Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN

Trabajo Fin de Grado

INSPECCIÓN AUTOMÁTICA DE BARNIZADO DE PCB

Alumno: Sebastián Liébanas Nicolás

Tutor: Prof. D. Pablo Cano Marchal

Prof. D. Javier Gámez García

Dpto.: Electrónica y Automática

Junio, 2019

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Electrónica

Don Pablo Cano Marchal, tutor del Trabajo Fin de Grado titulado *Inspección automática de barnizado de PCB*, que presenta Sebastián Liébanas Nicolás, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, junio de 2019

El alumno: Los tutores:

Sebastián Liébanas Nicolás Pablo Cano Marchal

Javier Gámez García

RESUMEN

La inspección del barnizado aplicado a las PCB se ha llevado a cabo tradicionalmente de forma manual. El objeto del presente trabajo es el desarrollo de un sistema prototipo para la inspección automática de dichas placas, el cual consta de dos partes diferenciadas: el diseño y construcción de la estación donde se tomarán las imágenes y la elaboración del código para el procesado de las mismas.

ABSTRACT

Traditionally, the inspection of conformal coating applied to PCBs has been done manually. The purpose of this study is the development of a prototype system for automatic inspection of these boards. It's formed by two different parts: design and building of the setup where the images will be taken and the code for image processing.

Palabras clave: PCB, Procesado de imagen, Inspección automática, Burbuja, MATLAB.

Con todo mi cariño y esfuerzo,
a mi familia,
a mis compañeros,
a mis profesores.

Inspección automática de barnizado de PCB

Índice general

Listado de figuras

Lista de tablas

1. Introducción

La idea de desarrollar una estación prototipo para la inspección de barnizado de PCB viene enmarcada dentro de la actividad del Grupo de Investigación de Robótica, Automática y Visión por Computador (GRAV) de la Universidad de Jaén, concretamente, como parte del Proyecto HEIBus. El objetivo del presente trabajo es el análisis de las placas de circuito impreso y la detección de las burbujas presentes en el barnizado de las mismas.

Este proyecto se ha compuesto de dos etapas principales: la parte hardware, que consiste en el diseño y construcción de la componente mecánica de la estación para inspección, o setup, la cual está constituida por la cámara y el actuador lineal, entre otros, y por otro lado, el software de visión, mediante el cual se analizan una serie de componentes para la búsqueda y detección de burbujas.

El código utilizado para el análisis de las PCB se ha desarrollado utilizando el entorno de programación de MATLAB, mientras que para el diseño mecánico de la estación prototipo de ha utilizado Autodesk Inventor.

1.1. Barniz

1.1.1 Concepto de barniz aplicado a las PCB

En primer lugar, es necesario conocer el objeto de estudio, el barniz que se le aplica a las placas. Así bien, se trata del recubrimiento utilizado para proteger la PCB y sus componentes, aislándolo y aumentando su resistencia, así como la rigidez dieléctrica entre conductores. Se trata de una película de un grosor que varía entre 25 y 200 µm (50 µm, típico) que se adhiere a la PCB adaptándose a su perfil irregular.

De esta forma, el barniz protege ante diferentes efectos adversos, ya sean de naturaleza mecánica, eléctrica o química, tales como:

- Humedad
- Polvo
- Radiación
- Vibraciones
- Shock térmico
- Corrosión

El uso de barniz (Conformal Coating) en las PCB (Printed Circuit Board) viene justificado por la necesidad que tienen éstas de funcionar en ambientes hostiles, a la vez que se demanda un nivel de compactación mayor de los componentes.

Así, asegurar el buen funcionamiento de la PCB puede llegar a ser vital, dado que la sociedad se encuentra inmersa en un proceso de digitalización debido al cual, la cantidad de componentes electrónicos está creciendo de forma exponencial.

Podría citarse a la industria automovilística, ya que el nivel de componentes electrónicos presente en los coches es cada vez mayor, por lo que será crucial, por ejemplo, evitar que exista algún fallo en la placa encarga del buen funcionamiento de los sistemas de seguridad.

1.1.2 Características del barniz

Sin embargo, no existe una solución perfecta, por lo que es importante elegir el recubrimiento más adecuado según las condiciones del entorno de funcionamiento de la PCB. Para ello, es necesario conocer las características del barniz.

- **Vida útil**: tiempo transcurrido entre el preparado del barniz y el comienzo de la curación del mismo. Un valor adecuado debe estar entre 30 min y 3 horas. Si es demasiado corto, puede resultar en una distribución no uniforme del recubrimiento.

- **Duración**: tiempo durante el cual se puede almacenar el barniz sin que se deteriore.

- **Viscosidad**: es necesario que la viscosidad no sea demasiado elevada para asegurar una buena distribución.

- **Contenido en sólidos**: porcentaje del barniz que curará tras la aplicación; debe ser de al menos un 20%

- **Temperatura de curación**: es un factor importante a tener en cuenta, ya que en los procesos de producción el tiempo es un factor clave que puede encarecer el producto si no se controla.

- **Propiedades eléctricas**: los recubrimientos superficiales deben tener una gran resistencia y rigidez dieléctrica para asegurar un buen funcionamiento. También debe tenerse en cuenta que pueden existir diferencias en el funcionamiento a diferentes frecuencias.

- **Permeabilidad a la humedad**: el barniz debe ser resistente a la humedad.

- **Compatibilidad química**: debe existir compatibilidad química con el sustrato de la PCB y los diferentes componentes, de la misma forma, que se debe ser resistente al contacto con sustancias químicas agresivas, como por ejemplo, hidrocarburos.

- **Durabilidad mecánica**: debe asegurar que la PCB no será dañada durante la manipulación de la misma, por abrasión, etc.

- **Reparación**: el barniz debe ser fácilmente reparable.

- **Propiedades térmicas**: la resistencia a las altas temperaturas generadas por los puntos calientes de la PCB, ya que esto puede derivar en un cambio en el resto de propiedades.

- **Resistencia a hongos**: debido a que algunos componentes orgánicos pueden servir como alimento a diferentes hongos, el barniz debe evitarlo con fungicidas.

- **Curado**: el barniz debe ser curado sin dejar defectos, tales como la contracción que puede provocar una mala adhesión o estrés mecánico a los componentes de la PCB.

TYPICAL CHARACTERISTICS OF VARIOUS CURED COATING MATERIALS

Properties	Acrylic	Urethane	Epoxy	Silicone
Volume resistivity, ohm/cm (50% RH, 23°C (73°F))	10^{15}	2 x 10^{11} 11 x 10^{14}	10^{12}-10^{17}	2 x 10^{15}
Dielectric constant, 60 cycles	3-4	5.4-7.6	3.5-5.0	2.7-3.1
Dielectric constant, 10^3 cycles	2.5-3.5	5.5-7.6	3.5-4.5	-
Dielectric constant, 10^6 cycles	2.2-3.2	4.2-5.1	3.3-4.0	2.6-2.7
Dissipation (power) factor, 60 cycles	.02-.04	.015-.048	.002-2.7	.007-.001
Dissipation (power) factor, 10^3 cycles	.02-.04	.043-.060	.002-.02	-
Dissipation (power) factor, 10^6 cycles	2.5-3.5	.05-.07	.030-.050	.001-.002
Thermal conductivity, 10^{-4}cal/sec/sq.cm/1°C/CM	3-6	1.7-7.4	4-5	3.5-7.5
Thermal expansion 10^{-5}/°C	5-9	10-20	4.5-6.5	6-9
Resistance to heat °C (°F) continuous	120 (250)	120 (250)	120 (250)	204 (400)
Effect of weak acids	None	Slight dissolve	None	Little or none
Effect of weak alkalis	None	Slight dissolve	None	Little or none
Effect of organic solvents	Attacked by ketones, aromatics & chlorinated solvents	Resists most	Generally resistant	Attacked by some

Figura 1: Características de los barnices

1.1.3 Clasificación de los barnices

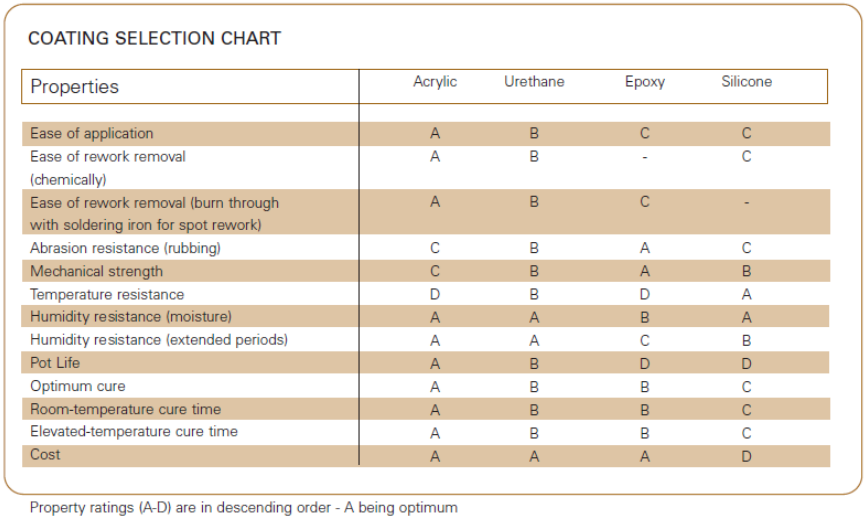
Diversas resinas naturales fueron probadas como recubrimiento superficial, pero ninguna consiguió pasar los test de estrés para comprobar su validez a altas y bajas temperaturas, por lo que en la actualidad, están hechos a base de polímeros. Se muestra a continuación las 5 opciones principales, clasificadas en base al material sólido con mayor presencia en el barniz:

- **Acrílicos**: tienen una rápida curación, así como una larga vida útil. Además, no presentan contracción y resulta ventajoso frente a las otras opciones en cuanto a las propiedades térmicas y eléctricas. Si se desea, puede eliminarse para su reparación

- **Epoxis**: presenta buena resistencia química y frente a la abrasión, sin embargo, no es una opción contra la humedad ni presenta rigidez dieléctrica. Además, es imposible de retirar.

- **Siliconas**: presentan alta resistencia térmica, rigidez dieléctrica y frente a la humedad, no así frente a químicos. Presenta una baja tensión superficial, lo que favorece una buena distribución del barniz, evitando que queden zonas sin cubrir.

- **Poliuretanos**: estos barnices presentan buenas características, salvo que, debido a su resistencia química, son difíciles de retirar para su reparación.
- **Curables con luz UV**: destaca su rapidez en la curación y un bajo contenido en COV (compuestos orgánicos volátiles). Están especialmente indicados par agrandes volúmenes de producción.



COATING SELECTION CHART

Properties	Acrylic	Urethane	Epoxy	Silicone
Ease of application	A	B	C	C
Ease of rework removal (chemically)	A	B	-	C
Ease of rework removal (burn through with soldering iron for spot rework)	A	B	C	-
Abrasion resistance (rubbing)	C	B	A	C
Mechanical strength	C	B	A	B
Temperature resistance	D	B	D	A
Humidity resistance (moisture)	A	A	B	A
Humidity resistance (extended periods)	A	A	C	B
Pot Life	A	B	D	D
Optimum cure	A	B	B	C
Room-temperature cure time	A	B	B	C
Elevated-temperature cure time	A	B	B	C
Cost	A	A	A	D

Property ratings (A-D) are in descending order - A being optimum

Figura 2: Selección de barnices

1.1.4 Métodos de aplicación del barniz

Para que cumpla su función correctamente, el barniz debe cubrir completamente la superficie a proteger, así como, no dejar al descubierto bordes afilados, como por ejemplo, los puntos de soldadura.

Para ello, además de elegir el tipo de barniz más adecuado, es crucial elegir también el método de aplicación.

- **Dipping**: consiste en sumergir la PCB en un baño del barniz, para lo cual, debe estar diseñada especialmente para ello. Es la forma más eficiente de aplicar un recubrimiento superficial, ya que, además de conseguir un reparto uniforme, permite un alto ritmo de producción (entre 350 y 700 piezas por hora).

  Un factor clave a la hora de aplicar el barniz es la velocidad de inmersión, ya que no debe ser demasiado alta, ya que esto podría provocar la

aparición de burbujas atrapadas en la PCB, lo que puede provocar un mal funcionamiento de la misma. Por ello, se recomienda, especialmente en el caso de PCB con elementos SMD (Surface Mount Device) mantener la placa sumergida hasta que dejen de emerger burbujas a la superficie.

Figura 3: Dipping

- **Selective Robotic Coating**: Se trata del método más empleado debido a que permite altas tasas de rendimiento, a la vez que utiliza maquinaria de alta precisión, evitando el desperdicio de material. Al contrario que al sumergir la pieza, mediante este método se pueden barnizar únicamente zonas específicas. Puede aplicarse de forma atomizada o líquida.

- **Spraying**: Este método es empleado para aquellas PCB que no están diseñadas específicamente para ser sumergidas. Sin embargo, este método es más caro y más lento que los casos anteriores, además requiere la extracción de humos.

Figura 4: Spraying

- **Brushing**: Al igual que el Spraying, no es un método indicado para la producción en masa, siendo difícil que no aparezcan burbujas.

1.2. Inspección automática

Todo esto está enmarcado en el contexto de “Industria 4.0”, expresión que denomina una manera de entender la industria, en la cual, prima la automatización de los procesos, la conexión a internet de los dispositivos y la interconexión entre ellos, también conocido como IoT (Internet of Things) o “Internet de las cosas”, y la combinación de las anteriores junto con grandes cantidades de información (Big Data) para transformar las fábricas convencionales en fábricas inteligentes.

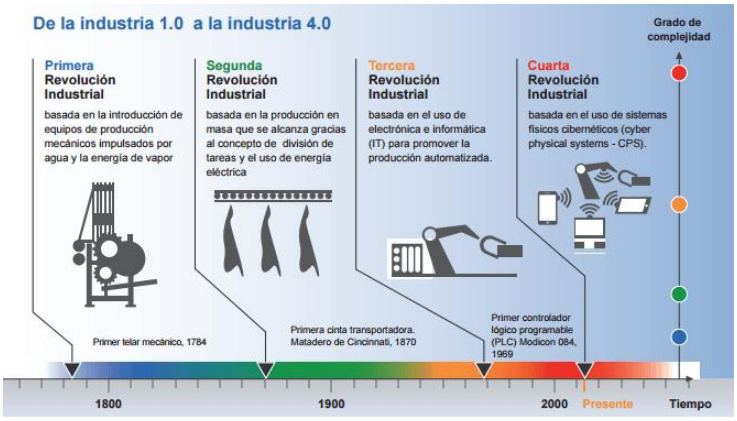


Figura 5: Evolución industrial

Se produce, por tanto, una transformación integral de la industria, ya que se optimizan los procesos de producción al establecer unos flujos de información que, de forma mucho más rápida e instantánea que por métodos convencionales, interconectan todas las fases por las que pasa el producto, ya sea de forma de forma interna (planificación, fabricación, calidad, etc.) o externa, es decir, con el cliente o usuario final.

Figura 6: Industria 4.0

De esta forma, se mejora la planificación, lo que redunda en la optimización de los recursos, algo positivo desde dos puntos de vista:

- **Económico**: al tener un mayor control de los recursos necesarios y disponibles, sean o no materiales, se evita, por ejemplo, un stock

innecesario y sus gastos derivados, o un uso poco productivo de las jornadas de los empleados.

- **Ecológico**: si se consigue reducir los recursos necesarios para producir lo mismo, esto repercutirá de forma positiva en el medio ambiente, ya que esta Cuarta revolución industrial, debe ir ligada a una componente ecológica.

Asimismo, este proceso de renovación de la industria también afectará al control de calidad, la evaluación de diferentes características de un producto o servicio, para su posterior comparación con un patrón a fin de clasificarlo según una serie de criterios, por ejemplo, si es válido o no. Por tanto, si no se asegura la calidad de un producto, los fallos presentes en el mismo terminarán por afectar negativamente a la propia empresa, produciendo pérdidas económicas, tanto por una reducción del número de clientes, como por el desperdicio del material que forma parte del producto defectuoso.

Tradicionalmente, el método empleado para el control de calidad han sido los sentidos humanos, especialmente la vista, lo cual, tiene grandes limitaciones debido a que las personas se ven afectadas por numerosos factores externos que pueden condicionar la respuesta, ya sean de tipo técnico, psicofísico, social, de organización o por el lugar de trabajo.

Factors	Examples
Technical	Type of defects; Defect visibility; Quality level; Standards (tests); Control automation; Other
Psychophysical	Age; Sex; Observation skills; Experience; Temperament; Creativity; Other
Organizational	Training; Scope of decision making; Feedback; Precise instructions; Other
Workplace environment	Light; Noise; Temperature; Work time; Workstation organization; Other
Social	Team communication; Pressure; Isolation; Other

Figura 7: Factores que afectan a la eficiencia de la inspección visual

Durante la primera mitad del siglo XX se pensaba que era la mejor opción, sin embargo, entre las décadas de los 50 y 70, se observó que en realidad era el eslabón más débil del proceso de control de calidad, apostando por una automatización total del proceso. Fue ya durante los años 80 y 90 cuando los responsables se dieron cuenta de que, en la práctica, no se podía llevar a cabo de forma exitosa el 100% de las veces debido a la flexibilidad que poseen los sistemas de visión automáticos.

Si las condiciones del entorno se ven modificadas, los resultados pueden verse afectados, por ejemplo, si cambia la iluminación o la pieza no viene siempre orientada de la misma forma, pueden dejar de apreciarse

defectos o darse el caso de que se presenten falsos positivos. Por el mismo motivo, resulta complicado construir sistemas que analicen varios modelos de producto. Se llegó a la conclusión de que un sistema óptimo es aquel que combina la inspección automática con la asistencia humana, un sistema híbrido.

Así, la inspección humana presentará una serie de ventajas y desventajas, con respecto a los sistemas automáticos de inspección visual.

Ventajas de la inspección visual humana vs inspección automática

- Cambios en la iluminación u orientación no tienen carácter crítico
- Capacidad deductiva
- No se ve afectada por cambios de color
- Capacidad de adaptarse a variaciones con rapidez

Desventajas de la inspección visual humana vs inspección automática

- Se ve afectada por la fatiga, hora y día.
- No se puede mantener la eficacia en el tiempo
- Suele ser más lenta que la inspección automática
- Variabilidad de resultados entre diferentes personas

Hasta el momento, la inspección en planta de las placas analizadas en este proyecto se ha llevado a cabo de manera manual. El operario situaba la PCB bajo una lámpara equipada con un aumento de 4x e iluminación UV. Esto produce un gran nivel de fatiga en el operario, ya que sumado al hecho de que se trata de una tarea repetitiva, el diámetro medio aproximado de las burbujas a detectar es de 500µm.

Figura 8: Lámpara UV

A esto se le añaden los efectos nocivos para la salud de la luz ultravioleta, aquella cuya longitud de onda se encuentra entre los 400nm y los 15nm, lo que termina de justificar la realización de estudio de viabilidad para la automatización del proceso.

1.3. Objetivos

Objetivos generales

- Desarrollo de un sistema prototipo de inspección basado en visión por computador para la detección de burbujas en PCB.
- Estudio de viabilidad para posible implementación en la industria.

Objetivos específicos

- Selección de material y rango espectral de trabajo: fuentes de iluminación, cámara y utillajes.
- Diseño y elaboración de la estación prototipo necesaria para la adquisición de imágenes.
- Desarrollo del software para el análisis de imágenes y detección de burbujas.

1.4. Contexto

Este proyecto de control de calidad se enmarca dentro del Grupo De Investigación de Robótica, Automática y Visión por Computador de la Universidad de Jaén (GRAV), en el cual se llevan a cabo tanto proyectos de investigación como el desarrollo e implementación de soluciones industriales para el sector agro y automovilístico, entre otros.

En estrecha colaboración con el GRAV se encuentra, la empresa de base tecnológica ISR, creada como spin-off del mismo en el año 2016, cuenta con una larga experiencia en estas áreas, siendo especialistas en la automatización de procesos y en la integración sensorial. Entre otros, han desarrollado equipos capaces de realizar tareas de inspección visual automática.

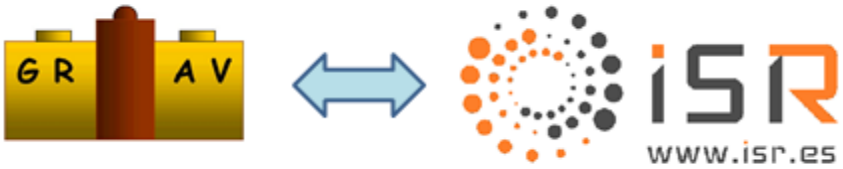


Figura 9: GRAV-ISR

Además, todo esto se encuentra dentro de un caso de estudio dentro del Proyecto HEIBUS. Se trata de un proyecto a nivel europeo cuya misión es la de acercar y favorecer las relaciones entre las empresas y las universidades de forma que tanto estudiantes y profesores como personal de la empresa colaboren en la resolución de problemas reales propuestos por esta última.

Figura 10: HEIBus

Para ello, dentro del proyecto se han implementado 3 modelos de colaboración:

- **Multidisciplinary Real Life Problem Solving** (RLPS)
  Se forman 3 equipos de alumnos de carácter multidisciplinar e internacional que compiten por ofrecer una solución óptima al problema planteado por la empresa. Se llevan a cabo durante todo un cuatrimestre, donde se lleva a cabo una semana intensiva en la que los miembros se conocen en persona y trabajan de forma presencial, además de reuniones

virtuales para coordinar el proyecto. Al final, la empresa otorga una puntuación y elige un equipo ganador.

- **Expert Level Real Life Problem Solving** (EXPERT)
  En este modelo se forman tres equipos de expertos de tres países diferentes y un cuarto formado por personal de la propia empresa que plantea el caso. Esto hace posible que empresas locales puedan disponer de un panel de expertos de carácter internacional, acelerando la implantación y mejora de las nuevas tecnologías, democratizando la Industria 4.0. Aquí se ubica nuestro trabajo.

- **Flexible Student Mentoring by Companies** (Flex Mentoring)
  Este último modelo consiste en que la empresa tutoriza a un grupo de estudiantes durante un periodo de tiempo, lo que desarrolla sus habilidades y supone un contacto mayor con el mundo laboral.

El proyecto completo consiste en la eliminación de las burbujas formadas en la PCB durante el proceso de barnizado, dividiéndose en 4 aspectos: prevención, reacción, detección y testeado, siendo la detección la parte asignada al equipo de la Universidad de Jaén. Debido a lo expuesto anteriormente, al equipo de expertos de la Universidad de Jaén, se le encargó la realización un estudio de viabilidad para la detección y caracterización de las burbujas mediante visión artificial.

Para dar una idea general del desarrollo del proyecto se comentará de forma breve aquellos equipos de los que ha sido necesario disponer y cómo están relacionados.

La primera tarea a la que hubo que enfrentarse fue la de seleccionar que tecnologías existentes emplear; los elementos a detectar son burbujas cuyo tamaño no excede del milímetro, por lo que decidió emplearse una cámara lineal, las cuales destacan por su gran resolución. Sin embargo, el uso de esta cámara no tiene sentido si no existe un movimiento relativo con respecto a ella, por lo que había que introducir un dispositivo que lo hiciese posible, un actuador lineal.

Además de la iluminación, para lo cual se optó por barras de LED, fue necesario diseñar un setup a modo de prototipo para realizar pruebas, empleándose una estructura regulable ya existente para fijar la cámara. Sin embargo, para colocar la iluminación ha hecho falta diseñar una estructura fabricada mediante perfiles estructurales. Para fijar las barras de LED a esta estructura se han diseñado unos enganches, los cuales se imprimieron en 3D, así como el utillaje necesario para situar la PCB sobre el actuador lineal.

1.5. Estado del arte

Aunque se haya mencionado en varias ocasiones la expresión Industria 4.0 y se trate de automatizar, la inspección de PCB no es algo novedoso. Prácticamente desde la invención de las placas de circuito impreso éstas se han inspeccionado en busca de conseguir la mayor calidad posible en el producto.

Fue en 1903 cuando Albert Hanson patenta un dispositivo con forma de hoja, con diferentes capas capaces de conducir, protegidas estas por un material aislante, precursora de las actuales PCB. Aunque no fue hasta las décadas de 1940 y 1950, con el trasfondo de la 2ª Guerra Mundial y la Guerra Fría, cuando la electrónica tuvo su auge, y con ello, el desarrollo de las PCB.

Uno de los hitos fue, cuando en 1956, un grupo de científicos estadounidenses pertenecientes al ejército de EE.UU. patentó el primer proceso de fabricación de PCB. Durante los años 70 y 80, la complejidad de las PCB fue en aumento, debido a las demandas de una industria tecnológica en auge. Esto, junto con el tamaño cada vez menor de los componentes provocó una alta densidad de los mismos en las placas, lo que hizo patente la necesidad de realizar controles de calidad más exhaustivos a las placas fabricadas.

Es en esta década cuando empiezan a aparecer estudios y artículos valorando la posibilidad de diseñar y desarrollar sistemas automatizados para la inspección de PCB, sobre todo por parte de investigadores japoneses. En uno de ellos se introduce como novedad con respecto al anterior el uso de luz fluorescente para detectar defectos en la placa, tales como cortocircuitos, ausencia de elementos, etc.

Sin embargo, no es hasta la década actual cuando se extiende la aplicación de los barnices a las placas para protegerlas y con ella, el desarrollo de soluciones industriales para la automatización del proceso de inspección. Una de ellas es la máquina desarrollada por la empresa IVS (Industrial Vision Systems), IVS-PCBi-C, la cual es capaz de funcionar totalmente integrada en el proceso de producción y de comprobar tanto como la presencia de los componentes como del barniz, así como los defectos presentes en la PCB.

Figura 11: IVS-PCBi-C

Otra solución comercial ofertada es la proporcionada por la empresa Nordson, la máquina FX-UV AOI, la cual, a diferencia de la máquina anterior, solo es capaz de detectar la presencia de barniz y burbujas en el mismo. Esta inspección la realiza gracias a un sistema de iluminación ultravioleta desarrollado por la misma compañía y con la posibilidad de contar con cámaras con cierto ángulo respecto a la vertical.

Es capaz de analizar de forma automática la calidad, consistencia y grosor del barnizado, mostrando los resultados en unos pocos segundos. Además estos pueden ser almacenados y revisados en un momento posterior.

Su potencial radica en que una mínima programación es requerida para modificar el modelo de placa que se analiza: menos de 30 minutos. Es capaz también de leer códigos de barras de cara a obtener la trazabilidad del producto.

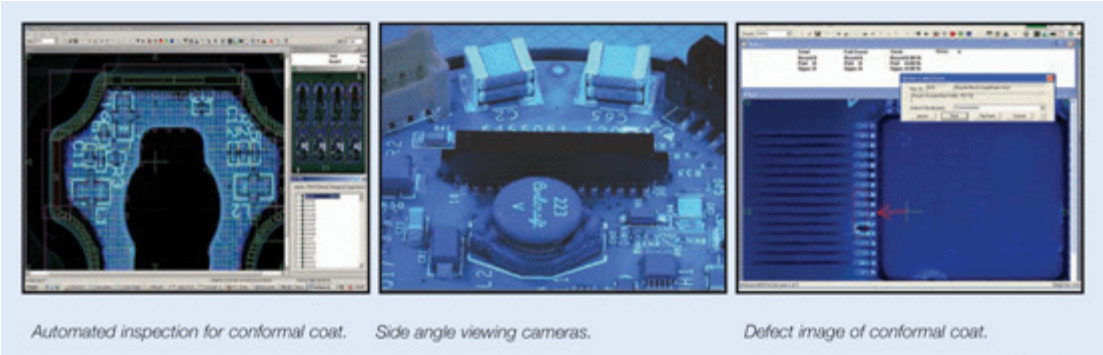


Figura 12: Características FX-UV AOI

Figura 13: FX-UV AOI

Cabe destacar, por último, otro ejemplo de sistema para la inspección de PCB desarrollado por CyberOptics, CX150i. De la misma forma que los anteriores, es capaz de analizar el barniz, además de la presencia o no de componentes, alcanzando tasas de análisis de $100\ \mathrm{cm}^2/\mathrm{s}$. Para ello utiliza luz ultravioleta estroboscópica y luz blanca.

Figura 14: CX150i

2. Diseño del setup

Como se ha detallado en la introducción, este proyecto consiste en la inspección del barnizado de PCB, concretamente, se centra en la detección y caracterización de las burbujas que pueden aparecer en ella tras la aplicación del mismo.

Para poner en situación, es necesario destacar que este proyecto partió prácticamente desde cero, teniendo como condiciones de partida 14 muestras de un único tipo de placa y la misión de realizar un estudio de viabilidad para la detección de esas burbujas. Otro dato con el que se contó fue la forma en la que actualmente se estaba llevando el análisis, bajo la incidencia de luz ultravioleta. Por tanto, ante este problema tan abierto a priori, hubo que establecer una estrategia a seguir, empezando por determinar con qué tecnologías y componentes se iba a empezar a trabajar.

A continuación se describirá detalladamente cada elemento de la estación de pruebas, clasificados según la función que desempeñan.

2.1 Iluminación

La comprobación visual del barnizado es posible debido a que éste dispone de una traza ultravioleta, haciendo que pueda ser inspeccionado bajo este tipo de iluminación. Si bien, existen diversas soluciones, tales como las bombillas fluorescentes tradicionales, lo más lógico era decantarse por un opción basada en LED.

2.1.1 Tecnología LED

En la actualidad y desde hace un tiempo atrás, la industria de la iluminación está apostando por el LED debido a sus ventajas frente a otras tecnologías anteriores, llegando incluso a conseguir que las bombillas incandescentes y fluorescentes tradicionales sean descatalogadas. Entre las principales ventajas de los LED frente a los sistemas de iluminación tradicionales está su gran eficiencia, es decir, la cantidad de lúmenes emitidos en relación a la potencia consumida. Además, al estar basados en el efecto físico antes descrito, éstos apenas se calientan en comparación con las bombillas fluorescentes o incandescentes.

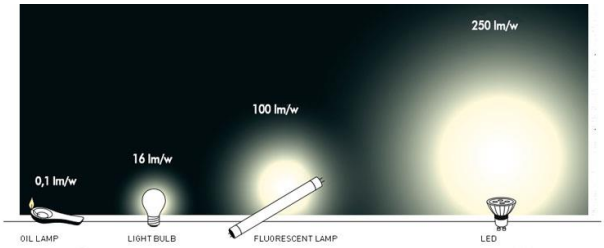


Figura 15: Eficiencia iluminación

2.1.2 Barras de iluminación LED

La tecnología elegida para la iluminación será por tanto LED, sin embargo, pueden distribuirse de múltiples formas: de forma matricial, radial, etc. En este caso, se ha optado por una iluminación en forma de barra.

Tal como se indica, la cámara empleada para la realización de este proyecto es de tipo lineal, lo cual significa que únicamente capta una línea de píxeles. Debe emplearse, por tanto, un sistema de iluminación que garantice que esa línea se encuentra bien iluminada y de manera eficiente.

En el caso de optar por un sistema de iluminación tipo foco, la eficiencia sería muy baja, ya que el ángulo de salida es demasiado elevado. De la misma forma, tampoco sería apropiada una iluminación de tipo radial, ya que la línea capturada quedaría escasamente iluminada.

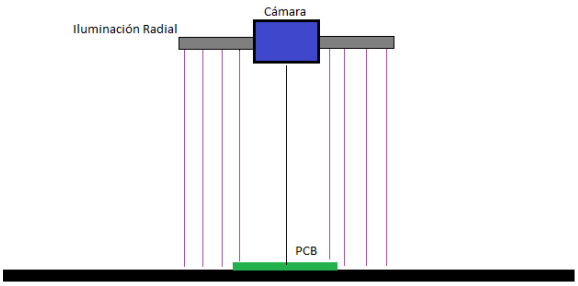


Figura 16: Iluminación radial

La solución óptima para este escenario es el empleo de una barra de iluminación, la cual maximiza la cantidad de lúmenes que refleja la línea capturada, optándose finalmente por barras EffiFlex.

