UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Gamir Artesero, Javier

Tutor/a: Ricolfe Viala, Carlos

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Autor

Javier Gamir Artesero

Tutor

Carlos Ricolfe Viala

CURSO ACADÉMICO: 2022/2023

Agradecimientos

A mis padres y mi hermano por siempre apoyarme en todo lo que hago y su intento de comprender lo que han sido todos estos meses de trabajo.

A mi tutor, por permitirme llevar a cabo un proyecto que me hacía mucha ilusión y por toda su ayuda, apoyo y sobre todo paciencia.

Y a mis compañeros, que gracias a ellos he podido disfrutar de los mejores años de mi vida, en especial a Abel Górriz y Marc Fontalba con los que he tenido el placer de trabajar y en los que he encontrado las mismas ganas que yo de hacer las cosas siempre lo mejor posible.

Índice General

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

DOCUMENTO Nº1: MEMORIA

Grado en Ingeniería Electrónica Industrial y Automática

Autor

Javier Gamir Artesero

Tutor

Carlos Ricolfe Viala

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Índice

Tabla de figuras

1. Objeto

1.1 Objeto

Este proyecto tiene como objeto desarrollar un sistema de empaquetado de productos mediante un robot antropomórfico y visión artificial. El objetivo principal es optimizar el proceso de empaquetado de productos en una línea de producción, aumentando la eficiencia y la precisión del proceso mediante la utilización de tecnología robótica y de visión artificial, permitiendo además una mayor flexibilidad a la hora de elegir donde empaquetar los productos.

En este proyecto se llevará a cabo la programación de uno de los robots antropomórficos disponibles en el laboratorio de robótica para que realice las operaciones de empaquetado de forma autónoma, utilizando la información proporcionada por el sistema de visión artificial y herramientas externas como la cinta transportadora.

Se desarrollará también un sistema de visión artificial para detectar la posición y orientación de los productos en la línea de producción, y proporcionar al robot la información necesaria para que realice la paletización con la mayor precisión posible.

Finalmente, en este proyecto se estudiarán las diferentes tecnologías de cámaras disponibles en el mercado para la detección de objetos en una línea de producción. Se seleccionará la cámara que mejor se adapte a las necesidades del caso particular, teniendo en cuenta factores como la resolución de imagen, el precio y el tamaño.

1.2 Parámetros del sistema

El sistema cuenta con dos herramientas gracias al espacio donde se encuentra, estas son una cinta transportadora y un sensor foto eléctrico de llegada al fin de la cinta, los cuáles permitirán el movimiento de los objetos y la detección de estos al llegar a cierta posición.

También será necesario disponer de un espacio de trabajo donde empaquetar los objetos y un método para especificarle al sistema dónde queremos que empaquete cada objeto.

2. Antecedentes

2.1 Introducción

En la actualidad, la visión artificial se está convirtiendo en una tecnología cada vez más utilizada por las empresas para automatizar procesos y mejorar la eficiencia en la producción. Esta tecnología permite a las máquinas "ver" y entender el entorno que las rodea, lo que les permite realizar tareas de manera autónoma y precisa.

La visión artificial se está aplicando en diferentes sectores, como la industria automotriz, la alimentaria, la farmacéutica, entre otras. En estos sectores, los robots equipados con sistemas de visión artificial son capaces de realizar tareas como el ensamblaje de piezas, la clasificación de productos o la inspección de calidad, entre otras.

La combinación de la visión artificial y la robótica ha permitido el desarrollo de sistemas más avanzados de automatización industrial. Los robots equipados con sistemas de visión artificial pueden detectar y reconocer objetos y patrones complejos, lo que les permite realizar tareas con mayor precisión y rapidez.

2.2 Motivo de Diseño

La razón que impulsa este proyecto es la creciente demanda por parte de los distintos sectores de la industria de la automatización de procesos que permitan reducir el efecto del factor humano en los procesos de producción, lo que se traduce en una reducción de costos y un aumento de la eficiencia.

De forma más concreta este proyecto busca agilizar y automatizar un proceso con productos variables del sector alimenticio que además permita cierta flexibilidad, demostrando la capacidad de estos sistemas en entornos cerrados y la creciente industria 4.0.

2.3 Historia

La historia de la robótica se remonta a la década de 1950, cuando se comenzó a explorar la posibilidad de crear máquinas capaces de realizar tareas de forma autónoma; el origen de la palabra 'robot' proviene de la palabra checa 'robota', que se refiere al trabajo realizado de manera forzada.

En la época de los 50, George Devol patentó el primer manipulador programable y junto a Joseph F. Engelberger, fundó Unimation, la primera empresa de robótica de la historia. Los robots eran controlados mediante dispositivos mecánicos y electrónicos y se limitaban a realizar tareas simples y repetitivas en la industria.

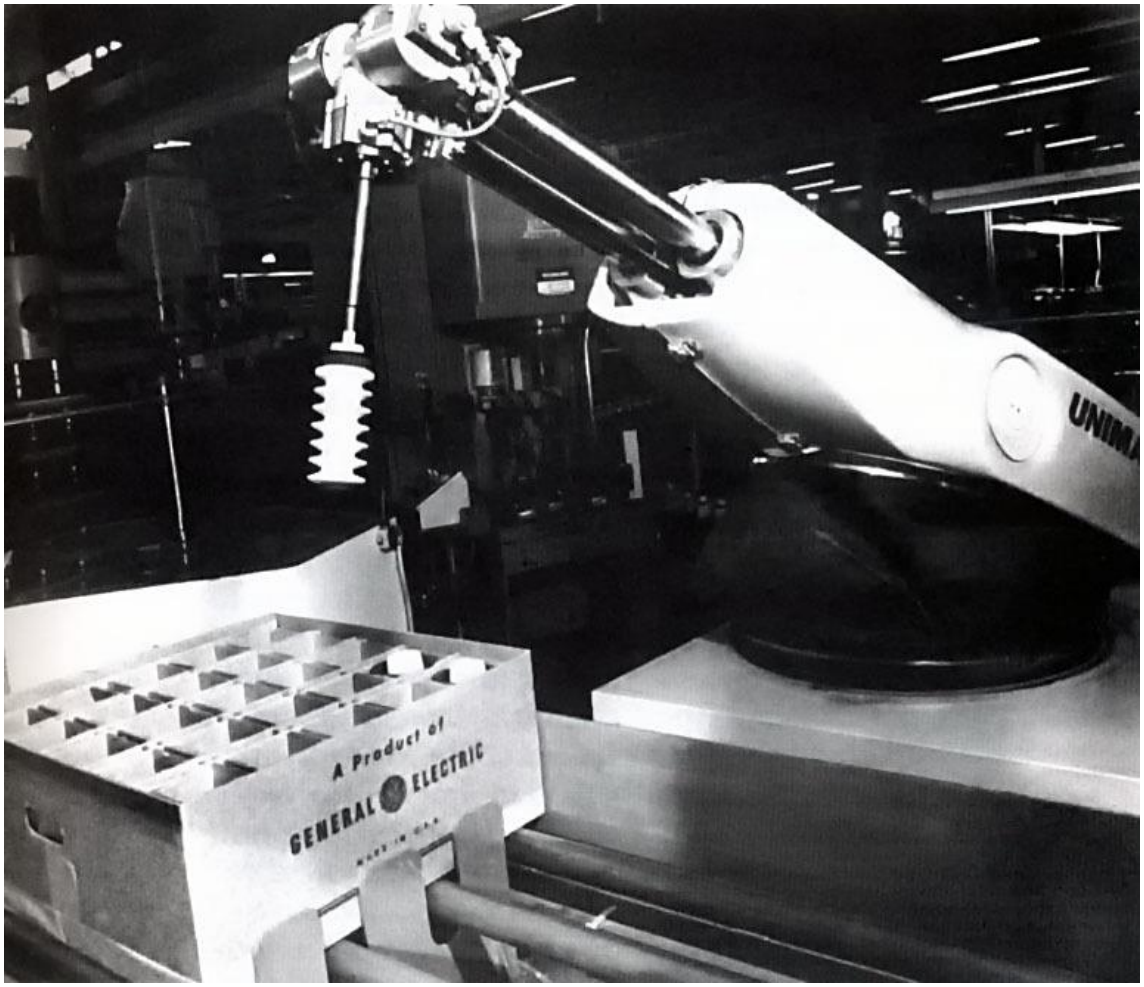


Figura 2.1 Robot Unimate (1948)

Fue en la década de 1960 cuando comenzó a surgir la idea de incorporar sistemas de visión artificial en los robots con la finalidad de imitar el sistema de visión humano y dotarles de grandes entradas de información, y con ello, entender el entorno que los rodea. En esta década se desarrollaron los primeros sistemas de visión artificial, basados principalmente en cámaras analógicas y sistemas de procesamiento de imágenes muy básicos.

En la década de 1970, se produjo un gran avance en la robótica con el surgimiento de los primeros robots industriales programables. Estos robots permitieron realizar tareas más complejas y precisas, y se convirtieron en una herramienta fundamental para la automatización de procesos en la industria. De forma similar se produjeron los primeros estudios en visión artificial que formaron los cimientos sobre los que se basa la misma hoy en día como la extracción de bordes de imágenes, el etiquetado de líneas, flujo óptico y estimación del movimiento.

En la década de 1980, se produjo un aumento de hasta el 80% de la producción y venta de robots para la industria y comenzaron a crearse en Europa y Asia. También se produjo otro avance significativo en la visión artificial con el desarrollo de los primeros sistemas de visión artificial basados en computadoras los cuáles permitieron procesar imágenes en tiempo real y llevar a cabo tareas más complejas, como la detección de objetos y la clasificación de imágenes.

En la década de 1990, la robótica y la visión artificial comenzaron a combinarse de manera más estrecha, lo que permitió el desarrollo de sistemas más avanzados de automatización industrial. Los robots equipados con sistemas de visión artificial se convirtieron en una herramienta cada vez más importante en la industria, permitiendo la realización de tareas complejas y precisas en una amplia variedad de sectores.

Figura 2.2 Robot colaborativo con cámara

2.4 Actualidad

2.4.1 Robótica

La robótica se encuentra ampliamente utilizada en múltiples sectores hoy en día, como es el sector automovilístico, la industria, la seguridad, la medicina, los hogares, la minería, la construcción, la agricultura, el espacio y múltiples más.

Aunque los robots se podrían considerar el eje principal de la fabricación competitiva de hoy en día, sobre todo en el montaje de automóviles y componentes relacionados, sigue habiendo retos que resolver para que la fabricación responda eficazmente a los cambios en el comportamiento de los consumidores y a los cambios globales en la competitividad.

La integración de varios tipos de controles (controladores lógicos programables (PLC), sensores de control numérico por ordenador (CNC)) con el controlador del robot han permitido evolucionar al sector permitiendo desde la proximidad entre el robot y el ser humano (robots colaborativos) junto a la

producción sin necesidad de limitación con vallas por seguridad hasta el ahorro energético.

Dentro de la robótica podemos encontrar diversas funciones, entre las que se destacan:

Manipulación

La manipulación en robótica comprende numerosos procesos como agarrar, transportar, empaquetar, paletizar y recogida. Una característica central y un reto importante en la ingeniería de los sistemas robóticos de manipulación es el diseño de la pinza y las estrategias de agarre asociadas dadas las propiedades físicas de la pieza de trabajo, los rendimientos y las incertidumbres relativas a la geometría y la ubicación del objeto.

Figura 2.3 Robot Manipulador con herramienta ventosa

Manipulación de alimentos

El sector alimenticio tiene un potencial significativo para la aplicación de robots, ya que se pueden lograr cambios fundamentales en la productividad, la calidad de los productos y la ergonomía de los trabajadores. En la automatización de los procesos dentro de este sector conlleva críticas demandas en cuanto a higiene, diseño, velocidad de operación, coste y facilidad de programación.

Es necesaria una alta velocidad y flexibilidad debido a la variabilidad de los productos intraclase e interclase, es por esto por lo que los robots SCARA y paralelos son los más reconocibles dentro del sector, sin embargo, esto no implica que no existan robots antropomórficos dentro del mismo. Además, dentro de este sector es muy común encontrar aplicaciones de visión artificial junto con los robots, como se comentará más adelante.

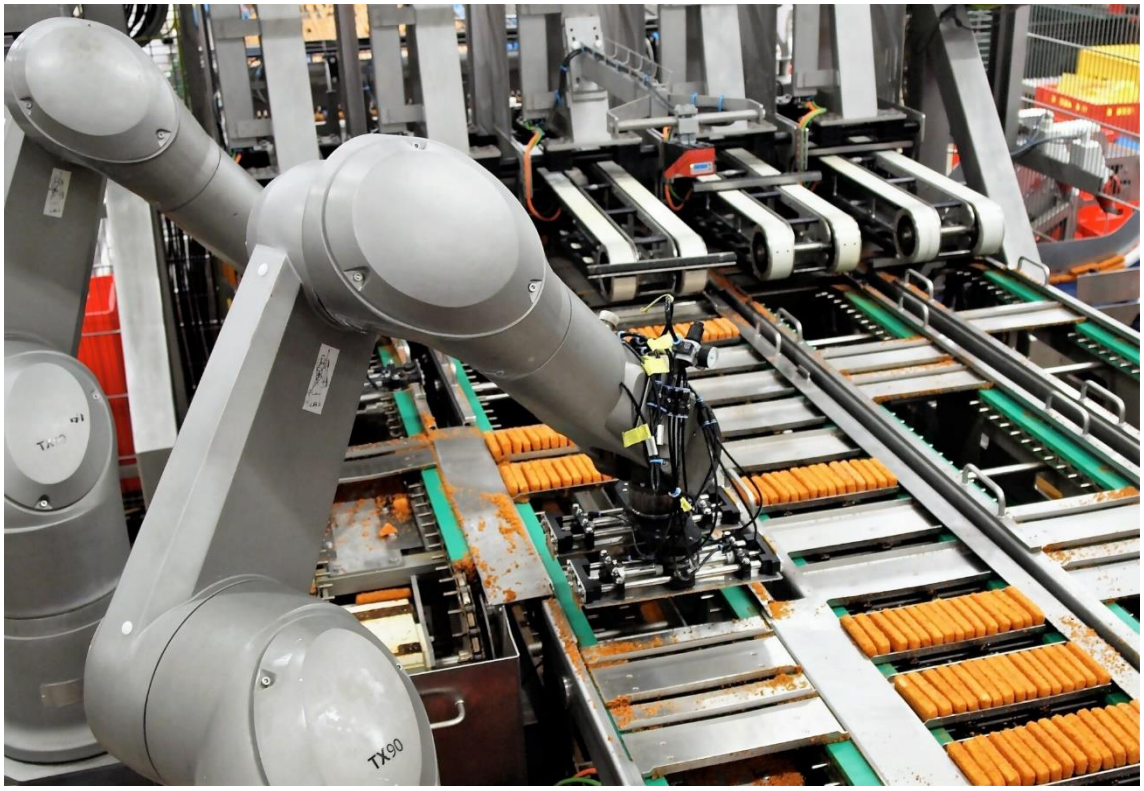


Figura 2.4 Robot Antropomórfico manipulador de alimentos

Recogida de Objetos

Generalmente, la práctica industrial en la planificación de la célula de trabajo del robot tiene como objetivo encontrar un compromiso entre la reducción de la variación de la ubicación de la pieza de trabajo y el coste de los sistemas de sensores para compensar incertidumbres o variaciones residuales.

Actualmente, casi todas las piezas llegan a robot de forma consistente, ya sea almacenadas en soportes o delimitadores, o siendo transportadas y orientadas mediante dispositivos vibratorios que permiten a las piezas colocarse en una orientación predecible para que el robot las agarre correctamente.

Sin embargo, los requisitos de coste y flexibilidad en la automatización de la fabricación provocan que se reduzcan los almacenes de piezas personalizados a soportes más universales, contenedores o cintas transportadoras, es por esto, que, si los productos son orientados de forma aleatoria mediante una cinta, han de ser identificados y localizados de forma correcta para que el robot no produzca colisiones ni produzca daños al objeto.

El desafío de agarrar piezas parcial o aleatoriamente ordenadas por un robot se ha denominado *binpicking* y ha sido investigado por numerosos investigadores desde mediados de los años 80. Aunque se han presentado numerosos enfoques, sólo recientemente las instalaciones de *binpicking* han encontrado su cabida en procesos manufactureros diarios.

Los algoritmos de recogida de objetos siguen una secuencia típica de pasos: adquisición inicial de datos de nubes de puntos, detección de objetos, estimación de la pose, planificación de la trayectoria y el agarre sin colisiones, agarre del objeto y colocación del objeto.

La identificación del objeto puede ser realizada de distintas formas, la más destacada hoy en día es la visión artificial.

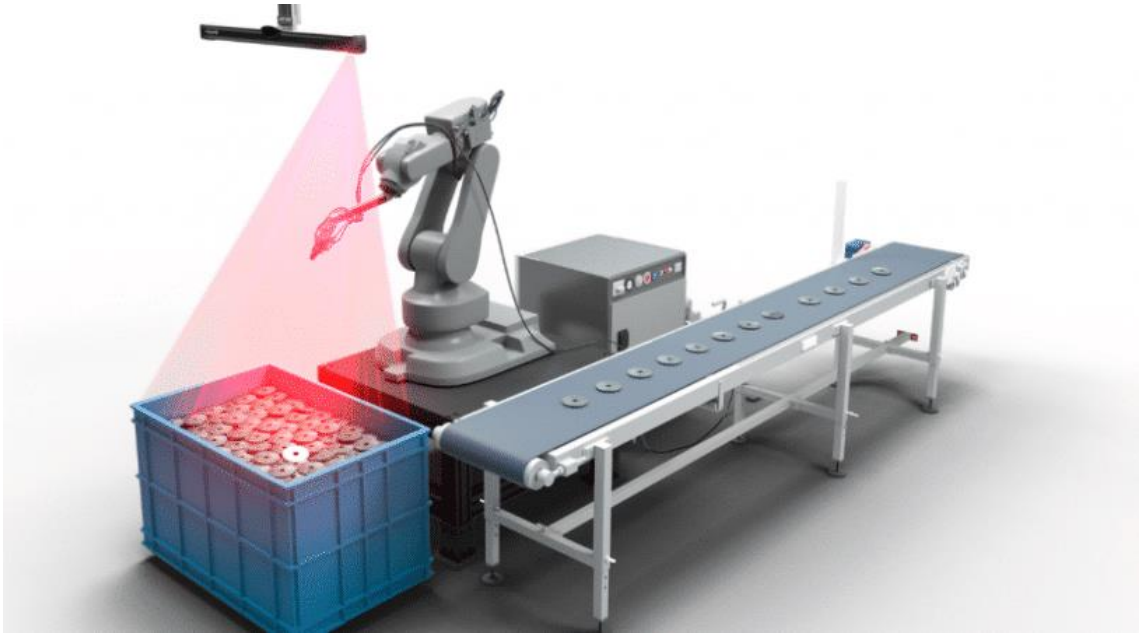


Figura 2.5 Robot Antropomórfico "Binpicking"

Soldadura

La soldadura es un proceso de fabricación que une materiales aplicando calor, a veces con presión. Normalmente el material de la pieza de trabajo se funde en el lugar del proceso, a menudo con material de relleno adicional. Los procesos en los que solemos encontrar a los robots son los procesos de soldadura por puntos, sobre todo en el montaje de carrocerías de automóviles y la soldadura por arco metálico con gas de protección (GMAW). Con el aumento de la compacidad de las fuentes láser y la precisión de movimiento de los robots, la soldadura láser está en auge.

Este sector se encuentra cada vez más potenciado por los robots debido a la dificultad que supone este trabajo, el cual requiere de expertos y de herramientas de alta precisión y que siempre trae consigo un mínimo de error humano, incluso llegando a algunas ocasiones a ser imposible.

Además, los soldadores están expuestos a condiciones de trabajo peligrosas (humos, posiciones de trabajo ergonómicas problemáticas, calor y ruido), por lo que el uso de robots ha pasado a ser beneficioso en los procesos GMAW incluso para los lotes más pequeños. Por lo general el proceso automático de soldadura por arco se basa en un electrodo de alambre consumible y un gas de protección que se alimenta a través de una pistola de soldadura.

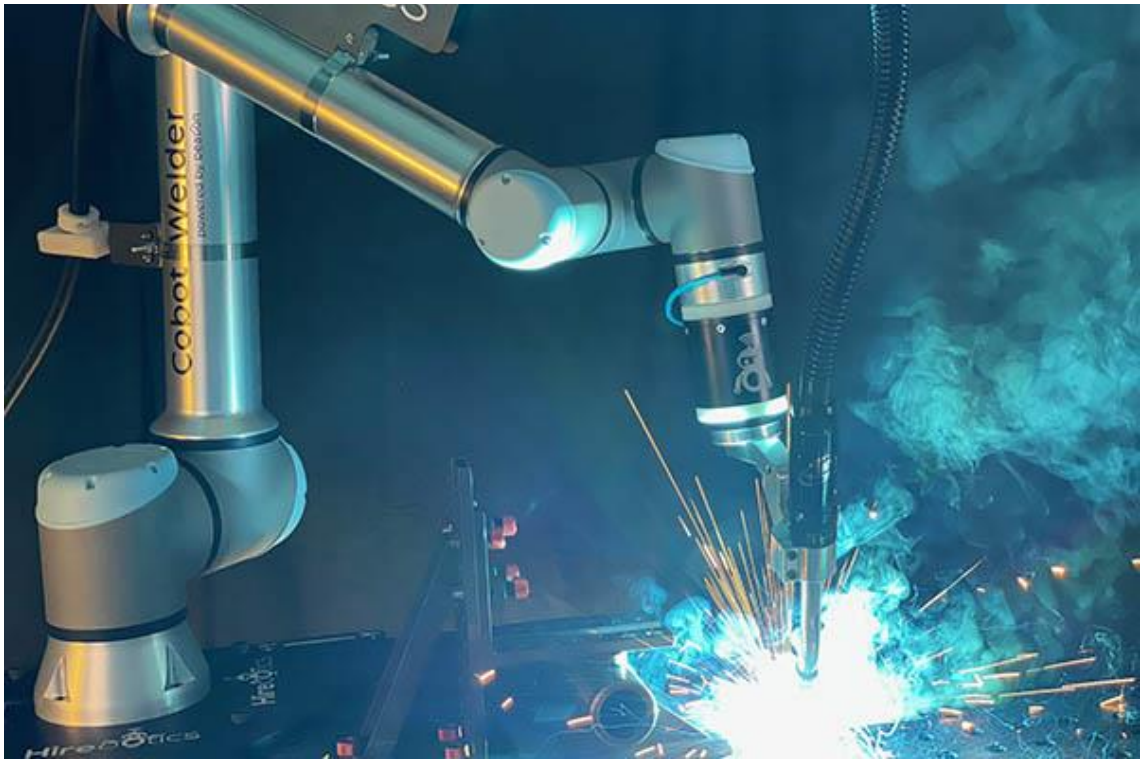


Figura 2.6 Robot Antropomórfico Colaborativo soldador

Ensamblaje

El ensamblaje en la fabricación describe la combinación de subsistemas o componentes en sistemas de mayor complejidad mediante la unión. El ensamblaje en la fabricación comprende cuatro grupos de procesos: unión, manipulación, control y procesos auxiliares (limpieza, ajuste, marcado, etc.).

La composición de estas cuatro funciones puede variar en función del tamaño del lote, el producto y el rendimiento: desde células de trabajo de montaje hasta líneas de montaje de alto rendimiento. Los procesos de montaje constituyen hasta el 80% del coste de fabricación de un producto y es aquí donde puede obtenerse la mayor ventaja competitiva.

Es por esto, que la ventaja competitiva en el ensamblaje tiene distintos aspectos entrelazados para tener en cuenta: diseño de montaje, diseño de la celda de ensamblaje, así como la logística.

Figura 2.7 Robot Antropomórfico Ensamblador para la industria del automóvil

Como se puede observar en la figura 2.6 la industria del automóvil emplea este tipo de robots en mayor proporción al resto de sectores, además la estadística apunta a que su uso se incrementará aún más en el futuro:

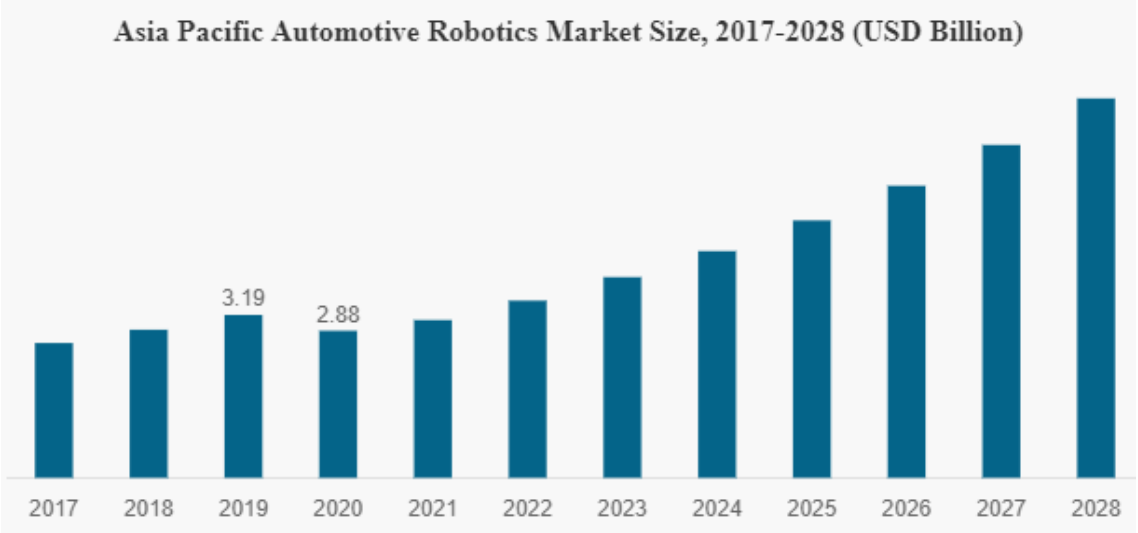


Figura 2.8 Creciente mercado de robots en el sector automovilístico. Obtenida de "FortuneBusinessInsights"

Pintura

Los robots actuales, que al principio se accionaban neumáticamente por razones anti-explosión, son totalmente eléctricos. También tienen ganchos y pinzas para abrir capós y puertas durante la tarea de pintura en el sector automovilístico.

Estos robots cuentan con muñecas huecas que alojan las mangueras de gas y pintura que permiten movimientos rápidos y ágiles. Las pistolas pulverizadoras de los robots han evolucionado mucho para ofrecer una calidad uniforme con la menor cantidad posible de pintura y disolvente, así como para cambiar de una pintura a otra.

Originalmente los robots tenían funcionamientos similares a los humanos, pues se trataba de que estos imitaran sus acciones, actualmente esto ya no es así gracias a la posibilidad de simular los procesos, esto permite optimizar la disposición de la pintura, su grosor y cobertura.

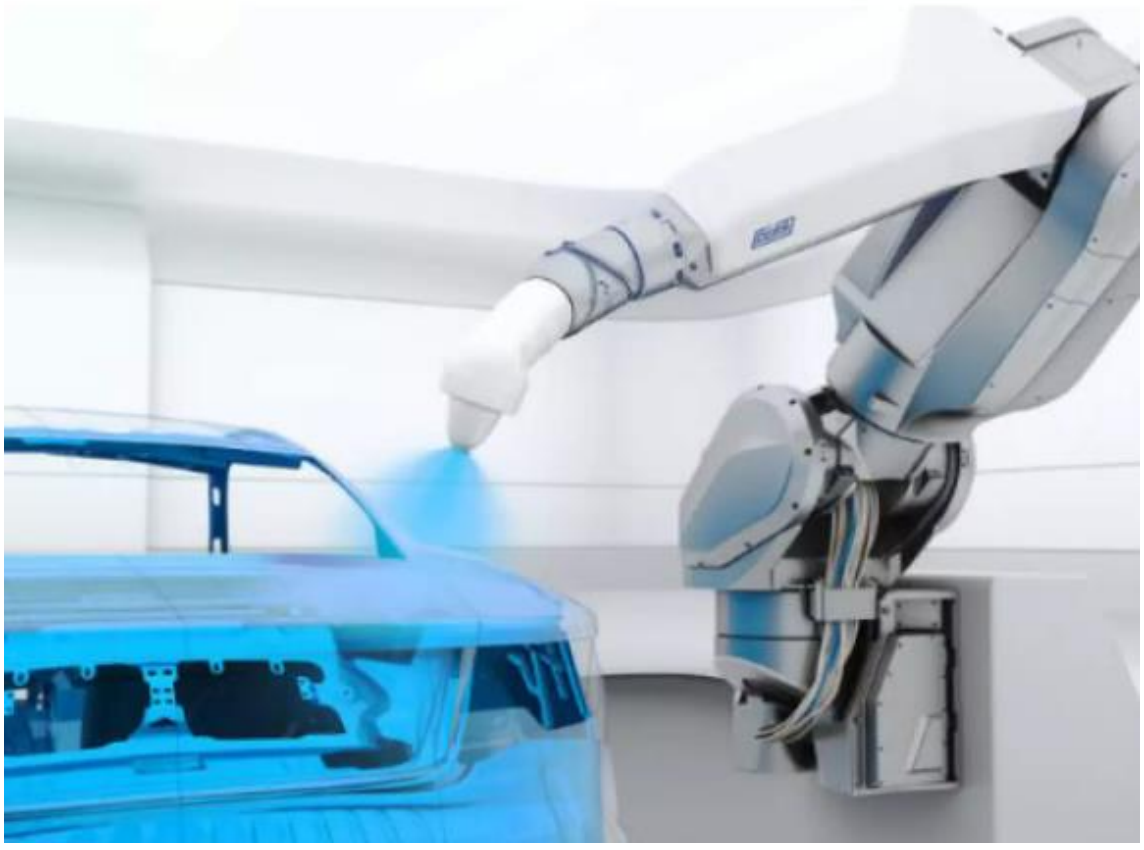


Figura 2.9 Robot Antropomórfico de pintura utilizado por la casa BMW

2.4.2 Visión Artificial

Desde su origen en el campo de la robótica y la automatización industrial la visión artificial ha evolucionado tanto en aplicaciones como en capacidades. Hoy en día podemos encontrar la visión artificial en sectores como la medicina, la agricultura, la conducción autónoma y la industria automovilística entre otros.

Actualmente este ámbito cuenta con algoritmos de reconocimiento de patrones y aprendizaje automático muy superiores a los que contaba al principio, lo que ha dado como resultado a una mayor precisión en el reconocimiento de objetos y características. Además, es posible utilizar estos algoritmos en tiempo real lo que permite un mayor rango de uso.

De forma paralela a la mejora software, el medio físico de la visión artificial ha mejorado en gran medida, contando actualmente con cámaras con altas resoluciones, capaces de medir profundidades, disponibles en todos los tamaños y de todo rango de precios.

Entre los usos actuales de la visión artificial se destacan:

Medicina

La visión artificial tiene cuatro grandes ámbitos cubiertos por documentación: el análisis clínico mediante imágenes, análisis preventivo y terapia, algoritmos fundamentales ara imágenes médicas y el aprendizaje automático para imágenes médicas.

Algunas de las aplicaciones que podemos encontrar dentro de este sector son: detección de tumores mediante ‘IA’, detección de cáncer por visión por computador, monitoreo de signos vitales y salud física, rehabilitación del paciente en casa, guía en operaciones de cirugía e identificación de pacientes.

