

Escuela de Ingeniería Industrial

TRABAJO FIN DE MÁSTER

Diseño de una estación de despaletizado mediante un sistema de visión robótica

Máster en ingeniería industrial

ALUMNO:

Daniel Lamas Novoa

DIRECTORES:

Enrique Paz Domonte

Ángel Antonio Fernández Montero

UniversidadeVigo

RESUMEN

En esta memoria se detalla el desarrollo de un proyecto piloto de una estación de despaletizado de tablas de madera mediante visión robótica en la sede de Santiago de Compostela de la empresa Financiera Maderera S.A. (FINSA). El diseño de la estación está orientado a un lugar concreto en la fábrica donde actualmente esta tarea se realiza de forma manual. Para poder hacer un buen diseño, fue necesario hacer un estudio de cómo se trabaja actualmente en dicha estación, que necesidades tiene y cómo es su entorno.

Conocida la problemática, se procedió al diseño de la arquitectura, estando compuesta por 3 sistemas bien diferenciados, y sensores y actuadores auxiliares. Estos 3 sistemas son: el sistema de gobierno, dirigido por un procesador lógico programable (PLC); el sistema de visión artificial; y el robótico. Estos 3 sistemas se comunican entre sí, estando dirigidos por el PLC.

El siguiente paso fue definir la secuencia de operaciones. Se diseñó la aplicación para que pudiese funcionar con cualquier medida de tabla. Cuando el PLC lo ordena, el sistema de visión artificial reconoce las tablas de un nivel del palé, calculando sus centros y orientaciones, y envía la información al PLC. Con los datos del sistema de visión y de los demás sensores, el PLC le da la orden al robot de ir retirando las tablas, enviándole los datos necesarios para ello. Este procedimiento se repite nivel tras nivel hasta completar el palé.

A continuación, se eligieron las tecnologías y herramientas que conforman la estación. Esta elección se hizo teniendo en cuenta no solo las necesidades de este proyecto, sino también el presupuesto disponible, y las necesidades y familiaridades con proveedores de la fábrica. Se decidió utilizar tecnología 2D para el sistema de visión y un sensor de distancia para obtener los datos de profundidad, y que tanto este sensor como la cámara fuesen solidarios a la garra del robot. También se decidió extraer las piezas mediante el uso de una garra de vacío por tener las tablas un acabado superficial lo suficientemente liso y ser poco porosas.

Elegidas las herramientas, se hizo el diseño de la estructura de unión entre la garra de vacío, la cámara, el sensor de distancia y la muñeca del robot. Se realizaron varios diseños, priorizando diferentes factores y pensados para diferentes métodos de fabricación, decidiendo finalmente realizar la estructura con fabricación 3D y diseñándola para que facilitase las tareas de calibración y redujese los tiempos de ciclo.

A continuación, se procedió a la programación de los 3 sistemas. El primer objetivo fue conseguir el reconocimiento de las tablas. Para ello se probaron diferentes algoritmos hasta decidir usar un algoritmo de crecimiento de región (*Region Growing*). Lograda la localización de las tablas, se programaron los otros 2 sistemas y se comprobaron sus funcionamientos por separado de forma virtual.

Una vez las herramientas llegaron a fábrica, se procedió al montaje de la estación en el laboratorio. En este punto se configuraron las comunicaciones entre los sistemas. Se hicieron las calibraciones de las herramientas del robot y de la conversión a magnitudes físicas del sistema de visión, y se ajustaron los parámetros de la cámara y la colocación de la iluminación. También se realizaron ajustes en los programas, necesarios para el correcto funcionamiento del conjunto.

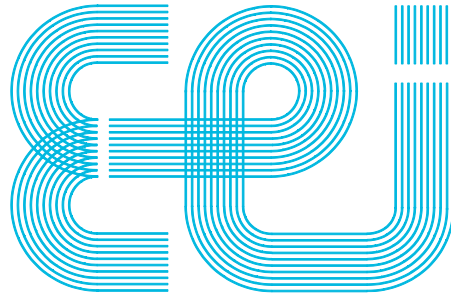
Finalmente, se realizaron pruebas con el fin de valorar la precisión y el tiempo de ciclo de la estación de visión robótica.

PALABRAS CLAVE

- Visión robótica
- Visión artificial
- Robótica
- Despaletizado
- *Region Growing*

ÍNDICE GENERAL

- Memoria
- Presupuesto
- Planos
- Diagrama de Gantt
- Anexo I: programas de la estación
- Anexo II: programas de desarrollo



Escuela de Ingeniería Industrial

TRABAJO FIN DE MÁSTER

*Diseño de una estación de despaletizado mediante un sistema
de visión robótica*

Máster en ingeniería industrial

Documento

MEMORIA

UniversidadeVigo

ÍNDICE DE LA MEMORIA

1	Introducción	1
1.1	Objetivos.....	1
1.2	Antecedentes.....	1
1.2.1	Visión artificial.....	2
1.2.2	Robótica	6
1.2.3	Visión robótica	9
2	Análisis en campo	11
3	Arquitectura y funcionamiento	13
3.1	Jerarquía.....	13
3.2	Posicionamiento.....	14
3.3	Comunicaciones.....	15
3.4	Alimentaciones	16
3.5	Funcionamiento general	18
4	Elección y diseño de herramientas.....	20
4.1	Robot	20
4.1.1	Manipulador (24).....	20
4.1.2	Unidad de control (25)	20
4.1.3	Software de desarrollo del programa (26).....	21
4.1.4	FlexPendant (27)	21
4.2	Sistema de visión artificial	21
4.2.1	Iluminación (28).....	21
4.2.2	Cámara (30).....	23
4.2.3	Lente.....	23
4.2.4	Filtro	24
4.2.5	Software de desarrollo del programa (31).....	24
4.2.6	PC	24
4.2.7	Plato de calibración	25
4.2.8	Software de configuración de la cámara (34).....	25
4.3	Sistema de agarre de las piezas.....	25
4.3.1	Garra de vacío (35).....	25
4.3.2	Mecanismo de control apertura/cierre (36) (37)	26
4.4	Sensor de distancia (38).....	26
4.5	Herramienta del robot.....	27
4.5.1	Software de diseño	27
4.5.2	Soporte de las herramientas del robot	27
4.6	Unidad de gobierno	31

4.6.1 Controlador lógico programable (PLC)	31
4.6.2 Periferia distribuida	32
4.6.3 Software de desarrollo del programa	33
4.7 Switch	33
4.8 Mesa de trabajo.....	33
4.9 Soporte de iluminación	34
4.10 Botonera.....	35
4.11 Elementos auxiliares.....	35
5 Montaje	36
6 PLC	40
6.1 TiaPortal V15	40
6.2 Hardware	40
6.3 Software.....	42
6.3.1 Main	43
6.3.2 Actualizacion_entradas	43
6.3.3 Modo_funcionamiento	44
6.3.4 F_Robot.....	46
6.3.5 F_Vision_artificial PLC	49
6.3.6 Actualizacion_salidas.....	50
7 Robot.....	51
7.1 RobotStudio 6.8 y Rapid	51
7.2 Estación virtual	53
7.3 Software.....	54
7.3.1 Procedimiento Main	56
7.3.2 Objetos de trabajo y puntos	57
7.3.3 Movimientos.....	57
7.4 Calibraciones	58
8 Visión artificial	61
8.1 Halcon.....	61
8.2 Algoritmo de reconocimiento	61
8.2.1 Canny	63
8.2.2 Deriche	65
8.2.3 Regiongrowing Segmentation	67
8.3 Ajuste de la cámara.....	68
8.4 Software.....	69
8.5 Calibración.....	71
9 HMI.....	73
10 Comunicaciones	74
10.1 Cámara-PC visión.....	74
10.2 Robot – PLC	74

10.2.1 Robot	74
10.2.2 PLC.....	75
10.3 PC visión – PLC	75
11 Pruebas y resultados.....	77
11.1 Reconocimiento de tablas	77
11.2 Efecto de la iluminación ambiente	82
11.3 Secuencia de órdenes y comunicaciones	83
11.4 Movimientos del robot.....	83
11.5 Descripción del funcionamiento de la estación	84
11.6 Precisión	89
11.7 Tiempo de ciclo	90
12 Conclusiones	91
13 Líneas futuras.....	92
14 Bibliografía	93

ÍNDICE DE ILUSTRACIONES

Ilustración 1. Tipos de iluminación (6).....	3
Ilustración 2. Diferencias debido a la iluminación (7).....	3
Ilustración 3. Ejemplo del funcionamiento de una lente (8).....	4
Ilustración 4. Imagen con diferentes kernel (9)	5
Ilustración 5. Dilatación y erosión (9)	5
Ilustración 6. Sistema típico de visión artificial (10).....	6
Ilustración 7. Robot humanoide Nao (15).....	7
Ilustración 8. Robot soldador (16)	7
Ilustración 9. Robot manipulador típico (19).....	8
Ilustración 10. Morfologías robots manipuladores. a) cartesiano, b) scara, c) esférico, d) cilíndrico, e) antropomórfico, f) paralelo (18).....	8
Ilustración 11. Representación esquemática de los problemas cinemáticos (20)	9
Ilustración 12. Tecnología de visión robótica de Fanuc (22).....	10
Ilustración 13. Palé.....	11
Ilustración 14. Ranuradora.....	12
Ilustración 15. ABB IRB-1200 5/0.9 (24)	20
Ilustración 16. IRC5 Compact (25).....	21
Ilustración 17. FlexPendant (28).....	21
Ilustración 18. Effi-Ring (30)	22
Ilustración 19. Colocación de las tablas con Effi-Ring.....	22
Ilustración 20. Effi-flex (29)	23
Ilustración 21. Cámara iDS UI-5200SE Rev.4 (31)	23
Ilustración 22. Cálculo de objetivo (31)	24
Ilustración 23. Plato de calibración.....	25
Ilustración 24. Garra de vacío Kenos KVG200.60. N213.CVL.S2 (36)	26
Ilustración 25. Electroválvula y bobina	26
Ilustración 26. Sensor de distancia IFM O1D100 (39).....	27
Ilustración 27. Garra inicial	28
Ilustración 28. Garra tipo caja.....	29
Ilustración 29. Garra definitiva	30
Ilustración 31. Unión garra de vacío	30
Ilustración 30. Brida.....	30
Ilustración 32. Unión cámara y sensor.....	31
Ilustración 33. PLC	32
Ilustración 34. Periferia distribuida.....	32
Ilustración 35. Macromate SNMP hub	33
Ilustración 36. D-IINK DES-1005D (49).....	33
Ilustración 37. Mesa de trabajo	34

Ilustración 38. Soporte de iluminación	34
Ilustración 39. Botonera.....	35
Ilustración 40. Conexiones.....	36
Ilustración 41. Conexiones internas del robot.....	37
Ilustración 42. Roscado de la garra del robot	38
Ilustración 43. Casquillos y garra del robot	38
Ilustración 44. Garra del robot montada	38
Ilustración 45. Estación.....	39
Ilustración 46. Vista de redes	41
Ilustración 47. Datos de posiciones de un robot	53
Ilustración 48. Estación virtual	54
Ilustración 49. Diferencias entre Offs (centro) y Reltool (derecha) partiendo desde el mismo punto (izquierda) (56).....	58
Ilustración 50. Calibración herramienta cámara	59
Ilustración 51. Calibración herramienta sensor	59
Ilustración 52. Calibración herramienta garra de vacío	59
Ilustración 53. Calibración ejes herramientas	60
Ilustración 54. Tablas sobre fondo blanco	62
Ilustración 55. Tablas sobre fondo negro.....	62
Ilustración 56. Imagen de pruebas	63
Ilustración 57. Tablas Canny	65
Ilustración 58. Tablas Deriche	66
Ilustración 59. Tablas regiongrowing	67
Ilustración 60. Cálculo relación pixel/mm.....	72
Ilustración 61. Tablas sobre plástico negro.....	77
Ilustración 62. Reconocimiento de tablas sobre plástico negro	78
Ilustración 63. Reconocimiento del palé sobre plástico negro	78
Ilustración 64. Tablas sobre melanina negra.....	79
Ilustración 65. Reconocimiento de tablas sobre melanina negra	79
Ilustración 66. Tablas sobre tela negra	80
Ilustración 67. Reconocimiento de tablas 240x90 sobre tela negra.....	80
Ilustración 68. Reconocimiento tablas 310x170 sobre tela negra.....	81
Ilustración 69. Reconocimiento tablas 170x90 sobre tela negra.....	81
Ilustración 70. Reconocimiento palé sobre tela negra	82
Ilustración 71. Reconocimiento tablas con diferentes orientaciones	82
Ilustración 72. Tablas observadas sin iluminación y sin filtro.....	83
Ilustración 73. Tablas observadas sin iluminación y con filtro	83
Ilustración 74. Robot en posición inicial	85
Ilustración 75. Cámara sobre palé.....	86
Ilustración 76. Sensor sobre palé	86
Ilustración 77. Detección del palé.....	86
Ilustración 78. Detección de tablas	87

Ilustración 79. Sensor sobre tabla	87
Ilustración 80. Garra de vacío sobre tabla	88
Ilustración 81. Robot moviendo tabla	88
Ilustración 82. Ubicación de las tablas despaletizadas	89
Ilustración 83. Estación enjaulada	92

ÍNDICE DE DIAGRAMAS

Diagrama 1. Jerarquía	14
Diagrama 2. Posicionamiento	15
Diagrama 3. Comunicaciones	16
Diagrama 4. Alimentaciones.....	17
Diagrama 5. Funcionamiento general	18
Diagrama 6. Main PLC	43
Diagrama 7. Modo_funcionamiento PLC.....	44
Diagrama 8. F_Robot PLC (1).....	46
Diagrama 9. F_Robot PLC (2).....	47
Diagrama 10. F_Vision_artificial PLC	49
Diagrama 11. Main Robot.....	56
Diagrama 12. Visión artificial PC (1).....	69
Diagrama 13. Visión artificial PC (2)	70

ÍNDICE DE ECUACIONES

Ecuación 1. Kernel Sobel horizontal	63
Ecuación 2. Kernel Sobel vertical.....	63
Ecuación 3. Valor gradiente Sobel.....	63
Ecuación 4. Dirección gradiente Sobel	64
Ecuación 5. IIR	65
Ecuación 6. Deriche izquierda-derecha búsqueda	65
Ecuación 7. Deriche derecha-izquierda búsqueda	65
Ecuación 8. Deriche resultado horizontal	65

ÍNDICE DE TABLAS

Tabla 1. Coeficientes Deriche66

1 INTRODUCCIÓN

Este trabajo de fin de máster de ingeniería industrial fue desarrollado durante las prácticas universitarias realizadas en la empresa Financiera Maderera S.A., conocida como Finsa (1), en su fábrica de Santiago de Compostela, bajo la supervisión de la oficina técnica.

Finsa es una empresa dedicada a la transformación de la madera. Se creó en 1931 como un aserradero en Portanxil (Ames), siendo hoy en día una de las empresas líderes en el sector de la transformación de la madera en Europa, con más de 3000 trabajadores y unos ingresos superiores a 800.000.000 € en 2017. Su principal actividad actual es la fabricación de tableros de madera aglomerada a partir de madera reciclada. Sin embargo, siguen manteniendo un importante nivel de ventas de productos fabricados en el aserradero a partir de madera de pino.

Los procesos en el aserradero requieren una elevada carga de revisión visual. Esto es debido a la forma irregular de los troncos y a los nudos y las vetas en los mismos, los cuales no son deseables en la mayoría de los productos finales. Además, los productos y subproductos con los que se trabajan tienen un alto grado de variabilidad. Esto dificulta la automatización de los procesos.

Para solucionarlo y modernizar el aserradero, han desarrollado proyectos utilizando tecnologías de la llamada industria 4.0. Una de las tecnologías utilizada ha sido la visión artificial, obteniendo resultados positivos.

Por ello, han decidido desarrollar un nuevo proyecto de visión robótica, consistente en otorgarle visión artificial a un brazo robótico, con el fin de que pueda reconocer la posición y orientación de una determinada referencia, pudiendo así manipularla.

1.1 Objetivos

El objetivo principal de este trabajo es el desarrollo de un proyecto piloto de una estación de despaletizado de tablas de madera mediante visión robótica. Para alcanzarlo, es necesario lograr los siguientes objetivos:

- Estudiar el entorno.
- Diseñar la arquitectura de la estación.
- Diseñar el funcionamiento de la estación.
- Elegir las herramientas.
- Diseñar y fabricar la herramienta del robot.
- Programar la unidad de control de la estación.
- Programar las trayectorias para el despaletizado.
- Programar el sistema de visión artificial para el reconocimiento de la pose de las tablas.
- Analizar la precisión de la aplicación.
- Analizar los tiempos de ciclo de la estación.

1.2 Antecedentes

Las exigencias actuales del mercado se acompañan de nuevas necesidades en las industrias. Los clientes demandan productos personalizados, lo que se traduce en la necesidad de modos de fabricaciones flexibles, capaces de adaptarse a la alta variabilidad de los procesos y productos, sin perder el alto nivel de automatización alcanzado.

Acompañando a estas necesidades, han surgido diferentes tecnologías innovadoras capaces de cubrir las necesidades comentadas anteriormente. Estos avances permiten posicionar a nuestro mundo en los prolegómenos de lo que se conoce como la 4ª revolución industrial (2).

Una de estas nuevas tecnologías utilizada en este camino hacia las llamadas fábricas inteligentes es la visión robótica. El objetivo de esta nueva tecnología es conseguir que los robots tengan un mayor conocimiento del medio en el que se encuentran, incrementando su capacidad de decisión. Para lograrlo, se equipa al robot con un sistema de visión artificial. Esto permite a la robótica un nuevo horizonte de tareas posibles, como la navegación, la asistencia doméstica o el reconocimiento y manipulación de objetos, entre otras aplicaciones (3) (4).

1.2.1 Visión artificial

Según la Automated Imaging Association (AIA) (5), la visión artificial es la extracción automática de información de imágenes digitales. Esto es posible gracias a una combinación de hardware y software que permite, sin contacto, analizar de forma automática el entorno.

A pesar de ser considerada la visión humana como la mejor para la interpretación de una escena, la visión artificial destaca en su elevada repetibilidad, velocidad y precisión en el análisis cuantitativo del entorno. Esta tecnología comenzó a desarrollarse en la década de 1950, y en 1980 empezó su uso industrial. Las principales aplicaciones industriales de los sistemas equipados con esta tecnología son:

- Medición: calculan distancias entre puntos, ubicaciones u objetos en las imágenes.
- Contaje: cuentan el número de veces que encuentran un determinado patrón en la escena.
- Localización: obtienen la ubicación de un determinado objeto.
- Identificación: capaces de leer códigos de barras, códigos QR, marcajes, etc., reconociendo así el objeto.

La combinación de estas aplicaciones permite desarrollar sistemas muy robustos. Los principales objetivos de estos sistemas son:

- Reducir defectos, localizándolos y midiendo su tamaño.
- Incrementar el rendimiento, detectando errores en la cadena de producción y solucionándolos lo antes posible.
- Conseguir trazabilidad, utilizando sistemas de identificación.
- Cumplir con las regulaciones al aumentando el nivel de control sobre el producto.

A nivel industrial, todos los sistemas de visión artificial constan de los elementos observables en la Ilustración 6, explicados a continuación:

- Iluminación. Es un elemento crítico, ya que de él depende en gran medida la información extraíble de la imagen. Como se puede observar en la Ilustración 2, una incorrecta iluminación complica la extracción de información de la imagen. Por ello, como se muestra en la Ilustración 1, existen diferentes métodos para iluminar un objeto, con el fin de conseguir imágenes que resalten las zonas o los elementos de esta que se necesitan en función de cada aplicación. Diseñando bien la iluminación se obtienen resultados mejores con un software de visión más sencillo. A mayores, con el fin de conseguir soluciones robustas, la iluminación recibida por el sistema de visión debe ser constante. Por ello, es necesario eliminar la luz ambiente. La mejor solución es encapsular lo que se desea observar con el fin de que solo incidan sobre ellos la luz procedente del sistema de iluminación. Pero a veces, debido a la casuística de cada aplicación, esto no es posible, por lo que se deben colocar filtros a la entrada del sensor de visión, para permitir únicamente el paso de la luz procedente de iluminación. Existen diferentes tipos de filtro, siendo algunos de los más usados el filtro por color (frecuencia de la luz) y el filtro por polarización (plano de oscilación).

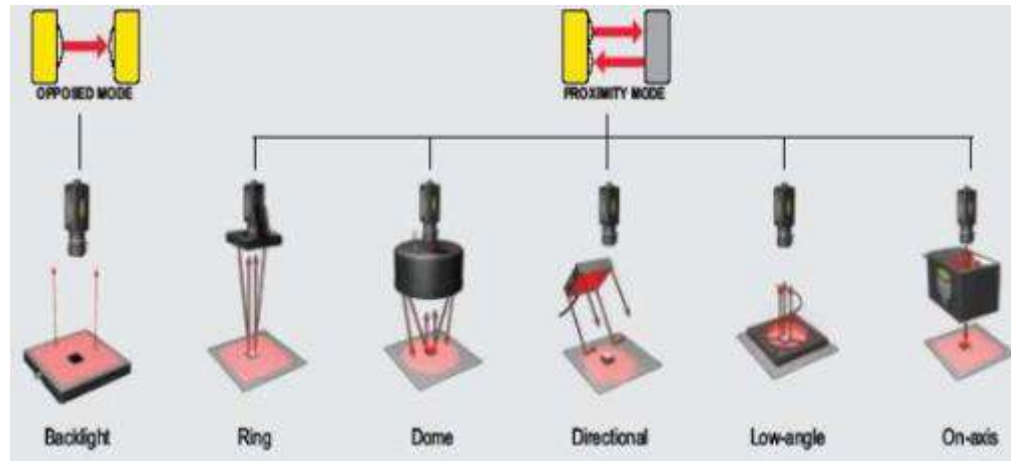


Ilustración 1. Tipos de iluminación (6)



Ilustración 2. Diferencias debido a la iluminación (7)

- Lente. Este elemento determina en gran medida la calidad, la resolución y las dimensiones de la imagen. Es una lente o un grupo de lentes que se encarga de realizar el enfoque y ajustar el tamaño de la imagen real al tamaño del sensor de imagen. Para decidir qué lente se necesita, se debe tener en cuenta su resolución, que debe ser igual o mayor que la del sensor de imagen, y su distancia focal. La distancia focal se determina en función de la superficie que se necesita observar y la distancia entre ella y el sensor de imagen. Su cálculo se hace con las ecuaciones de la óptica geométrica. Un ejemplo de cómo funcionan se puede apreciar en la Ilustración 3.

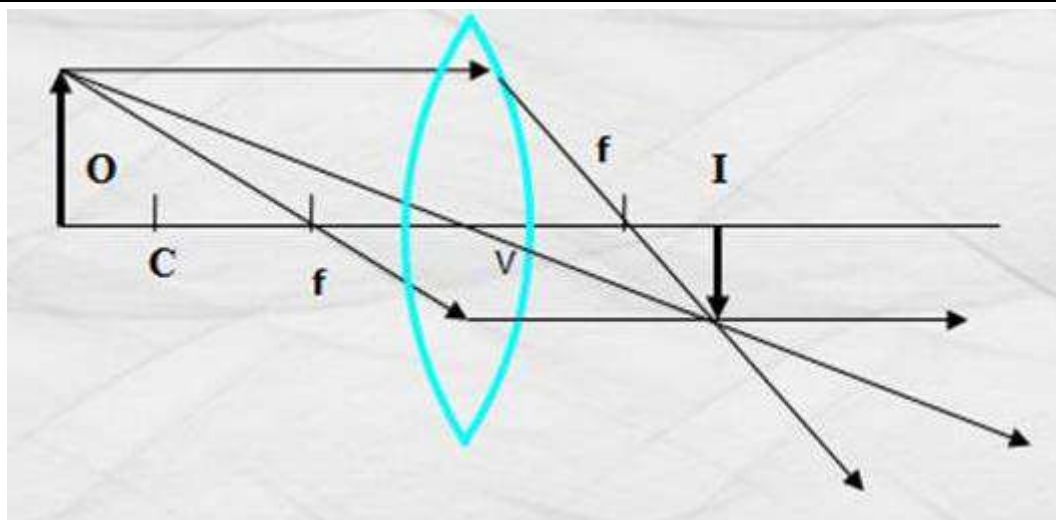


Ilustración 3. Ejemplo del funcionamiento de una lente (8)

- Sensor de imagen. Es la parte de la cámara que se encarga de transformar los fotones en señales eléctricas. Las características que se deben tener en cuenta en estos sensores es su resolución y el tipo de información que extraen de la realidad, pudiendo ser monocromáticas, a color, 2D o 3D. La resolución en un sensor de imagen viene determinada por el número de columnas de píxeles que lo conforman, el número de filas y el número de bits que tiene cada píxel. Un píxel es la menor unidad homogénea en una imagen digital, por lo que cuantos más píxeles tenga una imagen, menor tamaño de la imagen real representa, y cuantos más bits tenga cada píxel, más puede asemejarse a la realidad en tanto a color y nivel de gris.
- Procesado de imagen. En este elemento se lleva a cabo la extracción de la imagen digital mediante el uso de un software. La metodología a seguir suele ser la siguiente:
 - Tratamiento de imagen. Se hace de resaltar las zonas de interés. Cuanto mejor sea la iluminación, menor será el trabajo necesario en el procesamiento de imagen. Existen diferentes tipos de operaciones. Unos de los más utilizados son:
 - Operaciones sobre el histograma de grises. Operan sobre los píxeles de la imagen. Una de las más utilizadas es la binarización, convirtiendo los píxeles de la imagen a blanco o a negro en función de si superan o no un valor de consigna de gris.
 - Convolución. Operan sobre regiones que rodean a un píxel, modificando su valor en función del suyo propio y de los que le rodean. En función de la ponderación de los elementos se obtienen resultados diferentes. En la Ilustración 4 se puede ver cómo afecta la variación del *kernel* aplicado a la imagen. El *kernel* es la matriz que determina el valor de cada píxel de la imagen. Se coloca el centro de la matriz sobre un píxel y se calcula el valor de dicho punto como la media ponderada de todos los puntos bajo la matriz, siendo los coeficientes de ponderación de cada punto el valor de la matriz que coincide sobre cada uno.

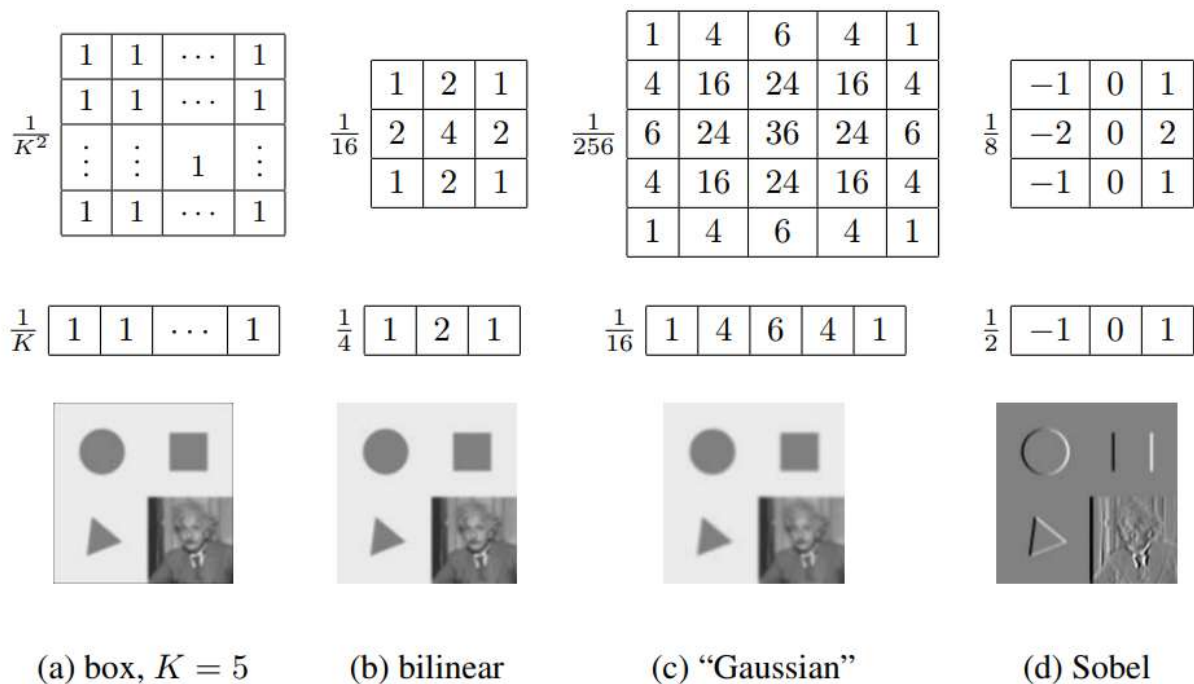


Ilustración 4. Imagen con diferentes kernel (9)

- No lineales. A diferencia de los anteriores, no se basan en una operación aritmética. Los más utilizados son el de erosión y el de dilatación y se aplican sobre imágenes binarias formadas por píxeles pertenecientes a un objeto y píxeles pertenecientes al fondo. La erosión añade al fondo píxeles del objeto que distan menos de cierto valor de píxeles del fondo. La dilatación es la operación análoga. Sus efectos se pueden ver en la Ilustración 5, siendo la primera letra la imagen original, y la segunda y la tercera tras aplicarle dilatación y erosión respectivamente.



Ilustración 5. Dilatación y erosión (9)

- Segmentación. Consisten en agrupar los píxeles que forman parte de objetos o zonas de la imagen que interesan de las que no. Existen diferentes algoritmos, pero todas ellas buscan características en cuanto a nivel de gris, regiones conexas, color, vértices, etcétera, que se distingan de las demás zonas de la imagen. Estos algoritmos suelen aplicar filtros a la imagen, utilizando uno u otro *kernel* en función de qué característica se busque.
- Etiquetado. Segmentada la imagen, los píxeles que pertenecen a una misma zona u objeto se agrupan. Esto puede ser más o menos complicado en función de la

- aplicación, ya que estas zonas pueden estar claramente diferenciadas o pueden llegar a estar superpuestas, complicando así el algoritmo necesario.
- Clasificación. Con la imagen segmentada y los elementos de interés etiquetados, estos se clasifican. Para ello, se comparan con un modelo calculando una serie de propiedades típicamente geométricas.
- Comunicación. La información obtenida se comunica a un nivel superior para realizar la toma de decisiones.

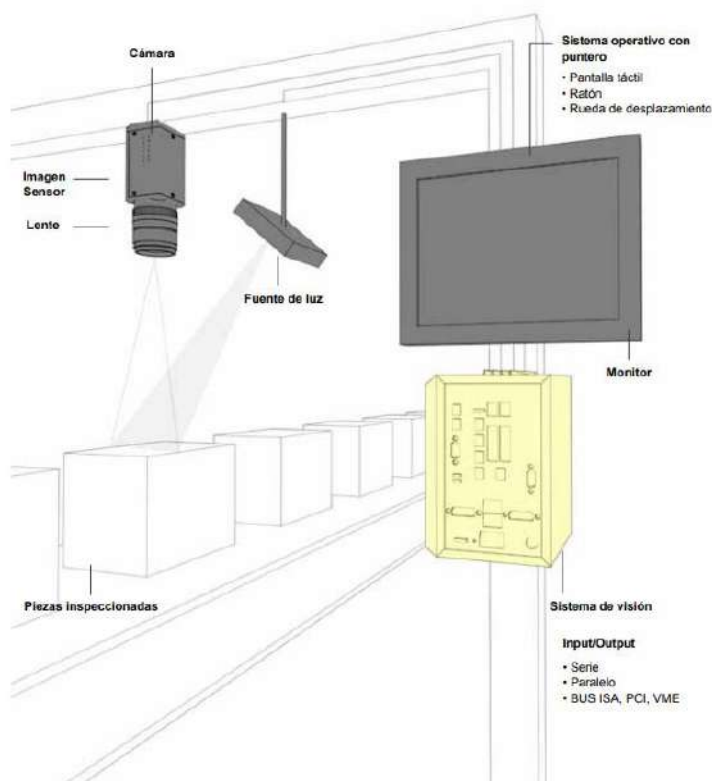


Ilustración 6. Sistema típico de visión artificial (10)

(5) (10) (9)

1.2.2 Robótica

El término robot aparece por primera vez en la obra del escritor Karel Čapek (1890-1938 R.U.R. (Robots Universales Rossum) refiriéndose a unos autómatas que trabajaban como obreros. Su origen proviene de la palabra checa «robot» cuyo significado es trabajo (11).

La robótica estudia el diseño, construcción, operación, estructura, manufactura y aplicación de los robots. En otras palabras: la ciencia que estudia a los robots. Una de las primeras referencias a un automatismo dirigido a ahorrar esfuerzo aparece en la obra *Lie Zi* atribuida a Lie Yukou en el siglo III a.C. (12). A lo largo de la historia, se han ido creando nuevos autómatas con diseños muy diferentes, principalmente basados en tecnología hidráulica y mecánica. En 1801, Joseph-Marie Charles Jacquard patenta en Francia un telar programable mediante tarjetas perforadas (13). Sin embargo, no es hasta la década de 1970, que gracias a la aparición de los microprocesadores y a los avances en la informática, aparecen los primeros robots industriales. La ventaja de estas herramientas es su capacidad de realizar una gran variedad de tareas automáticas gracias a la coordinación entre un procesador reprogramable y un mecanismo.

