retos y limitaciones de la tecnología y discutiremos posibles desarrollos futuros en este campo.

2.1 Historia e inicios de la técnica del baño de órganos

La técnica del baño de órganos es una herramienta fundamental en la investigación farmacológica y fisiológica desde hace más de un siglo. Los primeros experimentos con baños de órganos fueron realizados a finales del siglo XIX por J.N. Langley, que utilizó tejidos aislados para estudiar los efectos

de los fármacos en diversos procesos fisiológicos [5]. Los experimentos de Langley consistían en suspender un trozo de tejido en una pequeña cámara llena de una solución nutritiva y conectarlo a un transductor de fuerza que medía la contracción y relajación del tejido en respuesta a diferentes estímulos.

Con el tiempo, la técnica del baño de órganos ha ido evolucionando hasta incluir diversas modificaciones y mejoras. Un avance significativo fue la introducción de la técnica de baño de órganos múltiples, que permitió a los investigadores estudiar los efectos de los fármacos en múltiples tejidos aislados simultáneamente. Esta técnica consistía en suspender varias piezas de tejido en cámaras separadas, cada una de ellas conectada a un transductor de fuerza y un amplificador que registraba la respuesta del tejido a diferentes estímulos. [6]

Otro avance importante fue la introducción de sistemas automatizados de baño de órganos, que permitieron a los investigadores realizar experimentos con mayor precisión y eficacia. [7]

2.2 Evolución del robot para baño de órganos con múltiples pocillos

El desarrollo del robot de baño de órganos con múltiples pocillos fue una evolución natural del sistema automatizado de baño de órganos. Estos robots están diseñados para automatizar todo el experimento de baño de órganos, desde la adición de fármacos al baño hasta el registro de la respuesta del tejido. La principal ventaja de estos robots es su capacidad para realizar experimentos con varias muestras a la vez, lo que aumenta enormemente el rendimiento y la eficacia de los experimentos. [7]

El primer robot de baño de órganos con múltiples pocillos fue desarrollado a principios de la década de 2000 por la empresa farmacéutica GlaxoSmithKline. El robot, denominado Multi-Organ Bath (MOB), consistía en una placa de 48 pocillos, cada uno de los cuales contenía una pequeña cámara para alojar un trozo de tejido. El robot podía realizar experimentos con hasta 48 muestras de tejido simultáneamente, lo que aumentaba enormemente el rendimiento de los experimentos.

Desde el desarrollo del MOB, varios grupos de investigación y empresas han desarrollado otros robots multipocillo para baños de órganos. Uno de los robots de baño de órganos con múltiples pocillos más utilizados es el sistema DanioVision, desarrollado por Noldus Information Technology, mostrado en las figuras 9 y 10. El sistema DanioVision está diseñado para estudiar el comportamiento de embriones y larvas de pez cebra e incluye una placa de 24

pocillos, cada uno de los cuales contiene una pequeña cámara para alojar un embrión o larva de pez cebra. [8]

Figura 9: Cámara de observación en el robot DanoVision [8]

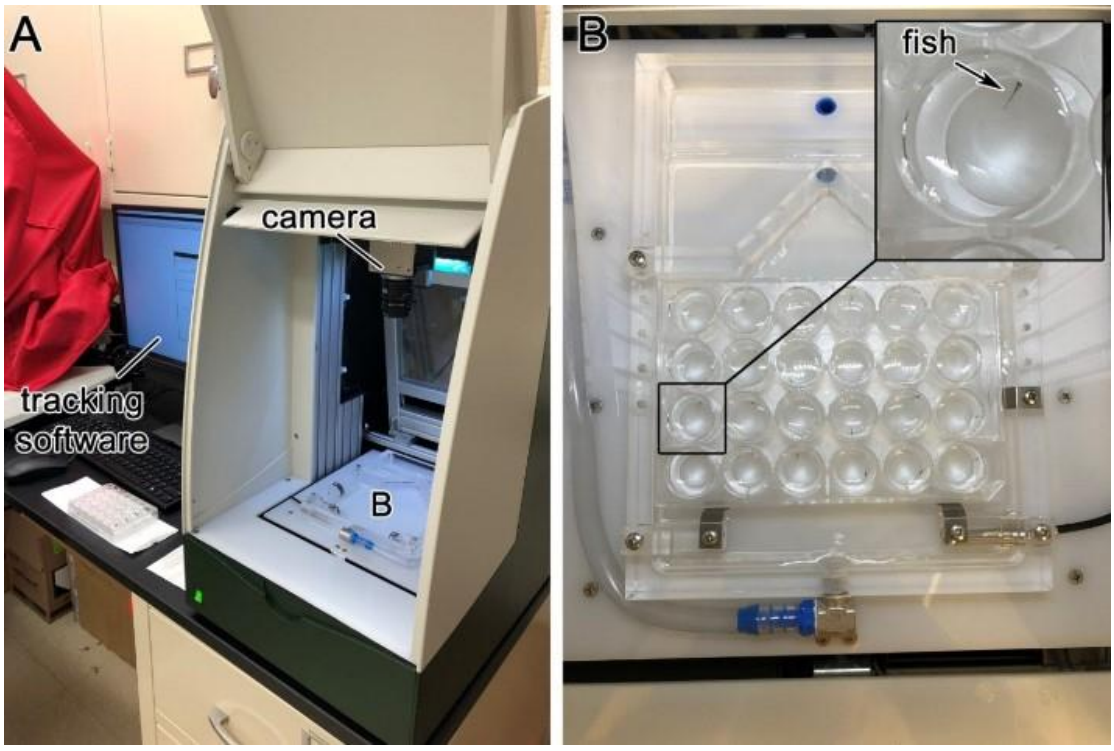


Figura 10: Exterior e interior de la cámara del robot Noldus DanioVision [9]

2.3 Aplicaciones y Limitaciones

Los robots de baño de órganos con múltiples pocillos tienen una amplia gama de aplicaciones en diversos campos de la investigación, como en el de la toxicología y la farmacología.

En el campo de la toxicología, estos robots son capaces de realizar experimentos con un gran número de muestras simultáneamente, lo que permite a los investigadores estudiar los efectos tóxicos de distintos compuestos en diversos tejidos y órganos. Esto es especialmente útil en el desarrollo de nuevos fármacos, ya que permite a los investigadores identificar posibles problemas de toxicidad en una fase temprana del proceso de desarrollo del fármaco.

Así mismo, también se utilizan mucho en el campo de la farmacología, estudiando los efectos de distintos fármacos en tejidos y órganos específicos, lo que permite a los investigadores comprender mejor la farmacocinética y la farmacodinámica de diferentes fármacos. Esta información puede utilizarse para optimizar la dosificación de fármacos y desarrollar nuevos medicamentos con perfiles terapéuticos mejorados. [10]

A pesar de sus numerosas ventajas, estos robots también presentan algunas limitaciones y retos que es necesario abordar. Uno de ellos es la variabilidad de las respuestas de los tejidos, que puede verse afectada por diversos factores, como la edad, la procedencia y las condiciones de almacenamiento de los tejidos. Además, el uso de tejidos aislados puede no reproducir completamente las complejas interacciones que se producen in vivo, lo que dificulta la predicción precisa de los efectos de los fármacos en humanos. [7]

Otro reto es la complejidad del análisis de datos, sobre todo en experimentos a gran escala que generan una gran cantidad de datos. El desarrollo de nuevas herramientas analíticas y algoritmos será crucial para el análisis eficaz de estos conjuntos de datos.

3. MEJORAS IMPLEMENTADAS

Como se ha mencionado previamente, el objetivo principal de este trabajo fin de máster es el de la mejora del funcionamiento del prototipo MuWOB así como el desarrollo de piezas necesarias para el mismo. También se planteó el desarrollo de software para la conexión, adquisición y tratamiento de imágenes, usando un lenguaje de programación no propietario.

Las mejoras implementadas se han dividido en dos grupos principales: mejoras hardware y mejoras software. En las mejoras hardware se incluye el desarrollo de piezas mecánicas necesarias para el correcto funcionamiento del robot. En las mejoras software se recogen varios programas escritos en lenguaje de programación Python para realizar diferentes funciones: conexión de la cámara a la Raspberry Pi, adquisición de imágenes, y procesamiento automatizado de imágenes para la toma de medidas de áreas de los tejidos a estudiar.

3.1 Diseño de paneles exteriores

El prototipo de estufa se encuentra cubierto por paneles plásticos de papel pluma. Esta solución temporal fue adaptada para proteger los componentes internos y aislar la lana de roca del exterior. Para su uso final, esta solución no es válida, ya que los paneles son débiles y están sujetos con cinta adhesiva (figura 11 y 12). Se propone el diseño y fabricación de paneles de chapa metálica unidos a la estructura de la estufa por medio de tornillos.

Figura 11: Estufa con paneles de papel pluma

Figura 12: Estufa sin paneles

Para el diseño de estos paneles se realizó un proceso exhaustivo de medición en la estructura metálica de la estufa. Tras conocer las diferentes dimensiones y usando un software de modelado Solidworks 2022 [11], se diseñaron los diferentes paneles a medida (figura 13). Estos paneles poseen un grosor de 1 mm, aunque esta medida puede ser sujeta a modificaciones según la disponibilidad del material. Las piezas diseñadas fueron las siguientes:

- Cubierta Lateral
- Cubierta Superior
- Cubierta Trasera
- Cubierta Frontal
- Cubierta Inferior

Se propone como material a utilizar el acero inoxidable, debido a sus propiedades, ya que es resistente a los cambios de temperatura y humedad que se van a producir en el interior de la estufa, evitando la creación de óxido.

Los planos de las piezas diseñadas pueden consultarse en el Anexo situado en el final del documento.

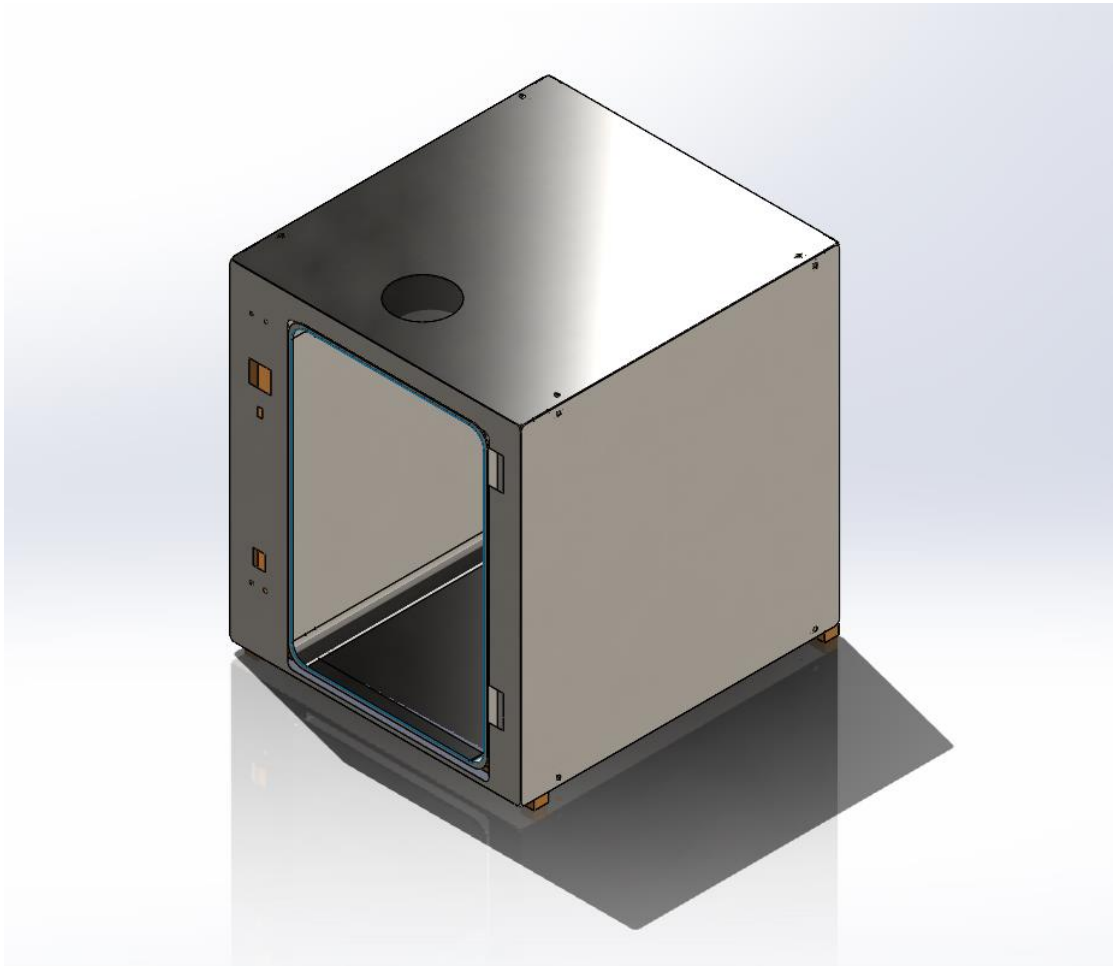


Figura 13: Paneles diseñados montados en el conjunto de la estufa

La fabricación de estos paneles no se realizó por limitaciones presentes en el taller de mecanizado de la Universidad de La Laguna. La gestión y compra de estos paneles a una empresa externa a la universidad queda fuera del alcance de este TFM. Los planos y archivos CAD quedan a disposición de los responsables del laboratorio de electrónica de la Universidad de La Laguna para su posterior fabricación.

3.2 Diseño de soporte para pocillos

Con el fin de asegurar la posición de la bandeja de pocillos para tener siempre la misma situación de estos en las fotografías, se diseña una pieza plástica que sujete el módulo descrito (figura 14).