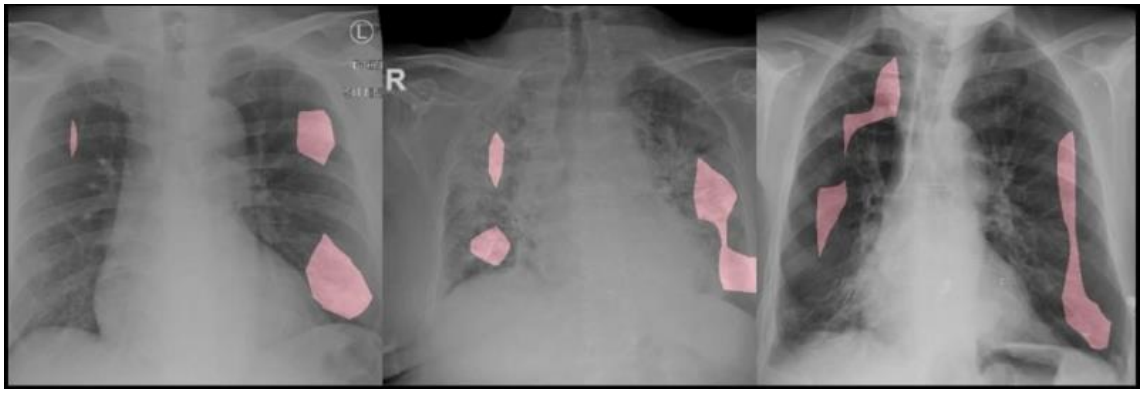


Figura 2.10 Uso de visión por computador para la detección de manchas en pulmones

Agricultura

En el sector agrícola podemos observar la visión artificial en aplicaciones como el monitoreo y gestión de cultivos. Mediante el uso de cámaras es posible aportar a los agricultores información detallada sobre el estado de las plantas y las cosechas, como la detección de enfermades y el estrés hídrico. Esto le permite al agricultor reducir la cantidad de pesticidas a utilizar y aumentar la eficiencia de la aplicación de fertilizantes.

Otra aplicación importante es el conteo y clasificación de frutas y verduras, mediante la identificación y conteo automático, se asegura una clasificación precisa y eficiente en función de distintas características físicas como la forma, el color, tamaño o calidad. De esta forma se agilizan procesos de clasificación y empaquetado, reduciendo la necesidad de trabajos manuales.

Además, la visión artificial se utiliza en la detección de las malas hierbas y la optimización de maquinaria agrícola. Gracias a la categorización de malas hierbas es posible seleccionar de mejor forma los herbicidas necesarios a utilizar y minimizar su uso en áreas no deseadas. Asimismo, mediante visión por computador es posible mejorar la calidad de los trabajos de los vehículos robóticos autónomos.

Deportes

El mundo del deporte comporta movimientos rápidos y precisos que no sólo suponen un reto para los competidores, sino que también pueden ser difíciles de analizar para los entrenadores y el público.

La naturaleza de muchos deportes imposibilita la toma de medidas mediante sensores pegados al cuerpo de muchos deportistas, lo que impulsa el uso de esta herramienta en el sector.

Mediante el uso de cámaras y visión artificial se puede realizar el análisis detallado de técnicas, posturas y acciones específicas, lo que da información muy valiosa para el planteamiento de estrategias, corrección de errores y mejora de rendimiento.

Otra aplicación importante y que cada vez tiene más aplicación es en la toma de decisiones en arbitraje y análisis de decisiones. Esto ayuda a reducir el factor humano y aumentar la imparcialidad en momentos cruciales de eventos deportivos.

Por último, permite mejorar la calidad de la experiencia para los espectadores, siendo en casos de deportes de alta velocidad que mediante tracking en movimiento se permite seguir la posición de los deportistas, permite ver repeticiones de momentos que no han podido ser vistos y análisis que enriquecen la experiencia de los aficionados.

Figura 2.11 Ejemplo de arbitraje mediante visión por computador

Robótica

El ámbito de la robótica la visión artificial ha revolucionado la forma en la que interactúan los robots con su entorno y la complejidad de las tareas que pueden realizar, esto ha sido posible ya que actualmente los robots son capaces de obtener información similar a la que los humanos son capaces a través de los ojos.

Muchas de las tareas que se han mencionado previamente en el apartado 2.4.1 como son el *Binpicking* y la manipulación tanto de objetos como de

alimentos emplean la visión artificial pues permite a los robots identificar los productos a procesar y el entorno del robot permitiendo interactuar de forma segura.

Esta herramienta permite capturar información visual detallada, siendo esto de vital importancia en robots autónomos móviles como es el caso de la exploración del planeta Marte, ya que permite al robot tener una auto localización en el entorno y le permite detectar obstáculos y en otras tareas como mapeo de entornos.

Una última aplicación es la interacción entre los humanos y los robots, gracias a que los algoritmos de visión permiten reconocer y comprender las acciones de los seres humanos y a estos, lo que facilita la comunicación entre ambos en entornos de trabajo compartido.

3. Estudio de Necesidades

3.1 Especificaciones del proyecto

El proyecto ha de ser capaz de realizar el empaquetado de forma autónoma desde que se decide dar inicio al proceso hasta que todos los productos hayan sido empaquetados.

Será necesario que se empleen los robots disponibles que proporciona la Universidad Politécnica de Valencia para su uso en este ámbito de trabajos, debido al alto coste que supone la compra de uno.

El sistema de visión artificial deberá de ser capaz de identificar 3 productos distintos: una lata de maíz, un quesito triangular y una lata de anchoas. Además, deberá poder comunicarse con el robot y transmitirle toda la información necesaria acerca de los mismos, como podría ser la posición.

El sistema deberá contar con flexibilidad en la colocación de los productos, permitiendo a cualquier usuario decidir donde se posicionan los empaquetados de cada uno.

3.2 Limitaciones

Debido a que es necesario utilizar las instalaciones de la Universidad Politécnica de Valencia se deberán emplear las herramientas disponibles, siendo estas unas pinzas o una ventosa. La limitación presente en las pinzas se encuentra en que estas podrían aplastar el quesito y solo están disponibles para

el robot UR3 Serie E; en cuanto a la ventosa, esta solo puede cumplir su función si la superficie es completamente lisa y sin rugosidades.

Figura 3.1 Herramienta de robot colaborativo: pinza

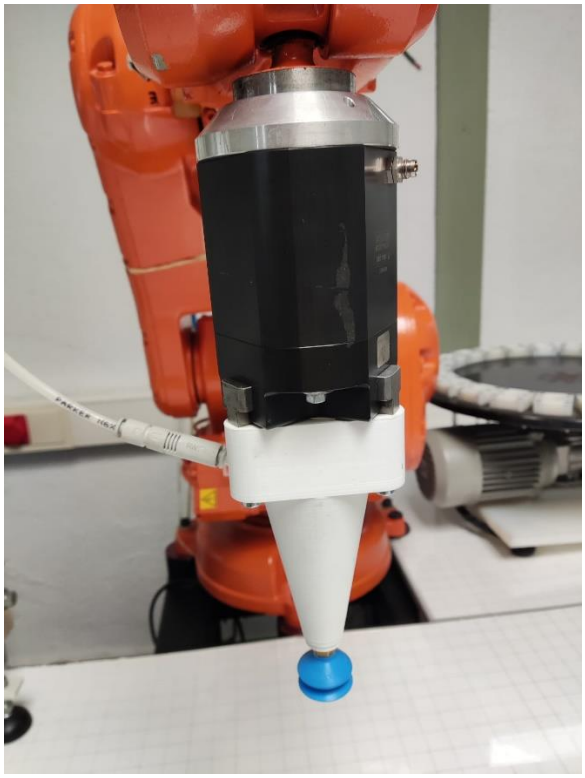


Figura 3.2 Herramienta de robot IRB140: Ventosa

3.3 Normativa tenida en cuenta para la redacción del proyecto

-Directiva 2014/35/UE del Parlamento Europeo y del Consejo de 26 de febrero de 2014 sobre la armonización de las legislaciones de los Estados miembros en materia de comercialización de material eléctrico destinado a utilizarse con determinados límites de tensión. **(D.C. 2014/35/UE)**

-Directiva 2006/42/CE del Parlamento Europeo y del Consejo, de 17 de mayo de 2006, relativa a las máquinas y por la que se modifica la Directiva 95/16/CE (refundición). **(D.C. 2006/42/CEE)**

-Real Decreto 186/2016, de 6 de mayo, por el que se regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos. **(D.C. 2011/65/EU)**

-Real Decreto 187/2016, de 6 de mayo, por el que se regulan las exigencias de seguridad del material eléctrico destinado a ser utilizado en determinados límites de tensión. **(BOE-A-2016-4443)**

-Real Decreto 2200/1995, de 28 de diciembre, por el que se aprueba el Reglamento de la Infraestructura para la Calidad y la Seguridad Industrial. (**BOE-A-1996-2468).** El cual tiene como última modificación y por lo tanto vigente:

Real Decreto 1072/2015, de 27 de noviembre, por el que se modifica el Real Decreto 2200/1995, de 28 de diciembre, por el que se aprueba el Reglamento de la Infraestructura para la Calidad y la Seguridad Industrial. **(BOE-A-2015-13530)**

4. Planteamiento de soluciones alternativas

4.1 Elección de Soluciones Alternativas

En este apartado se muestran las diferentes propuestas de solución para el proyecto de forma cuantitativa, en la que la opción con mayor puntuación justificará el porqué de la elección de dichos componentes. Los apartados remarcados con verde son la elección tomada tras haber evaluado diferentes factores característicos de diferentes alternativas.

Alternativa Robot	Programa	Alcance	Disposición	Herramienta	Seguridad	Total
UR3 Serie E	1	1	3	1	2	8
UR3 Serie CB	1	1	1	2	2	7
IRB 140	2	2	2	2	1	9

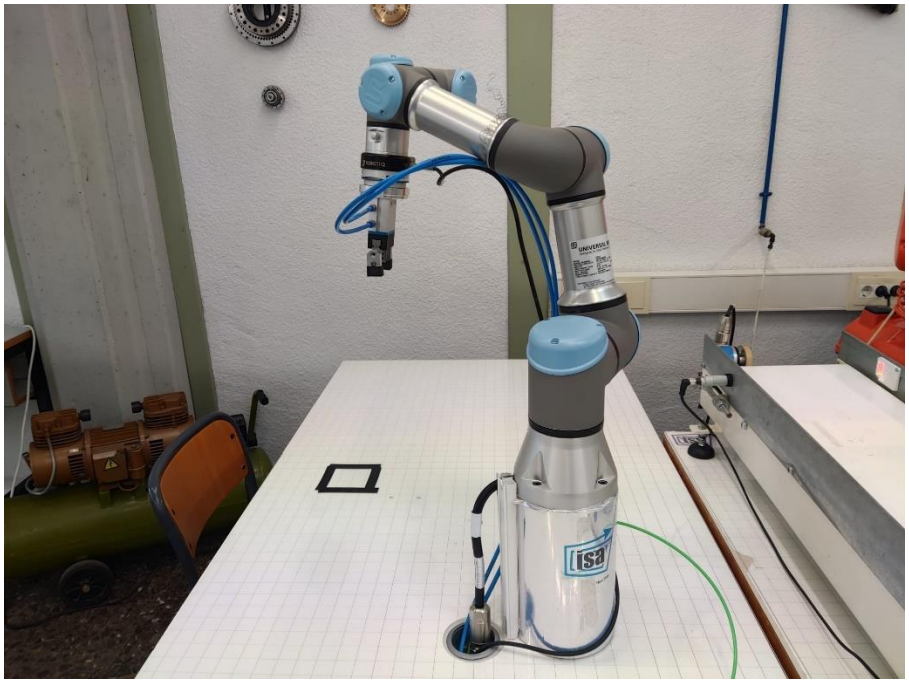


Figura 4.1 Robot UR3 Serie E

Figura 4.2 Robot UR3 Serie CB

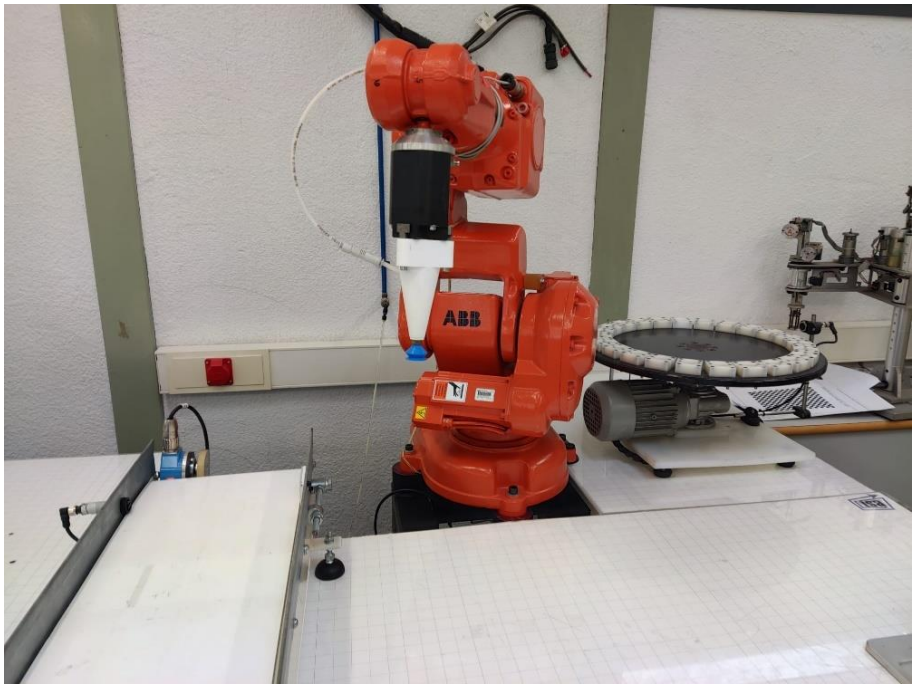


Figura 4.3 Robot IRB140

La valoración seguida para cada apartado ha sido la siguiente:

-Programa: se ha tenido en cuenta si se conocía el programa previamente.

-Alcance: Alcance del área de trabajo del robot.

-Disposición: Como se encuentra situado en el laboratorio el robot, teniendo en cuenta aspectos como la cinta transportadora.

-Herramienta: El robot colaborativo serie E emplea la pinza mientras que el robot IRB140 y el UR3 serie CB utiliza la ventosa, se considera la ventosa mejor opción dado el tipo de producto a utilizar.

-Seguridad: Los robots colaborativos son capaces de detectar al ser humano mientras realizan un proceso.

Alternativa Programa Visión Artificial	Precio	Documentación Online y comunidad de Usuarios	Conocimiento Previo	Flexibilidad	Funcionamiento On-line	Total
Python	3	2	1	3	2	11
MatLab	2	2	2	2	2	10
Sherlock	1	1	2	1	1	6

La valoración seguida para cada apartado ha sido la siguiente:

-Precio: De más económico a más caro.

-Documentación Online y comunidad de Usuarios:

Python es un lenguaje de programación con una gran historia detrás de sí y una gran comunidad desde hace muchos años, que cuenta con diversidad de foros oficiales y no oficiales y cuenta con una gran diversidad de paquetes de funciones para todo tipo de necesidades, de forma muy similar ocurre con el software Matlab. Por el contrario, Sherlock no dispone del mismo bagaje de usuarios.

-Conocimiento Previo: Se ha valorado si ha sido utilizado previamente.

-Flexibilidad: se ha considerado la capacidad de adaptabilidad de programación de cada opción, siendo Python un lenguaje de programación es el que cuenta con mayor flexibilidad, tras él, Matlab es un software diseñado para múltiples funciones, y por último Sherlock un software dedicado expresamente a la Visión Artificial.

- Funcionamiento On-line: En este apartado se valora la posibilidad de que el programa funcione de forma automática, sin necesidad de que el usuario intervenga suministrando manualmente la orden de tomar imágenes o proporcionando las mismas. Sherlock es capaz pero sólo con su versión premium de la que no dispone la Universidad.

Alternativa Cámara	Precio	Tamaño	Resolución	IMU	Total
C920 HD Pro Webcam	3	2	2	1	8
Microsoft LifeCam HD-3000	4	1	1	1	7
Intel Realsense D435i	1	4	4	2	11
Intel Realsense D415i	2	3	3	1	9

Para la valoración de estas alternativas se han seguido los siguientes criterios:

- Precio: De más económico a más caro.

-Tamaño: De menor a mayor tamaño.

-Resolución: Se valora la calidad de imagen capturada.

-IMU: En esta aplicación la IMU no aportará todo su potencial, pero es positivo disponer de estabilización de imagen, ya que el robot se desplaza.

Alternativa Disposición Cámara	Sencillez Diseño	Información del proceso	Relación Cámara-Robot	Total
Móvil con Robot	1	2	2	5
Fija	2	1	1	4

A continuación, se muestra la valorización realizada para cada apartado:

-Sencillez Diseño: la cámara fija permite elegir cualquier punto y orientación en el que se desea establecer.

-Información del proceso: la cámara móvil muestra cómo interactúa el robot con el sistema desde 'su punto de vista'.

-Relación Cámara-Robot: Se ha considerado más sencillo establecer la relación entre el sistema de referencia del robot con una cámara que se encuentra unida a él que una completamente independiente.

4.2 Resumen de la elección de Soluciones Alternativas

Han sido elegidas soluciones para el sistema que se desea implementar:

-El robot IRB140 debido al conocimiento previo del programa que lo controla, por su disposición en el espacio de trabajo y su alcance superior al resto.

-El lenguaje de programación Python por su comunidad de usuarios, precio y gran flexibilidad ante cualquier tipo de situación. Además, pese a que es una desventaja el no conocer nada al respecto de este, se considera una buena oportunidad para iniciarse en él y ganar conocimientos en otra herramienta más de trabajo.

-La cámara Intel Realsense D435i la cual cuenta con el mejor tamaño, una IMU capaz de estabilizar la imagen pese al movimiento y con gran resolución.

5. Descripción detallada de la solución adoptada

En el siguiente apartado se detallará la solución adoptada para el proyecto presente, primeramente, se especificarán la arquitectura del sistema y sus componentes, tras esto se explicarán elementos adicionales del sistema necesarios, a continuación, una descripción general del proceso y por último como se ha desarrollado el software.

Se ha decidido realizar una representación gráfica en forma de organigrama de la organización que se ha tenido para el diseño del proceso:

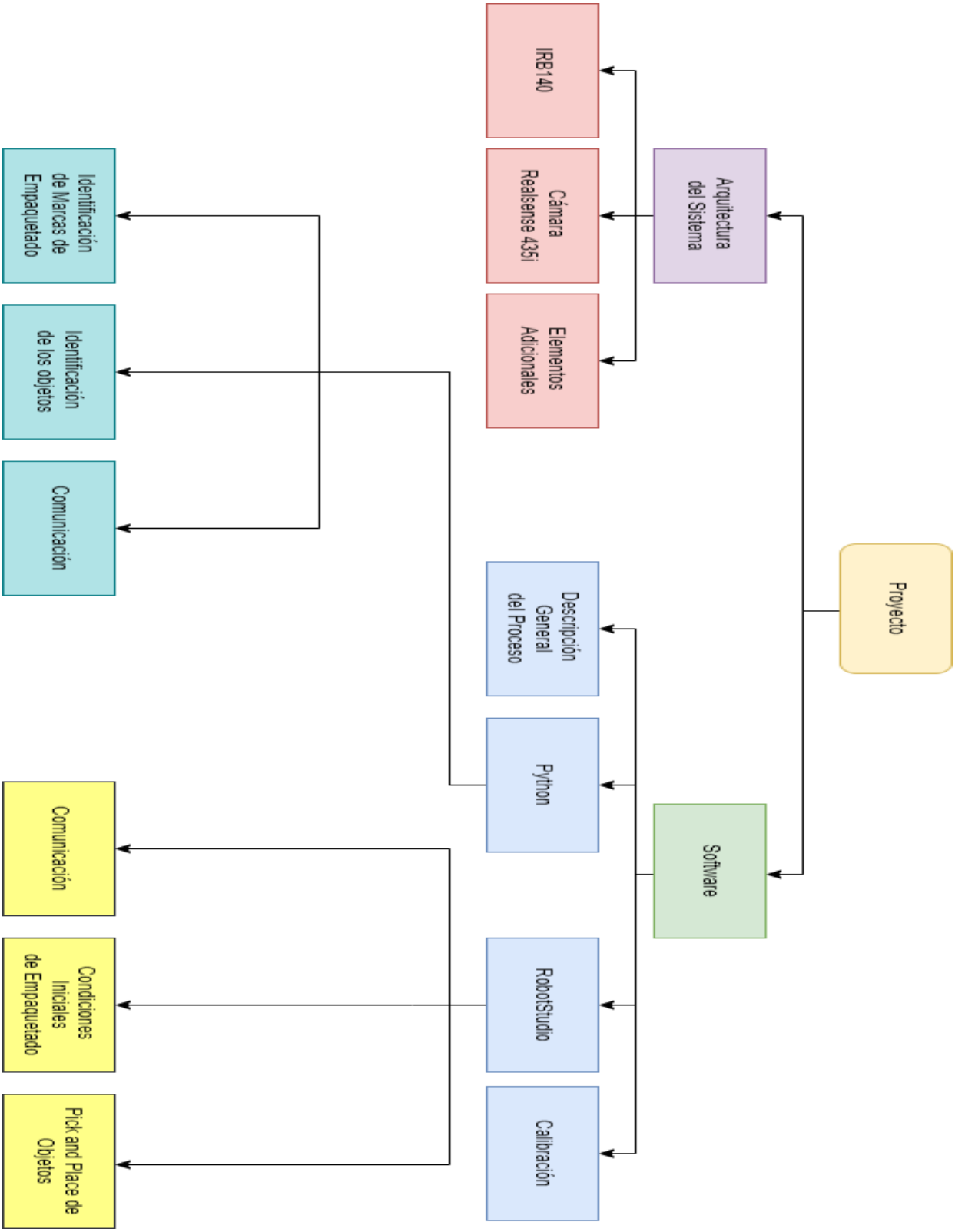


Figura 5.1 Organigrama del proyecto

5.1 Arquitectura del sistema

El sistema se compone de una cámara Intel Realsense D435i, un robot IRB140 y un PC Portátil. El sistema se encuentra interconectado de la siguiente forma: el ordenador portátil se encuentra conectado mediante un cable USB 3.0 por el que recibe la imagen que ha de ser procesada por el programa Python, y a su vez se conecta mediante un cable ethernet a la red de robots del laboratorio para acceder al robot IRB 140.

De esta forma la cámara y el robot se encuentran conectados entre sí mediante el portátil, el cual se encarga de procesar la imagen recibida de la cámara y transmitir la información importante extraída de ella al robot y de forma similar se encarga de recibir cuándo el robot necesita obtener dicha información, tomando entonces el *frame* del vídeo de la cámara en ese momento el cuál corresponde a la imagen a procesar.

Figura 5.2 Representación de transmisión de datos en el sistema

5.1.1 Robot IRB140

El robot IRB140 cuenta con una unidad de control en un armario situado fuera de la zona de acción de los robots del laboratorio. Esta unidad permite encender el robot, y elegir entre un modo manual, un modo manual con la velocidad máxima de los actuadores y un modo automático. Esta unidad es la que se encarga de procesar los scripts y manipular el robot.

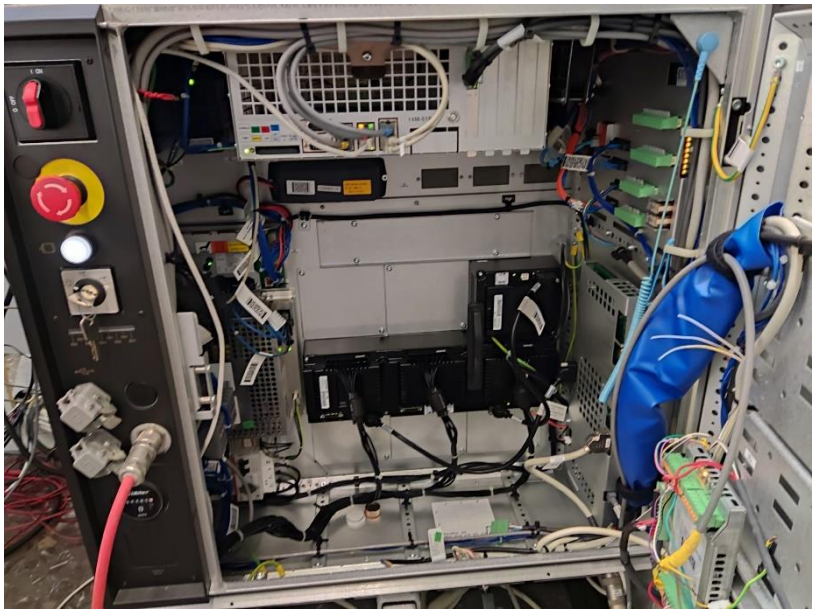


Figura 5.3 Unidad de control IRB140

En la figura 5.3 se pueden apreciar algunos elementos básicos de la unidad de control, situados en la parte superior izquierda, comenzando por un interruptor de On/Off que arranca el sistema, debajo de él una seta de emergencia, de igual forma un botón blanco que se encarga de arrancar los motores cuando se activa el modo automático y por último debajo de este un interruptor por llave que permite elegir entre los 3 modos mencionados previamente.

Además de la unidad de control, el robot cuenta con una pantalla táctil FlexPendant que permite manipular el robot manualmente, controlar la ejecución de scripts y tomar datos del robot como pueden ser la posición y orientación de este. A continuación, se mostrarán y detallarán algunos aspectos y funciones de la pantalla.

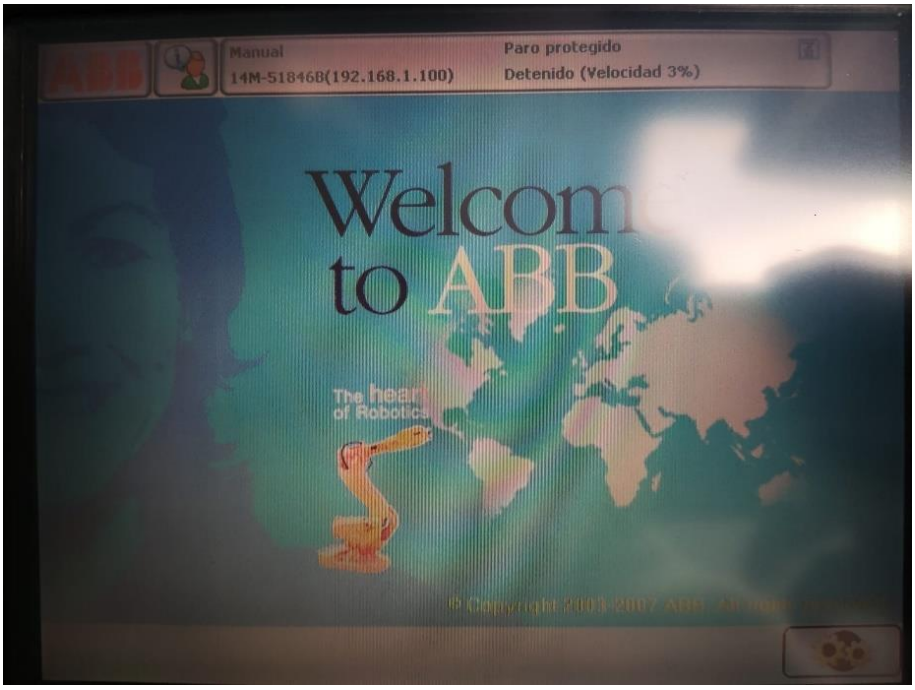


Figura 5.4 Pantalla de Inicio de la FlexPendant

Una vez desde la pantalla de inicio si pulsamos la esquina superior izquierda el botón con la palabra ABB podremos desplegar el menú de opciones que nos permite escoger la ventana a la que queremos acceder.

Figura 5.5 Pantalla de menú de la FlexPendant

En este menú es posible seleccionar múltiples opciones, en el proyecto sólo se han utilizado unas pocas opciones es por ello por lo que no se detallarán todas.

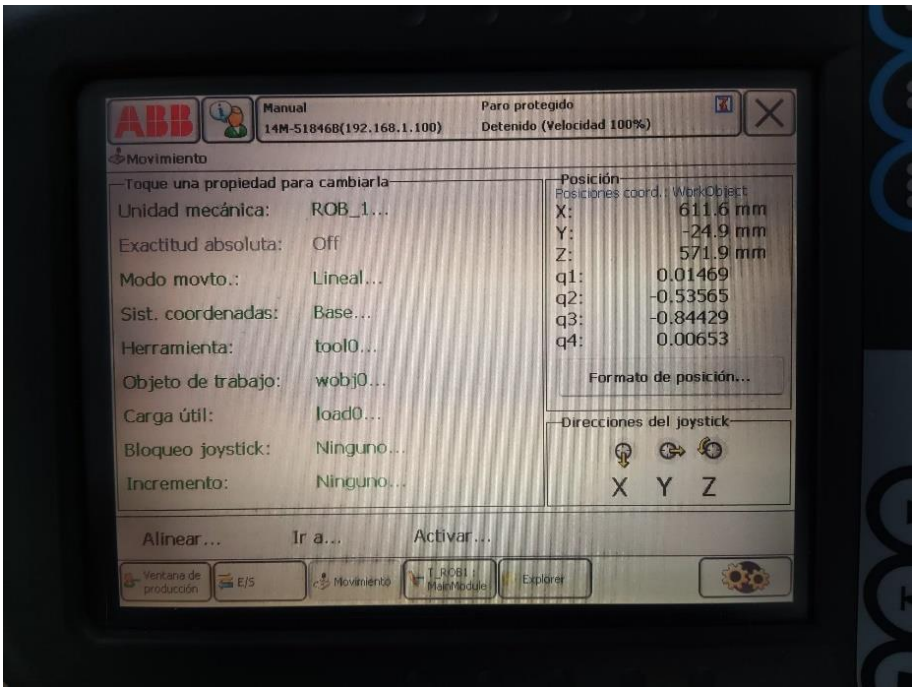


Figura 5.6 Pantalla 'Movimiento' de la FlexPendant

Si en el menú correspondiente a la figura 5.5 se pulsa sobre el botón 'Movimiento' se desplegará la interfaz de la figura 5.6, desde ella se podrá seleccionar el modo de movimiento que se quiere realizar y el formato en el que se desea visualizar la información, en este ejemplo se ha elegido un movimiento lineal y un formato de posición de cuaterniones.

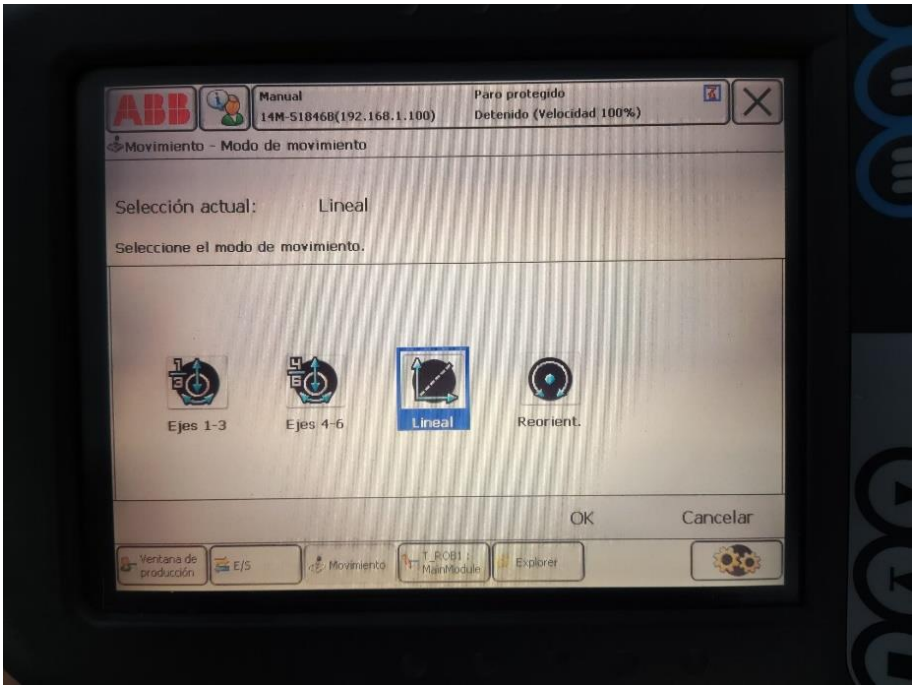


Figura 5.7 Pantalla 'Selección tipo de Movimiento' de la FlexPendant

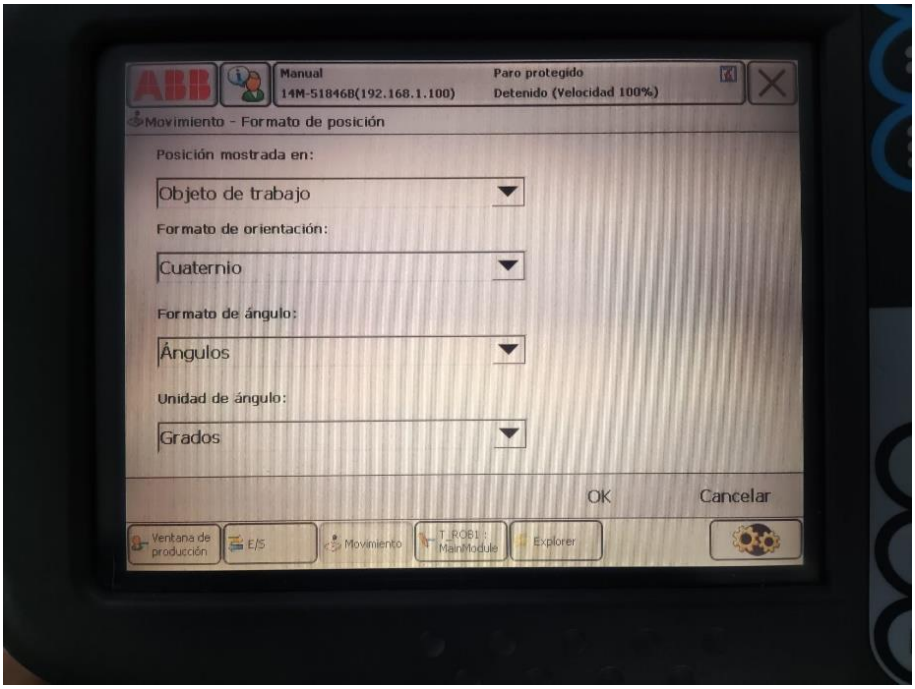


Figura 5.8 Pantalla 'Formato de Posición' de la FlexPendant

También se ha utilizado la opción del menú principal ‘Entradas y Salidas’ que permite visualizar qué entradas están registradas y su estado actual, dando opción a cambiar este.