Existen diferentes maneras de clasificar la gran variedad de robots existentes. Hay robots que imitan la fisonomía humana, como el de la Ilustración 7, capaces de interactuar con las

personas. Otros se desarrollan con fines terapéuticos, como es el caso de Nuka, un robot con forma de foca que es capaz de conseguir que las personas establezcan vínculos afectos con ellas, reduciendo problemas de depresión (14).

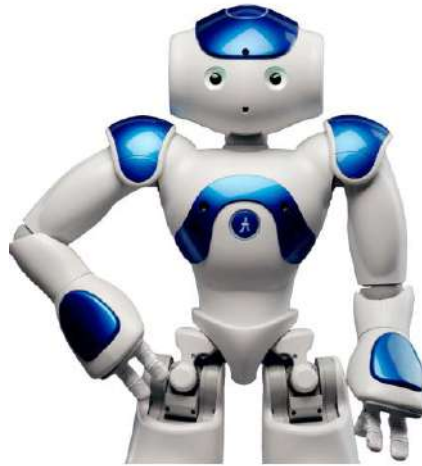


Ilustración 7. Robot humanoide Nao (15)

Uno de los más usados en la industria es el robot manipulador. Sus aplicaciones van desde el de manipular una herramienta realizando trabajos de soldadura como se ve en la Ilustración 8, que fue la aplicación más utilizada durante muchos años, siendo superada por la de manipulación.



Ilustración 8. Robot soldador (16)

Su uso en la industria sigue en aumento desde su aparición, siendo cada vez más precisos y seguros, llegando a realizar tareas de forma colaborativa con humanos. En España hay cerca de 35.000 robots, siendo muy utilizados en fabricación en serie en el sector del automóvil (17).

Están compuestos por:

- Manipulador. Es La estructura mecánica del robot. El diseño del manipulador suele imitar al brazo humano, como se aprecia en la Ilustración 9, pero existen muchas otras morfologías. Algunas de ellas se pueden ver en la Ilustración 10. Es capaz de mover objetos siguiendo trayectorias definidas o realizar diferentes trabajos a lo largo de ellas, en función de la herramienta acoplada al robot. Las articulaciones del mecanismo determinan sus grados de libertad, siendo 6 los necesarios para alcanzar todas las orientaciones posibles de un punto en el espacio: 3 de posición y 3 de

orientación. Este conjunto de variables de posición y orientación del robot en el espacio se conoce como pose (18).



Ilustración 9. Robot manipulador típico (19)

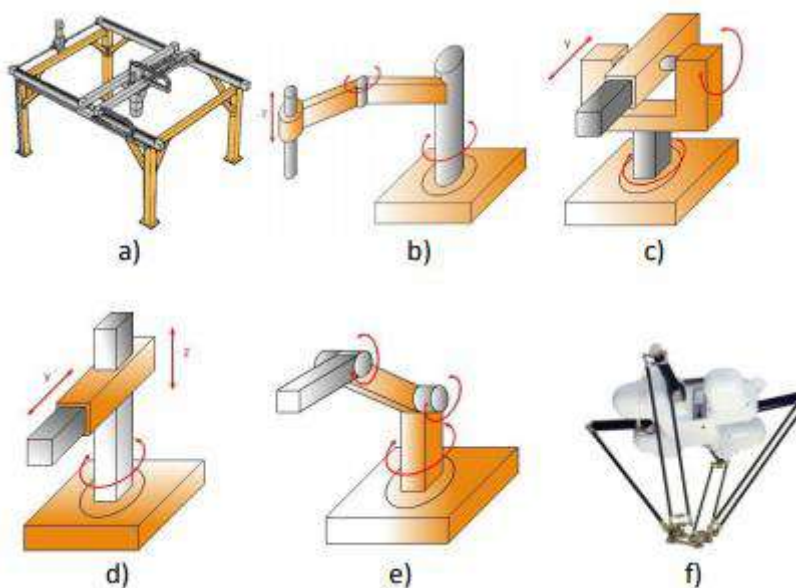


Ilustración 10. Morfologías robots manipuladores. a) cartesiano, b) scara, c) esférico, d) cilíndrico, e) antropomórfico, f) paralelo (18)

- **Sensores.** Informan a la unidad de control. Pueden ser internos o externos. Los primeros proporcionan información del estado del robot, necesarios para controlar los movimientos de este, y los segundos del entorno. Juegan un papel importante en el nivel de automatismo del robot.
- **Unidad de control o controladora.** Es el cerebro del robot. Se encarga de la lectura del programa informático y de determinar las alimentaciones de los motores del manipulador. El cálculo de la pose que debe adoptar el robot se hace mediante el uso del álgebra matricial y el razonamiento geométrico. Como se observa en la

Ilustración 11, la controladora del robot debe establecer las relaciones entre las posiciones de las articulaciones del robot con su pose en cada momento, para lograr así la trayectoria deseada (20). Para facilitar la programación de las tareas, han ido surgiendo lenguajes de programación específicos para facilitar su desarrollo. Existen diferentes niveles de lenguaje, yendo desde las especificaciones de los pasos de los motores, a la programación por trayectorias, estableciendo el propio lenguaje las relaciones entre los ángulos de las articulaciones y la pose del robot. Incluso se está trabajando en el desarrollo de un lenguaje en el que se especifiquen únicamente tareas, y que sea el propio robot el que planifique las trayectorias.

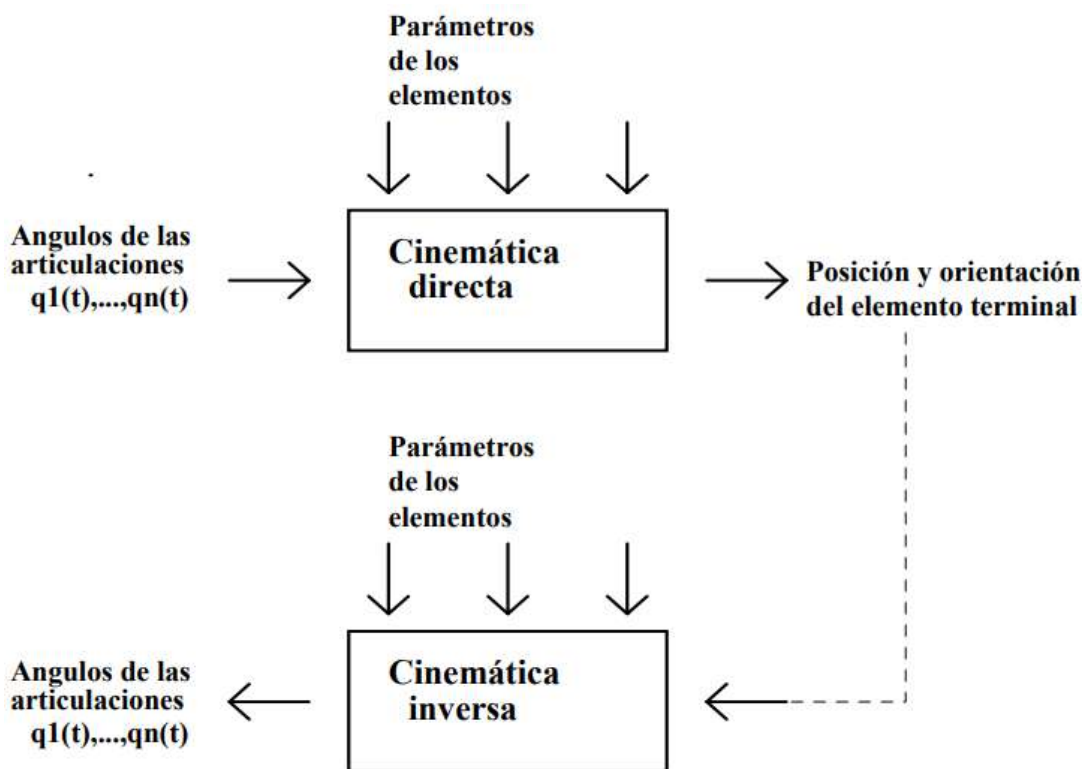


Ilustración 11. Representación esquemática de los problemas cinemáticos (20)

- Unidad de conversión de potencia. Proporciona la energía al robot.

1.2.3 Visión robótica

El uso de sensores en los robots les permite tener un mayor conocimiento del entorno, aumentando así su capacidad para desarrollar tareas más complejas. La visión robótica se basa en el acoplamiento de un sensor capaz de otorgarle visión artificial a un robot, siguiendo así con la línea de desarrollo actual que intenta imitar el funcionamiento de los humanos en las máquinas. Esta tecnología es posible gracias a los avances en visión artificial, en computación y en robótica. La comunicación entre el sistema de visión y el robot permite la realización de tareas en las que la dinámica del robot dependa de la posición de objetos cuya posición sea variable, incluso de la toma de decisión sobre qué tarea realizar en función de la información recibida del medio.

Una gran variedad de robots utilizan sistemas de visión artificial, como los humanoides que se mencionaron anteriormente o los sistemas de vehículos no tripulados.

En el mundo industrial, su uso principal es el de manipulación. Mediante visión artificial, se extrae información de posición y orientación sobre los objetos que el robot debe manipular. Esta información se le comunica a la controladora del robot. En base a dicha información, calcula las

actuaciones necesarias para que el robot se desplace hasta el objeto y realice la tarea. Se usa en tareas de despaletizado, como el de la Ilustración 12, de control de calidad o de clasificación de objetos (21).



Ilustración 12. Tecnología de visión robótica de Fanuc (22)

2 ANÁLISIS EN CAMPO

El primer paso en el desarrollo de un proyecto debe ser la recopilación de información que afecte de forma directa o indirecta al mismo. Esto permite hacer un diseño de la solución más optimizado, pudiendo tener en cuenta factores determinantes desde el primer momento. Por ello, se hizo un estudio de los elementos que influyen en la tarea del despaletizado de piezas para la alimentación de la ranuradora que se pretende automatizar.

La tarea se realiza en el aserradero seco de Finsa Santiago. Es una nave de 10 m de alto con iluminación desde el techo con tubos fluorescentes. Debido a la casuística de los trabajos en el aserradero, es una zona poco automatizada, con un gran movimiento de operarios.

Los palés tienen una medida fija de 1520 mm de largo, 1140 mm de ancho y 1400 mm de alto. Sin embargo, las medidas de las tablas no tienen una medida fija. Todas son de madera de pino seca, de baja porosidad, con un acabado superficial muy liso y con una densidad de 500 kg/m³. Tienen forma rectangular, con un espesor constante de 10 mm, un largo de entre 240 mm y 360 mm, y un ancho de entre 90 mm y 300 mm. Además, todas las piezas del mismo palé tienen las mismas medidas, pero por optimización de espacio, no todos los niveles de piezas del palé tienen la misma distribución. Un ejemplo de uno de estos palés se puede ver en la Ilustración 13.



Ilustración 13. Palé

Actualmente, el trabajo lo realizan operarios manualmente. Se posiciona el palé sobre el suelo, a 1 metro del punto en el que se depositan las piezas, visible en la Ilustración 14. La ranuradora permanece encendida durante todo el proceso y se va alimentando en continuo. La cadencia de trabajo es de 0'8 piezas/s, con un tiempo de trabajo medio diario es de 4 horas.



Ilustración 14. Ranuradora

Esta información permitió sentar las bases de las líneas de desarrollo a seguir para alcanzar una solución óptima. Debido a que las piezas tienen una geometría y posición sobre el palé variables, es necesario una herramienta capaz de localizarlas y otra que tenga la suficiente versatilidad como para poder cogerlas. Además, para el diseño del sistema de agarre se tuvo en cuenta que las piezas tienen una baja porosidad y un acabado liso.

3 ARQUITECTURA Y FUNCIONAMIENTO

Conocidas las necesidades de la estación de trabajo y su entorno, se diseñó su arquitectura. Este paso, conjunto con la búsqueda de herramientas, facilitó su elección, ya que permitió conocer más concretamente sus características necesarias.

En base a las conclusiones extraídas del análisis en campo, se optó por un sistema de Visión robótica. De esta forma, se reconocen las posiciones y orientaciones de las piezas utilizando un sistema de visión que, comunicándose con un robot, puedan ser agarradas correctamente y posicionadas en el lugar necesario gracias a la versatilidad de movimientos de un sistema de estas características.

Para el sistema de agarre, teniendo en cuenta el tipo de piezas, se optó por un sistema de agarre por vacío. Esto implica diseñar un circuito neumático que pueda activar y desactivar el efecto de succión cuando sea necesario.

Además, se decidió usar tecnología 2D junto con un sensor de profundidad. Las piezas se despaletizan por niveles horizontales para mantener la estructura del palé estable, por lo que, conociendo la altura de un punto del nivel, se conoce la altura de todas las piezas de ese mismo nivel. Tratando cada nivel de forma independiente, puede solucionarse con tecnología 2D. De esta forma se reduce sustancialmente el presupuesto del proyecto.

En el apartado de Elección y diseño de herramientas se concretan los motivos por los cuales se decidió usar cada una de ellas y cuales se probaron y se descartaron.

3.1 Jerarquía

En esta fábrica, y en la gran mayoría de ellas, los procesos están gobernados por controladores lógicos programables, conocidos como PLCs. Un PLC es una computadora utilizada para automatizar procesos electromecánicos, diseñados para trabajar en tiempo real y con un gran número de variables de entrada y de salida, permitiendo conectarse a una red mediante buses de datos, pudiendo ser observados e incluso gobernados de forma remota (23). Por este motivo, se decidió utilizar un PLC como elemento de gobierno de la estación. Este ejecuta un programa que controlase el sistema de visión, el del robot, el de agarre por vacío y el sensor de distancia.

A su vez, el sistema de visión está gobernado por un PC que ejecuta el programa de visión artificial, comunicándose con la cámara. Del mismo modo, la controladora del robot es la encargada de ejecutar el programa de trayectorias gobernando los movimientos del manipulador.

En el Diagrama 1 se puede apreciar la jerarquía de la aplicación a nivel conceptual.

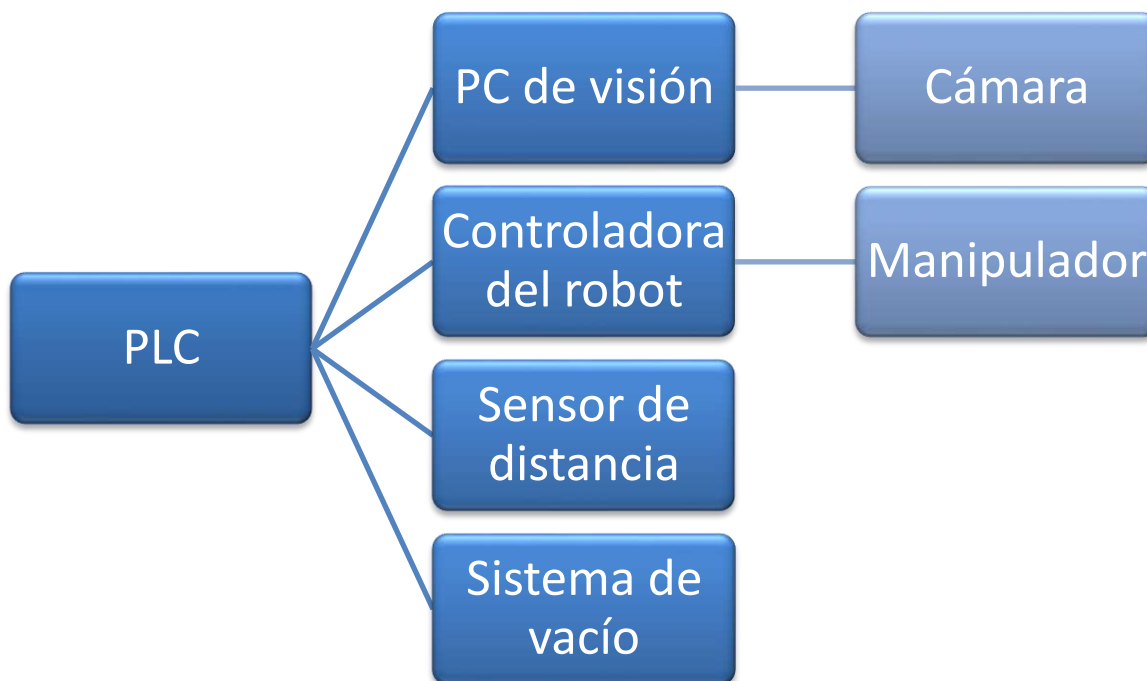


Diagrama 1. Jerarquía

3.2 Posicionamiento

El posicionamiento de las herramientas es un factor determinante, ya que una buena distribución simplifica enormemente la aplicación. Por ello, antes de programar y de elegir las herramientas, se barajaron las posibles opciones de colocación para optimizar la estación.

Primeramente, se hizo una clasificación de las herramientas necesarias en función de su aplicación.

Las herramientas encargadas de la ejecución del programa se sitúan en un entorno protegido, fuera del lugar del trabajo, ya que no necesitan estar cerca del lugar de trabajo.

El sistema de agarre por vacío se coloca como herramienta del robot. El robot se sitúa en un lugar en el que pueda posicionar la herramienta de agarre sobre todas las piezas y dejarlas en el punto de descarga. El sistema de agarre se conecta a un circuito neumático controlado con una electroválvula que se sitúa fuera del lugar de trabajo por el mismo motivo que los procesadores.

Por otra parte, la cámara y el sensor de profundidad podrían colocarse también como herramientas del robot. Si no se colocasen como una herramienta del robot, podrían tener un soporte fijo o móvil. Si fuese fijo, habría que colocar el palé dentro de los rangos de actuación de ambos sensores y siempre a la misma distancia de la cámara por problemas de enfoque, o estar equipada de una óptica de autoenfoque y hacer calibraciones cada vez que cambiase la altura, para poder conocer las posiciones de las piezas, ya que debido al efecto perspectiva, las distancias vistas en una imagen se ven afectadas por la distancia a la que se encuentran del punto de visión. Por esta razón, en sistemas de visión artificial en los que se pretende calcular intervalos espaciales, el sensor de imagen se encuentra a la misma distancia de las piezas y necesariamente debe ser conocida, haciendo previamente una calibración.

Otra opción sería hacer una estructura móvil, siendo a la aplicación más versátil que con la solución de la estructura fija, pero esto implicaría un diseño de una estructura con actuadores comunicados con los demás sistemas, aumentando su complejidad.

Por ello, se decidió posicionar ambos sensores como herramientas del robot, conservando la versatilidad de tener una estructura móvil, pero sin incorporar elementos que dificulten el diseño.

Por último, la posición de los elementos de adaptación del entorno, que en este caso son los de iluminación necesaria para el sistema de visión artificial, también fue sujeta a discusión. Se barajó la posibilidad de que fuesen acoplados al robot, estando así colocados en la misma posición relativa a la cámara, y no teniendo que hacer ninguna estructura de sujeción independiente. Sin embargo, como se explicará en el apartado de Elección y diseño de herramientas, no fue posible. Se hicieron pruebas con sistemas que pudiesen colocarse como una herramienta del robot, pero no se conseguía la iluminación necesaria. Por lo tanto, se escogió otro tipo de iluminación que se posiciona sobre un soporte externo. En el Diagrama 2 se puede apreciar el posicionamiento de los elementos de la estación de forma esquemática.

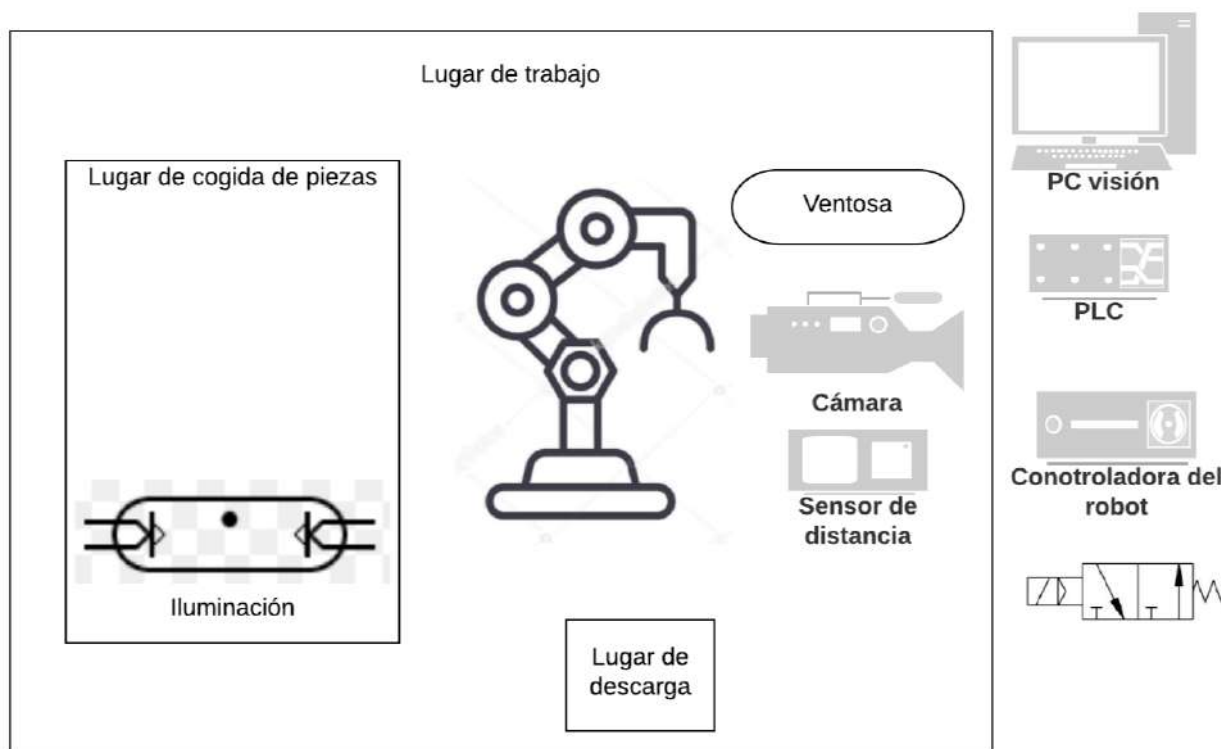


Diagrama 2. Posicionamiento

3.3 Comunicaciones

El PLC es el elemento que gobierna la estación. Por ello, los demás procesadores, que son la controladora del robot y el PC de visión, deben comunicarse con él. Estas comunicaciones se realizan mediante buses de campo, simplificando así su instalación. Un bus de campo es un sistema que permite la transmisión de datos cumpliendo la misma función que un gran número de bucles de corriente estándar. Además, de esta forma se puede conectar a la red de fábrica, pudiendo acceder a los dispositivos de forma remota. A mayores, el sensor de distancia y la electroválvula también se comunican con el PLC por tarjetas de entrada.

Por comodidad de diseño, se decidió que todos los cables de bus tuviesen RJ45 como interfaz física, pudiendo así conectarlos todos al mismo *switch* sin necesidad de adaptadores. Un *switch* es un dispositivo que permite el tránsito de información entre los cables conectados a él, repitiendo los mensajes que le llegan a los canales indicados.

En los robots industriales, la comunicación entre la controladora y el manipulador viene configurada y no son modificables por motivos de seguridad, por lo que es transparente a este proyecto. Sin embargo, la de la cámara con el PC de visión debe ser configurada. La cámara podría conectarse directamente al PC de visión, pero de esta forma se necesitaría un PC con un conector a mayores del usado para conectarse al *switch*, y no se podría acceder a la cámara de forma remota. Por este motivo, se decidió seleccionar una cámara que se comunicase mediante

RJ45, conectándola al *switch*. La distribución se puede ver en el Diagrama 3. En este punto únicamente se diseñaron los elementos físicos de las comunicaciones, no los protocolos utilizados. Estos se detallan más adelante, una vez seleccionadas las herramientas y los programas de utilizados.

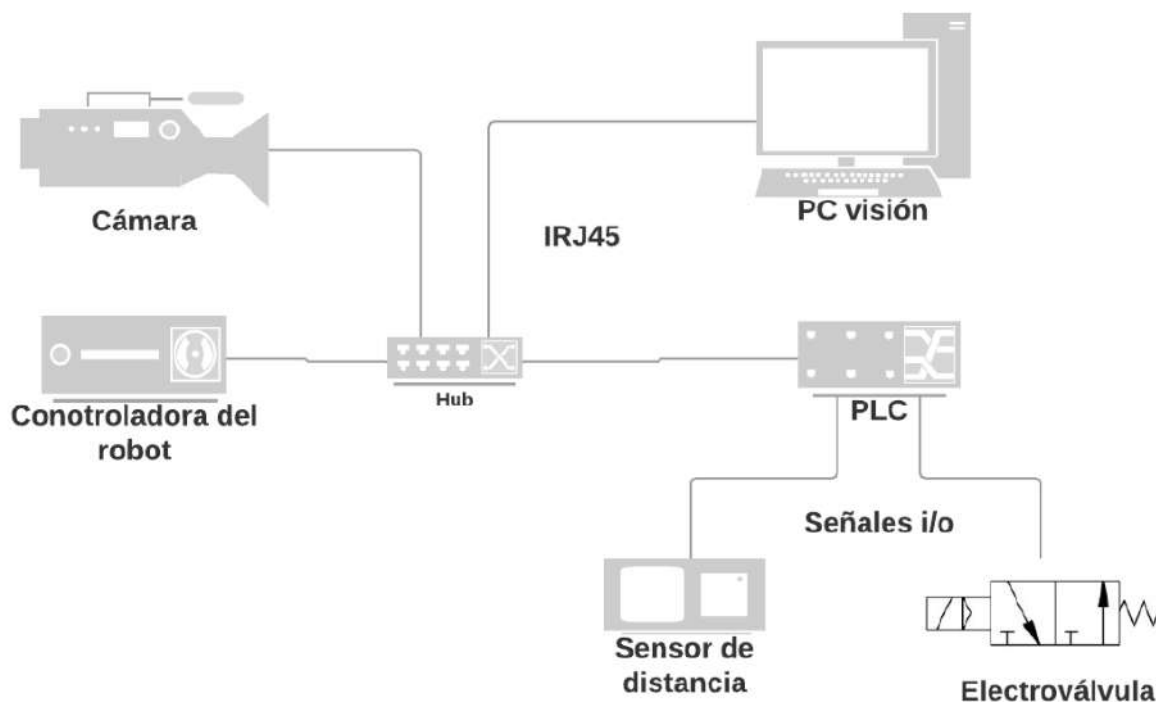


Diagrama 3. Comunicaciones

3.4 Alimentaciones

Las herramientas utilizadas necesitan 2 tipos de alimentación: eléctrica y neumática. La alimentación neumática se consigue a través de los puntos distribuidos en fábrica con una presión de 6 bar, y es necesaria para el sistema de vacío que permite la succión de las ventosas.

Por otra parte, las alimentaciones eléctricas utilizadas por el resto de los equipos vienen especificadas por los fabricantes. El PLC, el PC de visión y el *switch* necesitan corriente alterna de 230V, alimentación disponible en fábrica. El sensor de distancia, la iluminación y la cámara se alimentan a 24V de continua y se conectan a través de las alimentaciones disponibles en el PLC. El robot se alimenta con corriente alterna de 230V, pero desde una toma diferente por necesitar mayor potencia. En el Diagrama 4. Alimentaciones se puede ver dicha configuración.

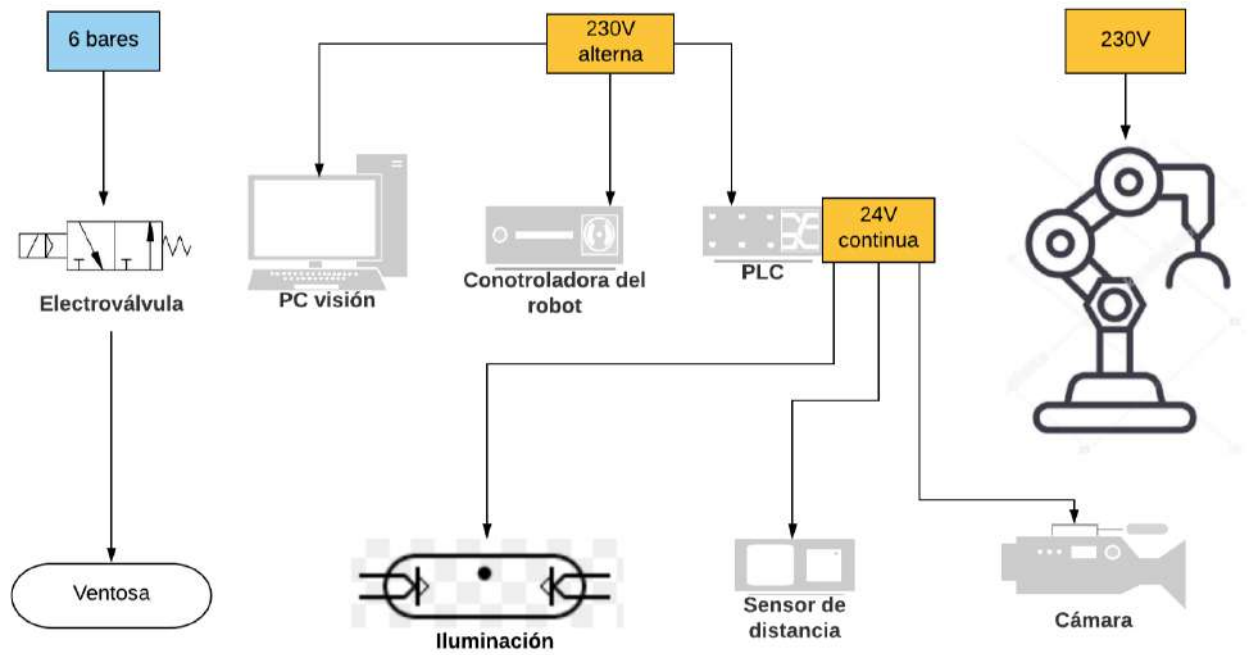


Diagrama 4. Alimentaciones

3.5 Funcionamiento general

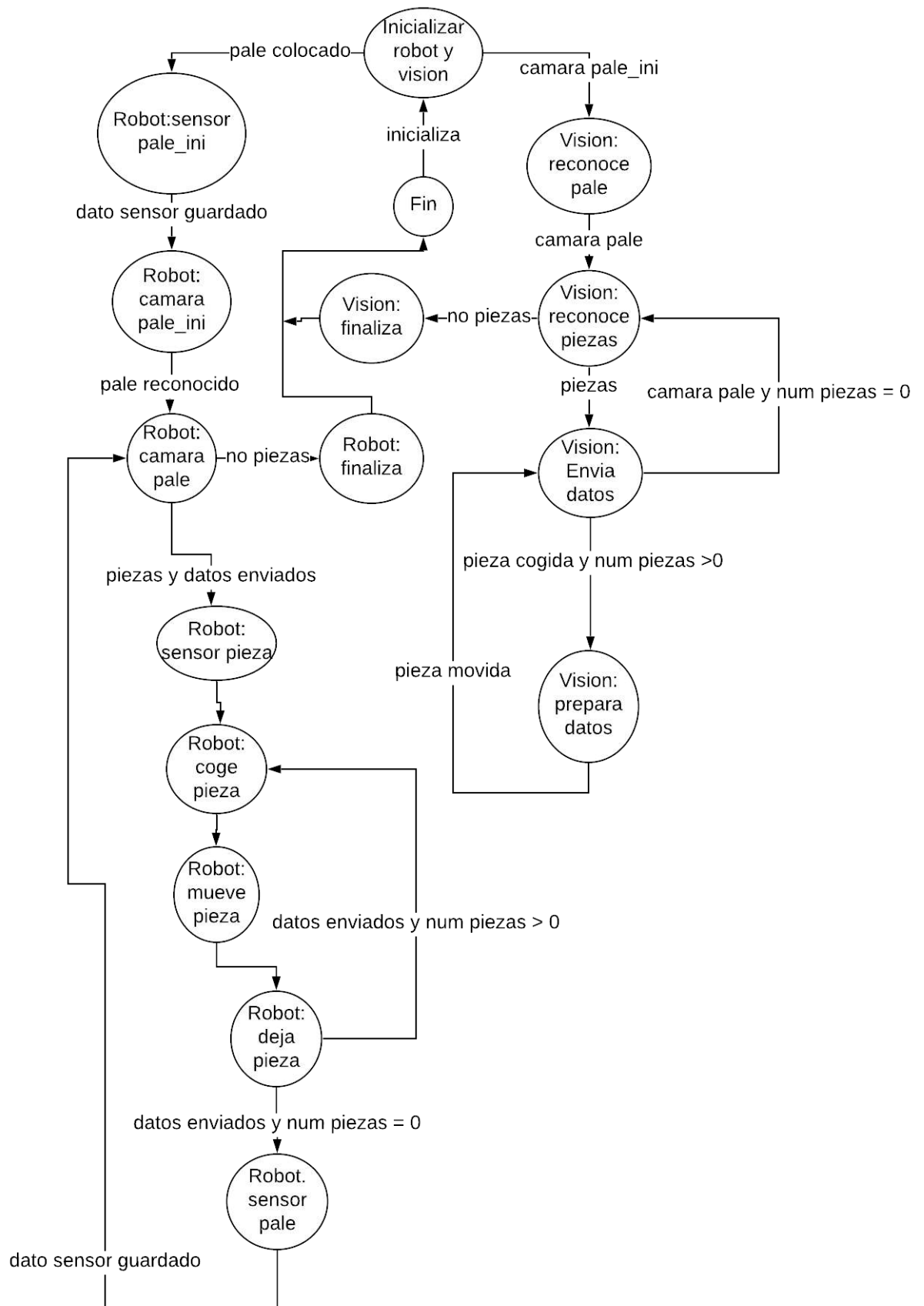


Diagrama 5. Funcionamiento general

Tras haber diseñado la arquitectura de la estación y en función de las necesidades de esta, se diseñó su proceso. Este se hizo tratando el sistema de visión y el del robot como sistemas independientes que se comunican entre sí.