Figura 17: Barras Effilux

En un inicio, se comenzó a trabajar únicamente con una barra de iluminación, pero cuando se empezaron a obtener imágenes nítidas se hizo patente la necesidad de disponer de una segunda debido a que los componentes, al no ser objetos planos, proyectan una sombra en una de sus caras, resultando imposible detectar burbujas en esa zona. Sin embargo, cuando se recibió la segunda barra, ésta resultó no ser el mismo modelo que la primera, existiendo, entre otros, diferencias en la potencia.

2.1.2.1 Características generales

Se trata de una barra metálica con una serie de LED cuya longitud de onda puede variar desde el ultravioleta hasta el infrarrojo, con una serie de características comunes a todas ellas.

Tabla 1: Características generales barra Effilux

Características		
Electrónicas	Alimentación	24V DC
	Modo de Iluminación	Continuo o Estroboscópica
	Conector	M12 4 pines
	Consumo	Depende del número de LED
Ópticas	Longitud de Onda	Desde UV a IR y blanco
	Sistema de proyección	4 posibles ángulos
	Opciones ópticas	Difusor, polarizador, etc
Mecánicas	Dimensiones	51x49 mm (Varias longitudes)
	Cierre	2 raíles
	Material	Cuerpo: Aleación de aluminio Pantalla: Metacrilato
Ambientales	Temperatura de operación	Entre 0℃ y 50℃
	IP Code	IP40 (Opción IP54)

2.1.2.2 Mecánica

Toda la superficie de la caja dispone de aletas, ya que está especialmente diseñada para evacuar el calor generado en el interior de la forma eficiente. Además, dispone de dos carriles en el exterior que permiten la fijación de la misma a un perfil estructural o a otra estructura cualquiera. Para ello, el fabricante recomienda utilizar tornillos y tuercas M4 y M6 para realizar las fijaciones, sin embargo, nosotros hemos preferido utilizar unos enganches que posteriormente se comentarán, los cuales se han diseñado con Autodesk Inventor y fabricado mediante impresión 3D. Estos se deslizan sobre los carriles de un perfil estructural a la vez que posee un extremo que lo hace por el carril posterior de la barra de iluminación.

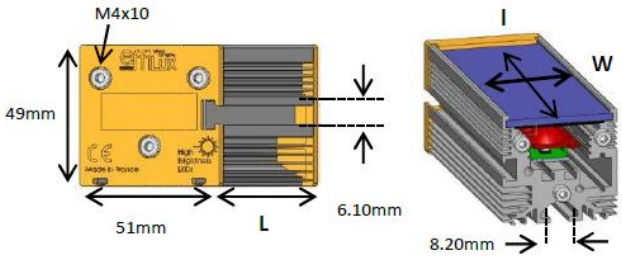


Figura 18: Sección barra Effilux

Estas barras se fabrican para varias longitudes hasta 4m variando el número de LED y siguiendo la ecuación que se muestra a continuación, siendo L la longitud mecánica de la barra, tal como se indica en la figura 18.

$$L1(mm) = 20 \cdot \text{Cantidad de LED} + 35mm$$
$$L2(mm) = 40 \cdot \text{Cantidad de LED} + 35mm$$

Se establece la diferenciación entre L1 y L2 debido a que existen 2 tipos de barras, las barras tipo L2 contienen la mitad de LED que las de tipo L1 a igualdad de longitud, por lo que tendrán menor potencia. Esto justifica lo expuesto al inicio de este apartado: la primera barra de la que dispusimos era del primer tipo, mientras que la que se solicitó después era del segundo, lo cual nos originó problemas, que finalmente se solventaron.

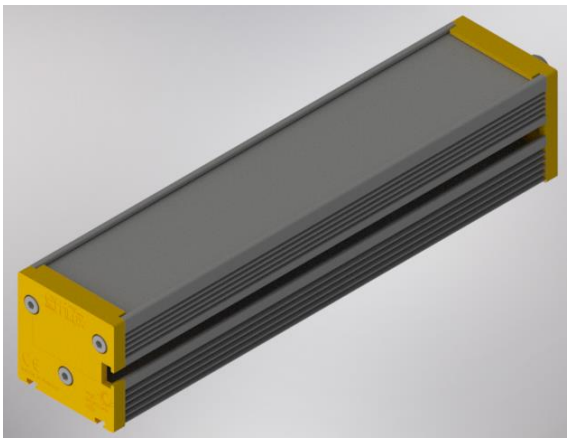


Figura 19: Renderizado barra Effilux

2.1.2.3 Electrónica

Para su alimentación, dispone de un conector tipo M12 con 4 pines, tal como se aprecia en las imágenes anteriores. Así, el cable que acompaña a la barra de iluminación dispone de cuatro contactos: el marrón y azul que sirve para alimentar, el blanco no está conectado, y otro negro, el cual puede cumplir dos funciones, o bien como regulador de intensidad (dimmer) o bien servir como disparador (trigger) pudiendo sincronizarlo con el disparo de la cámara. Si se quiere trabajar en modo continuo y/o al máximo de su

capacidad lumínica, como es el caso de este trabajo, esta pin debe estar conectado a +24V.

Contact arrangement[1]	Number	Color Contact	Designation
 M12 Male connector	1	Brown	+24V
	2	White	n.a.
 M12 Power Male connector	3	Blue	GND
Connector depends on electrical power consumption	4	Black	TRIG[2] - max 24V Analog Voltage

Figura 20: Pinout barra Effilux

Estas barras de iluminación estarán conectadas a fuentes de alimentación de tal manera que las conexiones de los diferentes pines se realicen de acuerdo al esquema indicado prestando especial atención a no invertir la polaridad y a que el pin del trigger/dimmer debe estar también conectado, en nuestro caso, a 24V.

En cuanto al consumo, si se trabaja en el modo continuo, por los LED circula una corriente de 350mA, existiendo al principio un pico de corriente de 1A que puede ser apreciado a simple vista al conectar los LED a la corriente.

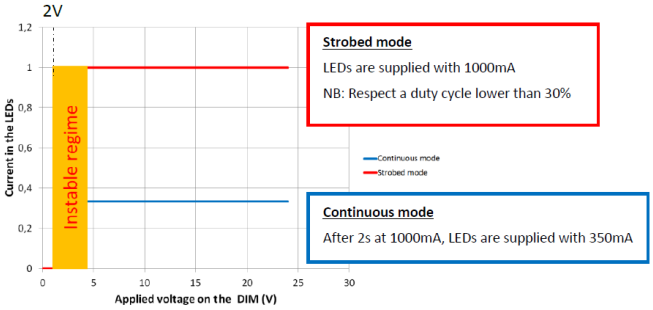


Figura 21: Consumo barra Effilux

2.1.2.4 Óptica

Además de las características mecánicas y electrónicas, es crucial comentar las características relacionadas con la óptica. Como se menciona, para algunos modelos de barras Effiflex, es posible modificar el ángulo de salida de la luz gracias a unas lentes de las que dispone.

En el interior de la caja existen unos carriles por los cuales pueden deslizarse estas lentes, permitiendo así cuatro posibles ángulos. Además, estas lentes tienen carácter modular, por lo que son perfectamente intercambiables y son compatibles con cualquier tamaño de barra. Tan solo es necesario desatornillar los tres tornillos que tiene en el lateral opuesto al conector, retirar la tapa, deslizar las lentes y ubicarlas en la posición requerida

Figura 22: Óptica barra Effilux

La longitud óptica, distancia que cubren los LED en la barra se calcula siguiendo la siguiente ecuación, siendo L equivalente a la longitud mecánica antes descrita como L1 y L2.

$$Lop(mm) = L(mm) - 20mm$$

En este proyecto se ha trabajado con los dos modelos siguientes:

Tabla 2: Datos barras Effilux

Barras Effilux		
Barra	EFFI-FLEX-10-405-TR-P0	EFFI-FLEX-L2-5-405-SD-P3
Número de LED	10	5
Longitud de Onda	405nm	405nm
Tipo de pantalla	Transparente	Semidifusa
Ángulo de salida	90° (P0)	10° (P3)
Potencia	44W	20W
Longitud caja	235mm	235mm
Longitud óptica	215mm	215mm
Refrigeración	Pasiva	Pasiva

Cabe destacar que esta barra, a diferencia de la anterior, si dispone de las lentes que permiten regular el ángulo de salida, por lo que aunque venga indicado $10°$ como ángulo de salida, se puede modificar adoptando los valores de $25°, 45°$ o $90°$.

Figura 23: Barra Effilux 1 (44W)

2.2 Actuador lineal

La comprobación visual del barnizado es posible debido a que éste dispone de una traza ultravioleta, haciendo que pueda ser inspeccionado bajo este tipo de iluminación. Si bien, existen diversas soluciones, tales como las bombillas fluorescentes tradicionales, lo más lógico era decantarse por un opción basada en LED.

Debido a que se ha usado una cámara lineal por la gran resolución que ésta ofrece, es necesario tener un movimiento relativo de la PCB con respecto a la cámara. Este movimiento se conseguirá gracias a un actuador lineal. En concreto, se cuenta con los siguientes elementos necesarios para el funcionamiento del actuador:

- Actuador eléctrico
- Controlador
- Conexión de alimentación
- Cable de comunicación con el actuador
- Cable USB para conexión controlador-PC

2.2.1 Hardware

2.2.1.1 Características

En cuanto al actuador lineal, se trata del modelo LEFS40RA-1000-R36PD de SMC, lo cual indica las siguientes características acerca de él:

- **LEF**: Indicador propio de la gama de actuadores lineales eléctricos.
- **S**: Ball Screw drive, es decir, tiene un husillo de bolas. Esto se traduce en una eficiencia mecánica de hasta el 90% frente al 50% de un husillo Acme tradicional, lo que supone un ahorro energético importante. Sin embargo, son más caros debido al ser más complicada su fabricación.
- **40**: Anchura del canal del actuador
- **R**: Right side parallel-type, indica en qué lado se encuentra el motor con respecto al eje de desplazamiento.
- **A**: Indica el tamaño del LEAD, que en este caso corresponde a 20 mm
- **1000**: Indica el tamaño de la carrera, que para nuestro actuador lineal es de 1000mm
- **R**: El cable empleado es de tipo robótico, es decir, permite un mayor grado de flexibilidad que un cable estándar.
- **3**: Expresa la longitud del cable del actuador, que en este caso es de 3m

- **6P**: Controlador del tipo “Step data input PNP”, es decir, se le pasa el número de la posición para que se desplace hasta ella, habiéndose guardado previamente estas posiciones.
- **D**: Diseñado para montaje sobre carril DIN.

2.2.1.2 Mecánica

Es necesario comentar brevemente las diferentes partes mecánicas de las que dispone el actuador antes de describir su funcionamiento.

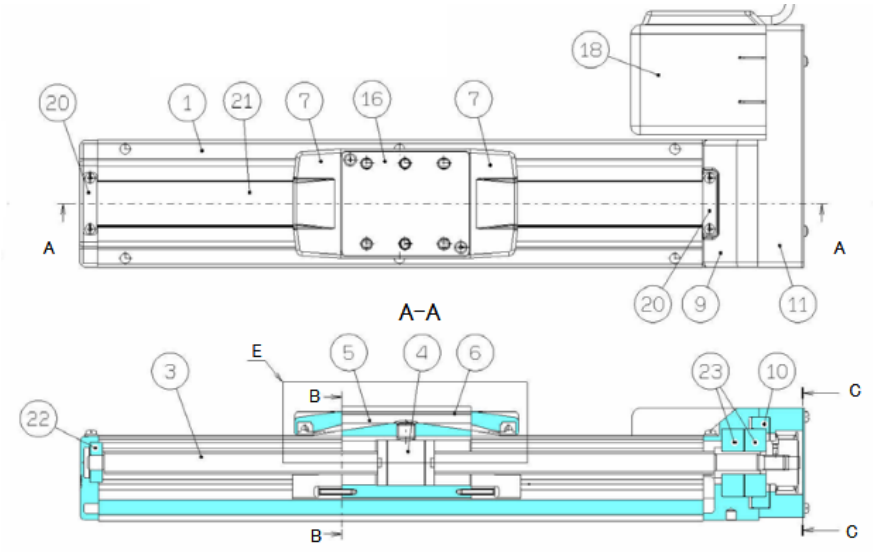


Figura 24: Plano actuador lineal

Si nos fijamos en la figura 24, se puede observar como dispone de varias partes diferenciadas. La parte 1 corresponde con el cuerpo del lineal, el cual está fabricado en una aleación de aluminio. El husillo de bolas se corresponde con los números 3 y 4, gracias a los cuales, al girar el motor (localizado bajo la carcasa con el número 18), se desplaza el plato o base, correspondiente con la pieza número 16, sobre la que irá atornillado el utillaje en el que se depositará la PCB.

2.2.1.3 Controlador

Para el funcionamiento de este dispositivo es necesario establecer una serie de comunicaciones entre el actuador y el controlador y entre éste y el PC.

Figura 25: Controlador actuador lineal

El primer paso es alimentar al controlador mediante el puerto CN1, donde se conectar un adaptador específico como el que se ve en la figura 25, el cual posee 5 terminales de conexión.

Tabla 3: Conexiones alimentación Controlador

Alimentación Controlador	
0V	GND común a M24V, C24V, EMG y BK RLS.
M24V	Alimentación positiva para el motor.
C24V	Alimentación positiva para el control.
EMG	Si no se le introduce una señal de 24V interrumpe el funcionamiento y apaga el motor; se enciende como un pulsador de emergencia.
BK RLS	Funciona de manera similar al terminal EMG, con la diferencia de que éste desactiva el freno del motor..

Por otra parte, la conexión actuador-controlador se produce gracias al cable del actuador, el cual se forma al unir el bus que sale del actuador (terminal hembra) con el que llega al puerto CN3 del controlador (terminal macho). El usuario no interactúa con este puerto, ya que es el controlador el que “traduce” la información enviada desde el PC o PLC.

Para conectarlo al PC solo es necesario conectar el cable de comunicación al puerto CN4 mediante el terminal RJ45 del que dispone, lo que hará necesario incluir una unidad de conversión que lo convierta a USB.

Sin embargo, conectarlo a un PLC es algo más laborioso, ya que es necesario conocer cada una de las entradas y salidas del actuador y el procedimiento a seguir para un correcto funcionamiento. Sería necesario disponer de un cable tipo BUS, conectándose un extremos al

puerto CN5 del controlador y separando en el otro cada uno de los puntos individuales de cara a acceder a cada una de las entradas y salidas, las cuales se ven reflejadas en la tablaX.

Tabla 4: Entradas controlador

Control PLC (Entradas actuador)		
No	**Función**	**Descripción**
A1	COM+	24V para la alimentación de las señales de entrada y salida.
A2	COM-	0V para la alimentación de las señales de entrada y salida.
A3	IN0	Código binario de la posición seleccionada para el desplazamiento. Se recuerda que esto se graba previamente en el controlador mediante el PC. LSB: IN0 // MSB: IN5
A4	IN1	
A5	IN2	
A6	IN3	
A7	IN4	
A8	IN5	
A9	SETUP	Si se activa realiza la inicialización del lineal, llevándolo a la posición inicial.
A10	HOLD	Si se está ejecutando un desplazamiento y se pone a nivel activo, deja en pausa la operación, reiniciándose al dejarlo a nivel bajo.
A11	DRIVE	Cuando se activa, el sistema escanea la posición indicada en IN0-IN5 y se desplaza hasta ella.
A12	RESET	Si se activa produce un Reset en la alarma
A13	SVON	Se pone a 1 o 0 para encender o apagar el motor.

Tabla 5: Salidas controlador

Control PLC (Salidas actuador)		
No	**Función**	**Descripción**
B1	OUT0	Cuando se activa DRIVE para ejecutar un movimiento y esta señal se vuelve a poner a 0, mediante estos bits se muestra la última posición seleccionada. LSB: OUT0 // MSB: OUT5
B2	OUT1	
B3	OUT2	
B4	OUT3	
B5	OUT4	
B6	OUT5	
B7	BUSY	Se activa durante el movimiento.
B8	AREA	Se activa cuando el actuador está situado entre unos límites configurables.
B9	SETON	Se pone a nivel alto cuando la información de posición es establecida.
B10	INP	Se utiliza para varios usos depende de la situación.
B11	SVRE	Indica si el motor está o no encendido.
B12	ESTOP	Señal sincronizada con EMG (CN1).
B13	ALARM	Si todo está correcto está activo.

Aunque en las primeras etapas del proyecto se comenzó a implementar el manejo del actuador lineal mediante un PLC, no se completó debido a que no resultaba imprescindible para realizar el estudio de viabilidad. En el hipotético caso de instalación en un ambiente industrial esta sería la forma de proceder. Para este caso bastaba con un control por parte del usuario desde el software propio que el fabricante suministra.

2.2.2 Software

En primer lugar, es necesario descargar de la página web oficial tanto el software ActController que permite manejar el actuador y configurar todos los parámetros, como los driver necesarios para que el PC reconozca el controlador.

2.2.2.1 ActController

Este software posee dos modos de funcionamiento, eligiéndose uno u otro dependiendo de las necesidades existentes.

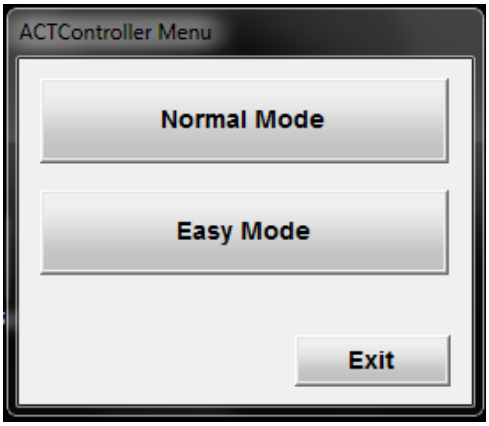


Figura 26: ActController Modos

- **Easy Mode**: permite únicamente editar los datos para el movimiento (posición, velocidad, aceleración) de tal forma que se guardan automáticamente en el controlador al modificarlos.
  También puede realizarse un test de movimiento, pudiendo llevarlo a cualquiera de las posiciones guardadas, o realizar desplazamientos puntuales que no quedarían registrados con las condiciones que se le indiquen. Permite además obtener la posición en la que se encuentra el cabezal en un momento determinado.

  Igualmente, también se pueden realizar desplazamiento de forma absoluta o relativa, es decir, incrementos de posición de la magnitud indicada, lo cual puede ser útil para realizar movimientos rápidos y acercamientos precisos y exactos en posiciones de interés.

Si no detecta ningún dispositivo conectado o no se ha configurado aún la conexión, aparecerá un mensaje al pulsar sobre Easy Mode indicándolo y preguntando si se desea editar la configuración de comunicación.

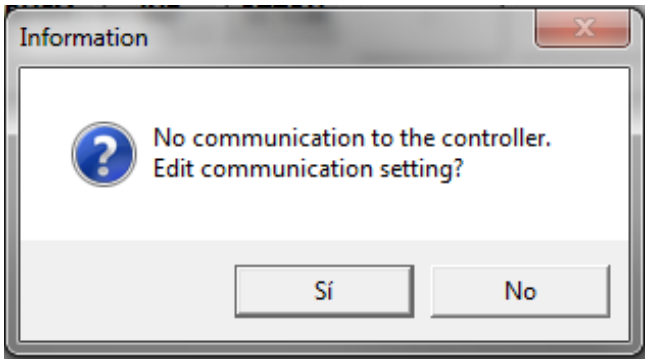


Figura 27: ActController - Actuador no detectado

Al seleccionar que sí se desea editar aparece una ventana de configuración en la cual se puede seleccionar el puerto serie al cual está conectado el controlador y la tasa de baudios (baud rate), es decir, la velocidad a la que se envían los bits por dicho puerto. En nuestro caso, el puerto al cual se conectó fue el COM7 y la velocidad se fijó a 38400bps (baudios por segundo).

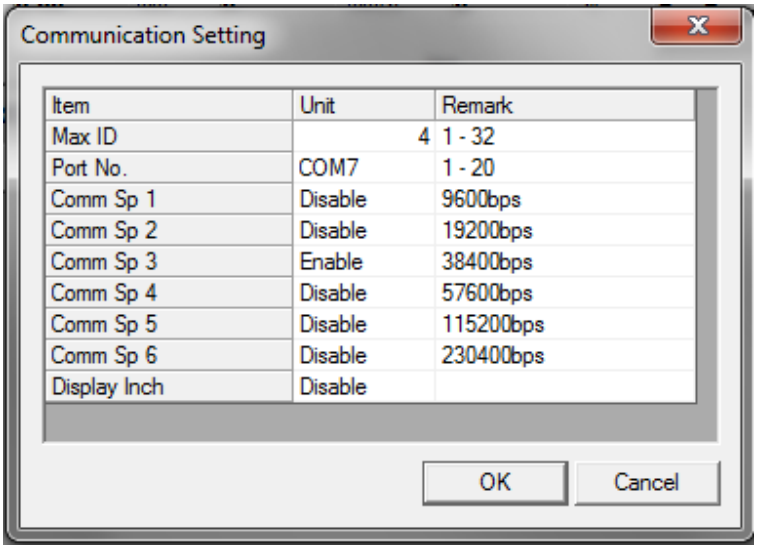


Figura 28: ActController - Configuración

Una vez completado con éxito este procedimiento, el programa reconoce el controlador y aparece la pantalla de control, en la cual se pueden modificar los datos de posición, desplazar el actuador, etc. En la esquina superior izquierda aparece el modelo de actuador lineal, mostrándose debajo de esta etiqueta los datos relativos a la posición,

velocidad y fuerza en tiempo real. Puede conmutarse además entre los modos de funcionamiento principales:

- **Monitor**: únicamente visualizar y monitorizar los datos antes indicados y el estado de algunas salidas del controlador. Cuando se conmuta a Test, aparece que advierte al usuario de que el motor va a activarse y compruebe la seguridad

- **Test**: en este caso se permite el control del actuador, pudiendo guardar los datos de movimiento y desplazar el cabezal, ya sea seleccionando cualquiera de las posiciones guardadas o moviéndolo de forma manual con la velocidad indicada.

  El procedimiento para desplazar el lineal a una posición guardada es simple: se selecciona la posición en el cuadro donde se encuentran los datos de movimiento y se pulsa la tecla “Drive”; el lineal se desplazará hasta la posición requerida y se detendrá allí a la espera de una nueva orden. Si durante el transcurso de una acción de movimiento se le ordena un nuevo desplazamiento interrumpe el que esté ejecutando y realiza el nuevo.

  Por la otra parte, para desplazar el lineal de forma libre, su pulsa sobre las flechas de “Jog” desplazándose para cada uno de los lados si ésta se encuentra pulsada con la velocidad que tenga indicada en la barra inferior izquierda “Move Speed”. Otra forma es realizar movimientos relativos de la distancia indicada en el label inferior “Move” y presionando una de sus teclas adyacentes.

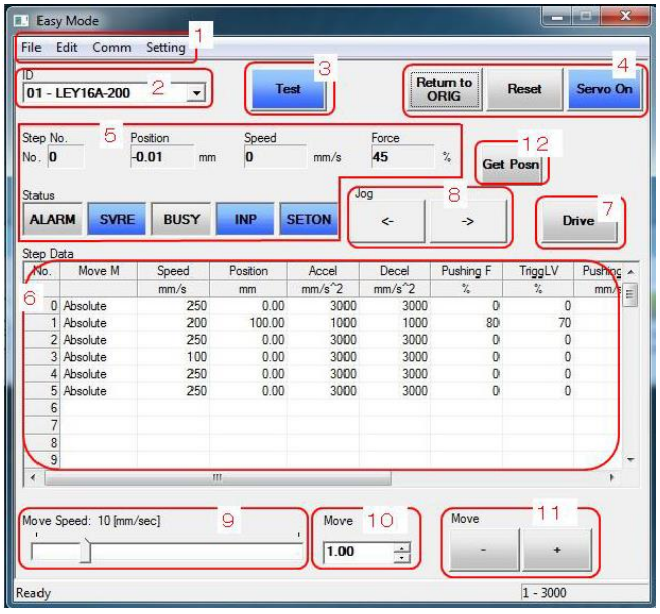


Figura 29: ActController - Easy Mode - Vista general

- **Normal Mode**: Este modo tiene un mayor número de funcionalidades con respecto al Easy Mode, ya que pueden comprobarse las diferentes señales, además de modificar parámetros y cargar o descargar información del controlador. De igual forma, en este modo también se pueden modificar los datos de posición.

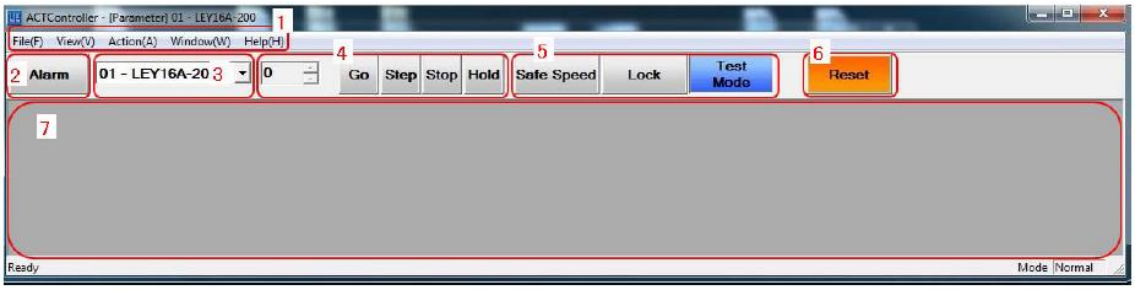


Figura 30: ActController - Normal Mode - Vista general

En este caso, si no detecta el controlador muestra un mensaje avisando de ello. Para modificar la configuración relativa a las comunicaciones se accede a través del menú superior (Action>Settings>System), accediendo al mismo menú que en el caso de Easy Mode.

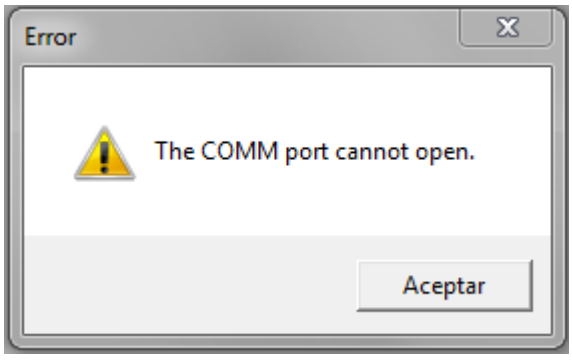


Figura 31: ActController - Puerto COM no detectado

Si se pulsa sobre el botón “Alarm” aparece una ventana con el estado actual y un histórico de alarmas, lo cual es útil para la depuración de errores. Junto a esto aparece la referencia del actuador lineal con el que se está trabajando.
Junto a esto aparece un grupo de botones (“Go”, “Step”, “Stop”,”Hold”) cuya utilidad es la de ejecutar el código de prueba que se le haya cargado al controlador mediante este modo.

A continuación, “Safe Speed” es utilizado para limitar la velocidad durante la fase de test, mientras que “Lock” bloquea el motor. “Test Mode” y “Reset” funcionan de forma similar a como lo hacen en Easy

Mode. Con los datos de posición y todo configurado, pueden cargarse código a modo de prueba (View>Drive Test). Únicamente posee tres comandos:

- [DRIVE]: Se indica a qué posición desplazarse y se ejecutará el desplazamiento.
- [WAIT]: Aplica un tiempo de espera definido en milisegundos
- [JUMP]: El programa retorna a la primera línea para volver a ejecutarse.

Para construir código con estos comandos, la ventana Drive Test dispone de una serie de botones, para añadir, eliminar o desplazar estas líneas, además de cargar o descargar un programa del controlador.

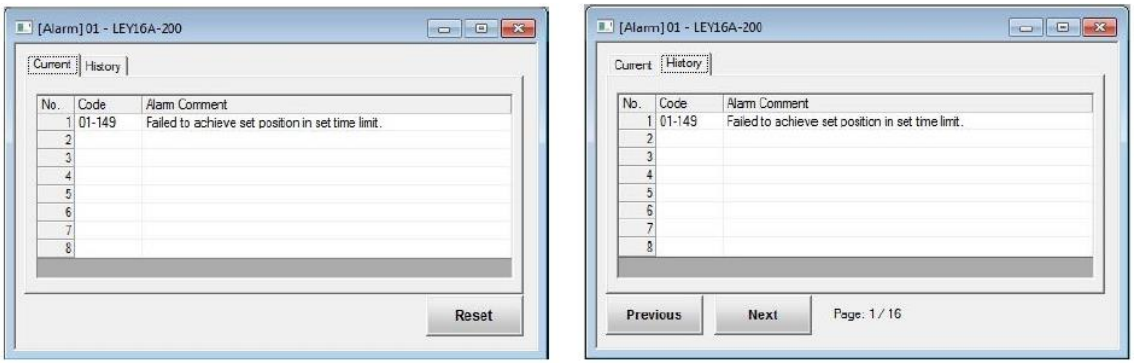


Figura 32: ActController - Histórico de alarmas

2.2.2.2 Configuración actual

En un inicio, al configurar todos los parámetros necesarios para la conexión se llevó a cabo un pequeño tanteo de la aplicación para conocer en profundidad cómo funcionaba el actuador. Tras eso, la idea inicial fue que el manejo del actuador estuviese controlado por un PLC, llegando incluso a hacer pruebas satisfactorias con el mismo.

Sin embargo, todo esto se hizo antes de que se formalizase el proyecto, cuando todavía no se conocía muy bien el objetivo de éste; la única información era que había que analizar las PCB, por lo que, con el objetivo de agilizar las tareas a desarrollar, se comenzó a trabajar con el actuador lineal.

Una vez que se planteó el problema más detalladamente, nos dimos cuenta de que en este estado no era útil tratar de controlar el actuador con un PLC por los siguientes motivos:

- Era necesario llevar a cabo gran cantidad de pruebas debido a que había que determinar los parámetros necesarios para todos los elementos del setup, siendo especialmente crítico el sincronismo entre la cámara y el desplazamiento de las placas.

- Se trata de un estudio de viabilidad, no de un proceso industrial ya implementado, por lo que el análisis de las PCB no es algo continuo y está sujeto a variaciones durante la fase de experimentación.

Así, se decidió trabajar con Easy Mode, ya que sus funcionalidades nos eran suficientes por el momento. Después de varias pruebas con cambios de posición y velocidades, actualmente se está trabajando con la siguiente configuración:

- **Home**: corresponde con la posición destinada al cambio de placa. El desplazamiento hacia ella se produce de forma rápida.

- **Previa**: se trata de una posición a la cual se desplaza la PCB rápidamente justo antes de que se produzca la adquisición de la imagen.

- **Análisis**: cuando se inicia el desplazamiento hacia esta posición se produce el disparo de la cámara, lo cual se hace de forma manual por el usuario. Se produce un avance lento por parte de la PCB, de tal manera que este avance está sincronizado con la frecuencia de adquisición de la cámara lineal.

2.3 Componentes mecánicos

2.3.1 PCB

A raíz de lo comentado anteriormente relacionado con el Proyecto Heibus, cuando una parte del caso de estudio relacionado con la eliminación de las burbujas fue asignado al grupo de expertos de la Universidad de Jaén, se acordó el envío de 14 PCB al grupo de investigación por parte de la empresa.