Para esto, se va a utilizar el mismo software de CAD previamente mencionado [11]. Tras la toma de medidas pertinentes, se diseña la pieza, obteniendo un

archivo STL, el cual va a ser posteriormente utilizado por un software de impresión 3D. El plano de la pieza puede ser encontrada en el Anexo

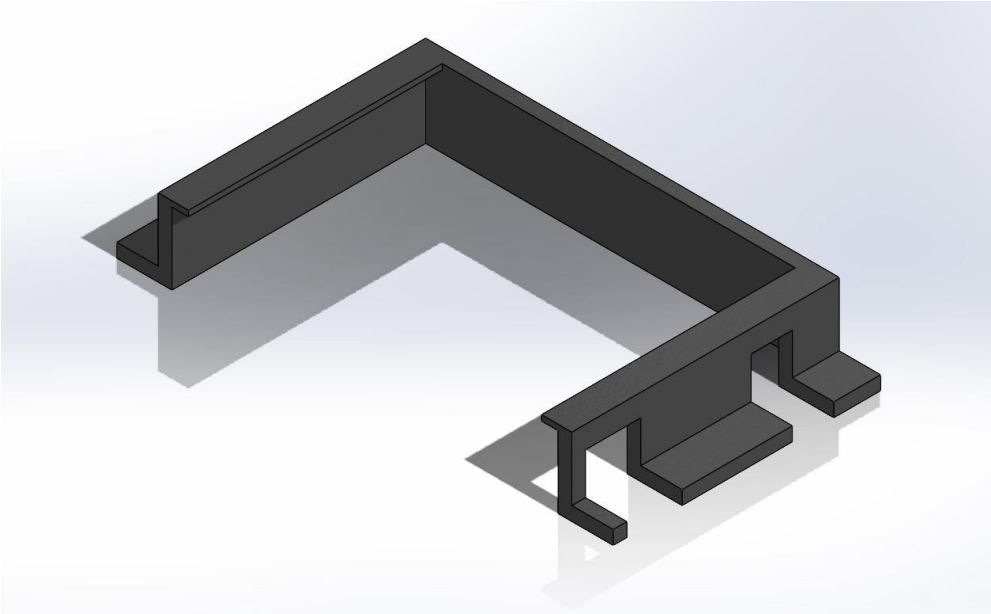


Figura 14: Diseño del soporte de pocillos

Tras completar el diseño, se procesa el mismo con el software de impresión 3D, en este caso se ha utilizado Cura [12] como se muestra en la figura 15

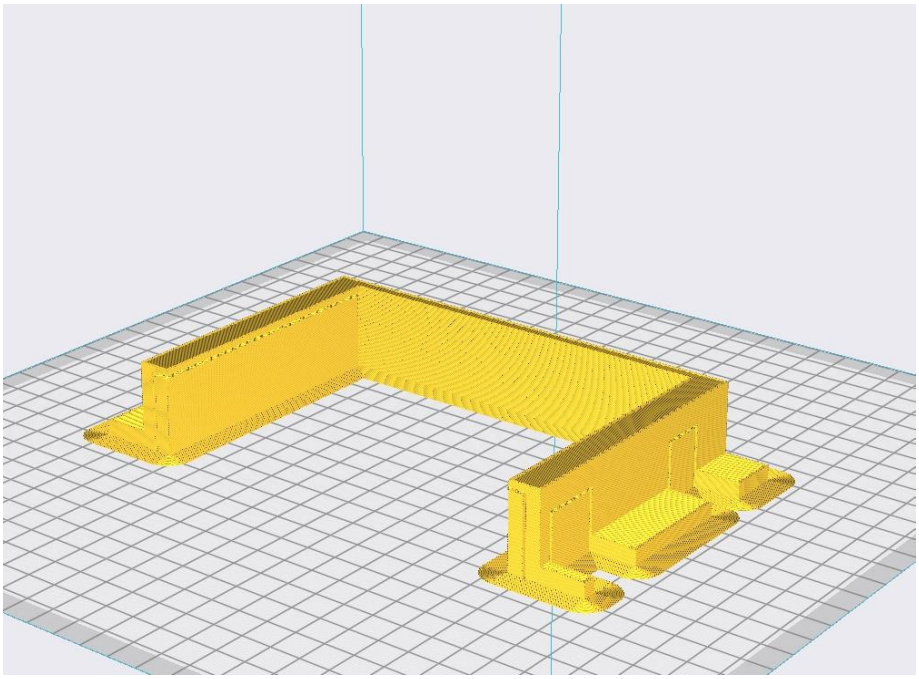


Figura 15: Soporte procesado en Cura

En este software se pueden configurar los diferentes parámetros y la impresora a utilizar. Las configuraciones seleccionadas para la impresión buscan obtener una buena relación entre la calidad de acabado y la duración de la impresión. En nuestro caso, el tiempo de impresión fue de 6 horas 45 minutos, y los parámetros utilizados se pueden consultar en la figura 16.

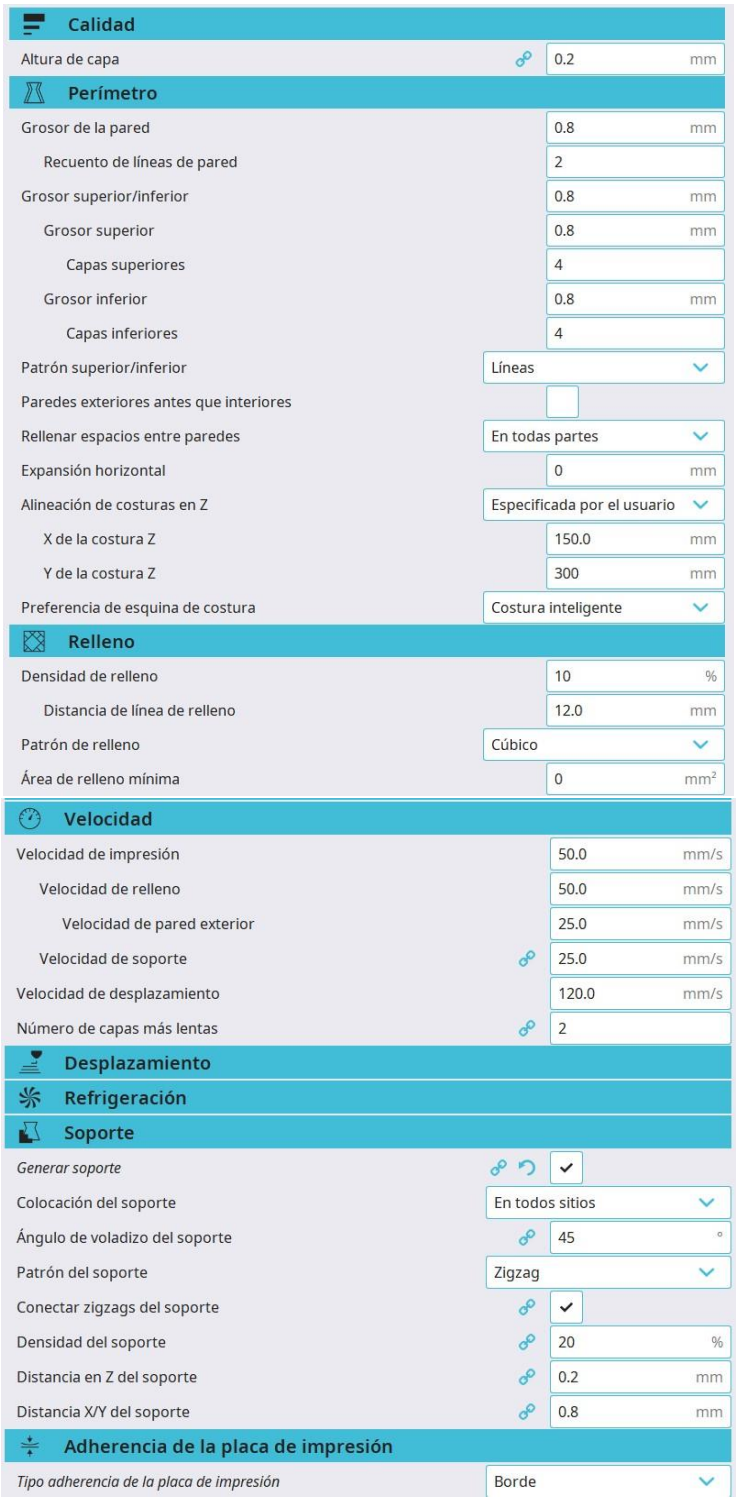


Figura 16: Parámetros de impresión 3D en Cura

Para la impresión se va a utilizar una impresora Creality CR-10s, usando PLA como material de impresión (figura 17). La impresora forma parte de la biblioteca de la facultad de Ingeniería Informática en la Universidad de La Laguna.

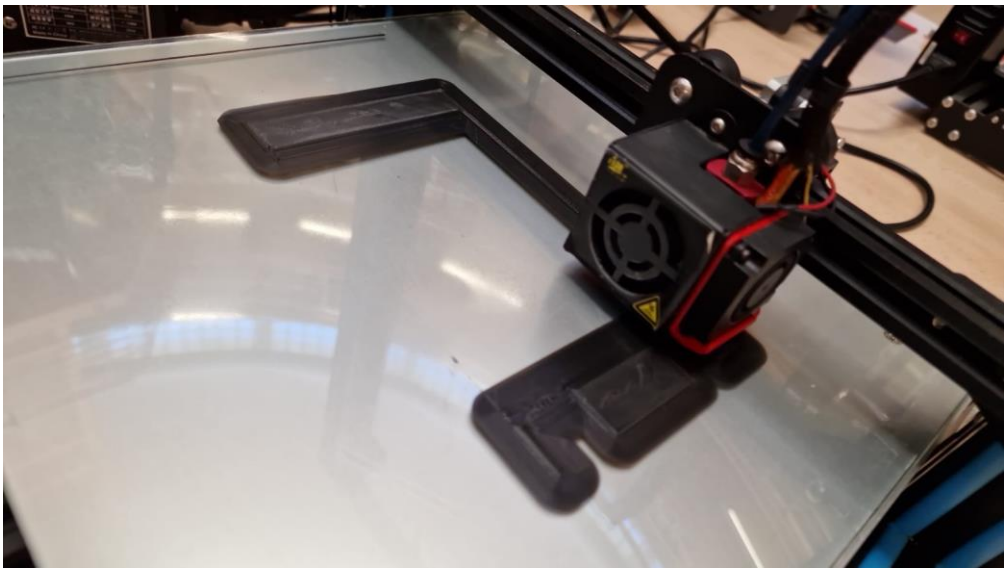


Figura 17: Impresión del soporte de pocillos

Tras la impresión se separa la pieza terminada de la base caliente y se inspecciona en busca de fallos en la impresión. En la figura 18 se puede observar el soporte terminado y colocado en su posición final, con una bandeja de pocillos colocada en su interior.

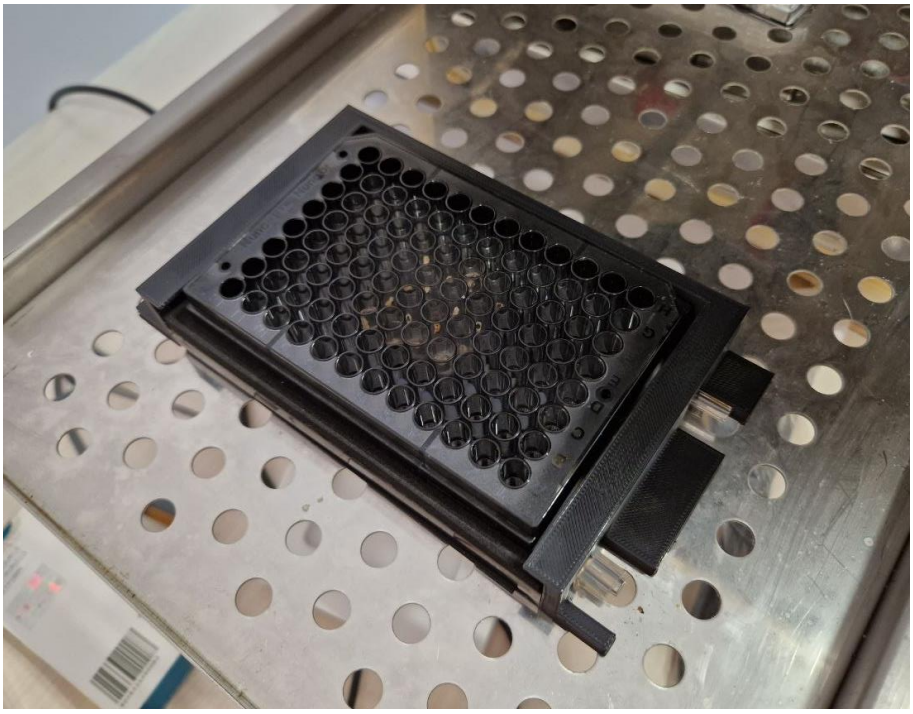


Figura 18: Soporte de pocillos instalado

3.3 Puesta en marcha de la cámara

Para la adquisición de imágenes el prototipo utiliza una cámara RGB del fabricante iDs [8], cuyas características más importantes se muestran en la tabla 1.

Fabricante	iDs
Modelo	UI-1490LE-C-HQ
Sensor	Onsemi MT9J003 1/2" CMOS
Resolución	3840 x 2748 (10,55 Mpx)
Frecuencia de imagen	3,2 fps

Tabla 1: Características de la cámara instalada en el prototipo.

Esta cámara se encuentra instalada en la parte superior de la estufa, en el exterior de esta (figura 19). A través de una abertura en la estufa, el objetivo de la cámara se introduce en el interior de la misma, de forma que la bandeja de pocillos quede centrada con la cámara.

Esta cámara es necesaria para el estudio de los diferentes cambios que sufre un tejido al reaccionar a un químico determinado. Analizando las imágenes tomadas por la cámara en una frecuencia fijada es posible detectar las variaciones pre y post tratamiento.

Figura 19: Cámara iDS en la parte superior de la estufa

En el prototipo inicial esta cámara es controlada por un script programado en Matlab. Para el desarrollo de un software en código abierto capaz de controlar la cámara se comprobó previamente el funcionamiento de la misma con la instalación del software provisto por el fabricante [13]. Este software nos permite comprobar el funcionamiento de la cámara así como la posición de las muestras. Este software, el cual es proporcionado por el fabricante de forma gratuita, fue ejecutado en un ordenador portatil conectado a la cámara. Se tomaron diferentes fotografías para comprobar el funcionamiento de la cámara (figura 20). Este software no se utilizará en el programa final, solo sirve a modo de comprobación del funcionamiento de la cámara.