En el Diagrama 5 se puede ver el proceso diseñado. Cuando deba iniciarse el proceso, se activa la orden de inicializar, preparando el sistema de visión y colocando el robot en una zona en la que no interfiera con las tareas del operario. Una vez inicializado el proceso, el palé se posiciona en su lugar en la estación, activándose la señal que indica dicho posicionamiento.

El robot tiene guardada una posición sobre el palé para poder llevar la cámara y el sensor a un punto desde el cual lo pueda observar. Sin embargo, no todos los palés tienen la misma altura. Al trabajar con una cámara 2D, es necesario que la distancia entre las piezas y la cámara en el momento de hacer la fotografía sea siempre la misma. Los sistemas de visión, al trabajar con imágenes digitales, miden las distancias a partir de píxeles. Un píxel es la menor unidad homogénea de color, en este caso de nivel de gris, de una imagen digital. Para poder hacer la conversión a las distancias reales, es necesario hacer una calibración, y esta depende de la distancia entre la cámara y el plano de la imagen. Esto implica que las magnitudes extraídas de una imagen solo serán las reales si el plano de la imagen coincide con el plano de calibración. Es por ello por lo que el robot posiciona el sensor en la posición sobre el palé. Se lee la distancia a la que se encuentra el palé y con esa información el robot posiciona la cámara sobre el palé a la altura para la que está calibrada.

Para otorgarle mayor robustez al sistema, no se reconocen directamente las piezas con la cámara posicionada sobre el palé, sino que primero se reconoce el palé con el fin de posicionar la cámara sobre su centro, realizando este procedimiento únicamente una vez por palé. El sistema de visión reconoce el palé, y tras enviarle la posición y orientación de este, el robot posiciona la cámara sobre su centro.

Seguidamente, el sistema de visión reconoce todas las piezas del primer nivel del palé y le envía los datos de la primera al robot. El robot posiciona el sensor sobre el centro de la pieza, recoge la información de la distancia a la que se encuentra, y la despaletiza con la garra de vacío. El despaletizado de cada pieza se divide en 3 fases: coger pieza, mover pieza y dejar pieza. Esto se hace por dos motivos: primero, porque la garra de vacío solamente debe estar succionando durante el proceso de mover la pieza, y segundo, para que el sistema de visión pueda saber si los datos de la pieza que envió ya fueron usados. Una vez terminado el procedimiento con la primera pieza, se repite el procedimiento hasta completar todas las piezas del nivel. Al terminar un nivel, el robot posiciona el sensor sobre el punto donde se posicionó la cámara para reconocer las piezas del nivel terminado. Se lee la distancia a la que se encuentra el palé, ya que al haber retirado piezas esta ha aumentado, y corrigiéndola, se posiciona la cámara de nuevo para volver a reconocer las piezas, repitiendo de nuevo el proceso para los siguientes niveles hasta completar el palé. Una vez extraídas las piezas del último nivel, el robot vuelve a posicionar el sensor sobre él, y corrigiendo la altura, posiciona la cámara para el reconocimiento de piezas, pero al estar el palé terminado, no se reconocerá ninguna, por lo que se da la orden de finalizar el proceso.

4 ELECCIÓN Y DISEÑO DE HERRAMIENTAS

Las herramientas especificadas en esta memoria son las utilizadas en el proyecto a nivel de laboratorio, no en puesta en fábrica. Esto permite comprobar la viabilidad del proyecto con un menor presupuesto y pudiendo trabajar desde el laboratorio, sin tener que interferir en la producción en fábrica.

Por el mismo motivo, se emplearon en la medida de lo posible herramientas ya disponibles en fábrica. No obstante, se seleccionaron las herramientas pensando en aprovechar el mayor número posible de ellas en la aplicación en campo.

Las hojas de características de las herramientas que están disponibles en internet se encuentran en las páginas indicadas en la bibliografía, así como sus modelos de diseño asistido por computadora (CAD).

4.1 Robot

Este sistema es el encargado de posicionar los sensores y herramientas en el lugar y momento adecuado, y de coger y dejar las piezas con la información extraída de los primeros. Se usó el robot de formación de la empresa. Sus componentes son los siguientes.

4.1.1 Manipulador (24)

La herramienta seleccionada para mover el sistema de agarre de las piezas fue el ABB IRB-1200 5/0.9 que se observa en la Ilustración 15, de 6 grados de libertad, capaz de mover 5 kg y con un alcance de hasta 0'9 m desde la base. Tiene 8 cables internos desde la base hasta la muñeca, capaces de transmitir hasta 24V. Además, está equipado con 4 conductos neumáticos internos también desde la base hasta la muñeca.



Ilustración 15. ABB IRB-1200 5/0.9 (24)

4.1.2 Unidad de control (25)

La controladora del Robot es la IRC5 Compact, visible en la Ilustración 16. Es capaz de ejecutar código en lenguaje de alto nivel para el control de movimientos del robot, llamado Rapid, propio de ABB. Interpreta el código y transmite las alimentaciones necesarias a los motores del Manipulador. Además, puede comunicarse con un dispositivo externo mediante Profinet.



Ilustración 16. IRC5 Compact (25)

4.1.3 Software de desarrollo del programa (26)

Para el desarrollo del programa del Robot se usó el programa propio de ABB RobotStudio. Permite escribir en lenguaje Rapid, propio de ABB para programar sus robots, hacer simulaciones creando objetos, y configurar las entradas y salidas de la controladora.

4.1.4 FlexPendant (27)

Para depurar el código se usó el FlexPendant que se observa en la Ilustración 17. Esta herramienta se conecta a la Unidad de control. Permite mover el Manipulador, programar en código Rapid y controlar la ejecución del programa. Es muy útil para depurar el programa, ya que permite mover el robot y guardar sus posiciones, corrigiendo así los errores de precisión en las coordenadas cometidos en la simulación.



Ilustración 17. FlexPendant (28)

4.2 Sistema de visión artificial

Estas herramientas son las utilizadas para reconocer la pose de las piezas. La pose de un elemento es su posición en el espacio y su orientación. Estos datos son necesarios para poder indicarle al Robot dónde se encuentran las piezas y con qué orientación están para poder cogerlas y dejarlas correctamente.

Como se explicó en el apartado de Visión robótica, son varios los elementos que se deben diseñar y dependen en gran medida del tipo de aplicación.

4.2.1 Iluminación (29)

Inicialmente, se intentó utilizar un sistema de iluminación que fuese solidario a la cámara. Se hicieron pruebas con el sistema Effi-Ring de la marca Efficlux (30). Este sistema, como se puede apreciar en la Ilustración 18, es un anillo formado por diodos emisores de luz (leds) que permite posicionar la cámara en el centro. Puede regularse la apertura del ángulo de emisión de los leds y colocarle pantallas difusoras. Este sistema está pensado para iluminar una zona pequeña, no un

área, como se necesita en esta aplicación. Por ello, este sistema generaba una zona demasiado iluminada en el centro, que provocaba saturación. Si se reducía el nivel de intensidad para que no saturase la imagen, quedaban oscuras las zonas de los bordes, como se observa en la Ilustración 19.



Ilustración 18. Effi-Ring (30)

Estas pruebas se hicieron en fábrica. El Effi-Ring se estaba utilizando para otro proyecto, que como se observa en la Ilustración 19, ya estaba colocado en una estructura fija junto con una cámara. Estas pruebas también permitieron ir avanzando en el software de visión, probando que métodos funcionaban mejor.

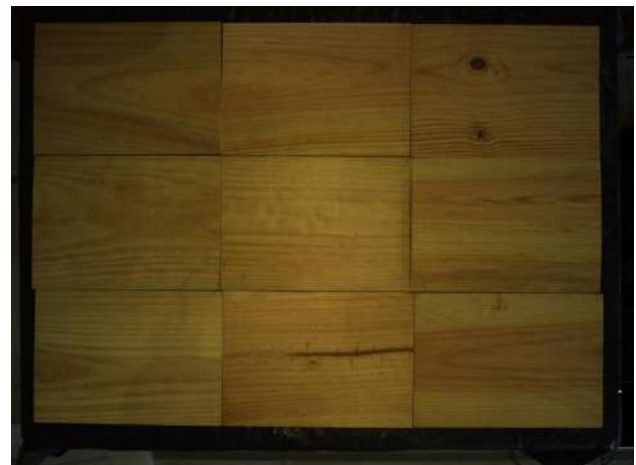


Ilustración 19. Colocación de las tablas con Effi-Ring

El problema con la iluminación no homogénea se trató con el proveedor de la empresa de sistemas de iluminación para conocer que otros sistemas tenían disponibles. Tras analizar la problemática de la aplicación y hacer varias pruebas con las tablas, se decidió utilizar el sistema de iluminación Effi-Flex (29). Como se observa en la Ilustración 20, es una barra de leds. Al igual que en el caso del anillo de iluminación, permite regular el ángulo de apertura y viene equipado con obturadores. Existen diferentes medidas en función del área que se desea iluminar. Para este proyecto se usaron 2 barras de 500 mm de largo con luz roja.



Ilustración 20. Effi-flex (29)

4.2.2 Cámara (31)

En fábrica, se utiliza la cámara iDS UI-5200SE Rev.4 (11) monocromo y a color. En esta aplicación, es posible realizar el conocimiento a partir de una imagen en grises, por lo que se decidió utilizar la cámara monocromo de la Ilustración 21. Además, transmite información por GigaEthernet con un conector RJ45, pudiendo así conectarla al *switch* sin necesidad de adaptador. El modelo CAD está disponible en la página del fabricante.



Ilustración 21. Cámara iDS UI-5200SE Rev.4 (31)

4.2.3 Lente

El área de trabajo con el que permite trabajar la iluminación seleccionada es de aproximadamente 500x400 mm. Posicionando la cámara a 500 mm de las tablas, un objetivo de 12 mm como se observa en la Ilustración 22 es capaz de cubrir toda el área de trabajo. El ángulo vertical de visión es de 45'8° y el horizontal de 59'9°.

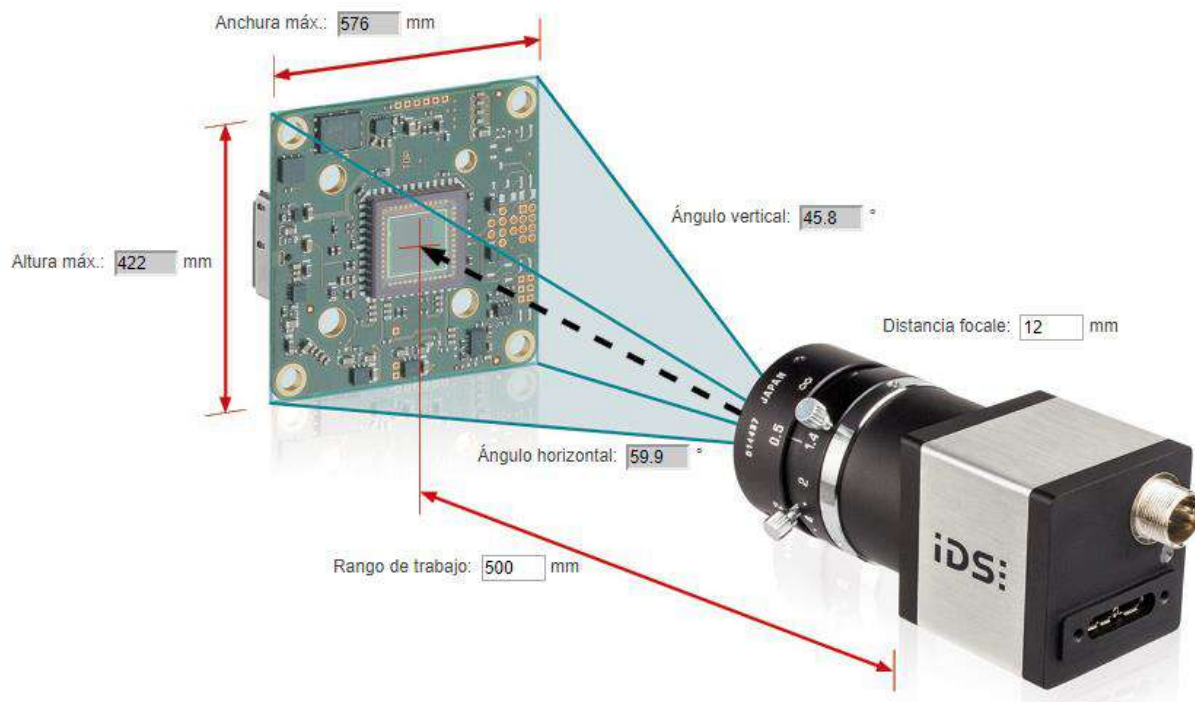


Ilustración 22. Cálculo de objetivo (31)

4.2.4 Filtro

Para que la luz ambiente no influyese, se colocó un filtro por color. El filtro es compatible con la lente, acoplándose perfectamente a ella. Este filtro permite pasar únicamente la luz roja, no permitiendo el paso de la luz ambiente y sí el de la iluminación de las barras al ser esta de color rojo.

4.2.5 Software de desarrollo del programa (32)

Para el desarrollo del programa se barajaron varias opciones. Una de ellas fue el Open CV. No se trata de un software, sino de una biblioteca libre de visión artificial desarrollada por la empresa Intel. La manera de trabajar con estas librerías es generando código. Está disponible en los lenguajes C++, C, Python y Java. Es ampliamente usado para sistemas de vigilancia, reconocimiento de objetos y visión robótica, entre otras (33).

Otra opción que se tuvo en cuenta fue el uso del software Sherlock de la compañía Teledyne Dalsa. El entorno de trabajo en este software es diferente al de Open CV. No se escribe código, sino que se usan un conjunto de herramientas que se aplican directamente sobre la imagen a tratar, facilitando de esta forma la programación, pero haciéndola menos versátil (34).

Finalmente, se decidió utilizar el software de visión artificial Halcon desarrollado por la empresa MVTec, la versión 13. Es ampliamente utilizado en aplicaciones industriales y médicas. Al igual que en el caso de Open CV, se trabaja generando código, pero de una forma guiada, ya que está previsto de una amplia librería de funciones, con ejemplos y explicaciones, indicando los atributos necesarios. A mayores, al igual que en Sherlock, tiene herramientas para aplicar directamente sobre la imagen. También tiene funciones para conectarse con cámaras y realizar calibraciones, facilitando así el desarrollo del programa de visión artificial (32).

4.2.6 PC

El elemento encargado de ejecutar el programa de visión artificial es el ordenador que se usó para el desarrollo del proyecto, un Lenovo ThinkPad Edge con un Intel Core i3. En campo, este ordenador no sería válido, ya que habría que utilizar un PC industrial.

4.2.7 Plato de calibración

Como se explicó anteriormente, se necesita conocer la posición de las tablas a partir de imágenes. Para ello, el sistema de visión artificial necesita conocer la relación entre la distancia en la imagen y la distancia en la realidad y realizar una calibración. El software Halcon 13 tiene asistentes para realizar calibraciones, siendo necesario indicar diversos parámetros y utilizar una plantilla. Estas plantillas se conocen como platos de calibración y suelen ser muy caras debido a su gran precisión. Sin embargo, para esta aplicación, es suficiente con fabricar uno a partir de un documento de un plato de calibración de gran calidad, ya que se obtiene una precisión suficiente para este proyecto. Para ello, se imprimió el documento PDF del plato de calibración facilitado por el suministrador de sistemas de visión artificial y se pegó a una losa de metacrilato de mismas dimensiones y con un espesor conocido. El resultado se puede ver en la Ilustración 23.

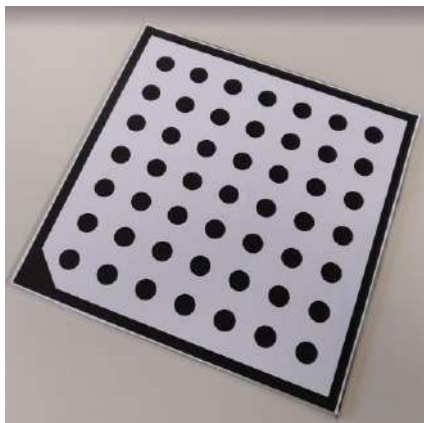


Ilustración 23. Plato de calibración

4.2.8 Software de configuración de la cámara (35)

Para poder configurar la cámara para realizar la comunicación con el programa de visión artificial, fue necesario utilizar el software iDS Camera Manager (35). Este programa permite reconocer la cámara y ajustar los parámetros de comunicación requeridos por el Halcon 13.

4.3 Sistema de agarre de las piezas

Se decidió agarrar las piezas mediante un sistema de vacío. Se barajó el uso de ventosas, siendo necesario el diseño del circuito neumático y de una estructura rígida portadora de las estas. Sin embargo, se decidió implantar la solución explicada a continuación.

4.3.1 Garra de vacío (36)

Para mover las piezas, se decidió usar la garra de vacío Kenos KVG200.60.N213.CVL.S2 de la marca Piab de la Ilustración 24. Esta solución es más eficaz que la de ventosas, ya que es una estructura rígida que consta de una almohadilla de espuma con orificios por los que succiona, proporcionando una mayor sujeción. Además, el circuito neumático está integrado, siendo necesaria únicamente la alimentación con aire a presión. Por último, sus dimensiones permiten trabajar con todas las medidas de piezas posibles. Sin embargo, se debe tener en cuenta que, debido a su diseño, no es capaz de succionar si se posiciona a más de 45° de inclinación sobre el eje horizontal. El modelo CAD está disponible en la página del fabricante. (36)



Ilustración 24. Garra de vacío Kenos KVG200.60. N213.CVL.S2 (36)

4.3.2 Mecanismo de control apertura/cierre (37) (38)

Para controlar el funcionamiento de la succión de la ventosa, se usó una electroválvula Festo MFH-3-1/8 (37). Se trata de una válvula de 2 vías, capaz de trabajar con presiones de entre 1'5 a 8 bar y con accionamiento eléctrico. Para controlar el accionamiento desde un dispositivo electrónico, se usó la bobina IMI Norgren 13J (38), fijada a la electroválvula con una arandela. El conjunto se puede ver en la Ilustración 25. De esta forma, se puede controlar el funcionamiento del sistema de succión mediante una señal digital.

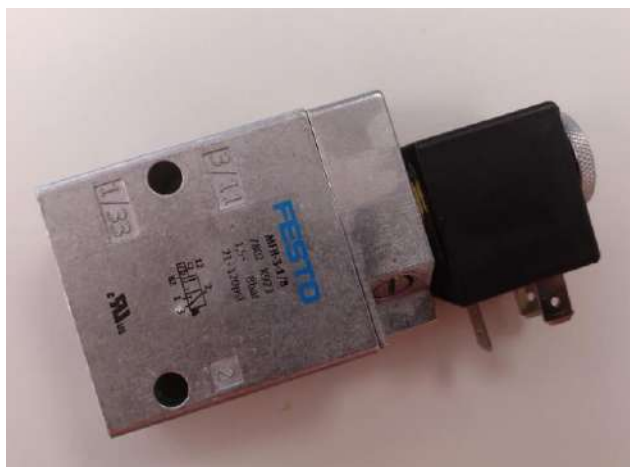


Ilustración 25. Electroválvula y bobina

4.4 Sensor de distancia (39)

Es necesario conocer a qué distancia se encuentran las piezas para poder posicionar la cámara a una distancia fija, ya que se trata de una cámara 2D. Por ello, se decidió utilizar el sensor laser de IFM O1D100 de la Ilustración 26. Tiene un rango de medición escalable y una función de la ventana ajustable. Además, está equipada con una salida analógica y otra digital. Necesita una alimentación de entre 18 y 30 V de corriente continua, por lo que se puede alimentar a través de los cables internos del robot. El modelo CAD está disponible en la página del fabricante (39).



Ilustración 26. Sensor de distancia IFM O1D100 (39)

4.5 Herramienta del robot

Todas las herramientas del robot deben unirse a la muñeca del Manipulador y la controladora debe conocer las constantes de inercia y la masa del conjunto, así como los puntos de referencia de cada herramienta y su orientación, para poder posicionar cada una en el lugar y con la orientación necesaria.

4.5.1 Software de diseño

Las herramientas que se acoplan al soporte tienen un modelo CAD 3D facilitado por el fabricante, salvo la lente. Por ello, se decidió utilizar un programa que permitiese hacer ensamblajes y diseñar en 3D. El programa utilizado fue el SolidWorks 2018. Además, utilizar este programa tiene otra ventaja, ya que permite guardar el ensamblaje en un tipo de archivo reconocible por el software de desarrollo del programa del robot, pudiendo importarlo como una herramienta.

4.5.2 Soporte de las herramientas del robot

Para el diseño, primeramente, se descargaron los archivos CAD 3D de las herramientas y se hizo el de la lente, ya que no está disponible por el fabricante. A mayores, se tomaron medidas de la zona de enganche de herramientas del robot, ya que el CAD de este no tenía las medidas exactas. Los archivos se incorporaron en un mismo ensamblaje y se fueron posicionando, buscando la mejor colocación posible, para a partir de esta, diseñar la estructura de unión de las piezas.

La primera distribución que se pensó estaba pensada en proteger al máximo las herramientas. Por ello, se colocaron la cámara y el sensor de forma que quedasen encima de la garra de vacío, para que no pudiesen recibir un impacto directo en movimientos verticales. A partir de esa disposición, se diseñó una estructura en 2 piezas. Una de las piezas soporta la garra de vacío, la cámara y el sensor. La otra, es una brida para unir el soporte con el robot. El diseño se puede ver en la Ilustración 27.

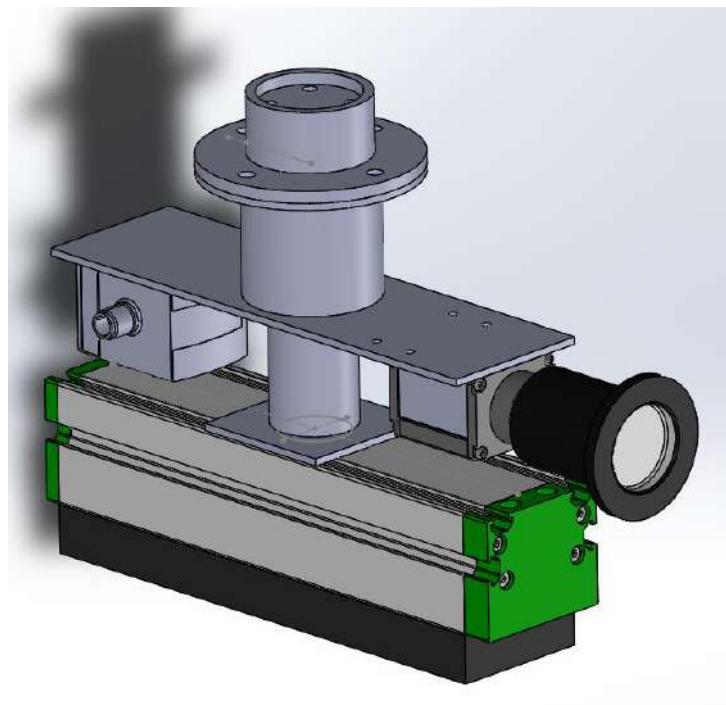


Ilustración 27. Garra inicial

La segunda distribución que se barajó, y la que se acabó eligiendo, fue la de posicionar todas las herramientas con la misma orientación de trabajo. Es decir, tanto la garra de vacío, como el sensor, como la cámara apuntando hacia la misma dirección. Esto simplifica la calibración de la posición de las herramientas que debe conocer la controladora y los movimientos del robot, reduciendo así el tiempo de ciclo. Se debe tener en cuenta que el elemento colocado más abajo debe ser la garra de vacío, y que el sensor de distancia y la cámara deben estar lo suficientemente elevados para garantizar que no puedan impactar contra el palé cuando la garra de vacío esté retirando las piezas. Sin embargo, esta nueva disposición obliga a tener en cuenta el ángulo de visión de la cámara, ya que los elementos colocados por debajo de él podrían interferir. Para ello, a partir de los ángulos vertical y horizontal que proporciona el fabricante de la lente, se representó en el programa de diseño el campo de visión de la cámara, posicionando los demás elementos de modo que no interfiriesen con él.

A partir de esta segunda disposición, se trabajó en el diseño de los elementos de unión. Un primer diseño estaba pensado para fabricar en aluminio. Se decidió partir de un cilindro al que se le soldaba una chapa plegada en forma de caja, de forma que la cámara y el sensor quedasen dentro de esta, tal y como se muestra en la Ilustración 28.

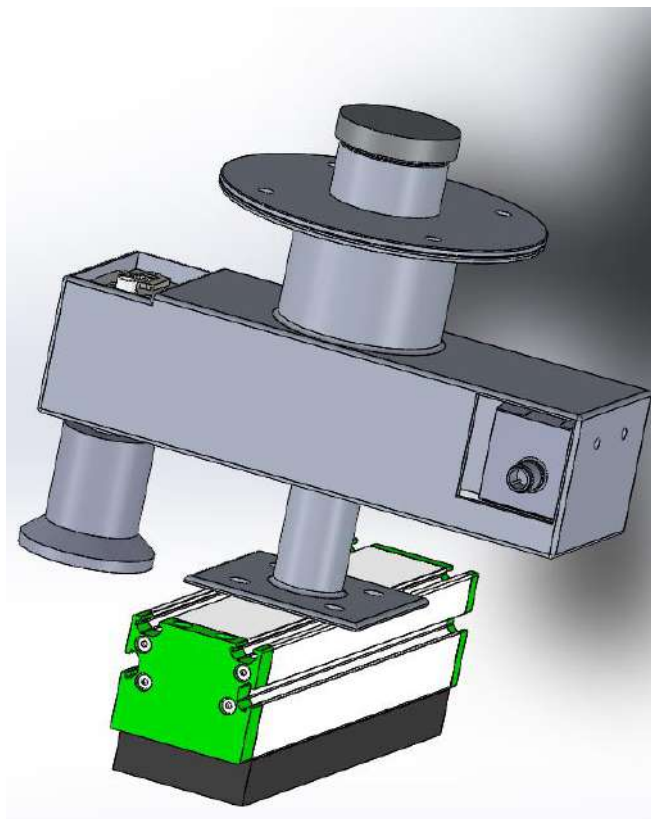


Ilustración 28. Garra tipo caja

Finalmente, se decidió hacer la pieza en plástico por fabricación aditiva. Se conservó el cilindro, pero se sustituyó la chapa doblada en forma de caja, la cual podía ocasionar vibraciones, por una solución nervada. Además, en vez de hacerlo en 2 piezas, se decidió hacerlo en 3, separando el soporte de las 3 herramientas en 2, una para la cámara y el sensor, y otra para la caja de vacío. Esto se hizo pensando en tareas de mantenimiento o en posibles cambios en la garra de vacío utilizada. Las uniones entre estas 3 piezas se diseñaron para que solo hubiese una orientación posible de colocaciones y para que las holguras fuesen mínimas. Además, en el soporte de la garra de vacío, se hicieron unos raíles que enganchan en los de la propia garra de vacío. En las uniones que es necesario atornillar en las piezas de plástico se diseñaron casquillos roscados que van pegados en los orificios de las piezas, ya que, si se hiciese la rosca sobre las piezas de plástico, estas se acabarían desgastando. El diseño definitivo se puede observar en la Ilustración 29, y las piezas que conforman en la Ilustración 30, Ilustración 31 e Ilustración 32. La empresa encargada de su fabricación fue Lupeon (40) y el material seleccionado para su fabricación fue poliamida 2200.

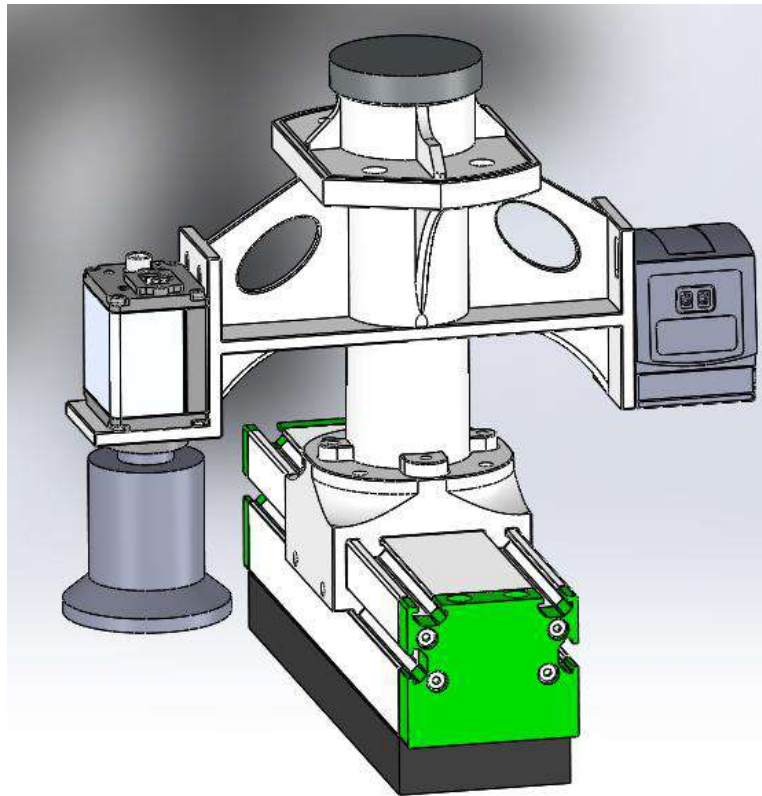


Ilustración 29. Garra definitiva

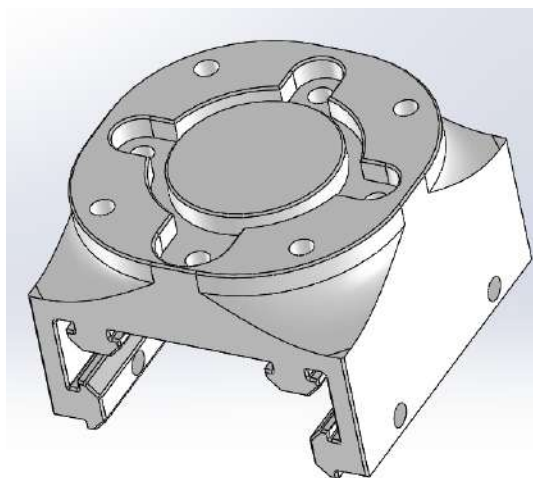


Ilustración 31. Unión garra de vacío

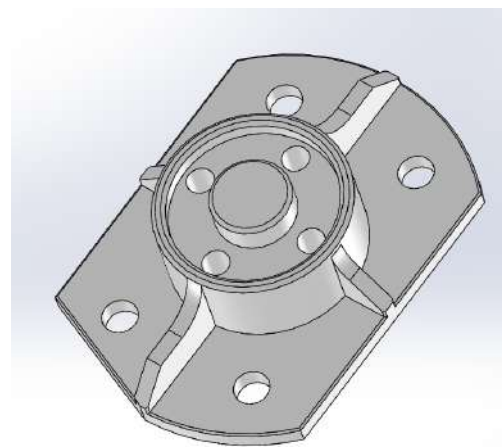


Ilustración 30. Brida

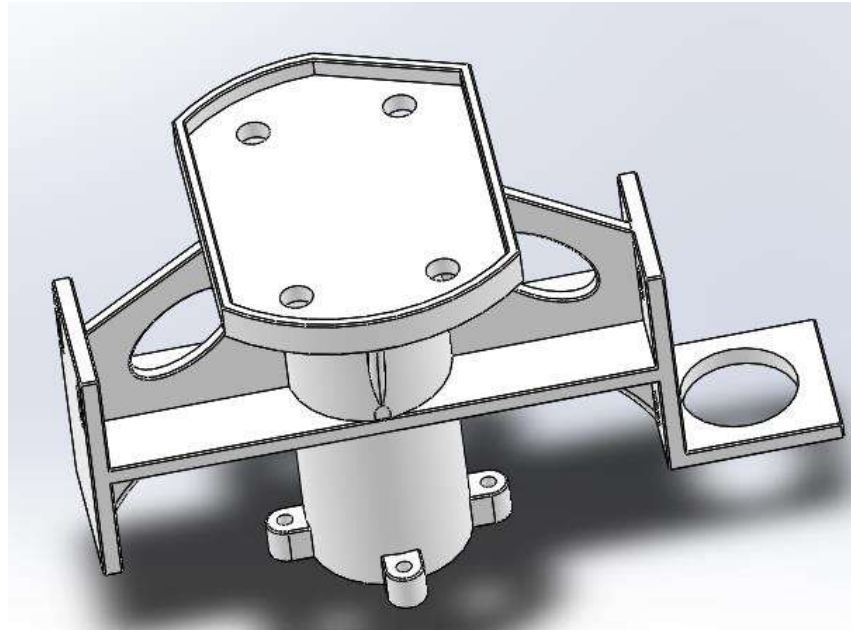


Ilustración 32. Unión cámara y sensor

Sus propiedades físicas relativas al sistema de coordenadas de la muñeca del robot son las siguientes:

- Masa = 1'73090 kg
- $I_x = 46820'46331 \text{ kg/mm}^2$
- $I_y = 46414'49521 \text{ kg/mm}^2$
- $I_z = 9273'91685 \text{ kg/mm}^2$

4.6 Unidad de gobierno

El sistema se gobierna desde un programa ejecutado en un PLC. Las herramientas utilizadas son las siguientes.