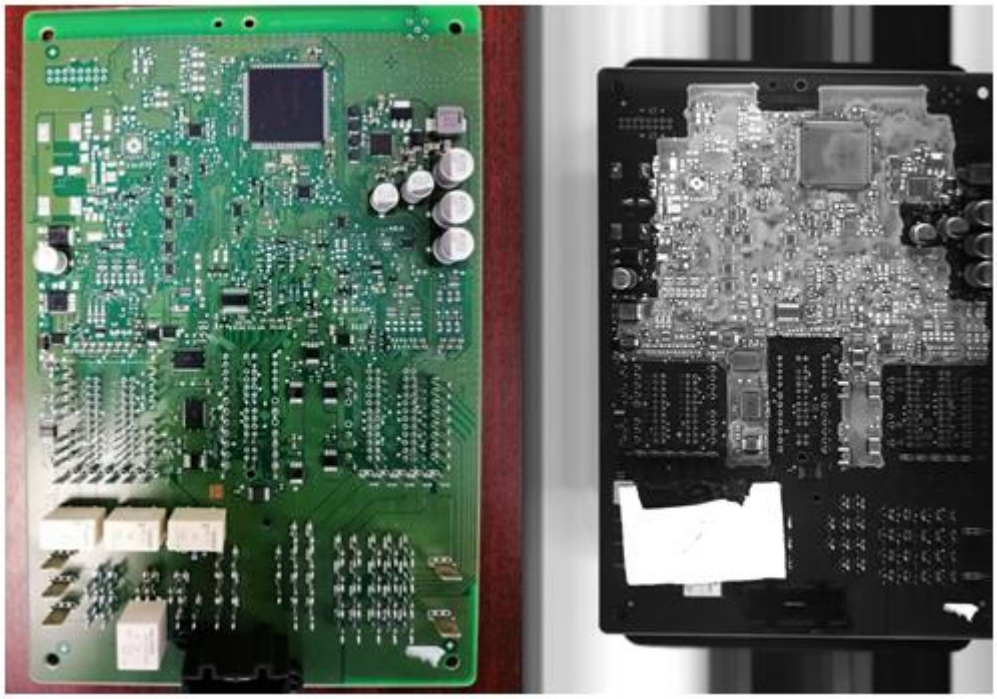


Figura 33: PCB // Imagen PCB Cámara lineal

Se trata de una placa rectangular de dimensiones 223x148mm con cuatro taladros en las esquinas, aunque no toda su superficie está cubierta por el barniz protector. Éste forma un área que cubre la mayoría de los componentes.

En general, los componentes pueden clasificarse en base al tipo de encapsulado del que disponen, siendo de tipo THD (Through hole device) o SMD (Surface Mount Device). Estos últimos no tienen pines que atraviesen la PCB, sino que se sueldan directamente sobre la cara de la placa en la que se encuentran, pudiendo bien disponer de una serie de patillas que sobresalen del componente o bien, de unos extremos metalizados que se sueldan directamente sobre los pads, aquellos puntos de la placa sobre los que se sueldan los componentes.

Figura 34: Componentes PCB

La mayoría de los componentes de la PCB son de tipo SMD, a excepción de una serie de conectores metálicos situados en la parte inferior, sin embargo, estos no se encuentran bajo la capa de barniz, al igual que los condensadores electrolíticos.

Es de vital importancia la detección de burbujas, ya que como se comentó, pueden favorecer la aparición de humedad en el entorno de los pines, debilitando la soldadura y pudiendo ocasionar una desconexión del pin, o provocar un cortocircuito entre pines.

2.3.2 Piezas 3D

Cuando se nos encargó el estudio de viabilidad al grupo de investigación, se decidió usar una cámara lineal por su alta resolución. Pero para ello era necesario dotar de movimiento relativo a la PCB respecto de la cámara, y debido a que ya se disponía del actuador lineal se decidió emplear esa estrategia, acompañada de las barras de iluminación. Sin embargo, no existía forma de colocar la PCB sobre el actuador lineal ni de fijar la iluminación a la estructura.

2.3.2.1 Utillaje

Al inicio del proyecto hubo que esperar unos días a que las PCB que se han analizado llegasen a la Universidad, tiempo que se aprovechó para contactar con el proveedor de los demás productos. Una vez que llegaron al GRAV, la primera tarea fue la de diseñar la manera de que la placa se mantuviese fija sobre el actuador lineal durante el desplazamiento de éste y a la vez fuese fácil de retirar y colocar por el usuario.

Otro a factor a tener en cuenta y del que es importante no olvidarse es que uno de los objetivos de este proyecto es el de realizar un estudio de viabilidad, por lo que debe primar la economía, ya que si la idea no progresase, esto no suponga pérdidas económicas, por lo que toda opción de diseñar y fabricar una pieza de estas dimensiones de forma externa al grupo quedó totalmente descartada.

Esto restringía la fabricación a una única alternativa, la impresión 3D. Una vez que esto quedó determinado, el siguiente paso fue el diseño del utillaje, para lo que hubo recopilar las necesidades que se tenían:

- Facilidad para colocar y retirar la PCB
- Adaptado a la base desplazable del lineal
- Posibilidad de análisis de la PCB por ambas caras

La idea fue diseñar una base con cuatro taladros centrales para unir este utillaje al lineal, con elevadores en las esquinas para conseguir que la PCB pudiera colocarse por ambas caras, lo que finalmente no ha sido necesario.

La clave de este diseño está en el extremo de cada elevador, la cual tiene forma de tronco de cono, esto permite que la punta entre en el taladro de la placa de forma rápida y sencilla, colocándose de forma correcta y fija por su propio peso, de tal forma que no se mueva durante la traslación en el eje del actuador lineal.

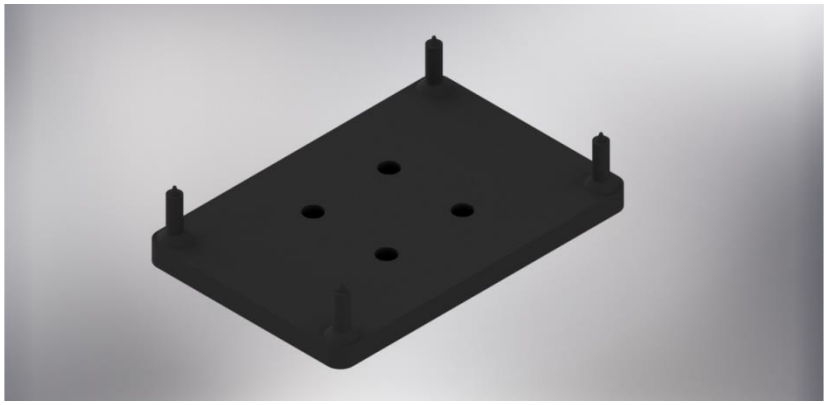


Figura 35: Renderizado utillaje

2.3.2.2 Enganche

De la misma forma, debido a que las barras de iluminación necesitan estar situadas sobre el eje del lineal, se necesitaba diseñar algún sistema de unión que permitiese fijar éstas a un perfil estructural. Por lo que los requisitos de diseño eran:

- Necesidad de unión a perfil estructural
- Adaptado al carril posterior de la barra de iluminación

Con este punto de partida, se decidió diseñar una pieza de forma cuadrada, de tal forma que en el interior tuviese unos salientes con la forma de los carriles exteriores de un perfil de 40x40mm, consiguiendo así que se sujetase en el perfil, y a la vez que pudiera desplazarse por éste para realizar un ajuste de la posición horizontal. En la parte inferior, dispone de un saliente con la forma del carril posterior de la barra de iluminación, funcionando de manera similar a lo descrito anteriormente con el perfil.

También impreso en 3D a modo de prototipo, se fabricaron primero dos para la primera barra de iluminación y otros dos cuando llegó la segunda barra.

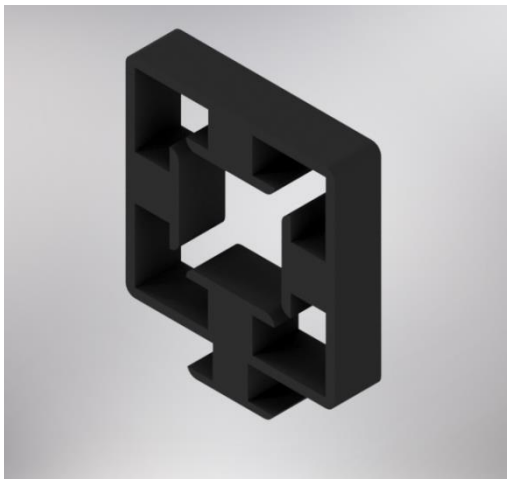


Figura 36: Renderizado enganche

2.3.3 Estructura iluminación

Como ya se comentó, el sistema que emplean actualmente los operarios para inspeccionar las placas consiste en una lámpara móvil, similar a los flexos tradicionales de escritorio, bajo la cual situaban la PCB. A diferencia de esto se pretendía diseñar una estructura para la iluminación con las siguientes características:

- Fija, de tal forma que una vez que se determinasen las condiciones óptimas de iluminación, éstas se mantuviesen invariables. Con ello nos referimos a posición y ángulo de la barra.
- Económico, ya que tampoco debía perderse el concepto de que se trata de un estudio de viabilidad y no de una solución industrial.

Es decir, había que buscar una solución que fuese eficaz y a la vez no supusiese un desembolso económico significativo, incluso reutilizando materiales disponibles como finalmente se produjo.

Una vez establecidas estas condiciones de partida, era necesario determinar cómo situar dicha iluminación, para lo cual se decidió situar la cámara de forma perpendicular sobre el actuador lineal y a un lado de la cámara, la barra de LED orientada para iluminar la PCB. Para ello se diseñó una primera estructura, con solo una barra de iluminación.

Figura 37: Estructura Versión 1

Sin embargo, como se describió, fue necesario añadir una segunda barra de LED, lo que hizo necesario la modificación del diseño de la estructura, ya que la primera versión incluía solamente una barra. Del mismo modo, se utilizó el material existente de la primera estructura para construir la segunda.

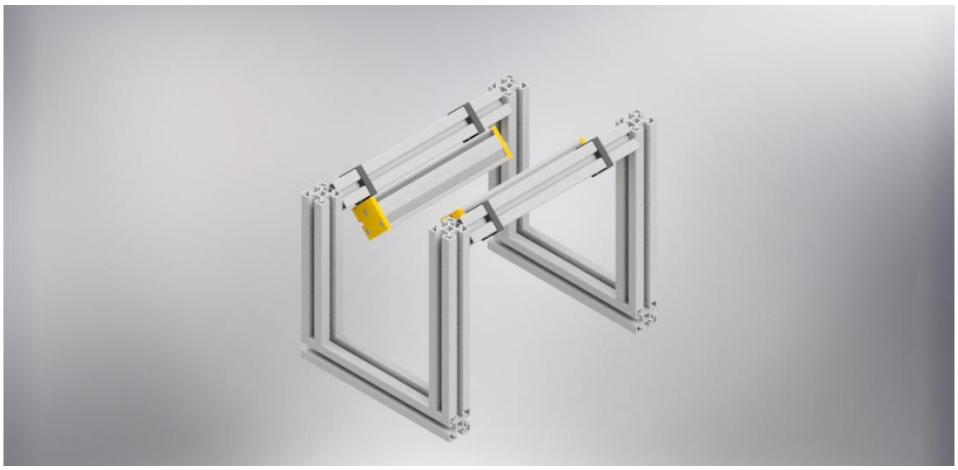


Figura 38: Estructura Versión 2

2.4 Cámara

La totalidad de los elementos anteriores son necesarios para la construcción de una estación prototipo, sin embargo, el dispositivo principal es la cámara. De ésta depende en gran parte el éxito del estudio de viabilidad.

2.4.1 Cámara lineal

Se valoraron las diferentes opciones y tecnologías existentes en el mercado referentes a cámaras industriales, existiendo dos grupos principales en base a las dimensiones de las imágenes captadas.

- **Cámaras matriciales**: se refiere a aquellas cámaras cuyo sensor está formado por una matriz de píxeles, tradicionalmente con una proporción 4:3, siendo de este tipo la mayoría de cámaras estándar e

industriales. El material del sensor es fotosensible, de tal manera que cuando incide luz sobre él, transforma los fotones en carga eléctrica que será acondicionada para formar la imagen.

- **Cámaras lineales**: la principal característica de estas cámaras es que únicamente capturan una línea de píxeles de forma simultánea, teniendo que componerlas para formar una imagen completa. Debido a ello, será necesario un movimiento relativo entre la cámara y el objeto a analizar.

  Su manejo es más complejo que el de una cámara matricial, ya que el sincronismo entre el disparo y la frecuencia de adquisición y el movimiento relativo del objeto es crítico. Sin embargo, ofrecen mejores prestaciones por un precio competitivo dado que permiten obtener imágenes a una alta velocidad y con resoluciones que van desde los 512 a los 16K pixeles.

2.4.2 Dalsa Linea 4K

Como ya se comentó, debido a la alta resolución requerida se optó por una cámara lineal, concretamente el modelo Linea 4K del fabricante Dalsa. Se trata de un modelo monolocolor, es decir, solo captura imágenes en escale de grises.

Figura 39: Cámara Dalsa LINEA

Se trata de una cámara basada en tecnología CMOS, es decir, la conversión de fotones a voltaje se produce en cada pixel de forma individual. Es sensible a un espectro de longitudes de onda que va desde los 400nm hasta los 1000nm, gracias lo cual es capaz de detectar la iluminación UV que hace posible la detección de las burbujas (405nm).

Características generales	
Resolución	4096x1
Tamaño píxel	7.04µm x7.04 µm
Frec. Máxima líneas	26 kHz
Salida	Gigabit Ethernet

En cuanto a las comunicaciones, utiliza un cable con conectores tipo RJ45 en ambos extremos para conectarse con el PC, y un terminal GPIO para alimentar la cámara y controlar sus entradas y salidas, permitiendo entre otros, el disparo de la cámara controlado por la señal de un sensor.

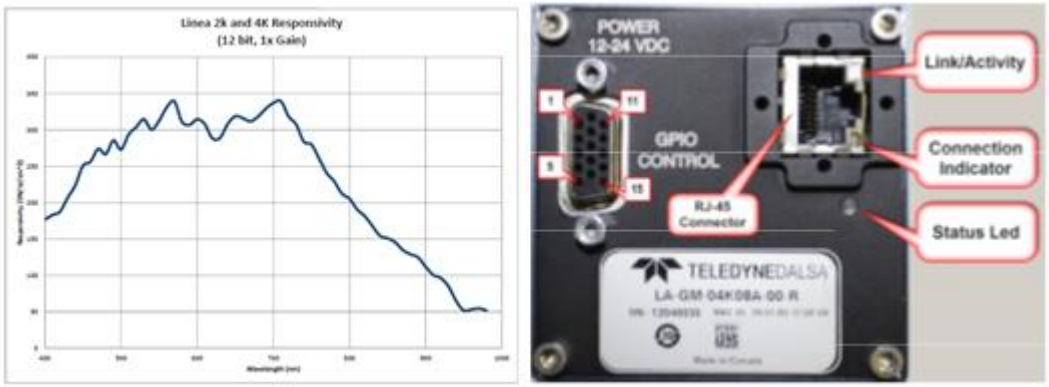


Figura 40: Curva responsividad // Pinout Cámara

2.4.3 Software: Sapera CamExpert

Existen varios parámetros claves a configurar para obtener las imágenes deseadas, lo cual puede realizarse desde el software propio que suministra el fabricante: Sapera CamExpert.

Tras la instalación, es posible establecer la conexión entre la cámara y el PC. Para ello se ejecuta Sapera CamExpert, apareciendo una interfaz como la que se muestra a continuación.

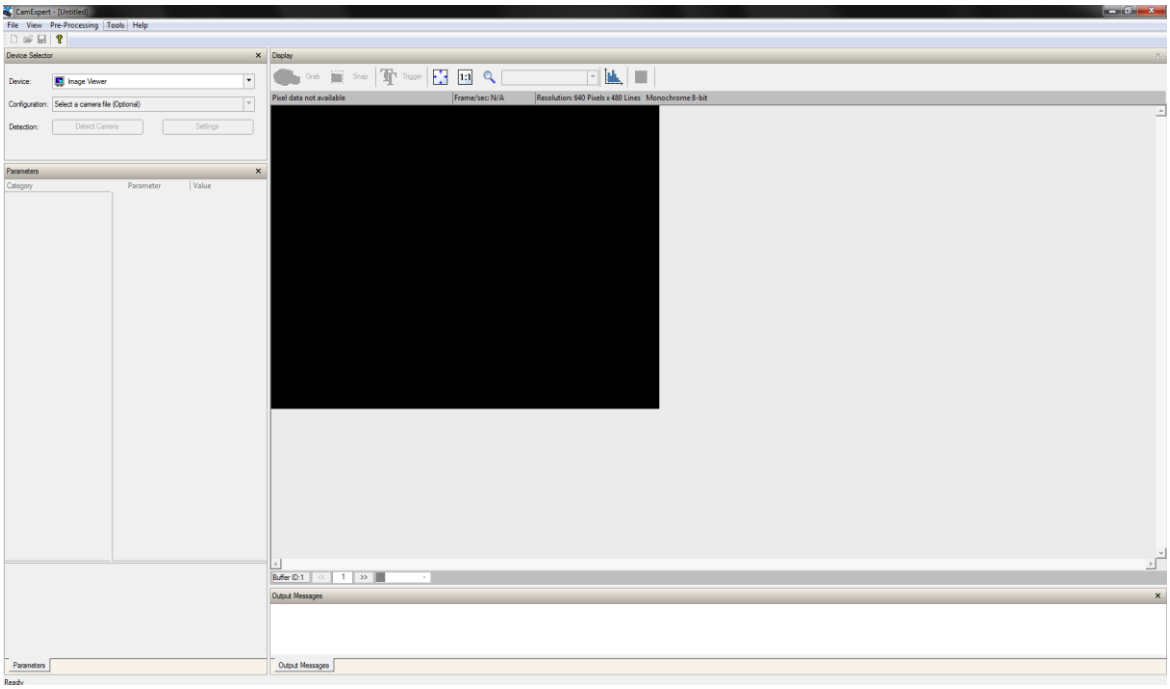


Figura 41: CamExpert - Vista general sin dispositivos

Aparece una vista por defecto en la que se muestra un cuadro negro donde luego se verán las imágenes, sin embargo, no se puede acceder a ninguna otra opción debido a que no se ha detectado la cámara. Cuando se conecta la cámara y es detectada por CamExpert, aparecen dos mensajes: uno en la barra de tareas indicando que un dispositivo se ha conectado o desconectado, y otro en forma de ventana en el propio software, preguntando al usuario si desea trabajar con la cámara detectada.

Figura 42: CamExpert - Barra de tareas

Figura 43: CamExpert - Dispositivo detectado

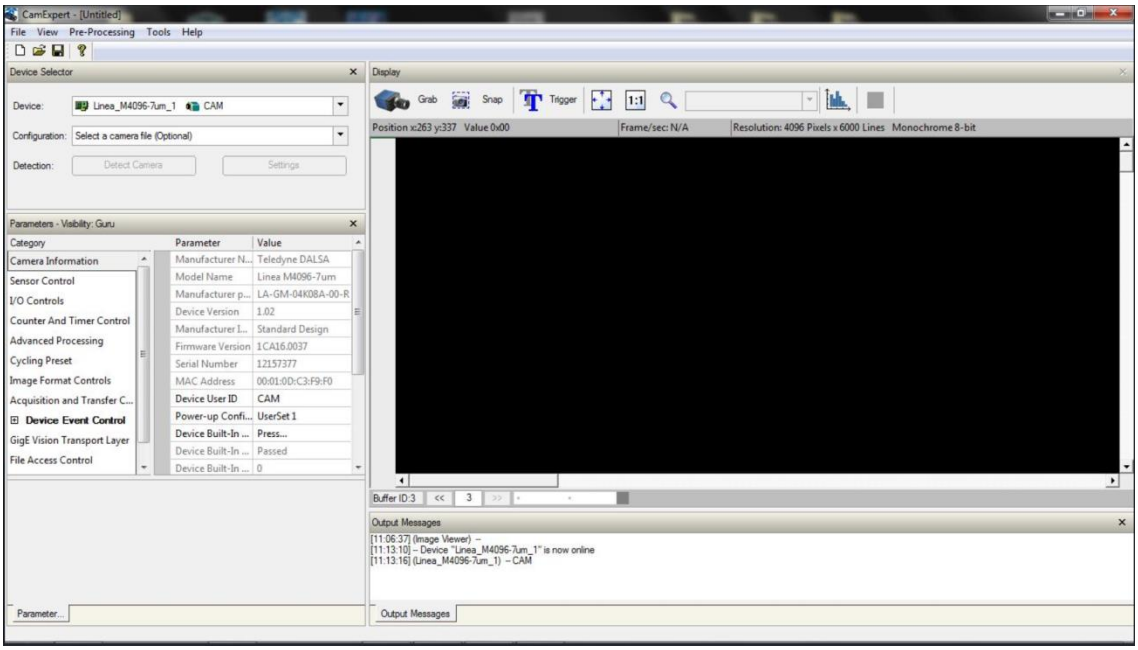


Figura 44: CamExpert - Vista general LINEA

Al aceptar el anterior mensaje, la GUI cambia de apariencia, desbloqueándose ahora todas las opciones que antes permanecían ocultas, tal como se aprecia en la figura anterior. Esta interfaz dispone de varias áreas dedicadas a diferentes funcionalidades:

- **Device Selection Drop Menu**

  En esta ventana únicamente se muestra el dispositivo detectado (Device), en este caso la cámara Linea 4K, con la posibilidad de cargarle una serie de perfiles de configuración (Configuration) ya sean preestablecidos o generados por el usuario.

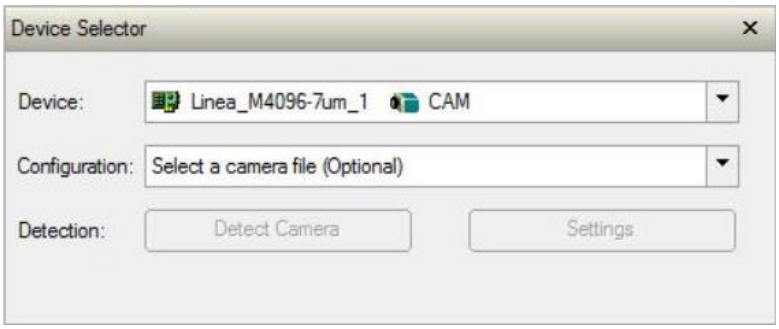


Figura 45: CamExpert – Device Selector

- **CamExpert Control Buttons**

  Aquí se encuentran los botones destinados a la captura y vista de imágenes. Así, de izquierda a derecha aparecen los siguientes botones:

  - **Acquisition Control Button**: Si se hace click una vez comienza a mostrar frames en vivo; si se vuelve a pulsar detiene la adquisición

Figura 46: CamExpert - Grab//Freeze

  - **Single Frame Grab**: Si se hace click sobre él se obtiene un solo frame.

Figura 47: CamExpert - Snap

  - **Software Trigger Button**: Envía una única señal de disparo si éste está activado y además está configurado para que sea vía software.

Figura 48: CamExpert – Trigger

- **CamExpert Display Controls**: Estos botones no afectan a la adquisición, sino que únicamente modifican el aspecto del frame que se está mostrando. De izquierda a derecha, el primero muestra la imagen completa, el segundo, el cuadrado superior de la imagen y el tercero se utiliza para hacer zoom sobre alguna zona concreta

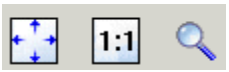


Figura 49: CamExpert – Display Controls

- **Histogram Tool**: Si se selecciona muestra el histograma del frame mostrado.

Figura 50: CamExpert – Histogram Tool

- **Acquisition Information**

  Se muestra únicamente información acerca de la adquisición y los frames adquiridos.

Figura 51: CamExpert – Acquisition Information

- **Parameters Category Selection**

  Permite la configuración de los parámetros del dispositivo conectado, pareciendo únicamente los que son aplicables a éste. Para evitar confusiones, elimina los que no le corresponden, por lo que la vista que aparece puede cambiar de un dispositivo a otro.

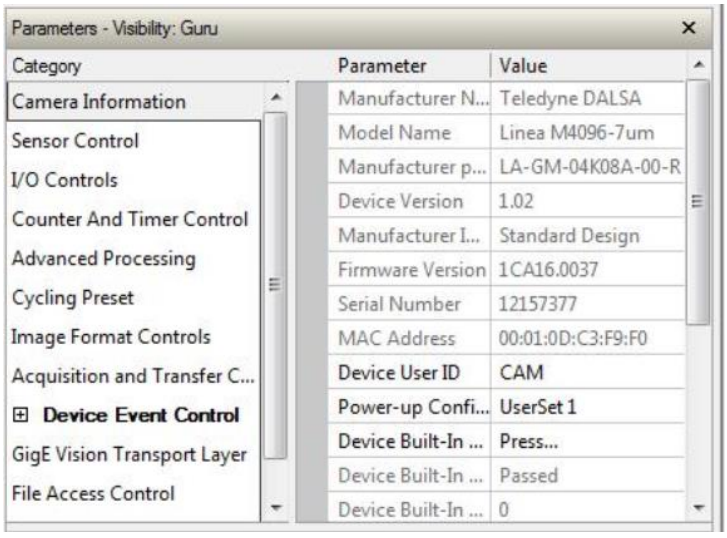


Figura 52: CamExpert – Parameters Category Selection

En la parte izquierda se encuentran las categorías de configuración, destacando entre todas ellas las siguientes:

- **Camera Information**: Muestra la información referente al modelo de cámara, así como el perfil de configuración que se carga al encender la cámara. Además, permite cargar y modificar dichos perfiles, mediante el cuadro de diálogo mostrado en la figura 53.

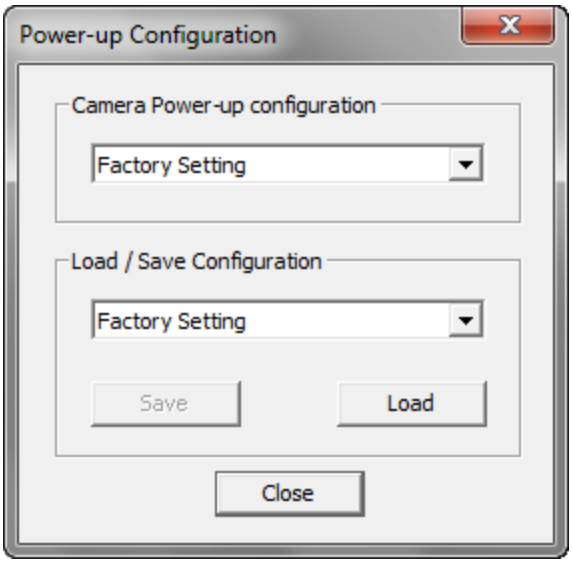


Figura 53: CamExpert – Power-up Configuration

- **Sensor Control:** Permite la configuración de los parámetros relacionados directamente con el sensor, tales como la frecuencia de adquisición de líneas en Hz o el tiempo de exposición en µs.

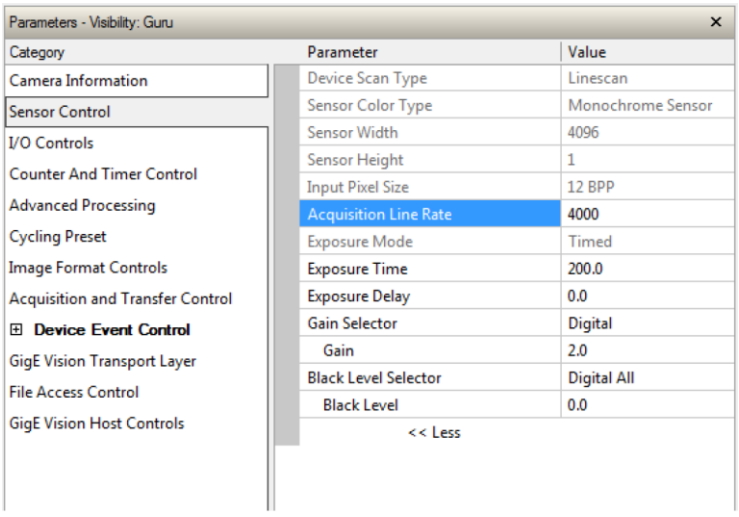


Figura 54: CamExpert - Sensor Control

- **I/O Control:** En esta categoría se pueden configurar los parámetros relativos a las entradas y salidas, ya que es posible configurar el disparo por sensor o sincronizar la frecuencia de adquisición mediante el uso de un encoder.

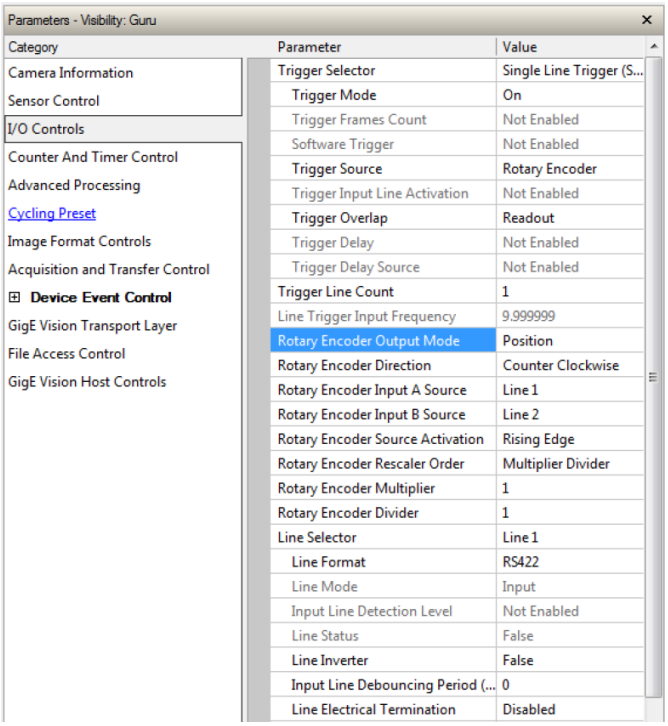


Figura 55: CamExpert - I/O Controls

- **Image Format Controls:** Se puede modificar el tamaño de las imágenes obtenidas, no existiendo límite para la altura; el formato del píxel también puede ser configurado en 8 u 12 bits.

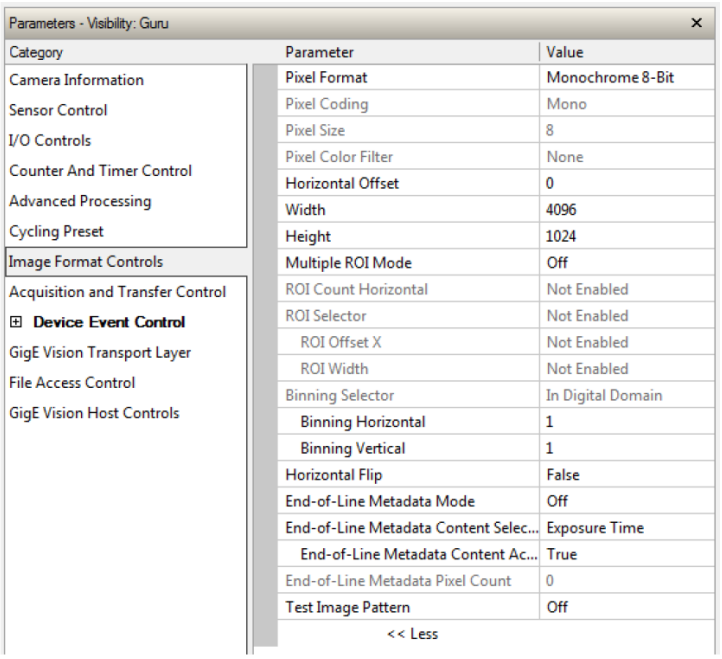


Figura 56: CamExpert - Image Format Controls

Debajo de esta sección aparece un bloque donde se informa al usuario del nombre, propiedades y rango de valores permitidos para una determinada característica.

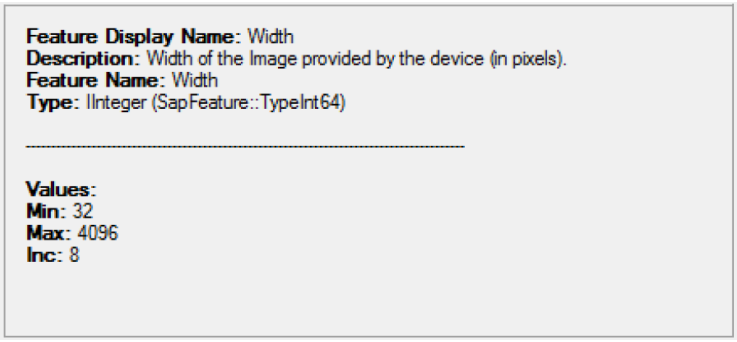


Figura 57: CamExpert - Descripción parámetro

En este ejemplo, indica que la propiedad seleccionada es la anchura y que su valor debe encontrarse entre 32 y 4096 píxeles, con incrementos de 8.