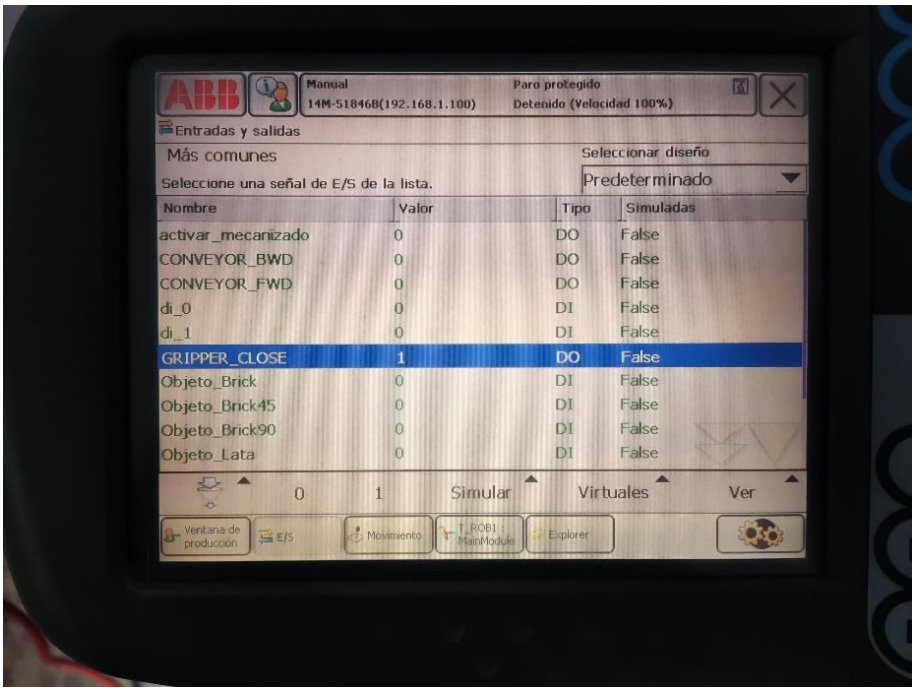


Figura 5.9 Pantalla ‘Entradas y Salidas’ de la FlexPendant

En el caso de la figura mostrada, se está seleccionando la salida digital que acciona la ventosa del robot.

Una última opción utilizada es ‘Editor de Programas’ que nos permite modificar el programa *main* que ejecuta el robot, pudiendo desplazar el puntero que lee el script a través de este, así como también editar las líneas del código.

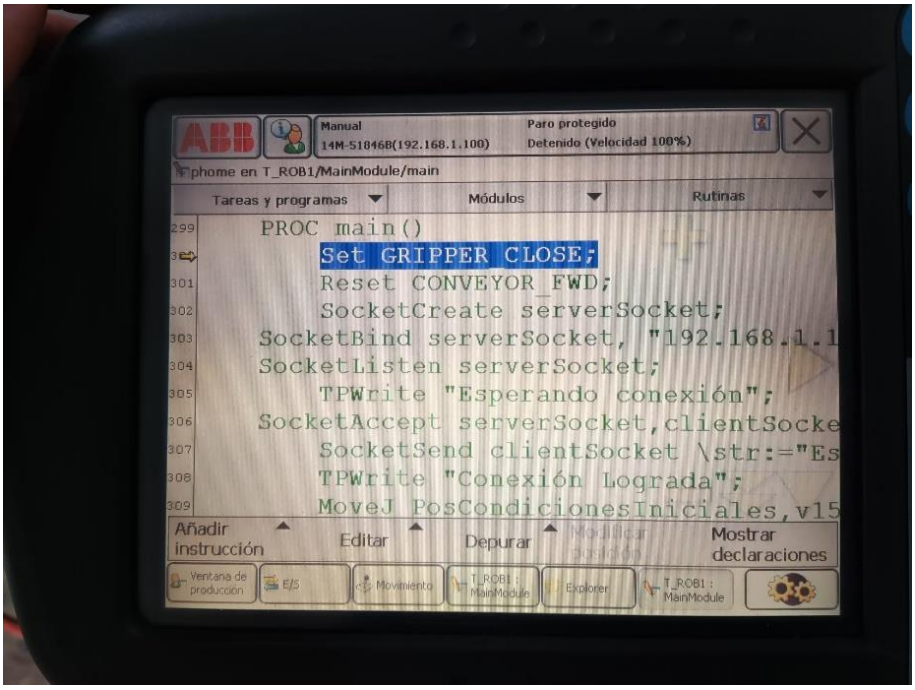


Figura 5.10 Pantalla 'Editor de Programas' de la FlexPendant

Por último, es posible utilizar la opción 'Copia de seguridad' que nos permite guardar en un Pendrive una copia del sistema del robot por completo y descargarla en el programa RobotStudio, sin embargo, el robot del laboratorio se encuentra desactualizado y es imposible realizarlo de esta forma, es por ello por lo que se ha realizado online, lo cual se detallará en el apartado de software.

5.1.2 Cámara Intel Realsense D435i

Figura 5.11 Presentación Cámara Realsese d435i proporcionada por la empresa Intel

En este apartado se detallarán los aspectos más importantes de la cámara escogida para el proyecto:

Entorno de uso	Interiores/Exteriores
Rango ideal	0.3-3 metros
Tecnología de profundidad	Estereoscópica
Resolución RGB	1920x1080
Frame Rate RGB	30 fps
Módulo de Cámara	Intel RealSense Module D430 + RGB Camera
Dimensiones de la cámara (LongitudxProfundidadxAltura)	90 mm x 25 mm x 25 mm

Además de estas características la cámara presenta una IMU (Unidad de medición Inercial) lo que permite realizar mediciones en movimiento y estabilizar la imagen.

También es importante añadir que la cámara cuenta con Intel® RealSense™ SDK 2.0, una plataforma *open-source* que proporciona herramientas, librerías y códigos de ejemplo para diversos lenguajes, es de esta plataforma de la que se ha obtenido una librería para poder obtener el video RGB de la cámara, esto se detallará mejor en el apartado de software.

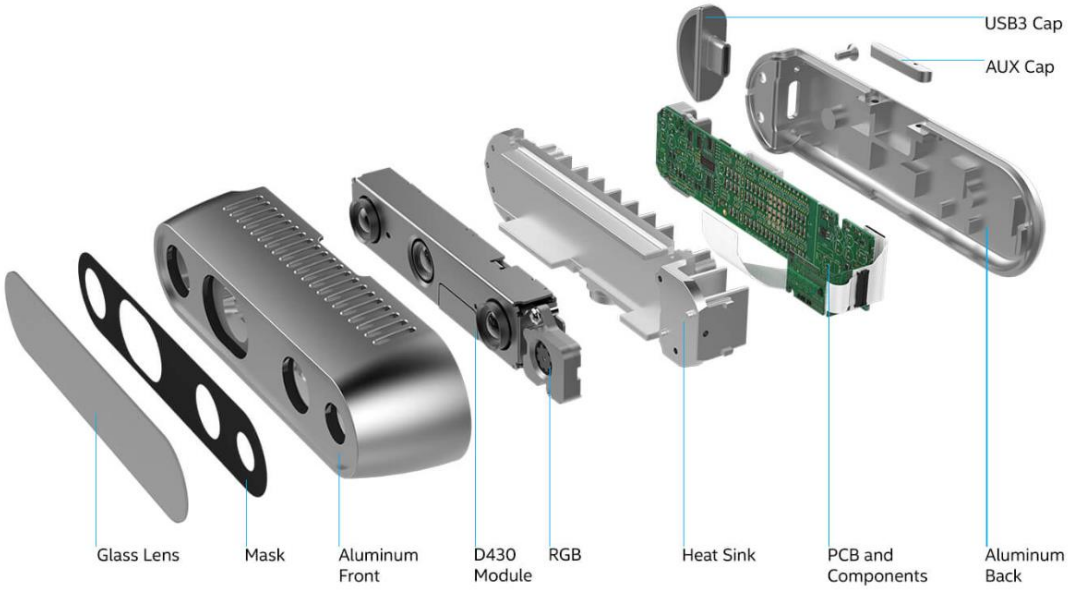


Figura 5.12 Composición de la cámara Intel Realsense D435i

5.1.3 Elementos Adicionales

Para la conexión entre PC y cámara, la empresa Intel proporciona con la adquisición un cable 3.0 USB, sin embargo, la longitud de este no es suficiente para conectarse al ordenador desde el robot, ya que el PC necesita estar cerca del switch del laboratorio para conectarse a este. Es por este motivo que se ha adquirido un cable 3.0 USB de 5 metros que permite la conexión.

A su vez será necesario un ordenador que se encargue de la programación de Python y de las lecturas de la cámara. El ordenador es necesario que pueda instalare una de las versiones de Python 3 y tenga acceso a un switch al que conectarse con el robot. Para el proyecto se ha utilizado un portátil ASUS TUF A15.

Asimismo, para el establecimiento de los puntos desde los que se quiere realizar cada empaquetado se ha decidido que se realizarán mediante tres pegatinas cuadradas de colores y el propio círculo de los quesitos, el cuál hará su función de envase. Una de las pegatinas quedará libre para un producto extra que se pudiera añadir.

Además, ha sido necesario crear un soporte para la cámara que se ha de acoplar a la herramienta, este soporte ha sido creado mediante la herramienta SolidWorks y se ha adaptado a las medidas de la herramienta ventosa con la que ya se contaba en el laboratorio, de forma que se acople entre la ventosa

creada de plástico y el elemento negro al que va atornillada. Los planos de la pieza se detallarán en el apartado de planos del documento.

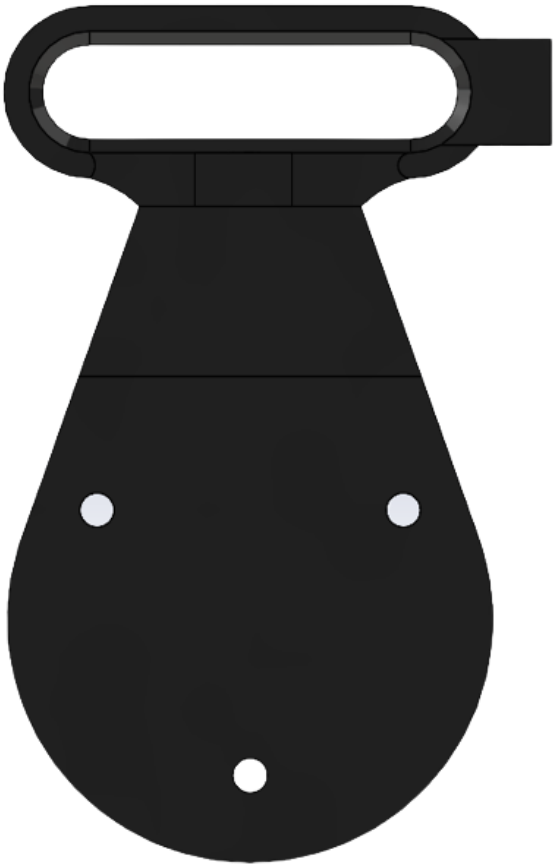


Figura 5.13 Soporte creado de cámara para IRB140

Por último, el sistema cuenta con un controlador externo a la unidad de control que se encarga de proporcionar las entradas y salidas del sistema de robots del DISA, entre ellas se utilizan el sensor foto eléctrico del final de la cinta, el motor de avance de la cinta y la activación de la ventosa del robot.

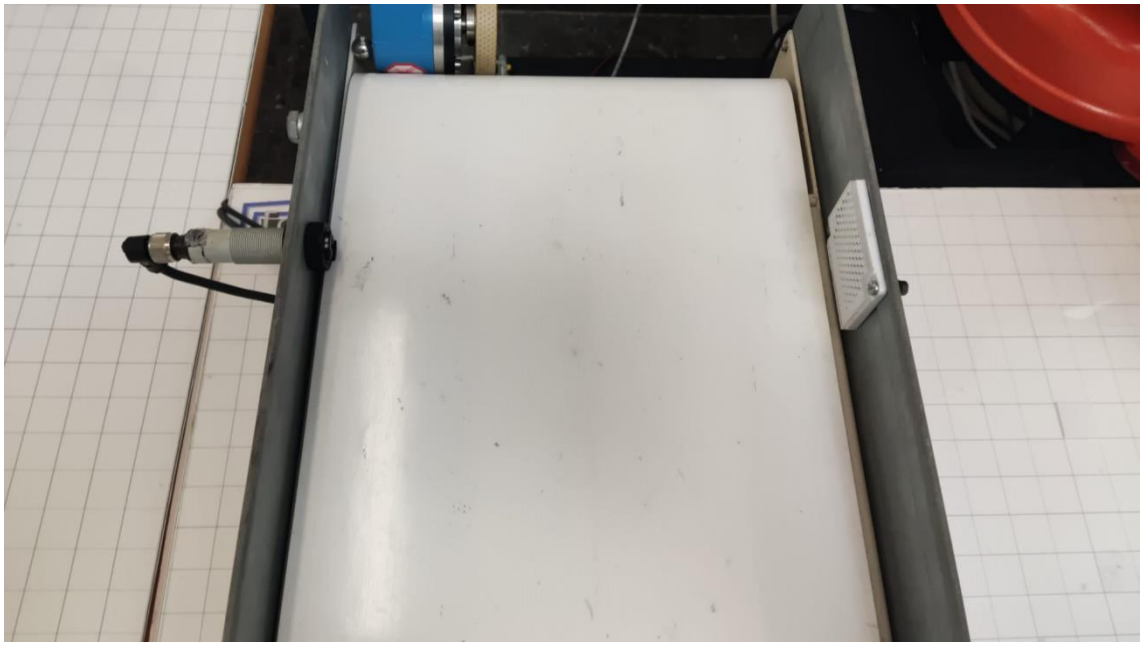


Figura 5.14 Cinta transportadora con Sensor Fotoeléctrico

5.2 Software

En el presente proyecto han sido utilizados tres softwares distintos para la implementación del sistema, se ha empleado el software Matlab para la calibración de la cámara gracias a un algoritmo denominado homografía, también ha sido utilizado el software RobotStudio ya que es el pertinente del robot IRB140 y por último el elegido como solución adaptada para la visión artificial, Python.

A continuación, se detallará el proceso de forma general y tras ello las partes pertinentes de RobotStudio y Python, la calibración se explicará en el anejo de Cálculos. Todos los *scripts* empleados en el proceso se adjuntarán en el anejo 'Códigos'.

5.2.1 Descripción General del Proceso

Para detallar la descripción general del proceso se ha decidido utilizar un diagrama de flujo que muestre *grosso modo* el funcionamiento de todo el sistema.

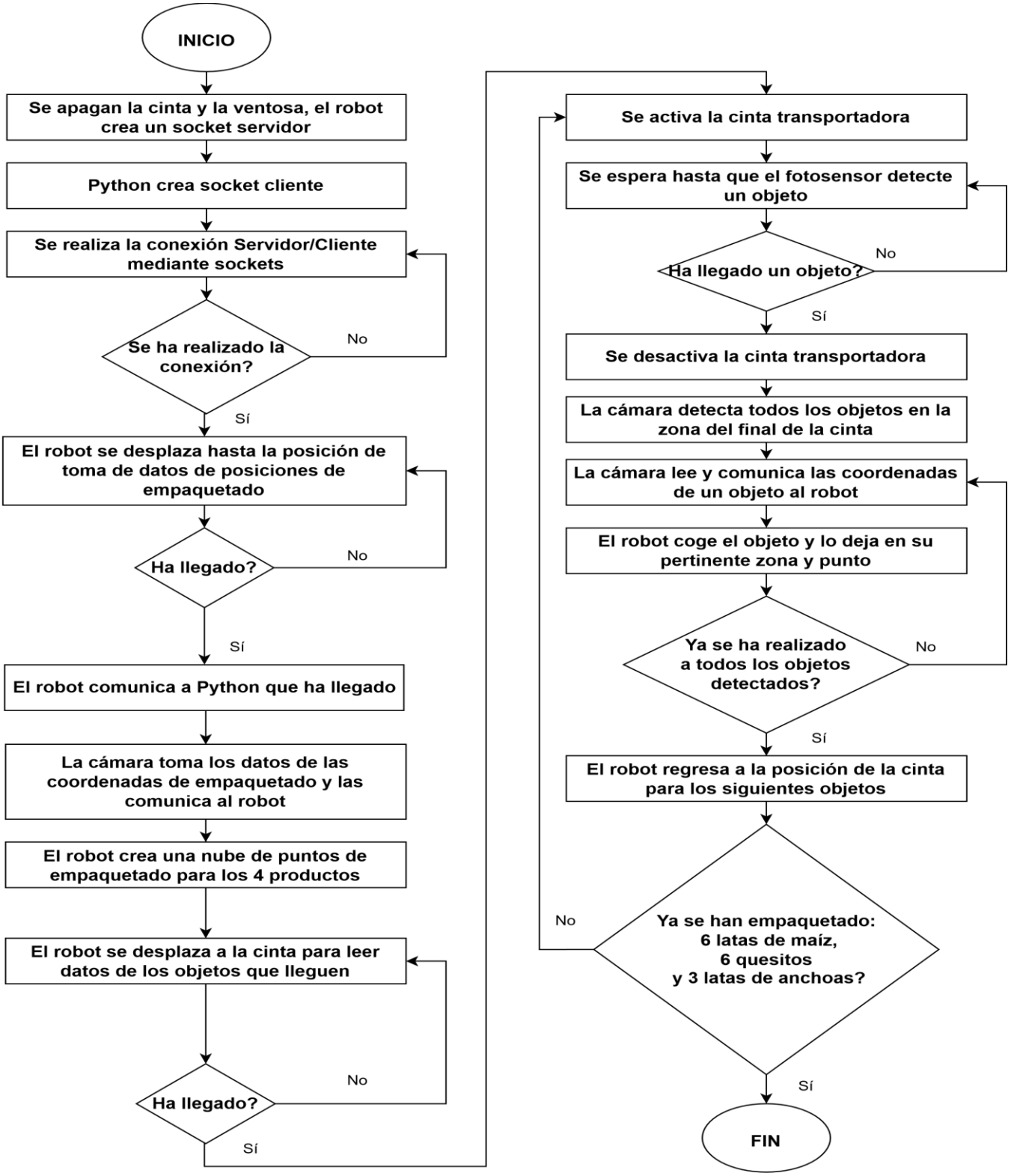


Figura 5.15 Diagrama general del proceso

5.2.2 Python

Desde Python es necesario identificar los patrones que marcan las zonas de empaquetado, distinguir entre los distintos objetos y comunicarle al robot toda la información necesaria al respecto de los dos casos, es por ello por lo que se procederá a explicar cada apartado por separado. Para la realización de estos cometidos han sido utilizadas 2 herramientas que aportan un conjunto de librerías: openCV y Sickit-Image. Además, se ha empleado una librería específica de la cámara a utilizar.

5.2.2.1 Identificación de marcas de empaquetado

Tal y como se ha comentado en el apartado de elementos del sistema para establecer dónde se desea realizar el empaquetado de cada producto se utilizarán pegatinas cuadradas de colores y el círculo de los quesitos.

El método para identificar estas marcas de empaquetado será mediante una umbralización por colores, es por este motivo que, ha sido modificado el color del fondo del círculo de los quesitos debido a que este era prácticamente idéntico al color de la mesa donde se empaquetan los productos, y debido a que no es posible modificar este aspecto se ha decidido cambiar el color del círculo.

Para realizar la umbralización por color se ha decidido utilizar dos espacios de colores, el ‘HSV’ y el ‘RGB’. El espacio de colores ‘HSV’ se ha utilizado para la identificación de los colores azul, verde y rojo y el espacio de color ‘RGB’ para el negro.

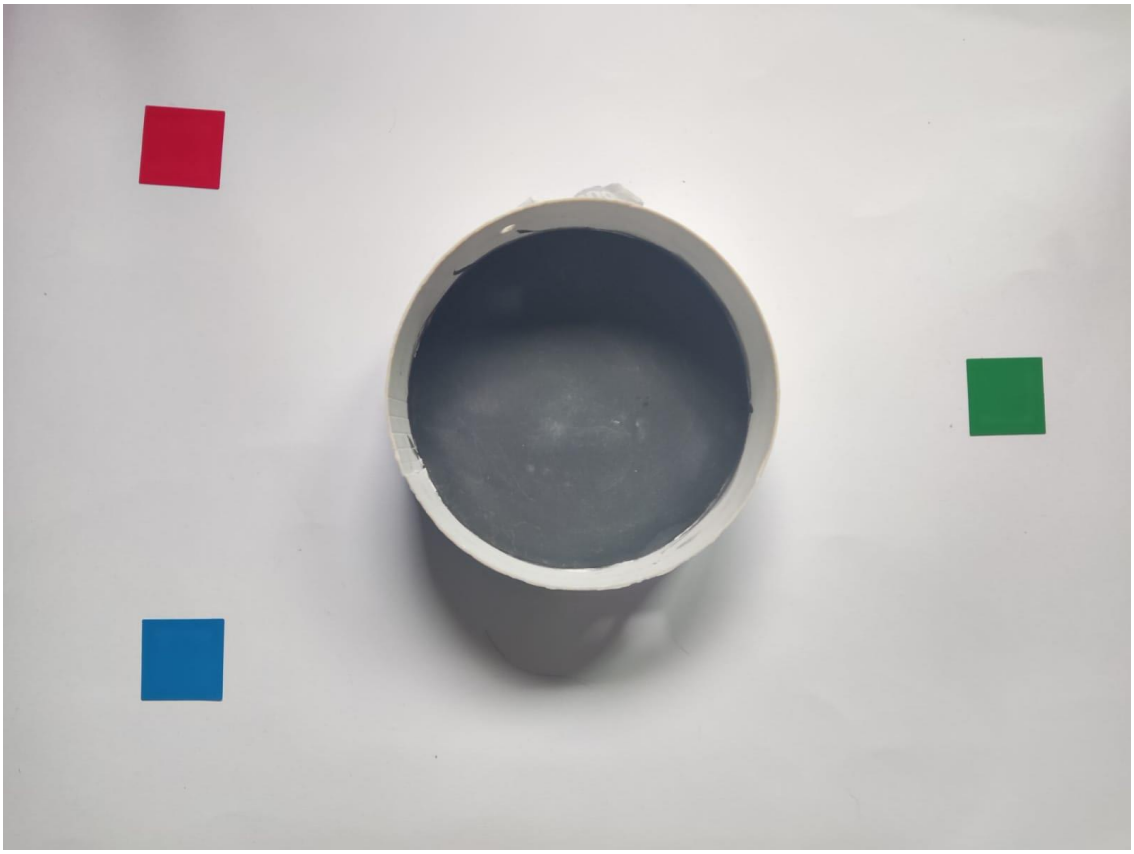


Figura 5.16 Elementos de marca de zona de empaquetado

El espacio HSV aporta una mejor capacidad de segmentación que el espacio RGB, esto se debe a que este se encuentra definido por tres componentes: matiz (*Hue*), saturación (*Saturation*) y valor (*Value*), lo que permite una independencia de los componentes de color (al contrario que en RGB) y una mayor robustez ante cambios de iluminación.

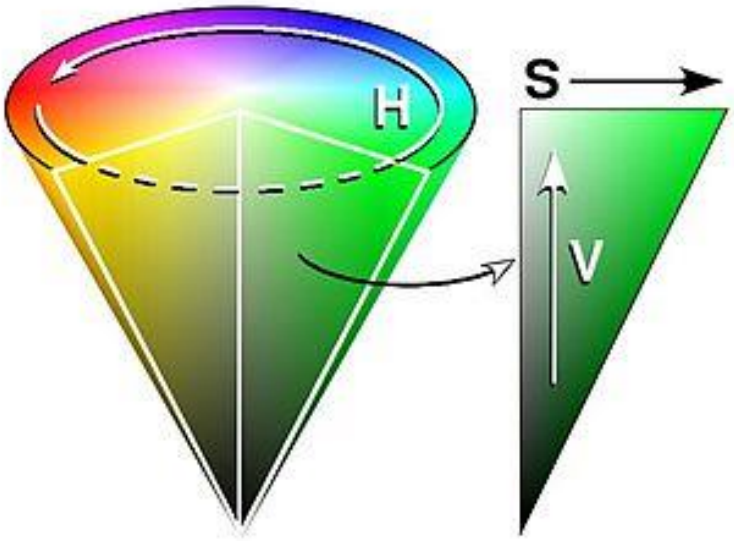


Figura 5.17 Espacio HSV en 3D

Tal y como se puede apreciar en la figura superior, es posible crear rangos de valores para los matices de los colores y mantener un rango completo de saturación y valor, lo que permitiría diferenciar entre todos los matices de color verde y todos los matices de color azul independientemente de la iluminación de la zona.

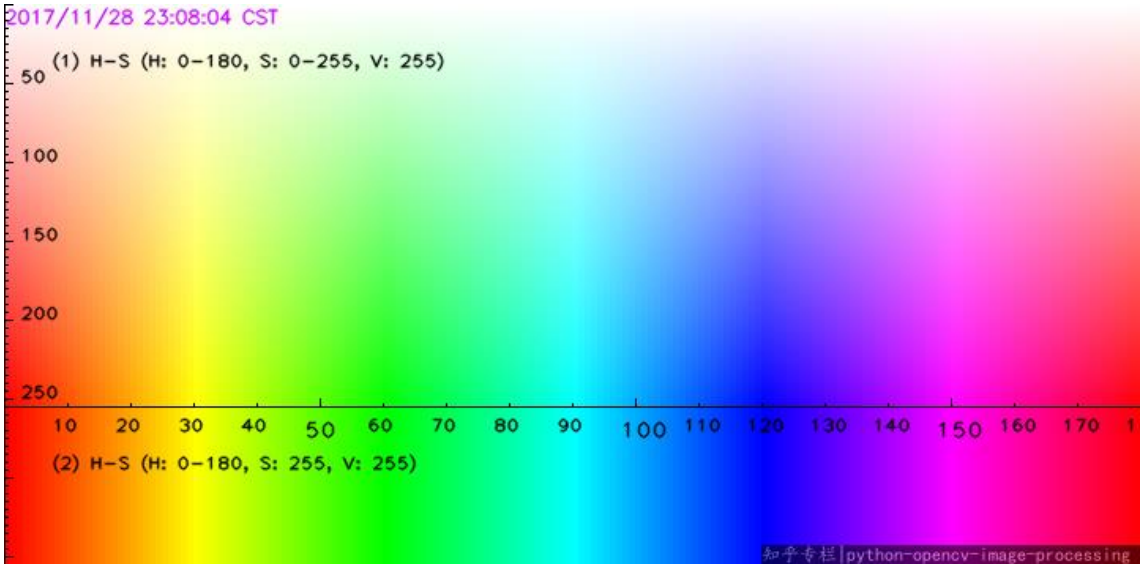


Figura 5.18 Espacio HSV en OpenCV

En OpenCV es posible acceder al espacio HSV representado en la figura superior, este permite identificar las pegatinas roja, azul y verde. Tal y como se puede apreciar el color rojo tiene un rango de valores en el canal de matiz de 0 a 20 y en el caso de los otros dos canales para obtener robustez frente a iluminación podemos especificar un rango completo de 0 a 255. En el caso del color azul el caso es idéntico excepto en el canal de matiz donde su rango será de 95 a 130 y en el caso del verde será de 35 a 90.

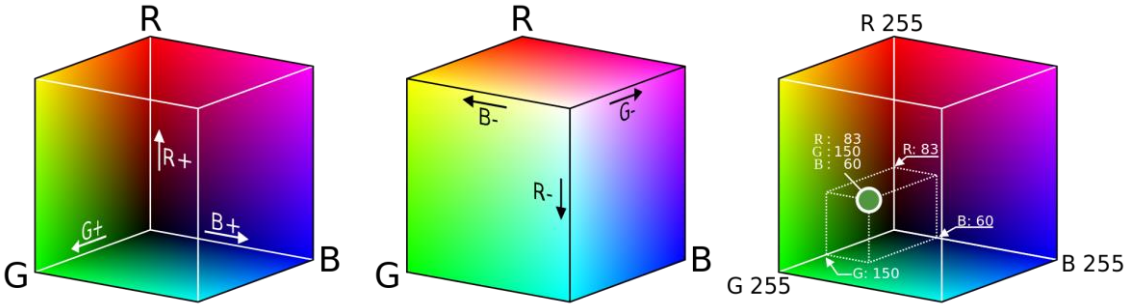


Figura 5.19 Espacio RGB en 3D

Por otro lado, el espacio RGB se forma por la dependencia entre los valores de color rojo, verde y azul, lo que implica que en este caso no es posible separar los colores blanco y negro de los matices de color que se desean. Es por este motivo que se obtendrán los valores RGB del fondo de la aplicación (la mesa del laboratorio) y el color negro del círculo de los quesitos, esto permite crear unos intervalos ajustados que permitan aislar los colores deseados. El código que se usará para la identificación de estos valores tiene por nombre 'ObtenerRGByNivelDeGris'.