4.6.1 Controlador lógico programable (PLC)

Se usó el PLC CPU 1513F-1PN de Siemens (41). Puede comunicarse mediante el protocolo Profinet RJ45 y también con Profisafe y con una región independiente interna de programación para la parte de seguridad. Como fuente de alimentación se usó la fuente SITOP PSU200M 10 A (42). A mayores, por motivos independientes a este proyecto, sino relacionado con otro realizado por Andrés Justo Domínguez sobre comunicaciones, se acopló al PLC y configuró el punto de acceso SCALANCE W774-1 (43). El conjunto se puede ver en la Ilustración 33.



Ilustración 33. PLC

4.6.2 Periferia distribuida

Para conectar el sensor de distancia y la señal de control de la electroválvula se necesitó de tarjetas de señales. Con el fin de mantener el PLC en una zona alejada de la zona de trabajo y utilizar buses para transmitir señales, se decidió colocar las tarjetas próximas al lugar de trabajo usando periferia distribuida. De este modo, los cables tienen que hacer un pequeño recorrido hasta las tarjetas, y estas se comunican con el PLC por bus. Para ello, se usó el módulo IM 155-6 PN HF (44), la tarjeta de entradas digitales DI 16x24VDC ST (45), la tarjeta de salidas digitales DQ 16x24VDC/0.5A ST (46), la tarjeta de salidas AI 4xI 2-,4-wire ST (47) y el módulo SIMATIC ET 200SP (48) para evitar las caídas de tensión en las tarjetas. Además, para alimentar a la periferia distribuida y a los sensores, se colocaron unas fichas para cables conectadas a 24 y 0 V. El conjunto se puede ver en la Ilustración 34.

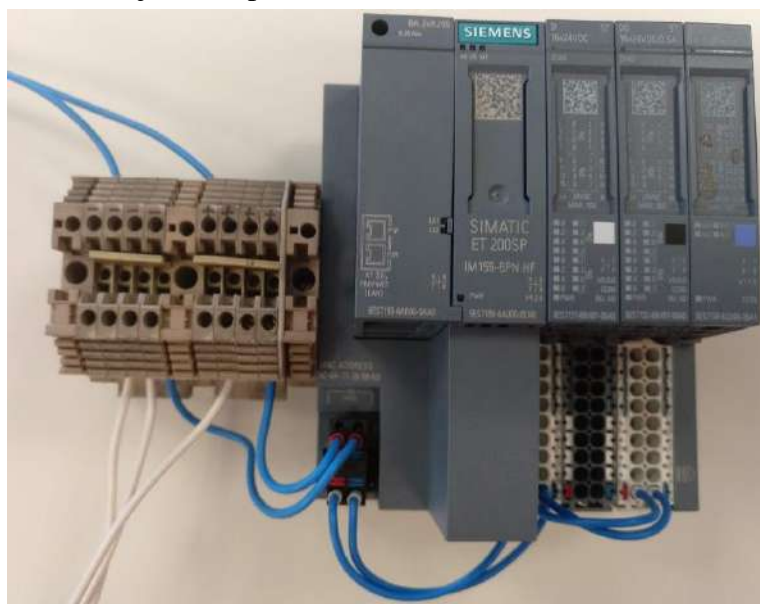


Ilustración 34. Periferia distribuida

4.6.3 Software de desarrollo del programa

Al tratarse de un PLC de la marca Siemens, el software utilizado para el desarrollo del programa fue el TiaPortal. Se usó la versión 15. Permite el uso de los lenguajes KOP, FUP, AWL y SCL, así como crear bloques de organización, de funciones, funciones y bloques de datos. También permite configurar las conexiones de la red, configurar variables de entrada y de salida, y demás opciones que se detallan en el capítulo del PLC.

4.7 Switch

Para conectar los diferentes dispositivos a la misma red se usó inicialmente un *hub* de Macromate. Un *hub* es un dispositivo que concentra el cableado de una red transmitiendo los paquetes de datos entre los puertos. A diferencia de un *switch*, el *hub* no reconoce a que puerto debe enviar los paquetes, sino que los envía a todos, lo cual puede producir colisiones, por lo que están en desuso. Como se observa en la Ilustración 35, tiene 9 conectores RJ45, suficientes para esta aplicación.



Ilustración 35. Macromate SNMP hub

Las comunicaciones entre todos los dispositivos en el desarrollo del proyecto fueron posibles hasta que se conectó la cámara. Los datos de esta no se transmitían, por lo que se decidió usar el *switch* D-Link DES-1005D (49) de la Ilustración 36, permitiendo así las comunicaciones entre todos los dispositivos involucrados.



Ilustración 36. D-LINK DES-1005D (49)

4.8 Mesa de trabajo

Junto al manipulador, se colocó la mesa de la Ilustración 37 para poder posicionar sobre ella las piezas con las que se desea trabajar. Tiene unas dimensiones 900x700 mm.



Ilustración 37. Mesa de trabajo

4.9 Soporte de iluminación

La iluminación juega un papel muy importante en el reconocimiento de las piezas. Por ello, es necesario que las barras de iluminación utilizadas puedan ser posicionadas correctamente. Para ello, su soporte debe ser ajustable, permitiendo variar la orientación y la altura de las barras. La sujeción utilizada consiste en un brazo articulado por barra que la une a la mesa. Los componentes utilizados fueron los siguientes:

- 2 brazos fricción variable sin soporte cámara Manfrotto (50)
- 4 súper pinzas Manfrotto (51)
- 4 uñas para súper pinza Manfrotto (52)
- 4 espigas 1/4'' 3/8'' (53)

Esta solución, que se puede ver en la Ilustración 38, es válida a nivel laboratorio, ya que permito comprobar la viabilidad del proyecto y realizar su desarrollo. Sin embargo, no lo es en campo por varios motivos, como la falta de robustez, o el hecho de que en fábrica los palés no van colocados encima de la mesa, y que al ir retirando las piezas la distancia entre ellas y las barras variaría considerablemente. Esto hay que tenerlo en cuenta para una posible implantación de esta estación en fábrica, diseñando otro soporte y otra forma de iluminación.



Ilustración 38. Soporte de iluminación

4.10 Botonera

Aunque se diseñó una interfaz hombre máquina (HMI) para la estación, no se disponía de él físicamente, por lo que se simulaba. Esto ralentizaba el funcionamiento del PC, por lo que se decidió usar la botonera de la Ilustración 39 para agilizar las pruebas.



Ilustración 39. Botonera

4.11 Elementos auxiliares

Por último, fueron necesarios herramientas y material de electricista para hacer las conexiones eléctricas, conectores racor y mangueras para las conexiones neumáticas, y herramientas de taller como sierras, machos de roscar, etcétera.

5 MONTAJE

Las herramientas se pidieron a los proveedores y, a medida que se recibían en fábrica, se configuraban, probando su funcionamiento mientras se desarrollaba la parte de software del proyecto.

Se hicieron las conexiones eléctricas de alimentación de todos los elementos usando las fichas disponibles solidarias a la periferia distribuida como se aprecia en la Ilustración 40. Aunque era posible utilizar los cables internos del robot para alimentar las herramientas de este, se decidió pasarlos por fuera porque no se disponía del adaptador necesario para conectarse a los pines que se observan en la Ilustración 41.

Las conexiones relativas a las comunicaciones se hicieron usando cables Profinet y Ethernet con conectores RJ45 en el caso de comunicaciones por bus de datos.

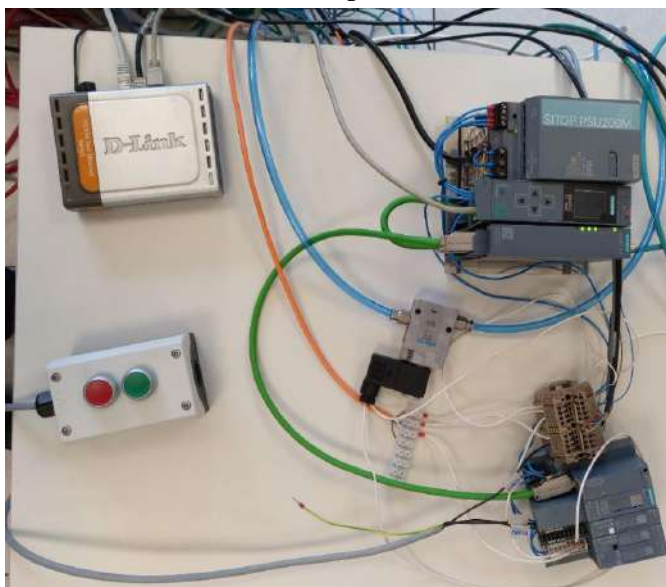


Ilustración 40. Conexiones

Por diseño de la controladora del robot, esta utiliza puertos físicos e IP diferentes para realizar las comunicaciones para trabajar con él en línea desde RobotStudio, de las comunicaciones que realiza por programa, en el caso de esta estación con el PLC. Por ello, fue necesario utilizar una tarjeta de comunicación externa para el PLC, ya que el PC solo disponía de una y era usada para comunicarse en la red del PLC. Por otra parte, tampoco fue posible utilizar el cable Ethernet interno del robot para la comunicación con la cámara debido a que la fábrica lo adquirió sin él.

Las conexiones de comunicaciones que no se realizaban por bus también se montaron, haciendo las configuraciones necesarias, seleccionando el modo de transmisión por corriente 4-20 mA en el sensor de distancia y definiendo el valor mínimo al que transmite 4 mA y el máximo al que transmite 20 mA.

Las conexiones neumáticas sí que se pudieron realizar utilizando los conductos internos del robot.



Ilustración 41. Conexiones internas del robot

Como se observa en la Ilustración 45, los cables y la manguera neumática que conectan con las herramientas del robot fueron guiados por el interior de una vaina abridada en varios puntos de este, permitiendo en todo momento los movimientos del robot sin generar tensión en los cables. De esta forma se evitan enredamientos entre los cables o que estos puedan interferir en la zona de trabajo de la estación.

Al recibir la garra del robot se comprobaron sus medidas. Se detectaron variaciones de hasta 0'3 mm con respecto a las medidas de diseño. Esto no supuso ningún inconveniente salvo en el acoplamiento de la garra de vacío, ya que esta no pasaba por los raíles. Para solventar el problema, se pintaron las zonas de contacto de la garra del robot con la garra de vacío con un rotulador. De esta forma, al internar acoplar las piezas, las zonas en las que se generaba fricción se localizaban debido a que las marcas se borraban. Dichas zonas fueron lijadas hasta conseguir el acoplamiento.

La fabricación de los casquillos para unir las herramientas se hizo en función de las medidas reales de la garra y de su material. Para asegurar un buen agarre del casquillo a la pieza, se decidió que los casquillos fuesen roscados a la garra y pegados con Loctite, en vez de únicamente pegados, como se diseñó inicialmente. Inicialmente, la garra fue diseñada para que los casquillos entrasen en agujeros de 5 mm de diámetro y se enroscasen en ellos tornillos de métrica 3. Para su fabricación, se vació un tornillo en el torno y se roscó su interior. Debido a que el macho de roscar de 3 mm era demasiado fino, se decidió que la rosca interna fuese de 4. La rosca externa, al ser el agujero de 5 mm, se decidió hacer de métrica 6.

Para poder introducir los casquillos en la garra se taladraron los agujeros con una broca de 5 mm, aumentando así su profundidad para poder hacer correctamente la rosca con el macho de roscar de 6 mm, ya que la profundidad de este no fue pensada para hacer una rosca, sino únicamente introducir la totalidad del casquillo sin roscar. También fueron taladrados los agujeros por los que se introducen los tornillos en la garra de unión de la cámara y el sensor con la misma broca, ya que estaban pensados para tornillos de métrica 3 y no de 4. Con los agujeros realizados, se roscaron tal y como se observa en la Ilustración 42.

Una vez realizadas las roscas, se atornilló una tuerca seguida del casquillo a un tornillo, apretando la tuerca contra el casquillo como se observa en la Ilustración 43. De esta forma el casquillo podía ser atornillado en el agujero sin que el tornillo girase sobre el mismo. Este procedimiento se aplicó para todos los casquillos aplicando Loctite en su rosca externa antes de enroscarlos. Para retirar el tornillo sin que saliese el casquillo, se fijaba la tuerca y se desenroscaba, impidiendo así que el casquillo se moviese.

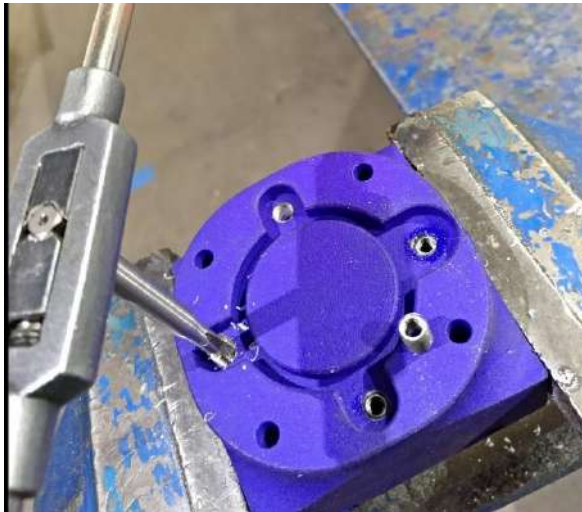


Ilustración 42. Roscado de la garra del robot



Ilustración 43. Casquillos y garra del robot



Ilustración 44. Garra del robot montada

Por último, se colocaron las barras de iluminación a los lados de la mesa de trabajo. En la Ilustración 44 se puede ver la garra del robot y en la Ilustración 45 la estación completa en el laboratorio.



Ilustración 45. Estación

6 PLC

En este apartado se explica todo lo relativo a la programación del PLC. Como se ha explicado anteriormente, el PLC es la unidad de gobierno. Su función es la de enviar las ordenes necesarias al resto de los elementos en función de la información recibida y encargarse de la seguridad de la estación.

6.1 TiaPortal V15

El programa utilizado para la configuración del PLC fue el TiaPortal V15. Permite crear proyectos en los cuales puedes añadir los dispositivos que se requieren utilizar, programarlos, escanear la red para detectar dispositivos, cargar y descargar programas, y conectarse mientras están ejecutando programas, pudiendo observar el estado de sus variables en tiempo real.

La forma de programar en este entorno es mediante bloques. Existen 4 tipos de bloques:

- Bloque de organización (OB). Son los bloques de orden superior y se ejecutan de forma cíclica
- Bloque de función (FB). Son bloques que depositan sus valores de forma permanente en bloques de datos de instancia
- Función (FC). Son bloques lógicos sin memoria
- Bloque de datos (DB). Sirven para almacenar datos del programa. Permite agrupar diferentes tipos de datos, pudiendo anidar dichas agrupaciones con el fin de tener un mayor orden.

Estos bloques, exceptuando los bloques de datos, pueden ser programados en distintos lenguajes. En este caso, se decidió trabajar en SCL. Es un lenguaje estructurado de Siemens basado en texto.

6.2 Hardware

Tras crear un proyecto, se escanearon los dispositivos conectados. Estos dispositivos habían sido utilizados en otras aplicaciones y daban diversos errores de conexión, por lo que fue necesario formatear la tarjeta de memoria y actualizar los *firmwares*. Al escanear los dispositivos, permite cambiar sus IP y sus nombres. La configuración de las IPs se explica en el apartado de Comunicaciones. Además, también se proporciona el número de referencia del dispositivo. Con ese número se fueron añadiendo al proyecto los dispositivos utilizados, los cuales se especifican en el apartado de Elección y diseño de herramientas, a partir del catálogo en el TiaPortal, haciendo las configuraciones de hardware necesarias para que coincidan con las reales y especificando las configurables.

- PLC. Nombre: PLC_visión_robótica Versión de firmware V2.5. Además, se habilitó comunicación OPC UA y se cambió la versión Runtime. La comunicación OPC UA se usó para comunicar el sistema de visión artificial con el PLC. Esto se explica en detalle en el apartado de Comunicaciones.
- Periferia distribuida. Nombre: tarjetas. Versión de firmware V3.3. Se le añadieron las tarjetas indicadas en el apartado de Elección y diseño de herramientas y se configuraron sus bases. Las bases son la parte de la tarjeta donde van conectados los cables de los sensores o actuadores. Estas necesitan alimentación de 24V, y dependiendo del tipo de tarjeta las obtienen de una forma u otra. A mayores, las de entrada analógica tienen más opciones. Las configuraciones realizadas fueron las siguientes:
 - Entradas digitales. Alimentación externa (color blanco)

- Salidas digitales. Alimentación desde la tarjeta de su izquierda (color gris)
- Entradas analógicas. Alimentación externa (color blanco). En el canal 0, que es al que se conectó el sensor, se configuró para que la medición la hiciese a 4 hilos por intensidad, con un rango de 4-20 mA por especificaciones del sensor utilizado.
- Scalance. Nombre: antena
- HMI. Nombre: HMI_2. Modelo TCP1500 Comfort
- GSD Basic V1.2. Nombre: robotlaboratorio

Los 2 últimos dispositivos no aparecieron al hacer el escaneado. El HMI, que son las siglas de interfaz hombre-máquina, porque no existe físicamente, sino que es una simulación en el propio ordenador utilizado para el desarrollo del proyecto.

El robot, porque el programa no es capaz de reconocerlo, sino que es necesario instalar el dispositivo GSD que pertenece al robot y agregarlo al proyecto. Esto se detalla en la parte de comunicaciones.

La antena se configuró por ser necesaria para otro proyecto el cual accede a este PLC.

Una vez añadidos los dispositivos y configuradas sus opciones de hardware, se indicaron las conexiones en la ventana de redes tal y como se observa en la Ilustración 46.

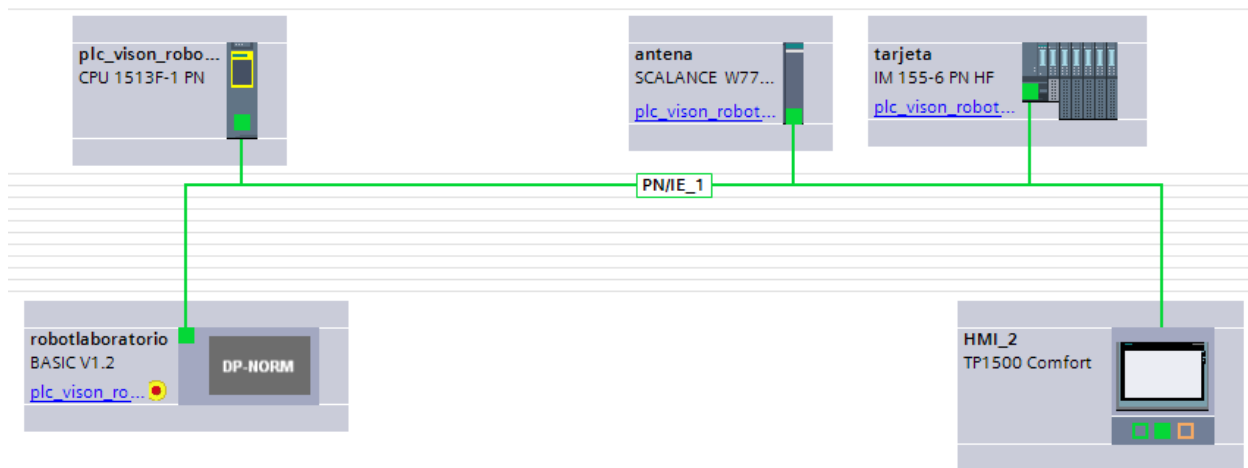


Ilustración 46. Vista de redes

Por último, se especificaron las direcciones de memoria de las variables físicas. Para tener un mayor orden, se declararon 2 tablas de variables: una para las entradas y las salidas de las tarjetas de la periferia distribuida, y otra para las variables del robot. Las tablas de variables son listas que permite tener organizadas las variables independientemente del lugar de memoria que ocupen. No fue necesario definir variables procedentes del sistema de visión artificial puesto que esta comunicación se realiza mediante OPC UA.

En la tarjeta de entradas analógicas se definió la variable “AI_distancia” como una entrada tipo *int* (16 bits) y en la tarjeta de salidas digitales la variable «ventosa» como una salida de tipo booleano, y ambas se vincularon a la tabla de variables de entradas y salidas de las tarjetas. A mayores, se definieron 2 variables en la tarjeta de salidas digitales que se sincronizaban con el DB del HMI para no tener que simularlo en el PC como ya se ha comentado.

En la tabla de variables del robot se vincularon las entradas y las salidas de este. En el caso de las tarjetas, el lugar físico de conexión se realiza por cables, pudiendo así vincular los bits afectados por dichas conexiones con las variables de las tarjetas. Sin embargo, el robot se comunica mediante un bus de datos, por lo que la declaración de variables no es tan trivial. Esto se explica en detalle en el apartado de Comunicaciones.

6.3 Software

Realizadas las configuraciones hardware necesarias se procedió a la programación del software. Primeramente, se decidió la estructura del programa, decidiendo cómo organizar las funciones, los bloques de datos y las variables.

Se decidió no trabajar con bloques de funciones puesto que no se creyó necesario. Estos bloques tienen asociado sus propios bloques de datos, ya que guardan sus valores de forma permanente. Se suelen utilizar cuando se tienen diferentes dispositivos iguales, ya que con el mismo programa se pueden controlar varios dispositivos iguales. En este caso, como solamente hay un robot en esta estación y un dispositivo de ejecución del programa de visión artificial, se decidió utilizar funciones.

Se crearon funciones independientes con bloques de datos también independientes para el control del robot y el sistema de visión artificial, a mayores de las demás funciones y bloques de datos necesarios. Esto se hizo así para tener un mayor orden en la estructura del programa, teniendo independizado las estructuras de los 2 sistemas, facilitando posibles tareas de modificación de programa o de dispositivos.

Los bloques utilizados fueron los siguientes:

- Main. OB encargado del programa principal.
- Actualización_entradas. FC que actualiza los DB con las variables de entrada.
- Modo_funcionamiento. FC que determina el modo de funcionamiento.
- F_Robot: FC que controla las órdenes enviadas al robot.
- F_vision_artificial: FC que controla las órdenes enviadas al sistemas de visión artificial.
- Actualización_salidas. FC que actualiza las variables de salida en función de los DB.
- ROBOT. DB con los datos recibidos y enviados al robot.
- VISION. DB con los datos recibidos y enviado al robot.
- OPC_VISION: DB que modifica y lee el sistema de visión.
- TARJETAS. DB con los datos de las variables de las tarjetas de la periferia distribuida.
- MODO. DB con los datos relativos al FC Modo_funcionamiento.
- HMI. DB con los datos relativos al HMI.

A continuación, se explica el funcionamiento de los bloques que conforman el programa:

6.3.1 Main

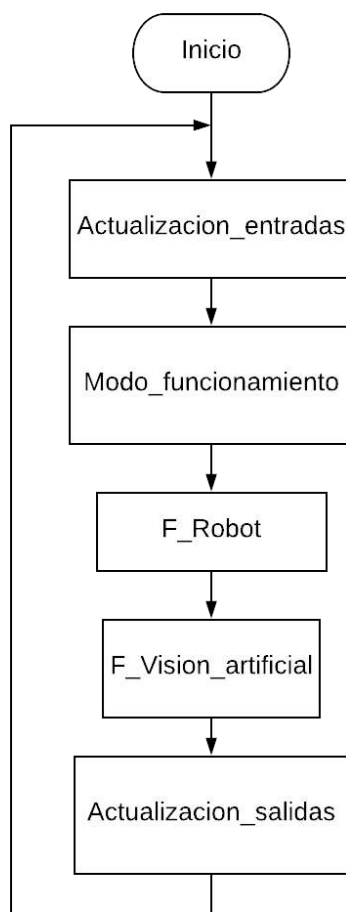


Diagrama 6. Main PLC

En el Diagrama 6 se muestra el bloque principal del programa. Se ejecuta de forma cíclica y en él se llaman a todas las funciones necesarias.

Las funciones leen y modifican los datos de los bloques de datos y no directamente las variables de entrada y salida. De esta forma, se guarda el estado de las entradas y todas las funciones se ejecutan bajo las mismas condiciones. Una vez ejecutadas todas, se actualizan todas las salidas de forma sincronizada.

Por ello, primeramente, se ejecuta la función *Actualizacion_entradas*, seguido de la función *Modo_funcionamiento*. Con los bloques de datos actualizados y el modo de funcionamiento escogido, se ejecutan las funciones *F_Robot* y *F_Vision_artificial*, los cuales determinarán los órdenes que se les deba transmitir a cada sistema. Por último, se ejecuta la función de *Actualizacion_salidas*, modificando el estado de estas en función de los bloques de datos.

6.3.2 Actualizacion_entradas

En este FC se igualan los datos relativos a las entradas del robot del DB ROBOT, VISION y TARJETAS a las variables de entrada del robot, los datos relativos a las entradas del DB OPC_VISION y las variables de las tarjetas de entrada, respectivamente, actualizando así todos los datos de variables que pueden ser modificados por los demás dispositivos. Todas las variables son booleanas salvo la variable «ventosa», que es una entrada analógica de tipo booleano que es necesario tratar. Esto se explica en el apartado de Comunicaciones. El DB HMI no se actualiza en esta función porque al simularse el dispositivo HMI modifica directamente sus datos.

6.3.3 Modo_funcionamiento

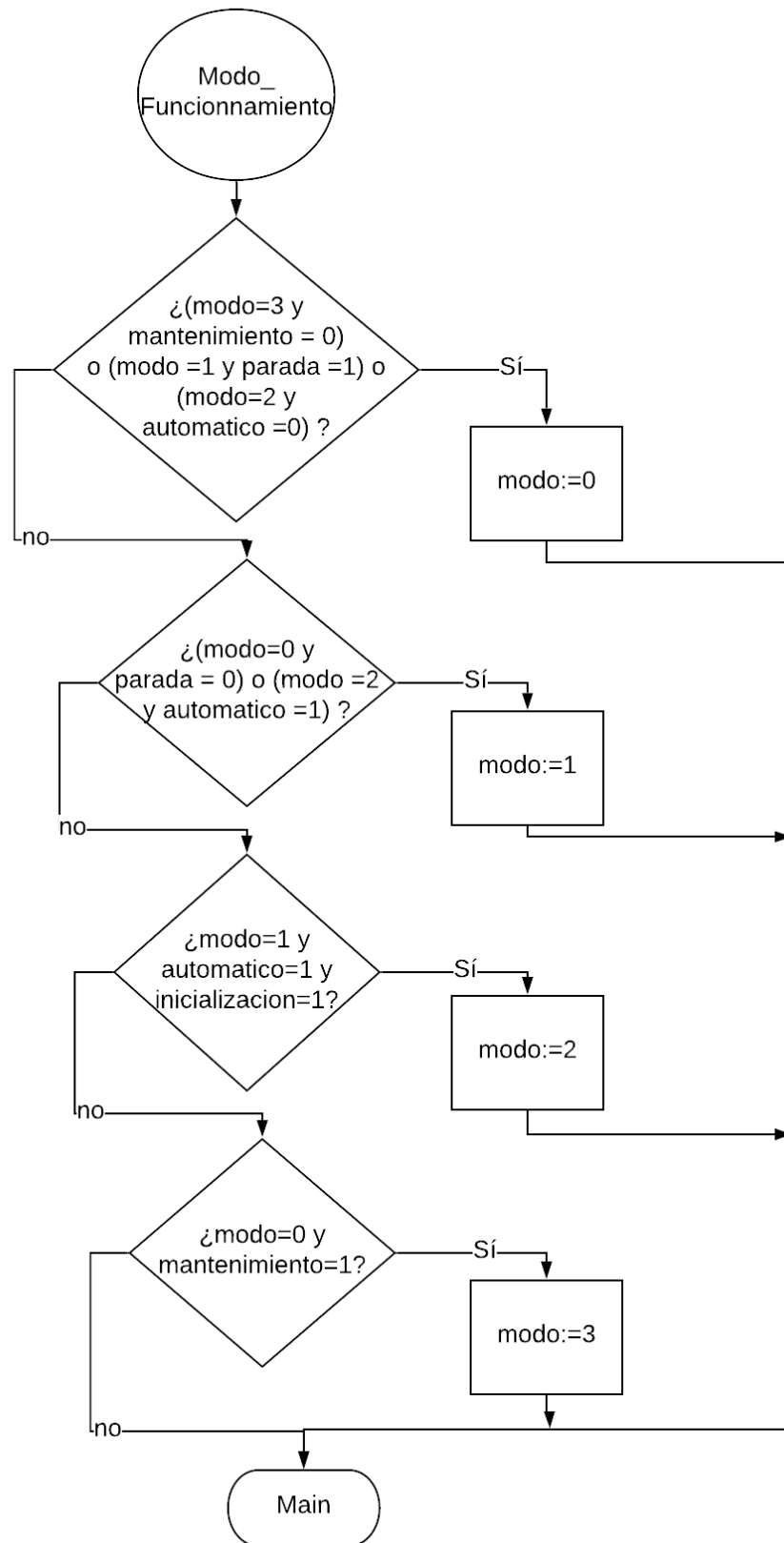


Diagrama 7. Modo_funcionamiento PLC

Para la elección del modo de funcionamiento se declaró una variable en el DB MODO llamada «modo» de tipo *int*. En función del valor de esta variable, la estación está en un modo u otro según el Diagrama 7. Se definieron los siguientes modos:

- modo=0: paro
- modo=1: manual
- modo=2: automático
- modo=3: mantenimiento

El modo de funcionamiento se determina por el estado del modo actual y los datos del DB HMI de la siguiente forma:

Si la estación se encuentra en modo paro y se selecciona el botón de inicialización (inicialización=1), pasa a marcha. Si se selecciona modo mantenimiento (mantenimiento=1), pasa a mantenimiento.

Si se encuentra en modo marcha y se selecciona el modo automático (automático=1), pasa a automático. Si se pulsa el botón de paro pasa a paro.

Si se encuentra en modo automático y se selecciona modo manual (automático=0), pasa a modo manual. Si se pulsa el botón de paro pasa a paro.

Por último, si se encuentra en modo mantenimiento y se deselecciona dicho modo (mantenimiento=0), pasa a paro.

Únicamente se ha diseñado la estación para trabajar en modo automático, siendo necesario la programación de los demás modos para su implantación en fábrica.