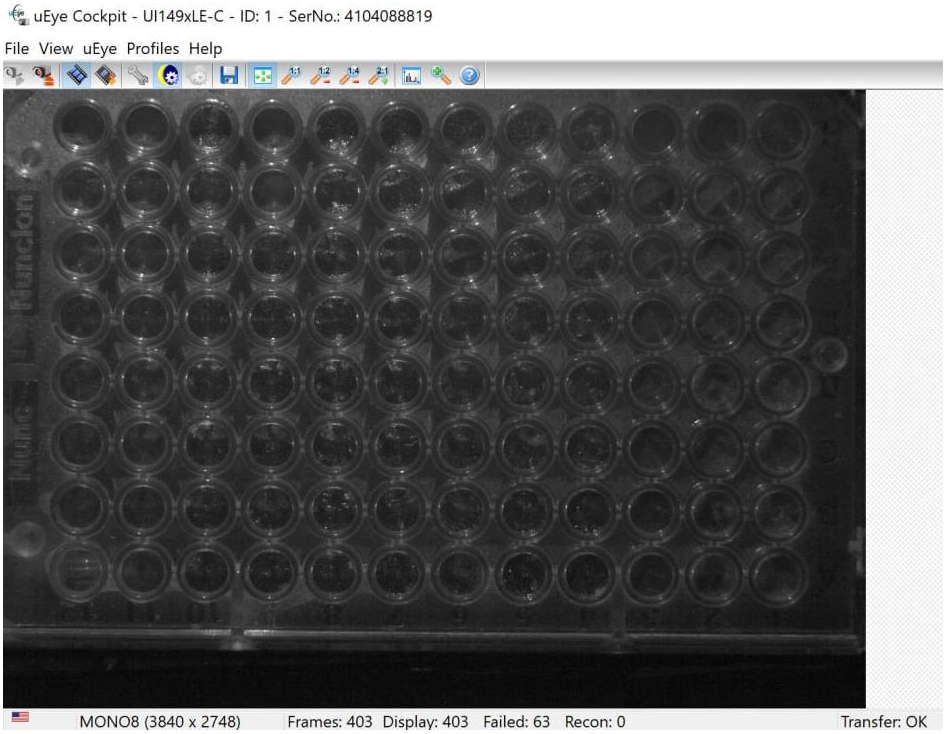


Figura 20: Imagen de los pocillos obtenida con el software uEye [13]

Se detectó una falta de iluminación en el interior de la estufa lo que provocaba que las imágenes no tuvieran el brillo necesario para ser procesadas. Con fin de solucionar este problema se instaló un foco de tipo led dentro de la estufa de manera provisional, con el objetivo de tener una luminosidad más adecuada (figura 21). Esta luz led se encuentra en un lado de la bandeja de muestras, pero no es la situación ideal, porque en las fotos se aprecian diferencias de iluminación de un lado a otro de la bandeja. Lo ideal sería poder instalar la luz en la parte superior de la estufa, de manera que la luz incida de manera uniforme en todos los pocillos. Debido a limitaciones de recursos no se ha podido desarrollar un sistema definitivo de iluminación. Este apartado se propone como un posible trabajo futuro.

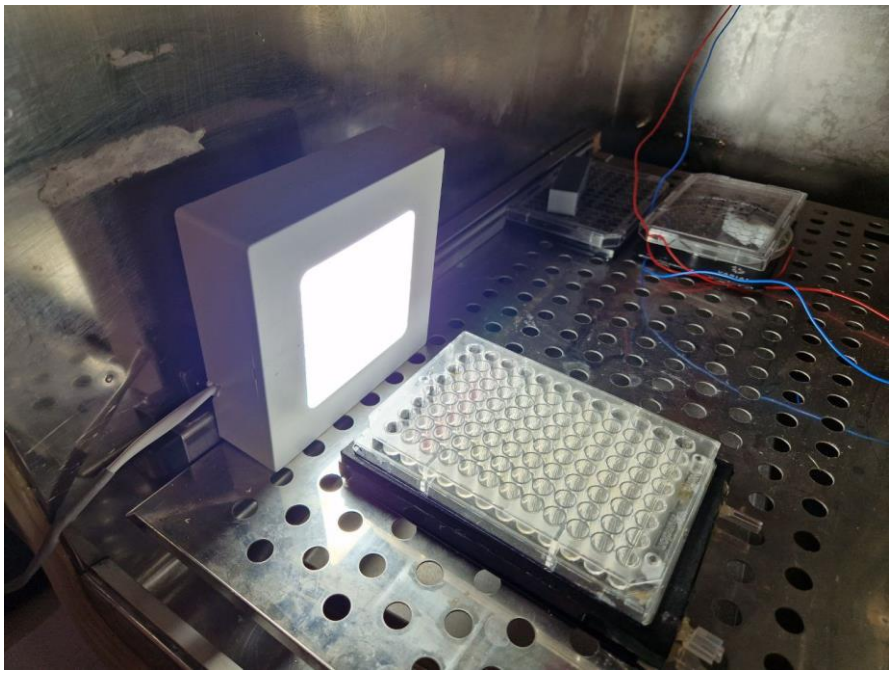


Figura 21: Instalación de iluminación LED temporal

Con el fin de obtener una imagen de cómo se verían las diferentes muestras, se usan granos de arroz en diferentes pocillos. Usando el software previamente mencionado [13], se obtiene una imagen lista para procesar (figura 22).

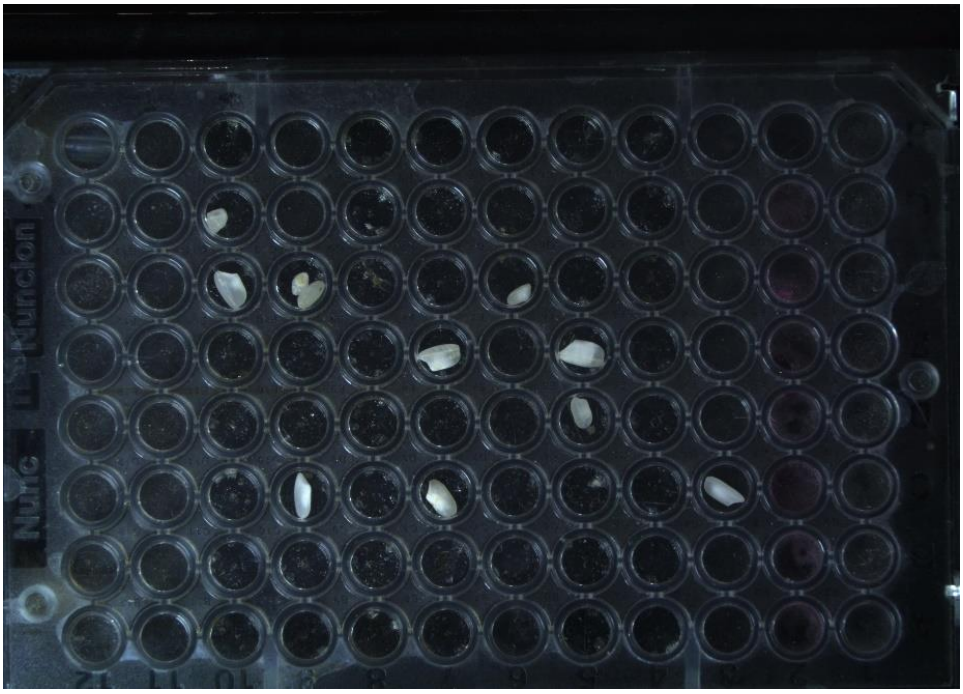


Figura 22: Imagen de pocillos obtenida con el software uEye

3.4 Instalación y puesta en marcha de Raspberry Pi 4

Para la conexión con la cámara, el control de esta y el procesamiento de imágenes se necesita un ordenador capaz de realizar estas funciones. Nuestro interés es integrar el equipo en el mismo prototipo. Por este motivo se selecciona un pequeño ordenador, modelo Raspberry Pi 4 [14] (figura 23), con las siguientes especificaciones:

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 2 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- Supports dual HDMI display output up to 4Kp60
- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)

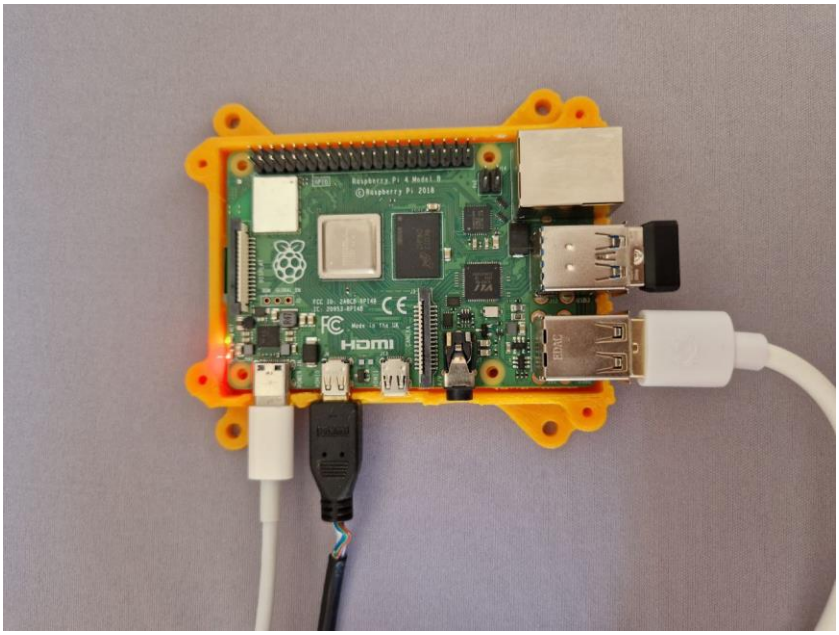


Figura 23: Raspberry Pi 4 utilizada en este TFM.

Se comienza con la instalación del sistema operativo en la Raspberry Pi, concretamente el sistema operativo Raspibian, también conocido como Raspberry Pi OS [15]. Raspbian es un sistema operativo libre basado en la distribución de Linux denominada Debian y optimizado para el hardware de Raspberry Pi. En este paso es necesario conectar la tarjeta micro SD a un ordenador externo (figura 24).

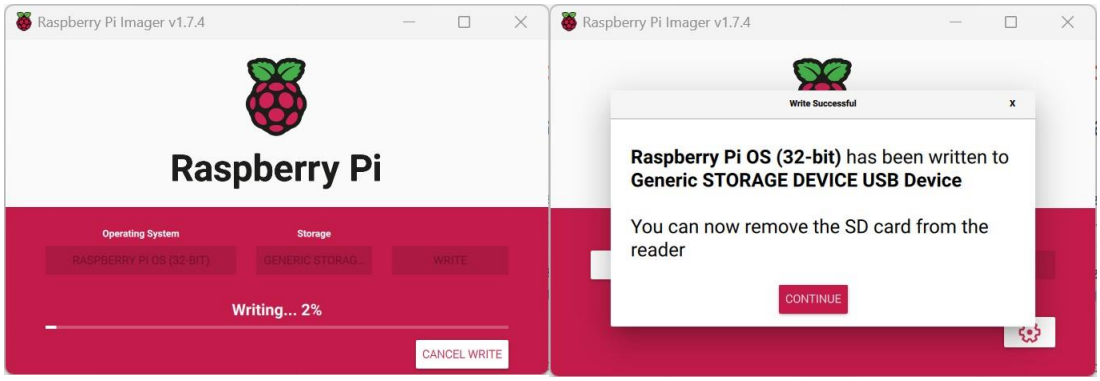


Figura 24: Preparación de Raspibian en tarjeta micro SD

Tras la escritura del sistema operativo en la tarjeta micro SD, se procede a su instalación y configuración en la Raspberry Pi. Para ello es necesario conectar teclado y ratón a la Raspberry Pi (figura 25).

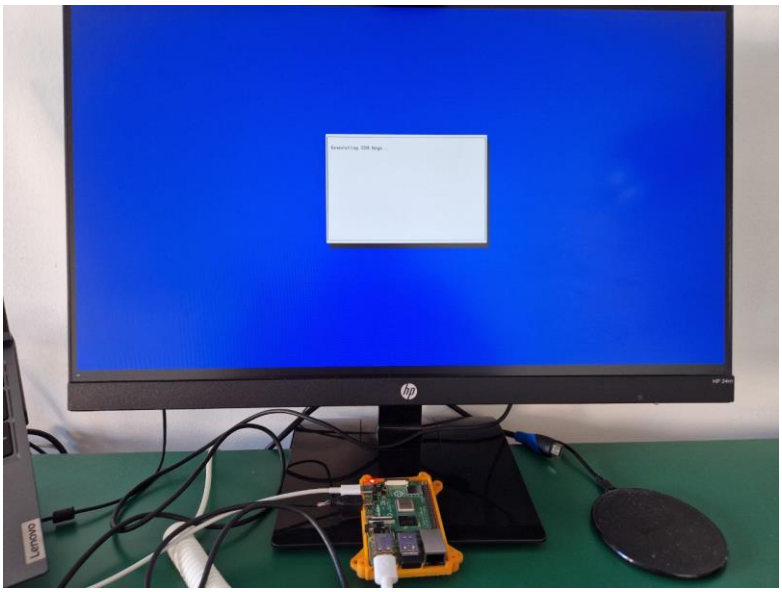


Figura 25: Instalación de Raspibian en la Raspberry Pi 4

Una vez completada la instalación, se configura un usuario (“alumno”) y una contraseña (“alumno”), y se instala Python 3.10.7 [16] siguiendo los siguientes comandos:

```
sudo apt install build-essential zlib1g-dev libncurses5-dev
libgdbm-dev libnss3-dev

tar -xzvf Python-3.10.7.tgz
cd Python-3.10.7/
./configure --enable-optimizations

sudo make altinstall
sudo rm /usr/bin/python
sudo ln -s /usr/local/bin/python3.10 /usr/bin/python
```

Figura 26: Instalación de Python en la Raspberry Pi 4

Con el código descrito se ha instalado Python y las herramientas necesarias para su funcionamiento. Tras esto, se procede a la instalación de las diferentes librerías de Python que se van a utilizar más adelante mediante el comando `pip`:

```
pip install Pillow
pip install matplotlib
pip install pyueeye
pip install numpy
pip install opencv-python
```

Por último, se instala un software para el control de la Raspberry Pi en remoto, con el fin de preparar la posibilidad de realizar cambios en la misma sin tener que disponer de un monitor y ratón/teclado. El software utilizado es VNC server [17],el cual se configura con un usuario y contraseña personal para operar en red local.