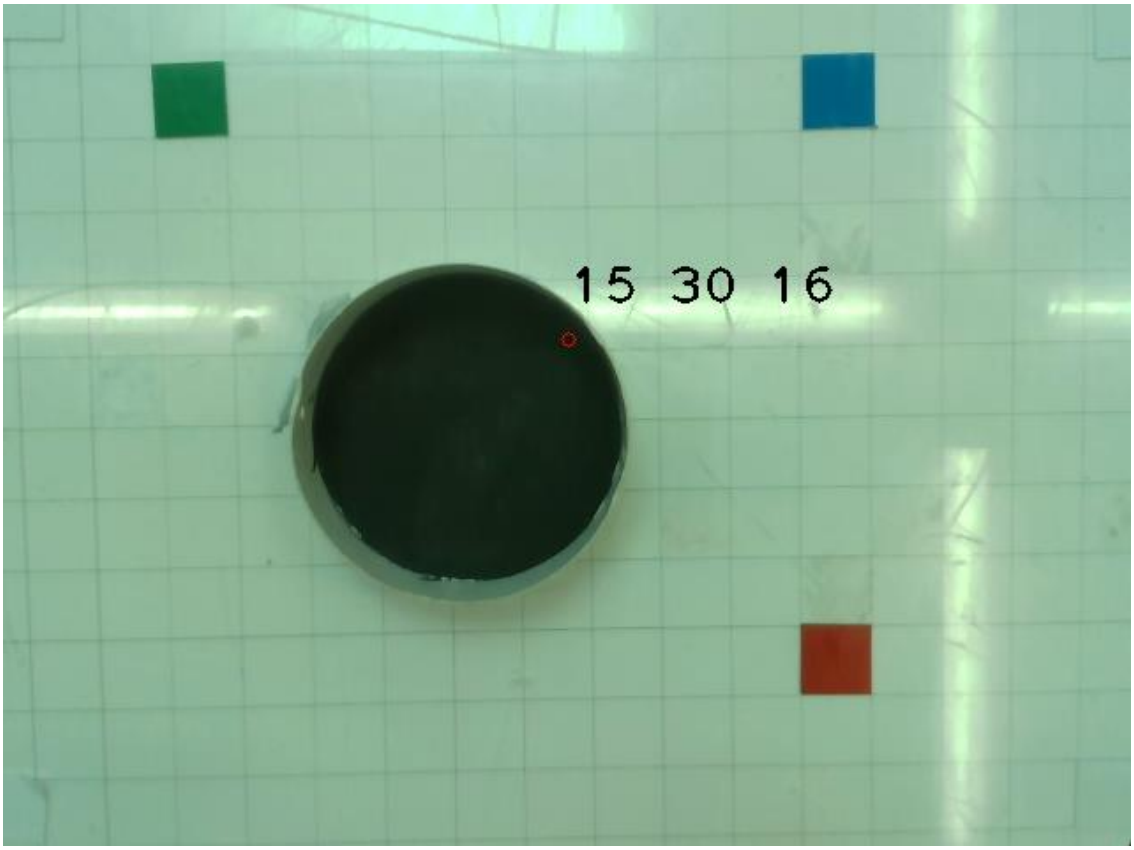


Figura 5.20 Identificación de los valores RGB necesarios

Una vez obtenidos todos los valores para los elementos de identificación de zona de empaquetado se crean máscaras con las que obtener de la imagen en color del conjunto, imágenes que aíslan cada elemento, estas imágenes que se obtienen se pasan a escala de grises y tras ello a binario.

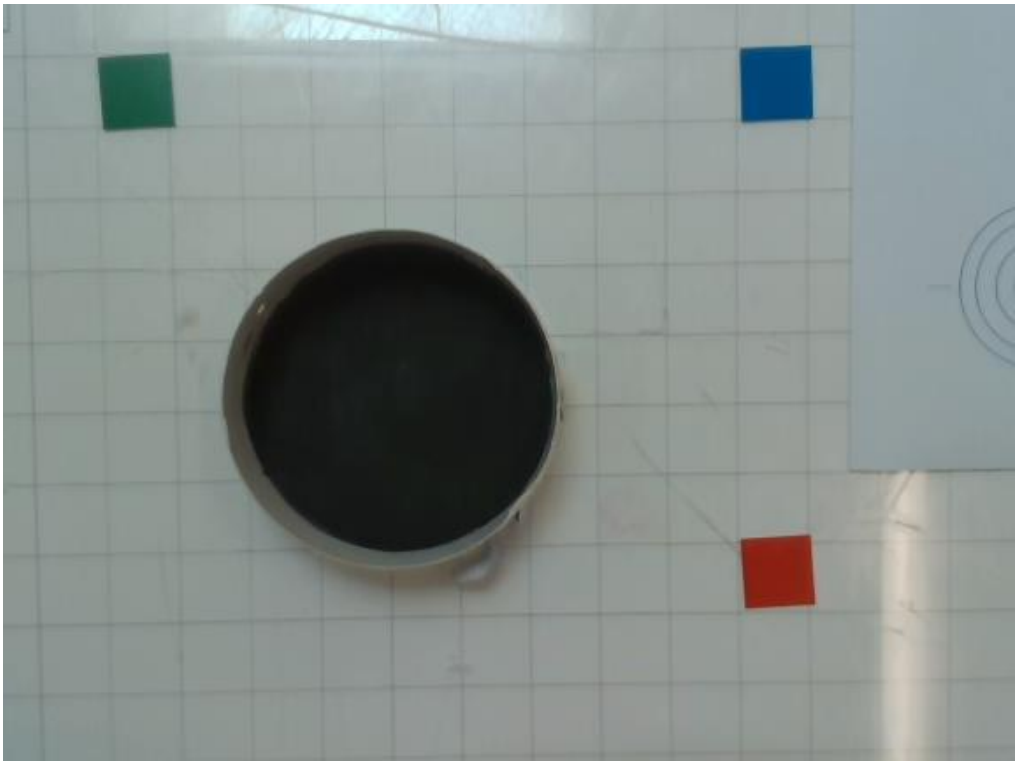


Figura 5.21 Imagen en color del conjunto

Figura 5.22 Imagen en color de la pegatina azul

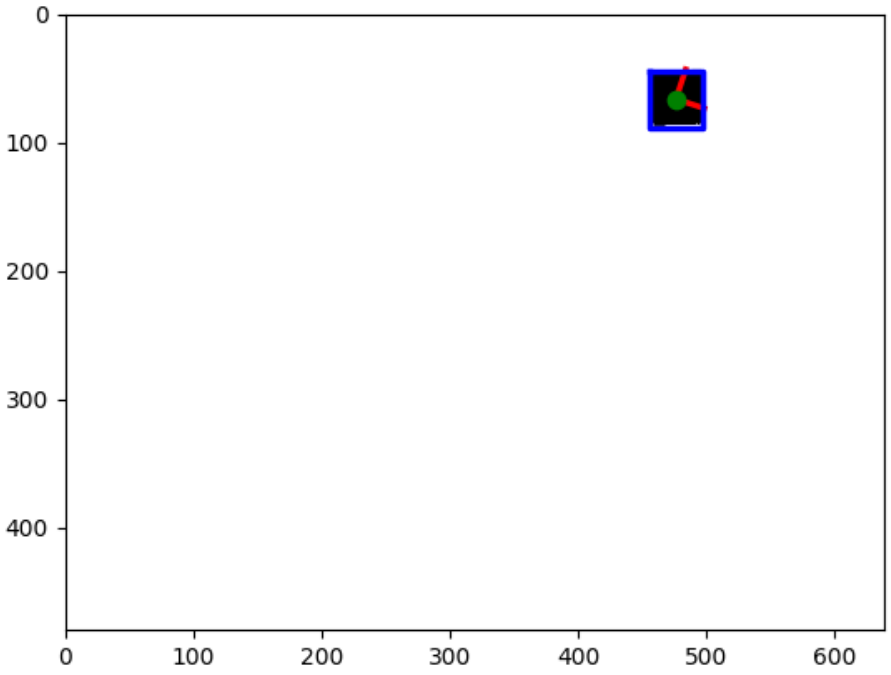


Figura 5.23 Imagen en blanco y negro de la pegatina azul

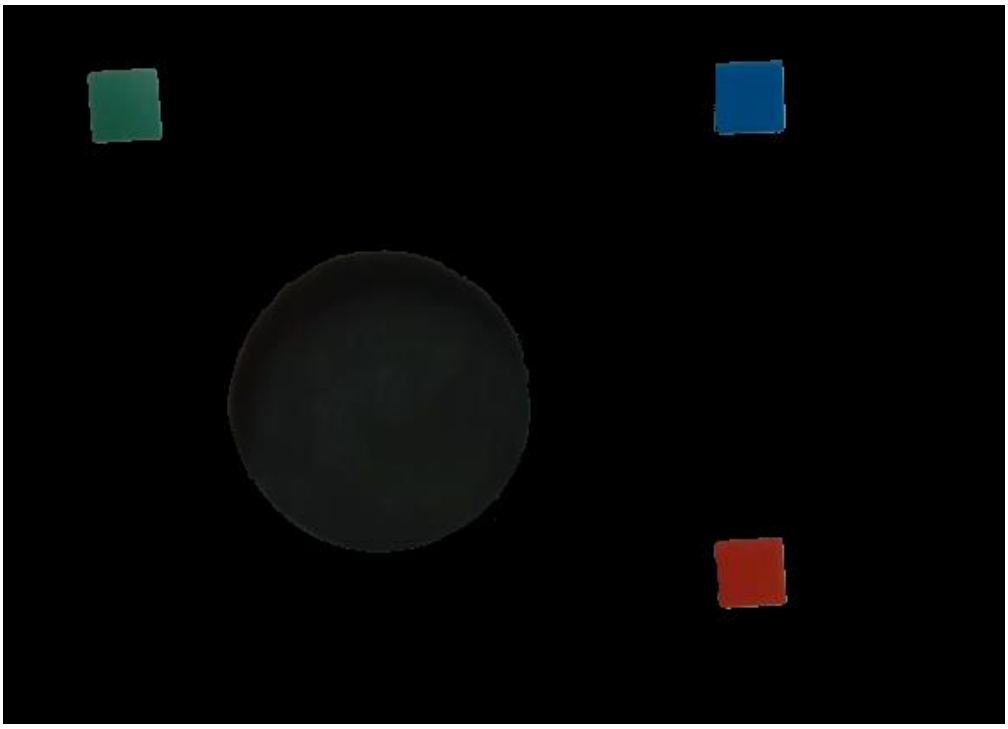


Figura 5.24 Imagen del conjunto umbralizado en color

Tras segmentar la imagen es posible obtener los centros de gravedad de cada elemento mediante la librería ‘skimage.measure’ desde la que se importan las siguientes funciones:

-Label: A partir de una imagen en blanco y negro esta función identifica objetos o elementos en un fondo.

-Regionprops: Obtiene de cada elemento identificado por ‘label’ una serie de características entre las que se destacan el área, la orientación y el centro de gravedad entre muchas otras.

Para obtener los centros de gravedad se le aplica a cada imagen binaria de cada elemento aislado la función ‘label’ y tras ello la función ‘regionprops’ que permitirá obtener y devolver los centros de gravedad.

Además, debido a que es posible que algunos elementos muy pequeños no hayan sido umbralizados por completo y la función ‘label’ los identifique, nos valdremos de la característica del área para únicamente devolver el centro de gravedad de aquellos elementos que sean superiores a cierto número, dejando como única opción el elemento que deseamos. A continuación, un ejemplo:

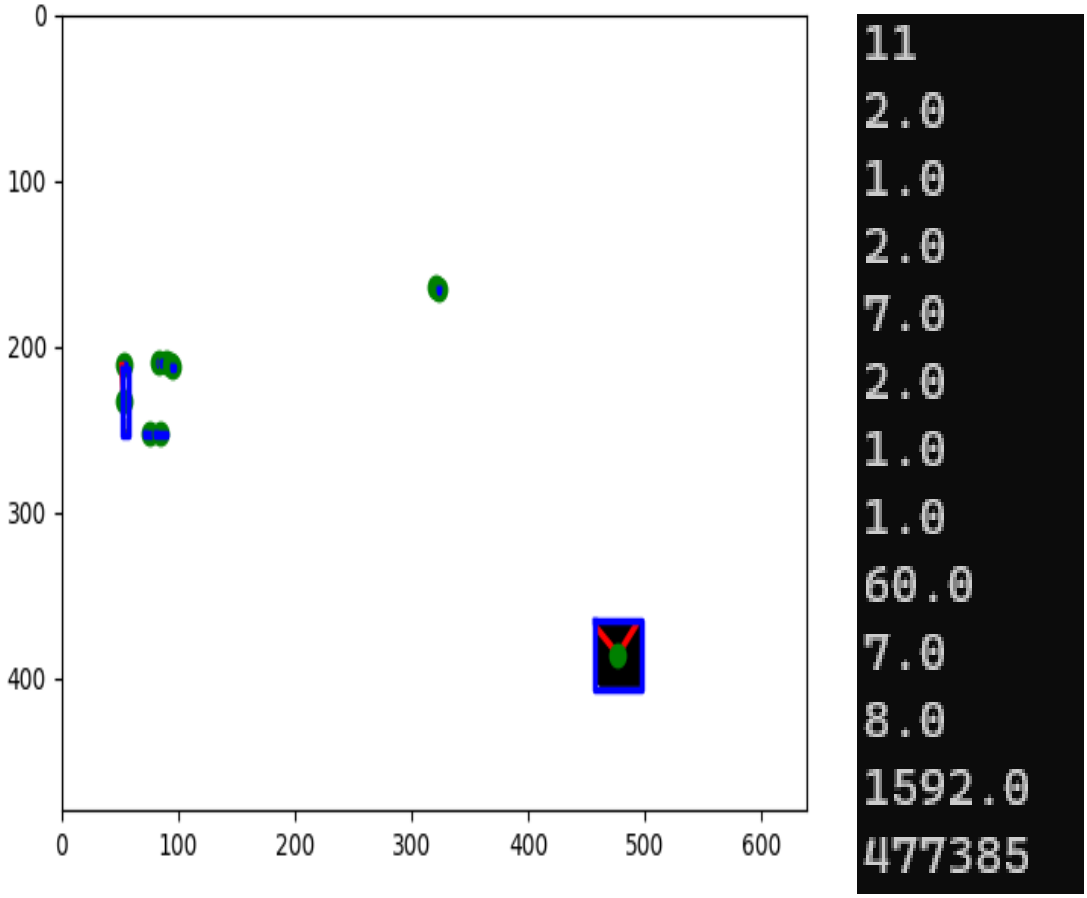


Figura 5.25 Identificación de pegatina y elementos no deseados

Otro aspecto que se desea obtener es la altura a la que se encuentran las pegatinas y el círculo, como estos no se desplazan en el eje Z, se puede tomar la altura del robot en estos puntos mediante la FlexPendant.

Todo el proceso explicado y el envío de la información extraída se encuentra en la función 'CondicionesIniciales()' del código principal de Python.

5.2.2.2 Identificación de los objetos

Los objetos que se tiene por objetivo empaquetar son: seis latas de maíz, seis quesitos triangulares y tres latas de anchoas. Los objetos se irán introduciendo en la cinta transportadora del laboratorio la cual los irá desplazando hasta el sensor foto eléctrico del final de carrera, lo que parará la cinta para proceder a su identificación.

Figura 5.26 Objetos para empaquetar

Una vez ha llegado como mínimo un objeto la cámara obtendrá una imagen desde una posición por encima de la cinta, por lo que, aquello que la cámara obtendrá será una imagen de la planta de los objetos y sus alrededores.

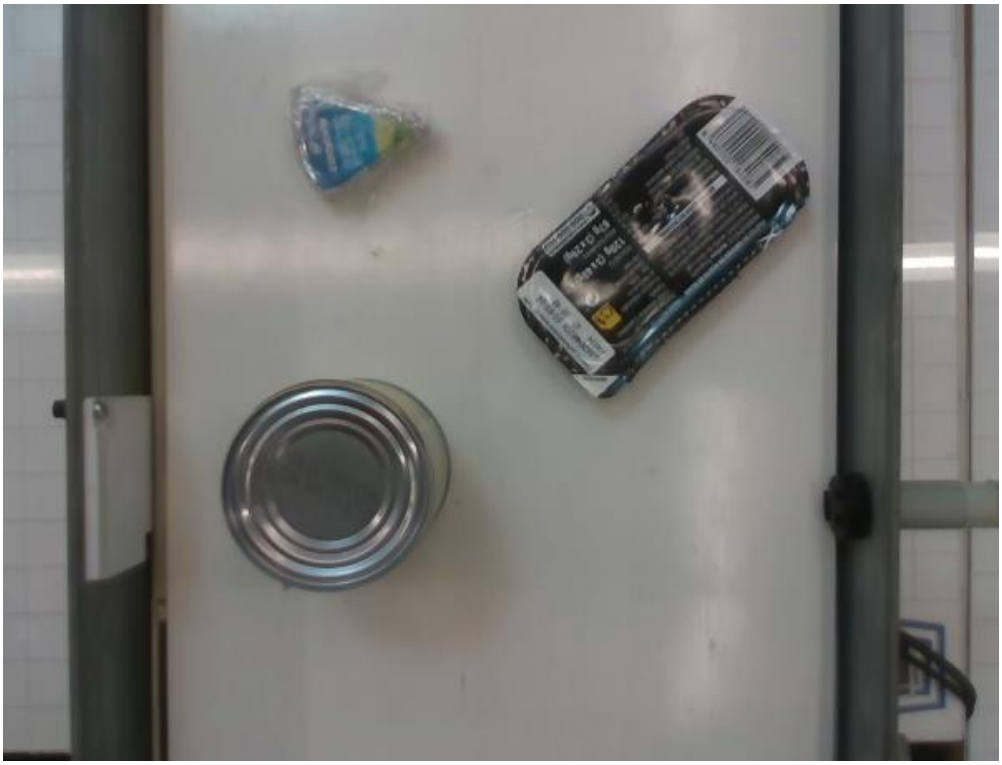


Figura 5.27 Captación de objetos en la cinta

Esta imagen necesitará un tratamiento para permitir a la función ‘label’ mencionada en el apartado anterior una correcta identificación de los objetos. El tratamiento consistirá en lo siguiente:

1. Se realizará una conversión de imagen RGB a imagen de escala de grises.

Figura 5.28 Imagen de objetos en escala de grises

2. Se realiza una detección de bordes por el método 'Sobel'.

Figura 5.29 Imagen de objetos sobel

3. Se convierte la imagen a binario a través de umbralización. El valor escogido ha sido obtenido a través de analizar los valores de gris de los objetos a analizar gracias al código 'ObtenerRGByNivelDeGris'.

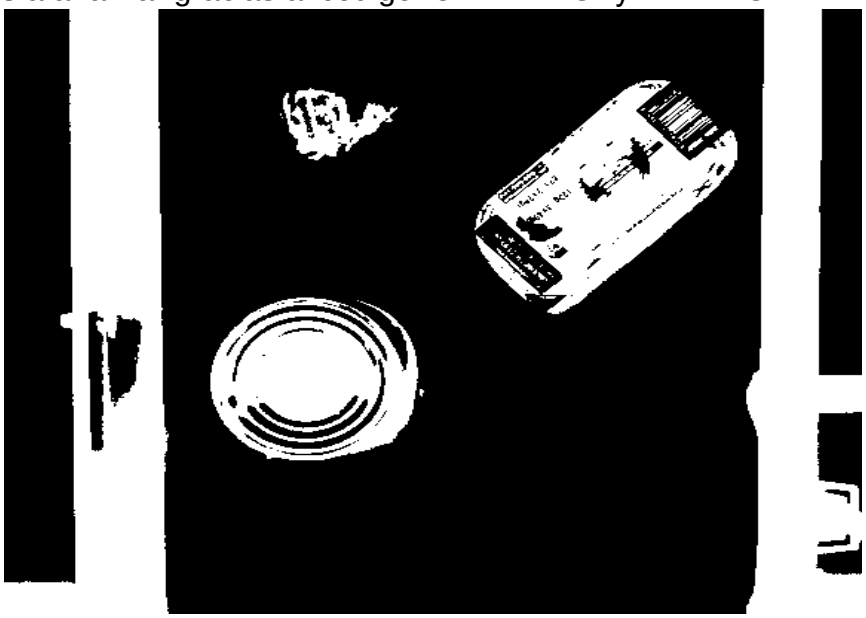


Figura 5.30 Imagen de objetos binarizada

4. Tal y como se puede apreciar los límites laterales de la cinta son captados por la cámara, para evitar que se identifiquen estos como objetos se realiza una operación de eliminación de bordes, siendo esta que todo aquello que tenga contacto con los límites de la imagen se elimine.

   Esto también es útil para eliminar aquellos objetos que puedan captarse en el límite de la imagen y por lo tanto se hayan podido captar sin encontrarse completos, ya que de esta forma se obtendrían sus centros de gravedad de forma errónea.

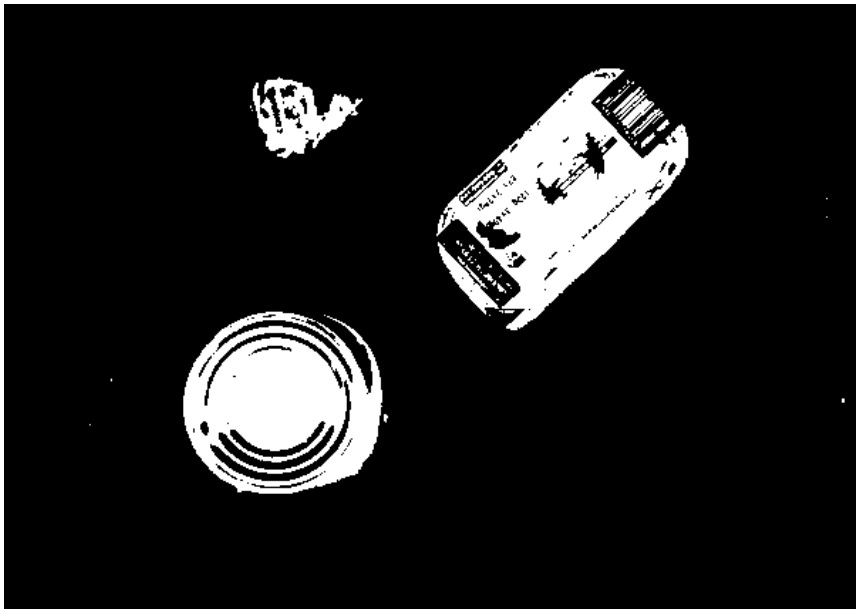


Figura 5.31 Imagen de objetos sin bordes

5. Para reducir el número de objetos pequeños no deseados como pueden ser sombras o manchas se aplica una operación de apertura. Para ello se aplica un *kernel* correspondiente a una matriz de 2*2 con una iteración.

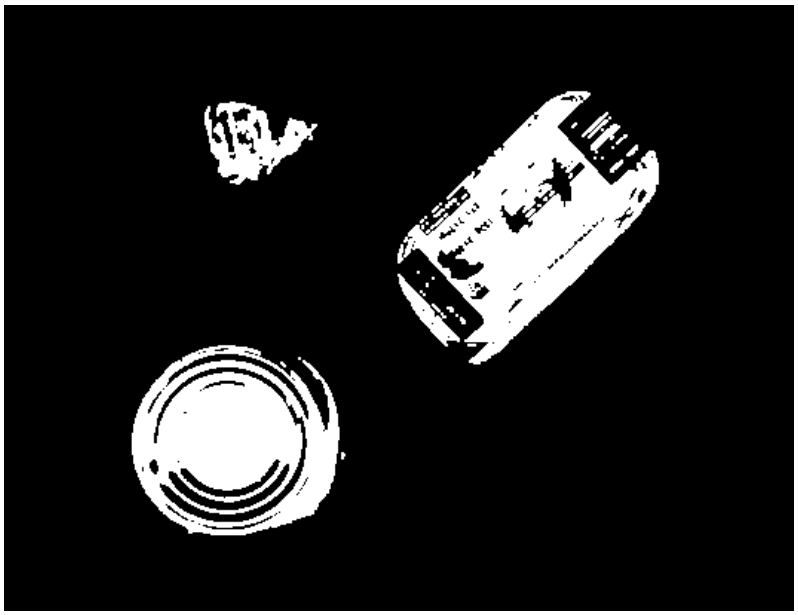


Figura 5.32 Imagen de objetos con operación de apertura

Debido a que se habrán reducido las áreas de los objetos, se realiza una dilatación para recuperar el tamaño anterior, en este caso con tres iteraciones.

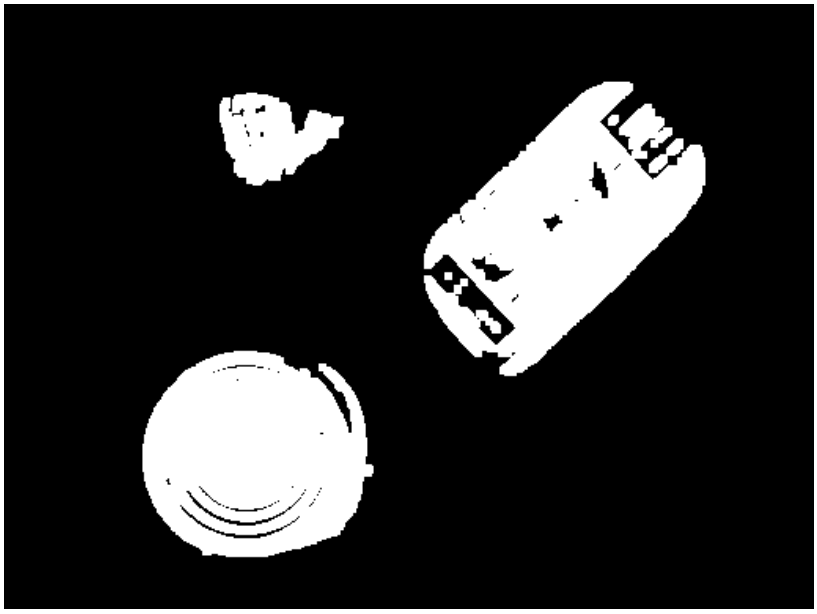


Figura 5.33 Imagen de objetos dilatada

6. A continuación, para rellenar los objetos como es el caso de los quesitos, los cuales tienen letras blancas y generan huecos dentro del objeto se realiza un cierre con el *kernel* de 2*2 y 20 iteraciones.

Figura 5.34 Imagen de objetos realizada una operación de cierre

7. Por último, se realiza una erosión para eliminar la dilatación previa al cierre

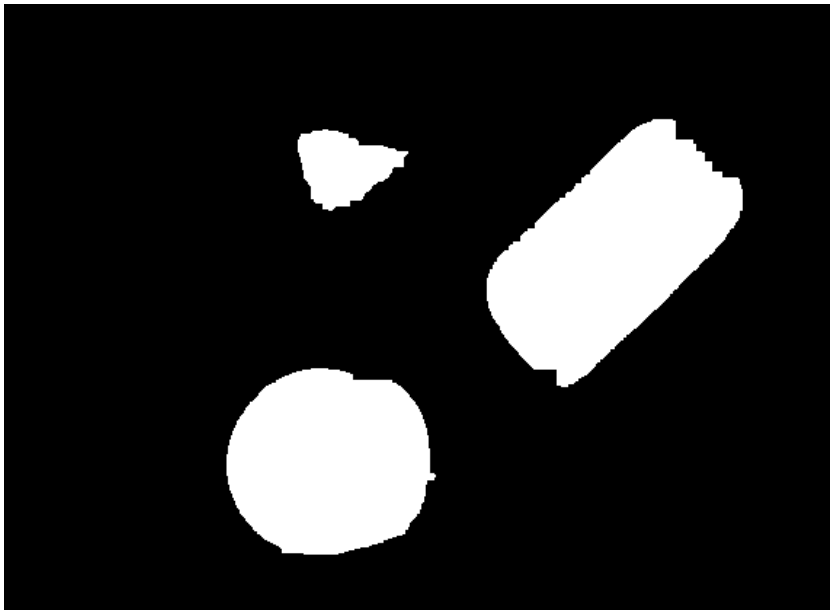


Figura 5.35 Imagen de objetos erosionada

Para la realización del tratamiento se ha empleado las librerías 'skimage.morphology' , 'skimage.segmentation' y 'skimage' .

Una vez obtenida la imagen final, se le aplica a esta la función 'label' para identificar los objetos. Tras ello se le aplica a la imagen obtenida la función 'regionprops' y contamos el número de elementos que existen en la imagen.

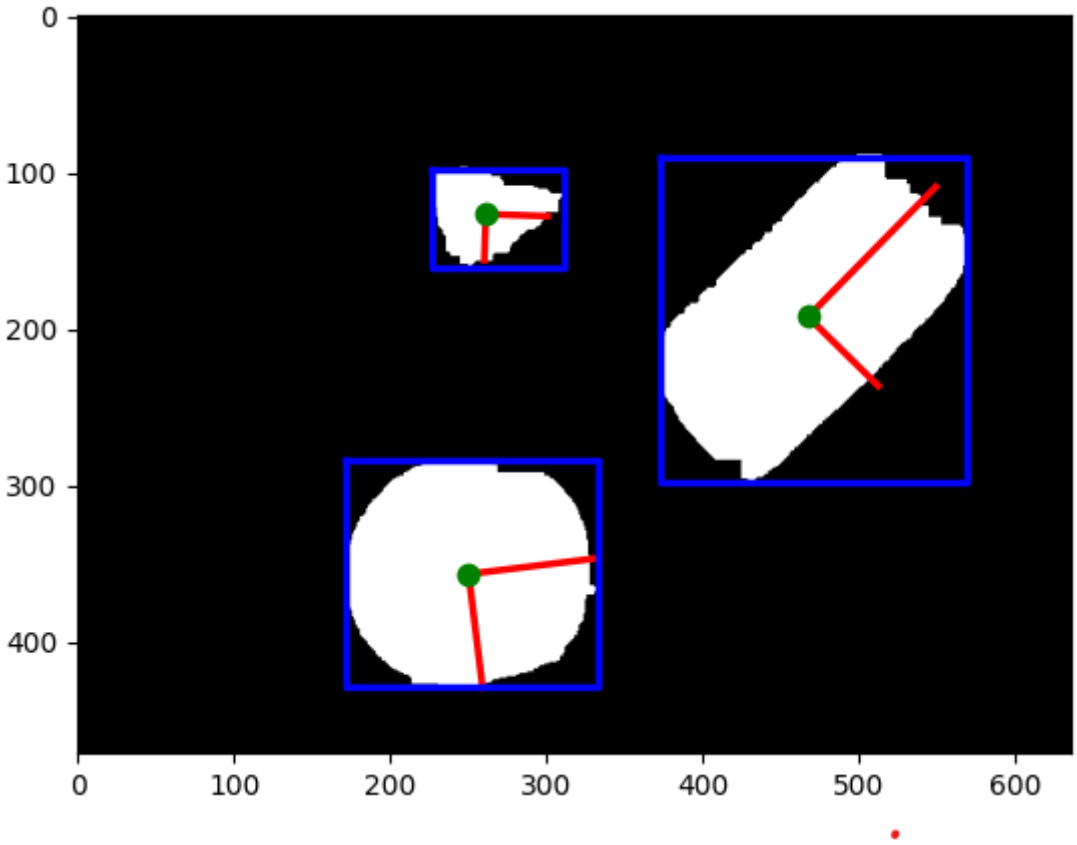
Para evitar contar elementos que no sean objetos, cada vez que se identifique un objeto de área menor a cierto nivel, se resta uno al número de objetos identificados.

Una vez obtenidos, de igual forma que se ha realizado con las pegatinas previamente, se obtienen las características de aquellos elementos que tengan un área superior a cierto nivel, siendo estos los objetos.

Las características que deseamos obtener son las siguientes: área, centro de gravedad, orientación y los ejes mayor y menor.

Para la identificación de cada objeto se hará uso de sus elementos físicos más característicos: los quesitos son los objetos más pequeños y con gran diferencia por lo que si el área del objeto es menor a cierto valor será uno de estos, en el caso de las latas de anchoas por su forma se utiliza la relación de su eje mayor con su eje menor ya que este último es mucho más pequeño que

el primero, y esto únicamente ocurre con este objeto. Por último, si no es ninguno de estos dos objetos, será una lata de maíz.




```
el número de objetos es: 3
La pieza 1 es una lata de anchoas
X= 467.24568044175277 Y= 192.02934627716422
Orientación= 45
La pieza 2 es un quesito
X= 261.05424954792045 Y= 126.36528028933093
Orientación= [178]
La pieza 3 es una lata de maíz
X= 250.0282562625659 Y= 356.5897951420964
Orientación= [-82]
```

Figura 5.36 Identificación de objetos

Una vez identificados se obtienen las características de cada elemento y en el caso de las latas de anchoa y de los quesitos es necesario aplicar una corrección a la orientación debido a que la función presenta dos problemas.

El primer problema es que la función tiene como referencia de 0 grados, 90 grados respecto la horizontal, por lo que es necesario realizar un offset para establecer un sistema de referencia más sencillo de entender. La segunda peculiaridad es que sus límites se encuentran entre -90 y +90 por lo que si el quesito se encuentra boca abajo el programa no hará distinción y le pondrá el mismo ángulo que tendría si estuviera boca arriba, ya que no es capaz de mostrar un ángulo superior a 180º respecto la horizontal.