6.3.4 F_Robot

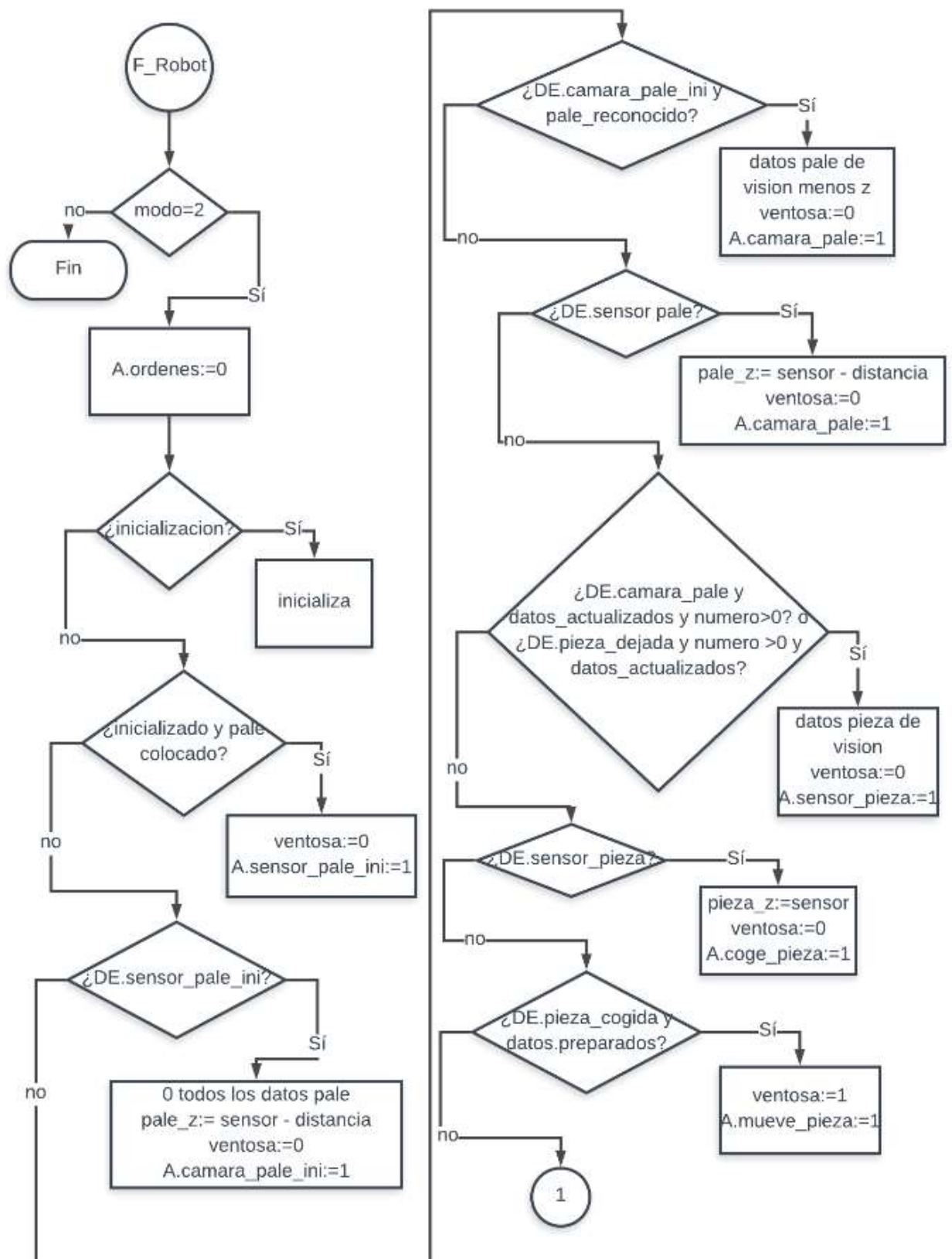


Diagrama 8. F_Robot PLC (1)

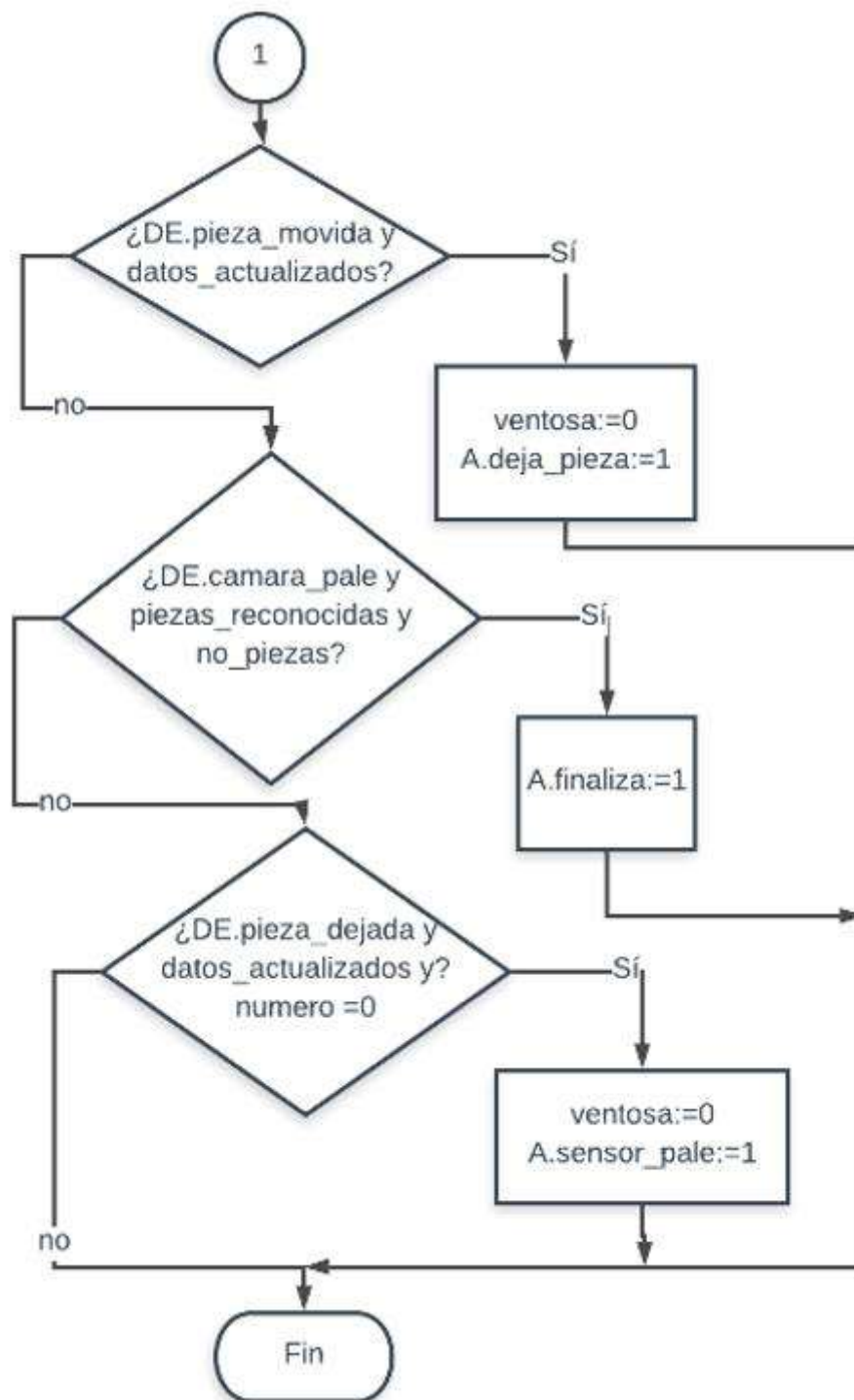


Diagrama 9. F_Robot PLC (2)

En el Diagrama 8 y Diagrama 9 se puede ver el funcionamiento del FC F_Robot. Este FC lee datos de los DB de ROBOT, VISION, HMI y TARJETAS, pero únicamente modifica el DB ROBOT y el DB TARJETAS, puesto que se encarga de controlar al Robot y controla la caja de vacío.

Cada vez que es ejecutado, se comprueba si la estación está en modo automático. Si lo está, desactiva los booleanos del DB relativos a las órdenes del Robot, y se van comprobando los datos del Robot y de Visión para decidir qué datos modificar para realizar el proceso según la secuencia del Diagrama 5.

La caja de vacío, controlada por el dato ventosa, solo se activa cuando el robot debe mover una pieza, y permanecerá activa hasta que se le transmita otra orden al robot.

Cuando el robot debe posicionar el sensor inicialmente sobre el palé, no se le transmiten los datos de posición del palé, ya que todavía no se conoce.

Los datos que se le transmiten al robot del palé y de las piezas los usa para hacer movimientos relativos al punto que tiene guardado para posicionar las herramientas que realizan las mediciones. Este punto el robot lo va actualizando. Por ello, en ciertas ocasiones es necesario actualizarle los valores, y en otras indicarle que esos datos son 0. Esto se explica en detalle en el apartado de Movimientos del robot.

El posicionamiento de la cámara inicialmente se hace para reconocer el palé. En este caso, se transmite el dato de cuánto debe moverse para estar a la altura idónea, que es la altura leída por el sensor menos la altura de visión a la que debe estar la cámara, otorgándoles a los demás datos relativos al pale el valor de 0, ya que aún se desconoce la posición del centro del palé.

La orden de posicionamiento de la cámara sobre el palé se efectúa procediendo de 2 situaciones. Una de ellas, tras reconocer el palé; la otra, tras extraer todas las piezas de un nivel y haber actualizado la distancia a la que se encuentran las siguientes. En el primer caso, únicamente se conoce el dato de la altura, por lo que se le proporcionan los datos del palé procedentes del DB VISION. En el segundo caso, el robot ya conoce los datos procedentes de visión, y estos siguen siendo los mismos, siendo el único que varía, y por tanto el que se actualiza en el DB ROBOT, el de la altura.

Para que el robot coloque el sensor sobre las piezas, se deben actualizar los datos de dicha pieza, y para que pueda ser cogida por la garra de vacío, se le envía el dato de la altura obtenida por el sensor.

Para asegurarse de que los datos recibidos son de la siguiente pieza y no de la ya retirada, antes de indicarle al robot que vaya al centro de una pieza, comprueba si los datos de la siguiente pieza están actualizados para ser mandados desde el pc de visión, y antes de indicarle al robot que mueva una pieza, comprueba si los siguientes datos están preparados.

6.3.5 F_Vision_artificial PLC

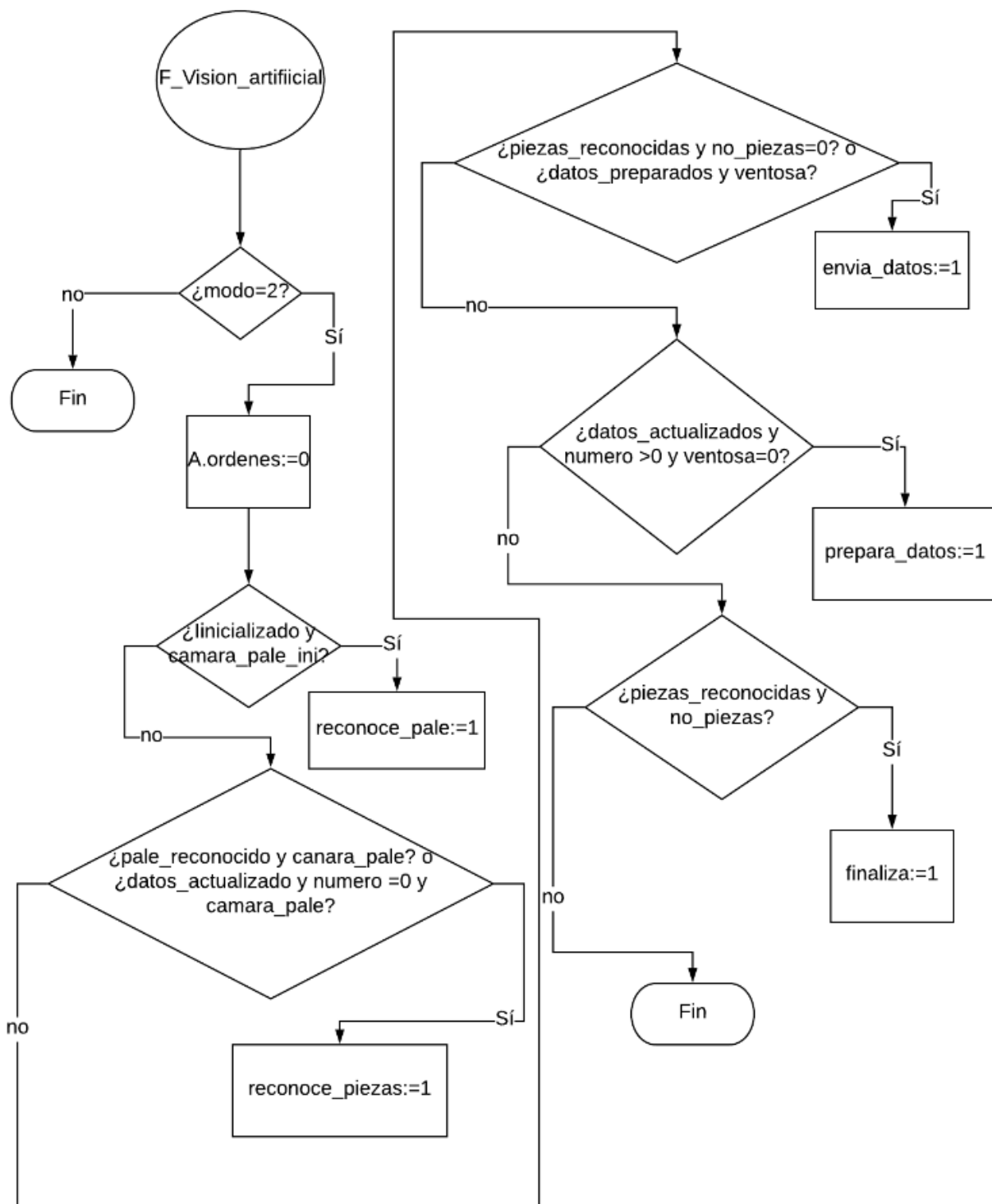


Diagrama 10. F_Vision_artificial PLC

El Diagrama 10 refleja el funcionamiento del FC F_Vision_artificial. Este FC lee datos de los DB de ROBOT, VISION y HMI, pero únicamente modifica el DB VISION puesto que se encarga de controlar al sistema de visión artificial.

Al igual que el FC F_Robot, comprueba si la estación está en modo automático, y si lo está, realiza las funciones necesarias para que la estación trabaje según el Diagrama 5, desactivando los booleanos relativos del DB VISION de las órdenes del sistema de visión y comprobando los estados de la planta para activar la orden necesaria.

Las ordenes de envía_datos y preparada_datos se usan para enviar los datos de las piezas detectadas. El proceso se hace en dos partes para poder controlar si se envía únicamente el dato de la siguiente pieza. De solo existir envía_datos, podrían no enviarse y seguir activa la señal debido a que los datos anteriores ya se enviaron, o estar enviando datos de piezas sin que al robot le dé tiempo a retirarlas.

Además, estos procesos se realizan teniendo en cuenta la variable ventosa. De esta forma, el proceso de preapara_datos de la siguiente pieza se realiza mientras el robot se dirige a la pieza anterior, y el de envía_datos mientras la está retirando. De esta forma se optimizan los tiempos de ciclo.

6.3.6 Actualizacion_salidas

En este FC se igualan las variables de salida del robot, los datos relativos a las salidas del DB VISION y las variables de las tarjetas de salida a los datos relativos a las salidas del DB ROBOT, VISION y TARJETAS, respectivamente. A mayores, también se modifica el DB HMI actualizando el estado de la variable modo.

7 ROBOT

El robot utilizado en esta estación, aunque se rige de las órdenes recibidas por el PLC, tiene su propio programa que ejecuta cíclicamente cuando se encuentra en modo automático. Esto ocurre con la inmensa mayoría de robots industriales, siendo la controladora la encargada de transmitirle las alimentaciones a los motores de las articulaciones del manipulador en función de las especificaciones del programa y de la lectura de sus sensores internos. Para ello, las diferentes casas de fabricantes de robots han desarrollado software y lenguaje propios para facilitar la tarea del programador del robot. A continuación, se explica el funcionamiento de las herramientas utilizadas para la configuración del robot y el programa de funcionamiento desarrollado junto con las calibraciones.

7.1 RobotStudio 6.8 y Rapid

RobotStudio es el software de ABB usado para la configuración de sus robots. Tiene librerías con diseños CAD de robots y de herramientas, y permite incorporar archivos para poder diseñar de forma virtual una estación con herramientas diseñadas por el programador. Además, el robot virtual tiene movilidad, pudiendo posicionar la herramienta con la configuración requerida, guardar las posiciones y crear trayectorias a partir de ellas. El software permite guardar estas posiciones y trayectorias en lenguaje Rapid, pudiendo así usarlas en el desarrollo del programa.

El desarrollo del programa en código Rapid también se programa desde este entorno, pudiendo sincronizarse con la estación virtual para así poder simular los movimientos del robot en el diseño CAD. Rapid es un programa textual de alto nivel específico para robots de ABB, formado por secuencia de instrucciones. El lenguaje está compuesto de la siguiente forma:

- Rutina principal. Es la rutina donde se inicia la ejecución y desde la que se llama a las demás rutinas.
- Conjunto de subrutinas. Permite estructurar el programa de forma modular dividiendo las rutinas.
- Datos de programa. Definen los datos necesarios para controlar el funcionamiento del robot.
- Módulos. Están formados por rutinas, datos de programa e instrucciones. Permiten un mayor orden del programa.
- Rutinas. Conjunto de datos e instrucciones. Pueden ser de 3 tipos:
 - Procedimientos. Permiten ejecutar instrucciones en función de parámetros. Debe existir siempre un único procedimiento llamado *main*, que es cíclico y en el que comienza la ejecución.
 - Funciones. Realizan operaciones sobre un determinado dato y devuelve su valor en función de los parámetros asignados.
 - Interrupciones. Realizan instrucciones por interrupción sin la posibilidad de ejecutarse en función de parámetros.
- Datos de programa. Conjunto de datos necesarios para definir los parámetros de funcionamiento de las rutinas.

Pueden ser de 3 tipos en función de las modificaciones del dato:

 - Constantes. Su valor no es reasignable.
 - Variables. Su valor es reasignable.
 - Persistentes. Su valor es reasignable, pero al variarse se varía también su valor de inicialización.

Los principales tipos de datos utilizados son los siguientes. En la Ilustración 47 se aclaran los datos relativos a las posiciones del robot

- *Num.* Valor numéricos reales entre -8388607 y 8388608.
- *Dnum.* Valores numéricos reales entre -4503599627370496 y 4503599627370496.
- *Bool.* Niveles lógicos verdadero/falso.
- *Wobdata.* Es un sistema de coordenadas. Las tareas realizadas sobre un mismo objeto están referidas a su objeto de trabajo. De esta forma, si se varía la posición del objeto real, con actualizar los datos del objeto de trabajo, las posiciones utilizadas en su rutina siguen siendo las correctas, ya que sus datos son relativos al objeto de trabajo especificado.
- *Tooldata.* Datos de las herramientas del robot. Se define la masa, inercia y TCP. El TCP (*tool centre point*) son los ejes que definen el punto y orientación de una herramienta con respecto de la muñeca del robot. Cuando el robot, con una determinada herramienta, adopta una pose, es el TCP de la herramienta la que se sitúa en la pose determinada.
- *Loaddata.* Datos de masa e inercias de los objetos transportados por el robot.
- *Speeddata.* Datos de las velocidades de movimiento del robot.
- *Zonedata.* Datos de la precisión del posicionamiento del robot sobre los *robtarg*.
- *Robtarg.* Son los puntos usados para configurar las trayectorias del robot. Se definen por datos de posición, orientación y configuración. Un punto de posición del robot está definido por la pose del TCP en ese punto y por la configuración de las articulaciones del robot, ya que pueden existir diferentes disposiciones de las articulaciones para una misma pose.
- Instrucciones. Hay diversos tipos de ellas. Las más utilizadas son:
 - IF. Ejecuta una serie de instrucciones en función de una condición.
 - Set. Modifica a valor verdadero una variable booleana de salida.
 - Reset. Modifica a valor falso una variable booleana de salida.
 - WaitDI. Espera a que una variable booleana de entrada esté en valor verdadero.
 - MoveL. Instrucción encargada del movimiento del robot. Es necesario especificarle el *robtarg* de destino, *speeddata*, *zonedata*, herramienta (para saber el TCP que se debe llevar al *robtarg*) y *wobdata* la que está referido el *robtarg*. Crear una trayectoria entre el *robtarg* en el que se encuentra antes de ejecutar la instrucción y el de destino. Esta trayectoria es lineal.
 - MoveJ. Igual que MoveL pero la trayectoria no es lineal, sino optimizando los movimientos de las articulaciones.
 - WaitTime. Espera un determinado tiempo.
 - GripLoad. Le indica al robot que la herramienta ha cogido un objeto, utilizando así sus datos para los cálculos de la dinámica del robot.
 - CRobT. Guarda la pose y configuración del robot en un *robtarg*.

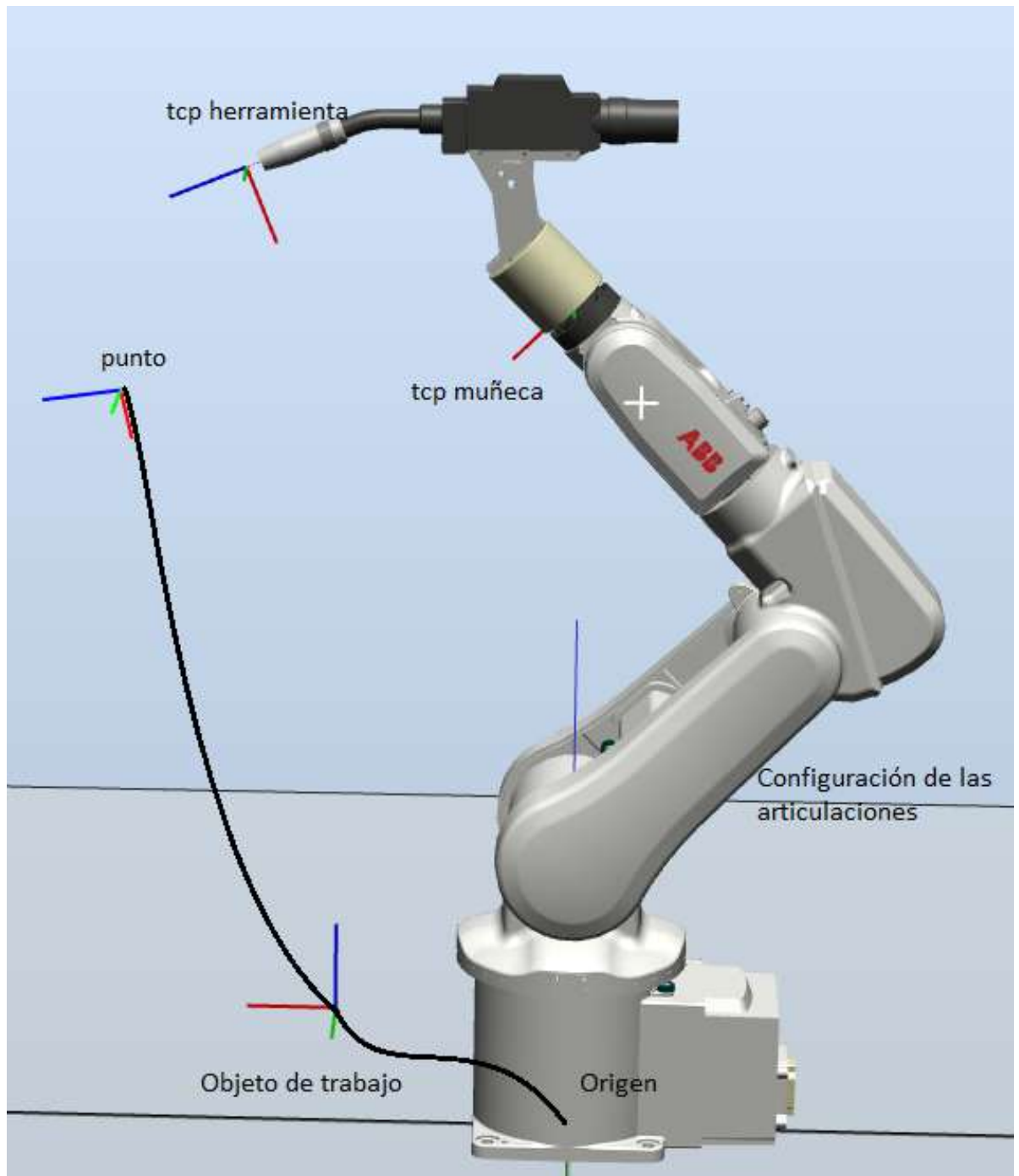


Ilustración 47. Datos de posiciones de un robot

Por último, las configuraciones especificadas en RobotStudio se pueden cargar desde este entorno conectándose al robot por el puerto de la controladora específico para ello. A mayores, permite realizar todas las funciones descritas anteriormente directamente sobre el robot real. (54) (55)

7.2 Estación virtual

Para crear la estación virtual de la Ilustración 48 en RobotStudio, se creó una nueva estación a partir de un *BackUp* del robot real. Se definieron los objetos de trabajo y las posiciones necesarias y se sincronizaron con el programa Rapid.

Además, se importó el archivo CAD de la herramienta como un objeto, y a partir de él se definió una herramienta, introduciendo sus datos físicos y definiendo los TCP de la garra de vacío, del sensor y de la cámara.

Tras desarrollar el programa en Rapid, se sincronizó este con la estación para comprobar los movimientos realizados por el robot y posteriormente se cargó al robot real, trabajando directamente en línea sobre él ajustando tiempos y parámetros.

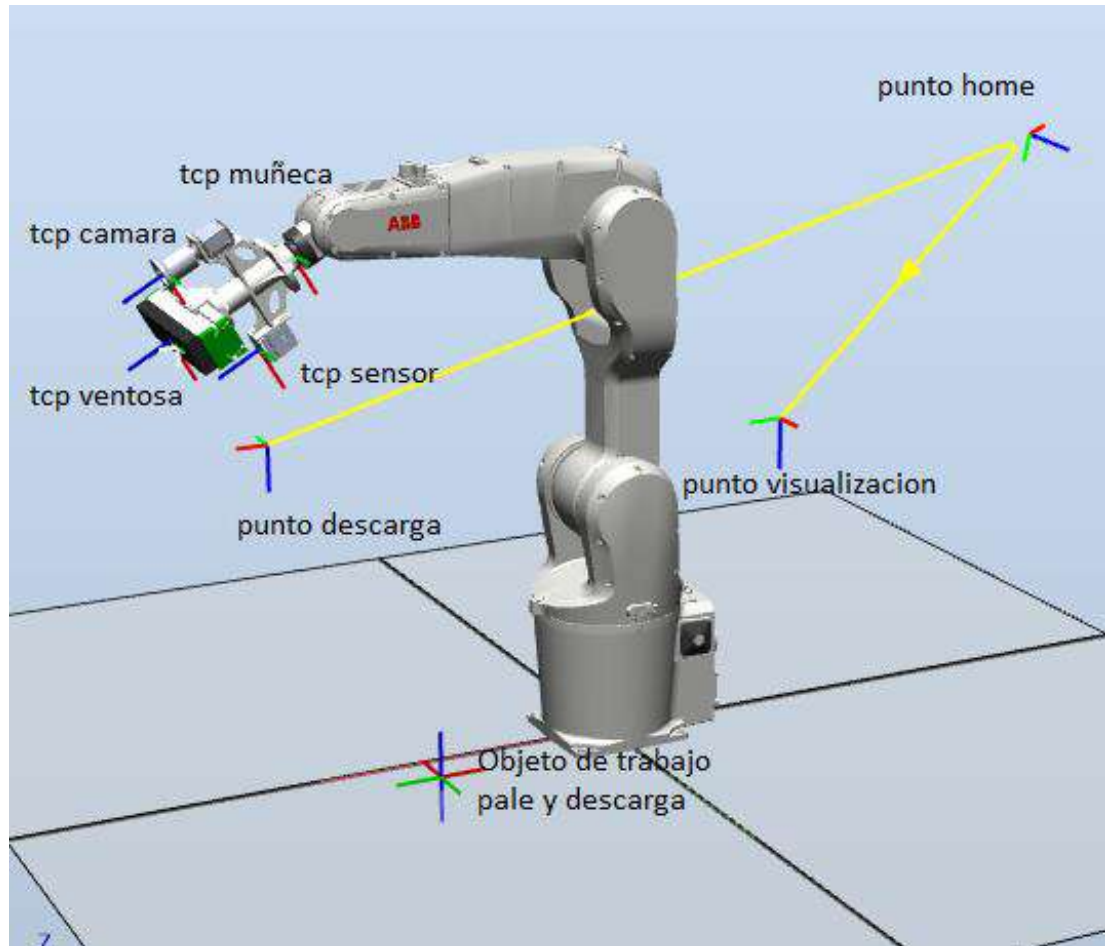


Ilustración 48. Estación virtual

7.3 Software

El programa se diseñó en base al Diagrama 5, pero únicamente teniendo en cuenta los procesos a realizar, sin tener en cuenta su secuencia, siendo el PLC el encargado de comunicarle qué procedimiento realizar en cada momento. El programa se estructuró de la siguiente forma:

- Módulo CalibData. Se especifican los objetos de trabajo utilizados: el del pale y el de la zona de descarga. Se definen las herramientas con los nombres ventosa, cámara y sensor, todas con las mismas especificaciones de masa e inercia, ya que forman parte de la misma estructura, pero con TCP diferentes. Se declaran 2 tipos de velocidades, una para movimientos rápidos y otro para lentos, y el parámetro *zonedata*. También se definen las variables tiempo de espera, usada para que el robot mantenga determinadas posiciones cierto tiempo, y margen, que se usa para hacer una aproximación a las piezas y al punto de descarga con mayor precisión.
- Módulo MainModule. Modulo que contiene al procedimiento principal.
 - Procedimiento *main*. Procedimiento principal cíclico en el que se decide qué procedimientos ejecutar.
- Módulo Points. Se especifican los *robtargets* utilizados.

- Módulo Funciones. Se definen las variables en las que son guardados los datos de posición y orientación del palé y de la pieza a extraer y se agrupan las siguientes rutinas:
 - Coger_pieza.
 - Dejar_pieza.
 - Fin.
 - Iniciar.
 - Ir_centro_pieza.
 - Mover_pieza.
 - Posicionar_cámara.
 - Posicionar_sensor.
 - Posicionar_sensor_pale_ini.
 - Función leerInt32.

7.3.1 Procedimiento Main

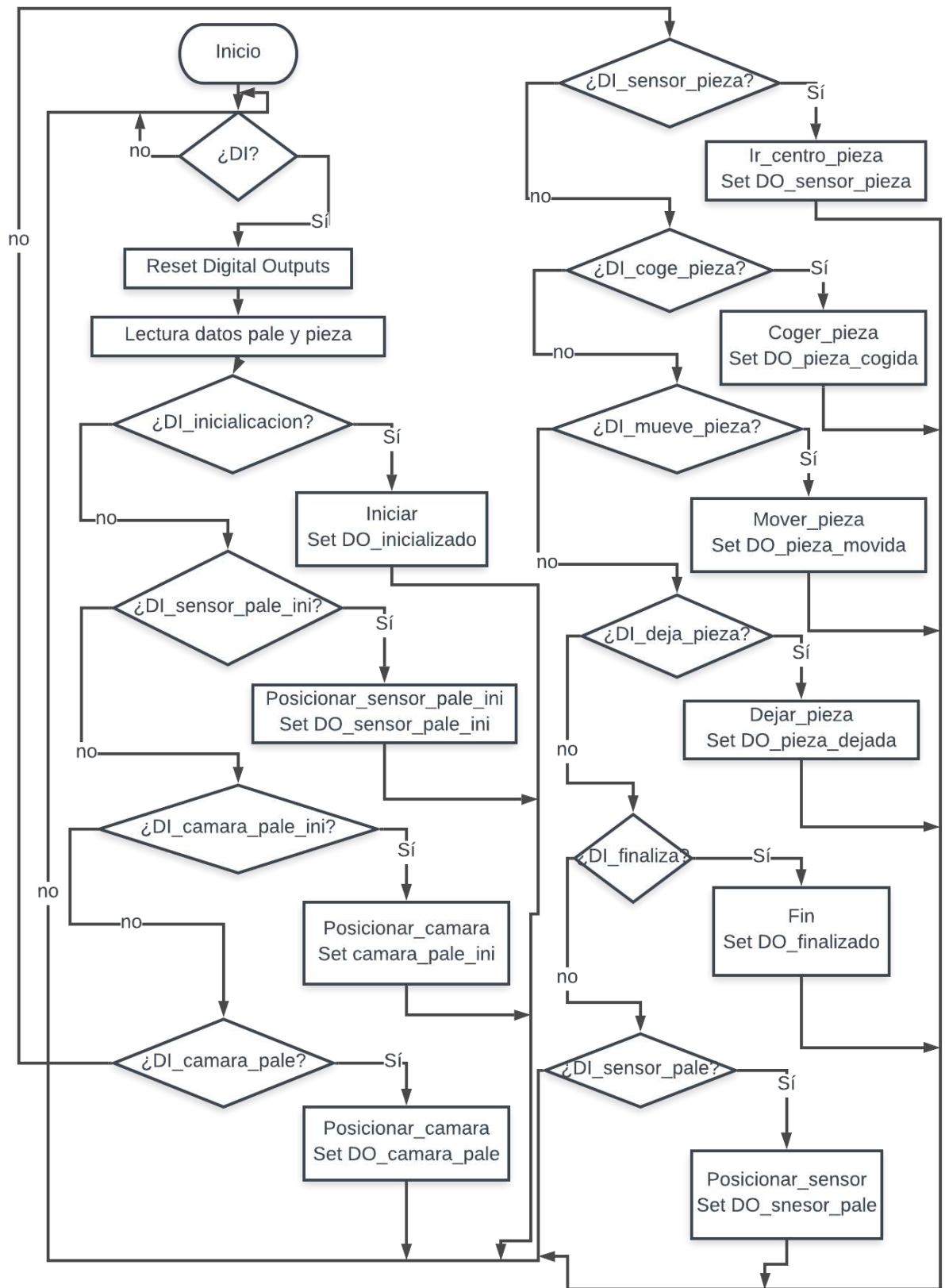


Diagrama 11. Main Robot

El procedimiento del Diagrama 11 es el único del módulo MainModule. Es el procedimiento principal y se ejecuta de forma cíclica. Comprueba si el PLC le transmite alguna orden. De ser

así, se guardan en variables el estado de las entradas por si estas variasen en medio de un ciclo, y se asigna el valor falso a todas las salidas digitales, que son las que le dan la información al PLC del estado del Robot. Después, actualiza el valor de las variables en las que se guarda la información del palé y de las piezas transmitida por el PLC usando la función leerInt32 (esto se explica más en detalle en el apartado de Comunicaciones). Con los datos actualizados, se comprueba qué orden transmitió el PLC, ejecutando la función correspondiente y asignando al acabar el valor verdadero a la salida digital que le da la información al PLC sobre el estado del robot.

7.3.2 Objetos de trabajo y puntos

Para la realización de los movimientos de despaletizado se definieron 4 puntos organizados en los siguientes objetos de trabajo:

- wobj0. Es un objeto de trabajo configurado por el fabricante del robot. Los demás objetos de trabajo están referidos a este. Se usa para el siguiente punto:
 - home. Se usa en el procedimiento Iniciar y Fin para posicionar el robot en un punto en el que permita colocar y retirar el palé.
- pale. Es usado para los puntos en los que el robot se posiciona sobre el palé o retira piezas de este. Los puntos referidos a él son los siguientes:
 - visualización_ini. Es el punto en el que se posiciona el sensor antes de conocer la posición exacta del mismo. Es constante.
 - visualizacion. Es el punto situado sobre el centro del palé a la altura para la que la cámara está calibrada. Es un punto variable, ya que depende de la posición del palé y de su altura. Por ello, se va actualizando a lo largo del programa.
- descarga. Usado para hacer referencia a la zona en la que se depositan las piezas. El punto referido a él es el siguiente:
 - deja. Es el punto en el que se deben dejar las piezas.