- **Message Window**

  Se trata de una ventana que muestra mensajes de salida a modo de histórico de las acciones realizadas durante la ejecución del software.

Figura 58: CamExpert - Message Window

3. Adquisición de imágenes

Cualquier sistema de visión artificial está compuesto principalmente de dos fases: una en la cual se adquieren las imágenes del objeto o escenario a analizar y otra posterior en la cual estas imágenes se procesan. En este capítulo se describirá detalladamente el proceso seguido en la adquisición de las imágenes relativas a las PCB.

Para la obtención de resultados satisfactorios, es crítico el control de la escena, lo que hará que también el procesado sea más sencillo e intuitivo; factores como la posición de la cámara, ángulo de la iluminación o la selección del objetivo adecuado son de especial importancia.

3.1 Procedimiento de adquisición

Con el objeto de que sirva de introducción y aclare el proceso seguido para la obtención de las imágenes, es necesario explicar de manera detallada el procedimiento seguido para la adquisición de éstas, para lo cual se han seguido los siguientes pasos:

1) Conectar correctamente el cableado necesario al PC: USB del actuador líneas y RJ45 de la cámara, y alimentar tanto la cámara como ambas barras.

2) Ejecutar la aplicación para el control del actuador lineal, ACT Controller, y el software de la cámara, Sapera CamExpert.

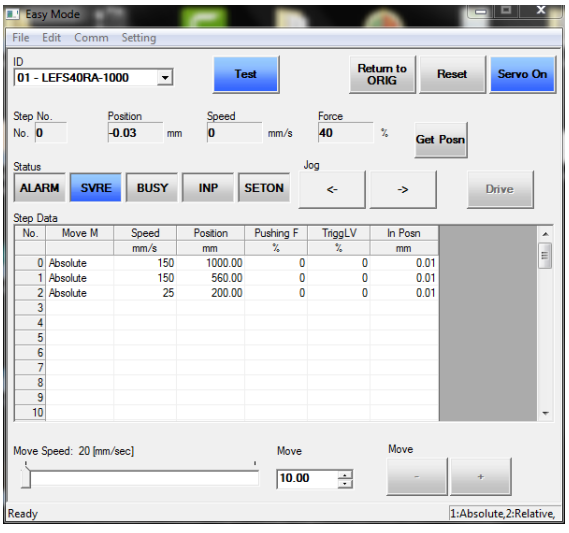


Figura 59: Adquisición – Vista general ActController

3) Desplazar el lineal hasta la posición 1 (1000mm) a 150 mm/s, la opuesta al cabezal donde se ubica el motor, para colocar la PCB a analizar. El valor de velocidad se debe al criterio del autor, ya que no necesita cálculo alguno a no estar sincronizada con ninguna otra.

4) Mover el lineal hasta la posición 2 (560mm) a 150mm/s como paso previo a la adquisición. El criterio para la elección de la velocidad ha sido el mismo que el del apartado anterior.

5) Desplazar la PCB a la posición 3 (200mm) a 25mm/s, pulsando seguidamente el botón Snap del software Sapera CamExpert. Este es un paso crítico dentro del proceso de adquisición, detallándose en el apartado 3.1.1 el cálculo necesario para determinar la velocidad del actuador lineal en esta fase.

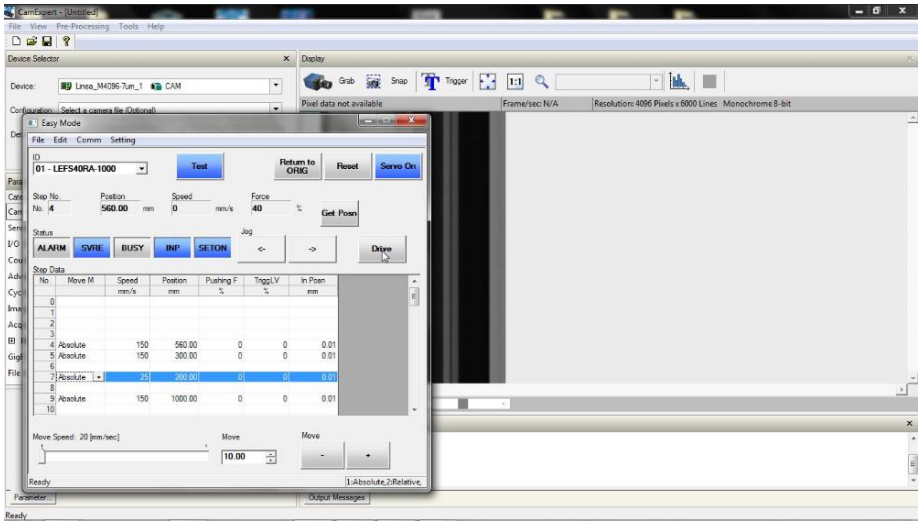


Figura 60: Adquisición - Inicio desplazamiento

La velocidad de avance del lineal está calibrada con la frecuencia de adquisición de la cámara, la cual se encuentra a su vez limitada por el tiempo de exposición, ya que al no disponer de una iluminación muy potente, el tiempo debía tener un valor alto, limitando la frecuencia de adquisición y el avance de la PCB.

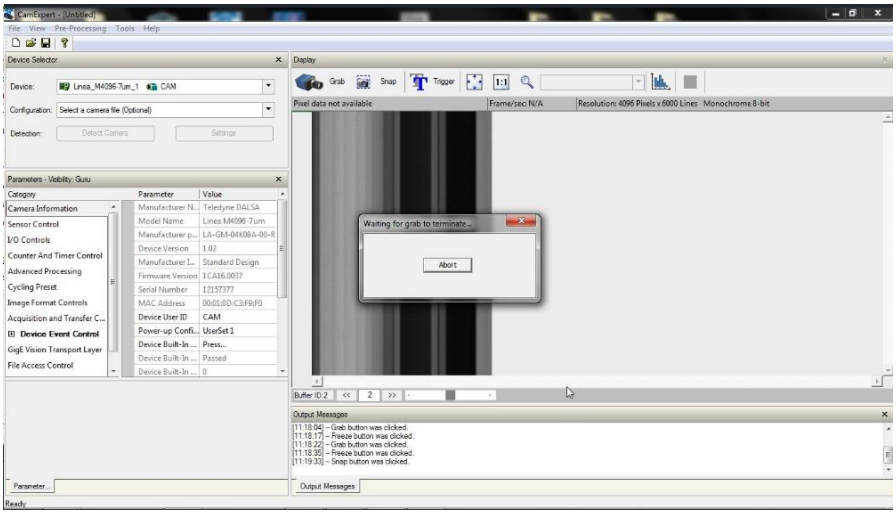


Figura 61: Adquisición - Captura imagen

En cuanto al valor de las posiciones 560mm y 200mm, comentar que se encuentran configuradas de esta manera para asegurar que aunque el usuario presente cierta variabilidad al pulsar el botón de captura, la PCB queda capturada de forma íntegra en la imagen. Ese ligero desajuste que presenta cada una de las 14 muestras es suplido posteriormente en la fase de procesado.

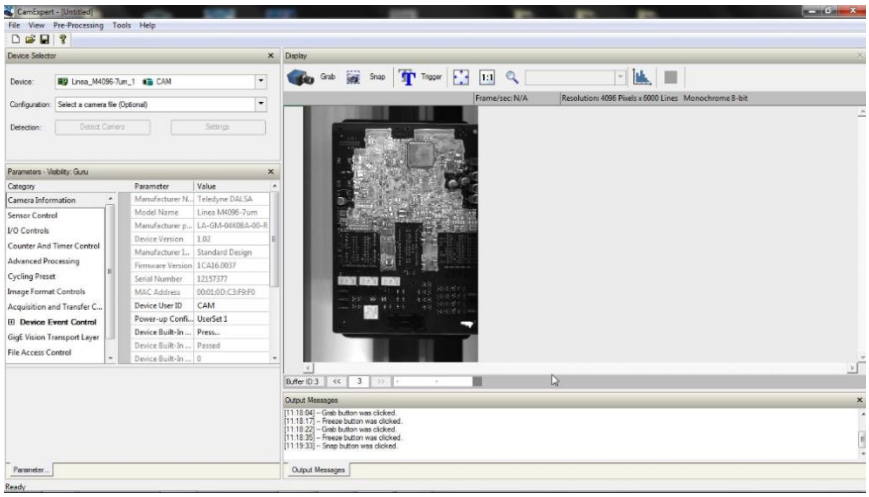


Figura 62: Adquisición - Imagen capturada

3.1.1 Cálculo de la velocidad del actuador lineal

Para justificar el valor asignado a la velocidad de desplazamiento del actuador lineal en el apartado 3.1, durante la captura de imágenes, se ha utilizado el modelo Pin-Hole, el cual establece una relación entre las dimensiones reales del objeto analizado y su proyección en el plano sensor de la cámara.

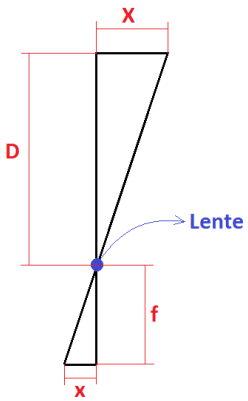


Figura 63: Modelo Pin-Hole

$$\begin{cases} x \rightarrow \text{Proyección en el plano sensor} \\ X \rightarrow \text{Dimensión real del objeto} \\ D \rightarrow \text{Distancia objeto-lente} \\ f \rightarrow \text{Distancia focal} \end{cases}$$

Este modelo relaciona mediante semejanza de triángulos las variables indicadas mediante la expresión indicada en la ecuación 1:

(1)

$$\frac{x}{f} = \frac{X}{D}$$

Para calcular la velocidad a la que debe desplazarse el actuador se realizará una modificación sobre lo indicado; *x'* representará el producto del tamaño de un píxel (líneas) por la cantidad de líneas capturadas por segundo, y *X'* la velocidad de desplazamiento del actuador, nuestra incógnita.

Datos

$$\begin{cases} x = 7.04\ \mu m \\ f = 23\ mm \\ F = 600\ Hz \\ D = 136\ mm \end{cases}$$

(2)

$$\boldsymbol{x'} = x \cdot F = 7.04\ \mu m/línea \cdot 600\ línea/s = \mathbf{4.224\ mm/s}$$

(3)

$$\boldsymbol{X'} = D \cdot \frac{x'}{f} = 136\ mm \cdot \frac{4.224\ mm/s}{23\ mm} = 24.976\ mm/s \rightarrow \mathbf{25\ mm/s}$$

Así, se determina la velocidad a la que necesita desplazarse el actuador lineal para estar correctamente sincronizado con la frecuencia de adquisición de la cámara; deberá desplazarse a 25mm/s durante la toma de la imagen

3.2 Proceso previo

Cuando se recibió el primer envío de material se disponía del siguiente material:

- Cámara lineal
- Dos barras de iluminación (365nm y 405nm)
- Filtro UV (365nm)
- Óptica planar Carl-Zeiss

El primer reto a superar era, por tanto, controlar correctamente la cámara lineal, lo cual no es algo intuitivo ni sencillo si no se está familiarizado con los sistemas de visión. Para obtener por tanto las primeras imágenes, además de establecer la conexión entre el PC y la cámara, era necesario acoplar la óptica a la cámara, cuyas características eran:

- Distancia focal: 85mm
- Distancia mínima de enfoque: 883mm

Se construyó así la primera versión del setup, la cual no disponía del actuador lineal; solamente estaba constituida por la cámara fijada a una mesa de reproducción Kaiser RT1 (estructura especialmente diseñada para crear prototipos de sistemas de adquisición). El objetivo era únicamente el de configurar la apertura del diafragma y familiarizarse con el entorno del software de la cámara.

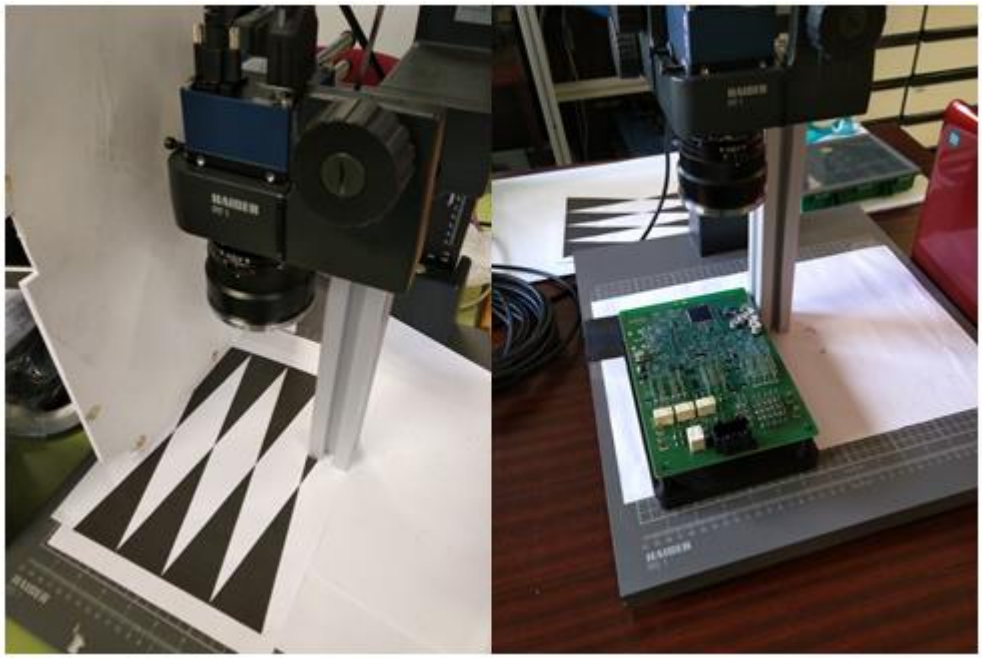


Figura 64: Setup - Pruebas iniciales

Una vez asimilado el funcionamiento de CamExpert, se produjeron tres cambios importantes:

- La introducción del actuador lineal para dotar de movimiento a la PCB
- Elevar lo máximo posible la cámara en la mesa de reproducción, con el objeto de situarse, como mínimo, el equivalente a la distancia mínima de enfoque de la PCB y ser capaces de enfocarla.
- Instalación de la barra de iluminación de 405nm para ser capaces de detectar el barniz.

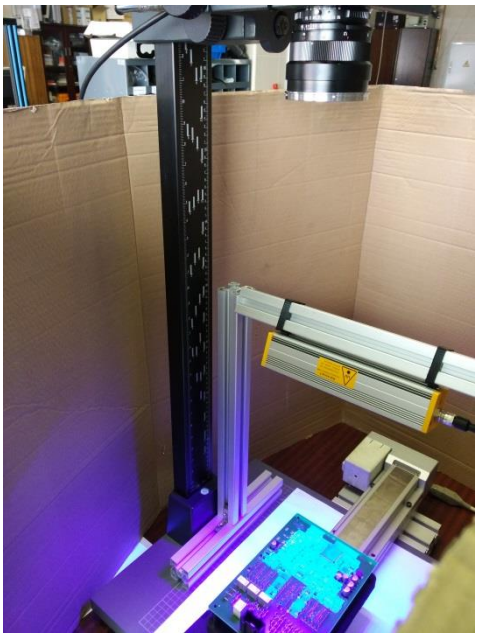


Figura 65: Setup - Prueba PCB 1

Con esta configuración, se comenzaron a tomar las primeras imágenes en las cuales se podía visualizar la PCB. A modo de ejemplo se muestran las siguientes figuras, siendo la figura 66 la primera imagen en la que se comenzaba a apreciar la placa.

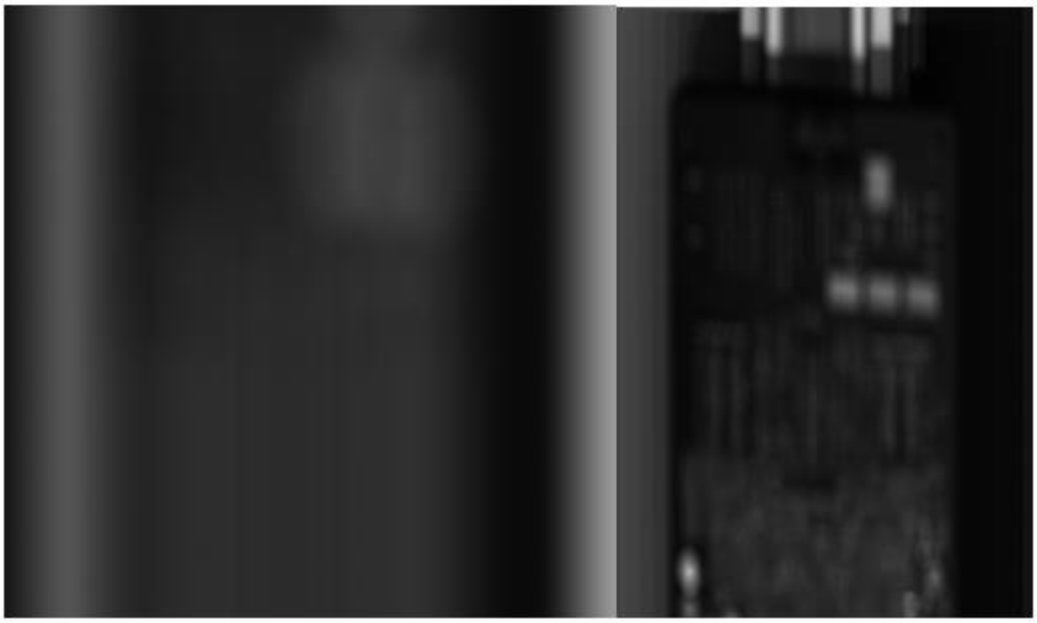


Figura 66: PCB 1 // PCB 2

De cara a una mejora progresiva de las imágenes, fue necesario actuar sobre una serie de parámetros con un doble objetivo: conseguir que la imagen estuviese enfocada, para lo cual fue necesario modificar la distancia de enfoque; y conseguir reproducir de forma exacta las proporciones exactas de la placa, ya que si la velocidad relativa no es la adecuada se producen deformaciones en las imágenes tomadas. Si la frecuencia de adquisición es demasiado baja con

respecto a la velocidad de desplazamiento del lineal, la PCB saldrá acortada; por el contrario, si es demasiado alta, ésta aparecerá alargada.

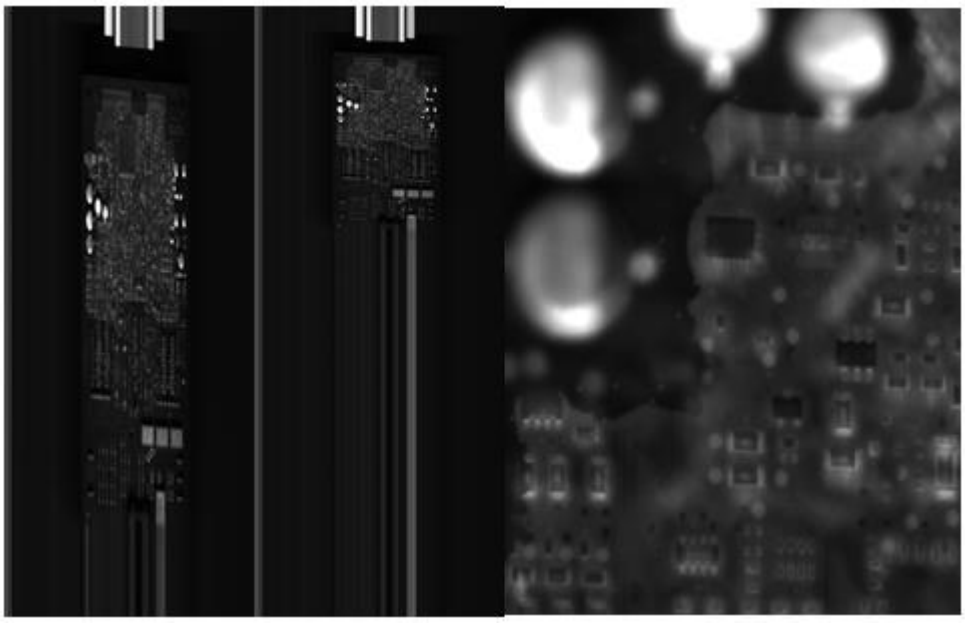


Figura 67: PCB 3 // PCB 4 // Detalle PCB 4

Se había conseguido adquirir imágenes con una proporción correcta, sin embargo, debido a que no era posible conseguir una distancia superior a la mínima de enfoque con la mesa de reproducción, ésta se abandonó temporalmente, pasando a realizarse las pruebas en una estructura existente en el laboratorio, consiguiendo una distancia de enfoque de 1.15m.

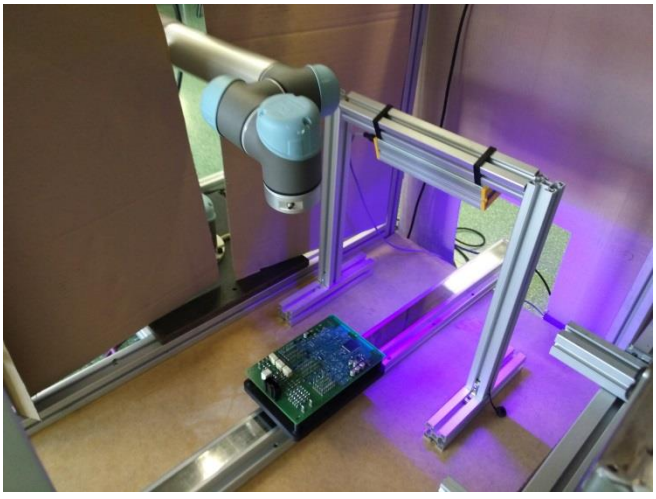


Figura 68: Setup - Prueba PCB 2

Las imágenes mejoraron con respecto a versiones anteriores, sin embargo, la región de interés (ROI), los componentes, se encontraban pixelados debido a la distancia a la que estaba situada la cámara; no se podían obtener mejores resultados con el objetivo del que disponíamos así que se solicitó uno que se adecuase mejor a las condiciones requeridas.

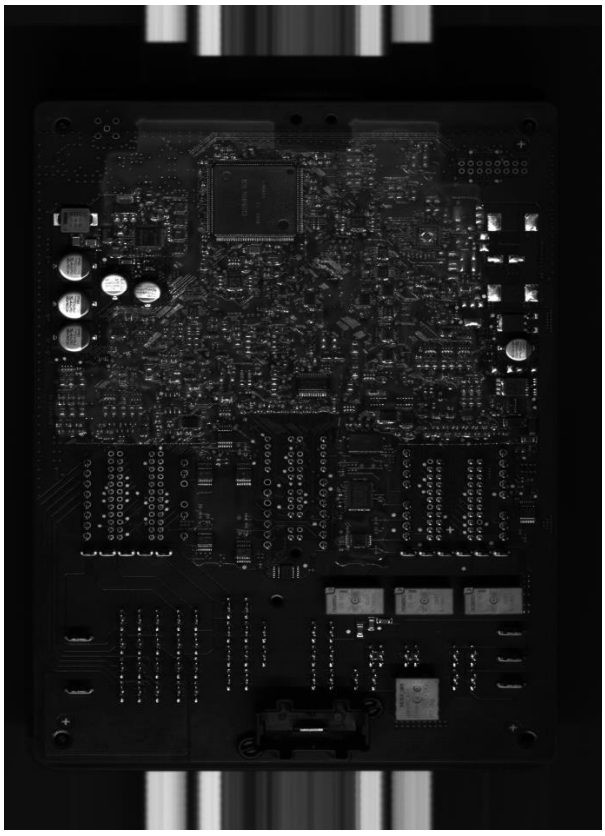


Figura 69: PCB 5

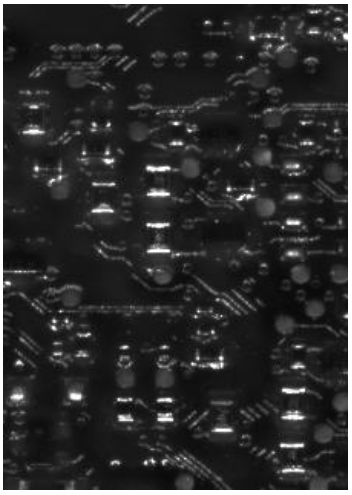


Figura 70: Detalle PCB 5

El nuevo objetivo, con el cual sí seríamos capaces de analizar los componentes en busca de burbujas tenía las siguientes características:

- Distancia focal: 24mm
- Distancia mínima de enfoque: 150mm

Con este material, los ensayos retornaron a la mesa de reproducción, modificándose por última vez la altura a la cual se situaría la cámara con respecto a la PCB, siendo ésta de igual valor a la distancia mínima de enfoque, y

la altura de la iluminación, la cual se redujo a la mitad. En esta nueva situación se obtuvieron imágenes como la mostrada en la figura 71.

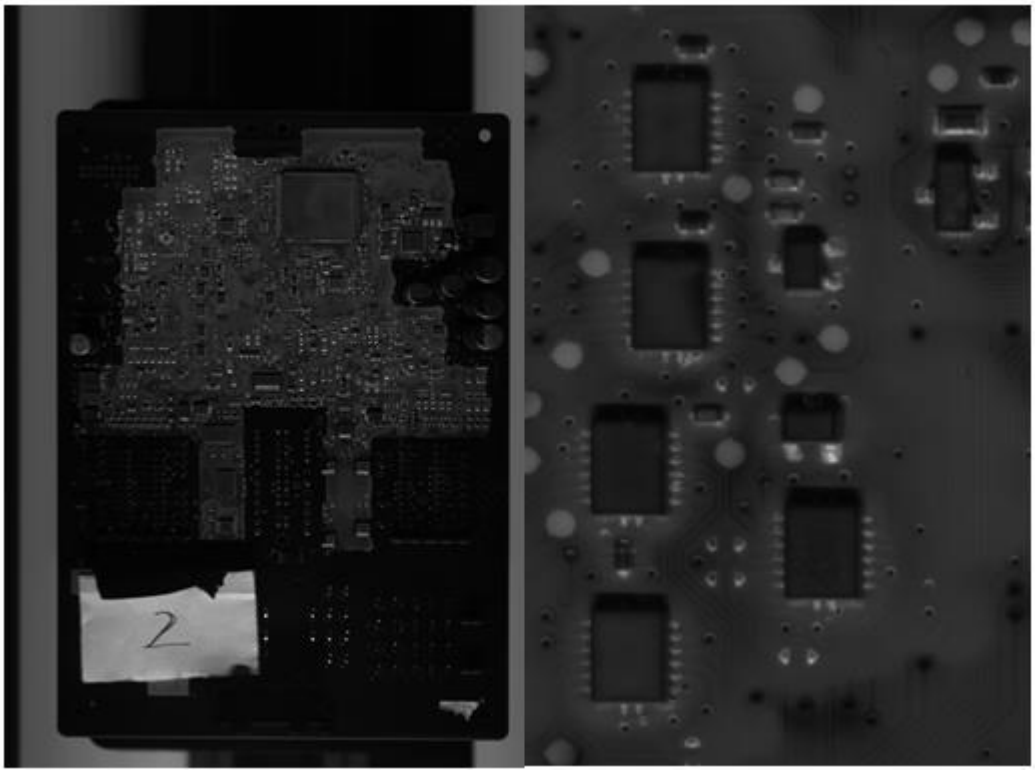


Figura 71: PCB 6 // Detalle PCB 6

Ya se comenzaban a visualizar gran parte de las burbujas, pero era necesario buscar una mayor diferenciación para conseguir un sistema de visión lo más robusto posible. Tras pasar un tiempo sin soluciones, surgieron las siguientes ideas:

- Modificar el ángulo de iluminación de tal manera que formase un ángulo de 45º con la vertical, con el objeto de incidiese la mayor cantidad posible de luz sobre las patillas de los componentes.

- Ajustar el tiempo de exposición de tal manera que la zona de las patillas se sature y las burbujas queden representadas en la imagen como círculos oscuros en medio de una zona clara.

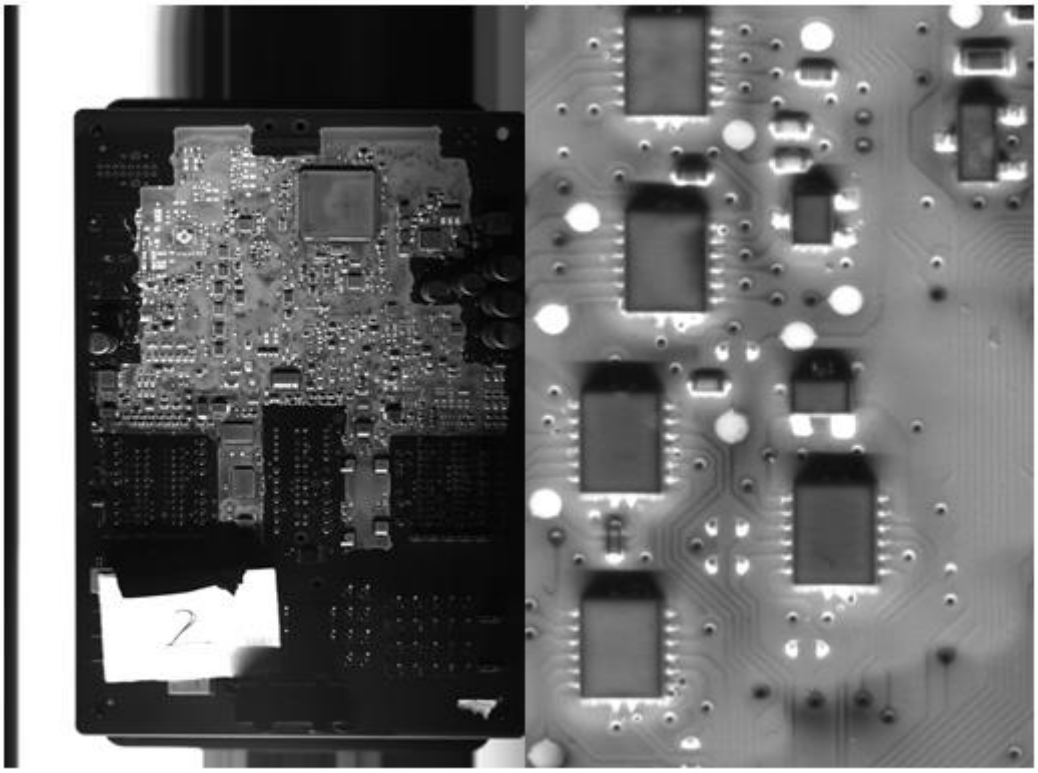


Figura 72: PCB 7 // Detalle PCB 7

Con estos datos, se consiguieron obtener imágenes como la de la figura 72, Aun así, la detección de burbujas solo resultaba posible en las partes iluminadas de los componentes. Al tratarse de objetos tridimensionales, al incidir una iluminación con un determinado ángulo, se producen sombras.

Así, en la parte inferior, especialmente, y en los laterales, en menor medida, el sistema de visión era capaz de detectar burbujas con una alta tasa de acierto, sin embargo, el objetivo era analizar el componente completo en busca de burbujas, por lo que se solicitó una segunda barra de iluminación.

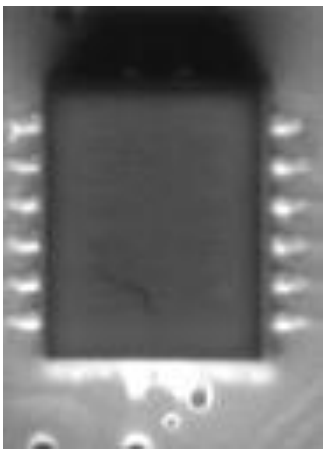


Figura 73: Componente PCB 7

La barra, debía tener las mismas características que la primera, de tal forma que las sombras se eliminasen y se pudiera analizar al completo el componente. Semanas después se recibió, pero no coincidía con el modelo de la primera barra de LED; esta nueva barra tenía aproximadamente la mitad de potencia (20W), lo cual hacía prever que no solucionaría por completo el problema. Aun así, se modificó la primera versión de la estructura para introducir la nueva barra y poder realizar pruebas de adquisición.