En el caso de las latas de anchoas al ser simétricas el segundo problema no ocurre, ya que no existe diferencia al aplicarle un giro de 180º. Sin embargo, el primero, sí afecta tal y como se procede a mostrar.

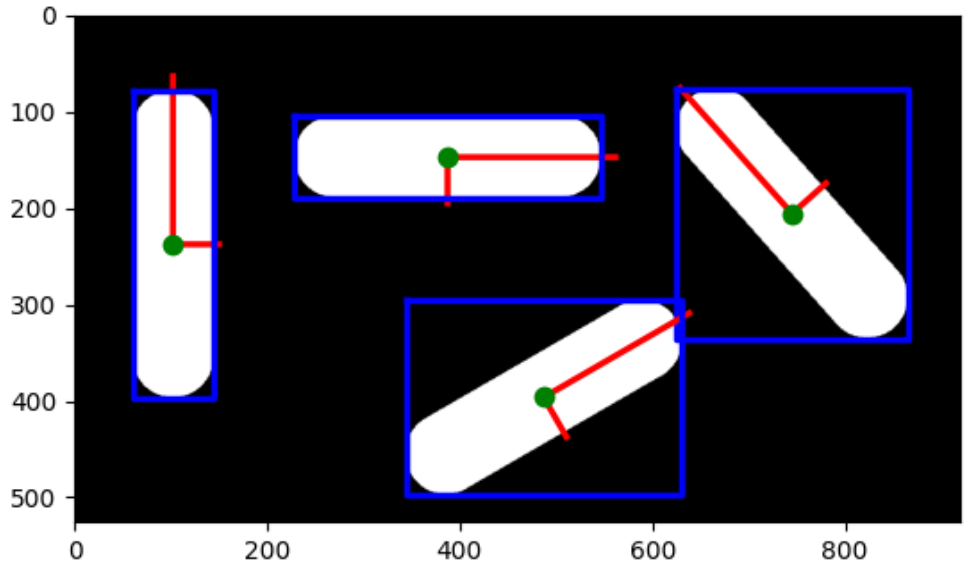


Figura 5.37 Representación orientación latas de anchoas

```
La pieza 1 es una lata de anchoas
X= 745.039368213228 Y= 205.59459032576507
Orientación= 41
La pieza 2 es una lata de anchoas
X= 102.03379861982434 Y= 237.45541875784l9
Orientación= [0]
La pieza 3 es una lata de anchoas
X= 387.29712560291756 Y= 147.02992039527862
Orientación= [-89]
La pieza 4 es una lata de anchoas
X= 487.0489692757286 Y= 396.218584629966
Orientación= [-60]
```

```
La pieza 1 es una lata de anchoas
X= 745.039368213228 Y= 205.59459032576507
Orientación= 131
La pieza 2 es una lata de anchoas
X= 102.03379861982434 Y= 237.45541875784l9
Orientación= [90]
La pieza 3 es una lata de anchoas
X= 387.29712560291756 Y= 147.02992039527862
Orientación= [1]
La pieza 4 es una lata de anchoas
X= 487.0489692757286 Y= 396.218584629966
Orientación= [30]
```

Figura 5.38 Orientaciones de las latas de anchoa de la figura 5.37 Pre-corrección/Post corrección

Las latas de anchoas identificadas han seguido el siguiente orden: primero la lata de más a la derecha, tras esto la lata más a la izquierda, la tercera la que se encuentra en el centro superior y la cuarta la restante. Tal y como se puede

apreciar, la segunda lata marca el 0 de la función y para la aplicación se desea que sea 90º, que con un simple offset de +90º se puede obtener.

En el caso de los quesitos se presenta la misma situación, además, es necesario obtener un rango de 0 a 360º. A continuación, se procede a mostrar un ejemplo del problema y su solución.

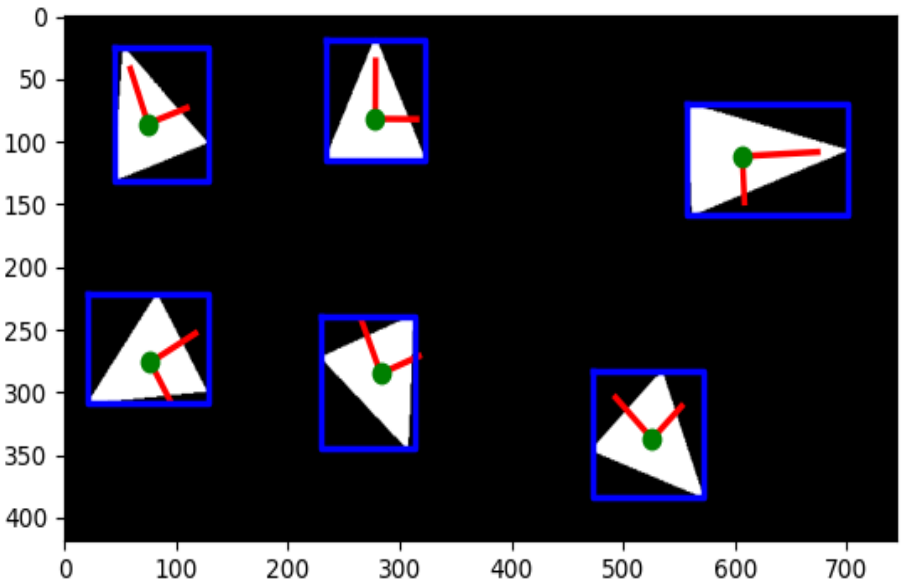


Figura 5.39 Representación orientación quesitos

```
La pieza 1 es un quesito
X= 278.0812696983341 Y= 81.72467357046375
Orientación= 0
La pieza 2 es un quesito
X= 74.60318886144172 Y= 85.52256905456996
Orientación= [19]
La pieza 3 es un quesito
X= 606.6133684863523 Y= 111.53024193548387
Orientación= [-87]
La pieza 4 es un quesito
X= 76.76579006518318 Y= 276.04944931445266
Orientación= [-60]
La pieza 5 es un quesito
X= 282.91268237934906 Y= 285.00987654320 99
Orientación= [22]
La pieza 6 es un quesito
X= 525.8730407523511 Y= 337.1605463502015
Orientación= [44]
```

```
La pieza 1 es un quesito
X= 278.0812696983341 Y= 81.72467357046375
Orientación= 270
La pieza 2 es un quesito
X= 74.60318886144172 Y= 85.52256905456996
Orientación= [289]
La pieza 3 es un quesito
X= 606.6133684863523 Y= 111.53024193548387
Orientación= [183]
La pieza 4 es un quesito
X= 76.76579006518318 Y= 276.04944931445266
Orientación= [30]
La pieza 5 es un quesito
X= 282.91268237934906 Y= 285.00987654320 99
Orientación= [112]
La pieza 6 es un quesito
X= 525.8730407523511 Y= 337.1605463502015
Orientación= [134]
```

Figura 5.40 Orientaciones de los quesitos de la figura 5.39 Pre-corrección/Post corrección

Tal y como se puede apreciar en la figura 5.39, el primer quesito identificado tiene un ángulo de 0º, este corresponde con el que se encuentra en la parte central superior, con la punta mirando hacia arriba. Este es el punto de referencia de la función y aquellos objetos que giren en el sentido de las agujas del reloj tendrán un ángulo negativo y en el sentido contrario un ángulo positivo.

Es por esto por lo que para obtener un ángulo de 0º cuando un quesito se encuentre tumbado con la punta hacia la izquierda es necesario realizar un offset de -90º.

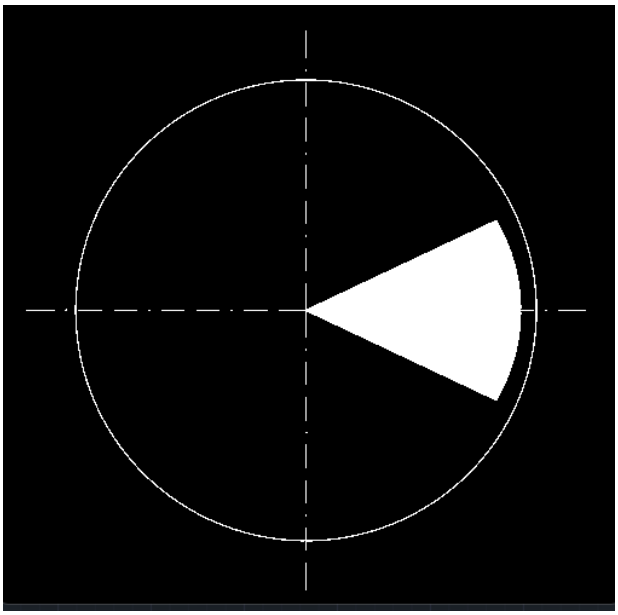


Figura 5.41 Sistema de Referencia establecido para los Quesitos

Además, tal y como se puede observar en la figura previamente mencionada y en la 5.40 la función no hace distinción de la geometría del quesito por lo que es necesario saber cuándo el quesito se encuentra apuntando hacia abajo o hacia arriba. Para esto se crea un punto en la dirección del ángulo que proporciona la función desde el centro de gravedad obtenido, colocándolo a cierta distancia.

Esto permitirá saber si en la imagen binaria el píxel tiene valor de 1 o 0, en caso de que el pixel tenga valor de 1 significa que el quesito se encuentra en esa dirección por lo que el sentido del ángulo es correcto, en cambio si el valor es 0 significa que no hay objeto en ese sentido y por lo tanto es necesario cambiarlo con un offset de 180º.

Esta corrección es posible verla en la figura 5.39 en el quesito situado en la parte central inferior, el cuál corresponde al quesito número 5 de la figura 5.40.

En este, el ángulo proporcionado por la función es 22, sin embargo, es necesario aplicarle la primera corrección de -90º para situarlo en el sistema de referencia deseado y tras ellos la segunda corrección de +180º ya que el quesito se encuentra apuntando hacia abajo.

Todo el proceso explicado y el envío al robot de todos los datos extraídos de cada objeto detectado se realiza en la función 'Análisis ()' del código principal de Python.

5.2.2.3 Comunicación

Para la comunicación con el robot desde Python se empleará la librería 'socket'.

Lo primero para establecer la comunicación es necesario que tanto el ordenador como el robot se encuentren en la misma red, en el caso del proyecto esto se realiza al conectarse ambos al mismo switch.

Tras esto es necesario establecer la dirección IP y puerto de comunicación del servidor, para la realización del proceso se ha decidido que la unidad de control del robot hará de servidor, esta cuenta con la siguiente dirección IP: '192.168.1.100' y el puerto de comunicación es el 30000.

Para la recepción de información se empleará la función 'sock.recv()' lo que permite al ordenador esperar hasta que le llegue la información del robot y este le indique que ya ha enviado todo el mensaje.

Para el envío de información se utilizará 'sock.send()', el cual se utilizará para enviar cadenas, debido a que la versión de Python utilizada es de Python 3 previamente al envío es necesario codificar la información mediante la función 'encode()'.

Tal y como se explicará posteriormente en el programa de RobotStudio será necesario conocer la posición exacta dentro de la cadena enviada desde Python de la información que se desea comunicar, es por esto por lo que se ha decidido establecer el siguiente método para comunicar los centros de gravedad de las condiciones iniciales de empaquetado:

1. Como es necesario conocer la posición en la cadena de cada elemento, esta se fijará a 3 cifras, para esto a cada número que el ordenador quiera enviar se le sumará 100, asegurando que siempre y cuando el valor se encuentre en el intervalo [-1000,-200] ∪ [0,899] este siempre tendrá una longitud de 3 cifras.

Debido al espacio de trabajo y los límites del robot, este no emplea ningún valor en X o Y superior a 800 o inferior a -800, por lo que es posible el uso de este método.

2. En el caso de que el valor pertenezca al intervalo (-200,0) se le sumará 300, obteniendo de nuevo una longitud de 3 cifras.

3. Es necesario que el robot al recibir la información sepa qué número se le ha sumado, de forma que pueda restarlo y tener la información original que se quiere enviar, esto se indicará mediante los símbolos '*' para los valores a los que se le ha sumado 100 y '?' para los valores a los que se le ha sumado 300.

4. Por último, el robot necesita saber el signo del valor enviado, es por ello por lo que en caso de que sea positivo se le añadirá a la cadena el símbolo '+' y en caso de ser negativo la conversión de entero a carácter ya lo aporta.

Con esto establecido, si Python deseara comunicar al robot que los centros de gravedad X e Y de la pegatina roja se encuentran en las posiciones +300 y -198 respectivamente, la información enviada en forma de cadena sería la siguiente: *+400?+102. Este método entonces asegura que los centros de gravedad de las pegatinas y el círculo de los quesitos tengan una longitud de 5 caracteres cada elemento del centro.

En el caso de las alturas es suficiente con sumar 100 ya que estas no pueden ser negativas.

De forma similar, en la comunicación de la posición de los objetos debido a la posición de la cinta en el espacio de trabajo los valores de los centros de gravedad se les suma 200 a previamente a su envío y se indica su signo de la misma forma que se ha explicado previamente. En el caso de la orientación ocurre lo mismo que con la distancia, puesto que esta nunca es negativa. Además, al final se añade un número que indique que tipo de objeto es.

Entonces un envío completo de información de posición de un objeto podría ser: +430-3505701453. Esta información implicaría que el objeto posee

un centro de gravedad en X de 230, de Y de -550, una distancia al objeto de 470, una orientación de 45º y que es un quesito.

5.2.3 RobotStudio

5.2.3.1 Comunicación

Para la comunicación con el ordenador se emplearán las herramientas de socket que tiene RobotStudio.

Primeramente, es necesario crear un servidor con la dirección IP del robot y el puerto de comunicación, en el caso del IRB140 del laboratorio son "192.168.1.100" y 30000 tal y como se había mencionado previamente.

A continuación, se establece el servidor a la escucha esperando que se conecte el ordenador y una vez se detecta se acepta la conexión.

Para la recepción de información se utilizará la función ‘SocketReceive’ y para el envío de esta se utilizará ‘SocketSend’. La información recibida y enviada será de tipo cadena.

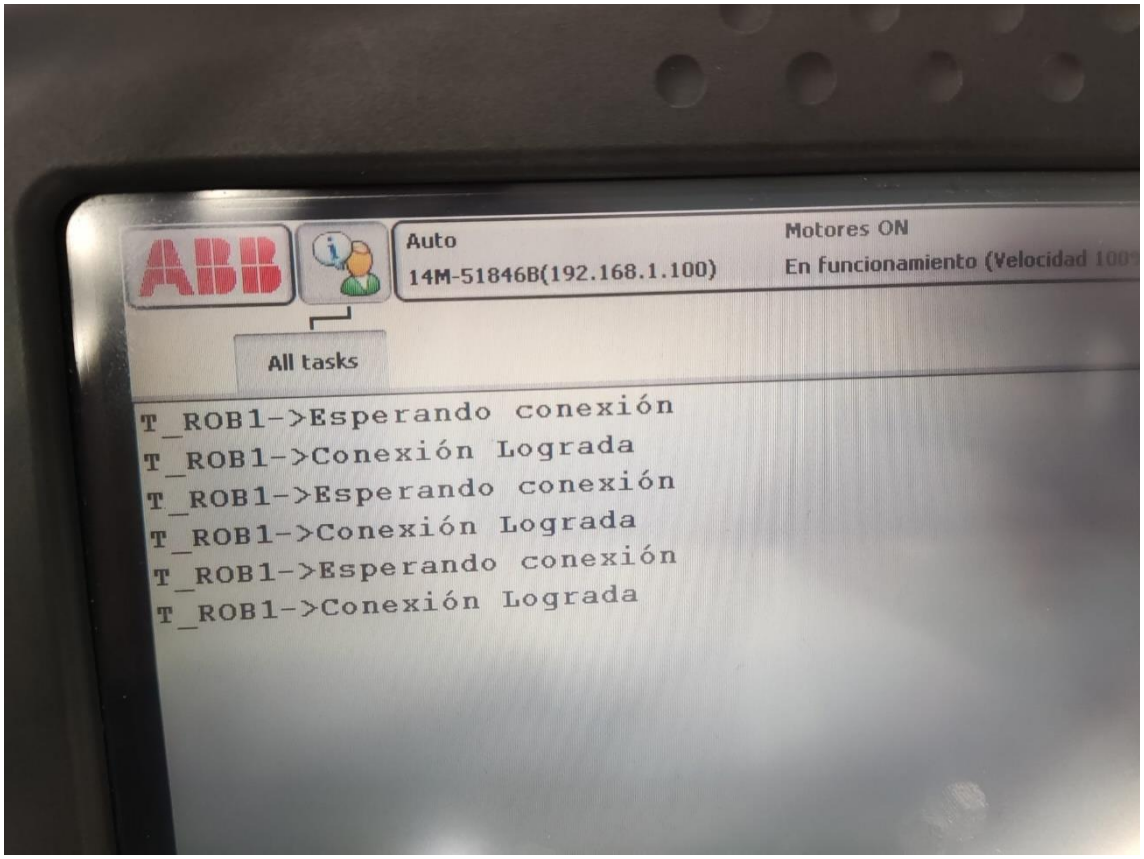


Figura 5.42 Conexión Robot/Ordenador

5.2.3.2 Condiciones iniciales de empaquetado

Para el establecimiento de las nubes de puntos donde se desean dejar los objetos se ha creado una función denominada 'CondicionesIciales()' en el código principal de RobotStudio la cual se encuentra compuesta de los siguientes pasos:

1. Se recibe una cadena con todos los valores de los centros de gravedad de las pegatinas de colores y el círculo de los quesitos, también con la distancia a los mismos.

2. Se obtienen de la cadena recibida cada elemento importante gracias a su posición mediante la función StrPart(variable, posición inicial, posición final) donde se indica la variable de la que se pretende obtener

información y las posiciones en la cadena donde inicia y termina la parte de la cadena que se desea extraer.

Los elementos importantes son: el símbolo que identifica si un número ha sido sumado +300 o +100 y el número correspondiente a cada centro de gravedad y altura.

3. Una vez separado, se transforma el número recibido como cadena a tipo entero, para ello se emplea la función 'StrToVal(str,val)', la cual para realizar el cambio es necesario crear una variable booleana que será el resultado de la función y tiene como entradas la cadena a transformar y el valor donde se almacenará el valor de la cadena en tipo entero.

   Por ejemplo:

   okCDGRojoX:=StrToVal(StringCDGRojoX,CDGRojoX)

   En este caso se ha creado la variable booleana 'okCDGRojoX' y se ha convertido la cadena 'StringCDGRojoX' en valor entero y almacenado en 'CDGRojoX'

4. Una vez obtenido se aplica una corrección al número obtenido en función del símbolo que se encuentra delante de este, si es una interrogación se resta 300 y si es un asterisco se resta 100.

5. Se crean matrices de puntos de cada objeto en función de sus centros de gravedades obtenidos, y sus características físicas como son el diámetro del objeto o su altura.

```
1. Maiz{1}:=[[CDGAzulX+DiaMaiz,CDGAzulY,(AltColores+AltMaiz)],[0.014
   68,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
2.
3. Maiz{2}:=[[CDGAzulX,CDGAzulY,(AltColores+AltMaiz)],[0.01468,-
   0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];


4. Maiz{3}:=[[CDGAzulX-
   DiaMaiz,CDGAzulY,(AltColores+AltMaiz)],[0.01468,-0.53568,-
   0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
5.
   Maiz{4}:=[[CDGAzulX+DiaMaiz,CDGAzulY,(AltColores+2*AltMaiz)],[0.0
   1468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
6.
```

```
7. Maiz{5}:=[[CDGAzulX,CDGAzulY,(AltColores+2*AltMaiz)],[0.01468,-
   0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
8.
9. Maiz{6}:=[[CDGAzulX-
   DiaMaiz,CDGAzulY,(AltColores+2*AltMaiz)],[0.01468,-0.53568,-
   0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
10.
11. Anchoas{1}:=[[CDGVerdeX,CDGVerdeY-
    RadioMenorAnchoas,AltColores+AltAnchoas],[0.01468,-0.53568,-
    0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
12.
13. Anchoas{2}:=[[CDGVerdeX,CDGVerdeY,AltColores+AltAnchoas],[0.01468
    ,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
14.
    Anchoas{3}:=[[CDGVerdeX,CDGVerdeY+RadioMenorAnchoas,AltColores
    +AltAnchoas],[0.01468,-0.53568,-
    0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
15.
16.
    Quesitos{1}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(0),CDGQu
    esitosY+RadioCirculoQuesitos*0.65*Cos(0),(AltCirculo+AltQuesitos)],[0.01
    468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
17.
    Quesitos{2}:=[[CDGQuesitosX+RadioCirculoQuesitos*Sin(60),CDGQuesito
    sY+RadioCirculoQuesitos*0.65*Cos(60),(AltCirculo+AltQuesitos)],[0.01468,
    -0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
18.
    Quesitos{3}:=[[CDGQuesitosX+RadioCirculoQuesitos*Sin(120),CDGQuesit
    osY+RadioCirculoQuesitos*0.65*Cos(120),(AltCirculo+AltQuesitos)],[0.014
    68,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
19.
    Quesitos{4}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(180),CDG
    QuesitosY+RadioCirculoQuesitos*0.65*Cos(180),(AltCirculo+AltQuesitos)],
    [0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
20.
    Quesitos{5}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(240),CDG
    QuesitosY+RadioCirculoQuesitos*0.65*Cos(240),(AltCirculo+AltQuesitos)],
    [0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
21.
    Quesitos{6}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(300),CDG
    QuesitosY+RadioCirculoQuesitos*0.65*Cos(300),(AltCirculo+AltQuesitos)],
    [0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]]
```

Figura 5.43 Creación de los puntos de empaquetado

Previamente a la llamada de esta función el robot se habrá desplazado mediante un movimiento de tipo 'MoveJ' hasta un punto previamente establecido para la calibración de la cámara con el robot en la zona donde se empaquetan los objetos.

5.2.3.3 Pick and place de objetos

Primeramente, el robot debe haber creado las matrices de puntos de empaquetado, tras esto se desplaza al punto de calibración cámara-robot de la cinta y avisa al ordenador de que ha llegado.

Una vez avisado el ordenador, se activa la cinta y se espera a que llegue un objeto, cuando este ha llegado y es detectado por el sensor foto eléctrico del final de carrera la cinta se para y se llama a la función creada 'MoverObjetos()' en el programa principal de RobotStudio. Esta función consiste en la realización de los siguientes pasos:

1. El robot le envía al ordenador una cadena preguntando cuantos objetos hay y este recibe como respuesta dicho número en forma de cadena, para después transformarlo en entero.

2. Se demanda al ordenador la información de uno de los objetos detectados que no se ha procesado todavía.

   Una vez recibida se le aplica una corrección a cada valor, ya que tal y como se ha explicado en el apartado 5.2.2.3 a los centros de gravedad de los objetos es necesario restarles 200 y a la altura y orientación 100. Para obtener los valores de la cadena se emplea la función 'StrToVal(str,val)' de la misma forma que se ha explicado previamente.

3. Se crea el punto en el espacio del robot en el que se encuentra el objeto actual con las coordenadas recibidas.

4. Se comprueba qué tipo de objeto es para sumar una unidad a la cantidad de objetos empaquetados de ese en particular. Además, si el objeto es una lata de anchoas o un quesito se obtiene el ángulo de giro del robot sobre el eje Z, resultante de la diferencia entre el ángulo del objeto y el ángulo deseado que ha sido recibido. En caso de ser una lata de maíz o una lata de refresco el ángulo de giro es 0, para las latas de anchoas será

90º el ángulo deseado y para los quesitos, al ser 6 quesitos serán 0, 60, 120, 180, 240 y 300.

En función de qué objeto es y cuantos se han empaquetado se establece el punto donde se debe dejar el objeto actual mediante las matrices de puntos previamente creadas.

5. Se realizan los siguientes movimientos y acciones:

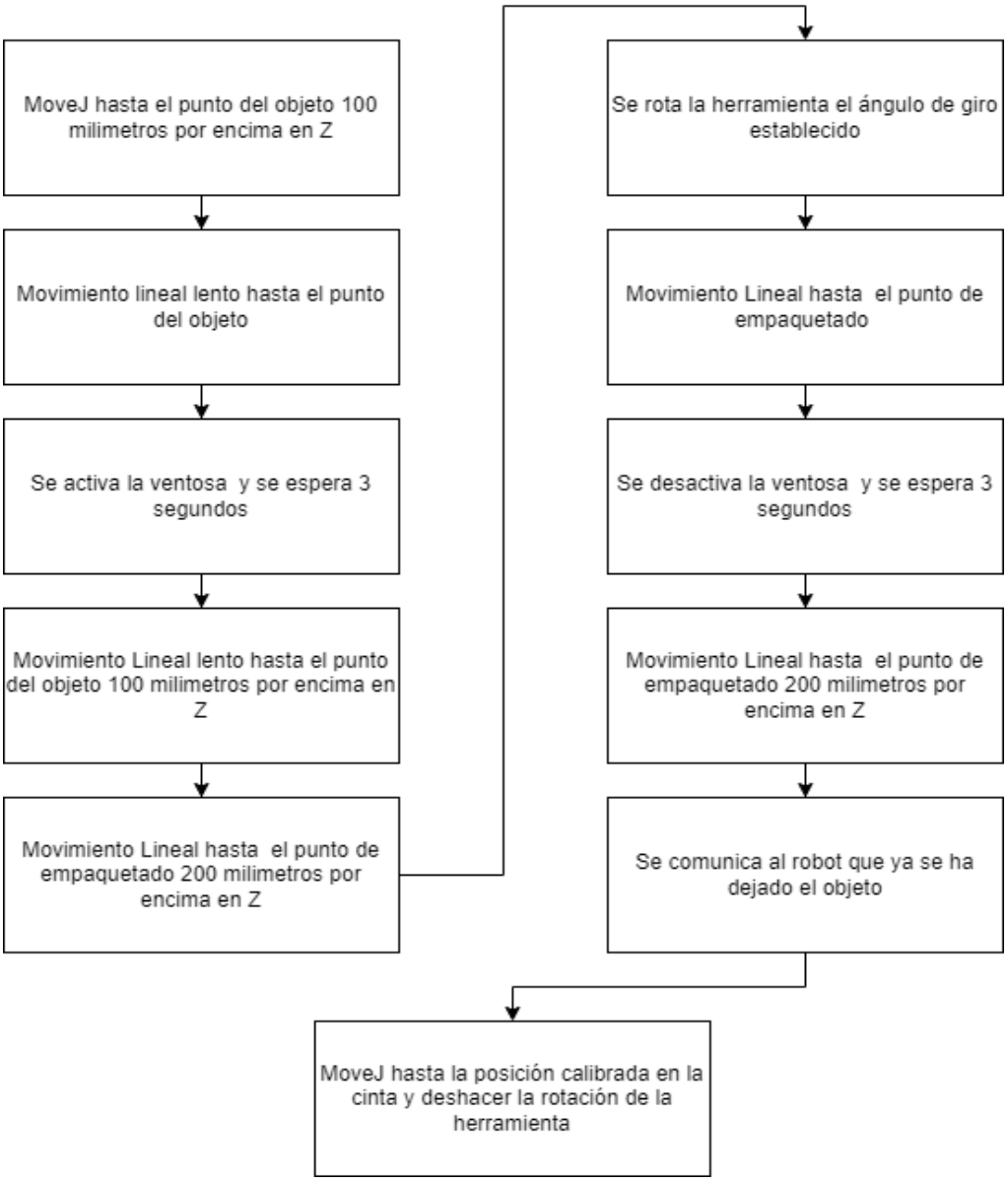


Figura 5.44 Movimientos de IRB140 para pick and place

6. Si no se han procesado todos los objetos detectados, se regresa al paso número 2.

Una vez finalizada la función, si todavía no se han empaquetado 6 latas de maíz, 3 latas de anchoas y 6 quesitos la cinta vuelve a activarse y se repiten todos los pasos explicados previamente a partir de la activación de la cinta.

En el caso de que sí se hayan colocado todos los objetos se cortan las comunicaciones y finaliza el proceso.

6. Resultados, conclusiones y mejoras del sistema

6.1. Resultados

Se han realizado múltiples empaquetados de productos cambiando las posiciones de cada uno de los productos. De todos los empaquetados se ha podido comprobar como el sistema es capaz de identificar a la perfección las marcas de empaquetado siempre, independientemente de la iluminación natural que depende de la hora del día.

También se puede afirmar que el sistema es capaz de identificar y colocar las latas de anchoas y de maíz siempre y de forma correcta, sin embargo, los quesitos presentan un problema debido a dos factores:

El primero, su superficie, la cual la ventosa no es capaz de absorber debido a que no es completamente lisa, para solucionarlo se utilizó papel de film que enrollaba a los quesitos.

Pese a que esta solución permitía succionar a la ventosa, en ocasiones se atascaba el papel y pese a dejar de succionar el quesito se quedaba enganchado al robot. Además, el tamaño añadido del papel impidió que no pudiera realizarse el empaquetado de 8 quesitos en lugar de 6, al no caber estos.

El segundo, el tamaño de la ventosa y el quesito. Al ser la ventosa tan grande en comparación del tamaño del objeto, se requería de una precisión absoluta en un punto muy concreto para que la ventosa fuera capaz de succionar el quesito. Esto además provocó que fuera necesario un movimiento más lento del objeto para evitar que este cayera.

A continuación, se presenta el resultado de uno de los experimentos.

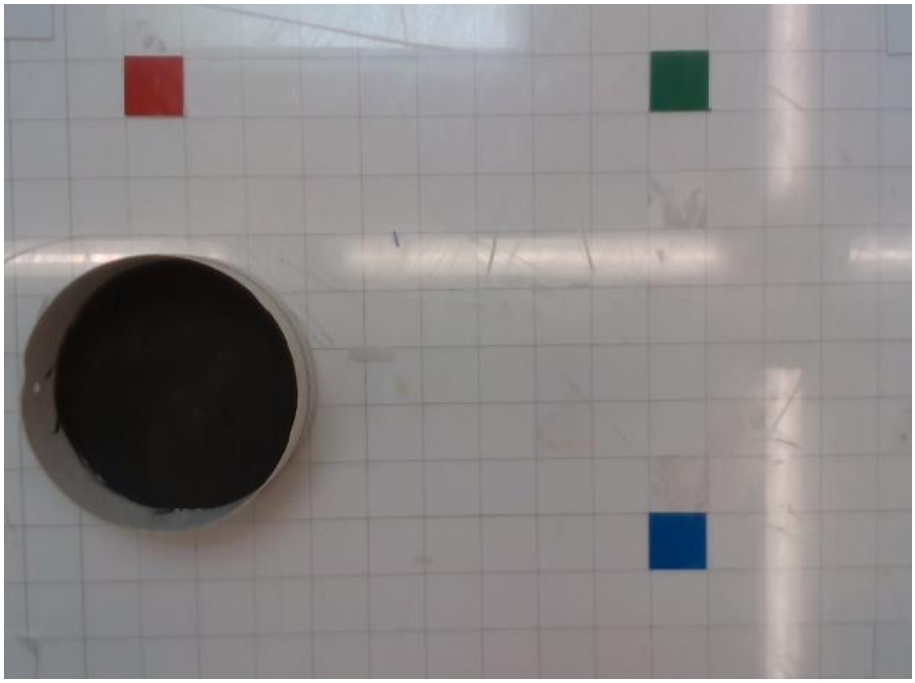


Figura 6.1 Condiciones de Inicio

Figura 6.2 Resultado del empaquetado

6.2. Conclusiones

Tras la conclusión del proyecto, se puede constatar que se ha logrado diseñar un algoritmo de visión artificial eficiente que cumpla con los objetivos establecidos al inicio del proyecto: la identificación de unas marcas de empaquetado, la identificación de distintos objetos y la extracción de sus características en el espacio.

Sin embargo, en cuanto al empaquetado mediante el robot tras los múltiples experimentos se llegó a la conclusión de que el sistema no era viable para el empaquetado de quesitos, debido a que este presentaba varios fallos y tenía un porcentaje de finalización del proceso de principio a fin sin fallos del 42 %, ya que se realizaron un total de 14 experimentos con todos los componentes juntos y tan solo en 6 los quesitos no supusieron un problema.