7.3.3 Movimientos

Para la realización de los movimientos se usó la instrucción MoveJ, con una *speeddata* lo más rápida posible y con una tolerancia de zona *zonedata* lo más permisiva posible.

Se usó MoveL, que realiza un movimiento lineal, para la aproximación y alejamiento a la pieza y al punto de descarga. Estos se realizan en las funciones *coger_pieza*, *mover_pieza* y *dejar_pieza*. El robot posiciona la caja de vacío sobre el centro de la pieza a una distancia de este definida por la variable *margen* y con una especificación de *zonedata*. A partir de ese punto, el robot desciende de forma lineal con una tolerancia *fine*, que está definida por el fabricante y otorga la mayor precisión posible. En el proceso de alejamiento del punto se hace el movimiento contrario, y lo mismo ocurre en el proceso de depositar la pieza en la zona de descarga.

La especificación de zona *fine* también es usada para los movimientos en los que se posiciona la cámara para que el sistema de visión tome una imagen o en los que se posiciona el sensor para recoger su medida.

Aunque se primó el hacer movimientos rápidos para disminuir el tiempo de ciclo, fue necesario hacer esperas de tiempo especificado en la variable *tiempo_espera*. Esto fue necesario en los procedimientos en los que se posiciona el sensor y la cámara, esperando a que los datos recogidos por ellos se estabilicen antes de indicarle al PLC que el robot acabó la tarea, para evitar que el PLC ordene la recogida de información antes de que los sensores estén midiendo correctamente.

Los movimientos en los que se usa la información de posiciones procedentes del PLC se realizan usando la función RelTool. Esta función utiliza como parámetros un punto, 3 datos de coordenadas y 3 de orientaciones, realizando un movimiento relativo al punto, pero al contrario

que la instrucción Offs, realiza el movimiento relativo según los ejes del punto de referencia y no del objeto de trabajo. En la Ilustración 49 se aprecia con claridad dicha diferencia. Esto se hace porque los datos transmitidos por la cámara están referidos al centro de la imagen, que coincide con el TCP de la herramienta cámara.

En el procedimiento Posiconiar_camara se realiza este tipo de movimiento relativo al punto visualización, y se actualiza dicho punto con los datos del fin del movimiento, ya que ese punto será el nuevo punto sobre el que la cámara realice la visualización.

En los procedimientos Ir_centro_pieza, coger_pieza y mover_pieza, se usa esta misma instrucción relativa al punto visualización, ya que los datos de las piezas han sido calculados por la cámara desde este punto, pero no se actualiza dicho punto, ya que todas las coordenadas calculadas por la pieza siguen siendo relativas a él.

En los demás procedimientos no se usaron movimientos relativos, sino que se usaron puntos constantes o el punto de visualización.

Por último, la instrucción Offs se usó para realizar las tareas de aproximación y alejamiento explicadas anteriormente en las que se usa la variable margen. Esto se hizo así puesto que el eje z del objeto de trabajo palé como descarga es vertical, por lo que realizando un movimiento relativo según dicho eje se consigue una aproximación y alejamiento normal al plano del palé y de descarga, independientemente de la orientación del punto.

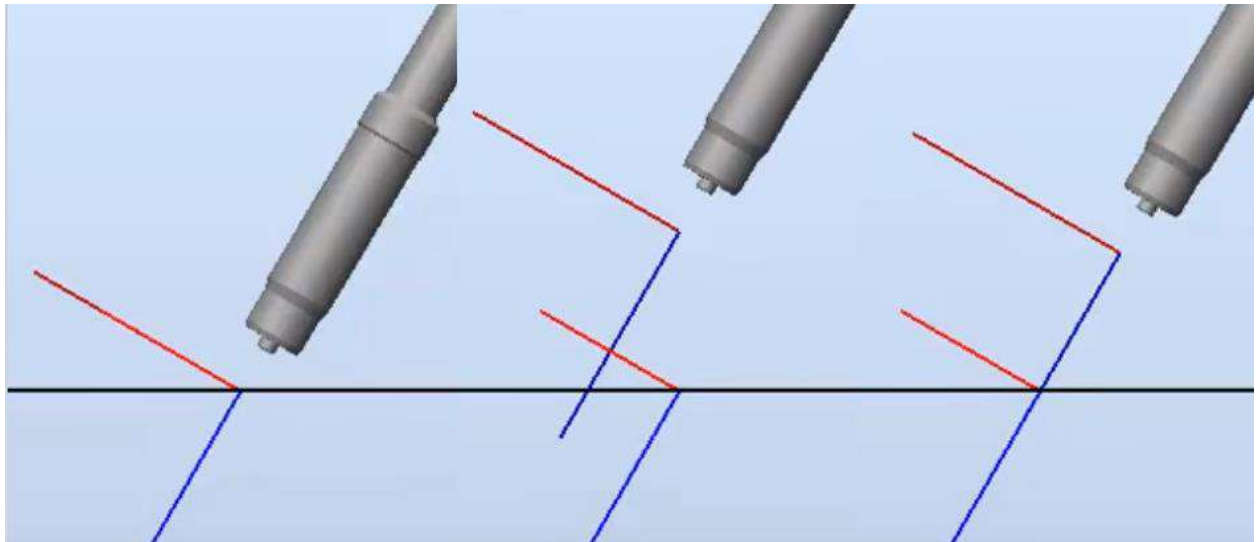


Ilustración 49. Diferencias entre Offs (centro) y Reltool (derecha) partiendo desde el mismo punto (izquierda) (56)

7.4 Calibraciones

Como se explicó anteriormente, el CAD de la herramienta se introdujo en la estación de RobotStudio, definiéndola como herramienta del robot, indicando su masa, sus constantes de inercia con respecto a los ejes de la muñeca del robot y los TCP de las 3 herramientas que forman la garra, y se cargaron los datos en la controladora del robot, quedando definidas 3 herramientas, con características físicas idénticas, pero con TCP diferentes.

Las características físicas afectan al comportamiento dinámico del robot y son suficientemente precisas para el funcionamiento de esta aplicación. Sin embargo, la definición de los TCP no lo es. Esto suele ser habitual en robótica, siendo necesario realizar una calibración. La controladora permite hacer estas calibraciones guiadas por el FlexPendant. Para definir un TCP de una herramienta desde el FlexPendant, es necesario registrar 4 configuraciones del robot, posicionando el punto que se desea definir como punto del TCP en el mismo lugar, pero con orientaciones diferentes. El punto en el que se posicionaron las herramientas fue marcado por la punta de un cable recubierto para protegerlas. En el caso de la cámara, para posicionar el centro

de esta sobre el cable, se usó el Halcon para ver las imágenes tomadas por la cámara y dibujar una X en el centro, como se muestra en la Ilustración 50. En la calibración del sensor se procedió de la misma forma, atendiendo al laser emitido por el sensor, tal y como se aprecia en la Ilustración 51. Para la garra de vacío se marcó su centro, como se ve en la Ilustración 52.

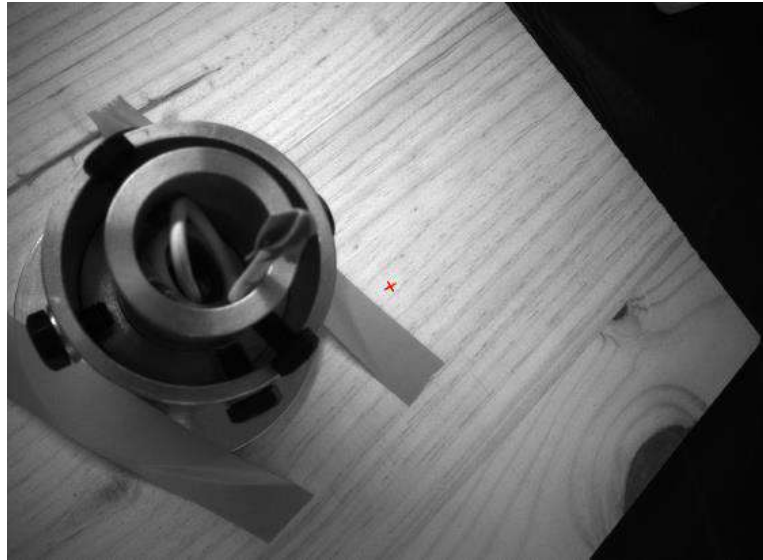


Ilustración 50. Calibración herramienta cámara



Ilustración 52. Calibración herramienta garra de vacío



Ilustración 51. Calibración herramienta sensor

Realizadas las calibraciones mediante el *flexpendant* quedan definidos los puntos del TCP de cada herramienta. No obstante, la orientación de estos es la misma que la de la muñeca. Para simplificar la información transmitida entre el sistema de visión y la cámara y el funcionamiento

del robot, se decidió que las 3 herramientas tuviesen la misma orientación y fuesen coincidente con el sistema de visión. Para ello, se posicionó la cámara de forma perpendicular a la mesa y se dibujaron en Halcon sobre la imagen capturada 3 puntos que indicaban la dirección de los ejes del sistema de visión. Sobre la mesa de la estación, observando las imágenes capturadas y los puntos, se representaron los ejes tal y como se observa en la Ilustración 53. Una vez representados, se posicionó el sensor sobre el punto en el que se encontraba la cámara y desde él se desplazó según los ejes del TCP. El eje z, que se define como saliente de las herramientas, era coincidente con el de visión, ya que la cámara se posicionó de forma perpendicular a la mesa. Sin embargo, al mover el sensor en la dirección x de su TCP, el láser no seguía los ejes representados. Observando el ángulo de desviación, se rotaron los TCP mediante instrucciones en lenguaje Rapid de las 3 herramientas hasta que los ejes de los TCP de las herramientas coincidiesen con los representados.

Se debe tener en cuenta que los ejes representados en la mesa son los del sistema de visión cuando la cámara se encuentra en el punto y orientación desde el que se tomó la imagen. Al variar la orientación de los TCP, si se le indica al robot que lleve la herramienta al punto en el que sacó la cámara la imagen, la cámara no tendría la misma orientación que inicialmente, puesto que para el robot la cámara ha sido girada, y los ejes del sistema de visión tendría unos ejes con direcciones diferentes a los representados. Por ello, antes de girar los TCP, se posicionaba una herramienta sobre el punto desde el que la cámara tomó la imagen, después se giraban los TCP, y sin haber movido el robot se actualizaba el punto indicándole que era en el que se encontraba el robot.



Ilustración 53. Calibración ejes herramientas

8 VISIÓN ARTIFICIAL

A continuación, se introduce la herramienta de programación Halcon utilizada, explicando su entorno. También se explica el funcionamiento del software desarrollado y los algoritmos utilizados, así como la calibración.

8.1 Halcon

Como se mencionó en el capítulo de herramientas, Halcon es un programa de visión artificial que permite trabajar escribiendo código, pero también con herramientas aplicadas directamente sobre imágenes. El entorno está compuesto por 4 ventanas:

- Ventana gráfica. Permite mostrar los resultados de aplicar determinadas funciones sobre una imagen, pudiendo así indicar la posición de puntos calculados, el contorno de figuras, y demás funciones útiles para el desarrollador del programa y para el operario, pudiendo mostrar la imagen en un HMI.
- Ventana de operador. Esta ventana se usa para buscar las funciones de la librería. Da información sobre la utilidad de la función y sobre el tipo de parámetros de entrada y de salida.
- Ventana de Variables. En ella se muestran todas las variables definidas en el programa con su valor. Se organizan en 2 tipos:
 - Icónicas. Son imágenes digitales, ya sean capturadas por la cámara, leídas de una dirección o el resultado de aplicarle una función a variable icónica. Cada imagen se trata como un vector, y cada elemento del vector son los contornos, puntos o regiones detectados al aplicarle una determinada función a una imagen. La ventana permite seleccionar una imagen y visualizarla en la ventana gráfica, independientemente de la ejecución del código, permitiendo superponer variables. Esto es muy útil durante el desarrollo del programa, ya que permite ver las diferencias entre los algoritmos utilizados de una forma visual rápidamente.
 - De control. Son variables, de tipo booleano, números reales, vectores, caracteres o listas formadas por combinaciones de ellas.
- Ventana de Programa. Usada para la escritura de código. Organiza el código en archivos independientes, siendo uno de ellos el principal, que es en el que comienza la ejecución del programa, y teniendo uno para cada función de la librería, existiendo la posibilidad de crear funciones nuevas por el programador.

8.2 Algoritmo de reconocimiento

El primer objetivo que se abordó fue el de conseguir el reconocimiento de las piezas. Como aún no se disponía de la cámara para este al inicio del desarrollo del programa, se usó otra disponible que se encontraba en fábrica para hacer pruebas, de la misma marca y modelo, pero a color. La cámara no se encontraba en el lugar de trabajo del desarrollador, ni estaba disponible exclusivamente para este proyecto. Por ello, se desplazaron tablas de diferentes tamaños hasta el lugar en el que se encontraba la cámara, y se tomaron imágenes guardándolas para poder trabajar con ellas, con diferente iluminación y sobre distintos fondos.

Se hicieron pruebas con diferentes sistemas de iluminación, usando también difusores, hasta tomar la decisión de escoger el Effi-flex por ser el que conseguía obtener una imagen con una iluminación homogénea.

Se probó a posicionar las tablas sobre un fondo blanco y un fondo negro, obteniendo mejores resultados con las imágenes sobre el fondo negro como se puede apreciar en la Ilustración 54 e Ilustración 55 ya que existe mayor contraste entre los objetos a reconocer y el fondo.

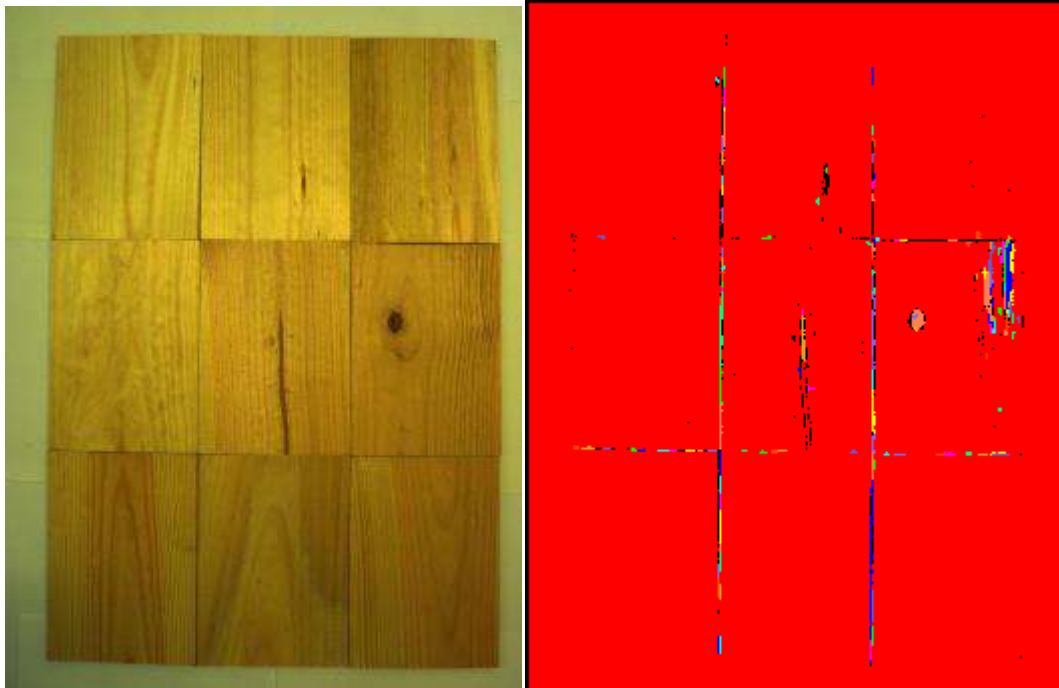


Ilustración 54. Tablas sobre fondo blanco

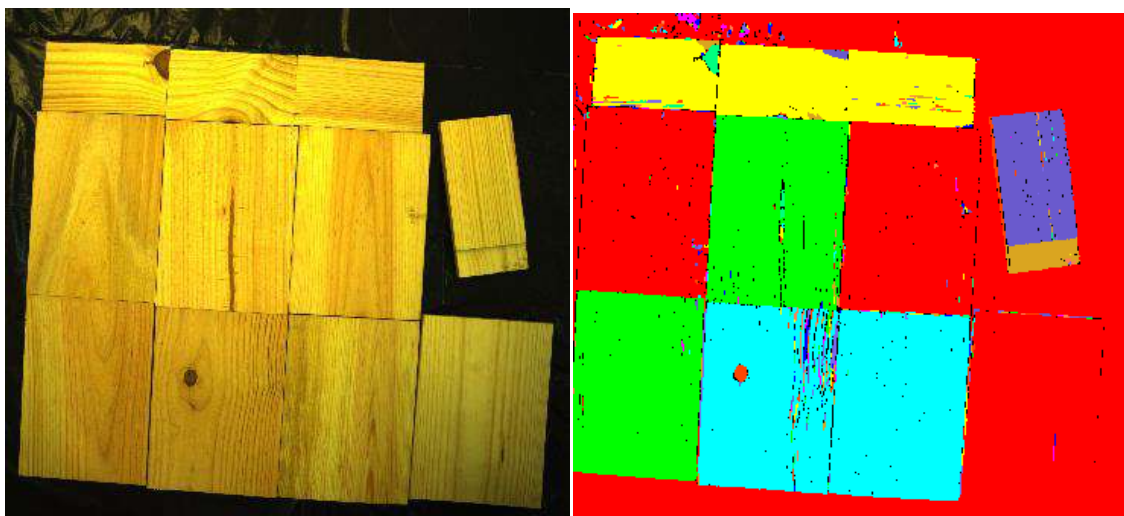


Ilustración 55. Tablas sobre fondo negro

Para la elección del algoritmo de reconocimiento, se hicieron pruebas con los siguientes para reconocer las tablas de la Ilustración 56, tomada desde la cámara a color disponible en fábrica convertida a una imagen en gris:



Ilustración 56. Imagen de pruebas

8.2.1 Canny (57) (58)

El algoritmo de Canny fue desarrollado por John F. Canny en 1986. Es un operador que utiliza múltiples etapas usado para detectar bordes, siendo sus objetivos los siguientes:

- Detectar el mayor número de puntos reales en los bordes.
- Localizar los puntos detectados lo más próximamente posible a los puntos reales que forman un borde.
- Detectar únicamente un punto por cada punto real que forma el borde (respuesta mínima).

Para ello, el algoritmo de Canny se compone de las siguientes 5 etapas:

1. Reducción de ruido. Para ello, se aplica un *kernel* Gaussiano. Los elementos de este *kernel* siguen una distribución gaussiana. El efecto es un suavizado sobre la imagen, como si esta estuviese desenfocada.
2. Encontrar gradientes. El algoritmo de Canny busca los puntos en los que el nivel de gris varía bruscamente. Para ello utiliza el operador de Sobel. Consiste en aplicar los *kernel* de la Ecuación 1 y Ecuación 2 para determinar la variación de nivel de gris en el eje horizontal y en el vertical.

$$G_x = \begin{matrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{matrix}$$

Ecuación 1. Kernel Sobel horizontal

$$G_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{matrix}$$

Ecuación 2. Kernel Sobel vertical

A partir de los datos obtenidos en cada píxel, se calcula el valor del gradiente y su dirección en cada punto con la Ecuación 3 y Ecuación 4.

$$G = \sqrt{(G_x + G_y)}$$

Ecuación 3. Valor gradiente Sobel

$$\Phi = \arctan \left(\frac{|G_x|}{|G_y|} \right)$$

Ecuación 4. Dirección gradiente Sobel

3. Eliminación de los valores no máximos. Trata de reducir el grosor de los contornos generados seleccionando los puntos en los que el gradiente es máximo en sus proximidades y eliminando los demás. Para ello, realiza el siguiente algoritmo:
 1. Se establecen una serie de direcciones y se redondean las direcciones de los gradientes calculados a la más próxima.
 2. Compara el gradiente con sus vecinos de la misma dirección.
 3. Si el gradiente es el mayor de sus vecinos con el mismo sentido, lo conserva, si no, lo elimina.
4. División por doble umbral. Con el fin de eliminar ruido, se establecen 2 valores umbrales dividiendo los contornos en 2. Los puntos mayores que el mayor umbral se marcan como fuertes, los que únicamente superan el menor se marcan como débiles, y los que no superan ninguno se eliminan.
5. Seguimiento de los bordes por histéresis. Los bordes formados por puntos marcados como fuertes determinan bordes reales. Los formados por puntos marcados como débiles, únicamente se seleccionan si están unidos a un borde formado por puntos fuertes, sino se eliminan.

Se intentó reconocer las tablas de la Ilustración 56 variando los parámetros del filtro Gaussiano y los valores umbrales para la división entre puntos fuertes y débiles. Los contornos detectados eran más que los reales, habiendo varios representando al mismo de forma segmentada, además de haber algunos que pertenecían a manchas en las tablas, siendo estos los de menor tamaño y estando aislados. Por ello, se unieron los contornos próximos y se eliminaron los de menor tamaño. El mejor resultado que se pudo obtener con este algoritmo es el de la Ilustración 57. De forma visual, se reconoce el lugar de cada pieza, pero no es posible desarrollar un programa robusto que detecte sus centros, ya que, salvo la tabla inferior derecha, no están delimitas por contornos cerrados. Esto implicaría que, en cada detección, las tablas estarían representadas por varios contornos de tamaños variados y sin cerrar, siendo complicado la selección de forma automatizada a que pieza corresponde a cada contorno, e indicarle como unirlos para calcular su centro.

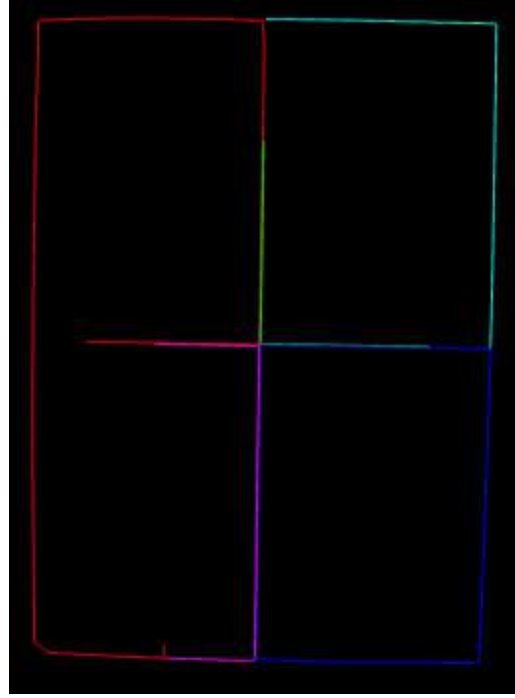


Ilustración 57. Tablas Canny

8.2.2 *Deriche* (59)

El siguiente algoritmo que se probó fue el de Deriche, desarrollado por Rachid Deriche en 1987. Este algoritmo está basado en el anterior, diferenciándose en la primera y segunda etapa. En la primera etapa, aplica un filtro de respuesta de impulsos infinitos (IIR (60)), según la Ecuación 5. La ventaja con respecto a Canny es que únicamente se depende del parámetro α .

$$f(x) = Sxe^{-\alpha|x|}$$

Ecuación 5. IIR

En la segunda etapa, se calculan los gradientes primero haciendo un reconocimiento horizontal y después uno vertical. En el reconocimiento horizontal, se aplica la Ecuación 6, recorriendo los píxeles de izquierda a derecha, y la Ecuación 7 de derecha a izquierda. Con los resultados se calcula el parámetro Φ según la Ecuación 8. A continuación, se realiza el mismo procedimiento, pero en dirección vertical, utilizando como entradas los resultados del vector Φ . Los coeficientes del filtro deben ser coherentes con la Tabla 1.

$$y_{ij}^1 = a_1x_{ij} + a_2x_{ij} + b_1y_{ij-1}^1 + b_2y_{ij-1}^1$$

Ecuación 6. Deriche izquierda-derecha búsqueda

$$y_{ij}^2 = a_3x_{ij} + a_4x_{ij} + b_1y_{ij+1}^1 + b_2y_{ij+1}^1$$

Ecuación 7. Deriche derecha-izquierda búsqueda

$$\Phi_{ij} = c_1(y_{ij}^1 + y_{ij}^2)$$

Ecuación 8. Deriche resultado horizontal

	smoothing	x-derivative	y-derivative
k	$\frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}}$	$\frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}}$	$\frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}}$
a_1	k	0	k
a_2	$ke^{-\alpha}(\alpha - 1)$	1	$ke^{-\alpha}(\alpha - 1)$
a_3	$ke^{-\alpha}(\alpha + 1)$	-1	$ke^{-\alpha}(\alpha + 1)$
a_4	$-ke^{-2\alpha}$	0	$-ke^{-2\alpha}$
a_5	k	k	0
a_6	$ke^{-\alpha}(\alpha - 1)$	$ke^{-\alpha}(\alpha - 1)$	1
a_7	$ke^{-\alpha}(\alpha + 1)$	$ke^{-\alpha}(\alpha + 1)$	-1
a_8	$-ke^{-2\alpha}$	$-ke^{-2\alpha}$	0
b_1	$2e^{-\alpha}$	$2e^{-\alpha}$	$2e^{-\alpha}$
b_2	$-e^{-2\alpha}$	$-e^{-2\alpha}$	$-e^{-2\alpha}$
c_1	1	$-(1 - e^{-\alpha})^2$	1
c_2	1	1	$-(1 - e^{-\alpha})^2$

Tabla 1. Coeficientes Deriche

El resultado, como se puede apreciar en la Ilustración 58, a diferencia del obtenido con el filtro de Canny, hay más partes de los contornos reales detectados, pero a costa de ser más cortos y tener partes superpuestas entre ellos. Además, sigue sin ser suficiente por las mismas razones que con el filtro de Canny.

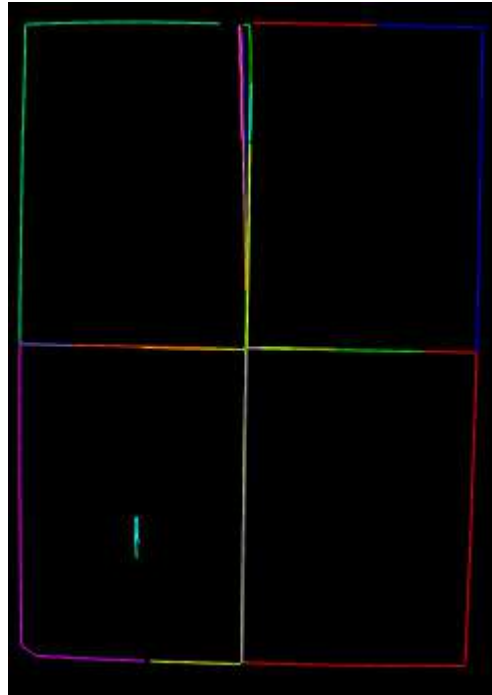


Ilustración 58. Tablas Deriche

8.2.3 *Regiongrowing Segmentation (61) (62)*

Tras haber probado con métodos para localizar los bordes, se decidió probar otro tipo de método basado en segmentación llamado Regiongrowing Segmentation. Este algoritmo utiliza la información del histograma de grises, pero utilizando también la ubicación de sus puntos. De esta forma, es capaz de unir píxeles semejantes. Su funcionamiento es el siguiente:

1. Partición de la imagen en semillas. Las semillas son los puntos tomados a partir de los cuales se formarán las regiones. La selección de estos puntos puede ser crítica para la optimización del tiempo de cálculo. Habitualmente se suelen usar los puntos con valor de gris mayores.
2. Ajustar un modelo plano a cada región de cada semilla.
3. En cada región comprueba la compatibilidad de los puntos vecinos a la región para tomar la decisión de añadirlos o no.
4. Si no hay puntos compatibles, se aumentan los puntos con los que se compara y se repite el paso 3 hasta un máximo de veces estipulado.
5. Si hay 2 regiones vecinas, se calculan sus parámetros.
6. Se comparan los errores de similitud entre las 2 regiones vecinas y el error de la nueva región global.
7. Si la diferencia entre los errores es menor que cierto valor se vuelve al paso 3.
8. Si no, se aumentan los puntos vecinos con los que se hace la comprobación.
9. Se recalcula la región con los parámetros modificados. Si su error decrece, se acepta el nuevo modelo y se sigue desde el paso 3. Si no, la región deja de crecer.

Tras hacer varias pruebas con diferentes parámetros relativos al tamaño mínimo de cada región y la tolerancia entre píxeles pertenecientes a la misma región, se consiguió reconocer las tablas como se puede ver en la Ilustración 59. Con estas regiones sí se puede sacar la información de su centro y orientación de una forma automatizada.

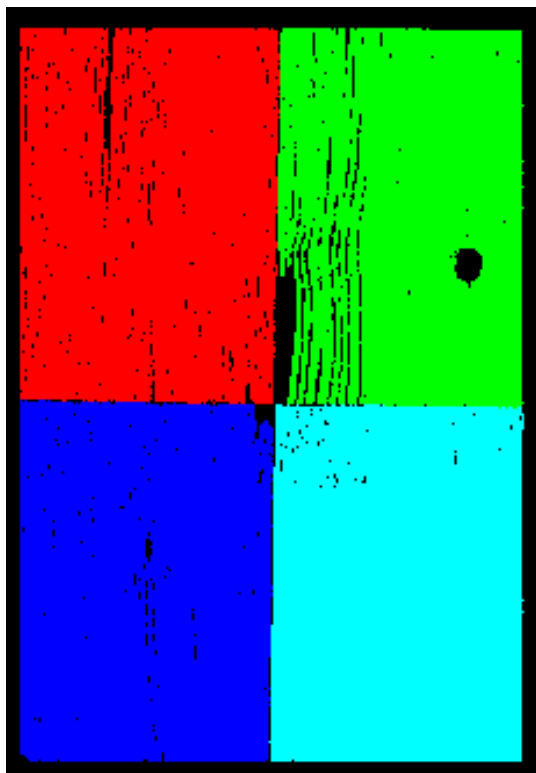


Ilustración 59. Tablas regiongrowing

8.3 Ajuste de la cámara

La cámara, junto con la lente, debe ajustarse a la casuística de la estación de despaletizado. Para ello, se pueden modificar tanto parámetros físicos como computacionales.

Los parámetros físicos son los relativos a la lente y son 2: apertura de iris y enfoque. El iris, al igual que en un ojo, determina la cantidad de luz que permite que llegue al sensor de imagen. El enfoque ajusta la imagen para que sea nítida a la distancia del plano que se desea observar.

Los parámetros computacionales se pueden ajustar desde el software del fabricante de la cámara o directamente desde el Halcon. Estos parámetros son los relativos al funcionamiento del sensor de imagen. El más importante, por lo menos para esta aplicación, es el tiempo de exposición. Este tiempo es el que el sensor está captando la luz para capturar una imagen. Cuanto mayor sea, más luz incide sobre él y más brillante es la imagen. Su efecto sobre las imágenes es similar al del iris, pero no exactamente igual. Ambos afectan a la luminosidad, pero el iris controla qué luz llega al sensor, restringiendo a medida que se cierra la luz más periférica. El tiempo de exposición no controla que luz capta el sensor, sino solamente la cantidad de esta. Las recomendaciones de los fabricantes para obtener imágenes de mayor calidad es cerrar lo máximo posible el iris para focalizar la luz recibida y ajustar el tiempo de exposición para conseguir una imagen bien iluminada sin saturación.