3.5 Script en Python para una adquisición simple de imagen

Tras haber realizado la instalación del entorno Linux en una Raspberry Pi, se realiza la primera parte del programa a desarrollar en Python, encargado de conectarse a la cámara y ordenar la adquisición de fotografías.

Se comienza el programa con la importación de las librerías necesarias, que para esta parte son:

- **Pyeye:** Librería del fabricante de la cámara para realizar la conexión con la cámara, con diferentes comandos relacionados con la adquisición de fotografías.
- **Numpy:** Librería usada para diferentes operaciones matemáticas.
- **Cv2:** Librería usada para realizar operaciones con imágenes.
- **Sys:** Librería usada para realizar operaciones con diferentes variables.
- **PIL**: Librería usada para realizar operaciones con imágenes.

Tras la importación de las librerías se procede a la definición de las variables, donde se comienza con la definición del puerto de conexión con la cámara, así como la información de la misma y de su sensor. Así mismo, se procede a la reserva de una memoria temporal para el guardado temporal de las imágenes, similar a una memoria caché.

En esta parte del código se da al usuario la elección de realizar la fotografía en color o en blanco y negro. Este cambio ha de realizarse modificando los bits por píxel y el número de canales. A continuación, se detalla la parte del código descrito:

```
"""
Programa para controlar la camara iDS en el robot MuWOB
Salvador Domínguez Durante
TFM Máster en Ing. Industrial
2023
"""

#Importacion de librerias necesarias
from pyueye import ueye
import numpy as np
import cv2
import sys
from PIL import Image




#Definicion de variables
hCam = ueye.HIDS(0)             #Conexion con la camara
sInfo = ueye.SENSORINFO()
cInfo = ueye.CAMINFO()
```

```
pcImageMemory = ueye.c_mem_p()
MemID = ueye.int()
rectAOI = ueye.IS_RECT()
pitch = ueye.INT()
nBitsPerPixel = ueye.INT(8)    # Bits por pixel (24 para color, 8
para B/W)
channels = 1                    # Numero de canales (3 para color,
1 para B/W)
m_nColorMode = ueye.INT()              # Y8/RGB16/RGB24/REG32
bytes_per_pixel = int(nBitsPerPixel / 8)
```

Tras haber realizado la importación de librerías y la definición de variables, se imprime por pantalla el inicio del programa. Se realiza la conexión con la cámara, obteniendo los valores de la misma. También se imprime por pantalla el modo seleccionado (color o blanco y negro) y los diferentes parámetros del modo. En caso de haber un error en algún paso descrito, se imprimirá un mensaje de error describiendo el mismo con el fin de que el usuario pueda identificarlo y solucionarlo. A continuación, se detalla la parte del código descrito:

```
print("Comienzo del programa")
print()


# Conexion con la camara
nRet = ueye.is_InitCamera(hCam, None)
if nRet != ueye.IS_SUCCESS:
    print("Error de conexion")

# Lectura de la informacion de la camara
nRet = ueye.is_GetCameraInfo(hCam, cInfo)
if nRet != ueye.IS_SUCCESS:
    print("Error de lectura de informacion camara")

# Lectura de la informacion del sensor
nRet = ueye.is_GetSensorInfo(hCam, sInfo)
if nRet != ueye.IS_SUCCESS:
    print("Error de lectura de informacion sensor")

nRet = ueye.is_ResetToDefault( hCam)
if nRet != ueye.IS_SUCCESS:
    print("Error de reseteo de informacion camara")

# Puesta en marcha del modo DIB
nRet = ueye.is_SetDisplayMode(hCam, ueye.IS_SET_DM_DIB)




# Puesta en marcha del modo color/BW e impresion por pantalla del
modo programado
if int.from_bytes(sInfo.nColorMode.value, byteorder='big') ==
ueye.IS_COLORMODE_BAYER:
```

```
    ueye.is_GetColorDepth(hCam, nBitsPerPixel, m_nColorMode)
    bytes_per_pixel = int(nBitsPerPixel / 8)
    print("IS_COLORMODE_BAYER: ", )
    print("\tm_nColorMode: \t\t", m_nColorMode)
    print("\tnBitsPerPixel: \t\t", nBitsPerPixel)
    print("\tbytes_per_pixel: \t\t", bytes_per_pixel)
    print()

elif int.from_bytes(sInfo.nColorMode.value, byteorder='big') ==
ueye.IS_COLORMODE_CBYCRY:
    m_nColorMode = ueye.IS_CM_BGRA8_PACKED
    nBitsPerPixel = ueye.INT(32)
    bytes_per_pixel = int(nBitsPerPixel / 8)
    print("IS_COLORMODE_CBYCRY: ", )
    print("\tm_nColorMode: \t\t", m_nColorMode)
    print("\tnBitsPerPixel: \t\t", nBitsPerPixel)
    print("\tbytes_per_pixel: \t\t", bytes_per_pixel)
    print()

elif int.from_bytes(sInfo.nColorMode.value, byteorder='big') ==
ueye.IS_COLORMODE_MONOCHROME:
    m_nColorMode = ueye.IS_CM_MONO8
    nBitsPerPixel = ueye.INT(8)
    bytes_per_pixel = int(nBitsPerPixel / 8)
    print("IS_COLORMODE_MONOCHROME: ", )
    print("\tm_nColorMode: \t\t", m_nColorMode)
    print("\tnBitsPerPixel: \t\t", nBitsPerPixel)
    print("\tbytes_per_pixel: \t\t", bytes_per_pixel)
    print()

else:
    m_nColorMode = ueye.IS_CM_MONO8
    nBitsPerPixel = ueye.INT(8)
    bytes_per_pixel = int(nBitsPerPixel / 8)
    print("else")
```

En el siguiente fragmento del código se da la posibilidad de recortar de forma digital una zona de interes propuesta por el usuario, previa la edición de los parámetros de altura y anchura en el código:

```
# Posible recorte de area de interes
nRet = ueye.is_AOI(hCam, ueye.IS_AOI_IMAGE_GET_AOI, rectAOI,
ueye.sizeof(rectAOI))
if nRet != ueye.IS_SUCCESS:
    print("Error area de interes")

width = rectAOI.s32Width
height = rectAOI.s32Height
```

En la parte final del código se imprime por pantalla las dimensiones de la imagen así como el modelo de cámara. También se reserva una memoria temporal para el guardado de la imagen con los parámetros previamente descritos. Por último se ordena la toma de una fotografía, el guardado de la misma en formato JPG en la carpeta donde se encuentra el script y la limpieza de la memoria temporal. Se finaliza el programa con la impresión por pantalla de un mensaje de confirmación de que la fotografía se ha guardado correctamente. A continuación, se detalla la parte del código descrito:

```
# Informacion de la camara y dimensiones de la imagen
print("Modelo de camara:\t\t", sInfo.strSensorName.decode('utf-8'))
print("Ancho maximo imagen:\t", width)
print("Altura maxima imagen:\t", height)
print()



# Reserva de memoria para la imagen con las dimensiones definidas
previamente y el modo de color
nRet = ueye.is_AllocImageMem(hCam, width, height, nBitsPerPixel,
pcImageMemory, MemID)
if nRet != ueye.IS_SUCCESS:
    print("AllocImageMem Error")
else:
        nRet = ueye.is_SetImageMem(hCam, pcImageMemory, MemID)
    if nRet != ueye.IS_SUCCESS:
        print("is_SetImageMem Error")
    else:
                nRet = ueye.is_SetColorMode(hCam, m_nColorMode)


#Toma de una fotografia

nret = ueye.is_FreezeVideo(hCam, ueye.IS_WAIT)
FileParams = ueye.IMAGE_FILE_PARAMS()
FileParams.pwchFileName = "imagen_muestra.JPG"
FileParams.nFileType = ueye.IS_IMG_JPG
FileParams.ppcImageMem = None
FileParams.pnImageID = None
nret = ueye.is_ImageFile(hCam, ueye.IS_IMAGE_FILE_CMD_SAVE,
FileParams, ueye.sizeof(FileParams))
print("Imagen guardada")
ueye.is_FreeImageMem(hCam, pcImageMemory, MemID)
ueye.is_ExitCamera(hCam)
```

Como resultado del programa descrito, se obtiene la imagen mostrada en la figura 27, donde se pueden observar los diferentes pocillos. La iluminación presente en la siguiente fotografía no es la ideal, quedando pendiente, como ya se indicó anteriormente, la instalación de iluminación led en el interior de la estufa en su parte superior.

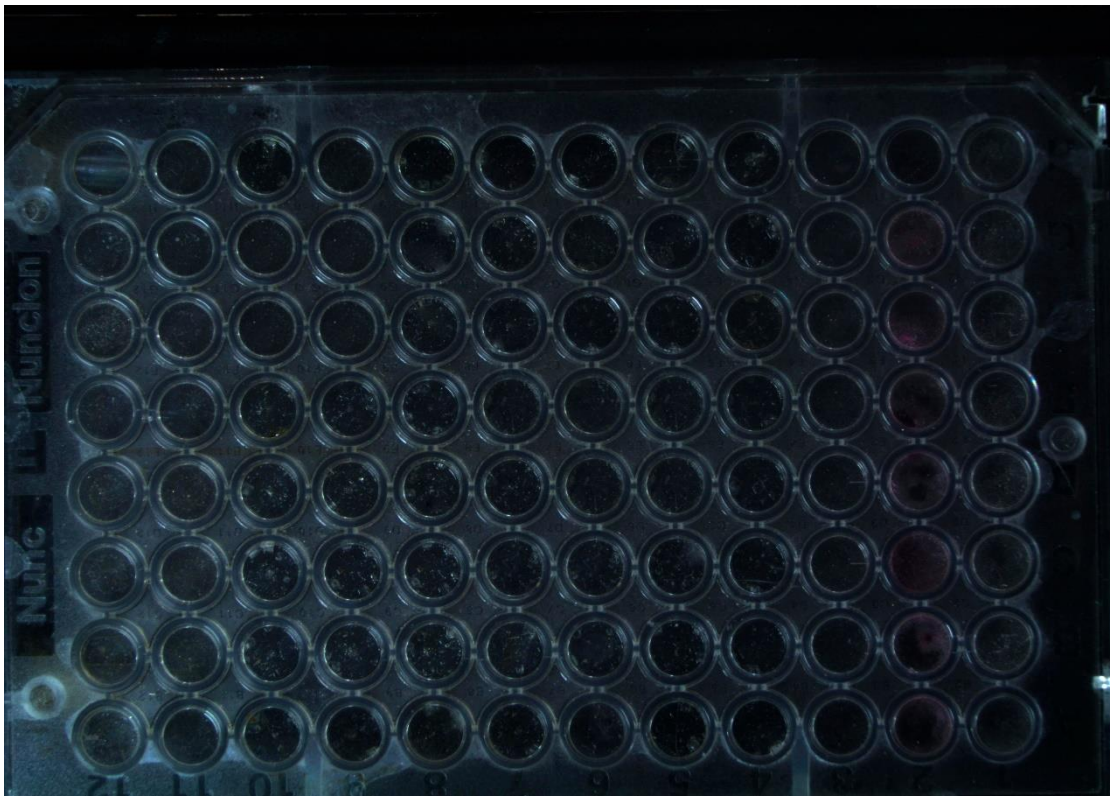


Figura 27: Fotografía obtenida con el código descrito

3.6 Script en Python para la adquisición múltiple de imágenes

Tras haber realizado un script para la adquisición de una imagen, se va a desarrollar otro para la toma múltiple de imágenes. En él se puede definir mediante la modificación de dos variables, el número de fotografías a realizar y el intervalo de captura de estas. Se comienza el programa con la descripción comentada del mismo y la importación de las librerías necesarias. Para esta parte se van a usar las siguientes librerías:

- **Os**: Librería usada para realizar operaciones dependientes del sistema operativo, tal como manipular archivos y directorios.
- **Time**: Librerías usadas para diferentes funciones relacionadas con el tiempo.
- **Pyeye:** Librería del fabricante de la cámara para realizar la conexión con la cámara, con diferentes comandos relacionados con la adquisición de fotografías.

Tras la importación de las librerías, se procede a la definición de una función que se encarga de tomar las imágenes mediante la cámara. El proceso de captura de imágenes es similar a la toma simple previamente mencionado. Las imágenes se guardan usando la fecha y hora como título de estas.