Por otro lado, las latas de maíz y las latas de anchoas sí han presentado resultados positivos y se ha concluido que el sistema podría empaquetar estos productos de forma rápida y sin error.

A lo largo del proyecto se han encontrado diversos problemas inesperados a parte del problema con la ventosa y los quesitos. Originalmente el

sistema podría empaquetar un cuarto objeto: latas de refresco, en la pegatina roja. Sin embargo, pese a que el sistema fue diseñado para ello no fue posible debido a que para su implementación era necesario medir la distancia al objeto para diferenciarlo de las latas de maíz

La cámara es capaz de medir la profundidad, sin embargo, debido al material de la lata, este no permitía a la cámara medir correctamente, por lo que al final no fue posible su inclusión. Pese a que no ha sido posible su inclusión debido a este problema, sí se pudo comprobar que era posible empaquetarlo con el robot. En los códigos adjuntos no se ha borrado todo lo relacionado a este objeto, se ha decidido no eliminarlo, porque fue trabajo realizado y que se podría aplicar con otro tipo de medición de distancia.

También se presentaron problemas con la iluminación del laboratorio a la hora de identificar los objetos, ya que la fuente principal de este es natural y por lo tanto variable con la hora del día, sin embargo, al final fue posible solucionarlos.

Resumiendo, en términos generales, los softwares desarrollados realizan de forma satisfactoria los objetivos del proyecto. No obstante, se han hallado varias sorpresas que no se esperaban, como el mal funcionamiento de la medida de profundidad y la ventosa con los quesitos.

Personalmente, he encontrado el proyecto muy gratificante, me ha permitido ahondar más en el campo del grado que más me ha gustado y en el que me quiero especializar, me ha permitido aprender un lenguaje de programación nuevo desde cero y aplicarlo a los conocimientos de visión artificial adquiridos. También me ha permitido manipular un robot antropomórfico por primera vez y en una situación real, dándome la oportunidad de darme cuenta de la dificultad de esto, de los pequeños problemas que se presentan y de cómo solucionarlos o adaptarse a ellos, lejos de simulaciones que poco tienen que ver. Por último, me ha brindado el lujo de poder decir que este es el camino que quiero seguir y que todo lo recorrido ha sido muy útil.

6.3. Mejoras del sistema

Para una mejor implementación del sistema se podrían realizar una serie de mejoras, como el uso de otra ventosa que necesite menor superficie para realizar el vacío, eliminando el segundo problema de los quesitos.

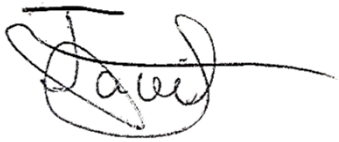
También se podría cambiar la cámara por una de condiciones iguales, salvo la medición de profundidad por un método distinto que permitiera incluir las latas de refresco en el proceso.

Además, se podría mejorar la iluminación del laboratorio por una menos variable permitiendo que los valores RGB y en escala de grises sean más estables en cualquier momento del día.

Estas mejoras no se han podido implementar previamente debido a que se encuentran fuera del alcance del proyecto.

VALENCIA, JULIO 2023

Javier Gamir Artesero



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

ANEJO Nº1: CÓDIGOS

Índice

1. Python

1.1. Código principal

```
import socket
import time
import math
import numpy as np
import cv2
import pyrealsense2

from realsense_depth import *
from skimage import filters
from skimage.measure import label, regionprops, regionprops_table
from skimage.segmentation import clear_border
from skimage.morphology import closing

kernel=np.ones((2,2),np.uint8)

# Inicializo la camara
dc = DepthCamera()

def TranfsformacionCondicionesIniciales(x,y):
    xh=0.0055*x-0.7209*y+754.495
    yh=-0.6695*x+0.0157*y+272.872
    h=0.0000353*x-0.000099328*y+1.089
    xc=xh/h
    yc=yh/h
```

```
    yc=yc
    return xc,yc

def TranfsformacionCinta(x,y):
    xh=-0.0022*x-0.5996*y+520.4481
    yh=-0.6318*x+0.0000050652*y-286.5
    h=0.0*x-0.0*y+1
    xc=xh/h
    yc=yh/h
    return xc,yc

def SacarCDG(image):
    #Labelear objetos
    ret,thresh= cv2.threshold(image,0,255,cv2.THRESH_BINARY_INV)
    labelled_image=label(thresh)

    regions = regionprops(labelled_image)
    for props in regions:
        #Obtengo el centro de gravedad de la figura y su orientación
        if(props.area>1000):
            y0, x0 = props.centroid
            x0,y0=TranfsformacionCondicionesIniciales(x0,y0)

            if(x0>-200 and x0<0):
                x0=int(x0)+300
                if(x0>0):
                    x0="?"+"+"+str(x0)
```

```
    else:
        x0="?"+str(x0)
else:
    x0=int(x0)+100
    if(x0>0):
        x0="*"+"+"+str(x0)
    else:
        x0="*"+str(x0)


if(y0>-200 and y0<0):
    y0=int(y0)+300
    if(y0>0):
        y0="?"+"+"+str(y0)
    else:
        y0="?"+str(y0)
else:
    y0=int(y0)+100
    if(y0>0):
        y0="*"+"+"+str(y0)
    else:
        y0="*"+str(y0)

return(x0+y0)
```

```
def Analisis (img,depth_frame):

        #Labelear objetos
    labelled_image=label(img)

    #Obtengo las regioprops (centroide 0=y y centroide 1=x)
    regions = regionprops(labelled_image)

    numerodeobjetos=len(regions)
    for props in regions:
        print(props.area)
        #Elimino posibles errores de detección de pequeñas partículas
        if(props.area<500):
            numerodeobjetos=numerodeobjetos-1
    numerodeobjetos=str(numerodeobjetos)
    sock.send(numerodeobjetos.encode())
    print('El numero de objetos es: '+ numerodeobjetos)

    for props in regions:
        if(props.area>500):
            PedidaInfo=sock.recv(1024);
            print (PedidaInfo)
            ret, depth_frame, frame = dc.get_frame()
            cv2.imshow("Estado del Trabajo",frame)
            cv2.waitKey(5)
```

```
#Obtengo el centro de gravedad de la figura y su orientación
y0, x0 = props.centroid
print(str(x0)+" "+ str(y0))
orientation = props.orientation
distancia = depth_frame[int(y0), int(x0)]

#Diferencio entre las posibles piezas
if (props.area<4000):
    pieza=3
    CDGZ=470
    #Quesito

elif (props.axis_major_length/props.axis_minor_length>1.5):
    pieza=2
    CDGZ=472
    #Anchoas

elif(props.area>5000):
    pieza=4
    CDGZ=528
    #Maiz



'''
elif(distancia<370):
    pieza=1
```

```
    CDGZ=569
    #Refresco
'''
```

#Obtengo puntos para comprobar si el triángulo está boca arriba o boca abajo

```
x2 = x0 - math.sin(orientation) * 0.5 * props.axis_major_length
y2 = y0 - math.cos(orientation) * 0.5 * props.axis_major_length

x2=int(x2)
y2=int(y2)

#Obtengo los cdg y orientaciones

orientacion=orientation*180/3.1416
if(pieza==2):
    orientacion=orientacion+90
```

if(pieza==3): #La pieza es un triángulo y debo comprobar si está boca abajo, en tal caso le sumo 180º a la orientación obtenida, si el ángulo es negativo, le sumo 360 para hacerlo positivo.

```
    #Y siempre le resto 90 grados para que me coincida que 0 grados es un quesito puesto en la horizontal con la punta apuntando a la izquierda
    orientacion=orientación-90
    if(img[y2,x2]==0):
        orientacion=orientacion+180
    elif(orientation*180/3.1416<0):
        orientacion=orientacion+360
```

```
#Hacer la transformada
xc,yc=TranfsformacionCinta(x0,y0)

xc=int(xc)+200
yc=int(yc)+200
distancia=int(distancia)+100
if(xc>0):
    xc="+"+str(xc)
else:
    xc=str(xc)

if(yc>0):
    yc="+"+str(yc)
else:
    yc=str(yc)

orientacion=int(orientacion)

CDGX=xc
CDGY=yc
CDGZ=str(CDGZ+100)
orientacion=str(orientacion+100)
Etiqueta=str(pieza)
print(xc+" "+yc+" "+CDGZ+" "+orientacion+" "+Etiqueta)
data=CDGX+CDGY+CDGZ+orientacion+Etiqueta
sock.send(data.encode())
```

```
        HaDejadoObjeto=sock.recv(1024);
        print (HaDejadoObjeto)
        ret, depth_frame, frame = dc.get_frame()
        cv2.imshow("Estado del Trabajo",frame)
        cv2.waitKey(5)
        Confirmacion='okay'
        sock.send(Confirmacion.encode())

def CondicionesIniciales(imagenInicial):
    # Crear el entorno de colores HSV
    hsv = cv2.cvtColor(imagenInicial, cv2.COLOR_BGR2HSV)

    #Primero se deben establecer los límites inferiores y superiores del thresholding

    #Máscara del rojo
    MascaraRojo=cv2.inRange(hsv, (0, 70, 70), (20, 255,255))

    #Máscara del azul
    MascaraAzul=cv2.inRange(hsv, (95, 70, 70), (130, 255,255))

    #Máscara del Verde
    MascaraVerde= cv2.inRange(hsv, (35, 50, 50), (95, 255,255))

    #Máscara del Gris
    MascaraGris= cv2.inRange(imagenInicial, (0, 0, 0), (45, 45,45))
```

```
    #Obtengo la imágen resultante de aplicar el filtro

frameRojoColor=cv2.bitwise_and(imagenInicial,imagenInicial,mask=MascaraRojo)

frameVerdeColor=cv2.bitwise_and(imagenInicial,imagenInicial,mask=MascaraVerde)

frameAzulColor=cv2.bitwise_and(imagenInicial,imagenInicial,mask=MascaraAzul)

frameGrisColor=cv2.bitwise_and(imagenInicial,imagenInicial,mask=MascaraGris)


    #Dicha imagen la paso a escala de grises
    Rojo_BGR2GRAY= cv2.cvtColor(frameRojoColor, cv2.COLOR_BGR2GRAY)
    Azul_BGR2GRAY= cv2.cvtColor(frameAzulColor, cv2.COLOR_BGR2GRAY)
    Verde_BGR2GRAY= cv2.cvtColor(frameVerdeColor, cv2.COLOR_BGR2GRAY)
    Gris_BGR2GRAY= cv2.cvtColor(frameGrisColor, cv2.COLOR_BGR2GRAY)


    #Y por último a binario

ret,frameRojo=cv2.threshold(Rojo_BGR2GRAY,0,255,cv2.THRESH_BINARY_INV)

ret,frameAzul=cv2.threshold(Azul_BGR2GRAY,0,255,cv2.THRESH_BINARY_INV)

ret,frameVerde=cv2.threshold(Verde_BGR2GRAY,0,255,cv2.THRESH_BINARY_INV)
```

```
ret,frameGris=cv2.threshold(Gris_BGR2GRAY,0,255,cv2.THRESH_BINARY_I
NV)


    #Encontrar la Pegatina Roja
    CDGRojo=SacarCDG(frameRojo)



    #Encontrar la Pegatina Azul
    CDGAzul=SacarCDG(frameAzul)
    cv2.imshow("frameAzul",frameAzul)
    cv2.waitKey(5)

    #Encontrar la Pegatina Verde
    CDGVerde=SacarCDG(frameVerde)

    #Obtener altura de los cuadrados de colores y el circulo
    distanciaCuadrado=303
    distanciaCirculo = 304
    distanciaCuadrado=str(distanciaCuadrado+100)
    distanciaCirculo = str(distanciaCirculo+100)

    #Encontrar el circulito de quesitos
    CDGGris=SacarCDG(frameGris)


CDGTotal=CDGRojo+CDGVerde+CDGAzul+CDGGris+distanciaCuadrado+dist
anciaCirculo
```

```
    sock.send(CDGTotal.encode())

    time.sleep(2)


IP = "192.168.1.100";
TCP_PORT = 30000          # puerto de la conexión
BUFFER_SIZE = 4096        # tamaño del búfer de datos
dc = DepthCamera()
# Crea un objeto de socket TCP/IP y establece la conexión

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((IP, TCP_PORT))
respuesta=sock.recv(1024)
print (respuesta)

EmpiezaProceso=sock.recv(1024)
ret, depth_frame, frame = dc.get_frame()
CondicionesIniciales(frame)
EsperoARobot=sock.recv(1024)

while(True):
    Demandadeobjetos=sock.recv(1024)
    ret, depth_frame, frame = dc.get_frame()
    #Aplicar Threshold
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    sobel=filters.sobel(gray)
```

```
    thresh=cv2.inRange(sobel, 0,110)
    cleared = clear_border(thresh)
    opening = cv2.morphologyEx(cleared, cv2.MORPH_OPEN, kernel, iterations=1)
    dilate=cv2.morphologyEx(opening, cv2.MORPH_DILATE, kernel, iterations=3)
    closing = cv2.morphologyEx(dilate, cv2.MORPH_CLOSE, kernel, iterations=20)
    erode=cv2.morphologyEx(closing, cv2.MORPH_ERODE, kernel, iterations=3)


    cv2.imshow("erode”, erode)
    cv2.waitKey(5)


    cv2.imshow("Estado del Trabajo", frame)
    cv2.waitKey(5)
    print (Demandadeobjetos)
    Analisis (erode,depth_frame)
```

1.2. Medir valores de RGB y escala de grises

```
import cv2
import pyrealsense2
from realsense_depth import *

point = (0,0)

def show_RGB(event, x, y, args, params):
    global point
    point = (x, y)

# Inicializo la camara
dc = DepthCamera()

# Creo el evento del ratón en el que recojo los puntos
cv2.namedWindow("Color frame")
cv2.setMouseCallback("Color frame", show_RGB)

while True:
    ret, depth_frame, color_frame = dc.get_frame()
    gray_frame = cv2.cvtColor(color_frame, cv2.COLOR_BGR2GRAY)
    # Mostrar RGB o Escala de grises para un punto
    cv2.circle(color_frame, point, 4, (0, 0, 255))
    b,g,r = (color_frame[point[1], point[0]])
    graylevel=(gray_frame[point[1], point[0]])
    data=str(r)+' '+str(g)+' '+str(b)
```

```
    data1=str(graylevel)

    cv2.putText(color_frame, data, (point[0], point[1] - 20), cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 0), 2)

    cv2.imshow("Color frame", color_frame)
    cv2.imshow("Gray frame", gray_frame)
    print(data)
    print(graylevel)
    key = cv2.waitKey(1)
    if cv2.waitKey(1) & 0xFF == ord('p'): #presionar p

      break
```

1.3. Código para uso de la cámara

```
import pyrealsense2 as rs
import numpy as np


class DepthCamera:
    def __init__(self):
        # Configurar los streams de color y profundidad
        self.pipeline = rs.pipeline()
        config = rs.config()

        # Obtener una línea con el dispositivo para establecer una resolución compatible

        pipeline_wrapper = rs.pipeline_wrapper(self.pipeline)
        pipeline_profile = config.resolve(pipeline_wrapper)
        device = pipeline_profile.get_device()
        device_product_line = str(device.get_info(rs.camera_info.product_line))

        config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
        config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)

        # Empezar el streaming
        self.pipeline.start(config)

    def get_frame(self):
        frames = self.pipeline.wait_for_frames()
```

```
    depth_frame = frames.get_depth_frame()
    color_frame = frames.get_color_frame()

    depth_image = np.asanyarray(depth_frame.get_data())
    color_image = np.asanyarray(color_frame.get_data())
    if not depth_frame or not color_frame:
        return False, None, None
    return True, depth_image, color_image

def release(self):
    self.pipeline.stop()
```

2. RobotStudio

2.1. Código principal

```
MODULE Proyecto
    VAR socketdev serverSocket;
    VAR socketdev clientSocket;

    VAR bool okCDGRojoX;
    VAR bool okCDGRojoY;
    VAR bool okCDGVerdeX;
    VAR bool okCDGVerdeY;
    VAR bool okCDGAzulX;
    VAR bool okCDGAzulY;
    VAR bool okCDGQuesitosX;
    VAR bool okCDGQuesitosY;
    VAR bool okAltColores;
    VAR bool okAltCirculo;

    VAR num CDGRojoX;
    VAR num CDGRojoY;
    VAR num CDGVerdeX;
    VAR num CDGVerdeY;
    VAR num CDGAzulX;
    VAR num CDGAzulY;
    VAR num CDGQuesitosX;
    VAR num CDGQuesitosY;
    VAR num AltColores;
    VAR num AltCirculo;
    VAR num DiaRefresco:=66; !Diametro de la cocacola
    VAR num AltRefresco:=110;
    VAR num DiaMaiz:=66;     !Diametro de lata de maíz
    VAR num AltMaiz:=71;     !Altura lata maíz
    VAR num AltAnchoas:=14;   !ALtura latas de anchoas
    VAR num RadioMenorAnchoas:=56.43;
    VAR num RadioCirculoQuesitos:=54;
    VAR num AltQuesitos:=14;
    VAR num AnguloAGirar;
    VAR num AngulosQuesitos{6}:=[0,60,120,180,240,300];

    VAR string simbolo;
    VAR string StringCDGRojoX;
    VAR string StringCDGRojoY;
```

```
42.   VAR string StringCDGVerdeX;
43.   VAR string StringCDGVerdeY;
44.   VAR string StringCDGAzulX;
45.   VAR string StringCDGAzulY;
46.   VAR string StringAltColores;
47.   VAR string StringAltCirculo;
48.   VAR string StringCDGQuesitosX;
49.   VAR string StringCDGQuesitosY;
50.
51.   CONST robtarget PosAnálisis:=[[278.3,-522.2,677.8],[0.01494,-0.53509,-
      0.84464,0.00668],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
52.   CONST robtarget
      PosCondicionesIniciales:=[[435.1,31.5,519.5],[0.01501,-0.53485,-
      0.84479,0.00654],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
53.   VAR robtarget
      PuntoObjeto:=[[100,200,300],[1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9
      ]];
54.   VAR robtarget
      PuntoPaletizar:=[[100,200,300],[1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9
      E9]];
55.   VAR robtarget PuntoDeGiro;
56.
57.   VAR string data;
58.   VAR string data1;
59.   VAR string data2;
60.   VAR string data3;
61.   VAR num numPiezas:=0;
62.   VAR bool oknumPiezas;
63.   VAR num PiezasMovidas;
64.
65.   VAR num LatasRefresco:=0;
66.   VAR num LatasMaiz:=0;
67.   VAR num LatasAnchoas:=0;
68.   VAR num NumeroQuesitos:=0;
69.
70.   VAR num MatrizCamara{4,4};
71.   VAR num MatrizRobot{4,4};
72.   VAR num MatrizTransformacion{4,4};
73.
74.   VAR bool okCDGX;
75.   VAR bool okCDGY;
76.   VAR bool okCDGZ;
77.   VAR bool okOri;
78.   VAR bool okLabel;
79.
```

```
80.    VAR string stringCDGX;
81.    VAR string stringCDGY;
82.    VAR string stringCDGZ;
83.    VAR string stringOri;
84.    VAR string stringLabel;
85.
86.    VAR num CDGX;
87.    VAR num CDGY;
88.    VAR num CDGZ;
89.    VAR num Ori;
90.    VAR num Label;
91.
92.    VAR robtarget Refresco{6};
93.    VAR robtarget Maiz{6};
94.    VAR robtarget Anchoas{3};
95.    VAR robtarget Quesitos{8};
96.
97.
98. PROC CondicionesIniciales()
99.            SocketReceive clientSocket \str:=data;
100.
101.           simbolo:=StrPart(data,1,1);
102.           StringCDGRojoX:= StrPart(data,2,4);  !Le indico la variable que
    quiero separar y le indico donde empieza y cuanto ocupa en strings
103.           okCDGRojoX:=StrToVal(StringCDGRojoX,CDGRojoX);
    !Cambio de string a num
104.       IF simbolo="*" THEN
105.               CDGRojoX:=CDGRojoX-100;
106.       ENDIF
107.
108.       IF simbolo="?" THEN
109.               CDGRojoX:=CDGRojoX-300;
110.       ENDIF
111.
112.       simbolo:=StrPart(data,6,1);
113.       StringCDGRojoY:= StrPart(data,7,4);
114.       okCDGRojoY:=StrToVal(StringCDGRojoY,CDGRojoY); !Cambio
    de string a num
115.       IF simbolo="*" THEN
116.               CDGRojoY:=CDGRojoY-100;
117.       ENDIF
118.
119.       IF simbolo="?" THEN
120.               CDGRojoY:=CDGRojoY-300;
121.       ENDIF
```

```
122.
123.        simbolo:=StrPart(data,11,1);
124.        StringCDGVerdeX:= StrPart(data,12,4);  !Le indico la variable que
    quiero separar y le indico donde empieza y cuanto ocupa en strings
125.
        okCDGVerdeX:=StrToVal(StringCDGVerdeX,CDGVerdeX); !Cambio
    de string a num
126.        IF simbolo="*" THEN
127.                CDGVerdeX:=CDGVerdeX-100;
128.        ENDIF
129.
130.        IF simbolo="?" THEN
131.                CDGVerdeX:=CDGVerdeX-300;
132.        ENDIF
133.
134.        simbolo:=StrPart(data,16,1);
135.        StringCDGVerdeY:= StrPart(data,17,4);
136.
        okCDGVerdeY:=StrToVal(StringCDGVerdeY,CDGVerdeY); !Cambio
    de string a num
137.        IF simbolo="*" THEN
138.                CDGVerdeY:=CDGVerdeY-100;
139.        ENDIF
140.
141.        IF simbolo="?" THEN
142.                CDGVerdeY:=CDGVerdeY-300;
143.        ENDIF
144.
145.        simbolo:=StrPart(data,21,1);
146.        StringCDGAzulX:= StrPart(data,22,4);  !Le indico la variable que
    quiero separar y le indico donde empieza y cuanto ocupa en strings
147.        okCDGAzulX:=StrToVal(StringCDGAzulX,CDGAzulX); !Cambio
    de string a num
148.        IF simbolo="*" THEN
149.                CDGAzulX:=CDGAzulX-100;
150.        ENDIF
151.
152.        IF simbolo="?" THEN
153.                CDGAzulX:=CDGAzulX-300;
154.        ENDIF
155.
156.        simbolo:=StrPart(data,26,1);
157.        StringCDGAzulY:= StrPart(data,27,4);
158.          okCDGAzulY:=StrToVal(StringCDGAzulY,CDGAzulY);
    !Cambio de string a num
```

```
159.        IF simbolo="*" THEN
160.                CDGAzulY:=CDGAzulY-100;
161.        ENDIF
162.
163.        IF simbolo="?" THEN
164.                CDGAzulY:=CDGAzulY-300;
165.        ENDIF
166.
167.        simbolo:=StrPart(data,31,1);
168.        StringCDGQuesitosX:= StrPart(data,32,4);
169.
      okCDGQuesitosX:=StrToVal(StringCDGQuesitosX,CDGQuesitosX);
   !Cambio de string a num
170.       IF simbolo="*" THEN
171.                CDGQuesitosX:=CDGQuesitosX-100;
172.        ENDIF
173.
174.        IF simbolo="?" THEN
175.                CDGQuesitosX:=CDGQuesitosX-300;
176.        ENDIF
177.
178.        simbolo:=StrPart(data,36,1);
179.        StringCDGQuesitosY:= StrPart(data,37,4);
180.
      okCDGQuesitosY:=StrToVal(StringCDGQuesitosY,CDGQuesitosY);
   !Cambio de string a num
181.        IF simbolo="*" THEN
182.                CDGQuesitosY:=CDGQuesitosY-100;
183.        ENDIF
184.
185.        IF simbolo="?" THEN
186.                CDGQuesitosY:=CDGQuesitosY-300;
187.        ENDIF
188.
189.        StringAltColores:= StrPart(data,41,3);
190.        okAltColores:=StrToVal(StringAltColores,AltColores); !Cambio de
   string a num
191.       AltColores:=AltColores-100;
192.
193.        StringAltCirculo:= StrPart(data,44,3);
194.        okAltCirculo:=StrToVal(StringAltCirculo,AltCirculo); !Cambio de
   string a num
AltCirculo:=AltCirculo-100;
195.
```

```
196. Refresco{1}:=[[CDGRojoX+DiaRefresco/2,CDGRojoY+DiaRefresco,(AltColores+AltRefresco)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
197. Refresco{2}:=[[CDGRojoX+DiaRefresco/2,CDGRojoY,(AltColores+AltRefresco)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
198. Refresco{3}:=[[CDGRojoX+DiaRefresco/2,CDGRojoY-DiaRefresco,(AltColores+AltRefresco)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
199. Refresco{4}:=[[CDGRojoX-DiaRefresco/2,CDGRojoY+DiaRefresco,(AltColores+AltRefresco)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
200. Refresco{5}:=[[CDGRojoX-DiaRefresco/2,CDGRojoY,(AltColores+AltRefresco)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
201. Refresco{6}:=[[CDGRojoX-DiaRefresco/2,CDGRojoY-DiaRefresco,(AltColores+AltRefresco)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
202.
203. Maiz{1}:=[[CDGAzulX+DiaMaiz,CDGAzulY,(AltColores+AltMaiz)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
204. Maiz{2}:=[[CDGAzulX,CDGAzulY,(AltColores+AltMaiz)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
205. Maiz{3}:=[[CDGAzulX-DiaMaiz,CDGAzulY,(AltColores+AltMaiz)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
206. Maiz{4}:=[[CDGAzulX+DiaMaiz,CDGAzulY,(AltColores+2*AltMaiz)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
207. Maiz{5}:=[[CDGAzulX,CDGAzulY,(AltColores+2*AltMaiz)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
208. Maiz{6}:=[[CDGAzulX-DiaMaiz,CDGAzulY,(AltColores+2*AltMaiz)],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
209.
210. Anchoas{1}:=[[CDGVerdeX,CDGVerdeY-RadioMenorAnchoas,AltColores+AltAnchoas],[0.01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
```

```
211.
    Anchoas{2}:=[[CDGVerdeX,CDGVerdeY,AltColores+AltAnchoas],[0.0146
    8,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
212.
    Anchoas{3}:=[[CDGVerdeX,CDGVerdeY+RadioMenorAnchoas,AltColore
    s+AltAnchoas],[0.01468,-0.53568,-
    0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
213.
214.
    Quesitos{1}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(0),CDGQ
    uesitosY+RadioCirculoQuesitos*0.65*Cos(0),(AltCirculo+AltQuesitos)],[0.
    01468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
215.
    Quesitos{2}:=[[CDGQuesitosX+RadioCirculoQuesitos*Sin(60),CDGQuesit
    osY+RadioCirculoQuesitos*0.65*Cos(60),(AltCirculo+AltQuesitos)],[0.014
    68,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
216.
    Quesitos{3}:=[[CDGQuesitosX+RadioCirculoQuesitos*Sin(120),CDGQues
    itosY+RadioCirculoQuesitos*0.65*Cos(120),(AltCirculo+AltQuesitos)],[0.0
    1468,-0.53568,-0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
217.
    Quesitos{4}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(180),CDG
    QuesitosY+RadioCirculoQuesitos*0.65*Cos(180),(AltCirculo+AltQuesitos)
    ],[0.01468,-0.53568,-
    0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
218.
    Quesitos{5}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(240),CDG
    QuesitosY+RadioCirculoQuesitos*0.65*Cos(240),(AltCirculo+AltQuesitos)
    ],[0.01468,-0.53568,-
    0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
219.
    Quesitos{6}:=[[CDGQuesitosX+RadioCirculoQuesitos*0.65*Sin(300),CDG
    QuesitosY+RadioCirculoQuesitos*0.65*Cos(300),(AltCirculo+AltQuesitos)
    ],[0.01468,-0.53568,-
    0.84430,0.00653],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]]
220.                    !ENDWHILE
221.        ENDPROC
222.
223.        PROC MoverObjetos()
224.          !Pregunto cuantos objetos hay
225.          WaitTime 3;
226.          SocketSend clientSocket \str:="Cuantos Objetos hay";
227.                    SocketReceive clientSocket \str:=data1;
228.          oknumPiezas:=StrToVal(data1,numPiezas);
229.
```

```
230.        FOR PiezasMovidas FROM 1 TO  numPiezas STEP 1 DO
231.            SocketSend clientSocket \str:="Dime Data";
232.            SocketReceive clientSocket \str:=data2;
233.            StringCDGX:= StrPart(data2,1,4);  !Le indico la variable que
    quiero separar y le indico donde empieza y cuanto ocupa en strings
234.            okCDGX:=StrToVal(StringCDGX,CDGX); !Cambio de string a
    num
235.            CDGX:=CDGX-200;
236.
237.            StringCDGY:= StrPart(data2,5,4);  !Le indico la variable que
    quiero separar y le indico donde empieza y cuanto ocupa en strings
238.            okCDGY:=StrToVal(StringCDGY,CDGY); !Cambio de string a
    num
239.            CDGY:=CDGY-200;
240.
241.            StringCDGZ:= StrPart(data2,9,3);  !Le indico la variable que
    quiero separar y le indico donde empieza y cuanto ocupa en strings
242.            okCDGZ:=StrToVal(StringCDGZ,CDGZ); !Cambio de string a
    num
243.            CDGZ:=CDGZ-100;
244.
245.            StringOri:= StrPart(data2,12,3);  !Le indico la variable que quiero
    separar y le indico donde empieza y cuanto ocupa en strings
246.            okOri:=StrToVal(StringOri,Ori); !Cambio de string a num
247.            Ori:=Ori-100;
248.            Ori:=Ori;
249.
250.            StringLabel:= StrPart(data2,15,1);  !Le indico la variable que
    quiero separar y le indico donde empieza y cuanto ocupa en strings
251.            okLabel:=StrToVal(StringLabel,Label); !Cambio de string a num
252.
253.            PuntoObjeto:=[[CDGX,CDGY,CDGZ],[0.01476,-0.53577,-
    0.84421,0.00661],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
254.
255.            IF Label=1 THEN
256.            LatasRefresco:=LatasRefresco+1;
257.            PuntoPaletizar:=Refresco{LatasRefresco};
258.            AnguloAGirar:=0;
259.            ENDIF
260.
261.            IF Label=2 THEN
262.            LatasAnchoas:=LatasAnchoas+1;
263.            PuntoPaletizar:=Anchoas{LatasAnchoas};
264.            AnguloAGirar:=(Ori-90);
265.            ENDIF
```

```
266.
267.            IF Label=3 THEN
268.               NumeroQuesitos:=NumeroQuesitos+1;
269.               PuntoPaletizar:=Quesitos{NumeroQuesitos};
270.               AnguloAGirar:= Ori- AngulosQuesitos{NumeroQuesitos};
271.               IF AnguloAGirar>180 THEN
272.                  AnguloAGirar:=AnguloAGirar-360;
273.               ENDIF
274.               IF AnguloAGirar<180 THEN
275.                  AnguloAGirar:=AnguloAGirar+360;
276.               ENDIF
277.            ENDIF
278.
279.            IF Label=4 THEN
280.            LatasMaiz:=LatasMaiz+1;
281.            PuntoPaletizar:=Maiz{LatasMaiz};
282.            AnguloAGirar:=0;
283.            ENDIF
284.
285.            MoveJ  Offs(PuntoObjeto,0,0,100),v150,z5,tool0;
286.            MoveL PuntoObjeto,v10,z0,tool0; !Llegamos a donde está el
    objeto
287.            RESET GRIPPER_CLOSE;
288.            WaitTime 3;
289.            MoveL  Offs(PuntoObjeto,0,0,100),v20,z5,tool0;
290.            MoveL  Offs(PuntoPaletizar, 0, 0, 200),v50,z0,tool0;
291.            PuntoDeGiro:= CRobT(\Tool:=tool0 \Wobj:=wobj0);
292.            MoveL RelTool (PuntoDeGiro, 0, 0, 0 \Rz:= AnguloAGirar), v10,
    z1, tool0;
293.            MoveL RelTool (PuntoPaletizar, 0, 0, 0 \Rz:= AnguloAGirar),
    v10, z1, tool0;
294.            SET GRIPPER_CLOSE;
295.            WaitTime 3;
296.            MoveL RelTool (PuntoDeGiro, 0, 0, 0 \Rz:= AnguloAGirar), v50,
    z1, tool0;
297.            SocketSend clientSocket \str:="Ya he dejado el objeto";
298.                     SocketReceive clientSocket \str:=data3;
299.            MoveJ  PosAnálisis,v150,z0,tool0;
300.
301.         ENDFOR
302.      ENDPROC
303.
304.      PROC main()
305.         Set GRIPPER_CLOSE;
306.         Reset CONVEYOR_FWD;
```

```
307.        SocketCreate serverSocket;
308.        SocketBind serverSocket, "192.168.1.100", 30000;
309.        SocketListen serverSocket;
310.        TPWrite "Esperando conexión";
311.        SocketAccept serverSocket,clientSocket,\Time:=60000;  !Espero a
    que python se conecte
312.        SocketSend clientSocket \str:="Estoy listo"; !Le envío a python este
    mensaje
313.        TPWrite "Conexión Lograda";
314.        MoveJ PosCondicionesIniciales,v150,fine,tool0;
315.        SocketSend clientSocket \str:="Estoy en posicion"; !Le envío a
    python este mensaje
316.        WaitTime 3;
317.        CondicionesIniciales;
318.
319.        moveJ PosAnálisis,v40,z5,tool0;
320.        SocketSend clientSocket \str:="Estoy en posicion de analisis"; !Le
    envío a python este mensaje
321.        ConfL \Off;
322.        WHILE LatasAnchoas<3 OR NumeroQuesitos<6 OR LatasMaiz<6
    DO
323.        Set CONVEYOR_FWD;
324.        WaitDI CONVEYOR_OBJ_SEN,1;
325.        Reset CONVEYOR_FWD;
326.        MoverObjetos;
327.        ENDWHILE
328.
329.        SocketClose clientSocket;
330.      ENDPROC
331.  ENDMODULE
```

3. Matlab

3.1. Main Cinta

```
% Calibración en el plano de la cinta
% Las dos primeras columnas corresponden a las coordenadas X e Y de los puntos medidos con el robot.
% Las columnas 4 y 5 corresponden a las coordenadas X e Y de los puntos medidos desde la cámara.

X=[289.8 -533.7 1 391.85 400.26 1;
    290.1 -437.7 1 237.54 401.26 1;
    468.7 -441.9 1 238.33 112.53 1;
    468.3 -538.2 1 393.27 112.33 1;
    376.6 -489.2 1 315.3 256.99 1];

x=X(:,1:3)';
u=X(:,4:6)';

figure
hold on
plot(x(1,:), x(2,:), 'rx')
plot(u(1,:), u(2,:), 'bx')
for i=1:5
    texto = sprintf('%d', i);
    text(x(1,i), x(2,i), texto);
    text(u(1,i), u(2,i), texto);
end

H=homografia(x,u)


xe=inv(H)*u;

xee=[xe(1,:)./xe(3,:);xe(2,:)./xe(3,:);xe(3,:)./xe(3,:)];
error=xee-x

% Los valores del vector error deben ser cercanos a cero.
% La matriz a utilizar para hacer la conversión de las coordenadas de la cámara al robot es la inversa de H.
```

3.2. Main Condiciones Iniciales

```
% Calibración en el plano de la cinta
% Las dos primeras columnas corresponden a las coordenadas X e Y de los puntos medidos con el robot.
% Las columnas 4 y 5 corresponden a las coordenadas X e Y de los puntos medidos desde la cámara.

X=[469.5 -69.6 1 526.86 353.22 1;
   472.3 192.8 1 110.97 354.0 1;
   613.7 186.3 1 109.47 129.02 1;
   606.2 -73.1 1 529.86 129.5 1;
   539.9 59.2 1 319.92 241.86 1];

x=X(:,1:3)';
u=X(:,4:6)';

figure
hold on
plot(x(1,:), x(2,:), 'rx')
plot(u(1,:), u(2,:), 'bx')
for i=1:5
    texto = sprintf('%d', i);
    text(x(1,i), x(2,i), texto);
    text(u(1,i), u(2,i), texto);
end

H=homografia(x,u)


xe=inv(H)*u;

xee=[xe(1,:)./xe(3,:);xe(2,:)./xe(3,:);xe(3,:)./xe(3,:)];
error=xee-x

% Los valores del vector error deben ser cercanos a cero.
% La matriz a utilizar para hacer la conversión de las coordenadas de la cámara al robot es la inversa de H.