8.4 Software

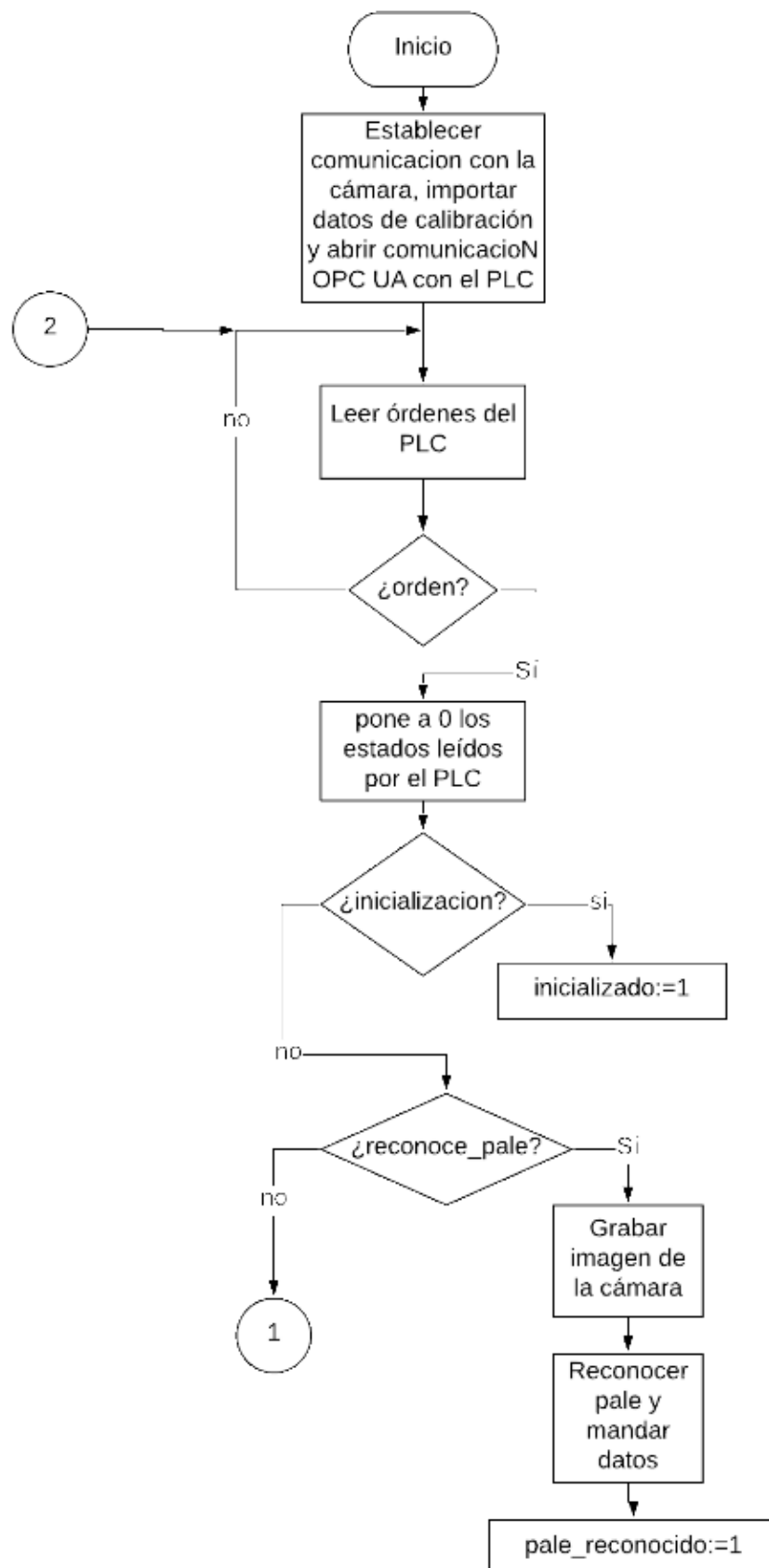


Diagrama 12. Visión artificial PC (1)

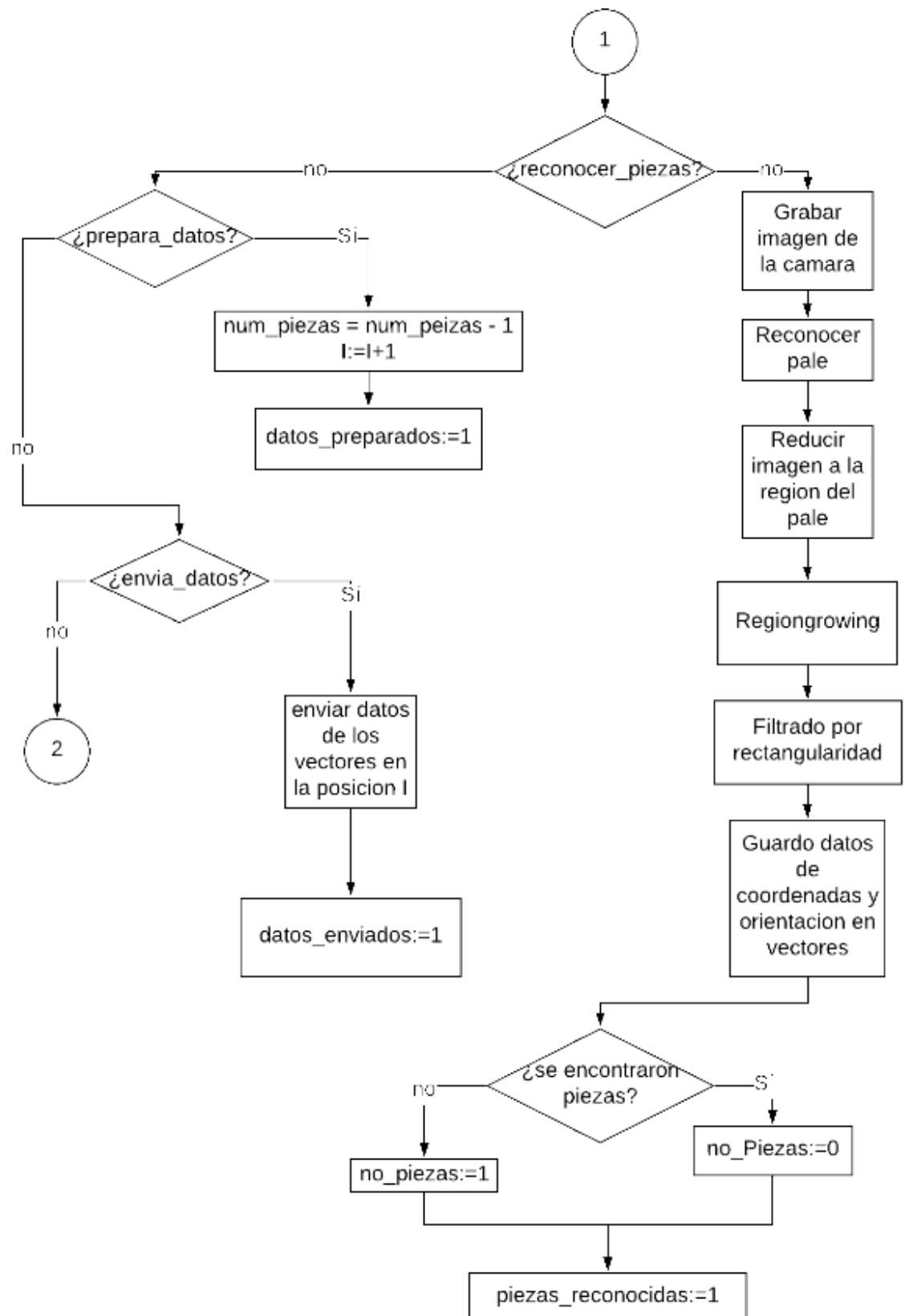


Diagrama 13. Visión artificial PC (2)

Como se explica en el Diagrama 12 y Diagrama 13, el sistema de visión artificial, al igual que el robot, comprueba cíclicamente las órdenes recibidas por el PLC para decidir qué código ejecutar e indicándole cual es la última orden que ha realizado.

Si recibe la orden `reconoce_pale`, realiza un filtrado por umbral de grises, binarizando entre pale y fondo, ya que, al ser el fondo negro y las tablas marrones, en la imagen sin color existe gran contraste. Se podría reconocer con el mismo método que se usa para reconocer las tablas, pero filtrar por umbral es suficiente y más rápido. Para conseguir que el sistema sea más preciso, se hace una apertura a la región del palé, que consiste en un erosionado seguido de una dilatación. Ambas operaciones se hacen siguiendo direcciones rectangulares, ya que se sabe que el palé tiene esta forma.

Al detectar la orden `reconocer_piezas`, se reconoce el palé igualmente, pero no para enviarle la información al PLC, sino para reducir el dominio de la imagen capturada por la cámara a únicamente la zona del palé. De esta forma, el sistema se vuelve mucho más robusto al eliminar el ruido perteneciente al fondo. Seguidamente, se ejecuta el algoritmo *regiongrowing* sobre las tablas. Para evitar posibles errores en la detección, se eliminan las regiones que no tienen una forma rectangular, y las que sí lo tienen se sustituyen por el menor rectángulo que contenga toda la región, para obtener mayor precisión de su pose.

Los datos de las tablas se guardan en vectores pertenecientes a las coordenadas X, coordenadas Y y orientación con respecto al eje Y, estableciendo el origen de coordenadas en el centro de la imagen, coincidente con la posición del TCP de la cámara en el momento de tomar la imagen. Los datos de estos vectores se van enviando por orden al PLC a medida que se van solicitando con las órdenes `envía_datos` y `prepara_datos`.

8.5 Calibración

El objetivo de calibrar el sistema de visión es poder extraer medidas reales a partir de imágenes digitales. Halcon está equipado con un asistente que facilita la calibración, obteniendo como resultado unos archivos que realizan la conversión de píxeles a medidas reales siempre y cuando la posición relativa de la cámara con respecto al plano de visión sea la misma que la de la calibración. Para ello, se debe posicionar la cámara en el lugar de trabajo, y el plato de calibración sobre el plano de visión, y se toman imágenes variando la posición de dicho plato.

Sin embargo, esto no fue posible de hacer debido a que la iluminación no es suficientemente homogénea como para que el asistente haga la calibración con la precisión requerida por Halcon. Por ello, se decidió calcular el área en píxeles de un objeto de dimensiones conocidas con diferentes orientaciones y en varias ubicaciones para conocer la relación entre píxeles y mm, como se observa en la Ilustración 60, obteniendo una desviación de la media entre los valores píxel/mm calculados menor de 1 píxel/mm, lo que indica que, al ser 1 píxel la medida mínima del sistema de visión, la relación píxel/mm es constante en todo el plano de visión.



Ilustración 60. Cálculo relación pixel/mm

9 HMI

Como no se disponía de un HMI, se simuló en el TiaPortal. El HMI seleccionado fue el TP1500 Comfort de Siemens. Se configuraron los botones de automático, paro y mantenimiento para la elección del modo de funcionamiento, así como un indicador de la variable modo. También se configuraron los pulsadores inicialización y palé colocado para poder simular el proceso.

A mayores, se añadió un indicador de alarma, un botón de alarma y un pulsador de ACK. Estos serán utilizados en el programa de seguridad del PLC necesario para implementar la estación en fábrica.

Para poder simular el HMI desde el TiaPortal y que este se comunicase con el PLC, fue necesario ajustar la interfaz PC/PG del ordenador, seleccionando en la tarjeta de red por la que el ordenador se comunica con el PLC la opción de permitir comunicación con Step7online.

10 COMUNICACIONES

Las comunicaciones entre los diferentes dispositivos por bus se configuraron utilizando direcciones IP. Una dirección IP es un número que identifica a una interfaz de red de comunicaciones digitales. Este número está formado por un número de 32 bits y tienen 2 funcionalidades: identificar la red e identificar el anfitrión o *host*. La red es el conjunto de los equipos conectados, siendo posible la comunicación entre ellos si todos se encuentran en la misma red. Los anfitriones son cada uno de los dispositivos que son partícipes de las comunicaciones en dicha red, y su número de identificación debe ser único en la red en la que se comunica. El número de bits de la dirección IP destinada a cada funcionalidad viene determinado por la máscara de red, que es un número propio de cada tipo de red (63).

Se consultó la red usada en Finsa para la comunicación en fábrica y qué direcciones estaban libres, asignando las siguientes a cada dispositivo, siendo la máscara utilizada 255.255.248.0:

- PLC: 172.16.131.228
- PC: 172.16.131.229
- Controladora del robot: 172.16.131.230
- Periferia distribuida: 172.16.131.233
- HMI: 172.16.131.235
- Cámara: 172.16.131.236

10.1 Cámara-PC visión

La cámara iDS UI-5200SE Rev.4 se conectó mediante cable Ethernet al PC, y usando el programa del fabricante iDS camera Manager, se configuró la conexión para poder ser detectada desde el Halcon. Seguidamente, en Halcon, usando el asistente de adquisición de imagen, se detectó la cámara y se ajustaron sus parámetros, generando el código necesario para obtener imágenes digitales cíclicamente (64).

10.2 Robot – PLC

La controladora del robot utiliza el estándar Profinet para realizar las comunicaciones por bus. El protocolo envía y recibe un número determinado de bits de la red. Estos bits están ordenados para poder ser identificados, y es desde las interfaces desde donde se realizan las vinculaciones entre ellos y sus variables. Por ello, fue necesario hacer las configuraciones pertinentes en el PLC y en la controladora.

10.2.1 Robot

Las configuraciones en la controladora se hicieron en RobotStudio, en las configuraciones de entradas y salidas, especificando el puerto físico de la controladora por el que se conecta el bus. Se pueden configurar hasta 4 tarjetas en el bus. Una de entradas, otra de salidas, una de entradas de seguridad y otra de salidas de seguridad. Se decidió configurar 2 tarjetas: una de entradas y otra de salidas, de 128 bytes cada una, mapeando las variables utilizadas. En el mapeado de variables se indica qué bits pertenece a cada variable. RobotStudio permite declarar 3 tipos de variables: digital, analógica y grupo. Las variables de tipo digital únicamente usan un bit, por ello fueron usadas para transmitir booleanos. Las analógicas usan varios bits para formar un número real, pero sin la posibilidad de especificar el signo. Las de tipo grupo utilizan varios bits, formando un número real. Se le debe especificar la forma en leer los bits, haciéndolo cuadrar con

la misma forma en el que el PLC transmite y lee. Este tipo fue el utilizado para transmitir los datos de coordenadas y giros.

Para transmitir los datos de coordenada, se mapearon variables de 4 bytes de tamaño. El PLC envía el dato en mm en el caso de coordenadas y grados en caso de orientación multiplicados por 100 para poder transmitir decimales. Este número no cabe en una variable de tipo *num*, por lo que se guarda en un *dnum*. Además, únicamente lee números positivos de 0 a 4294967296, mientras que el PLC transmite entre -2147483647 y 2147483647, de modo que un número negativo lo interpreta como su valor absoluto sumado a 2147483647. Por ello, fue necesario hacer una función para solventar el problema, dividir el resultado entre 100 para trabajar en mm y grados y guardarlos en una variable de tipo *num*, ya que son las que exigen las funciones de movimientos propias del robot.

10.2.2 PLC

Para poder reconocer la controladora del robot desde el PLC fue necesario importar en el proyecto el archivo GSDML de la controladora. Un archivo GSDML es un archivo de descripción general de una estación (GSD) escrito en XML. Los archivos GSD son descripciones de dispositivos IO proporcionados por sus fabricantes necesarios para trabajar de forma compatible entre dispositivos diferentes (65).

Una vez importado el archivo GSDML en el TiaPortal, se cargó en el proyecto como dispositivo hardware, haciendo sus pertinentes configuraciones. Al igual que en la controladora del robot, permite configurar las tarjetas de comunicación del bus. Se añadieron las mismas tarjetas y del mismo tamaño que en la controladora. Para hacer la lectura y escritura, se crearon 2 tipos de variables: *Input_Robot* y *Output_Robot*. Estas variables están compuestas a su vez por variables booleanas y dobles enteros. *Input_Robot* y *Output_Robot* son del mismo tamaño que la tarjeta de entrada y salida del GSDML, respectivamente, y las variables internas que forman *Input_Robot* coinciden con las de la tarjeta de salidas de la controladora y las de *Output_Robot* con la de entradas, definiendo como variable de reserva el espacio en la memoria que queda libre hasta completar el tamaño de las tarjetas.

De esta forma, al principio de cada ciclo en el PLC, se copia el estado de los bits de la tarjeta de entradas del GSDML en una variable de tipo *Input_Robot*. Durante el ciclo, se leen y se modifican las variables internas de *Input_Robot* y *Output_Robot*, respectivamente. Al final del ciclo, se iguala la tarjeta de salida del GSDML a la variable *Output_Robot*.

La tarjeta de entrada de GSDML representa la de salida de la controladora y viceversa.

10.3 PC visión – PLC

La comunicación entre el PLC y el PC de visión se hizo utilizando el estándar de comunicación de incrustación y enlazado de objetos para el proceso de control (OPC) en su versión UA. El estándar clásico OPC está basado en Microsoft DCOM y resuelve problemas de interoperabilidad, ofreciendo una interfaz común para comunicar datos entre componentes de software individuales. OPC UA añade a las características de OPC clásico un elevado grado de estandarización. (66)

Desde el entorno de Halcon, se creó un servidor OPC a partir de la IP del PLC y el puerto destinado para la comunicación OPC abriendo una conexión entre el PLC y el programa escrito en Halcon. De esta forma, se pueden establecer canales que permiten leer y escribir una variable, únicamente conociendo el nombre con el que está declarada y el bloque de datos en el que se encuentra, sin necesidad de conocer su ubicación en el mapa de memoria. El tiempo de ejecución para la apertura es elevado, por lo que se abren al inicio del programa y únicamente se escribe o se lee a través de ellos, reduciendo considerablemente los tiempos de ejecución.

El programa de visión artificial lee y escribe únicamente en las variables del DB OPC VISION. El PLC lee a principio del ciclo este DB y escribe en él al final. Además, las variables en las que escribe el PLC son las leídas por el sistema de visión y viceversa, no existiendo ninguna variable leída o escrita por ambos. De esta forma, se garantiza el correcto funcionamiento de la comunicación.

11 PRUEBAS Y RESULTADOS

Realizado el montaje de la estación y desarrollados los programas se procedió a realizar pruebas del funcionamiento, ajustando los parámetros necesarios y modificando la situación de los elementos que influyen en el funcionamiento del proceso.

11.1 Reconocimiento de tablas

El primer ajuste que se realizó fueron los necesarios para conseguir reconocer las tablas. La iluminación en una aplicación de visión artificial es crítica. Por ello, en vez de modificar el código de reconocimiento, se modificó la iluminación que llegaba a la cámara, puesto que se había comprobado el funcionamiento del algoritmo en su etapa de desarrollo, realizando pruebas en un ambiente con una buena iluminación.

El primer objetivo fue el de conseguir una iluminación homogénea. Para ello, se equiparon las barras led con el difusor opalino y se configuraron para que la luz se emitiese con una apertura de 90°. Además, se manipulando el brazo articulado de su soporte hasta lograr la iluminación deseada.

El segundo objetivo fue el de lograr resaltar al máximo la información deseada, y reducir la no deseada, es decir, conseguir un gran contraste entre las tablas y el fondo. Por ello, se decidió que el fondo fuese negro.

El primer fondo utilizado fue polietileno de baja densidad. Como se aprecia en la Ilustración 61, este material genera reflejos que dificultan el funcionamiento el correcto funcionamiento del algoritmo en su primera etapa, en la que realiza la diferenciación entre palé y fondo. En la Ilustración 63, se observa cómo parte de las tablas las reconoce como fondo, dificultando su reconocimiento como se ve en la Ilustración 62.

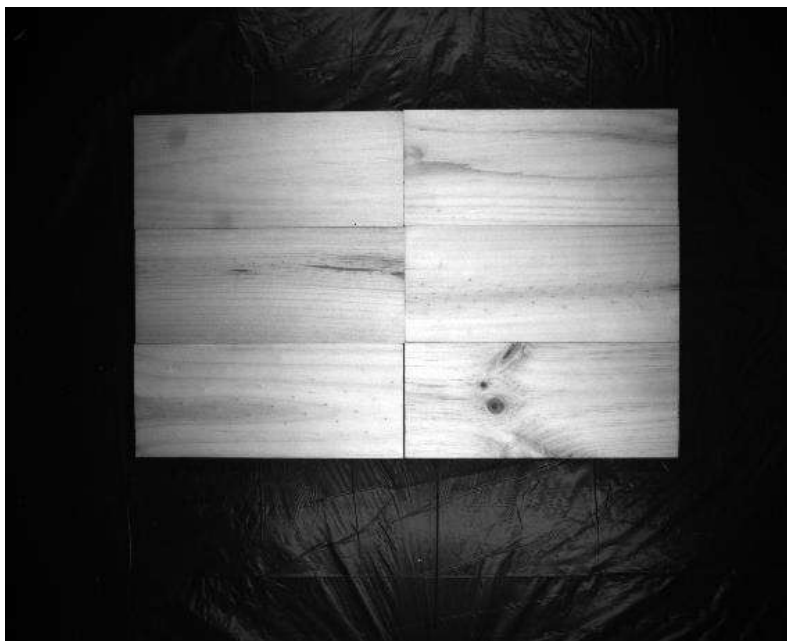


Ilustración 61. Tablas sobre plástico negro

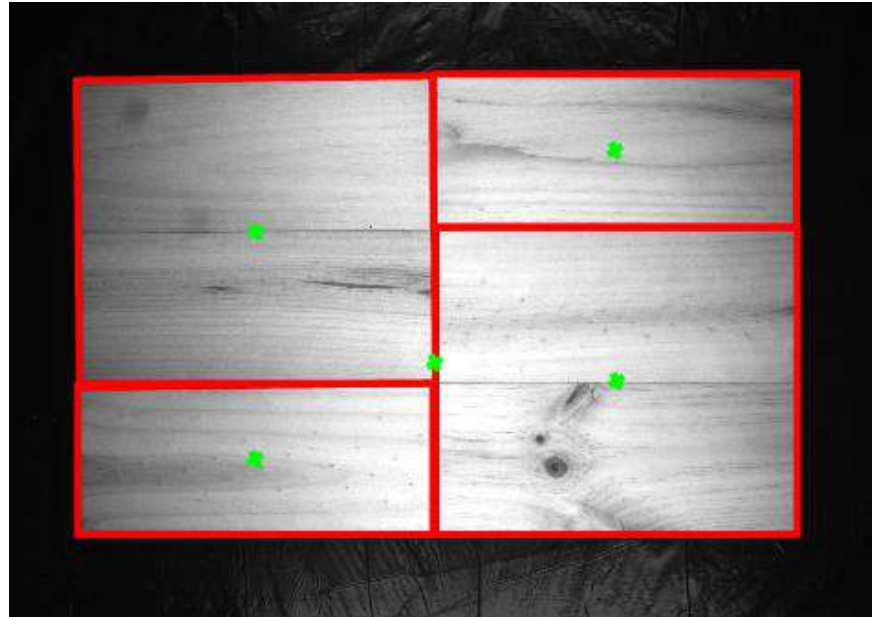


Ilustración 62. Reconocimiento de tablas sobre plástico negro



Ilustración 63. Reconocimiento del palé sobre plástico negro

Por ello, se decidió probar con otro material con un acabado mate. Se utilizó melanina, y aunque el resultado mejoró como se aprecia en la Ilustración 64 e Ilustración 65, este fondo seguía generando reflejos.

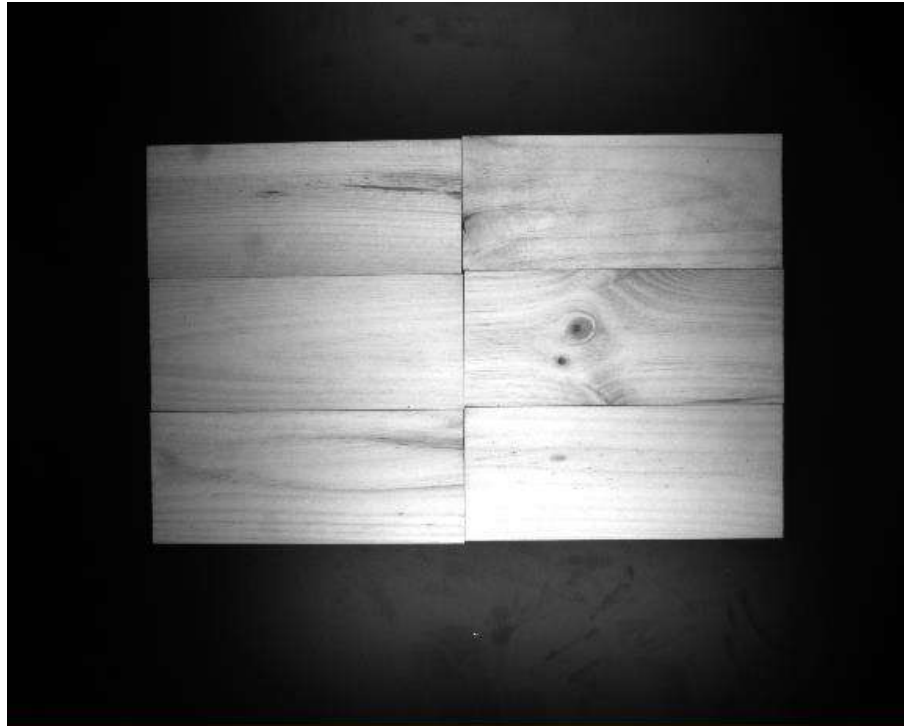


Ilustración 64. Tablas sobre melanina negra

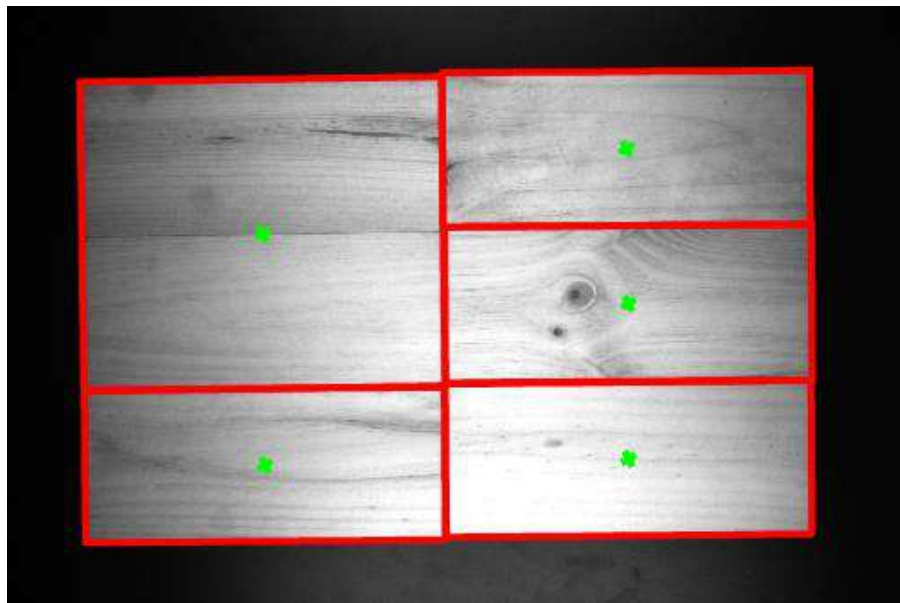


Ilustración 65. Reconocimiento de tablas sobre melanina negra

Finalmente, se optó por usar tela negra, ya que fue el material que menos reflejos generaba, como se puede apreciar en la Ilustración 66. Con el fondo seleccionado y la colocación de la iluminación se hicieron pruebas con diversas orientaciones y tamaños de tablas.

Las pruebas realizadas variando los tamaños de las tablas se hicieron reconociendo las de menor tamaño, como las de la Ilustración 69, y de tamaños intermedios, como las de la Ilustración 67 e Ilustración 68. Con estas pruebas se comprobó la robustez del algoritmo y además permitió reconocer el área de observación bajo la cual es posible garantizar un funcionamiento óptimo de la aplicación, siendo este de 500x300. En la Ilustración 68, se puede apreciar cómo se detecta como fondo los extremos de las tablas debido a que en esas zonas no reciben una buena iluminación.

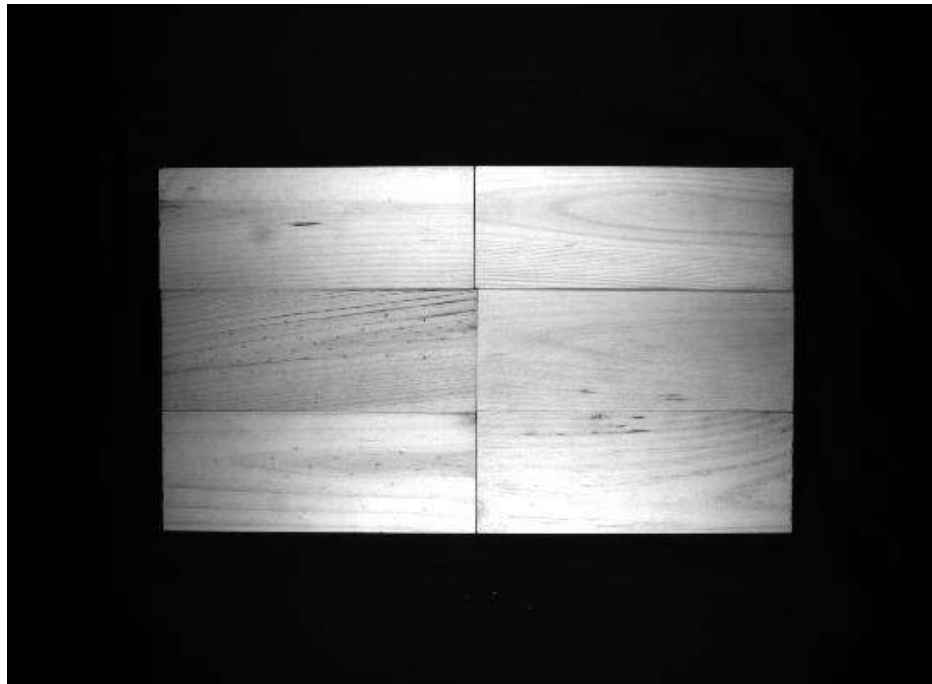


Ilustración 66. Tablas sobre tela negra

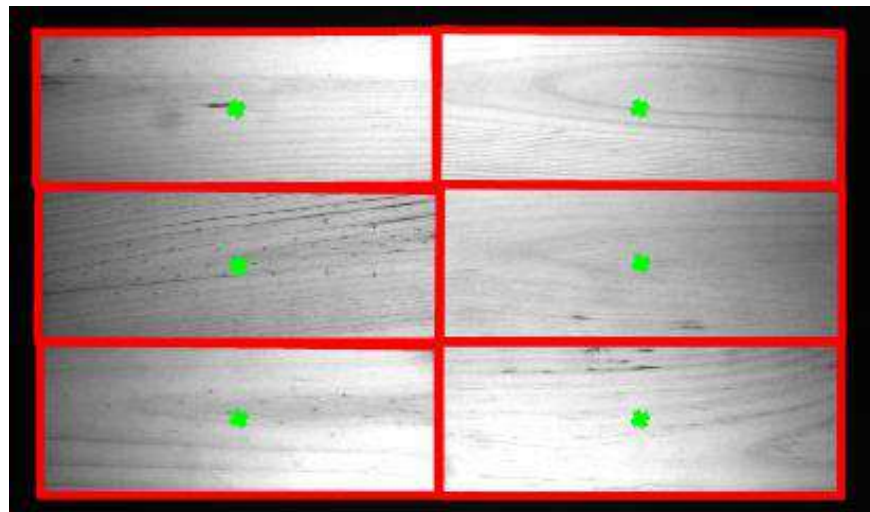


Ilustración 67. Reconocimiento de tablas 240x90 sobre tela negra

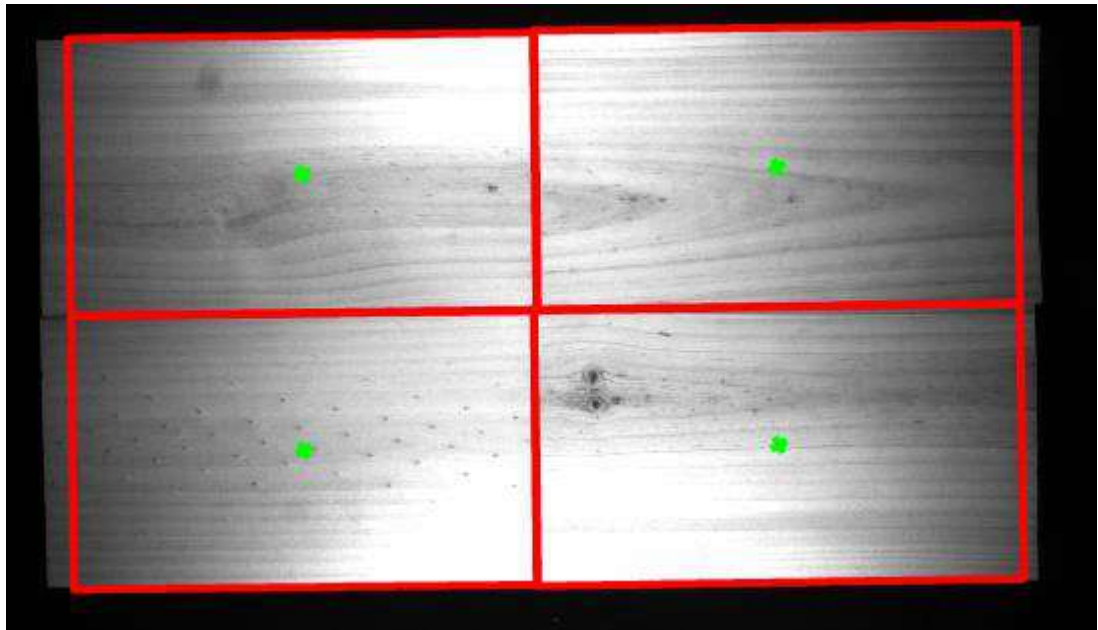


Ilustración 68. Reconocimiento tablas 310x170 sobre tela negra

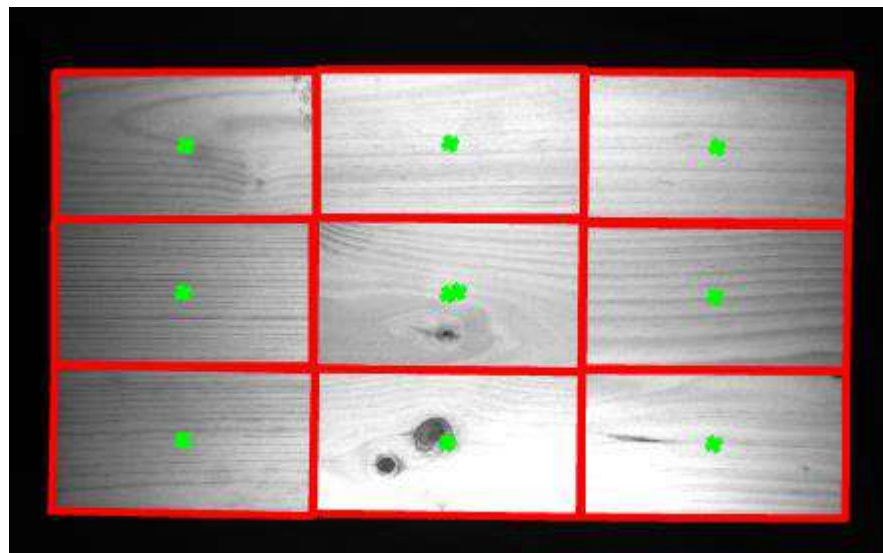


Ilustración 69. Reconocimiento tablas 170x90 sobre tela negra

Por otro lado, también se hicieron pruebas variando la orientación de las tablas. Todas las tablas llegan con un cierto orden sobre el palé. Por ello, para mejorar el funcionamiento, se reconoce primeramente el palé para posicionar la cámara en su centro con la correcta orientación. En la Ilustración 70 se puede ver cómo se detecta correctamente el palé. Sin embargo, para obtener una mayor robustez o por si algunas tablas estuviesen en horizontal y otras en vertical en el plano del palé, también se comprobó, como se aprecia en la Ilustración 71, que el algoritmo es capaz de detectar varias tablas con diferentes orientaciones.