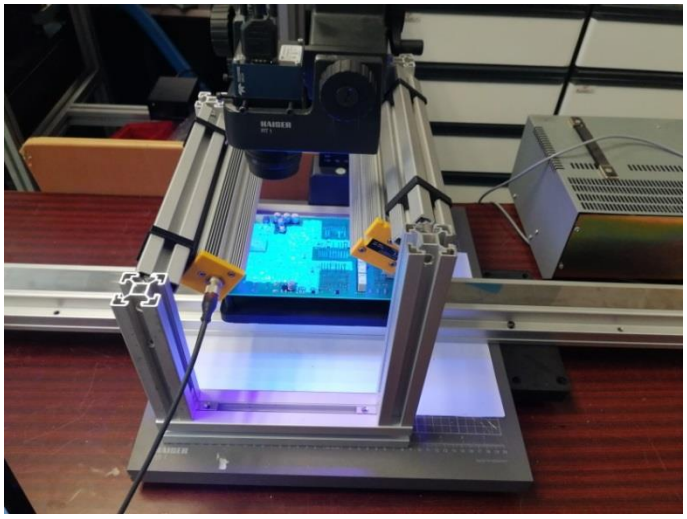


Figura 74: Setup - Prueba PCB 3

Una vez se modificó la estructura se realizaron nuevas pruebas de adquisición, modificando únicamente el tiempo de exposición, el cual tuvo que reducirse al haber introducido mayor cantidad de iluminación. Con todo esto, las imágenes mejoraron y las zonas oscuras redujeron.

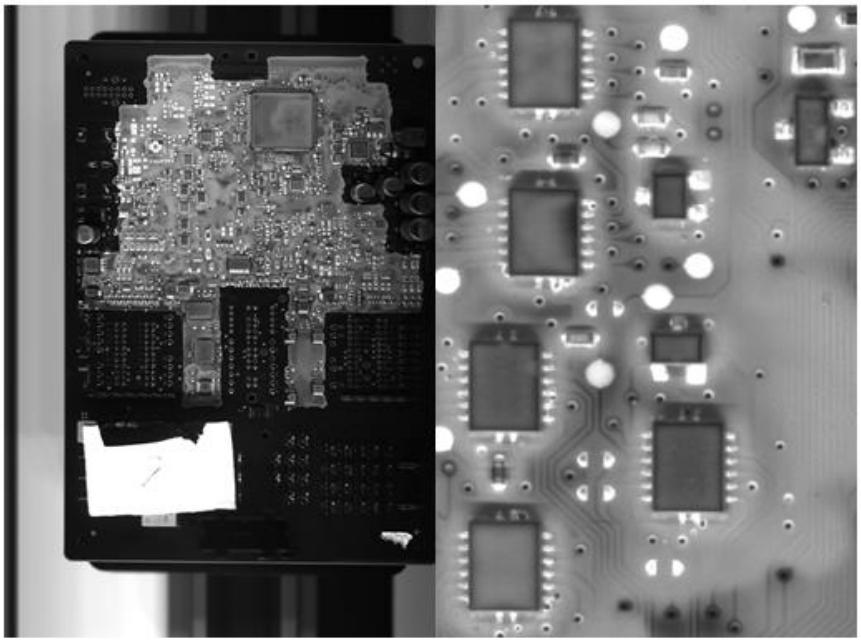


Figura 75: PCB 8 // Detalle PCB 8

Con estas imágenes, se continuaron las pruebas de procesado, llegándose a detectar en ocasiones burbujas ubicadas dentro de la zona oscurecida. A su vez, esto provocaba que menos burbujas de las zonas iluminadas fuesen detectadas; al intentar adecuar el procesado para detectar burbujas en la zona oscura, se produce un desajuste que hace que no se detecten en zonas iluminadas.

Llegada a esta situación, se llegó a pensar que no era posible mejorar las imágenes con el material del que se disponía, valorando la posibilidad de analizar solamente la zona inferior, cuando surgió la idea de igualar la iluminación de ambas barras. Hasta el momento, las dos estaban alimentadas a 24V con la misma fuente de alimentación, por lo que hubo que separar la primera barra (44W) y alimentarla con una fuente de alimentación regulable de tal manera que se la iluminación de ambas barras fuese similar.

Al no disponer de datos tales como una curva de iluminación frente a tensión de alimentación en la hoja de características de las barras Effilux, este proceso tuvo que ser llevado a cabo de manera heurística, realizando un proceso iterativo:

1) Se toma una imagen y se observa la sombra proyectada por el componente.

2) Si existe sombra de la parte superior, se reduce la tensión de alimentación de la barra; si se produce en la parte inferior, se aumenta.

3.3 Estado actual

En relación a lo expuesto en el apartado anterior, con estas condiciones se llegó a un punto en el que resultaba difícil mejorar significativamente la calidad de las imágenes con el material disponible, por lo que se decidió utilizar las imágenes obtenidas con esta configuración para la versión final del proyecto.

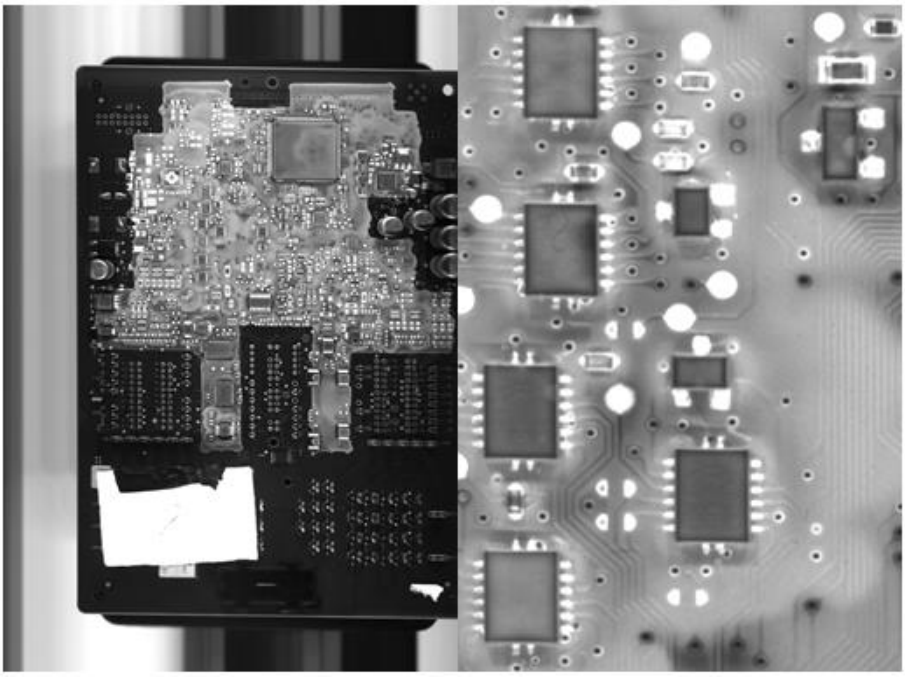


Figura 76: PCB Final // Detalle 1 PCB Final

Tal como se aprecia, las burbujas se encuentran lo suficientemente diferenciadas de las patillas en los cuatro lados de los componentes como para que puedan ser analizados.

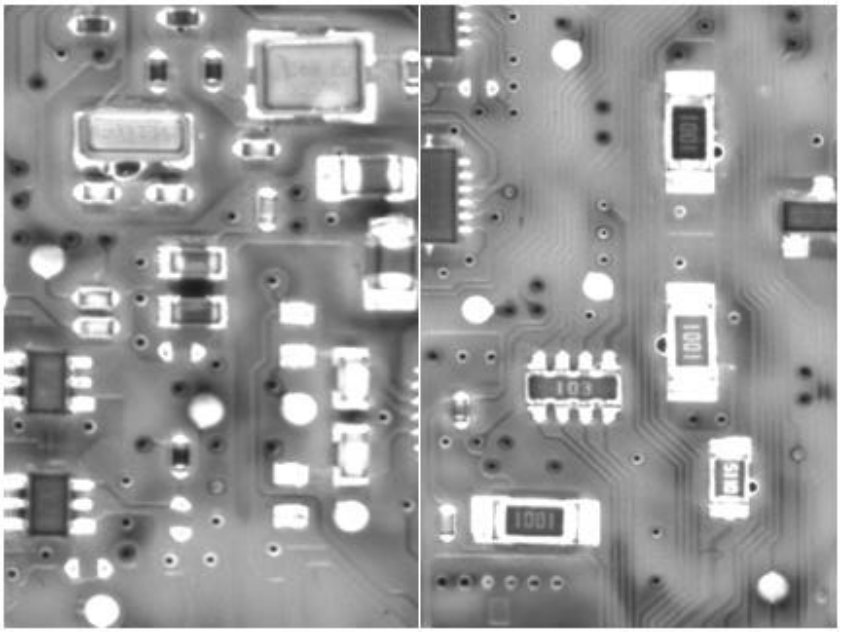


Figura 77: Detalle 2 PCB Final // Detalle 3 PCB Final

A modo de resumen, cabe destacar la configuración y composición de la versión final del setup:

- Actuador lineal SMC
- Estructura iluminación
  - Barra Effilux 405nm 44W (Barra 1)
  - Barra Effilux 405nm 20W (Barra 2)
- Cámara Lineal Dalsa Linea 4K
- Mesa de reproducción Kaiser RT1
- Fuentes de alimentación
  - 24V/2.5A: Cámara y Barra 2
  - 0-30V/3A: Actuador lineal (24V)
  - 0-30V/3A: Barra 1 (17 V)

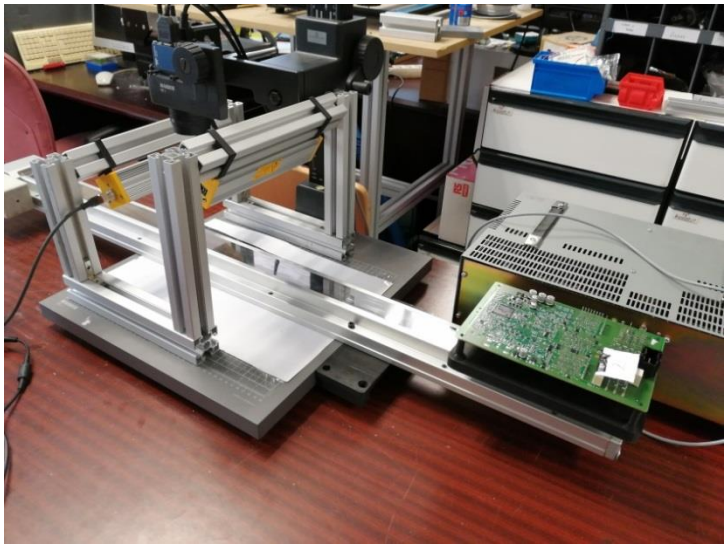


Figura 78: Setup Final 1

Figura 79: Setup Final 2

4. Software para procesado

En la introducción del capítulo anterior se explicó que un sistema de visión tenía dos fases principales diferenciadas: la adquisición de las imágenes y el procesado de las mismas. Así bien, este capítulo está dedicado a esta segunda parte, la cual consiste en el tratamiento de las imágenes y un análisis posterior para la búsqueda de determinadas características u objetos, mediante la utilización de software.

Se pretende, por tanto, determinar la presencia o no de burbujas en las patillas o alrededores de componentes. Sin embargo, el procesado de imágenes no es un proceso directo e indivisible, sino que está compuesto por una serie de etapas que se suceden progresivamente.

Figura 80: Fases Procesado Imágenes

Este proceso comienza con la imagen a procesar ya tomada, siendo a menudo necesario una fase de preprocesado, es decir, aplicar una serie de operaciones a la imagen con el objeto de adecuarla al posterior procesado, por ejemplo, reduciendo su nivel de ruido.

Tras esta fase, pueden detectarse los bordes de la imagen, con la finalidad de poder separar la imagen en objetos diferenciados en base a la imagen de bordes. Sin embargo, esto no es totalmente necesario ya que la separación de esos objetos, llamada segmentación, puede realizarse por otros métodos como ocurre en este proyecto.

Una vez se tienen esos objetos separados y totalmente diferenciados, se procede a la extracción de características de los mismos, tales como área, perímetro, etc. Estas características se utilizarán a continuación para reconocer dichos objetos.

A modo de ejemplo, si el caso de estudio es el reconocimiento de figuras geométricas simples, el sistema tendrá que determinar si se trata de un círculo, cuadrado o triángulo en base a las características obtenidas de los objetos de la imagen.

Una vez finalizado este proceso, se interpretan los resultados y si estos no resultan satisfactorios, se vuelve al inicio buscando posibles puntos de mejora. El objetivo es obtener un sistema de visión robusto que ofrezca el mayor porcentaje de detección posible y no presente falsos positivos.

4.1 MATLAB

En relación a lo anterior, es necesario diseñar el software para realizar la fase de procesado, pero para ello, se necesita una herramienta que lo haga posible. En concreto, para este proyecto se ha utilizado MATLAB, un entorno de desarrollo con su propio lenguaje de programación.

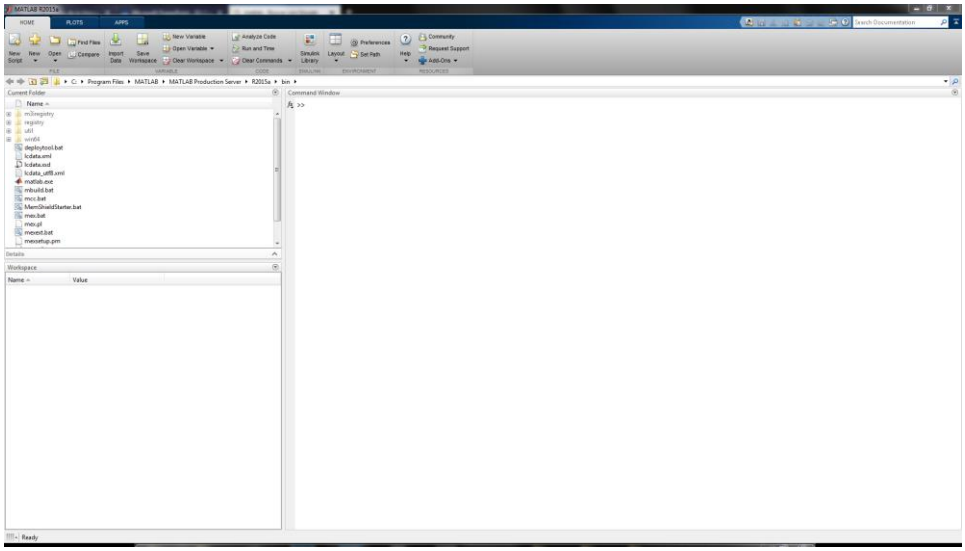


Figura 81: MATLAB - Vista general

En la figura anterior se muestra la vista que tiene MATLAB cuando se inicia, donde pueden apreciarse las diferentes áreas. Comenzando por la parte superior, se encuentra el menú principal, el cual cuenta a su vez con tres pestañas:

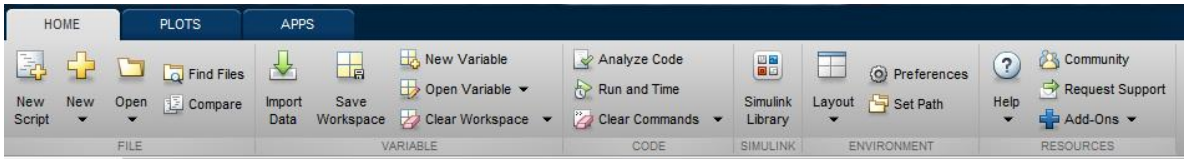


Figura 82: MATLAB - HOME

- **HOME**: Se puede crear un nuevo programa o script, función o clase, así como abrir cualquier archivo compatible. Además, se dispone de un bloque de opciones para la gestión de las diferentes variables, pudiendo crear nuevas, borrar, importar o exportar.

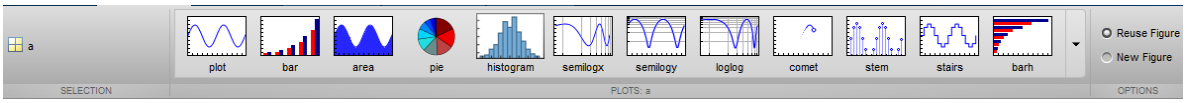


Figura 83: MATLAB - PLOTS

- **PLOTS**: Seleccionando una determinada variable, permite graficarla de diferentes maneras, ya sea con un gráfico de barras, circular, etc.

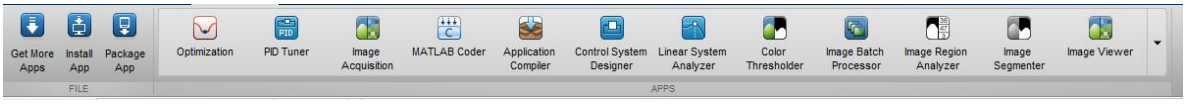


Figura 84: MATLAB - APPS

- **APPS**: En esta sección se permite la gestión de las diferentes extensiones de las que dispone MATLAB, llamadas Toolbox (Caja de herramientas). Se puede, por tanto, además de ejecutarlas, descargar e instalar otras nuevas que le sean de utilidad al usuario.

  Existen toolboxes diseñadas para diferentes necesidades: sintonizado de un controlador PID, procesamiento de imagen, diseño y entrenamiento de redes neuronales, etc. Facilitan la tarea al usuario y, en ocasiones, le sirven de introducción y aprendizaje.

Por otro lado, a la izquierda de la pantalla inicial, se encuentra un navegador para desplazarse entre las diferentes carpetas. Esto es necesario debido a la forma de funcionar de MATLAB; necesita una carpeta base sobre la que trabajar, de tal manera que si se ejecuta un determinado script, la carpeta activa debe ser donde se encuentre alojado este archivo.

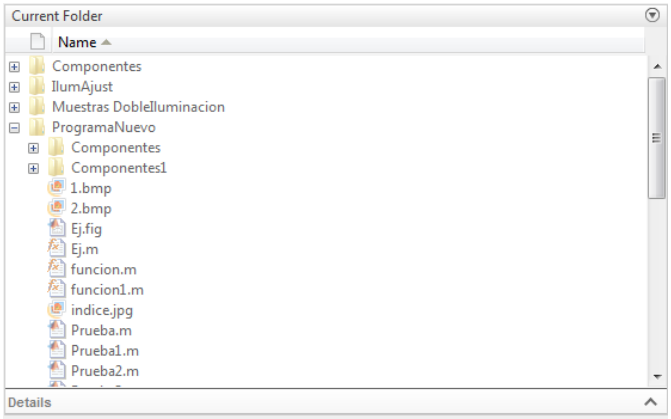


Figura 85: MATLAB - Current Folder

Si al ejecutarlo, la carpeta activa no coincide con la del script, aparece el siguiente mensaje en pantalla, de tal forma que permite modificar la carpeta activa y activar la del script.

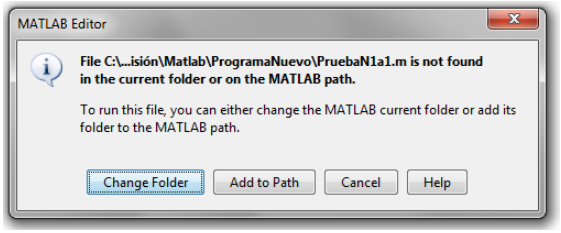


Figura 86: MATLAB – Change folder

En la parte inferior izquierda se encuentra el Workspace o espacio de trabajo, donde se muestran todas las variables junto con sus características. Si no se ha ejecutado ningún script desde que se inició MATLAB aparecerá en blanco. A modo de ejemplo, en la siguiente figura se muestra cómo quedaría el Workspace tras la ejecución de un script de prueba, apreciándose el estado de todas las variables tras la ejecución, así como de qué tipo es cada una.

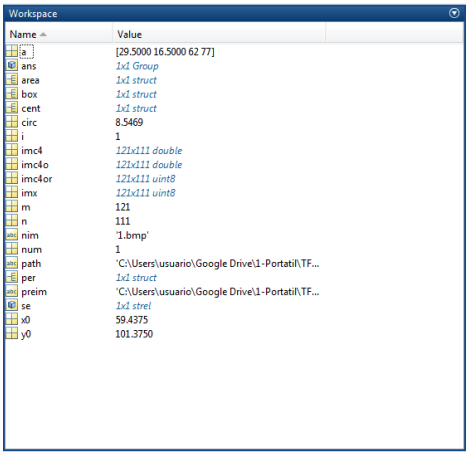
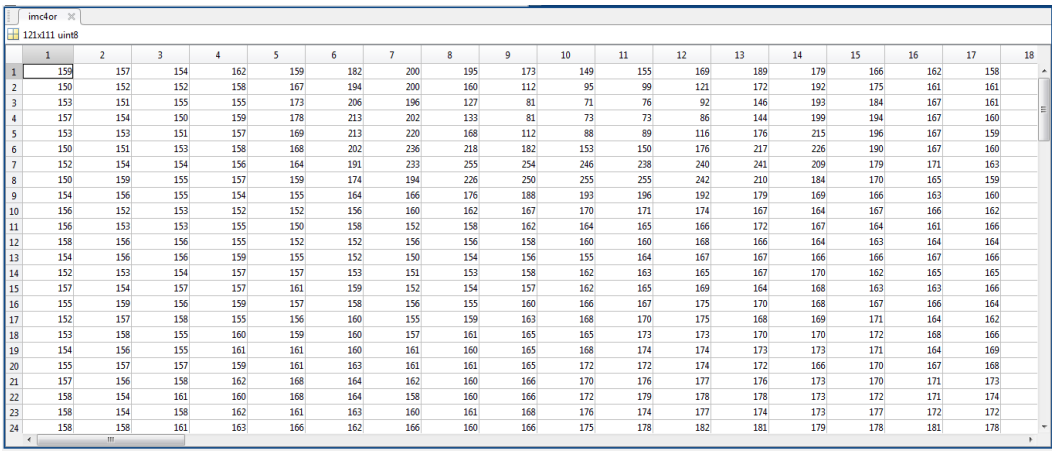


Figura 87: MATLAB - Workspace

Si selecciona una de ellas, se despliega una nueva ventana en la cual puede observarse el valor. En el caso de una imagen en escala de grises, se muestra en forma de matriz, siendo cada valor, el nivel de gris del píxel correspondiente.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	159	157	154	162	159	182	200	195	173	149	155	169	189	179	166	162	158	
2	150	152	152	158	167	194	200	160	112	95	99	121	172	192	175	161	161	
3	153	151	155	155	173	206	196	127	81	71	76	92	146	193	184	167	161	
4	157	154	150	159	178	213	202	133	81	73	73	86	144	199	194	167	160	
5	153	153	151	157	169	213	220	168	112	88	89	116	176	215	196	167	159	
6	150	151	153	158	168	202	236	218	182	153	150	176	217	226	190	167	160	
7	152	154	154	156	164	191	233	255	254	246	238	240	241	209	179	171	163	
8	150	159	155	157	159	174	194	226	250	255	255	242	210	184	170	165	159	
9	154	156	155	154	155	164	166	176	188	193	196	192	179	169	166	163	160	
10	156	152	153	152	152	156	160	162	167	170	171	174	167	164	167	166	162	
11	156	153	153	155	150	158	152	158	162	164	165	166	172	167	164	161	166	
12	158	156	156	155	152	152	156	156	158	160	160	168	166	164	163	164	164	
13	154	156	156	159	155	152	150	154	156	155	164	167	167	166	166	167	166	
14	152	153	154	157	157	153	151	153	158	162	163	165	167	170	162	165	165	
15	157	154	157	157	161	159	152	154	157	162	165	169	164	168	163	163	166	
16	155	159	156	159	157	158	156	155	160	166	167	175	170	168	167	166	164	
17	152	157	158	155	156	160	155	159	163	168	170	175	168	169	171	164	162	
18	153	158	155	160	159	160	157	161	165	165	173	173	170	170	172	168	166	
19	154	156	155	161	161	160	161	160	165	168	174	174	173	173	171	164	169	
20	155	157	157	159	161	163	161	161	165	172	172	174	172	166	170	167	168	
21	157	156	158	162	168	164	162	160	166	170	176	177	176	173	170	171	173	
22	158	154	161	160	168	164	158	160	166	172	179	178	178	173	172	171	174	
23	158	154	158	162	161	163	160	161	168	176	174	177	174	173	177	172	172	
24	158	158	161	163	166	162	166	160	166	175	178	182	181	179	178	181	178	

Figura 88: MATLAB - Matriz imagen

Existe también la Command Window o ventana de comandos, donde se puede tanto mostrar como introducir texto. Los errores y advertencias ocurridos durante la ejecución de un script se muestran aquí, así como también pueden introducirse funciones que realicen una determinada tarea con las variables existentes, añadir o modificar variables, etc.

Figura 89: MATLAB - Command Window

Por último, cabe destacar que cuando se abre un script, se despliega un editor de texto que permite modificarlo. Aquí es donde se lleva a cabo el desarrollo del código, ya que dispone de numerosas funcionalidades: detector de errores en el código, sugerencias de funciones, información sobre los parámetros de entrada y salida de las mismas, etc. Además, en el menú principal aparecen tres nuevas pestañas, destacando aquella que está dedicada a la edición y depuración del código.

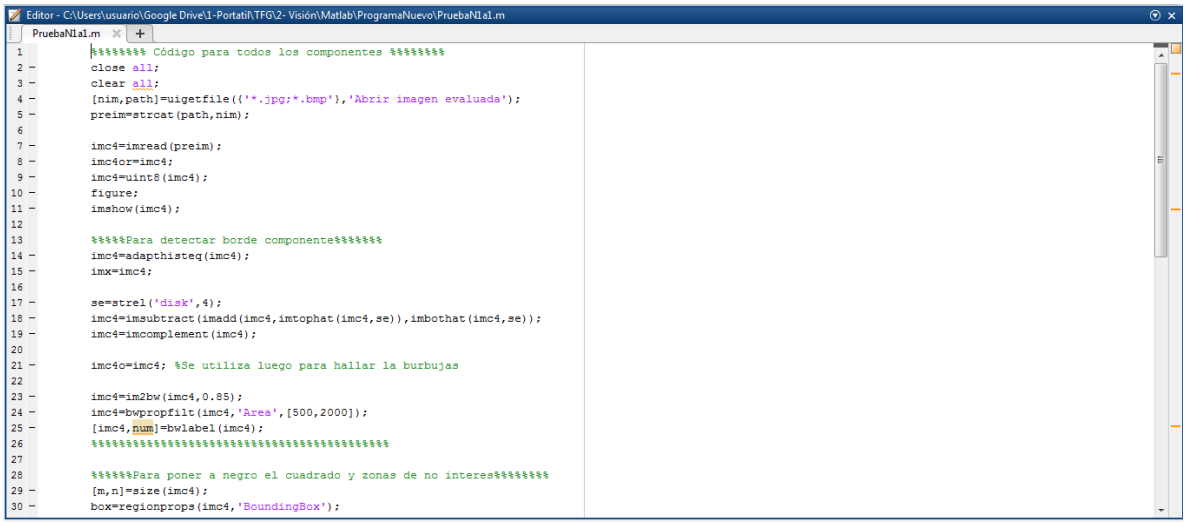


Figura 90: MATLAB - Editor

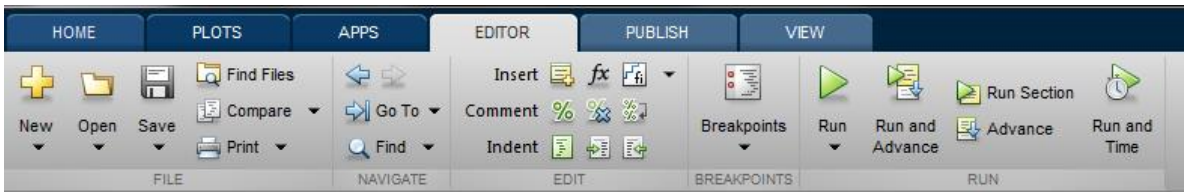


Figura 91: MATLAB - Editor Menu

4.2 Software desarrollado

Este entorno de programación, MATLAB, ha servido de base para el diseño y desarrollo del software que ha hecho posible la detección de burbujas en el barniz de las PCB. Ha constituido un proceso iterativo, ya que han sido varias las ocasiones en las que llegado a un resultado final, no ha resultado satisfactorio debido a los problemas surgidos que se comentaron en el apartado dedicado a la adquisición de imágenes.

Aun así, tras solventar la mayor parte de los problemas se ha logrado obtener resultados significativos que muestran la viabilidad del prototipo.

4.2.1 Interfaz gráfica de usuario

El funcionamiento del software de procesamiento está basado en el desarrollo de una GUI, es decir, una interfaz gráfica de usuario, de tal forma que pueda ser usada fácilmente tanto como por el desarrollador del código como por un usuario no experto.

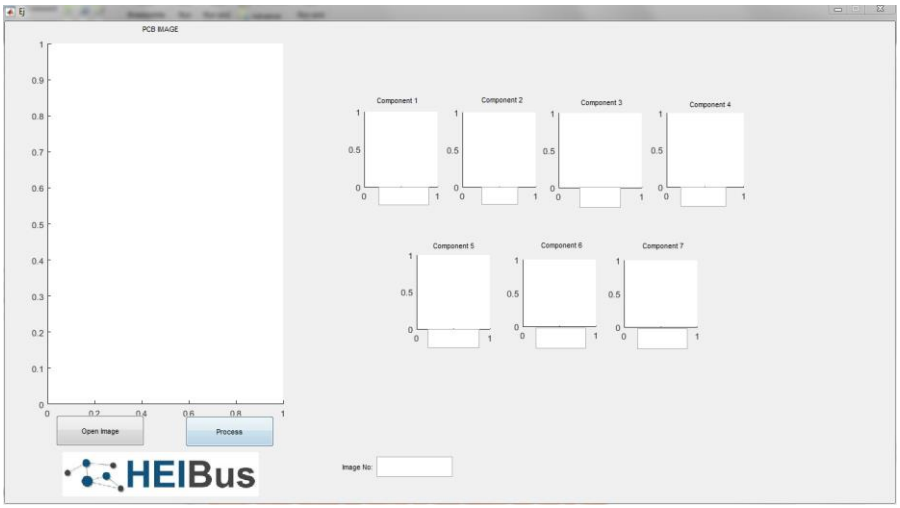


Figura 92: GUI - Vista inicial

Tras ejecutar el programa principal, se abre la ventana mostrada en la figura 92, la interfaz gráfica en torno a la que se vertebra todo el software de procesado. Se pueden diferenciar los elementos según las siguientes categorías:

- **Botones**: Se pueden observar dos botones en la parte inferior de la interfaz que corresponden con las dos fases diferenciadas del procesado: abrir la imagen a analizar (Open Image) y procesar una muestra de los componentes (Process). Al pulsar sobre cada uno de ellos se ejecuta la fase respectiva.

- **Figure:** Uno de los elementos que tiene MATLAB para elaborar interfaces son los Figure, cuadros que pueden ser empleados para mostrar gráficos, imágenes, etc. En esta GUI, existen dos grupos de Figure: a la izquierda, aparece un cuadro con el título “PCB Image” donde aparecerá la imagen completa de la PCB bajo inspección cuando ésta sea seleccionada. En la parte central, se encuentran los siete Figure destinados a cada uno de los componentes analizados.

- **Label:** Las etiquetas, o Label, se utilizan tanto para la entrada como para la salida de texto; en esta interfaz solo muestran texto. En la parte inferior aparece una etiqueta con el título “Image No:” donde aparece el número de PCB analizada cuando es seleccionada, mientras que debajo de los Figure correspondientes a los componentes, aparece “OK” y “KO” al finalizar el análisis indicando la presencia o no de burbujas.

Figura 93: GUI - Vista postprocesado

4.2.2 Etapa 1: Selección de imagen

Para iniciar la etapa de procesado, se parte de que las imágenes ya se han adquirido previamente, por lo que se cuenta con muestras de cada una que han de ser analizadas. De esta forma, el primer paso será seleccionar la imagen a procesar.