```

3.3. Homografia

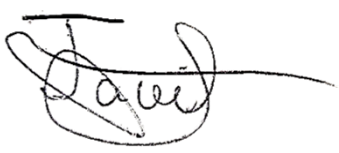
```
4.  function [H, A] = homografia(X,U)
5.  % [Hest] = homomografia(X,U)
6.  % Calcula la homografia entre dos imagenes de las cuales se tienen
    puntos que coinciden X y U
7.  % X - matriz con los puntos en la imagen 1 en coordenadas homogeneas
    organizados en columnas
8.  % U - matriz de los puntos en la imagen 2 en coordenadas homogeneas
    organizados en columnas
9.  % H - homografía estimada estimada
10.
11.
12. % Estimacion de la homografia
13. nPuntos = size(X,2);
14.
15. A=[];
16. for (i=1:1:nPuntos)
17.     A=[A; X(:,i)' zeros(1,3) -U(1,i)*X(:,i)';
18.         zeros(1,3) X(:,i)' -U(2,i)*X(:,i)'];
19. end
20.
21. % Homografia estimada
22. b = -A(:,9);
23. a = A(:,1:8);
24. C = inv(a'*a);
25. H = C*a'*b;
26.
27. H=[H;1];
28.
29. H=reshape(H,3,3)';
```

3.4. Calibración por mínimos cuadrados en cinta

```
%% Calibración cámara:

xc=[391.85 237.54 238.33 393.27 315.3]';
yc=[400.26 401.26 112.53 112.33 256.99]';
uc=[1 1 1 1 1]';

xr=[289.8 290.1 468.7 468.3 376.6]';
yr=[-533.7 -437.7 -441.9 -538.2 -489.2]';

% xc=[269 270 242 209 211 138 82 24 25]';
% yc=[ 324 265 324 358 299 298 398 396 369]';
% uc=[1 1 1 1 1 1 1 1 1]';
%
% xr=[573.4 747.7 571.5 471.6 643.7 646.5 348.1 353 433.1]';
% yr=[-213.3 -215.8 -132 -38.4 -39 185.5 347.8 523.9 522.7]';

C=[xc yc uc];

M=[xr yr uc];

MT3=inv(C'*C)*C'*M
MT4=(C'*C)\(C'*M)

MT=[]
pos_robot1=[269 270 1]*MT3
pos_robot2= MT3'*[269; 270; 1]

% pos_robot1 y pos_robot2 deben ser igual

for i=1:15

    error=[xr(i) yr(i) 1]-[xc(i) yc(i) 1]*MT3;
    errorx(i)=error(1);
    errory(i)=error(2);

end
```

VALENCIA, JULIO 2023

Javier Gamir Artesero



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

ANEJO Nº2: CÁLCULOS

Calibración Cámara-Robot

Para la calibración entre el robot y la cámara, tal y como se ha mencionado previamente ha sido utilizado un algoritmo en el programa Matlab de homografía, lo que permite obtener una relación mediante una matriz de transformación que relaciona las coordenadas que proporciona la cámara con coordenadas homogéneas.

Para la aplicación de este algoritmo ha sido creada una imagen de calibración casera, con la cual se obtendrán las coordenadas en el espacio del robot posicionándolo de forma manual en el centro de estos y las coordenadas que proporciona la cámara con el lenguaje Python.

Debido a que este método no asegura una calibración fielmente precisa fuera de la zona establecida en la calibración, se tomarán dos medidas:

-La primera medida es que se usará una imagen de calibración para un folio din A4 para la zona de la cinta transportadora donde se recogen los objetos y la misma imagen, pero acondicionada a un folio din A3 en la zona de elección de empaquetado de productos, para de esta forma asegurar una mayor zona de trabajo posible.

-La segunda medida es que los puntos desde los cuales se tomen las coordenadas con la cámara serán los mismos puntos que se usarán para tomar los datos de los objetos y los datos de las marcas de empaquetado, esto ha sido decidido para garantizar la mayor fiabilidad de la calibración.

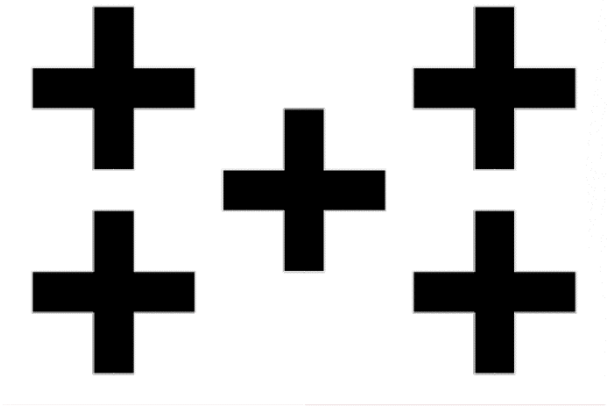


Figura Calibración.1. Patrón de calibración

A continuación, se procede a mostrar y explicar paso por paso como realizar la calibración explicada:

1. Se coloca el patrón de calibración en la cinta transportadora

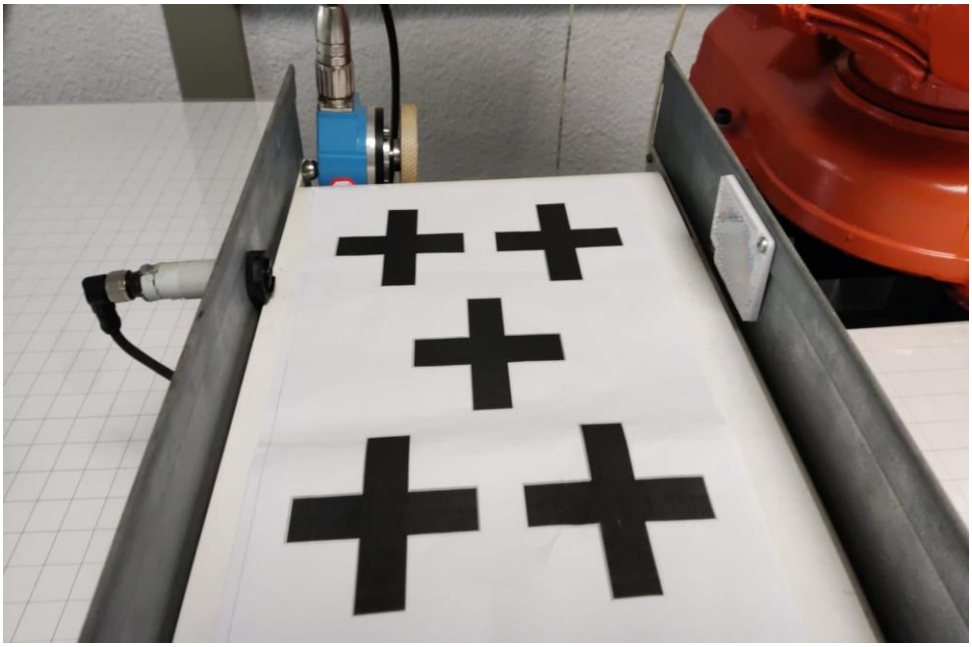


Figura Calibración.2. Patrón de calibración

2. Se posiciona la ventosa del robot en el centro de las cruces del patrón y se toma una lectura de la posición del robot.

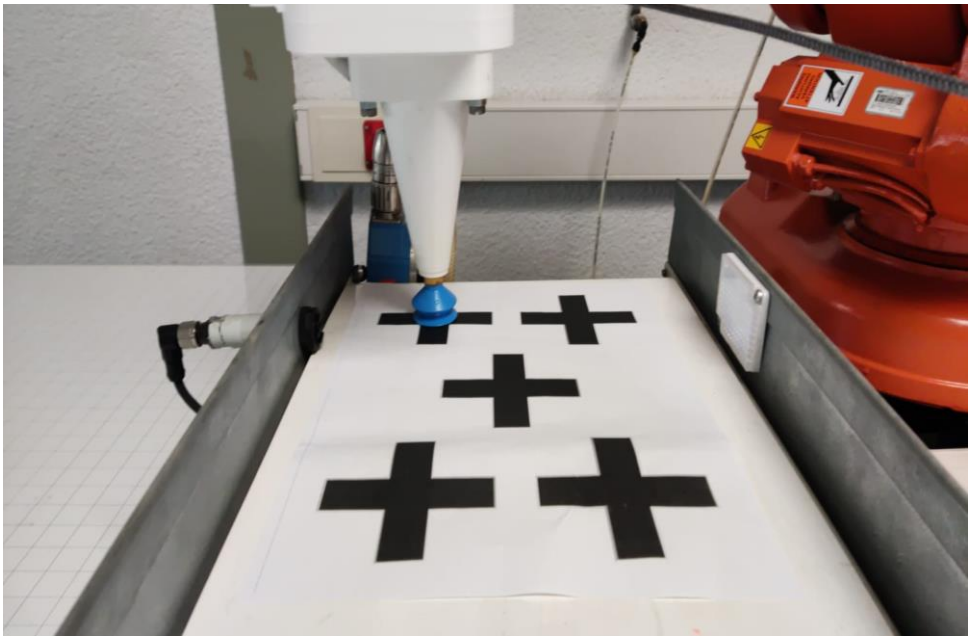


Figura Calibración.3. Posicionamiento de la ventosa en centro de cruces

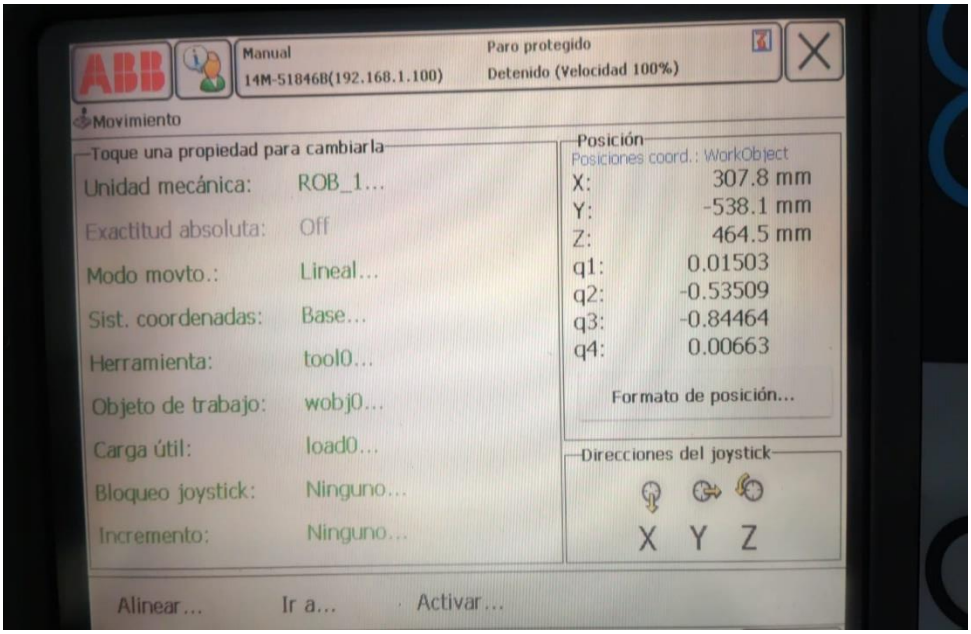


Figura Calibración.4. Lectura de la posición del robot

3. Se repite el proceso para los cinco centros, a mayor cantidad de puntos tomados mayor calidad presentará la calibración, pero debido a que se tendrá un máximo de tres objetos por el tamaño de la cinta, con cinco puntos será suficiente.

4. Una vez se han tomado las lecturas con el robot, se obtienen las coordenadas que lee la cámara mediante una toma de captura del patrón y el lenguaje Python, haciendo uso del conjunto de librerías 'Scikit-Image', en este caso concreto la librería 'Sckit-Image.measure'

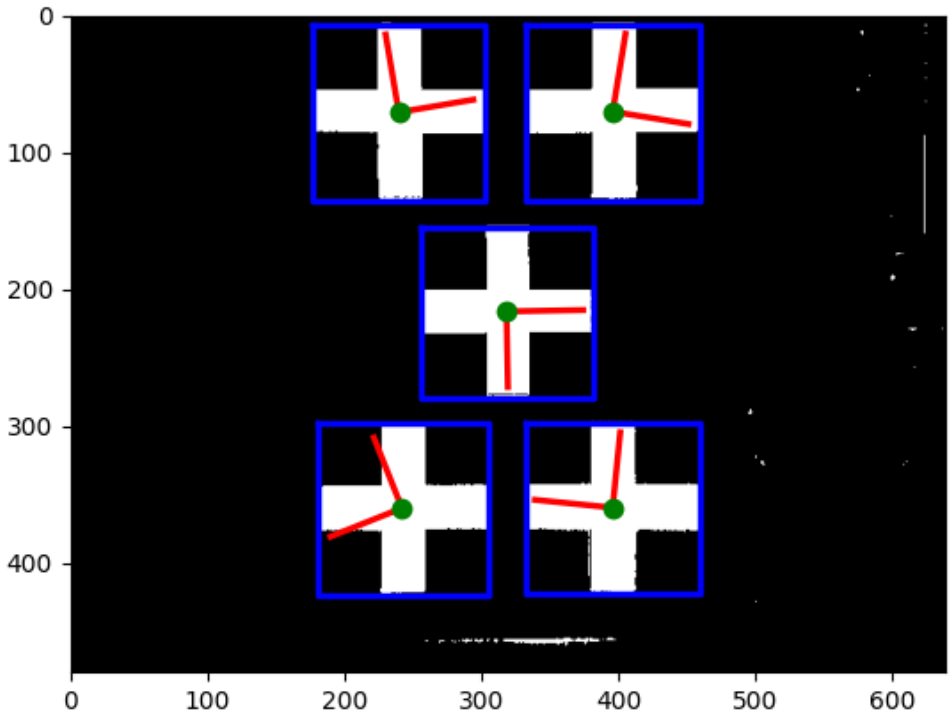


Figura Calibración.5. Obtención de puntos leídos por la cámara

5. Una vez obtenidos los 5 puntos en ambas partes, se introduce esta información en el archivo main.m que se adjuntará en el anexo de códigos, obteniendo de esta forma la matriz de transformación para las coordenadas en los ejes X e Y de la cámara a homogéneas:

$$H = \begin{bmatrix} -0.0701 & -1.5193 & -418.4623 \\ -1.5287 & 0.0526 & 846.6075 \\ -0.0001 & 0.0000 & 1.0000 \end{bmatrix}$$

Figura Calibración.6. Matriz de transformación de coordenadas de Robot a homogéneas

6. Una vez obtenida la matriz, es necesario obtener la inversa, para obtener la transformación de coordenadas de cámara a homogéneas.

$$H^{-1} = \begin{bmatrix} -0.0192 & -0.6999 & 584.4861 \\ -0.6590 & 0.0560 & -323.1598 \\ 0.0000 & -0.0001 & 1.0755 \end{bmatrix}$$

Figura Calibracón.7. Matriz de transformación de coordenadas de Cámara a homogéneas

7. Una vez obtenida la nueva matriz, es posible realizar un producto de la matriz con el vector de coordenadas de la cámara, obteniendo las coordenadas homogéneas.

$$\begin{bmatrix} x_h \\ y_h \\ H \end{bmatrix} = \begin{bmatrix} -0.0192 & -0.6999 & 584.4861 \\ -0.6590 & 0.0560 & -323.1598 \\ 0.0000 & -0.0001 & 1.0755 \end{bmatrix} \cdot \begin{bmatrix} x_{cámara} \\ y_{cámara} \\ 1 \end{bmatrix}$$

Figura Calibración.8. Operación para obtención de coordenadas homogéneas

8. Debido a que el robot emplea coordenadas cartesianas será necesario convertir las coordenadas homogéneas a cartesianas, mediante las siguientes operaciones:

$$x_{Robot} = \frac{x_h}{H} \qquad y_{Robot} = \frac{y_h}{H}$$

Figura Calibración.9. Obtención de coordenadas cartesianas

Estos valores son los que el ordenador deberá comunicar mediante Python al robot.

Una calibración alternativa al *script* de homografía mencionado al principio es por mínimos cuadrados, este necesita tomar las mismas medidas de la rejilla de calibración con el robot y con la cámara, una vez obtenidos los puntos se obtiene una matriz de calibración que relacione un punto de la imagen con un el de la posición X e Y del robot:

$$P_M = [m_x, m_y, 1] \qquad \hat{P}_M = P_C \cdot \theta = [C_x, C_y, 1] \cdot \begin{bmatrix} \hat{\theta}_{11} & \hat{\theta}_{12} & \hat{\theta}_{13} \\ \hat{\theta}_{21} & \hat{\theta}_{22} & \hat{\theta}_{23} \\ \hat{\theta}_{31} & \hat{\theta}_{32} & \hat{\theta}_{33} \end{bmatrix}$$

Figura Calibración.10. Calibrado de cámara mediante mínimos cuadrados

El error de P_M y de $\hat{P}_M$ ha de ser el mínimo posible.

El error de calibración será:

$$e = [e_x, e_y, 1] = P_M - \hat{P}_M = [m_x, m_y, 1] - [C_x, C_y, 1] \cdot \begin{bmatrix} \hat{\theta}_{11} & \hat{\theta}_{12} & \hat{\theta}_{13} \\ \hat{\theta}_{21} & \hat{\theta}_{22} & \hat{\theta}_{23} \\ \hat{\theta}_{31} & \hat{\theta}_{32} & \hat{\theta}_{33} \end{bmatrix}$$

Figura Calibración.11. Error de calibración por mínimos cuadrados

Debido a que se toman más de un punto los vectores mostrados serían matrices de puntos, por ello expresado en formato matricial se obtiene:

$$E = M - C \cdot \hat{\theta}$$

Figura Calibración.12. Error de calibración en formato matricial

Para aplicar mínimos cuadrados se escoge un índice J como el cuadrado del error:

$$J = \frac{1}{2} \cdot \sum_{i=1}^{N} {e_i}^2 = \frac{1}{2} \cdot E^T \cdot E$$

Figura Calibración.13. Índice de mínimos cuadrados del error

Para minimizar este error se obtiene la derivada respecto de la matriz de calibración y se iguala a 0:

$$\frac{\partial J}{\partial \hat{\theta}} = -C^T \cdot \left(M - C \cdot \hat{\theta}\right) = 0 \rightarrow \hat{\theta} = (C^T \cdot C)^{-1} \cdot C^T \cdot M$$

Figura Calibración.14. Obtención de la matriz de calibración mediante la derivada del error por mínimos cuadrados

Una vez obtenida se realiza la siguiente operación para obtener los puntos del robot respecto de los de la cámara:

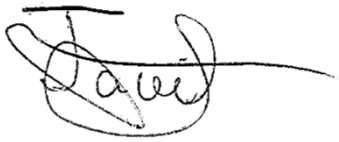
$${P_M}^T = \hat{\theta}^T \cdot {P_C}^T = \begin{bmatrix} \hat{\theta}_{11} & \hat{\theta}_{21} & \hat{\theta}_{31} \\ \hat{\theta}_{12} & \hat{\theta}_{22} & \hat{\theta}_{32} \\ \hat{\theta}_{13} & \hat{\theta}_{23} & \hat{\theta}_{33} \end{bmatrix}$$

Figura Calibración.15. Obtención de puntos del robot con los leídos por la cámara

Ambos métodos son igual de viables, sin embargo, se probaron ambos y el de homografía dio un muy ligero error superior al de mínimos cuadrados.

VALENCIA, JULIO 2023

Javier Gamir Artesero



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

ANEJO Nº3: BIBLIOGRAFÍA

[1]. *Automotive robotics market size, share & growth [2021-2028]*. (s. f.). Recuperado 14 de junio de 2023, de https://www.fortunebusinessinsights.com/automotive-robotics-market-105578

[2]. Canu, S. (2021, marzo 11). *Distance detection with Depth Camera (Intel Realsense d435i)*. Pysource. https://pysource.com/2021/03/11/distance-detection-with-depth-camera-intel-realsense-d435i/

[3]. Computer vision. (2023). En *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Computer_vision&oldid=1160058115

[4]. *Depth camera d435i*. (s. f.). Intel® RealSense™ Depth and Tracking Cameras. Recuperado 14 de junio de 2023, de https://www.intelrealsense.com/depth-camera-d435i/

[5]. Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J., & Socher, R. (2021). Deep learning-enabled medical computer vision. *Npj Digital Medicine*, *4*(1), 1-9. https://doi.org/10.1038/s41746-020-00376-2

[6]. Gomes, J. F. S., & Leta, F. R. (2012). Applications of computer vision techniques in the agriculture and food industry: A review. *European Food Research and Technology*, *235*(6), 989-1000. https://doi.org/10.1007/s00217-012-1844-2

[7]. *HOW TO - Programación con sockets*. (s. f.). Python documentation. Recuperado 14 de junio de 2023, de https://docs.python.org/3/howto/sockets.html

[8]. Khatib, O., & Siciliano, B. (Eds.). (2016). *Springer handbook of robotics* (2nd ed. 2016). Springer International Publishing : Imprint: Springer.

[9]. *Measure region properties—Skimage 0.21.0 documentation*. (s. f.). Recuperado 14 de junio de 2023, de https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_regionprops.html

[10]. Modelo de color HSV. (2022). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Modelo_de_color_HSV&oldid=148004756

[11]. *OpenCV: OpenCV modules*. (s. f.). Recuperado 14 de junio de 2023, de https://docs.opencv.org/4.x/

[12]. *Python*. (s. f.). Intel® RealSense™ Developer Documentation. Recuperado 14 de junio de 2023, de https://dev.intelrealsense.com/docs/python2

[13]. *Riesgos ergonómicos—Trabajos repetitivos—Insst—Portal insst—Insst*. (s. f.). Portal INSST. Recuperado 14 de junio de 2023, de https://www.insst.es/materias/riesgos/riesgos-ergonomicos/carga-de-trabajo/trabajos-repetitivos

[14]. *Robotstudio*. (2023, mayo 19). ABB Robotics User Forums. https://forums.robotstudio.com/categories/robotstudio

[15]. Sankowski, D., & Nowakowski, J. (2014). *Computer vision in robotics and industrial applications*. World Scientific.

[16]. *Socket communication*. (2013, julio 15). ABB Robotics User Forums. https://forums.robotstudio.com/discussion/8119/socket-communication

[17]. Van Der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., & Yu, T. (2014). Scikit-image: Image processing in python. *PeerJ*, *2*, e453. https://doi.org/10.7717/peerj.453

[18]. *Especificaciones del producto IRB140*. ABB Library. Recuperado 14 de junio de 2023 https://library.e.abb.com/public/84e6cb203eef4658839e7cf66e8eaf71/3HAC041346%20PS%20IRB%20140-es.pdf?x-sign=i6wBjS8ViFzMvIEFjnXXLDzi6JE1nG5Ihan+5+eSpZszxWQwNzfLZGLuRQALFfj8

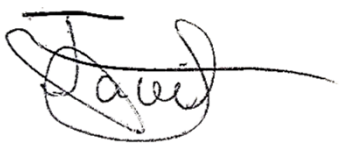
[19]. *Technical reference manual RAPID Instructions, Functions and Data types*. ABB Library. Recuperado 14 de junio de 2023. https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf

[20]. *Manual del operador RobotStudio*. ABB Library. Recuperado 14 de junio de 2023. https://library.e.abb.com/public/6aeb483836740e11c1257b4b0052375b/3HAC032104-005_revE_es.pdf

[21]. Carlos Ricolfe Viala, Antonio Sánchez Salmerón, Ángel Valera Fernandez. . *Tema 6 Visión Artificial* [Diapositivas de PowerPoint]. Facultad de Informática, Universidad Politécnica de Valenica. Diapositivas de 70 a 74.