Al final del código se definen las variables modificables donde se encuentran el número de imágenes a tomar y el intervalo entre las mismas. El código desarrollado puede encontrarse a continuación:

```
"""
Programa para la toma multiple de imágenes con la camara iDS en el
robot MuWOB

Salvador Domínguez Durante
TFM Master en Ing. Industrial
2023
"""

#Importacion de librerias necesarias para el funcionamiento del
programa

import os
import time
from pyueye import ueye
from pyueye import ueye
import numpy as np
import cv2
import sys
from PIL import Image
import datetime




def capture_photos(num_photos, capture_interval):

    #Definicion de variables
    hCam = ueye.HIDS(0)             #Conexion con la camara
    sInfo = ueye.SENSORINFO()
    cInfo = ueye.CAMINFO()
    pcImageMemory = ueye.c_mem_p()
    MemID = ueye.int()
    rectAOI = ueye.IS_RECT()
    pitch = ueye.INT()
    nBitsPerPixel = ueye.INT(8)    # Bits por pixel (24 para color,
8 para B/W)
    channels = 1                       # Numero de canales (3 para
color, 1 para B/W)
    m_nColorMode = ueye.INT()        # Y8/RGB16/RGB24/REG32
    bytes_per_pixel = int(nBitsPerPixel / 8)
```

```
    # Conexion con la camara
    nRet = ueye.is_InitCamera(hCam, None)
    if nRet != ueye.IS_SUCCESS:
        print("Error de conexion")

    # Lectura de la informacion de la camara
    nRet = ueye.is_GetCameraInfo(hCam, cInfo)
    if nRet != ueye.IS_SUCCESS:
        print("Error de lectura de informacion camara")

    # Lectura de la informacion del sensor
    nRet = ueye.is_GetSensorInfo(hCam, sInfo)
    if nRet != ueye.IS_SUCCESS:
        print("Error de lectura de informacion sensor")

    nRet = ueye.is_ResetToDefault( hCam)
    if nRet != ueye.IS_SUCCESS:
        print("Error de reseteo de informacion camara")

    # Puesta en marcha del modo DIB
    nRet = ueye.is_SetDisplayMode(hCam, ueye.IS_SET_DM_DIB)

    # Puesta en marcha de el modo color/BW e impresion por pantalla
del modo programado
    if  int.from_bytes(sInfo.nColorMode.value,  byteorder='big')  ==
ueye.IS_COLORMODE_BAYER:
        ueye.is_GetColorDepth(hCam, nBitsPerPixel, m_nColorMode)
        bytes_per_pixel = int(nBitsPerPixel / 8)
        print("IS_COLORMODE_BAYER: ", )
        print("\tm_nColorMode: \t\t", m_nColorMode)
        print("\tnBitsPerPixel: \t\t", nBitsPerPixel)
        print("\tbytes_per_pixel: \t\t", bytes_per_pixel)
        print()

    elif int.from_bytes(sInfo.nColorMode.value, byteorder='big') ==
ueye.IS_COLORMODE_CBYCRY:
        m_nColorMode = ueye.IS_CM_BGRA8_PACKED
        nBitsPerPixel = ueye.INT(32)
        bytes_per_pixel = int(nBitsPerPixel / 8)
        print("IS_COLORMODE_CBYCRY: ", )
        print("\tm_nColorMode: \t\t", m_nColorMode)
        print("\tnBitsPerPixel: \t\t", nBitsPerPixel)
        print("\tbytes_per_pixel: \t\t", bytes_per_pixel)
        print()

    elif int.from_bytes(sInfo.nColorMode.value, byteorder='big') ==
ueye.IS_COLORMODE_MONOCHROME:
        m_nColorMode = ueye.IS_CM_MONO8
        nBitsPerPixel = ueye.INT(8)
        bytes_per_pixel = int(nBitsPerPixel / 8)
        print("IS_COLORMODE_MONOCHROME: ", )
        print("\tm_nColorMode: \t\t", m_nColorMode)
        print("\tnBitsPerPixel: \t\t", nBitsPerPixel)
        print("\tbytes_per_pixel: \t\t", bytes_per_pixel)
        print()
```

```
    else:
        m_nColorMode = ueye.IS_CM_MONO8
        nBitsPerPixel = ueye.INT(8)
        bytes_per_pixel = int(nBitsPerPixel / 8)
        print("else")

    # Posible recorte de area de interes
    nRet = ueye.is_AOI(hCam, ueye.IS_AOI_IMAGE_GET_AOI, rectAOI,
ueye.sizeof(rectAOI))
    if nRet != ueye.IS_SUCCESS:
        print("Error area de interes")

    width = rectAOI.s32Width
    height = rectAOI.s32Height

    # Informacion de la camara y dimensiones de la imagen
    print("Modelo de camara:\t\t", sInfo.strSensorName.decode('utf-
8'))
    print("Ancho maximo imagen:\t", width)
    print("Altura maxima imagen:\t", height)
    print()


    # Asignacion de memoria
    nRet = ueye.is_AllocImageMem(hCam, width, height, nBitsPerPixel,
pcImageMemory, MemID)
    if nRet != ueye.IS_SUCCESS:
        print("is_AllocImageMem ERROR")
    else:
    # Convertir memoria en memoria activa
        nRet = ueye.is_SetImageMem(hCam, pcImageMemory, MemID)
        if nRet != ueye.IS_SUCCESS:
            print("is_SetImageMem ERROR")
        else:
        # Modo de funcionamiento (BN o Color)
            nRet = ueye.is_SetColorMode(hCam, m_nColorMode)

        # Toma de una fotografia
            for i in range(num_photos):
                timestamp                                          =
datetime.datetime.now().strftime("%Y%m%d_%H%M%S")

                nret = ueye.is_FreezeVideo(hCam, ueye.IS_WAIT)

            # Nombre del archivo segun fecha y hora
                filename = f"{timestamp}.JPG"

                FileParams = ueye.IMAGE_FILE_PARAMS()
                FileParams.pwchFileName = filename
                FileParams.nFileType = ueye.IS_IMG_JPG
                FileParams.ppcImageMem = None
                FileParams.pnImageID = None
                nret           =           ueye.is_ImageFile(hCam,
ueye.IS_IMAGE_FILE_CMD_SAVE, FileParams, ueye.sizeof(FileParams))
                print("Imagen guardada:", filename)
```

```
                time.sleep(capture_interval)

# Liberado de memoria
    ueye.is_FreeImageMem(hCam, pcImageMemory, MemID)
    ueye.is_ExitCamera(hCam)



# Variables a definir por el usuario

#Numero de fotos a realizar
num_photos = 5

#Segundos entre cada foto
capture_interval = 5


capture_photos(num_photos, capture_interval)
```

Como resultado del programa descrito, se obtienen el número de imágenes solicitadas por el usuario y con el intervalo de toma de las mismas. Se imprime por pantalla la confirmación de guardado y el nombre de cada una de las imágenes tomadas.

3.7 Script en Python para el tratamiento de las imágenes adquiridas

En el siguiente programa se van a procesar las imágenes adquiridas para realizar la medición del área del tejido en diferentes fotografías de la misma muestra en momentos diferentes.

Se comienza el programa con la descripción comentada del mismo y la importación de las librerías necesarias. Para esta parte se van a usar principalmente las siguientes librerías:

- **Pylab**: Librería usada para realizar las gráficas comparativas del área de la muestra a través del tiempo.
- **Numpy y math**: Librerías usadas para diferentes operaciones matemáticas.
- **Skimage:** Librería usada para realizar operaciones con imágenes.

Tras la importación de las librerías, se procede a la importación de una imagen que contiene la máscara de los pocillos. Esta imagen ha sido creada usando un software de edición de imágenes con el objetivo de identificar las áreas de los

pocillos donde se van a encontrar las muestras. Debido a que la iluminación y la posición de los pocillos es siempre la misma, la máscara es válida para todos los experimentos futuros. Tras importar la máscara, la imagen se limpia de los posibles pixeles sueltos (figura 28).

```
"""
Programa para tratar las imagenes en el robot MuWOB
Salvador Domínguez Durante
TFM Máster en Ing. Industrial
2023
"""

#Importacion de librerias necesarias
import os
import glob
import pylab as plt
import numpy as np
from skimage import io
from skimage import color
from skimage.morphology import closing, square
from skimage import measure
from skimage.measure import label, regionprops
from skimage.filters import threshold_otsu

#Importacion de la mascara

masc = io.imread('mascara_pocillos.jpg')

masc = color.rgb2gray(masc) #Escala de grises

plt.imshow(masc,cmap=plt.cm.gray)

plt.axis('off')

plt.show()

# limpiar la mascara de pocillos de píxeles sueltos

bw = closing(masc > 0.99, square(10))

plt.imshow(bw,cmap=plt.cm.gray)

plt.axis('off')

plt.show()

mask=bw
```

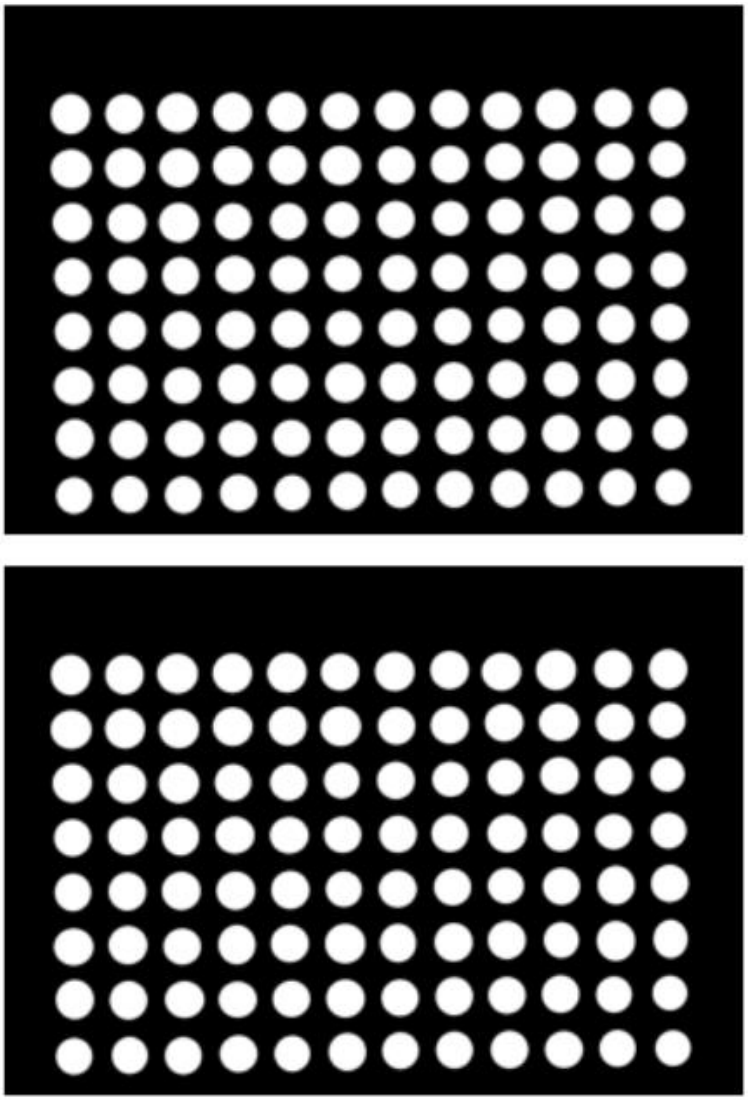


Figura 28: Máscara de los pocillos pre y post limpieza

Tras ello se procede al etiquetado de los pocillos usando la máscara post limpieza (figura 30). El código devuelve una matriz con los pocillos etiquetados y el número de ellos, en nuestro caso 96 pocillos:

```
labels = measure.label(mask, connectivity=2, background=0)

fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 4))

ax.imshow(labels)
plt.show()

labels.max()
```

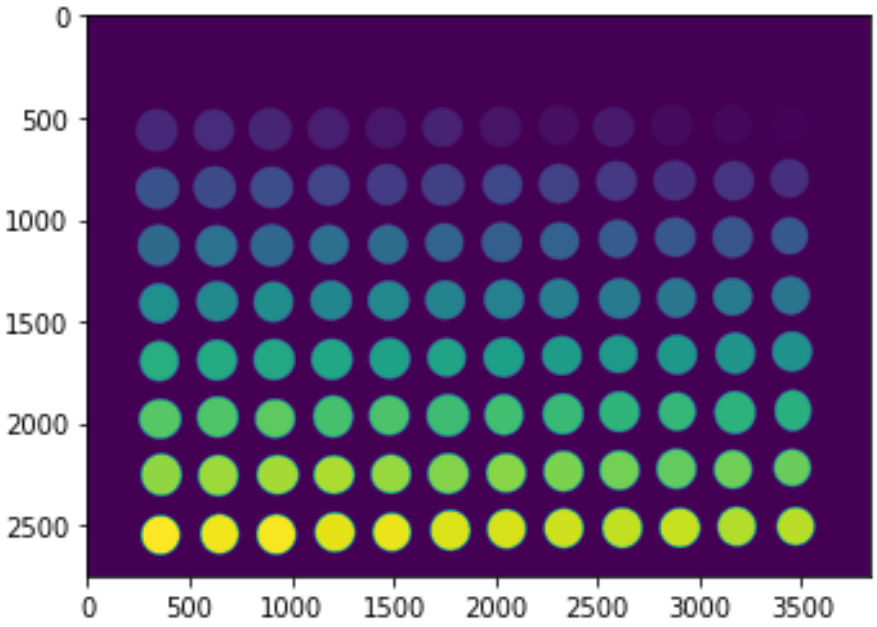


Figura 29: Imagen de los pocillos etiquetados

A continuación, se importa una imagen de los pocillos vacíos, para detectar los reflejos presentes. Estos reflejos dan lugar a zonas blancas en la imagen que no se corresponden a tejidos analizados. Como la iluminación es la misma para todas las imágenes, podemos calcular la posición de estos reflejos sobre una bandeja vacía, para luego eliminarlos en las bandejas con muestras. Se pueden observar los reflejos obtenidos en la figura 30. Para detectar los reflejos en cada pocillo, se calcula un umbral para cada uno aplicando el método de Otsu [18].Estos umbrales se guardan para ser aplicados posteriormente en el análisis de la evolución de los tejidos en cada pocillo.

```
# Cargar imagen de pocillos vacíos para ver los reflejos - cálculo
de los umbrales de cada pocillo

im = io.imread('pocillos_0.jpg')
im2 = color.rgb2gray(im) #Escala de grises - imagen original
im_filtrada=mask*im2 # Imagen original solo en los pocillos

fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 4))
ax.imshow(im_filtrada,cmap=plt.cm.gray)
plt.show()

# umbralizamos cada pocillo por separado - son 96 pocillos -
calculamos el umbral de cada uno
umbral=[]
limpiar_pocillos = np.zeros(labels.shape, dtype='uint8')
for i, label in enumerate(np.unique(labels)):
    if label == 0:
        umbral.append(0)  # umbral del fondo a 0
```

```
        continue
    # Construimos una máscara para mostrar sólo las componentes
conectadas de la etiqueta actual (cada pocillo).

    label_mask = np.zeros(labels.shape, dtype='uint8')
    label_mask[labels == label] = 255
    label_mask = color.rgb2gray(label_mask)
    label2 =im_filtrada*label_mask

    umb = threshold_otsu(label2[labels == label]) #cálculo del
umbral de cada pocillo
    umbral.append(umb)
    im_umb = label2 > umb
    limpiar_pocillos[labels==label]=im_umb[labels==label]


fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 4))
ax.imshow(limpiar_pocillos,cmap=plt.cm.gray)
plt.show()
```

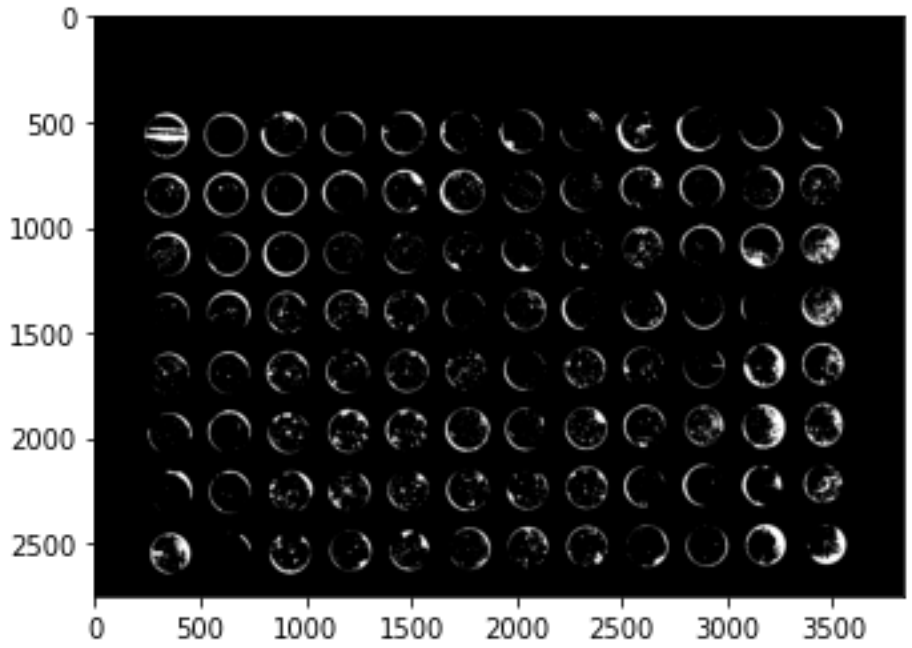


Figura 30: Imagen de reflejos obtenidos en los pocillos

Tras esto, se procede a importar las imágenes de muestras con su diferente evolución a través del tiempo. En la figura 31 se puede observar una imagen sin procesar con diferentes muestras en ciertos pocillos. Para la simulación de la evolución de los tejidos se han utilizado granos de arroz a los que se les ha modificado su tamaño a lo largo de las imágenes.