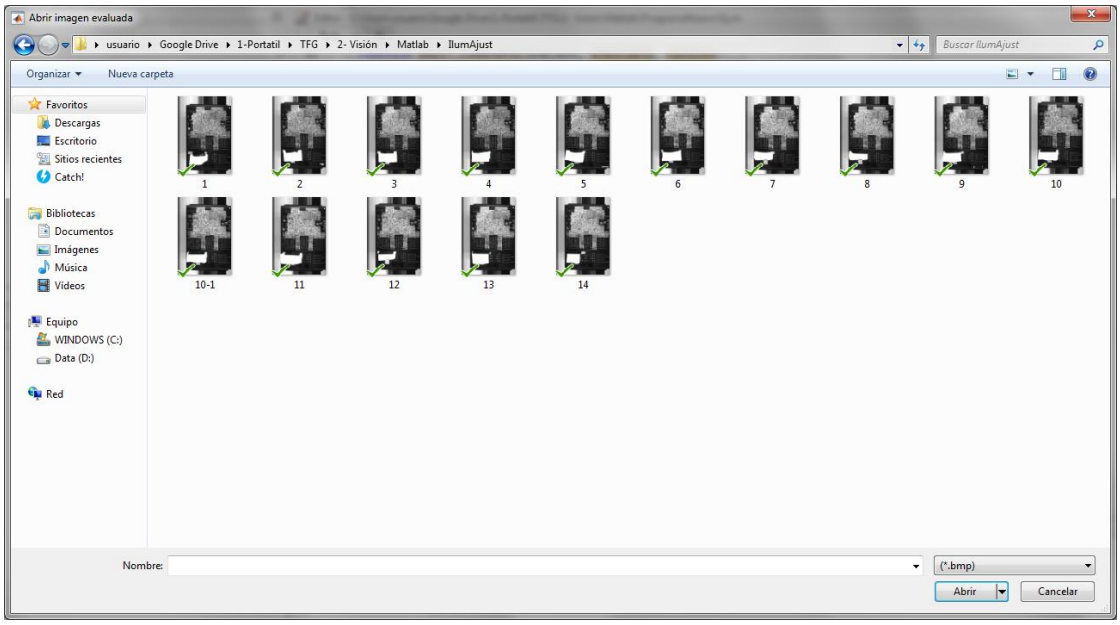


Figura 94: GUI - Abrir imagen evaluada

Al pulsar sobre el botón “Open Image” se despliega una ventana para explorar las diferentes carpetas del PC y seleccionar la imagen deseada, siendo BMP el único formato aceptado, ya que es con el que guardan las imágenes desde el software de la cámara, Sapera CamExpert. En el caso de existir la necesidad de modificar o ampliar el número de formatos aceptados, tan solo sería necesario añadir un parámetro al código. Para ello, se ha empleado la función **uigetfile**, cuya configuración es la siguiente**:**

```
[FileName,PathName,FilterIndex] = uigetfile(FilterSpec,DialogTitle);
```

Salidas

- FileName: Devuelve el nombre del archivo seleccionado; en este caso se asigna como **nim.**

- PathName: Esta salida devuelve la dirección de la carpeta donde se encuentra alojado la imagen seleccionada; se asigna a la variable **path.**

- FilterIndex: Devolverá un 1 si se ha seleccionado alguna imagen o un 0 si se presiona en Cancelar; se asigna a la variable **ok.**

Entradas

- FilterSpec: Se especifican los formatos de archivo que serán aceptados; en este caso solo se aceptan imágenes con formato **BMP**.

- DialogTitle: Se emplea para mostrar un título para la ventana del explorador desplegado; en este caso se muestra **"Abrir imagen evaluada".**

Con esta información, la función aparecerá en el código de la siguiente forma:

```
[nim,path,ok]=uigetfile({'*.bmp'},'Abrir imagen evaluada');
```

En el caso de que la variable **ok** sea igual a 1, es decir, que se ha seleccionado una imagen válida, es necesario unir la dirección del archivo con su nombre. A esto se le denomina concatenar cadenas de texto o string, lo cual se realiza mediante la siguiente función de MATLAB:

```
s = strcat(s1,...,sN)
```

Salidas

- s: Devuelve la cadena de texto resultado de unir las que se le hayan pasado como parámetro de entrada; se le denomina como **preim.**

Entradas

- s1…sN: Cada una de las cadenas de texto que se quiere unir; en este caso se unen solo dos: **path** y **nim**.

```
preim=strcat(path,nim);
```

Para guardar la imagen como una variable solo es necesario pasarle este último string como parámetro de entrada a la función **imread**:

```
A = imread(filename);
```

Salidas

- A: Devuelve el archivo seleccionado en una variable; en este caso se trata de una imagen en escala de grises, por lo que la guardará en la variable **im** en forma de una única matriz de dimensiones 6000x4096.

Entradas

- filename: Se trata del nombre del archivo, siendo necesario introducir solo el nombre si se encuentra alojado en la carpeta activa de MATLAB. En caso contrario, como éste, es necesario introducir el directorio completo, **preim.**

Tras seleccionar la imagen, ésta queda guardada en MATLAB como una variable, pudiendo trabajar sobre ella desde el código implementado. Sin embargo, no toda la placa tiene que ser analizada, sino solo aquella zona en la cual se ha aplicado el barniz, dentro de la cual se ubican un gran número de componentes.

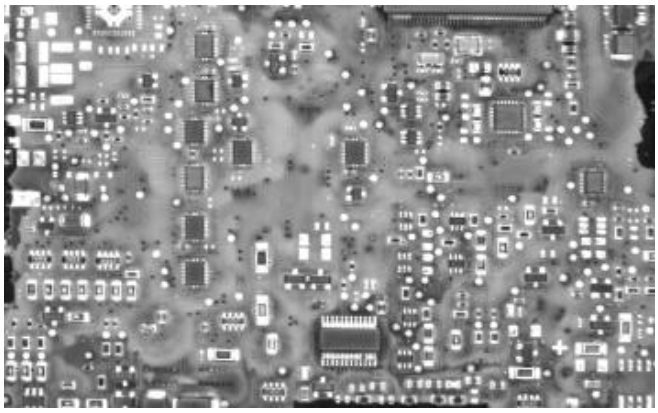


Figura 95: Vista detalle PCB a analizar

La siguiente tarea a llevar a cabo sería segmentar cada uno de los componentes mediante un algoritmo, es decir, separar cada uno de ellos de forma automática para su posterior análisis y así realizar un control de barnizado de la PCB completa. Sin embargo, y dada la dificultad de la segmentación por componente, se ha optado por una segmentación por posición. Todas las PCB son idénticas en cuanto a componentes y posición se refiere, por lo que se investigó la posibilidad de buscar un punto de referencia común y utilizarlo como origen de cada una de las imágenes. De esta forma, todos los componentes quedarían perfectamente localizados para su posterior selección y segmentación.

Esta idea es válida para un proyecto como éste, un estudio de viabilidad o prueba de concepto, no para una solución industrial sólida. Pero debido a que se trata de un proyecto de I+D con recursos limitados, este es uno de los problemas que no ha dado tiempo a solventar, lo que no implica que ya se hayan ideado estrategias a seguir en el caso de que se continúe con el desarrollo del prototipo. Una de ellas, y de las que posiblemente aporte mejores resultados es la introducción de una cámara RGB de tal forma que se pueda segmentar por color, algo que no es posible con las muestras actuales al ser imágenes en escala de grises.

Se decidió, por tanto, continuar con la línea de la segmentación por posición. Hubo que buscar un punto de referencia que tuviesen todas las placas y lo suficientemente diferenciado de su entorno de tal forma que sirviese como punto de referencia y alinear todas las PCB, barajándose diversas opciones. Unos días atrás, una de las patas que sostiene la PCB se partió por la base, quedando uno de los cuatro taladros de la PCB libre, de tal manera, que al no ser totalmente necesario para la sujeción de las PCB y quedando éstas fijas durante el ensayo, se decidió elegir el taladro como punto de referencia.

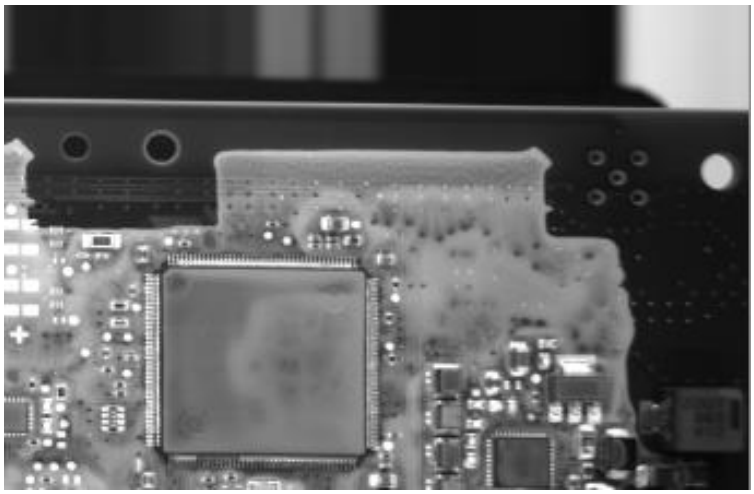


Figura 96: Vista detalle esquina superior derecha

En la figura 96 se aprecia como el taladro de la placa queda libre y aparece un círculo claro rodeado de un entorno muy oscuro, lo cual es fácilmente diferenciable. El objetivo era el de alinear las imágenes, existiendo únicamente un grado de libertad: solo existe variabilidad de las imágenes en la vertical, no existiendo desplazamiento en la dirección horizontal. Concretamente, el punto de referencia será el centroide del círculo, habiéndose seguido los siguientes pasos para la localización y cálculo del mismo.

1) Dado que se conoce de forma aproximada la zona donde puede ubicarse el círculo, se dibuja un rectángulo en la imagen original de tal manera que lo que quede en su interior se mantenga y el exterior se ponga a cero, es decir, en el negro absoluto.

   Para ello, en primer lugar se crean 2 vectores con las cuatros esquinas del rectángulo que se quiere conseguir:

```
c = [3850 4096 4096 3850];
r = [400 400 1200 1200];
```

   Estos vectores se usarán como parámetros de entrada en la función **roipoly**:

```
BW = roipoly(I, c, r);
```

   Salidas

   - BW: Devuelve una imagen lógica con los píxeles que quedan dentro del polígono a 1, y a 0 los que quedan fuera del mismo. En nuestro caso se asigna a la variable BW.

   Entradas

   - I: Imagen a la cual se le quiere hallar la ROI (Region of Interest); se le pasa la variable **im.**

- c y r: Son dos vectores que indican los vértices del polígono, conteniendo c los valores de la coordenada horizontal de cada punto y r, la vertical.

```
BW = roipoly(im,c,r);
```

Sin embargo, se quiere obtener una imagen como la de la figura 97, para lo cual será necesario realizar una serie de transformaciones sobre la imagen binaria BW. En primer lugar se pasa a formato *unsigned integer (uint8)* mediante el uso de la función **im2uint8**, de tal forma que donde antes había un 1 lógico, ahora estará el valor entero 255, manteniéndose los 0, ahora de tipo *uint8.* El objetivo es multiplicar por uno el nivel de gris de aquellos píxeles que nos interesan y por cero los que no, para que queden en negro. Para ello, se divide la matriz entre 255 para poder multiplicarla por la imagen original.

Se han empleado las siguientes funciones en el orden que en el que se muestran a continuación:

```
BW=im2uint8(BW);
BW=BW*(1/255);
r=im.*BW;
```

El resultado se asigna a la variable r, que se corresponderá con la imagen de la figura 97.

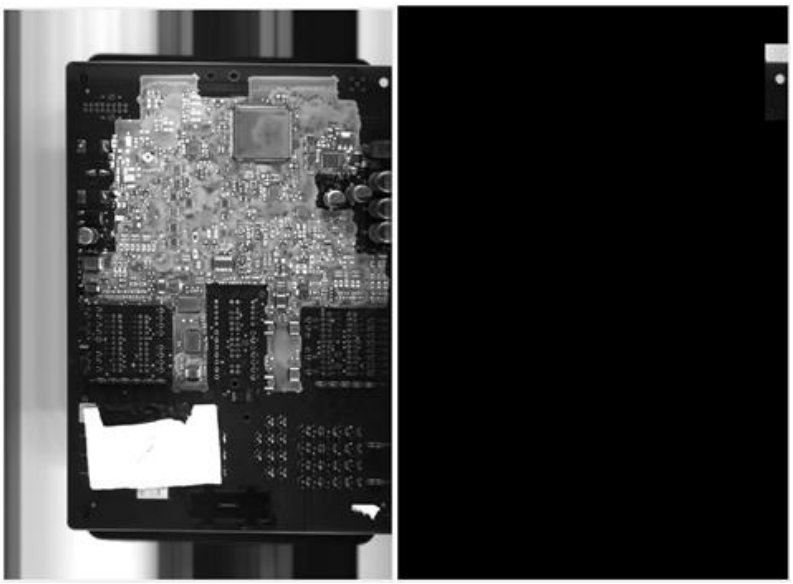


Figura 97: PCB a analizar // Esquina recortada

2) Una vez se obtiene esta imagen, es mucho más fácil identificar el círculo y calcular su centroide. Para ello, es necesario binarizar previamente la imagen, es decir, crear una imagen lógica en base a un umbral, de tal forma que los niveles de gris que se sitúen por encima del mismo se convierten en un uno lógico (True) y los que se encuentren por debajo, en cero (False). Esto se realiza con la función **im2bw**.

```
BW = im2bw(I, level);
```

Salidas

- BW: Devuelve una imagen lógica en la cual a todos aquellos píxeles cuyo nivel de intensidad de gris se sitúe por encima del umbral *level* se les asigna un uno lógico y un cero en el caso contrario. Se asigna a la variable **r.**

Entradas

- I: Imagen a la cual se le quiere hallar la ROI (Region of Interest); se le pasa la variable **im.**

- level: Umbral en forma normalizada, es decir entre 0 y 1. En este caso tendrá un valor de **0.2**, el cual se ha hallado heurísticamente y permite que todos aquellos píxeles cuyo valor de gris sea superior a 51 se pongan a nivel alto, y por tanto, se pueda segmentar de manera efectiva el círculo.

```
r=im2bw(r,0.2);
```

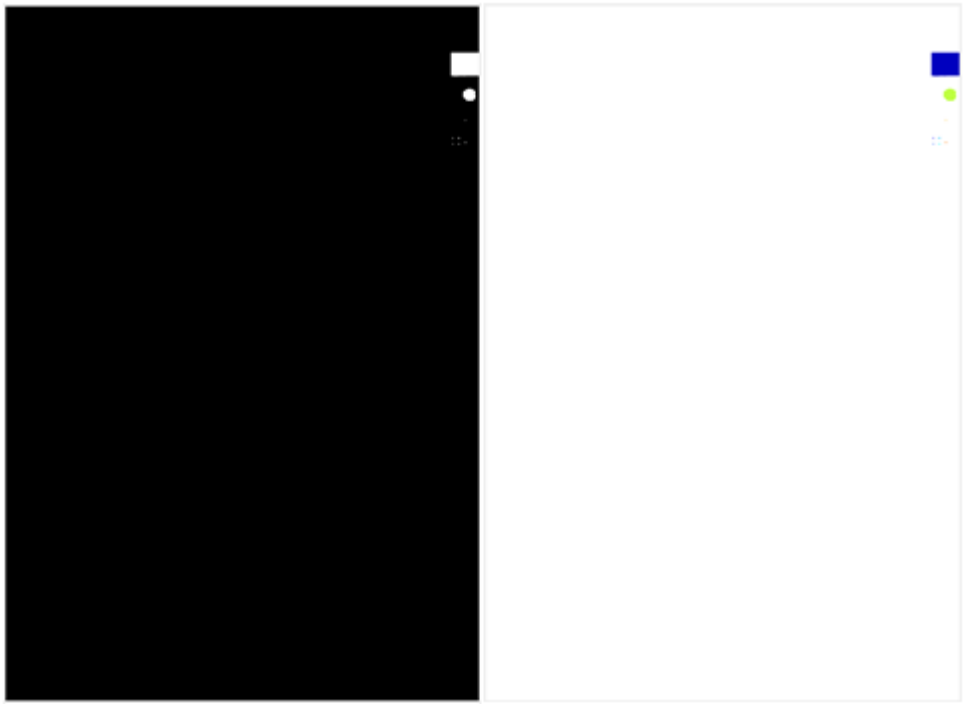


Figura 98: Esquina binarizada // Regiones coloreadas

3) Una vez que se tiene la imagen binarizada, es necesario aplicar una función que pasándole esta imagen como parámetro de entrada, devuelve como salida las diferentes regiones en las que se divide la imagen binaria. Es decir, declara una zona como región si se trata de un grupo de unos lógicos adyacentes y separados de otras regiones, asignándole un índice a cada una de ellas. La función utilizada es **bwlabel**.

Salidas

- L: Devuelve una matriz en la que se le asigna un índice (1,2,…,n) a los píxeles de cada una de las regiones detectadas. Se asigna a **r**.

- num: Muestra el número de regiones identificadas; se mantiene el mismo nombre para la variable.

Entradas

- BW: Imagen a la cual se le quieren detectar las regiones. En este caso, **r**.

```
[r,num]=bwlabel(r);
```

Además, para que el usuario pueda visualizarlo se ha utilizado la función **label2rgb**, que asigna un color diferente a cada una de las regiones, tal como se aprecia en la figura 98. El parámetro de entrada es la imagen con los índices de las regiones, devolviendo una imagen RGB.

4) Al tener las diferentes regiones identificadas, pueden extraerse características relativas a las mismas, de tal forma que sea posible el reconocimiento del círculo. En este caso, se han calculado tres propiedades para cada región: centroide, área y perímetro. Se ha utilizado para ello la función **regionprops**.

```
stats = regionprops(BW,properties);
```

Salidas

- stats: Devuelve una variable tipo estructura con toda la información acerca de las propiedades que se hayan seleccionado. En el código se han calculado 3 propiedades distintas por separado, de tal forma que se han asignado a las variables **cent** (coordenadas del centroide), **area** y **per** (perímetro de la región).

Entradas

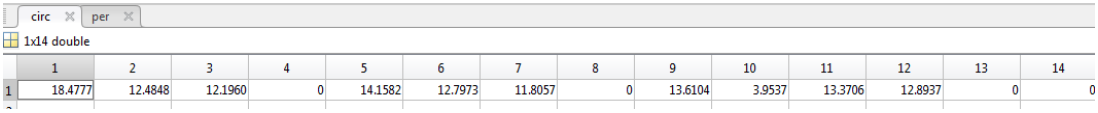
- BW: Imagen con las regiones indexadas. Se le introduce la variable **r**.
- properties: Se le introduce aquella propiedad o propiedades que se quieren hallar para cada una de las regiones; en este caso se hallan por separado “Centroid‟, “Area‟ y “Perimeter‟.

```
cent=regionprops(r,'Centroid');
area=regionprops(r,'Area');
per=regionprops(r,'Perimeter');
```

Se ha calculado otra propiedad de forma indirecta, la circularidad, la cual se calcula mediante la siguiente expresión:

$$C = \frac{P^2}{A} \qquad \begin{cases} \mathrm{C} \rightarrow \text{Circularidad} \\ \mathrm{P} \rightarrow \text{Perímetro} \\ \mathrm{A} \rightarrow \text{Área} \end{cases}$$

Calculando estos valores y realizando una comparación para cada una de las regiones, se puede determinar con exactitud el círculo, y por consiguiente, el centroide del mismo. Se han utilizado la circularidad y el area, comprobando si los valores de esas propiedades para cada una de las regiones se encuentra dentro de un rango.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	18.4777	12.4848	12.1960	0	14.1582	12.7973	11.8057	0	13.6104	3.9537	13.3706	12.8937	0	0

Figura 99: Circularidad

El paso siguiente será desplazar la imagen completa hasta un punto de referencia común. Contamos ya con el centroide del círculo, cuyas coordenadas se denominarán como (x0,y0). Por otra parte, necesitamos conocer las dimensiones de la imagen, lo que se puede calcular aplicando la función **size**. El objetivo es hacer coincidir el centroide del círculo con la esquina superior derecha.

```
[m,n] = size(X);
```

Salidas

- m y n: m corresponderá con el número de filas, 6000 en este caso, y n con el de las columnas, 4096.

Entradas

- X: Imagen a la se le calcula el tamaño. En este caso, **im**.

```
[m,n]=size(im);
```

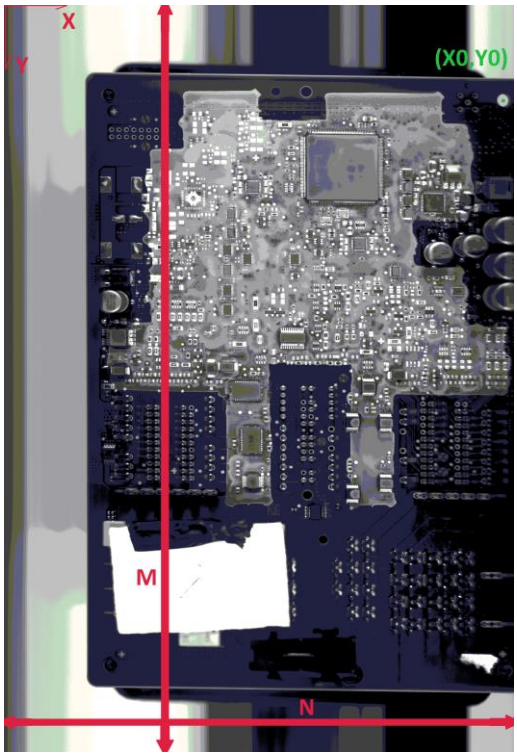


Figura 100: Dimensiones PCB

A continuación, se utilizan los parámetros calculados antes para desplazar la imagen, para lo que se utiliza la función **circshift**.

```
Y = circshift(A,K);
```

Salidas

- Y: Devuelve la imagen que se le pasa como parámetro de entrada desplazada. Se guarda en la variable **imnew**.

Entradas

- A: Imagen que se quiere desplazar. En este caso, **im**.
- K: Vector de 2 elementos en el cual, el primer número se corresponderá con el desplazamiento positivo en el eje vertical (hacia abajo), y el segundo, con el desplazamiento positivo en la horizontal (hacia la derecha)

De esta forma y conociendo que las dimensiones de la imagen son M (largo) y N (ancho), se puede deducir que los desplazamientos serán los siguientes:

```
imnew=circshift(im,[-y0,n-x0]);
```

Una vez realizada esta operación es posible visualizar el resultado, tal como se muestra en la figura 101.

Figura 101: PCB Desplazada

Se observa la rotación de la imagen, es decir, las filas superiores que se encontraban por encima del centroide han pasado a estar ubicadas en la parte inferior, de igual forma que las filas que se encontraban a la derecha del centroide, ahora se ubican a la izquierda de la imagen. Así bien, una vez que se someta a este proceso a cada una de las muestras de la PCB, todas se encontrarán alineadas.

Aunque este proceso se ideó con la finalidad de realizar una segmentación por posición, también podría emplearse en una solución industrial para corregir los posibles desplazamientos accidentales que se produzcan.

Para finalizar esta etapa, se produce el recorte de cada uno de los componentes seleccionados de manera que puedan ser analizados posteriormente. Para ello se utiliza la función **imcrop**, la cual genera una imagen rectangular.

```
I2 = imcrop(I, rect);
```

Salidas

- I2: Devuelve la imagen recortada. Se guarda en la variable **imcx**, siendo x un número del 1 al 7, uno por cada uno de los componentes.

Entradas

- I: Imagen principal a la que se le quiere realizar el recorte. En este caso será la variable **im**.
- rect: Vector de 4 elementos en el cual cada número significa, en orden:
  - Coordenada X de la esquina superior izquierda
  - Coordenada Y de la esquina superior izquierda
  - Dimensión en X (Ancho)
  - Dimensión en Y (Largo)

De esta forma se recortan las imágenes para siete componentes con patillas, como los ejemplos mostrados en la figura 102.

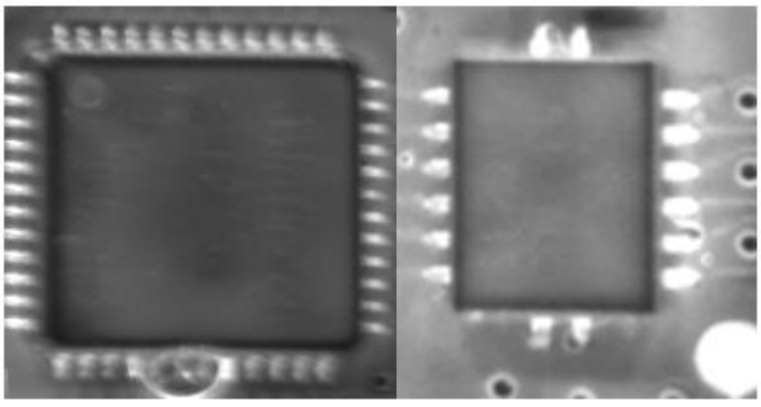


Figura 102: Componente 1 // Componente 2

4.2.3 Etapa 2: Detección de burbujas

Las imágenes de componentes individuales servirán como punto de partida para esta segunda fase de procesado, aquella en la que las burbujas son detectadas. Se analizarán, por tanto, 7 componentes diferentes para cada una de las PCB, siendo 6 similares al mostrado en la figura 102b (Componentes 2-7) y otro como el mostrado en la figura 102a, al que llamaremos Componente 1.

Para cada uno de los dos grupos de componentes se ha empleado un tipo de análisis diferente: para el Componente 1 se ha utilizado correlación entre imágenes, mientras que para el resto de componentes se han detectado las burbujas una por una.

4.2.3.1 Componente 1

Se trata de un componente SMD, el cual tiene forma cuadrada, con 12 pines por lado y de dimensiones 6x6mm. Fue el primer componente que se trató de analizar ya que, durante la fase de adquisición, nos dimos cuenta de que presentaba una burbuja en la mayoría de los casos y además, ésta era visible a simple vista, sin la necesidad de utilizar luz ultravioleta. Para su análisis, previamente se ha escogido una muestra correcta del componente para que sirva de modelo, concretamente la de la placa 10.

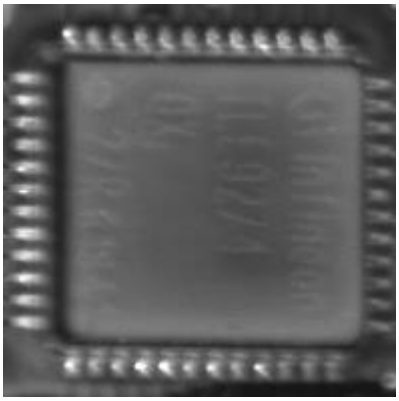


Figura 103: Componente 1 OK

En este caso, el código no entraña mucha complejidad: se emplea la función **corr2** para obtener como parámetro de salida un coeficiente de correlación que se encuentra en el rango [-1,1], pasándole como parámetros de entrada las dos imágenes a comparar. Matemáticamente, la función **corr2** utiliza la siguiente expresión:

$$c = \frac{\sum_m \sum_n \left(A_{mn} - \bar{\bar{A}}\right)(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{\bar{A}})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}}$$

A y B representan a cada una de las imágenes, siendo m y n el número de filas y columnas, $\bar{A}$ el valor medio de gris para esa imagen, y A_{mn} el valor de gris de cada píxel.

De esta forma se calculan dos coeficientes de correlación para evitar falsos positivos:

- **Coeficiente 1 (n)**: se compara la imagen del componente analizado, **im2c1**, con otra imagen de un componente correcto, **im1c1.**

```
n=corr2(im1c1,im2c1);
```

- **Coeficiente 2 (nx)**: se compara la imagen del componente analizado, **im2c1**, con una imagen de un componente con burbuja, **imko1**.

```
nx=corr2(imko1,im2c1);
```

Una vez que se tienen los dos coeficientes calculados, se realiza una doble comparación para determinar la presencia de burbujas; el coeficiente 1 al tener una imagen buena como referencia, debe ser mayor de un umbral, mientras que el coeficiente 2 debe ser menor de otro umbral al comparar con una imagen mala.

Si se determina que el componente no tiene burbuja, se dibuja un cuadro verde en torno al mismo en la imagen de la PCB y se imprime la expresión OK en la etiqueta situada debajo del Figure donde se muestra el componente.

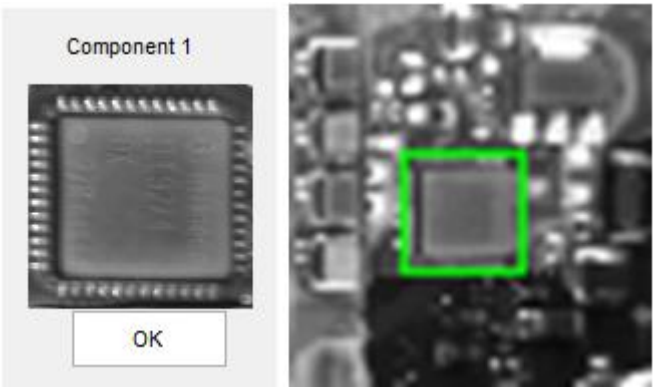


Figura 104: Análisis Componente 1 OK

En el caso de que se haya detectado alguna burbuja en las patillas del componente, el cuadro dibujado será de color rojo y aparecerá KO en la etiqueta inferior.

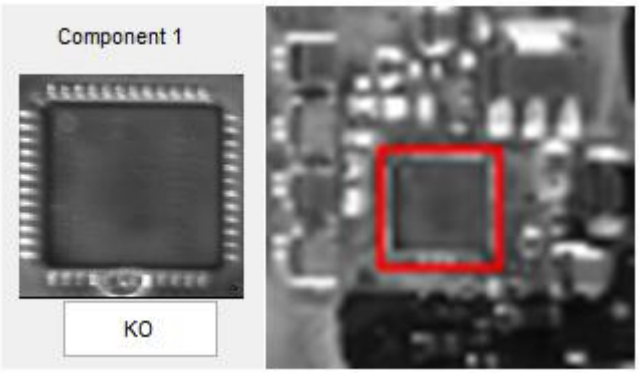


Figura 105: Análisis Componente 1 KO

4.2.3.2 Componentes 2-7

En este caso, nos encontramos con un componente SMD de forma rectangular, de dimensiones 3.2x2.2mm, el cual posee 2 pines en las partes superior e inferior, y 6 en cada uno de los laterales. Fue elegido por diversos motivos:

- Es uno de los encapsulados con patillas más repetidos en la placa.
- Presentaban burbujas en la mayoría de los casos.

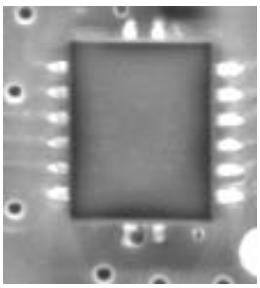


Figura 106: Componente 2 a analizar

Este componente, a diferencia del mostrado en el apartado anterior, se repite varias veces en la PCB, por lo que no sería eficiente repetir 6 veces el código perteneciente al procesado en el programa principal. Se ha optado por construir una función o subrutina que sea llamada cada vez que se quieran detectar las burbujas de un componente de este tipo. La función se denomina ProcBurb:

- **Parámetro de entrada**: Imagen a analizar
- **Parámetros de salida**: Número de burbujas detectadas (n) y color del cuadro para ese componente (c)

Esta función realiza todo el proceso de detección de burbujas mediante el siguiente procedimiento:

1) La zona de interés para el análisis son las patillas, por lo que el resto de la imagen, además de no aportar nada, introduce falsos positivos. Debido a ello, es necesario seleccionar esta zona; un método eficaz consiste en detectar el rectángulo negro perteneciente al encapsulado y, a partir de ahí, eliminar las zonas no deseadas.

   Para reconocer este rectángulo, en primer lugar, es necesario realizar la ecualización de la imagen de cara a mejorar el contraste de la misma y resaltar en negro el borde del encapsulado.