VALENCIA, JULIO 2023

Javier Gamir Artesero



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

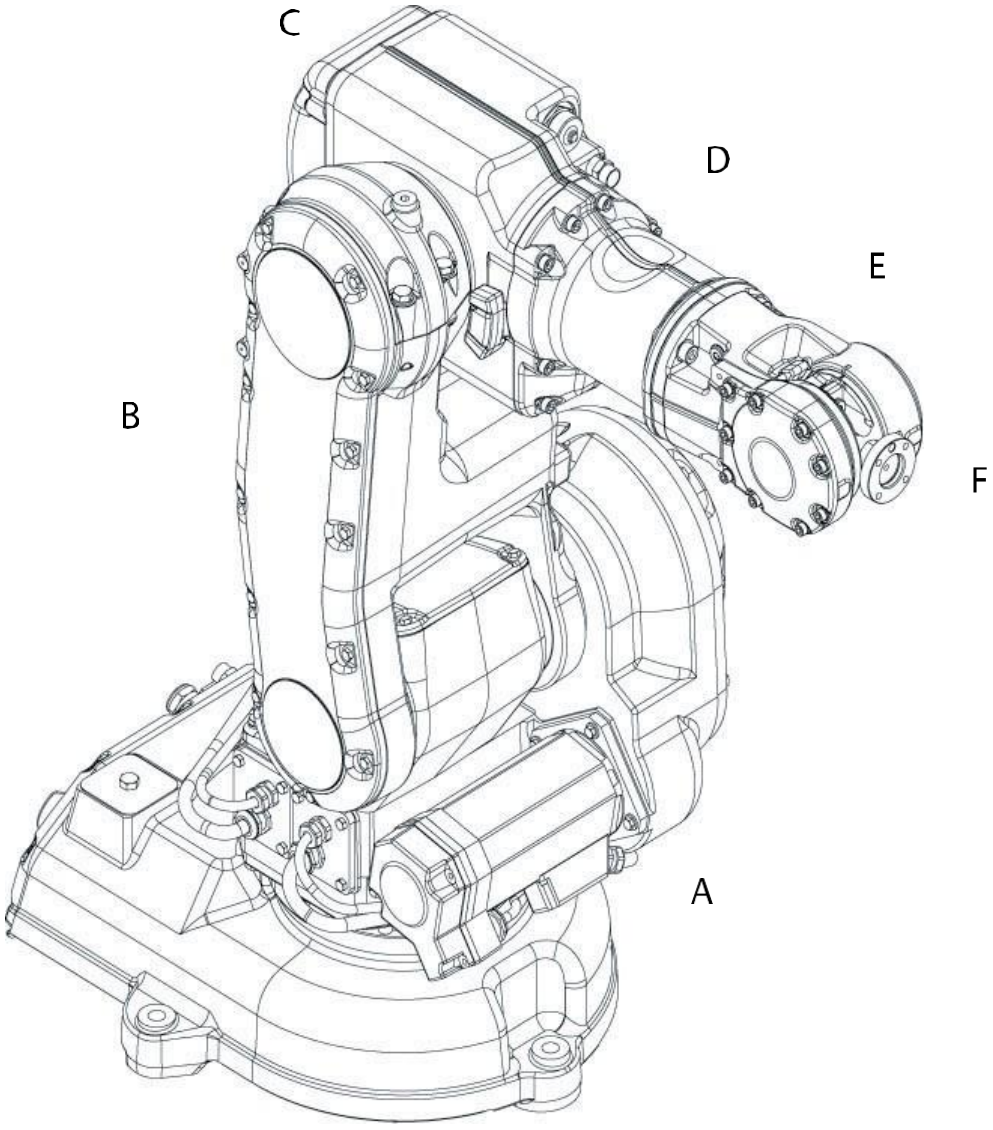
Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

ANEJO Nº4: FICHAS DE DATOS

Índice

IRB140

Ejes del manipulador



xx1000000859

Posición	Descripción	Posición	Descripción
A	Eje 1	B	Eje 2
C	Eje 3	D	Eje 4
E	Eje 5	F	Eje 6

Generalidades

El IRB 140-6/0.8 está disponible en dos versiones y todos admiten el montaje enel suelo, en posición invertida o en pared con cualquier ángulo (inclinado alrededor del eje X o Y). La variante de alta velocidad, la IRB 140T, ofrece un tiempo de cicloaún más reducido:

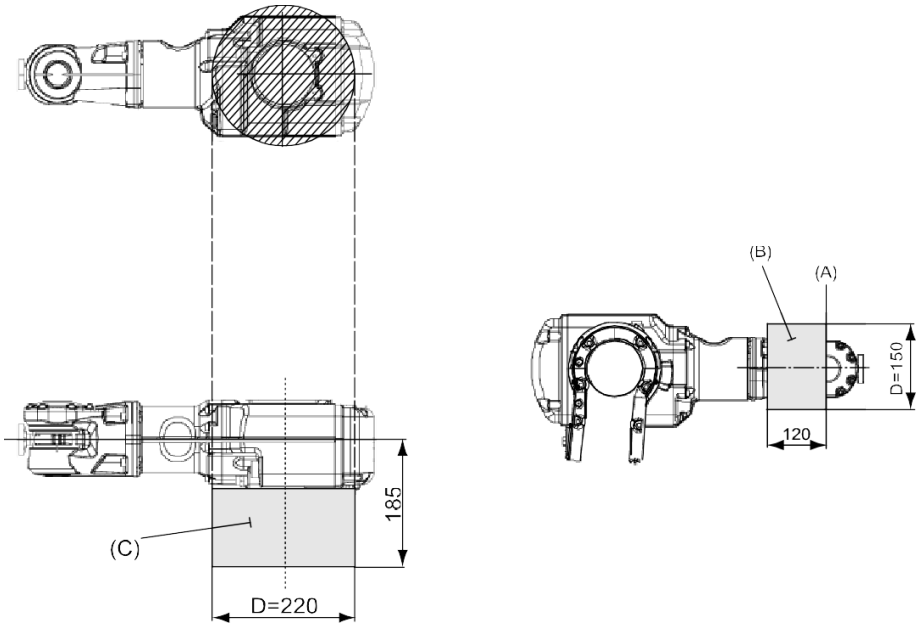
Tipo de robot	Capacidad de manejo (kg)	Alcance (m)	Tipo de robot
IRB 140	6 kg	0.8 m	IRB 140
IRB 140T	6 kg	0.8 m	IRB 140T

Peso del manipulador

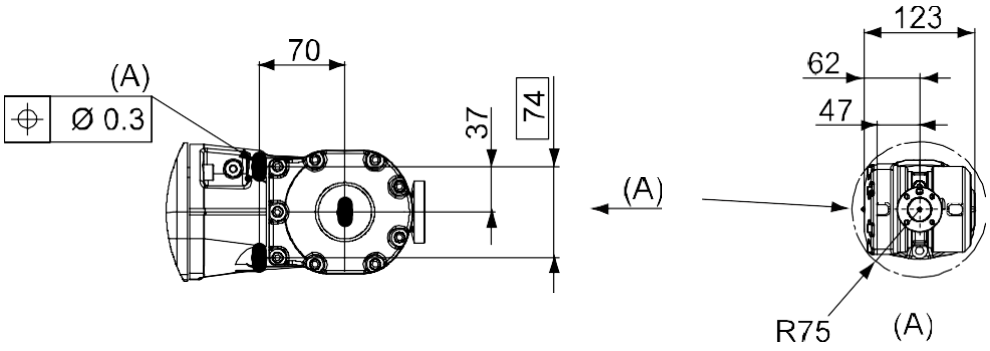
Datos	Descripción
Manipulador	98 kg (sin los cables al controlador)

Montaje de equipos

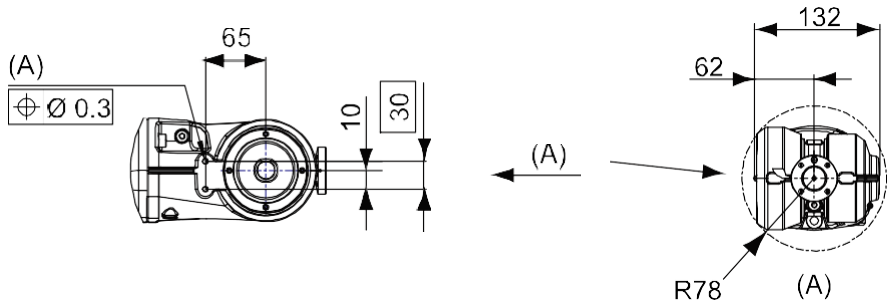
Es posible montar cargas adicionales en la muñeca y en la carcasa del brazo superior. Las definiciones de las áreas de carga y la carga permitida se muestranen la figura que aparece a continuación. El centro de gravedad de la carga adicionaldebe estar dentro de las áreas de carga marcadas. El robot se suministra con orificios para el montaje de equipos adicionales.



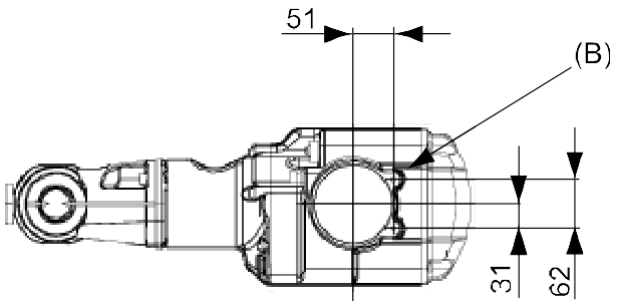
Diseño de la muñeca del IRB 140 IRC5



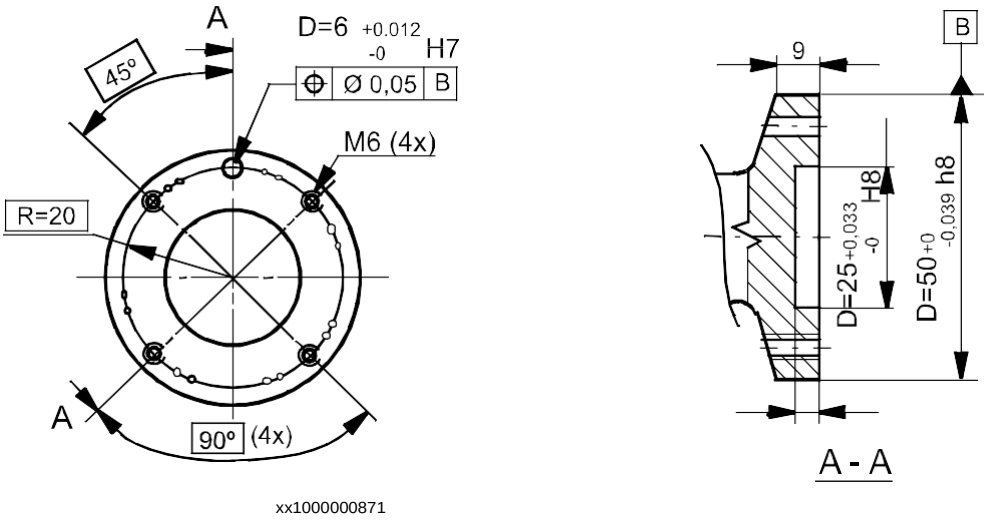
Diseño de la muñeca del IRB 140 IRC5, tipo C



Carcasa del brazo superior



Brida para herramientas del robot



Cámara Intel Realsense 435i

Features

Use environment:
Indoor/Outdoor

Ideal range:
.3 m to 3 m

Image sensor technology:
Global Shutter

Depth

Depth technology:
Stereoscopic

Depth Field of View (FOV):
87° × 58°

Minimum depth distance (Min-Z) at max resolution:
~28 cm

Depth output resolution:
Up to 1280 × 720

Depth Accuracy:
<2% at 2 m[1]

Depth frame rate:
Up to 90 fps

RGB

RGB frame resolution:
1920 × 1080

RGB sensor FOV (H × V):
69° × 42°

RGB frame rate:
30 fps

RGB sensor resolution:
2 MP

RGB sensor technology:
Rolling Shutter

Major Components

Camera module:
Intel RealSense Module D430 + RGB Camera

Vision processor board:
Intel RealSense Vision Processor D4

Physical

Form factor:
Camera Peripheral

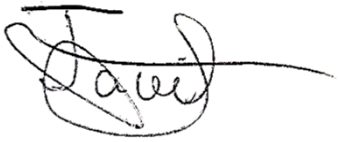
Connectors:
USB-C* 3.1 Gen 1*

Length × Depth × Height:
90 mm × 25 mm × 25 mm

Mounting mechanism:
– One 1/4-20 UNC thread mounting point.
– Two M3 thread mounting points.

VALENCIA, JULIO 2023

Javier Gamir Artesero



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

ANEJO Nº5: Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030

Objetivos de Desarrollo Sostenibles	Alto	**Medio**	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				✓
ODS 2. **Hambre cero.**	✓			
ODS 3. **Salud y bienestar.**				✓
ODS 4. **Educación de calidad.**				✓
ODS 5. **Igualdad de género.**				✓
ODS 6. **Agua limpia y saneamiento.**				✓
ODS 7. **Energía asequible y no contaminante.**				✓
ODS 8. **Trabajo decente y crecimiento económico.**	✓			
ODS 9. **Industria, innovación e infraestructuras.**		✓		
ODS 10. **Reducción de las desigualdades**.				✓
ODS 11. **Ciudades y comunidades sostenibles**.				✓
ODS 12. **Producción y consumo responsables.**		✓		
ODS 13. **Acción por el clima.**				✓
ODS 14. **Vida submarina.**				✓
ODS 15. **Vida de ecosistemas terrestres.**				✓
ODS 16. **Paz, justicia e instituciones sólidas.**				✓
ODS 17. **Alianzas para lograr objetivos.**			✓	

Descripción de la alineación del TFG/TFM con los ODS

ODS 2. Hambre Cero: El proyecto tiene como objetivo principal mejorar la eficiencia en el empaquetado de productos alimenticios mejorando así su producción y por lo tanto aportando un impacto positivo en el suministro de alimentos.

Esto se encuentra alienado con los siguientes puntos de la ODS 2:

-2.1: Ayudará al objetivo de poner fin al hambre al permitir una mayor producción de alimentos y por lo tanto aumentar la cantidad de estos.

-2.4: Ayudará a la producción sostenible de alimentos al reducir el error humano y basarse en la energía eléctrica.

-2.c: Contribuirá al buen funcionamiento de los mercados al aumentar la disponibilidad de productos.

ODS 8. Trabajo decente y crecimiento económico: El proyecto permitirá eliminar una tarea repetitiva y físicamente exigente como es el empaquetado de productos reconocidas por el Instituto Nacional de Seguridad y Salud en el Trabajo como posible causa de trastornos musculoesqueléticos y siendo una de las principales causas de enfermedad y lesiones de origen laboral.

Lo recientemente explicado se encuentra alineado con los siguientes subapartados de la ODS 8:

-8.2: Se contribuye mediante innovación y modernización tecnológica.

-8.4: Mejora el consumo eficiente de recursos.

-8.6: Puede crear puestos de trabajo de jóvenes ingenieros.

ODS 9. Industria, innovación e infraestructuras: Impulsa la innovación en la industria alimentaria al aumentar el número de aplicaciones en ella de elementos de robótica y visión artificial. Los subapartados que se encuentran alineados con el proyecto son:

-9.1: La implementación de un robot antropomórfico y un sistema de visión artificial infiere la necesidad de la creación de una infraestructura avanzada, sostenible y de calidad. Además, impulsa la creación de otras infraestructuras como puede ser de transporte de los productos empaquetados.

-9.4: Reduce el número de recursos utilizados en el proceso.

ODS 12. Producción y consumo responsables: Se reduce el número de alimentos desechados por el error humano y permite una manipulación de alimentos higiénica y adecuada. Esto se encuentra en consonancia con los siguientes subapartados:

-12.3: Permite reducir el desperdicio de alimentos.

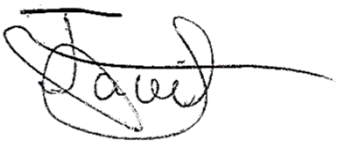
-12.6: Aporta un enfoque sostenible para el proceso de empaquetado automático.

ODS 17. Alianzas para lograr objetivos: Mediante este proyecto se promueve el intercambio de información y cooperación de la industria robótica y la industria alimentaria.

El subapartado que se encuentra alineado con esto es el 17.6 al mejorar la cooperación regional e internacional en materia de ciencia, tecnología e innovación aumentar el intercambio de conocimientos en condiciones mutuamente convenidas

VALENCIA, JULIO 2023

Javier Gamir Artesero



Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

DOCUMENTO Nº2: PLANOS

Grado en Ingeniería Electrónica Industrial y Automática

Autor

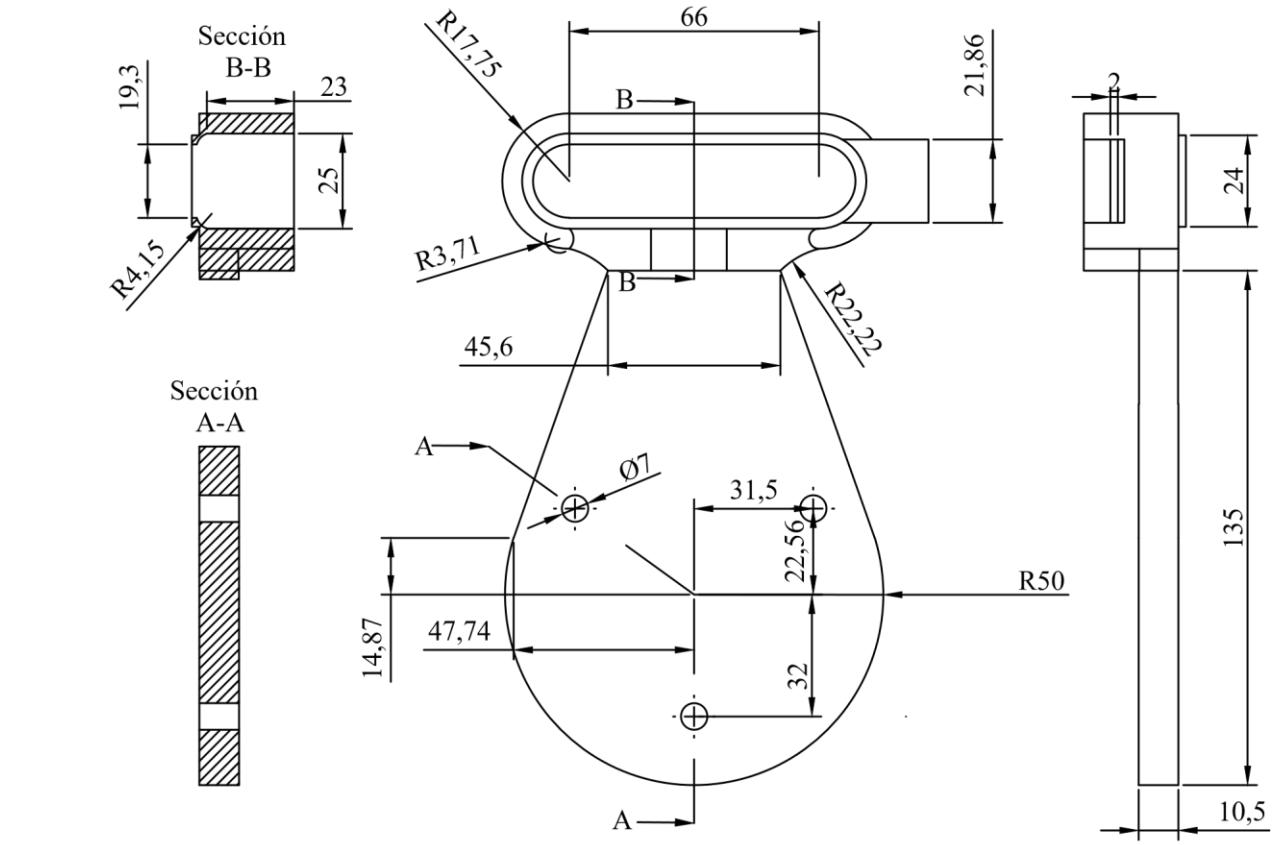
Javier Gamir Artesero

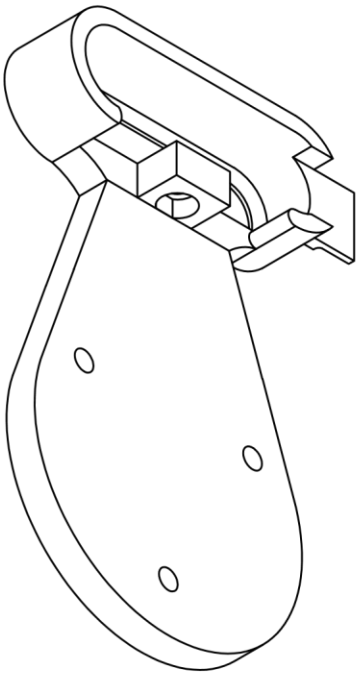
Tutor

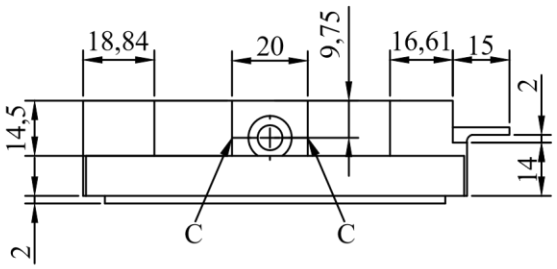
Carlos Ricolfe Viala

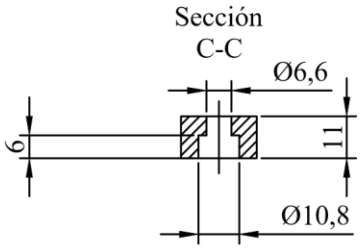
CURSO ACADÉMICO: 2022/2023

Índice









ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO PROYECTO DE OFICINA TÉCNICA Titulo del proyecto: Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico Titular: Javier Gamir Artesero		Fecha: 25/06/2023 Escala: 1:2
Autor: Javier Gamir Artesero	Denominación del plano Soporte Cámara Realsense 435i	Número del plano 01

Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

DOCUMENTO Nº3: PLIEGO DE CONDICIONES

Grado en Ingeniería Electrónica Industrial y Automática

Autor

Javier Gamir Artesero

Tutor

Carlos Ricolfe Viala

CURSO ACADÉMICO: 2022/2023

Índice

1. Condiciones Generales

Este proyecto tiene carácter de obligado cumplimiento una vez sellado y legalizado, debiendo ser objeto de aprobación previa todas aquellas modificaciones al mismo durante su ejecución.

1.1. Vigencia

Este Pliego de Condiciones, con todos sus articulados, estará en vigor durante la ejecución del desarrollo del proceso y hasta la terminación de este, entendiéndose que las partes a que hace referencia éste, se aceptarán en todos sus puntos por el adjudicatario del proceso. Frente a posibles discrepancias, el orden de prioridad de los documentos básicos del Proyecto será el siguiente:

1).- Planos.

2).- Pliego de Condiciones.

3).- Presupuesto.

4).- Memoria.

1.2. Descripción

Esta especificación se refiere a la programación del robot IRB140 del laboratorio de Robótica del DISA junto a sus herramientas externas como son la cinta transportadora y los fotosensores. También se incluye la instalación en el mismo de una cámara Intel Realsense 435i junto a un acople.

Quedan excluidas de esta especificación la instalación del robot en el alojamiento y la instalación eléctrica de los equipos necesarios para la conexión de la cinta transportadora y el resto de las salidas digitales implementadas en el controlador del robot. También queda excluida la instalación neumática para el uso de la herramienta ventosa.

1.3. Pliegos oficiales

Respecto al ámbito europeo, la normativa que hace referencia al proyecto es Directiva 2006/42/CE del Parlamento Europeo y del Consejo, de 17 de mayo de 2006, relativa a las máquinas y por la que se modifica la Directiva 95/16/CE.

Asimismo, la directiva 2014/35/UE del Parlamento Europeo y del Consejo de 26 de febrero de 2014 sobre la armonización de las legislaciones de los Estados miembros en materia de comercialización de material eléctrico destinado a utilizarse con determinados límites de tensión.

También el Real Decreto 187/2016, de 6 de mayo, por el que se regulan las exigencias de seguridad del material eléctrico destinado a ser utilizado en determinados límites de tensión.

Además, del Real Decreto 186/2016, de 6 de mayo, por el que se regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos.

A su vez, el Real Decreto 1072/2015, de 27 de noviembre, por el que se modifica el Real Decreto 2200/1995, de 28 de diciembre, por el que se aprueba el Reglamento de la Infraestructura para la Calidad y la Seguridad Industrial.

1.4. Modificaciones

Durante la ejecución del proyecto, se podrán realizar cuantas modificaciones se estimen oportunas, siempre que las mismas sean aprobadas por el responsable de la Dirección del Proyecto, y en todo momento, de acuerdo con la entidad contratante.

2. Condiciones Técnicas

Las funciones de director del proyecto son las de revisión del trabajo realizado, programación de los trabajos, reconocimiento de los materiales utilizados y autorizaciones referentes al proyecto.

En el caso de que los materiales no fueran especificados, los que se utilicen deberán cumplir los requisitos mínimos de funcionamiento y tolerancia que se requiere, siendo obligatorio que sean normalizados y sometidos a la aprobación del director del proyecto.

Todos los trabajos se ejecutarán con estricta sujeción al proyecto que ha servido de base a la contratación y a las modificaciones que hayan sido aprobadas. En caso de dudas u omisiones, o con motivo de reforma del presupuesto, se formará un comité entre proyectistas, director del proyecto y, si se cree oportuno, el contratista, para decidir la solución más adecuada y económica.

3. Materiales

3.1. Arquitectura del sistema

3.1.1. Robot

El robot antropomórfico ha de ser un robot industrial de 6 ejes con una carga útil de 6 kg y diseñado específicamente para industrias de fabricación que utilizan una automatización flexible basada en robots. Además, deberá tener una herramienta ventosa y una unidad de control. El modelo deberá de ser el IRB140 o equivalente al juicio del director del proyecto.

3.1.2. Cámara

La cámara tendrá unas dimensiones de 90 mm x 25 mm x 25 mm, deberá tener una resolución no menor de 1920x1080 y una tasa de mínimo 30 fps. Además, será necesario que esta cuente con medición de profundidad y con entrada de USB 3.0. Por ello el modelo será la cámara Intel Realsense d435i o equivalente al juicio del director del proyecto.

3.1.3. Ordenador

El ordenador deberá tener disponibilidad de instalación del programa Python con una de sus versiones de Python 3, a su vez deberá tener acceso a un switch para acceder a la red del robot y contar con una velocidad de procesamiento mínima de 2500 MHz y una RAM mínima de 4 GB. El modelo será un GREED® Intel Core i7 4790 Multimedia PC o equivalente al juicio del director del proyecto.

3.1.4. Cable USB

El cable USB de conexión de la cámara con el ordenador deberá de ser de 5 metros tipo USB 3.0. El modelo será un CLEEFUN o equivalente al juicio del director del proyecto.

3.1.5. Cinta transportadora

La cinta transportadora deberá de ser de tipo de banda y que sea posible manipularla como salida digital, además deberá de tener unas dimensiones mínimas de 200 mm de ancho y 500 mm metro de longitud. El modelo será VEVOR o equivalente al juicio del director del proyecto.

3.1.6. Sensor fotoeléctrico

El sensor fotoeléctrico deberá de tener un alcance mínimo de 200 mm y una alimentación de 10 a 30 Voltios. El modelo será Sensor Fotoeléctrico Láser NPN o equivalente al juicio del director del proyecto.

3.2. Software

3.2.1. Software Visión Artificial

El programa relacionado con la cámara deberá de tener librerías de visión artificial que sean capaces de realizar transformaciones básicas a imágenes como umbralización, erosión, dilatación, cierre y eliminación de bordes.

Además, deberá de ser capaz de aplicar etiquetado de figuras y obtención de sus características como área, centro de gravedad y orientación.

También deberá de ser capaz de comunicarse mediante el uso de sockets. El programa será Python 3.0 o equivalente al juicio del director del proyecto.

3.2.2. Software control del robot

El programa relacionado con el robot deberá permitir la comunicación por sockets y el manejo absoluto del mismo, mediante movimientos absolutos o lineales. El programa será RobotStudio o equivalente al juicio del director del proyecto.

4. Condiciones de uso, mantenimiento y seguridad.

4.1. Obligaciones de usuario

El mantenimiento del robot, la cámara, la cinta y el sensor foto eléctrico será a cargo de la unidad contratante. Estas operaciones se deben realizar siguiendo el manual que ofrece cada vendedor de cada producto.

5. Certificados y documentación

Con anterioridad al comienzo de los trabajos de la programación del presente proyecto, la Dirección del Proyecto podrá solicitar certificados de homologación de los materiales de que se compone el mismo, así como documentación y catálogos en los que se indiquen sus características principales.

6. Condiciones de la ejecución

6.1. Descripción del proceso de ejecución

6.1.1. Disposición de materiales

El robot deberá estar colocado en una zona segura como una celda de trabajo, dispuesto con la cinta transportadora a un lado y con una zona libre para el empaquetado de productos en otro lado.

La cámara deberá de ser instalada en el robot haciendo uso del acople creado para la misma presente en el documento de planos entre la brida del robot y la herramienta ventosa.

El ordenador deberá encontrarse cerca del switch de conexión a la red del robot y a una distancia de seguridad del robot, no superior a 5 metros de este.

La cinta transportadora ha de encontrarse dispuesta en uno de los lados del robot y limpia, para evitar la detección de manchas como objetos. Además, deberá estar conectada a la unidad de control del robot.

El sensor fotoeléctrico deberá encontrarse al final de la cinta transportadora en posición perpendicular al ancho de esta. Además, deberá estar conectado a la unidad de control del robot.

Las instalaciones deberán de estar bien iluminadas ya sea de forma natural o de forma artificial.

6.1.2. Ejecución del software

Primeramente, deberá de tenerse instalado el programa Python 3 con el conjunto de librerías de 'OpenCV' y 'Scikit-Image' en el ordenador de trabajo.

Tras esto deben colocarse las marcas de empaquetado en la zona determinada para ello.

A continuación, se deberá de dar inicio al programa del robot para que este inicie la comunicación como servidor, una vez dado se da inicio al programa del ordenador para que inicie como cliente.

Por último, disponer los objetos en la cinta transportadora para ser empaquetados.

7. Pruebas de servicio

-Se deberá de comprobar que no ante un cambio de funcionamiento de modo manual a automático el usuario ha de dar su confirmación.

-Se deberá comprobar que para el modo de funcionamiento manual del robot es necesario activar el movimiento de forma que el usuario ha de estar atento a lo que está haciendo.

-Se deberá de comprobar que la cámara se encuentra calibrada con el robot, para ello se deberá obtener un punto real con el robot y un punto con la cámara, transformarlo y comprobar si son el mismo.

-Se deberá comprobar la iluminación de la instalación mediante si la cámara detecta correctamente las marcas de empaquetado y los objetos previo al inicio del proceso.

8. Certificados

8.1. Marcado CE

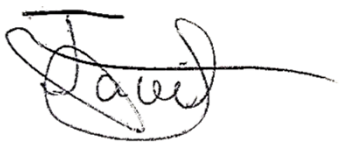
Todos los productos adquiridos deberán contar con su pertinente marcado CE. No se deberá admitir bajo ningún concepto aquellos productos que no cuenten con uno.

8.2. Ecodiseño

Todos los materiales y productos adquiridos deberán tener el “Certificado ecológico europeo” que acredite que los productos se producen siguiendo una estricta normativa basada en la protección del medio ambiente y del propio producto.

VALENCIA, JULIO 2023

Javier Gamir Artesero



Diseño y puesta en marcha de un sistema de empaquetado flexible mediante robot antropomórfico

DOCUMENTO Nº4: PRESUPUESTO

Grado en Ingeniería Electrónica Industrial y Automática

Autor

Javier Gamir Artesero

Tutor

Carlos Ricolfe Viala

CURSO ACADÉMICO: 2022/2023

Índice

1. Cuadro de Precios Unitarios

Referencia	Unidades	Descripción	Precio (€)
Materiales			
m1	ud.	IRB140 2017	18500,00
m2	ud.	Cámara Intel Realsense d435i	334,00
m3	ud.	Cable USB Extensor 3.0 5 metros	8,00
m4	ud.	Plástico PVC	0,01
m5	ud.	GREED® Intel Core i7 4790 Multimedia PC	499,90
m6	ud.	Cinta transportadora VEVOR 500x200x750mm	276,00
m7	ud.	Sensor Fotoeléctrico Láser NPN	8,90
m8	ud.	Python	0,00
m9	ud.	Herramienta Ventosa	200,00
m10	ud.	RobotStudio	3400,00
Secciones			
s1	ch	Impresión 3D	30,00
s2	h	Programación del Robot	
s3	h	Programación de la visión Artificial	
M.O.D			
h1	h	Empresa de impresión 3d	30,00
h2	h	Programador	17,00

2. Cuadro de Precios Auxiliares

Referencia	Unidades	Descripción	Precio	Cantidad	Parcial
d1	ud.	Fabricación de Acople para la cámara			
Materiales					
m4	ud.	Plástico PVC	0,01	250,00	1,55
Secciones					
S1	ch	Impresión 3D	30,00	1	30
M.O.D.					
h1	h	Empresa de impresión 3d	30,00	1	30

Precio Total d1=61.55 €

Referencia	Unidades	Descripción	Precio	Cantidad	Parcial
d2	ud.	Programar Robot			
Materiales					
m1	ud.	IRB140 2017	18500,00	1,00	18500,00
m5	ud.	GREED® Intel Core i7 4790 Multimedia PC	499,90	1,00	499,90
m6	ud.	Cinta transportadora VEVOR 500x200x750mm	276,00	1,00	276,00
m7	ud.	Sensor Fotoeléctrico Láser NPN	8,90	1,00	8,90
m9	ud.	Herramienta Ventosa	200,00	1,00	200,00
m10	ud.	RobotStudio	3400,00	1,00	3400,00
Secciones					
s2	h	Programación del Robot		30,00	0
M.O.D.					
h2	h	Programador	17,00	30,00	510,00

Precio Total d2=19010 €

Referencia	Unidades	Descripción	Precio	Cantidad	Parcial
d3	ud.	Programar Visión Artificial			
Materiales					
m2	ud.	Cámara Intel Realsense d435i	334,00	1,00	334,00
m3	ud.	Cable USB Extensor 3.0 5 metros	8,00	1,00	8,00
m5	ud.	GREED® Intel Core i7 4790 Multimedia PC	0,00	1,00	0,00
m7	ud.	Python	0,00	1,00	0,00
Secciones					
S3	h	Programación de la visión Artificial		40,00	0
M.O.D.					
h2	h	Programador	17,00	40,00	680,00

Precio Total d3=1022 €

3. Cuadro de Precios Descompuestos

D1	ud.	Desarrollo del sistema			
Materiales					
d1	ud.	Fabricación de Acople para la cámara	61,55	1,00	61,55
d2	ud.	Programar Robot	19010,00	1,00	19010,00
d3	ud.	Programar Visión Artificial	1022,00	1,00	1022,00
CD02					
%	Porcentaje	Costes Directos	0,02		401,87

Precio Total D1=20495,42 €

4. Resumen del presupuesto

Referencia	Unidades	Descripción	Precio	Cantidad	Importe
Ref	Ud	Descripción	Precio	Cantidad	IMPORTE
D1	ud	Montaje de la mecánica del coche RC	20495,42	1,00	20495,42
Presupuesto de ejecución material			762,42	1,00	20495,42
Gastos Generales	%		0,06	1,00	1265,08
Beneficio industrial	%		0,13	1,00	2664,40
Presupuesto de ejecución por contrata			24424,90	1,00	24424,90
Porcentaje de honorarios y trámites	%		0,10	1,00	2049,54
IVA sobre el porcentaje de honorarios	%		0,21	1,00	4304,04

TOTAL PRESUPUESTO DE EJECUCIÓN GENERAL=

TRENTA MIL SETECIENTOS SETENTA Y OCHO CON CUARENTA Y OCHO EUROS (30778,48)

VALENCIA, JULIO 2023

Javier Gamir Artesero