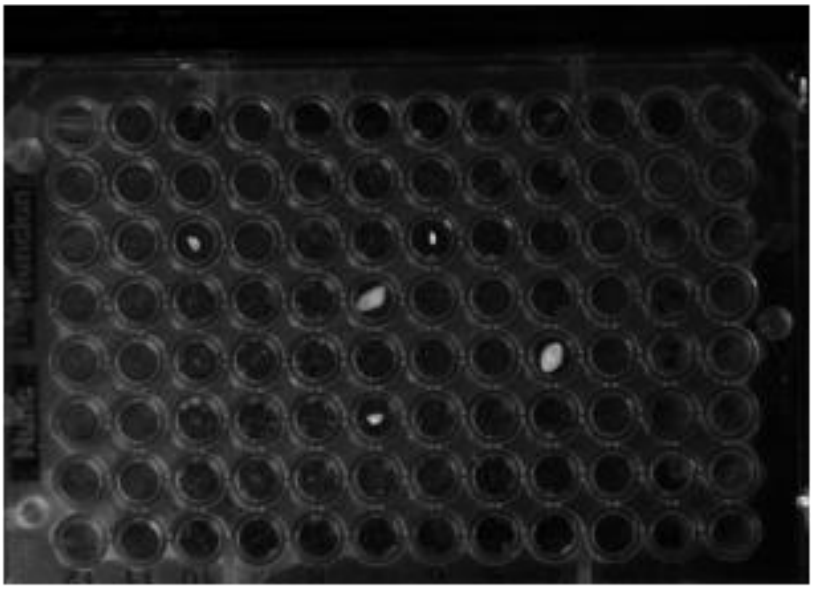


Figura 31: Bandeja de pocillos con muestras

Se define una función que permita analizar y procesar las diferentes imágenes adquiridas en un mismo experimento. Para ello se definen diferentes variables como el número de imágenes, la ruta de estas y el número de pocillos. En la función desarrollada se importan las imágenes, se convierten a escala de grises y se construye una máscara para mostrar sólo las componentes conectadas de cada pocillo, denominado "label" en el código. Tras esto el área de cada pocillo es almacenada en un array que es devuelto de la función. Las imágenes procesadas muestran únicamente las muestras presentes como se puede observar en la figura 32. El código descrito se desarrolla a continuación.

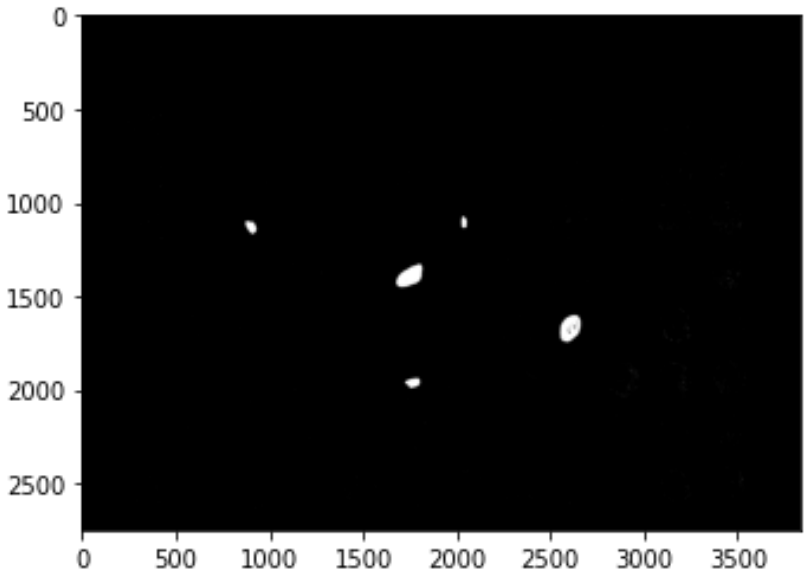


Figura 32: Muestras tras el procesado de la imagen

```
def process_images(folder_path, num_images,
num_pocillos,umbral,labels):
    # Obtener la lista de archivos de imagen en la carpeta
    image_paths     =     sorted(glob.glob(os.path.join(folder_path,
"*.jpg")), key=os.path.getmtime, reverse=True)[:num_images]

    # Matriz para almacenar las áreas de los pocillos de cada imagen
    array = []

    for i, path in enumerate(image_paths):
        # Carga la imagen
        image = io.imread(path)
        fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 4))
        ax.imshow(image,cmap=plt.cm.gray)
        plt.show()

        # Convertir la imagen a escala de grises
        im2 = color.rgb2gray(image)   # Escala de grises - imagen
original
        im_filtrada = mask * im2   # Imagen original solo en los
pocillos


         # Construimos una máscara para mostrar sólo las componentes
conectadas de la etiqueta actual (cada pocillo).
        im_final = np.zeros(labels.shape, dtype='uint8')
        area=[]
        for n,label in enumerate(np.unique(labels)):
            if label == 0:

                continue

            label_mask = np.zeros(labels.shape, dtype='uint8')
            label_mask[labels == label] = 255
            label_mask = color.rgb2gray(label_mask)
            label2 =im_filtrada*label_mask
            label3=limpiar_pocillos*label_mask

            im_umb = label2 > umbral[n]
            im_umb [label3==255]=0
            im_final[labels==label]=im_umb[labels==label]
            total=np.count_nonzero(im_umb)
            area.append(total)


        array = np.append(array, np.array(area), axis=0)


        fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 4))
        ax.imshow(im_final,cmap=plt.cm.gray)
        plt.show()



    return array
```

Para finalizar el programa, se genera una gráfica con la evolución del área en cada pocillo (figura 33). Para esto se convierte el array de las áreas previamente en una matriz de cuyas dimensiones son el número de imágenes como filas y el número de pocillos como columnas. Se ha implementado la representación de forma que solo aquellas áreas significativas sean dibujadas eliminando así aquellas que sean igual al cero por ser pocillos vacíos. Para esto se ha definido un umbral que puede ser modificado por el usuario al inicio del programa.

```
#Definición del umbral de representación
umb_rep=1000
num_images = len(array) // 96  # Calculo del número de imagenes
num_areas = 96  # Número de pocillos

matrix = np.reshape(array, (num_images, num_areas))

# Seleccion de areas por encima del umbral de representacion
areas_above_1000 = np.where(np.any(matrix > umb_rep, axis=0))[0]

# Representación de areas por encima del umbral
for area in areas_above_1000:
    plt.plot(matrix[:, area], label=f"Pocillo {area+1}")

plt.xlabel('Imagen')
plt.ylabel('Area')
plt.title('Evolución de la muestra')
plt.legend()
plt.show()
```

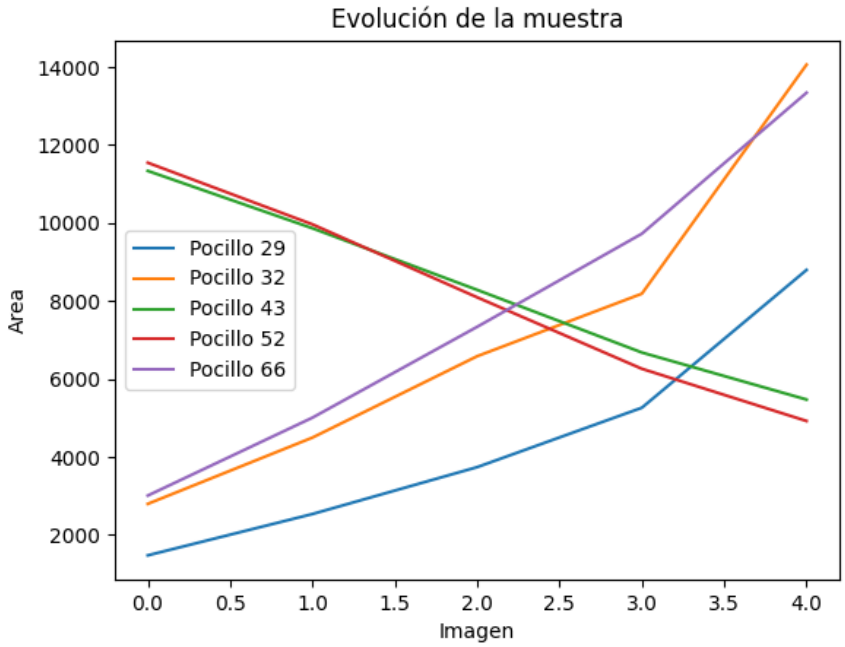


Figura 33: Evolución de las áreas representativas

4. PRESUPUESTO

Se ha elaborado un presupuesto simple con los diferentes costes de las horas de trabajo empleadas y los materiales empleados para la realización de este TFM. El coste de los materiales empleados ha sido asumido por el Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna. El presupuesto se ha dividido en dos apartados, obteniendo un coste total de **9735,82 €.** Este presupuesto realizado se puede observar en la siguiente tabla:

Concepto	Coste unitario	Cantidad	Total
Mano de obra	30€/hora	300	9000€
		TOTAL (IGIC no incluido)	9000€
		TOTAL (IGIC incluido)	**9630 €**

Concepto	Coste unitario	Cantidad	Total
Raspberry Pi 4	71,40€	1	71,40€
Tarjeta MicroSD 16 GB	7€	1	7€
Cableado Raspberry Pi 4	15€	1	15€
Material de Impresión 3D (PLA)	25€ /kg	0,02 kg	0,5€
Foco LED	5€	1	5€
		TOTAL (IGIC no incluido)	98,9€
		TOTAL (IGIC incluido)	**105,82 €**

Tabla 2: Presupuesto Trabajo Fin de Máster

5. CONCLUSIONES Y TRABAJO FUTURO

El objetivo de este Trabajo Fin de Máster (TFM) fue el estudio del estado actual del prototipo MuWOB y la propuesta de mejoras al mismo. Tras la finalización del TFM se han propuesto y desarrollado las siguientes mejoras:

- Mejoras de diseño físico, como los paneles exteriores de la estufa y un soporte para la bandeja de pocillos, con el fin de mejorar el funcionamiento del robot previamente desarrollado en la Universidad de La Laguna.

- Mediante el uso de un lenguaje de programación abierto, se ha conseguido desarrollar distintos programas capaces de realizar diferentes funciones en la adquisición y posterior procesado de imágenes de muestras. Usando estos programas, se ha demostrado su funcionalidad para la medición de áreas de los tejidos ensayados a través de la evolución que sufren al ser expuestos a diferentes fármacos.

- Se ha sustituido el uso de un ordenador fijo, responsable de la toma y tratamiento de estas imágenes, por una Raspberry Pi 4, la cual permite no depender de un ordenador externo al encontrarse integrada en el propio robot.

En este Trabajo Fin de Máster (TFM) se han aplicado los conocimientos adquiridos durante el propio máster, destacando el papel del segundo año de especialización en automática y robótica. Así mismo, también se ha puesto en práctica lo aprendido en el grado en Ingeniería Mecánica. Estos conocimientos van desde el modelado CAD hasta la programación en Python y el tratamiento de imágenes. Recalcar los conocimientos aprendidos en el área de óptica y robótica, destacando lo aprendido en el manejo de la cámara iDS.

Como trabajo futuro se proponen las siguientes líneas de desarrollo:

- Implementación de comunicación entre el sistema instalado en este Trabajo Fin de Máster, consistente en la Raspberry Pi 4, y la aplicación Android previamente existente, con el fin de implementar en esta última la función para llamar a la función de toma de fotografías y mediciones de áreas ya presentes en la Raspberry Pi 4.

- Instalación definitiva de iluminación de tipo LED en el interior de la estufa, con un estudio lumínico más completo, que permita una iluminación homogénea en la bandeja de pocillos.

- Fabricación en acero inoxidable de los paneles exteriores diseñados en este Trabajo Fin de Máster y su posterior instalación en la estufa.

6. CONCLUSIONS AND FUTURE WORK

The objective of this Master's Thesis (TFM) was to study the current state of the MuWOB prototype and to propose improvements to it. After the completion of the TFM, the following improvements have been proposed and developed:

- Physical design improvements, such as the exterior panels of the oven and a support for the well tray, in order to improve the operation of the robot previously developed at the University of La Laguna.

- By using an open programming language, it has been possible to develop different programs capable of performing different functions in the acquisition and subsequent processing of sample images. Using these programmes, their functionality has been demonstrated for measuring the areas of the tissues tested through the evolution they undergo when exposed to different drugs.

- The use of a fixed computer, responsible for taking and processing these images, has been replaced by a Raspberry Pi 4, which does not depend on an external computer as it is integrated into the robot itself.