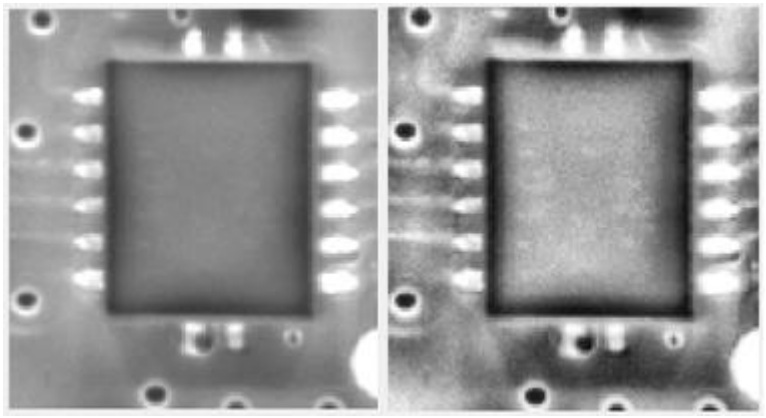


Figura 107: Imagen C2 // Imagen C2 ecualizada

Sin embargo, no se trata de una ecualización estándar, sino de una ecualización adaptativa con la función **adapthisteq**, actuando en pequeñas parcelas de la imagen llamadas “Azulejos” en lugar de actuar globalmente, lo que mejora el contraste de la imagen y evita que se realce el ruido. Por defecto, se establece que el número de azulejos sean 64, 8 columnas por 8 filas.

```
J = adapthisteq(I);
```

Salida

- J: Devuelve la imagen ecualizada. Se guarda en la variable **im**.

Entrada

- I: Imagen que se quiere ecualizar. En este caso será la variable **im**.

```
im=adapthisteq(im);
```

Una vez que se tiene esta imagen se le aplica un procesado morfológico, para lo cual es necesario obtener los siguientes resultados:

- **Elemento estructurante**: para cualquier operación de procesado morfológico es necesario establecer previamente un elemento estructurante. Para ello, se ha utilizado la función de MATLAB strel.

```
SE = strel(shape, parameters);
```

Para que la función devuelva un elemento estructurante, **se**, es necesario introducirle una serie de parámetros de entrada: para el primero, *shape,* se indica la forma del elemento estructurante, en este caso, será un disco (**disk**). En el segundo, *parameters*, se pueden introducir datos específicos para cada uno de las formas; para el disco se ha indicado el radio del mismo, **4** píxeles.

```
se=strel('disk',4);
```

- **Imtophat**: se trata de una función de MATLAB que realiza la apertura de la imagen según el elemento estructurante anterior y resta el resultado a la imagen original.

```
IM2 = imtophat(IM,SE);
```

Únicamente se le introducen como parámetros de entrada la imagen **im**, y el elemento estructurante, **se**, para obtener la imagen modificada, **im**.

- **Imbothat**: se trata de una función de MATLAB que realiza el cierre de la imagen según el elemento estructurante y le resta la imagen original. Los parámetros coinciden con los de la función imtophat.

```
IM2 = imbothat(IM,SE);
```

Con estos elementos se realiza la siguiente operación matemática con el fin de obtener un mayor contraste y poder segmentar correctamente el rectángulo del encapsulado.

$$\mathrm{im} = \left(imor + \mathrm{imtophat}(imor)\right) - \mathrm{imbothat}(imor)$$

$$\begin{cases} \mathrm{im} \rightarrow \text{Imagen resultado} \\ \mathrm{imor} \rightarrow \text{Imagen original} \end{cases}$$

```
im=imsubtract(imadd(im,imtophat(im,se)),imbothat(im,se));
```

Tras aplicar esta operación se obtiene el siguiente resultado, donde puede apreciarse como los bordes quedan resaltados en negro con respecto a su entorno; para quedarnos con esa zona al binarizar, le aplicamos el complemento a la imagen con la función **imcomplement**, es decir, invierte los niveles de gris.

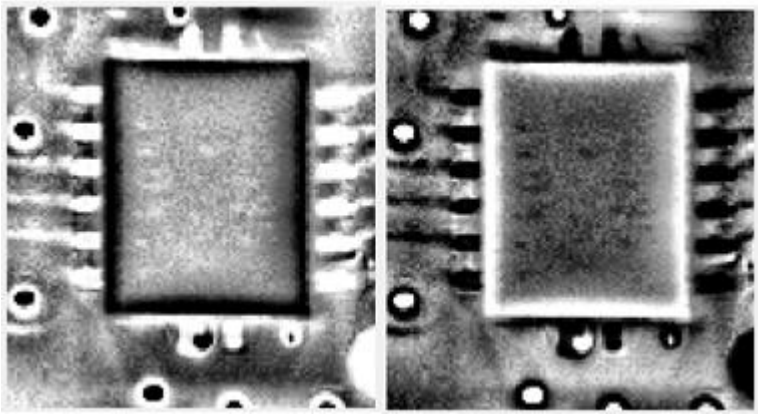


Figura 108: Imagen postoperación // Imagen invertida

```
IM2 = imcomplement(IM);
```

Salida

- IM2: Devuelve la imagen en escala de grises con los niveles de gris invertidos; para cada píxel se le resta el valor de intensidad del mismo a 255. Se guarda en la variable **im**.

Entrada

- IM: Imagen original a invertir. En este caso será la variable **im**.

```
im=imcomplement(im);
```

Esta última imagen, la relativa a la figura 108, se utilizará posteriormente para hallar las burbujas, ya que éstas quedan diferenciadas de su entorno. Por el momento servirá para hallar el rectángulo que forma el encapsulado del componente, la cual se binarizará con un umbral normalizado de **0.85**, obteniendo una imagen como la de la figura 109a. Sin embargo, solo nos interesará el objeto más grande, por lo que se eliminarán todos aquellos cuya área sea inferior a un umbral.

Figura 109: Reconocimiento de regiones // Filtrado de regiones

Una vez aquí, se realiza un proceso parecido al reconocimiento del círculo en el alineamiento de las imágenes; se reconocen las regiones, que en este caso será solo una y se le extrae mediante la función **regionprops**, una nueva

característica llamada **BoundingBox**, el rectángulo de mínimas dimensiones que alberga a la totalidad de los píxeles de la región, coincidiendo con el encapsulado.

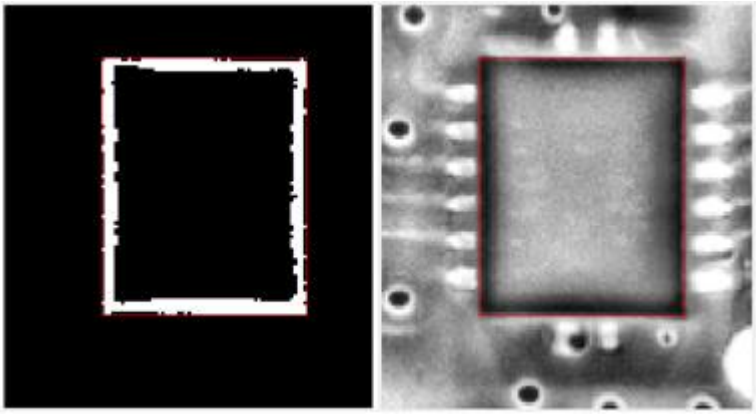


Figura 110: BoundingBox sobre región // BoundingBox sobre componente

2) Los datos que se conocen respectivos a la BoundingBox, son las coordenadas de la esquina superior izquierda, el ancho y el largo, por lo que se tiene perfectamente ubicado al componente independientemente de las pequeñas variaciones que pudiese sufrir la posición del mismo a pesar del alineamiento, haciendo más robusto al sistema. Con esto, se vuelve a trabajar con la imagen de la figura 108b, eliminando toda aquella información no necesaria y manteniendo únicamente un marco alrededor del componente, donde se ubicarán los pines.

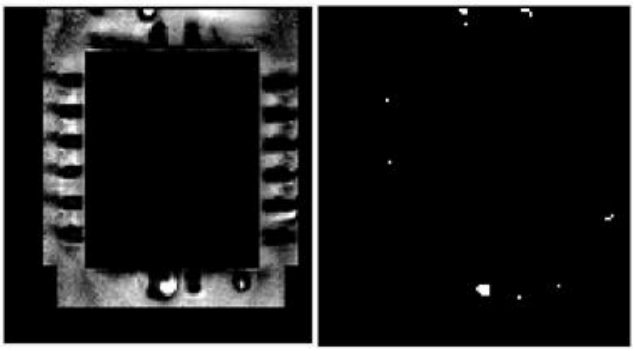


Figura 111: Eliminación información no relevante // Reconocimiento burbujas

A continuación, la imagen se binariza con un umbral normalizado de 1, obteniendo el resultado visto en la figura 111; las burbujas se distinguen por ser agrupaciones de píxeles con forma circular, cosa que permitirá distinguirlas del ruido.

De nuevo, se filtran las áreas más pequeñas y se extraen las características necesarias para calcular la circularidad, propiedad que servirá como criterio para el clasificador, determinando si se trata o no de una burbuja.

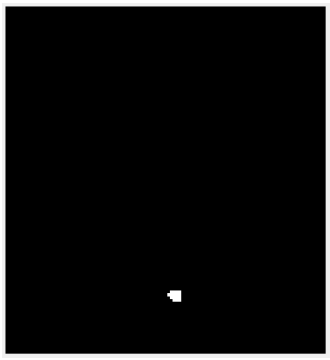


Figura 112: Filtrado regiones burbujas

Por último, se dibujará un círculo rojo sobre cada una de las burbujas detectadas a la vez que se incrementa el parámetro de salida n, devolviendo el número total de ellas. En el caso de que ninguna sea detectada, el parámetro de salida correspondiente al color del cuadro de la imagen de la PCB será verde; en caso contrario, será rojo.

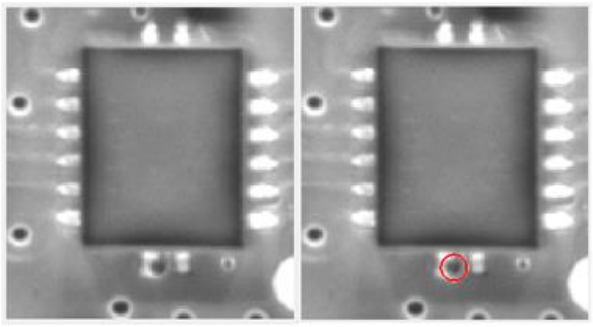


Figura 113: Imagen original vs Imagen con burbuja detectada

4.2.4 Mejoras de procesado en curso

Es usual que en el entorno de la visión por computador se realicen análisis comparativos ya sean de una imagen completa o de una parte de ella. Por ejemplo, el reconocimiento facial en la seguridad de los aeropuertos, o la lectura de matrículas en los aparcamientos. El denominador común es que existe un patrón con el cual comparar una muestra cualquiera.

En el caso de las PCB ocurre lo mismo: las placas son siempre las mismas y los componentes están localizados de forma idéntica. Es por ello que se decidió realizar pruebas siguiendo este procedimiento.

1) En primer lugar, se abre un explorador para abrir dos imágenes diferentes del mismo tipo de componente: una muestra de un componente correcto y otra de un componente que presenta burbujas, con el fin de compararlas. La función utilizada para ello es **uigetfile**, empleándose de forma similar a la descrita en el apartado 4.2.2.

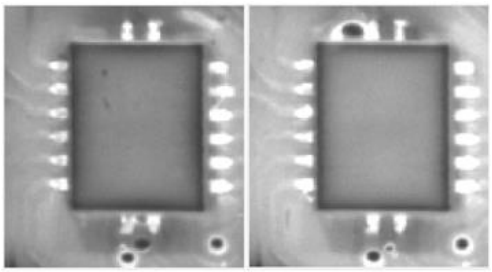


Figura 114: Componente sin burbuja // Componente con burbuja

2) Tras esto, se actúa de forma similar que en el apartado 4.2.3.2, es decir, reconociendo el encapsulado del componente. De esta forma es posible conocer tanto la posición como el tamaño del mismo, haciendo posible la superposición de ambas imágenes. Para ello, se ha desarrollado la función **DComp**, a la cual se le pasa como parámetro de entrada la imagen original del componente, y devuelve como salida un vector de 4 elementos: los dos primeros corresponden a las coordenadas de la esquina superior izquierda del componente, y los dos siguientes, el ancho y el alto respectivamente. Esto se aplica a ambas imágenes.

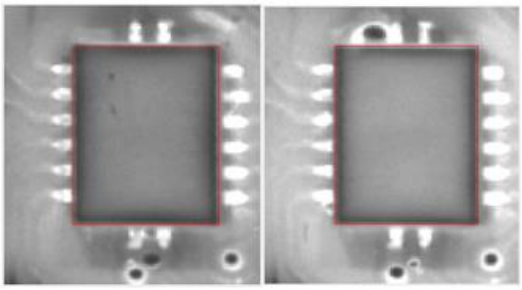


Figura 115: Reconocimiento del encapsulado

3) Una vez conocidas estas dimensiones, se produce una doble transformación de la imagen del componente con burbujas:

   - Se hace coincidir la esquina superior izquierda del componente con burbujas con la de la imagen del componente correcto mediante la función **circshift** explicada en el apartado 4.2.2.

   - Seguidamente, se redimensiona la segunda imagen, la del componente con burbujas para hacer que coincidan las patillas de ambos componentes, permitiendo que éstas puedan ser eliminadas cuando se cree una imagen en diferencias.

   Para ello, debe calcularse previamente el número de filas y columnas que debe tener la nueva imagen; en el ejemplo de las filas, se multiplica el número de filas de la imagen original por un coeficiente, siendo este último la división del ancho del componente correcto entre el ancho del que

presenta burbujas. De esta forma, se hace coincidir tanto el largo como el ancho de ambos componentes.
Se utiliza la función **imresize**, cuya configuración es la siguiente:

```
B = imresize(A, [numrows numcols])
```

Salida

- B: Devuelve la imagen redimensionada. Se guarda en la variable **im2**.

Entradas

- A: Imagen que se quiere redimensionar. En este caso será la variable **im2**.
- [numrows numcols]: Corresponden con el número de filas y columnas que tendrá la nueva imagen, en este caso, **c1** y **c2.**

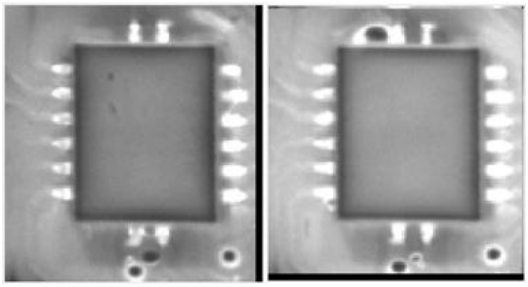


Figura 116: Imágenes coincidentes

4) Una vez que se tienen estas imágenes, se resta la que contiene la burbuja a la original, generándose una imagen como la siguiente.

Figura 117: Imagen en diferencias

Puede apreciarse en la parte superior la burbuja que aparece en el componente analizado. En un caso ideal, en el que las condiciones de adquisición fuesen totalmente fijas, solo tendría que aparecer esa burbuja, manteniéndose el resto en negro, por lo que la segmentación sería sencilla: solo habría que detectar aquello que no fuese totalmente negro.

Por tanto, con las imágenes actuales no es posible utilizar este método, ya que éstas no son lo suficientemente robustas. En el caso de que el proyecto continúe y se controlen totalmente las condiciones de adquisición e iluminación, podría probarse la viabilidad de este método.

5. Resultados

Tras la descripción detallada de cada una de las partes que forman parte del proyecto, es fácil tener una idea general del mismo y conocer el contexto en el que está situado. Por ello, y a razón de los resultados obtenidos puede decirse que los resultados han sido los esperados.

El punto de partida fue únicamente la necesidad de detectar las burbujas en el barnizado que se le aplica las placas de circuito impreso, desconociendo esta problemática previamente. Con todo ello, y gracias la evolución descrita, se consiguió diseñar una estación prototipo y el software necesario para la detección de burbujas. En cuanto al hardware, finalmente se ha diseñado y desarrollado una estación para ensayos o setup como el que aparece en la figura 118.

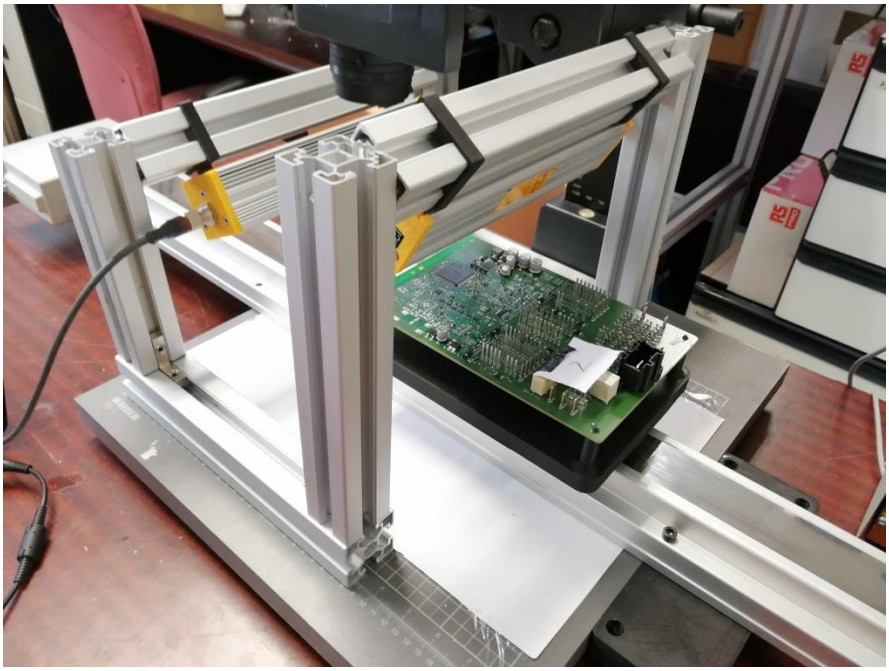


Figura 118: Estación prototipo

Por otro lado, cabe destacar que para su construcción se han utilizado, entre otros, elementos puramente mecánicos y estructurales diseñados y fabricados por el autor: utillaje, enganches para iluminación y estructura de iluminación, lo cual ha supuesto un reto ya que el autor no disponía de experiencia ni conocimientos previos en cuanto a CAD dado el perfil electrónico del mismo, por lo que resultó necesario solicitar la asistencia de algún compañero del GRAV con especialidad en mecánica cuando se diseñó y fabricó el utillaje mediante impresión 3D, la primera pieza de la que se dispuso. Gracias a ello, se pudo adquirir progresivamente la habilidad necesaria para el diseño del enganche y las dos versiones de la estructura de iluminación.

Figura 119: Versión 1 Estructura Iluminación

También se emplearon equipos comerciales tales como la cámara, la iluminación y el actuador lineal, cuyo manejo hubo que investigar y desarrollar a lo largo de las diferentes etapas por la que el proyecto ha transcurrido, las cuales se plasman en la figura 120.

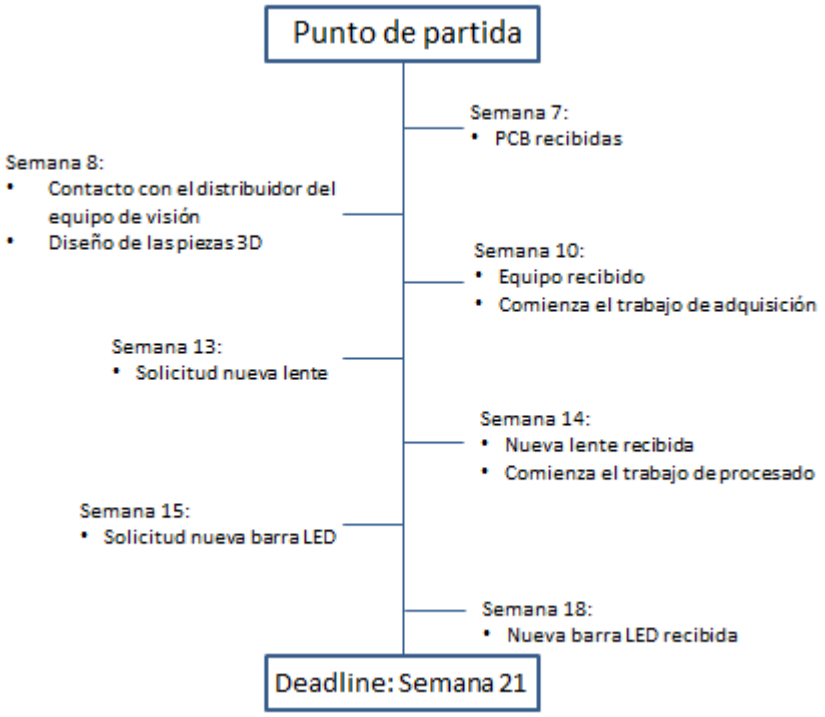


Figura 120: Cronograma

Figura 121: Cámara LINEA con objetivo

En paralelo al desarrollo de la estación prototipo, se llevó a cabo el procesado de imágenes, el cual culminó con la creación del software descrito en el capítulo 4, basado en una interfaz de usuario que permite seleccionar una imagen de una PCB para ser analizada y seguidamente, procesar una muestra de los componentes con patillas y mostrar los resultados para cada componente en esta GUI.

Una vez llegado a este punto, donde la estación estaba completamente construida y validada, y el software de procesado había pasado por sucesivas etapas, se llevó a cabo un análisis de la totalidad de las placas de las que se disponía.

- Cantidad de PCB: **14**
- Componentes analizados por placa: **7**

Cantidad total de componentes analizados: **98**

Al analizarlos, se encontraron burbujas de diferentes tamaños, teniendo algunas como diámetro hasta 1 píxel. Debido al pequeño tamaño de algunas y a las características de la cámara, por el momento el sistema solo es capaz de detectar burbujas de hasta 150 µm de diámetro aproximadamente. Es por eso que se distinga entre burbujas detectables y no detectables, lo cual se debe a las condiciones del sistema y no al software.

- Cantidad de burbujas detectables: **138**
- Cantidad de burbujas detectadas: **124**

Porcentaje de burbujas detectadas: **89.85%**

Por otra parte y uno de los principales problemas a resolver, son los falsos positivos que aparecen debido a las condiciones de iluminación. Concretamente, se han producido un total de 34, lo que supone un 25% de falsos positivos. Sin embargo, el 77% de ellos se producen en los laterales de los componentes. Esto se debe a que, a diferencia de las partes superior e inferior de los mismos, no existe una fuente de iluminación que incida de manera directa en el patillaje lateral, incrementando la aparición de falsos positivos. Si esto se soluciona, el ensayo presentaría únicamente un 5.7% de falsas burbujas detectadas.

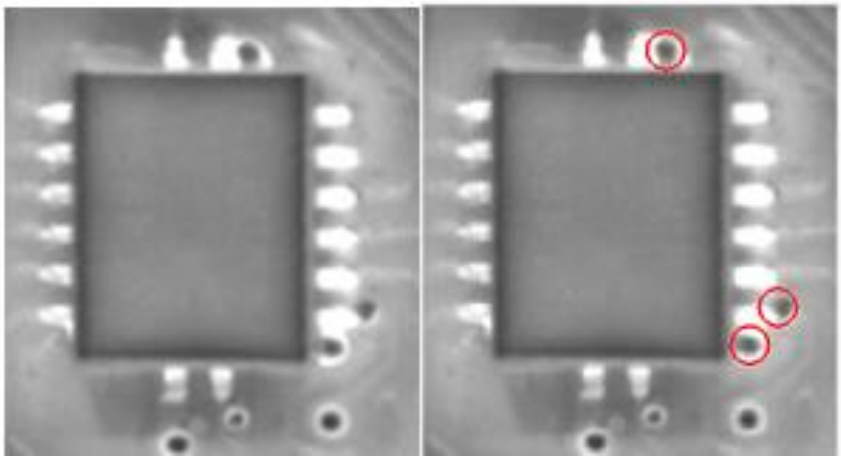


Figura 122: Componente con burbujas

Por ejemplo, la figura 122 se corresponde con un componente que ha sido analizado habiendo encontrado la totalidad de las burbujas del mismo, en concreto, 3, y no presentando ningún falso positivo. Por el contrario, componente mostrado en la figura 123 presenta 2 falsos positivos en su lado derecho.

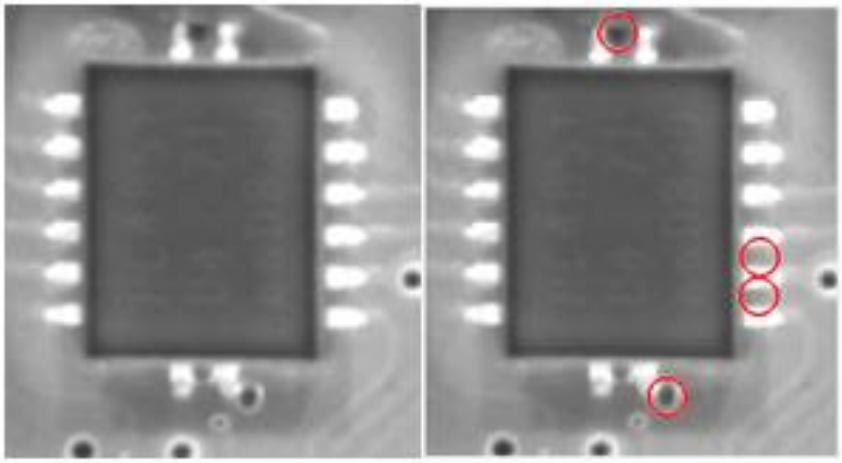


Figura 123: Componentes con falsos positivos

6. Conclusión y trabajos futuros

Finalmente, con los datos obtenidos se ha demostrado la viabilidad del proyecto, ya que alrededor del 90% de las burbujas que reúnen las condiciones para ser reconocidas por el sistema han sido detectadas. Esto se ve reforzado por la idea de que se han obtenidos unos resultados satisfactorios en una primera iteración, habiéndose planteado ya las bases para una hipotética segunda, lo cual mejoraría significativamente el número de burbujas detectadas y reduciría los falsos positivos.

De cara a esta segunda versión de la estación de inspección de barnizado, se ha contemplado las siguientes modificaciones o mejoras:

- Mejora del sistema de iluminación: Gran parte de los problemas que aparecen a la hora de detectar las burbujas vienen ocasionados por las características de la iluminación, ya que ésta solo ilumina de forma directa las partes superior e inferior de los componentes, además de que cada barra tiene una potencia diferente. Esto hace que, a pesar del ajuste que se le realizó a una de ellas para homogeneizarlas, sigan existiendo pequeños desajustes entre una y otra, lo cual dificulta el desarrollo de un código robusto y homogéneo para el análisis de la PCB.

  Por ello, una de las líneas de actuación sería, o bien homogeneizar ambas iluminaciones e introducir otras dos barras laterales de las mismas características, o bien, crear un sistema de iluminación propio. Esta última idea surge a raíz del know-how que posee el GRAV e ISR en el desarrollo de éstos, ya que ha sido necesario para proyectos anteriores.

  Sea cual sea la opción elegida existe un criterio claro, la iluminación debe incidir de forma directa sobre el patillaje de cara a saturar esa zona y poder detectar fácilmente las burbujas.

- Empleo de una cámara lineal de mayor resolución: Otro de los problemas que presenta la versión actual del sistema es que las burbujas con un diámetro menor a un cierto tamaño no puede ser detectadas debido a que tan solo ocupan un área de un par de píxeles. Una solución a esto puede ser el empleo de una cámara lineal de mayor resolución; la actual tiene una de 4K, existiendo resoluciones de 8K y 16K para el mismo modelo.

  Esta modificación también ayudaría a mejorar la tasa de burbujas detectadas debido a que se obtendría una mejor definición de las imágenes.

- Cámara matricial: También se ha valorado el cambio de la cámara lineal por una matricial de alta resolución. Con esta nueva configuración y manteniendo un uso similar del actuador lineal, sería posible adquirir imágenes parciales de la PCB de cara a optimizar la resolución de las imágenes y analizarlas por separado. Además, eliminaría la necesidad de sincronización entre la velocidad de desplazamiento del actuador lineal y la frecuencia de adquisición de la cámara lineal.

- Mejora del código: Además de realizar mejoras en la parte hardware del prototipo, es necesario probar y desarrollar nuevos algoritmos de detección de burbujas.

  Con todo esto, sería posible la construcción de una nueva y mejorada versión de la estación prototipo de inspección de barnizado de PCB, en la cual se vería aumentada la tasa de detección de burbujas y reducido al mínimo el número de falsos positivos.

7. Presupuesto

Artículo	Descripción	Unid.	Precio	Subtotal
DAL-LA-GM-04K08A	Camara Linea 7.04um 4096x1-26kHz-Mono-CMOS-GigE	1	1.758,10	1.758,10
CEI-MV-1-1-3-5M	Cable Gige 5m 1Conec.RJ45Hori 1 Con.RJ45Normal-CableEstandar	1	64,19	64,19
INF-CAB-GPIO-LAG	Cable I/O DB15pin-9Hilos 3m para Linea GigE	1	59,00	59,00
EFF-FL05M-405-SD-P3	Iluminación	1	725,00	725,00
EFF-FL10-405-TR-P0	Ilum. LED Flood Lineal 215mm-90º-UV405-NoDiff	1	895,00	895,00
SVL-5PM12-10	Cable Power M12-5p/Hilos 10m para Iluminacion SVL	2	70,86	141,72
OPT-AZ-MF-2428	Optica AZ 24mm 43mm F2.8 F Ctor	1	171,60	171,60
OPT-AZ-ADFT34.5	Adaptador de M42 a F para Falcon2/Piranha4 2K/Linea	1	77,00	77,00
EDR-75-24	Fuente Alimentación 24V 3.2A	1	28,96	28,96
5009	Perfil básico 40x40 aluminio anodizado natural (m)	2,218	10,84	24,04
832-0223	Filamento para impresora 3D RS PRO Negro 2.85mm PLA (kg)	0,221	30,99	6,85
		Total		4093,18
			21% IVA	859,57
		Total Euros		4952,75

8. Bibliografía y referencias

[1] Medgyes, Bálint & Ripka, Gabor. (2007). Qualifying Methods of Conformal Coatings used on Assembled Printed Circuit Boards. 429 – 433.

[2] Kujawińska, Agnieszka & Vogt, Katarzyna. (2015). Human Factors in Visual Quality Control. Management and Production Engineering Review.

[3] Smith, C.J. & Adendorff, K. (2012). ADVANTAGES AND LIMITATIONS OF AN AUTOMATED VISUAL INSPECTION SYSTEM. The South African Journal of Industrial Engineering.

[4] HumiSeal. "*Conformal coatings facts & data handbook" https:// www.humiseal.com/.*

[5] Vision Engineering. *"Conformal Coating Inspection & Coating Faults"* http://www.visioneng.com/

[6] Autodesk. "*The history of PCB* " https://www.autodesk.com/products/eagle/blog/history-of-pcbs/

[7] Rapid PCB. "*History of printed circuit boards" http://www.rapidpcb.com/history-of-printed-circuit-boards.html/*

[8] Y. Hara, H. Doi, K. Karasaki and T. Iida. (1988) "A system for PCB automated inspection using fluorescent light," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 1. 69-78.

[9] Industrial Vision Systems. "*Automated Inspection Machine IVS-PCBi-C*" https://www.industrialvision.co.uk/uploads/products/PCB_ConformalCoatInspection/PCB%20Conformal%20Coat%20Inspection%20IVS-PCBi-C.pdf

[10] CyberOptics. "*Cx150i*" https://www.cyberoptics.com/automated-optical-inspection-cx150i-try-now.php

[11] Nordson Yestech "FX-UV AOI" https://www.nordson.com/en/divisions/yestech/products/conformal-coating-equipment/fx-uv-aoi.

[12] Heibus Project. http://www.heibus.eu/

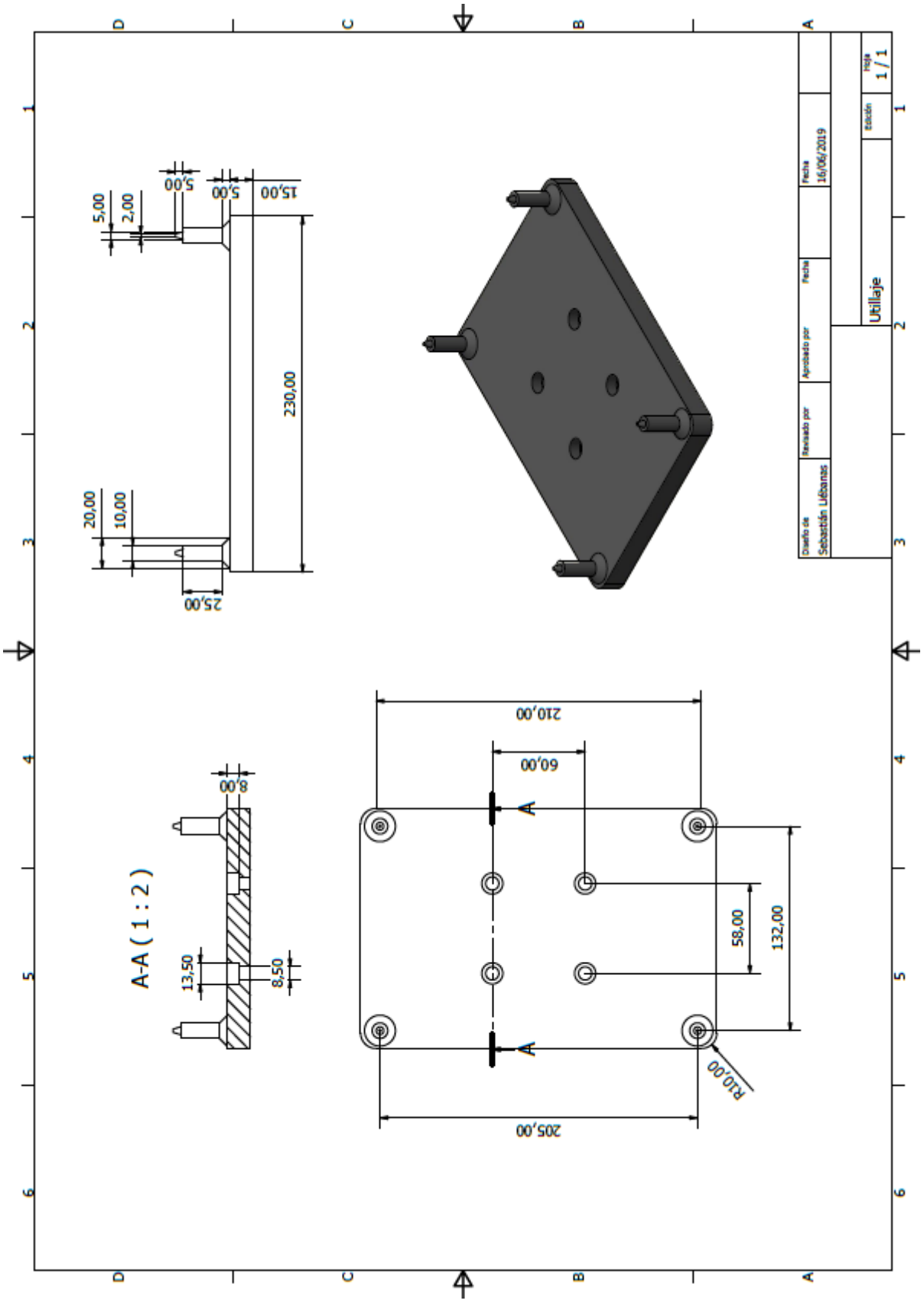
[13] Effilux "Effi-flex: Barra de LED con haz regulable" http://www.effilux.fr/es/products/led-bar/effi-flex/

[14] Teledyne Dalsa "Linea" https://www.teledynedalsa.com/en/products/imaging/cameras/linea/

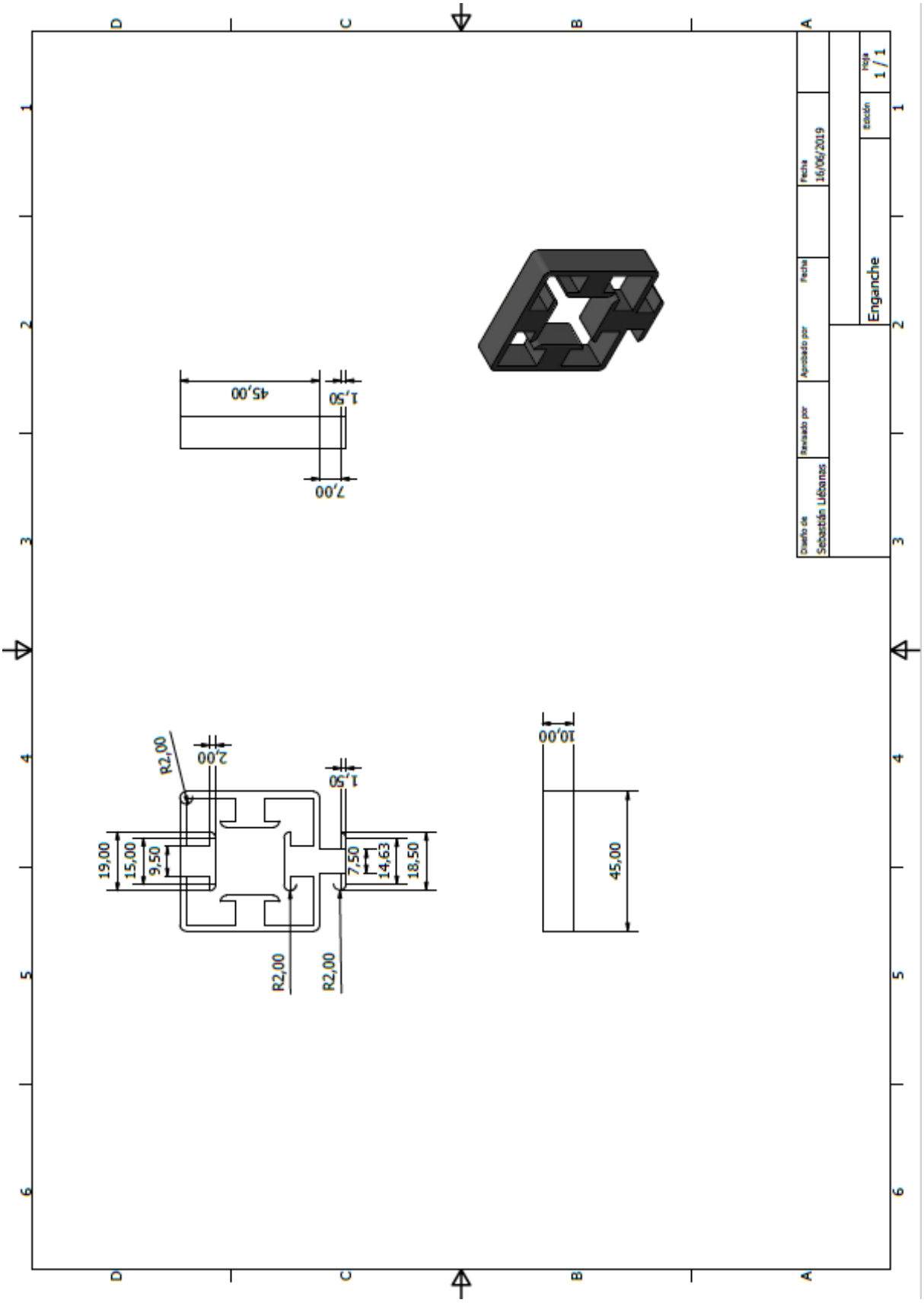
[15] SMC. “LEFS, Actuador eléctrico”.https://www.smc.eu/portal_ssl/WebContent/digital_catalog_2/jsp/view_product_configurator.jsp?dc_product_id=133857

9. Anexo I: Planos

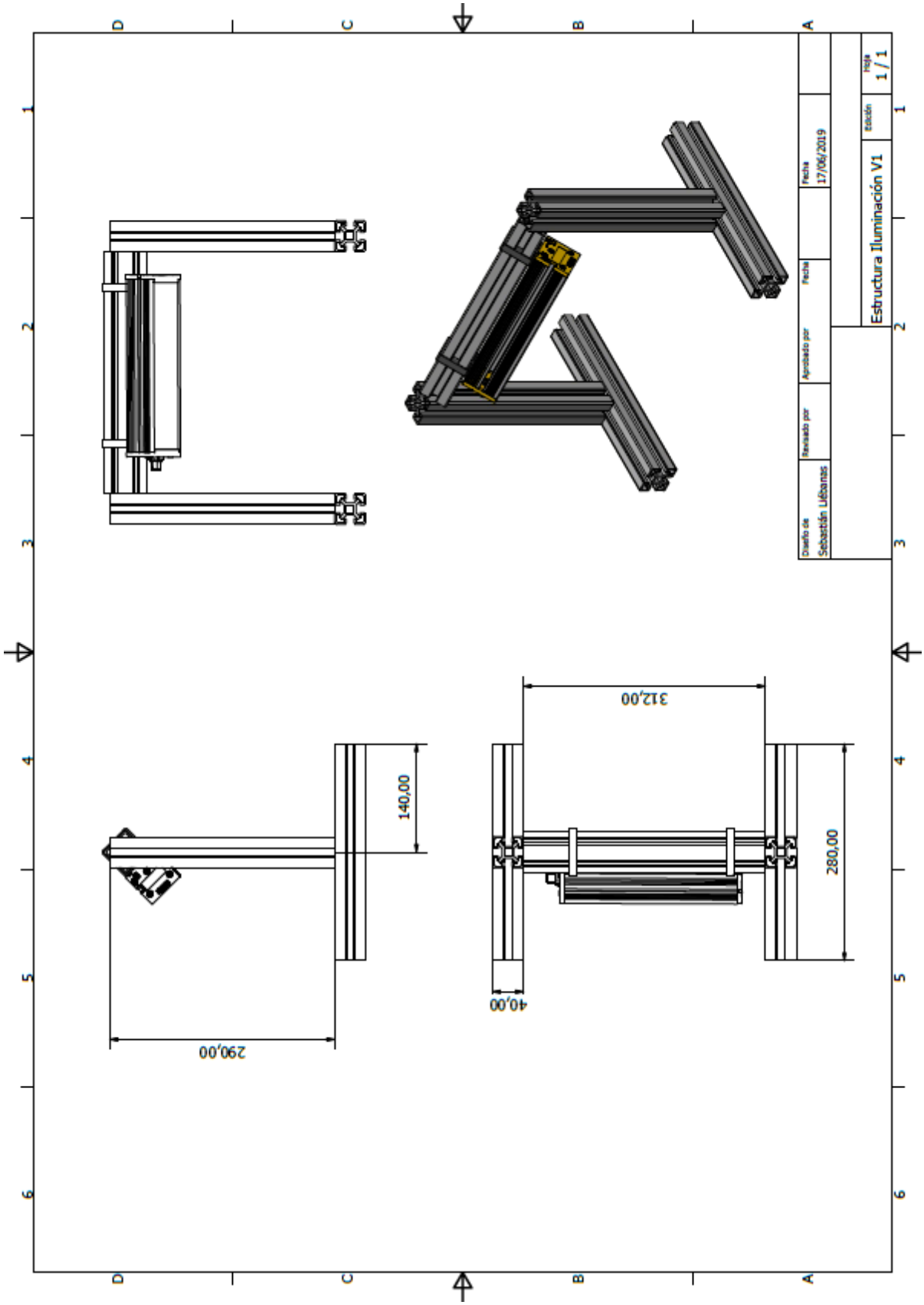
9.1. Plano Utillaje



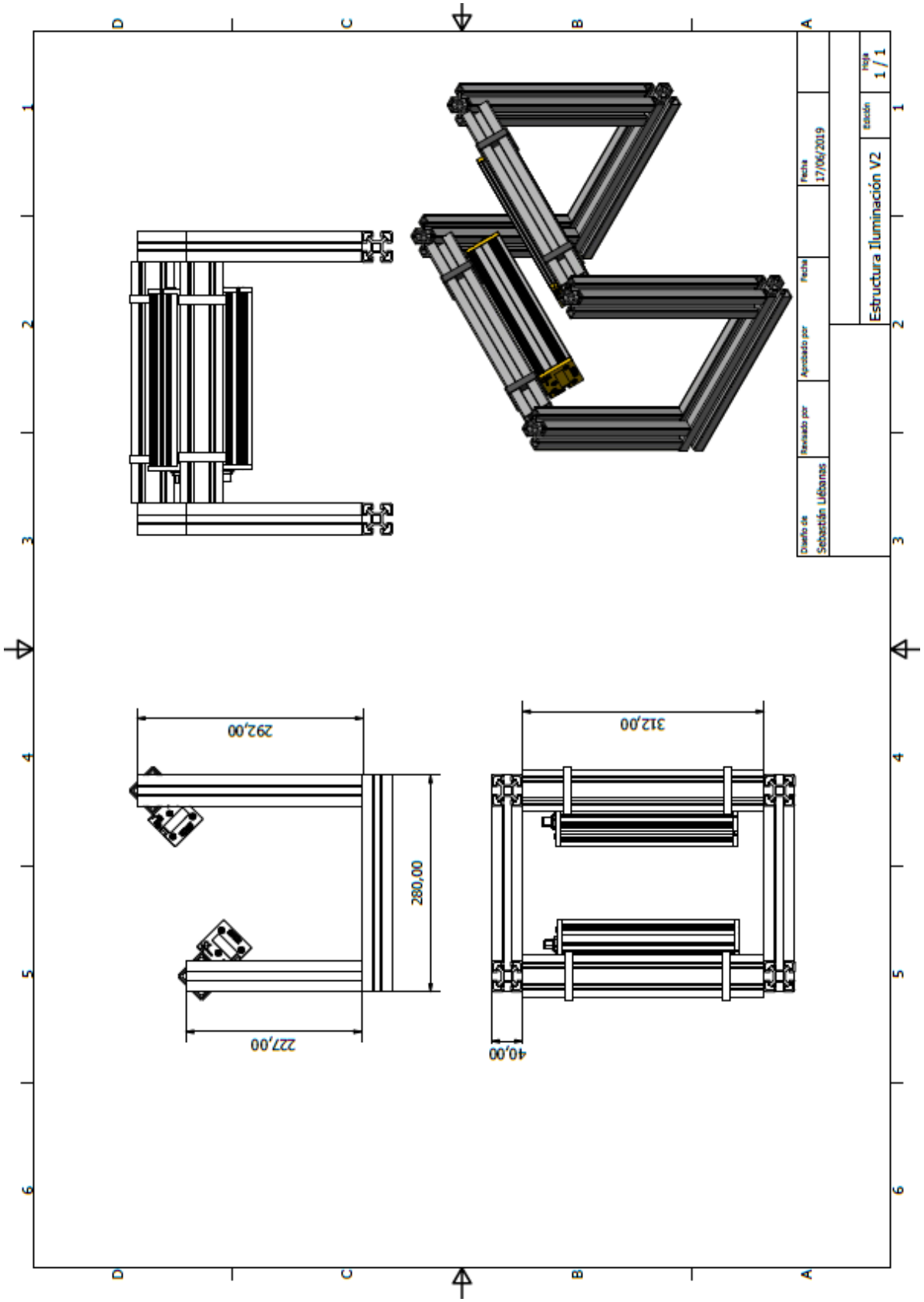
9.2. Plano Enganche



9.3. Plano Estructura Iluminación V1



9.4. Plano Estructura Iluminación V2



10. Anexo II: Código Software desarrollado

Está compuesto por el programa principal (Ej.m y Ej.fig) y por la función ProcBurb (ProcBurb.m), utilizada para la detección de burbujas una por una en los componentes 2 al 7.

10.1. Programa principal (Ej.m)

```
%%%%%%%%%%%%%%%%%%%   PROGRAMA PRINCIPAL  %%%%%%%%%%%%%%%%%%%
function varargout = Ej(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Ej_OpeningFcn, ...
                   'gui_OutputFcn',  @Ej_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                      'gui_Callback',   []);
  if nargin && ischar(varargin{1})
      gui_State.gui_Callback = str2func(varargin{1});
  end

  if nargout
      [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
  else
      gui_mainfcn(gui_State, varargin{:});
  end
  function Ej_OpeningFcn(hObject, eventdata, handles, varargin)
  handles.output = hObject;
  guidata(hObject, handles);
  axes(handles.axesIm);imshow('indice.jpg');
  function varargout = Ej_OutputFcn(hObject, eventdata, handles)
  varargout{1} = handles.output;
  function edit1_CreateFcn(hObject, eventdata, handles)
  if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
  end
  function edit2_CreateFcn(hObject, eventdata, handles)
  if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
  end
  function edit3_CreateFcn(hObject, eventdata, handles)
  if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
  end
  function edit4_CreateFcn(hObject, eventdata, handles)
  if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
  end
  function edit5_CreateFcn(hObject, eventdata, handles)

```

```
44.   if ispc && isequal(get(hObject,'BackgroundColor'),
   get(0,'defaultUicontrolBackgroundColor'))
45.       set(hObject,'BackgroundColor','white');
46.   end
47.   function edit6_CreateFcn(hObject, eventdata, handles)
48.   if ispc && isequal(get(hObject,'BackgroundColor'),
   get(0,'defaultUicontrolBackgroundColor'))
49.       set(hObject,'BackgroundColor','white');
50.   end
51.   function edit7_CreateFcn(hObject, eventdata, handles)
52.
53.   if ispc && isequal(get(hObject,'BackgroundColor'),
   get(0,'defaultUicontrolBackgroundColor'))
54.       set(hObject,'BackgroundColor','white');
55.   end
56.   function editID_CreateFcn(hObject, eventdata, handles)
57.   if ispc && isequal(get(hObject,'BackgroundColor'),
   get(0,'defaultUicontrolBackgroundColor'))
58.       set(hObject,'BackgroundColor','white');
59.   end
60.   function OpenImage_Callback(hObject, eventdata, handles)
61.
62.   %Lee la imagen OK del componente 1
63.   im=imread('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\IlumAjust\10-1.bmp');
64.
65.   %Lee la imagen a evaluar y la muestra en la GUI
66.   [nim,path]=uigetfile({'*.bmp'},'Abrir imagen evaluada');
67.   set(handles.editID,'String',nim);
68.   preim=strcat(path,nim);
69.   if nim~=0
70.   im1=imread(preim);
71.   axes(handles.axesMain);imshow(im1);
72.
73.   %Pone a 0 todo lo que no es la zona del círculo de referencia
74.   c = [3850 4096 4096 3850];
75.   r = [400 400 1200 1200];
76.   BW = roipoly(im,c,r);
77.   BW=im2uint8(BW);
78.   BW=BW*(1/255);
79.   r=im.*BW;
80.   r1=im1.*BW;
81.
82.   %Halla círculo referencia y centroide imagen 1
83.   r=im2bw(r,0.2);
84.   [r,num]=bwlabel(r);
85.   cent=regionprops(r,'Centroid');
86.   area=regionprops(r,'Area');
87.   per=regionprops(r,'Perimeter');
88.
89.   %Halla círculo referencia y centroide imagen 2
90.   r1=im2bw(r1,0.2);
91.   [r1,num1]=bwlabel(r1);
92.   cent1=regionprops(r1,'Centroid');
```

```
93.   area1=regionprops(r1,'Area');
94.   per1=regionprops(r1,'Perimeter');
95.
96.   %Comprueba las figuras encontradas imagen 1
97.   for i=1:num
98.
   circ(i)=((per(i).Perimeter(1)*per(i).Perimeter(1))/area(i).Area(1));
99.       if(circ(i)>12.5 && circ(i)<31 && area(i).Area(1)>5900 &&
   area(i).Area(1)<11000)
100.          x0=cent(i).Centroid(1);
101.          y0=cent(i).Centroid(2);
102.      end
103. end
104.
105. %Comprueba las figuras encontradas imagen 2
106. for i=1:num1
107.
   circ1(i)=((per1(i).Perimeter(1)*per1(i).Perimeter(1))/area1(i).Area(1));
108.      if(circ1(i)>12.5 && circ1(i)<31 && area1(i).Area(1)>5900 &&
   area1(i).Area(1)<11000)
109.          x1=cent1(i).Centroid(1);
110.          y1=cent1(i).Centroid(2);
111.      end
112. end
113.
114. %Halla el tamaño de imagen 1 y pasa a Int el Centroide 1
115. [m,n]=size(im);
116. x0=int16(x0);
117. y0=int16(y0);
118.
119. %Halla el tamaño de imagen 2 y pasa a Int el Centroide 2
120. [m1,n1]=size(im1);
121. x1=int16(x1);
122. y1=int16(y1);
123.
124. set(handles.edit1,'UserData',n1);
125. set(handles.edit2,'UserData',x1);
126. set(handles.edit3,'UserData',y1);
127.
128. %Sitúa el origen de ambas imágenes en el mismo punto
129. imnew=circshift(im,[-y0,n-x0]);
130. imnew1=circshift(im1,[-y1,n1-x1]);
131.
132. %Rectángulo para seleccionar ROI (Zona barniz)
133. c = [900 n n 900];
134. r = [0 0 3400 3400];
135. BW = roipoly(im,c,r);
136. BW=im2uint8(BW);
137. BW=BW*(1/255);
138. imnew=imnew.*BW;
139. imnew1=imnew1.*BW;
140. imwrite(imnew,'1.bmp');
141. imwrite(imnew1,'2.bmp');
142.
```

```
143. %Selección Componente1-Imagen1
144.     im1c1=imcrop(imnew,[3435 730 195 195]);
145.     set(handles.axesMain,'UserData',im1c1);
146.
147. %Selección Componente1-Imagen2
148.     im2c1=imcrop(imnew1,[3435 730 195 195]);
149.     axes(handles.axes1);cla;imshow(im2c1);
150.     set(handles.axes1,'UserData',im2c1);
151.     path=strcat('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\1\',nim);
152.     imwrite(im2c1,path);
153.
154. %Selección Componente2
155.     imc2=imcrop(imnew1,[1860 860 110 120]);
156.     axes(handles.axes2);cla;imshow(imc2);
157.     set(handles.axes2,'UserData',imc2);
158.     path=strcat('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\2\',nim);
159.     imwrite(imc2,path);
160.
161. %Selección Componente3
162.     imc3=imcrop(imnew1,[1860 1010 110 120]);
163.     axes(handles.axes3);cla;imshow(imc3);
164.     set(handles.axes3,'UserData',imc3);
165.     path=strcat('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\3\',nim);
166.     imwrite(imc3,path);
167.
168. %Selección Componente4
169.     imc4=imcrop(imnew1,[1825 1145 110 120]);
170.     axes(handles.axes4);cla;imshow(imc4);
171.     set(handles.axes4,'UserData',imc4);
172.     path=strcat('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\4\',nim);
173.     imwrite(imc4,path);
174.
175. %Selección Componente5
176.     imc5=imcrop(imnew1,[1825 1305 110 120]);
177.     axes(handles.axes5);cla;imshow(imc5);
178.     set(handles.axes5,'UserData',imc5);
179.     path=strcat('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\5\',nim);
180.     imwrite(imc5,path);
181.
182. %Selección Componente6
183.     imc6=imcrop(imnew1,[1985 1220 110 120]);
184.     axes(handles.axes6);cla;imshow(imc6);
185.     set(handles.axes6,'UserData',imc6);
186.     path=strcat('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\6\',nim);
187.     imwrite(imc6,path);
188.
189. %Selección Componente7
190.     imc7=imcrop(imnew1,[1825 1470 110 120]);
```

```
191.      axes(handles.axes7);cla;imshow(imc7);
192.      set(handles.axes7,'UserData',imc7);
193.      path=strcat('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\7\',nim);
194.      imwrite(imc7,path);
195.  end
196.  function Process_Callback(hObject, eventdata, handles)
197.
198.  %Cargar datos necesarios
199.  im1c1=get(handles.axesMain,'UserData');
200.  im2c1=get(handles.axes1,'UserData');
201.  imc2=get(handles.axes2,'UserData');
202.  imc3=get(handles.axes3,'UserData');
203.  imc4=get(handles.axes4,'UserData');
204.  imc5=get(handles.axes5,'UserData');
205.  imc6=get(handles.axes6,'UserData');
206.  imc7=get(handles.axes7,'UserData');
207.
208.  n1=get(handles.edit1,'UserData');
209.  x1=get(handles.edit2,'UserData');
210.  y1=get(handles.edit3,'UserData');
211.
212.  %Procesado Componente1
213.      axes(handles.axes1);
214.      n=corr2(im1c1,im2c1);
215.
216.      imko1=imread('C:\Users\usuario\Google Drive\1-Portatil\TFG\2-
   Visión\Matlab\ProgramaNuevo\Componentes1\1\2.bmp');
217.      nx=corr2(imko1,im2c1);
218.
219.      if n>0.34 || nx<0.42
220.          set(handles.edit1,'String','OK');
221.          c1='g';
222.      else
223.          set(handles.edit1,'String','KO');
224.          c1='r';
225.      end
226.  %Procesado Componente2
227.      axes(handles.axes2);
228.      [n2,c2]=ProcBurb(imc2);
229.      if n2==0
230.          set(handles.edit2,'String','OK');
231.      else
232.          set(handles.edit2,'String','KO');
233.      end
234.  %Procesado Componentes3
235.      axes(handles.axes3);
236.      [n3,c3]=ProcBurb(imc3);
237.      if n3==0
238.          set(handles.edit3,'String','OK');
239.      else
240.          set(handles.edit3,'String','KO');
241.      end
```

```
%Procesado Componente4
    axes(handles.axes4);
    [n4,c4]=ProcBurb(imc4);
    if n4==0
        set(handles.edit4,'String','OK');
    else
        set(handles.edit4,'String','KO');
    end
%Procesado Componente5
    axes(handles.axes5);
    [n5,c5]=ProcBurb(imc5);
    if n5==0
        set(handles.edit5,'String','OK');
    else
        set(handles.edit5,'String','KO');
    end
%Procesado Componente6
    axes(handles.axes6);
    [n6,c6]=ProcBurb(imc6);
    if n6==0
        set(handles.edit6,'String','OK');
    else
        set(handles.edit6,'String','KO');
    end
%Procesado Componente7
    axes(handles.axes7);
    [n7,c7]=ProcBurb(imc7);
    if n7==0
        set(handles.edit7,'String','OK');
    else
        set(handles.edit7,'String','KO');
    end


%Dibujo recuadro componente
axes(handles.axesMain);
rectangle('Position',[(3435-(n1-x1)) 730+y1 195 195],'EdgeColor',c1);
rectangle('Position',[(1870-(n1-x1)) 870+y1 90 100],'EdgeColor',c2);
rectangle('Position',[(1870-(n1-x1)) 1020+y1 90 100],'EdgeColor',c3);
rectangle('Position',[(1835-(n1-x1)) 1155+y1 90 100],'EdgeColor',c4);
rectangle('Position',[(1835-(n1-x1)) 1315+y1 90 100],'EdgeColor',c5);
rectangle('Position',[(1995-(n1-x1)) 1230+y1 90 100],'EdgeColor',c6);
rectangle('Position',[(1835-(n1-x1)) 1480+y1 90 100],'EdgeColor',c7);
```

10.2. Función ProcBurb (ProcBurb.m)

```
%%%%%%%% Código para encontrar burbujas en los componentes 2-7 %%%%%%%%
function [b,color] = funcion(imc4)
    b=0;
    color='g';
    %%%%%Para detectar borde componente%%%%%%%
    imc4=adapthisteq(imc4);
    se=strel('disk',4);
    imc4=imsubtract(imadd(imc4,imtophat(imc4,se)),imbothat(imc4,se));
    imc4=imcomplement(imc4); %% Aquí imc4 puede utilizarse para después hallar las burbujas
      imc4o=imc4;
      imc4=im2bw(imc4,0.85);
      imc4=bwpropfilt(imc4,'Area',[500,2000]);
      [imc4,num]=bwlabel(imc4);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %%%%%%%Para poner a negro el cuadrado y zonas de no interes%%%%%%%%
      [m,n]=size(imc4);
      box=regionprops(imc4,'BoundingBox');
      a(:,:)=box(1).BoundingBox(1,:);
      imc4o(a(2):a(2)+a(4),a(1):a(1)+a(3))=0;
      imc4o(a(2)+a(4)+15:m,:)=0;
      imc4o(1:a(2)-15,:)=0;
      imc4o(:,1:a(1)-15)=0;
      imc4o(:,a(1)+a(3)+15:n)=0;
      imc4o(a(2)+a(4):m,1:a(1)-10)=0;
      imc4o(a(2)+a(4):m,a(1)+a(3)+10:n)=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %%%%%%%Para detectar burbuja%%%%%%%%

      imc4o=imc4o==255;
      imc4o=bwpropfilt(imc4o,'Area',[6,500]);
      [imc4o,num]=bwlabel(imc4o);
      cent=regionprops(imc4o,'Centroid');
      area=regionprops(imc4o,'Area');
      per=regionprops(imc4o,'Perimeter');
      if num>0
          for i=1:num
circ(i)=((per(i).Perimeter(1)*per(i).Perimeter(1))/area(i).Area(1));
              if(((circ(i)>1 && circ(i)<13.1)&&cent(i).Centroid(2)>5) || (area(i).Area(1)>20 && area(i).Area(1)<40)) && cent(i).Centroid(2)>5 %%Antes 12.7
                  x0=cent(i).Centroid(1);
                  y0=cent(i).Centroid(2);
                  viscircles([x0 y0],5);
                  b=b+1;
                  color='r';
              end
```

```
47.          end
48.      end
49.  end
```

11. Anexo III: Código Mejoras de procesado en curso

Está compuesto por el programa principal (Diferencias.m) y por la función DComp (DComp.m), utilizada para la detección del encapsulado del componente.

11.1. Programa principal (Diferencias.m)

```
1. %%%%%%%% Código para crear imagen en diferencias de dos componentes
   %%%%%%
2. clear all;
3. close all;
4. %Lectura de componentes
5. [nim,path]=uigetfile({'*.bmp'},'Abrir imagen evaluada');
6. preim=strcat(path,nim);
7. if nim~=0
8. im1=imread(preim);
9.
10.
11.   [nim,path]=uigetfile({'*.bmp'},'Abrir imagen evaluada');
12.   preim=strcat(path,nim);
13.   if nim~=0
14.   im2=imread(preim);
15.   [m,n]=size(im1);
16.
17.   %Cálculo coordenadas esquina superior izquerda
18.   [a]=DComp(im1);
19.   [b]=DComp(im2);
20.
21.   %Desplzamiento y redimensionamiento de la imagen a evaluar
22.   im2=circshift(im2,[a(2)-b(2),a(1)-b(1)]);
23.   c1=round((a(4)/b(4))*m);
24.   c2=round((a(3)/b(3))*n);
25.   im2=imresize(im2,[c1 c2]);
26.   if m<c1
27.       for i=m:c1
28.           im1(i,:)=0;
29.       end
30.   end
31.   if m>c1
32.       for i=c1:m
33.           im2(i,:)=0;
34.       end
35.   end
36.   if n<c2
37.       for i=n:c2
38.           im1(:,i)=0;
39.       end
40.   end
41.   if n>c2
42.       for i=c2:n
```

```
43.           im2(:,i)=0;
44.       end
45.   end
46.
47.   %Imagen en diferencias
48.   figure;
49.   imdif=(im1-im2);
50.   imshow(imdif);
51.   end
52.   end
```

11.2. Función DComp (DComp.m)

```
1. function [a] = DComp(imin)
2. im1=imin;
3. im1=adapthisteq(im1);
4. se=strel('disk',4);
5. im1=imsubtract(imadd(im1,imtophat(im1,se)),imbothat(im1,se));
6. im1=imcomplement(im1);
7. im1x=im1;
8. im1=im2bw(im1,0.85);
9. im1=bwpropfilt(im1,'Area',[500,2000]);
10.   [im1,num]=bwlabel(im1);
11.   box=regionprops(im1,'BoundingBox');
12.   a(:,:)=box(1).BoundingBox(1,:);
13.   [m,n]=size(im1);
14.   end
```