In this Master's Thesis (TFM), the knowledge acquired during the Master's degree itself has been applied, highlighting the role of the second year of specialisation in automation and robotics. Likewise, what has been learnt in the degree in Mechanical Engineering has also been put into practice. This knowledge ranges from CAD modelling to Python programming and image processing. Knowledge has also been obtained in the area of optics and robotics, highlighting what has been learned in the handling of the iDS camera.

The following lines of development are proposed as future work:

- Implementation of communication between the system installed in this Master's Thesis, consisting of the Raspberry Pi 4, and the previously existing Android application, in order to implement in the latter the function to call the function for taking photographs and measurements of areas already present in the Raspberry Pi 4.
- Final installation of LED lighting inside the cooker, with a more complete lighting study, allowing homogeneous lighting in the well tray.

- Manufacture in stainless steel of the exterior panels designed in this Master's thesis and their subsequent installation in the cooker.

REFERENCIAS

[1] E. M. Hernández, «MuROB, automatización del baño de órganos por métodos ópticos,» Trabajo Fin de Máster, Máster en Ingeniería Industrial, Universidad de la Laguna, 2022.

[2] R. B. Jurado y M. J. R. Valido, «Nuevos instrumentos para la investigación en el laboratorio de Farmacología y Fisiología.,» ULL, San Cristóbal de La Laguna, 2017.

[3] D. Díaz-Martín, J. G. Hernández-Jiménez, M. Rodríguez-Valido y R. Borges, «Measuring the Contractile Response of Isolated Tissue Using an Image Sensor,» *Sensors,* nº 15, pp. 9179-9188, 2015.

[4] G. E. Rose y R. N. Patel, «A simple system of multiple organ-baths,» *Experientia,* nº 34, p. pages 550–551, 1978.

[5] MAEHLE y ANDREAS-HOLGER, «"Receptive Substances": John Newport Langley (1852–1925) and his Path to a Receptor Theory of Drug Action,» *Cambridge University Press,* vol. 2, nº 48, p. 153–174, 2004.

[6] G. SEGARRA, P. MEDINA, A. ACUÑA, C. DOMENECH y J. B. MARTINEZ, «Comparative effects of dilator drugs on human penile dorsal artery and deep dorsal vein,» *Clin Sci (Lond,* vol. 96, nº 1, pp. 59-65, 1999.

[7] R. Borges, D. Díaz-Martín, J. G. Hernández-Jimenez, M. Rodríguez-Valido y B. Beltrán, «Analyzing isolated blood vessel contraction in multi-well plates,» *Naunyn Schmiedebergs Arch Pharmacol,* vol. 389, 2016.

[8] Noldus, «https://www.noldus.com/daniovision/observation-chamber,» Noldus, 2023. [En línea]. Available: https://www.noldus.com/daniovision/observation-chamber. [Último acceso: 2023].

[9] J. N. Shields, E. C. Hales, L. E. Ranspach, X. Luo, S. Orr y D. Runft, «Supplementary Materials: Exposure of Larval Zebrafish to the Insecticide Propoxur Induced Developmental Delays that Correlate with Behavioral Abnormalities and Altered Expression of hspb9 and hspb11,» *Toxics 2019,* vol. 7, nº 50, 2019.

[10] B. Jespersen, N. R. Tykocki, S. W. Watts y P. J. Cobbett, «Measurement of Smooth Muscle Function in the Isolated Tissue Bath-applications to Pharmacology Research,» *J Vis Exp,* vol. 19, p. 95, 2015.

[11] Solidworks, «Solidworks home page,» 2023. [En línea]. Available: https://www.solidworks.com/es.

[12] Ultimaker, «UltiMaker Cura Homepage,» 2023. [En línea]. Available: https://ultimaker.com/software/ultimaker-cura/.

[13] iDS, «uEye Camera Software,» 2023. [En línea]. Available: https://es.ids-imaging.com/ids-software-suite.html.

[14] Raspberry Pi, «Raspberry Pi 4,» 2023. [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[15] R. Pi, «Raspberry Pi OS Home Page,» 2023. [En línea]. Available: https://www.raspberrypi.com/software/.

[16] Python, «Python 3.10.7,» 2022. [En línea]. Available: https://www.python.org/downloads/release/python-3107/.

[17] R. VNC, «Real VNC,» 2023. [En línea]. Available: https://www.realvnc.com/es/.

[18] Wikipedia, «Método del valor umbral,» 2023. [En línea]. Available: https://es.wikipedia.org/wiki/M%C3%A9todo_del_valor_umbral.

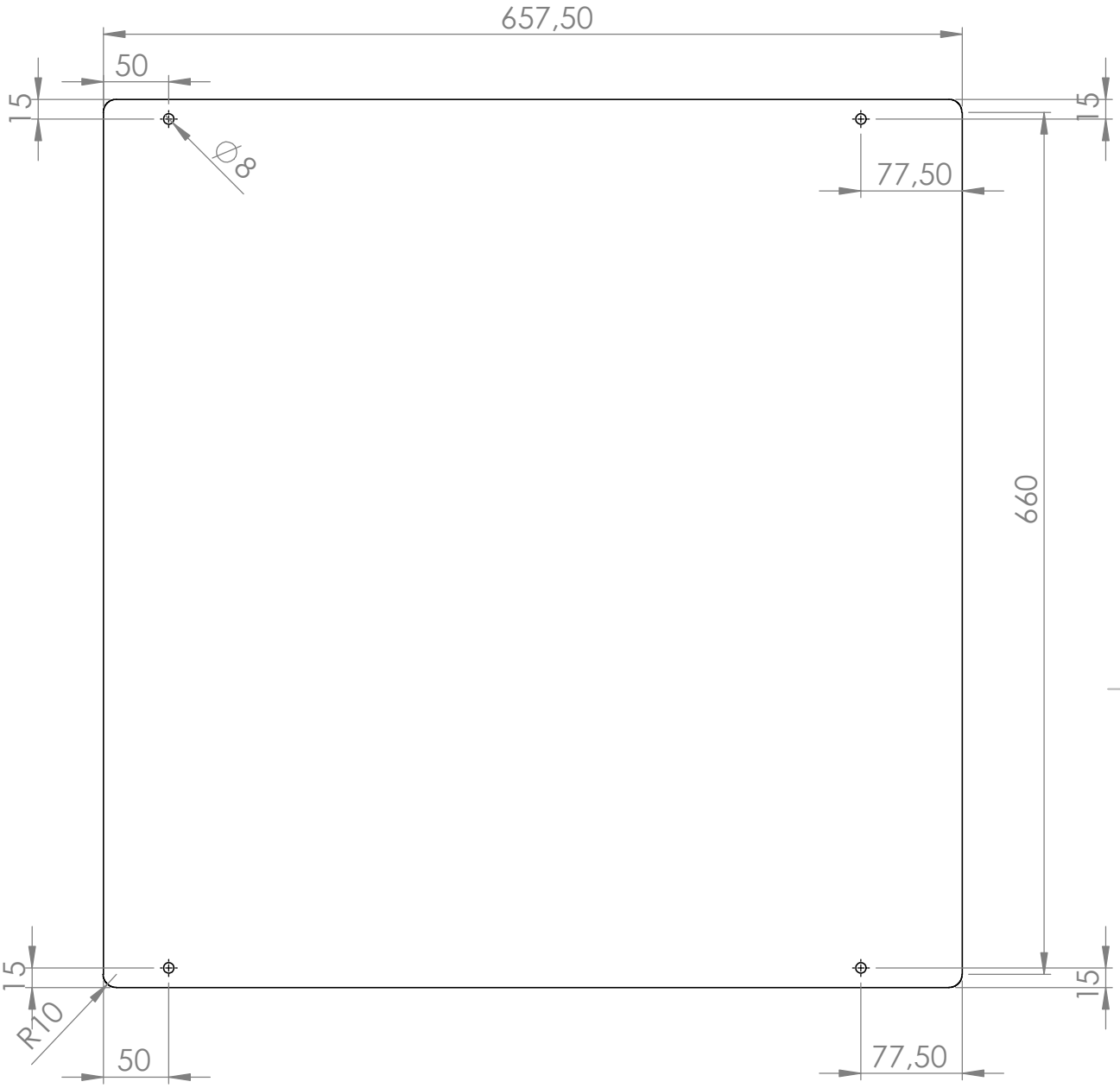
[19] iDS, «UI-1490LE Datasheet,» 2023. [En línea]. Available: https://es.ids-imaging.com/store/ui-1490le.html.

ANEXO

1. Planos

A continuación, se adjuntan los diferentes planos realizados en CAD, de las siguientes piezas:

- Cubierta Lateral
- Cubierta Superior
- Cubierta Trasera
- Cubierta Frontal
- Cubierta Inferior
- Soporte Bandeja Pocillos



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.			
VERIF.			
APROB.			
FABR.			
CALID.			

TÍTULO:

MATERIAL:

N.º DE DIBUJO

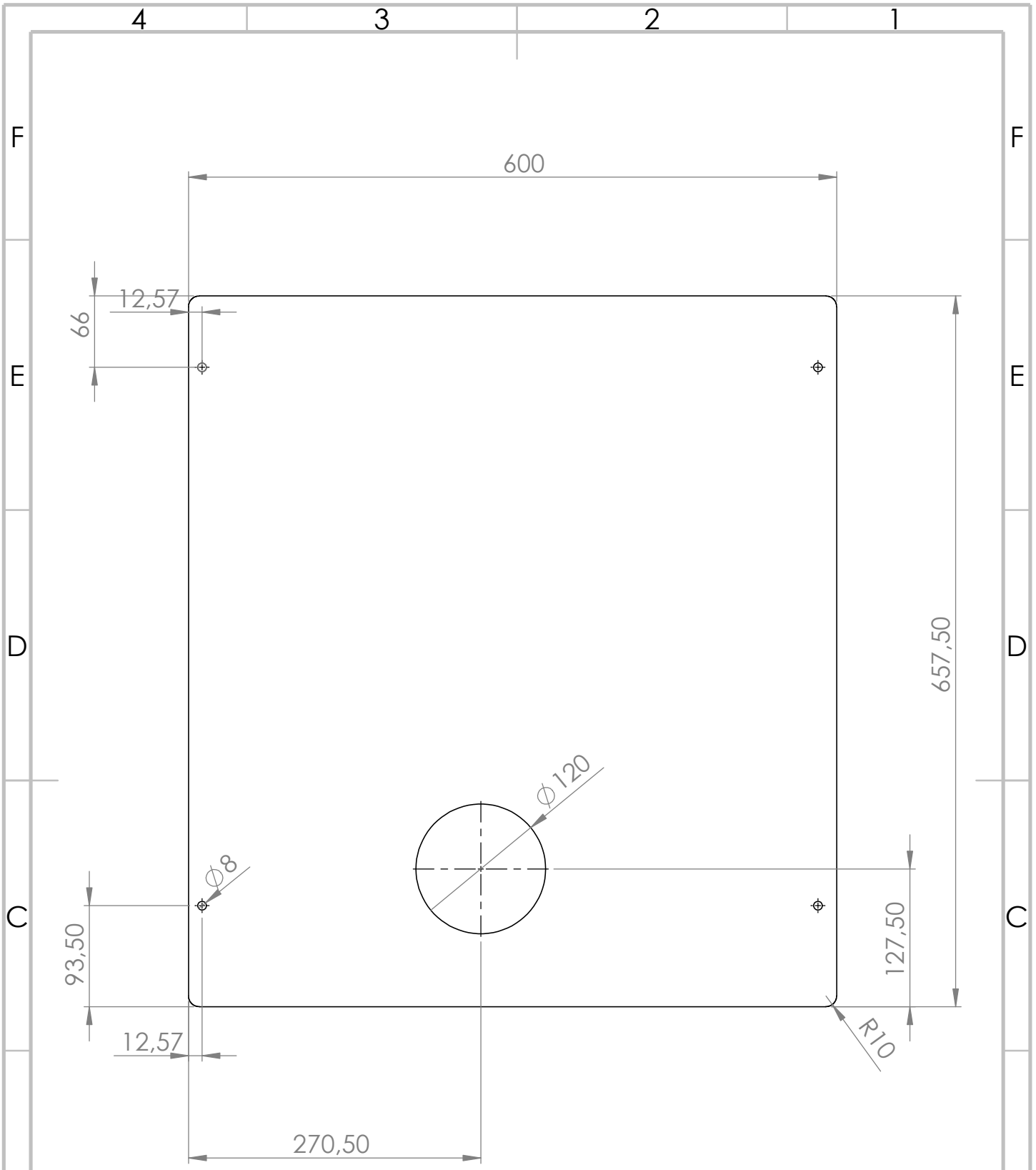
Laterales

A4

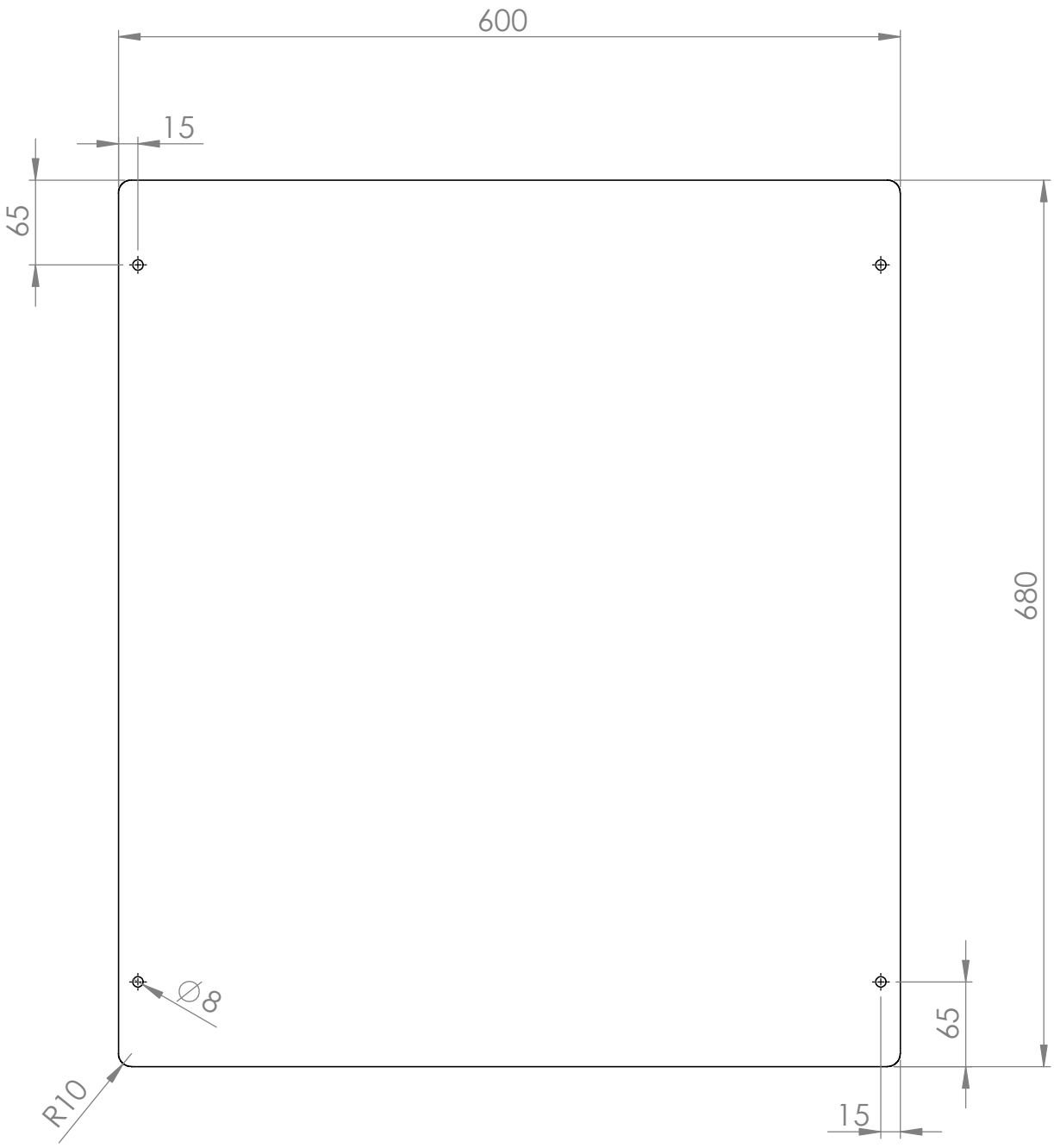
PESO:

ESCALA:1:5

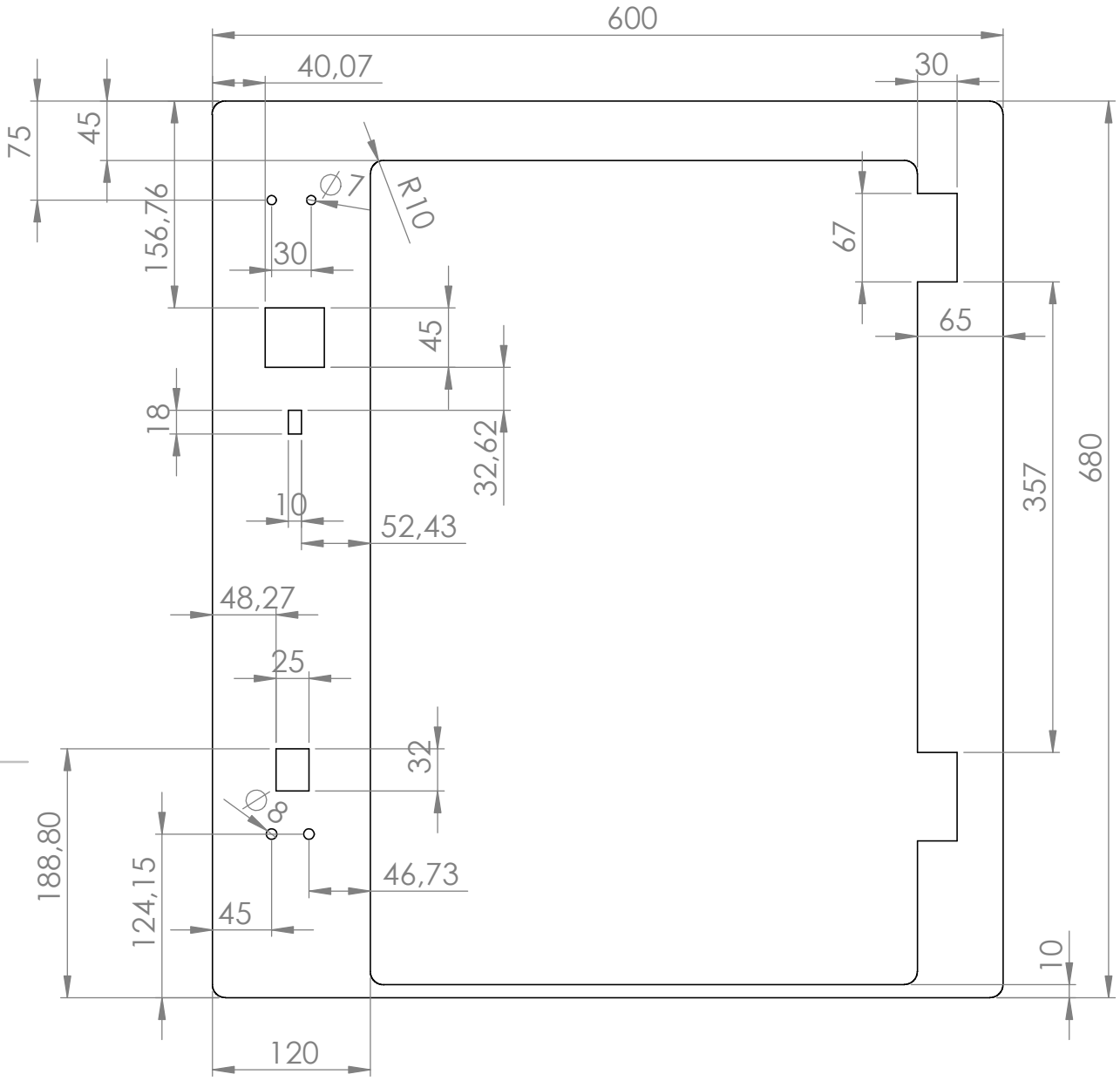
HOJA 1 DE 1



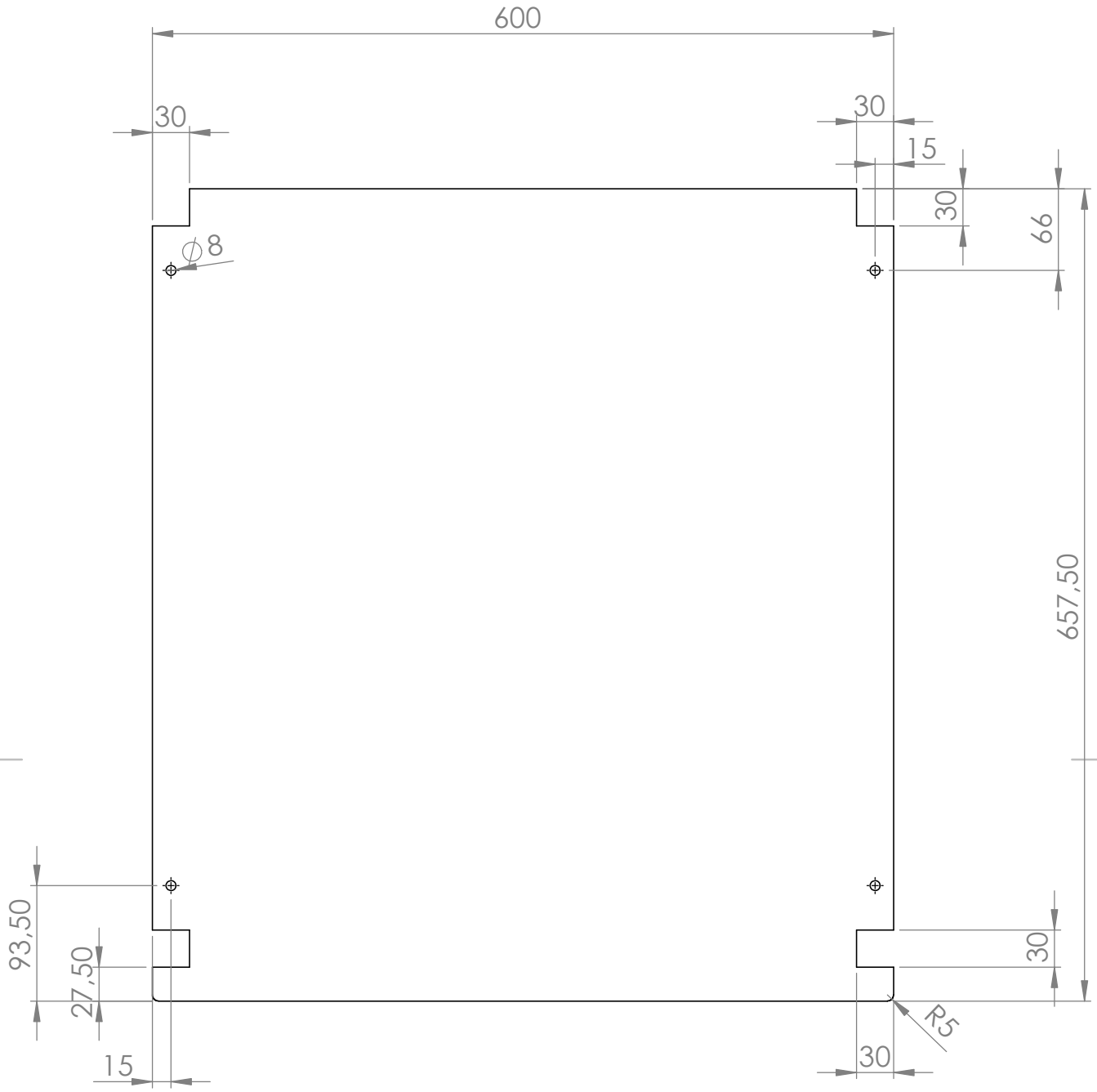
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN		
	NOMBRE	FIRMA	FECHA			TÍTULO:
DIBUJ.						
VERIF.						
APROB.						
FABR.						
CALID.				MATERIAL:		N.º DE DIBUJO Lado Superior
						A4
				PESO:	ESCALA:1:5	HOJA 1 DE 1



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.							
VERIF.							
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
						Trasera	
				PESO:		ESCALA:1:5	HOJA 1 DE 1



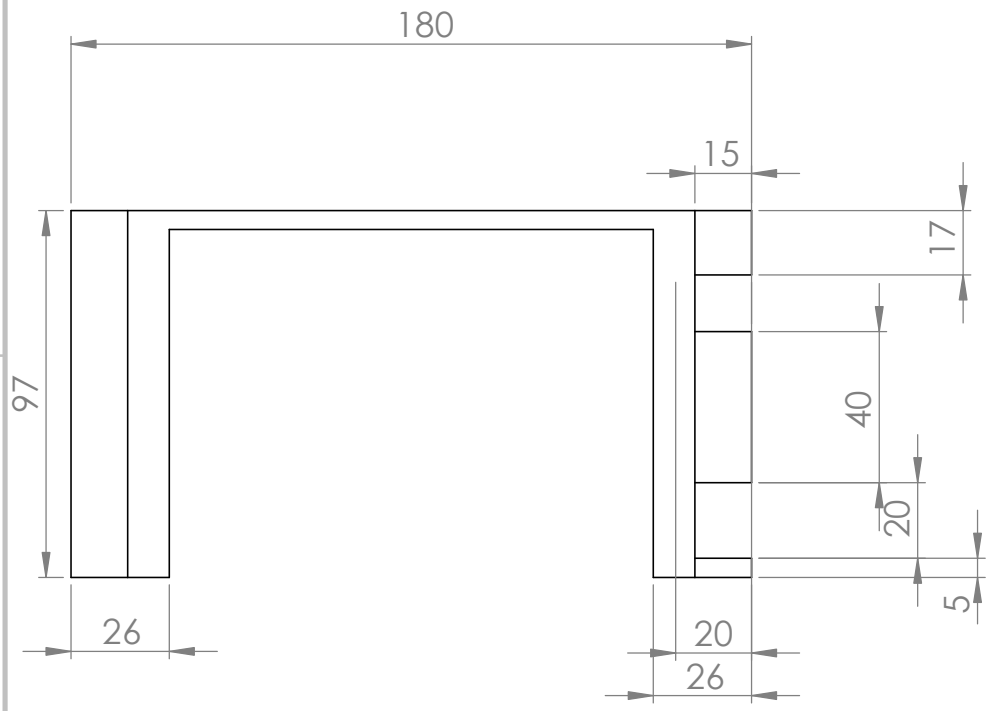
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:
DIBUJ.						
VERIF.						
APROB.						
FABR.						
CALID.				MATERIAL:		N.º DE DIBUJO
						Frontal
				PESO:	ESCALA:1:5	HOJA 1 DE 1

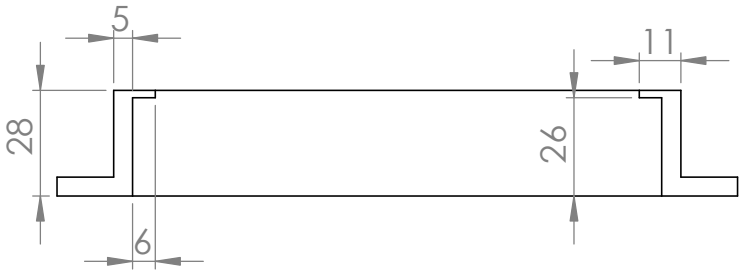


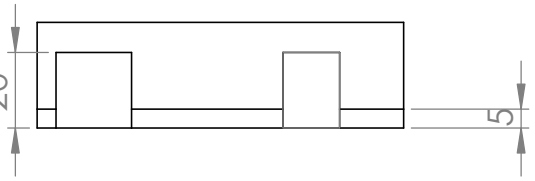
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:
DIBUJ.						
VERIF.						
APROB.						
FABR.						
CALID.				MATERIAL:		N.º DE DIBUJO
						Inferior
				PESO:		ESCALA:1:5

A4

HOJA 1 DE 1







SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.			
VERIF.			
APROB.			
FABR.			
CALID.			

TÍTULO:

MATERIAL:

N.º DE DIBUJO

Soporte Pocillos

A4

PESO:

ESCALA:1:5

HOJA 1 DE 1