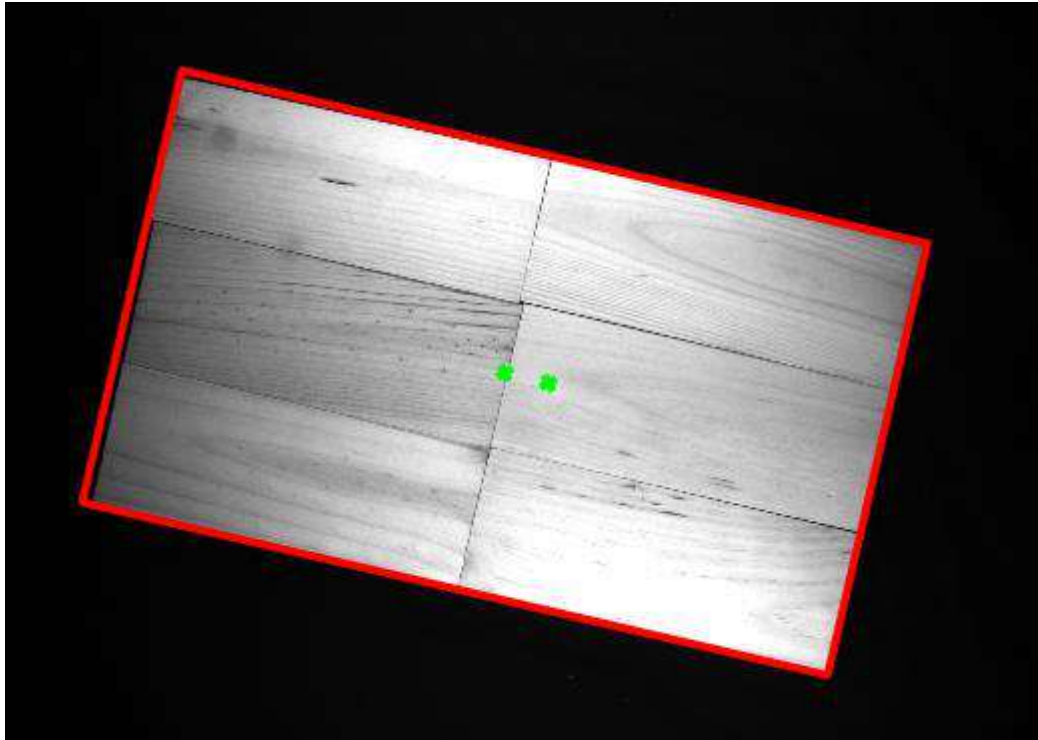


Ilustración 70. Reconocimiento palé sobre tela negra

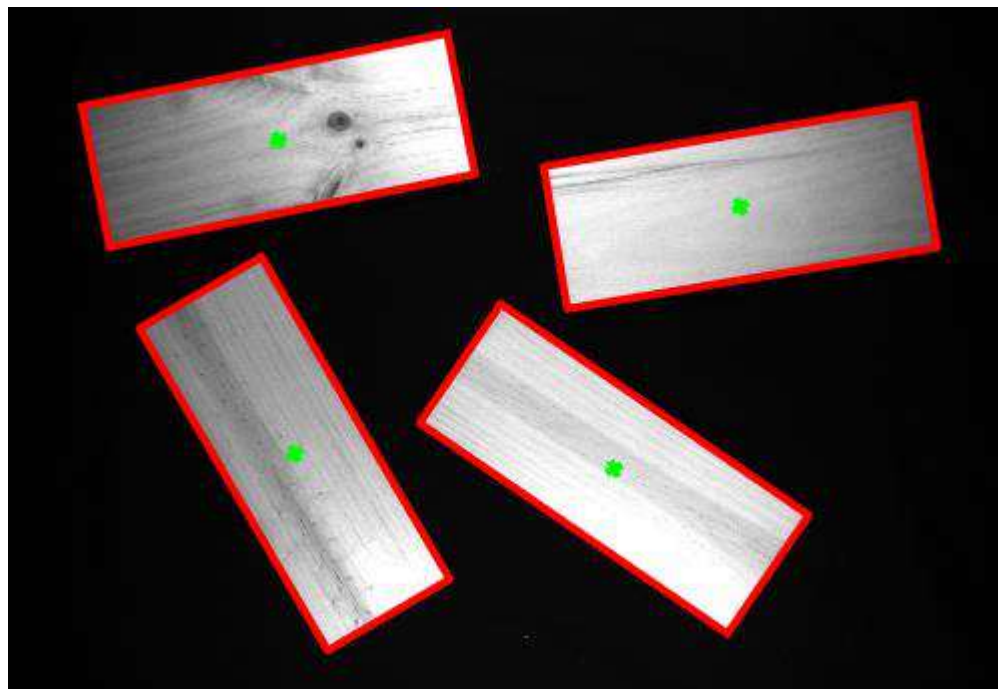


Ilustración 71. Reconocimiento tablas con diferentes orientaciones

11.2 Efecto de la iluminación ambiente

Posicionadas las barras de iluminación para obtener un correcto reconocimiento de las tablas, se procedió a comprobar el efecto de esta y del filtro de color. En la Ilustración 72 se ven las tablas iluminadas únicamente con luz ambiente y sin ningún filtro en la óptica de la cámara. Las tablas se encuentran bien iluminadas, siendo posible realizar un correcto reconocimiento de estas. Sin embargo, esta iluminación no está controlada, por lo que no se puede garantizar que la calidad de las imágenes sea la necesaria para esta aplicación.

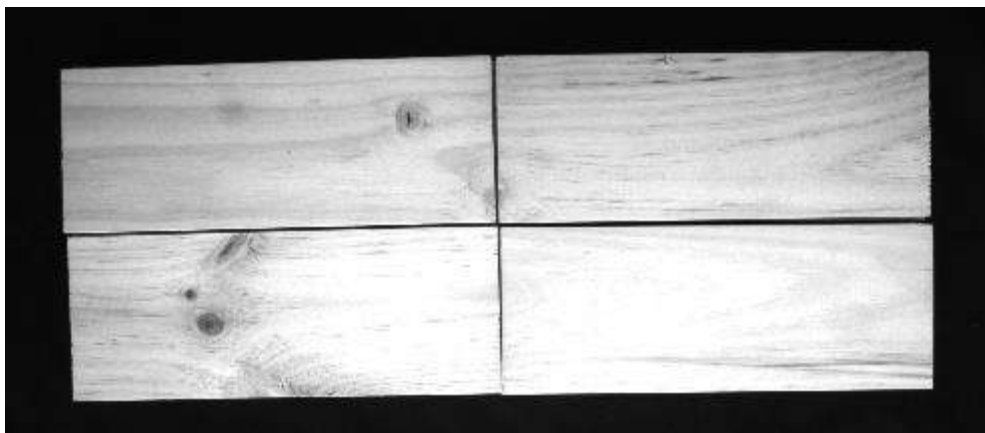


Ilustración 72. Tablas observadas sin iluminación y sin filtro

En la Ilustración 73 se puede ver el efecto del filtro sobre la iluminación ambiente. La imagen obtenida se ve oscurecida, complicando el reconocimiento de las tablas. De esta forma, se comprueba que el filtro elimina en gran medida la luz ambiente, obteniendo el resultado deseable, el cual es reducir considerablemente su efecto ya que se trata de un factor no controlable.



Ilustración 73. Tablas observadas sin iluminación y con filtro

11.3 Secuencia de órdenes y comunicaciones

Una vez conseguido el reconocimiento de las tablas y configurado el posicionamiento de los elementos de la estación, se probó el funcionamiento de las comunicaciones de la estación. Se detectó una ralentización en el envío de la imagen de la cámara hasta el PC de aproximadamente 1 segundo. También se observó un tiempo similar entre la orden de conmutación de la electroválvula y el cambio entre succión y reposo en la garra de vacío. Con estos datos, se ajustaron los tiempos de espera del robot para asegurar el correcto funcionamiento de las comunicaciones.

Además, se comprobó el correcto funcionamiento de la secuencia de las tareas de cada dispositivo de la estación impuesta por el PLC, observando las variaciones en los DB, sin ejecutar los movimientos completos con el robot, asegurándose que los datos de coordenadas y orientaciones eran correctos.

11.4 Movimientos del robot

Comprobado el correcto funcionamiento de las comunicaciones, del orden del ciclo y el reconocimiento de las tablas, se ajustaron los movimientos del robot para adaptarlos al espacio de trabajo de la estación y las herramientas utilizadas.

Lo primero que se comprobó es que, en la trayectoria de dejar pieza, el robot colisionaría con la iluminación. Por ello, se definió un punto intermedio evitando así la colisión.

Por otra parte, fue necesario ajustar cuánto debe bajar el robot más del nivel de la tabla para comprimir la espuma de la garra de vacío, pudiendo generar así la succión suficiente para manipularla, sin llegar a dañar la herramienta. Para ello, a la coordenada z de la tabla que el robot va a coger se le añade un cierto valor para que baje más del nivel de esta. Este valor se determinó de forma experimental, aumentándolo hasta lograr el correcto funcionamiento de la garra de vacío.

También se comprobaron si las calibraciones de las herramientas eran las correctas ejecutando ciclos completos variando la posición de las tablas.

Por último, se modificaron las velocidades de las trayectorias del robot para reducir lo máximo posible el tiempo de ciclo. Por seguridad, todas las pruebas explicadas en esta memoria hasta este momento se hicieron teniendo el robot en modo manual por seguridad. Este modo permite la ejecución del programa de forma continua, pero limita la velocidad máxima en las trayectorias. Para hacer pruebas a velocidades superiores de las permitidas en modo manual, se configuró el robot en modo automático. Se debe tener en cuenta que la peligrosidad para el programador se incrementa considerablemente al hacer dicho cambio, por lo hubo que tomar medidas de seguridad, guardando una distancia con el robot suficientemente amplia y teniendo siempre alcanzable la seta de paro de emergencia para evitar posibles accidentes. Los parámetros de velocidad se fueron aumentando hasta detectar que el funcionamiento de la estación no era el correcto, encontrando así el límite posible en la Rapidez del robot. Esta limitación en la velocidad no fue debida a ninguna configuración diseñada en este proyecto ni al incorrecto funcionamiento de ninguna de las herramientas del robot, y tampoco a las comunicaciones, puesto que ya se habían establecido los tiempos de espera necesarios. Fue debida a la plataforma sobre la que se encuentra el robot que, al no estar anclada al suelo, tiembla debido a la inercia en los movimientos del robot, pudiendo ocasionar un vuelco de esta.

11.5 Descripción del funcionamiento de la estación

Tras hacer las pruebas necesarias en los sistemas involucrados y las comunicaciones entre ellas, se procedió a la realización de varios ciclos, variando la colocación de las tablas para comprobar su funcionamiento.

1. El ciclo comienza cuando se pulsa el botón de inicializar en el HMI o en la botonera. Al hacerlo, el sistema de visión inicializa las variables del programa y el robot se posiciona de forma que permita colocar las tablas sobre la mesa, tal y como se ve en la Ilustración 74.

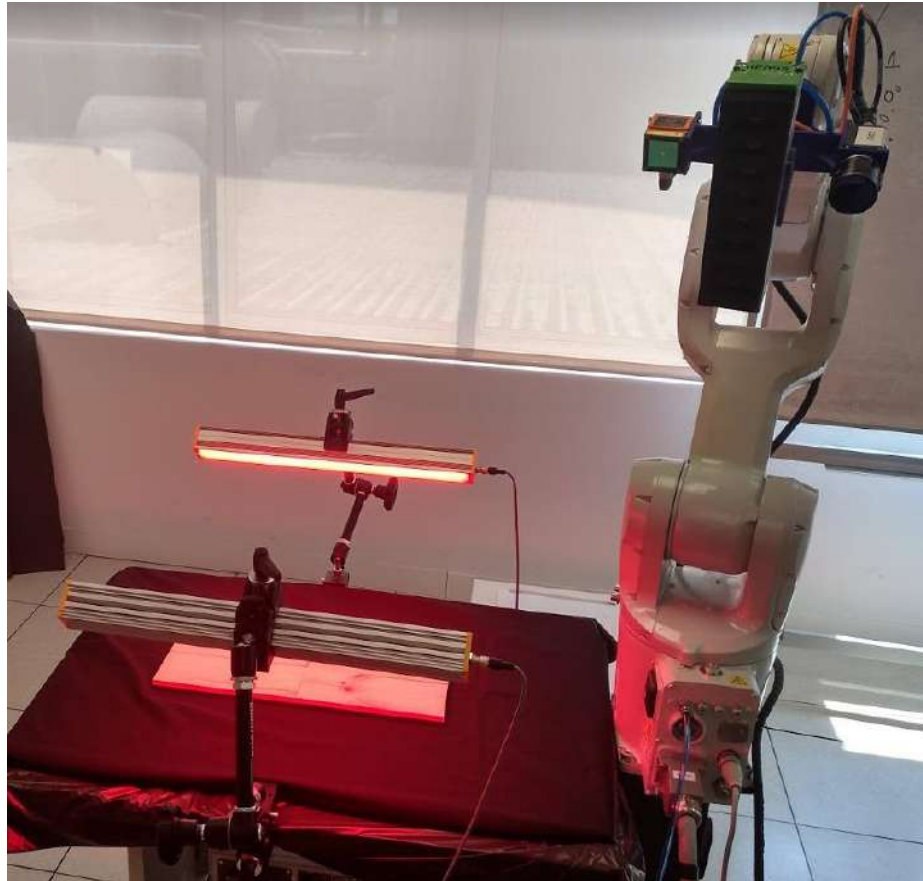


Ilustración 74. Robot en posición inicial

2. La estación se quedará en esta posición hasta que se pulse el botón que indica que el palé ya está colocado. En ese momento se sitúa el sensor laser sobre las tablas, como se aprecia en la Ilustración 76 y se guarda el dato de la altura.
3. Acto seguido se posiciona la cámara sobre ellas a la altura para la que fue calibrada el sistema, tal y como se ve en la Ilustración 75.

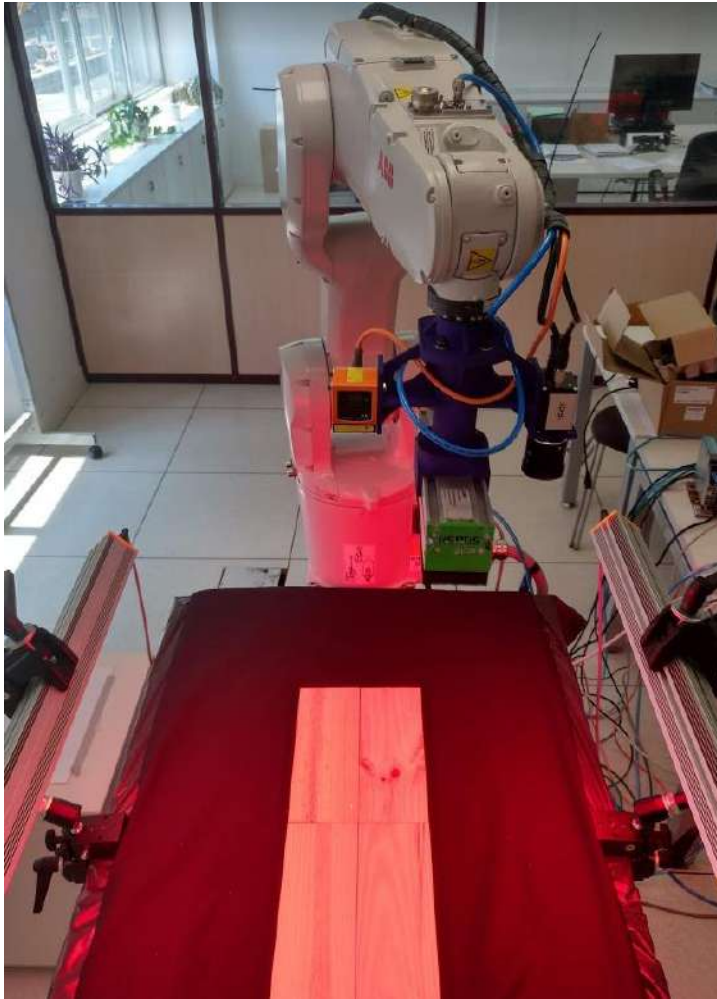


Ilustración 76. Sensor sobre palé

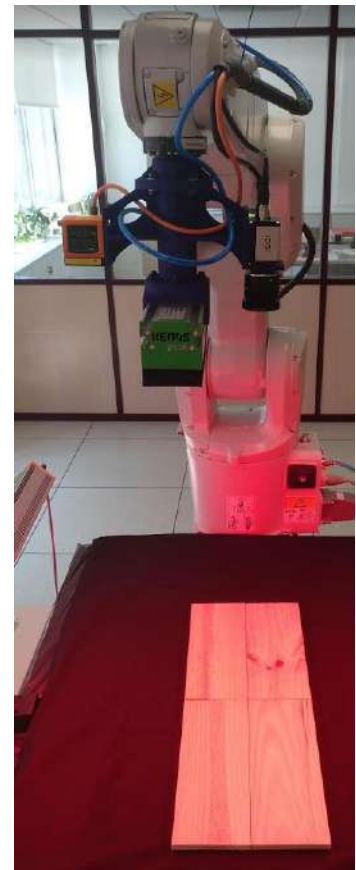


Ilustración 75. Cámara sobre palé

4. Con la cámara sobre las tablas se reconoce el palé. En la Ilustración 77 se ve dicho reconocimiento, en el cual el sistema de visión calcula cuanto debe desplazarse el centro de la cámara para que ambas cruces, que representan el centro de la cámara y del palé, sean coincidentes, y que la inclinación del palé observada por la cámara sea 0.

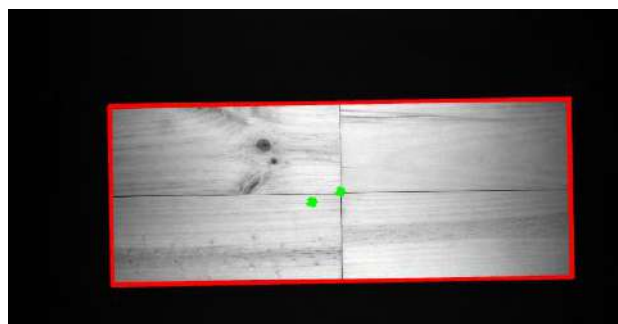


Ilustración 77. Detección del palé

5. Con los datos del sistema de visión se posiciona la cámara de forma que esta esté centrada sobre el palé, y se reconocen las tablas como se observa en la Ilustración 78, calculando sus coordenadas relativas y sus orientaciones con respecto al centro de la cámara.

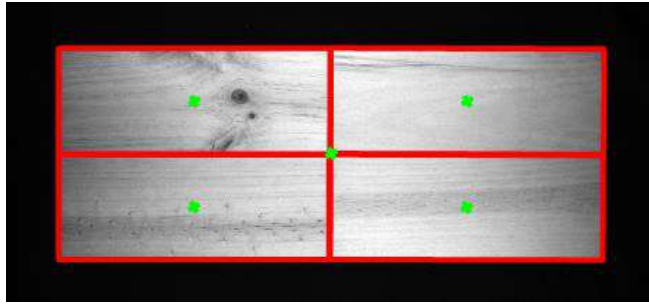


Ilustración 78. Detección de tablas

6. El sensor laser se posiciona sobre la primera de las tablas reconocidas por el sistema de visión, como se ve en la Ilustración 79, utilizando el dato medido para conocer a qué altura se encuentra dicha tabla.



Ilustración 79. Sensor sobre tabla

7. La garra de vacío se desplaza hasta la tabla tal y como se ve en la Ilustración 80. Una vez ha descendido la distancia adecuada, se conmuta la electroválvula para que se genere la succión necesaria para que la tabla se pueda mover solidaria a la herramienta del robot.



Ilustración 80. Garra de vacío sobre tabla

8. La tabla se retira del palé y se desplaza hasta su lugar de descarga, tal y como se ve en la Ilustración 81.
9. Los pasos 5, 6 y 7 se repiten hasta retirar todas las tablas detectadas por el sistema de visión.
10. Retiradas todas las tablas de un mismo nivel, se repite el proceso desde el punto 2.
11. Una vez retiradas todas las tablas del palé, al llegar al punto 3 no se detectará ninguna tabla, posicionando el robot de nuevo en la posición inicial.

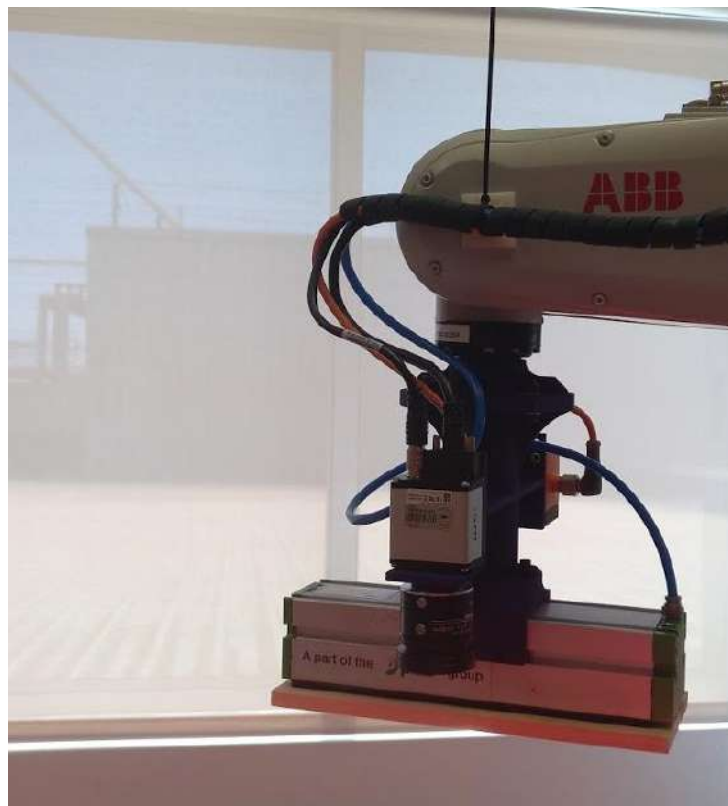


Ilustración 81. Robot moviendo tabla

11.6 Precisión

Para validar la aplicación diseñada es necesario hacer un estudio de la precisión de esta. Existen varios factores que afectan, siendo unos más influyentes que otros. Los que menos afectan son la precisión en los movimientos del robot y el número de cifras significativas de las coordenadas. El robot es capaz de moverse con una precisión de hasta centésimas de milímetro, y las coordenadas que se le indican constan de hasta décimas de milímetro, precisiones suficientes para esta aplicación.

Otros factores que afectan al error son los relativos a la calibración de las herramientas del robot. Los errores de precisión en su centro, y sobre todo en su orientación, producen errores mayores en la precisión con la que el robot manipula las tablas. Dado que todas las herramientas por diseño se posicionan con la misma orientación, los errores relativos a ellas se ven reducidos. Sin embargo, sí que afecta de especial forma el error entre la orientación entre las herramientas y el sistema de visión, ya que su calibración se hizo de forma manual. También afecta de forma considerable la calibración de la herramienta de la garra de vacío, siendo esta la más complicada para definir su centro exacto como se detalla en el capítulo de calibraciones de herramientas del robot.

Por último, el cálculo de las coordenadas y orientación de las tablas a partir del reconocimiento de las tablas por el sistema de visión artificial es susceptible de cometer imprecisiones debido al diversos factores, principalmente el de la iluminación y la similitud entre las tablas, pudiendo no corresponder con exactitud la tabla real con la región que el algoritmo reconoce como tabla, estando desplazado su centro.

Como el objetivo de la aplicación es despaletizar las tablas y posicionarlas en un lugar en concreto, se comprobó la exactitud de forma experimental con la que las tablas son colocadas en la posición deseada. Para ello, se hicieron varias pruebas haciendo funcionar la estación de manera automática, dibujando en un papel, colocado bajo el lugar de descarga, el contorno de las tablas. En la Ilustración 82 se puede ver la silueta de 12 tablas de 240x90 mm en el lugar en el que la estación las posicionó. Todas ellas se encuentran dentro de un rectángulo de 260x100 mm. Repitiendo este experimento se concluyó que la aplicación es capaz de posicionar las tablas dentro de un espacio delimitado por un rectángulo de medidas 3 mm mayor en sus 2 dimensiones que el de las tablas, cuya orientación y centro se encuentren en el punto de descarga deseado.

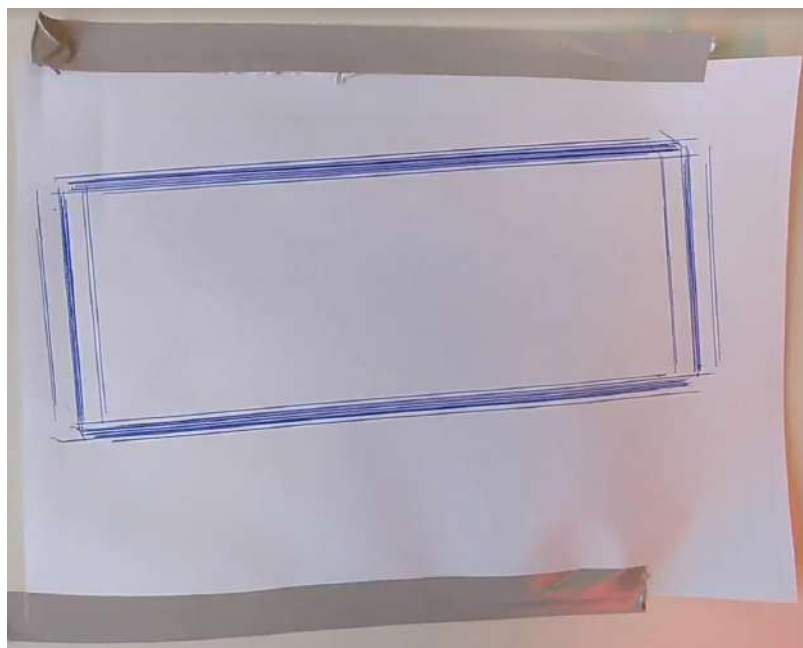


Ilustración 82. Ubicación de las tablas despaletizadas

11.7 Tiempo de ciclo

Para completar las comprobaciones y estudios de la aplicación, se hizo un estudio de los tiempos que invertía el sistema en cada proceso. Para ello, al igual que para las pruebas de precisión, se hicieron varios ciclos extrayendo los siguientes resultados:

- Tiempo medio desde que se indica que el palé está colocado hasta que lo reconoce y se centra la cámara sobre él: 6 s, empleando 3 s el sistema de visión.
- Tiempo medio en reconocer las piezas e iniciar el movimiento de extracción de ellas: 6 s, empleando 4 s el sistema de visión.
- Tiempo medio en extraer una pieza: 9 s.

A priori, estos tiempos son elevados para un funcionamiento aceptable en planta para esta aplicación. No obstante, se deben tener en cuenta las siguientes consideraciones:

- Los movimientos del robot no pueden ser más rápidos en el laboratorio debido a que este no se encuentra anclado a una plataforma estable.
- En la extracción de las piezas, el robot realiza una trayectoria amplia para esquivar la iluminación. Esta iluminación en fábrica podría ser diseñada con soportes pensados en optimizar dicha trayectoria.
- El sistema de visión es mejorable ya que se ejecuta el programa en un ordenador de bajas prestaciones, con un procesador Intel Core i3.
- La distancia a la que se encuentran las tablas del robot se comprueba en cada una de ellas. Si se puede garantizar que los palés llegan completos, teniendo todas las tablas visibles desde el sistema de visión a la misma distancia, el tiempo de ciclo se reduciría considerablemente, ya que únicamente sería necesario una comprobación de distancia por nivel de palé.
- Aunque actualmente el tiempo de ciclo a mano sea de 0'8 s/tabla, la estación no tiene necesidad de trabajar a esa velocidad puesto que solamente funciona una media de 4 h diaria, por lo que trabajando más lentamente la estación diseñada podría realizar el mismo volumen de trabajo en un día. Sería necesario hacer una valoración económica para comprobar la rentabilidad del proyecto.

12 CONCLUSIONES

De la realización de los trabajos realizados en esta memoria se extraen los siguientes resultados:

- Del análisis en campo se concluye que la estación actual se encuentra un entorno hostil, con una iluminación a gran altura de tubos fluorescentes y con gran cantidad de polvo de madera en suspensión. Trabaja una media de 4 horas diarias a 0'8 s/tabla.
- La arquitectura diseñada es un sistema de visión robótica compuesta por 3 sistemas: PLC, robot y sistema de visión, siendo el primero el encargado de gobernar la aplicación. El sistema de visión elegido funciona con tecnología 2D por lo que es necesario el uso de un sensor de distancia.
- El funcionamiento de la estación fue diseñado para facilitar la colocación del palé e ir retirando las tablas por niveles haciendo un único reconocimiento por nivel.
- Las herramientas se seleccionaron en base a las necesidades de la estación, al presupuesto disponible y a la disponibilidad y reutilización de estas en fábrica.
- La herramienta del robot fue diseñada y construida mediante fabricación aditiva, obteniendo un resultado ligero y compacto con una disposición de las herramientas con la misma orientación, facilitando así las tareas de calibración y optimizando las trayectorias del robot.
- El PLC se programó haciéndolo el único sistema conocedor del orden de las tareas de la aplicación. Gestiona el envío de órdenes a los demás sistemas en función de los estados de estos.
- Las trayectorias del robot se programaron como funciones dependientes de las órdenes y datos enviados por el PLC.
- El sistema de visión es capaz de reconocer el centro de las tablas y del palé, dependiendo únicamente de la iluminación diseñada, y de enviarle los datos al PLC cuando este se lo ordena. Para el reconocimiento de tablas se utiliza el algoritmo Regoingrowing Segmentation, y las tablas detectadas se rectangularizan, puesto que todas tienen esta forma, para obtener mayor precisión.
- El sistema es capaz de posicionar las tablas en un espacio rectangular de 3 mm mayor en sus 2 dimensiones que las medidas de la tabla despaletizada.
- El sistema emplea 6 s en el reconocimiento del palé, 6 s en el reconocimiento de las tablas de un nivel y 9 s en la extracción de cada tabla, estando limitada por el anclaje del robot en el laboratorio y por un PC de visión no adecuado.
- Por último, y como resultado de todo lo anterior, se concluye que se desarrolló el proyecto piloto de una estación de despaletizado de tablas de madera mediante visión robótica.

13 LÍNEAS FUTURAS

En este trabajo se ha desarrollado un proyecto piloto, que para su puesta en funcionamiento en fábrica es necesario hacerle una serie de modificaciones y comprobaciones.

La primera de ellas, siendo la más necesaria, es el desarrollo del sistema de seguridad, tanto a nivel físico como a nivel de programación. La mayoría de las estaciones que funcionan con robots no colaborativos se suelen enjaular con el fin de evitar posibles colisiones con otros elementos o con humanos. En las zonas donde esto no es posible deben colocarse sensores que garanticen la seguridad de la estación. Se puede ver un ejemplo de esto en fábrica en la Ilustración 83. Los sensores utilizados para medidas de seguridad deben de transmitir su información utilizando un protocolo seguro, ya sea mediante un bus o a unas tarjetas de seguridad. Con la información de estos se debe desarrollar un programa que garantice el correcto funcionamiento de las alarmas y medidas de seguridad.



Ilustración 83. Estación enjaulada

Por otra parte, también es necesario adaptar las herramientas utilizadas y su colocación a fábrica. El robot utilizado en este proyecto tiene un buen tamaño y alcance para hacer pruebas en laboratorio, pero para cumplir las funciones necesarias en fábrica. Lo mismo ocurre con la iluminación, que no cubre toda el área de un palé y su soporte no es lo suficientemente seguro, ni su colocación es óptima para facilitar los movimientos del robot. De igual forma sucede con el PC de visión robótica, ya que para trabajar en fábrica debe tratarse de un PC industrial.

Además, sería conveniente realizar más pruebas y modificaciones con el fin de otorgar mayor robustez, pudiendo ser interesante utilizar tecnología 3D en vez de 2D, ahorrando además tiempo al no tener que utilizar el sensor de distancia.

Por último, sería necesario realizar una valoración económica teniendo en cuenta el desarrollo del proyecto con los cambios anteriormente mencionados para hacer un estudio de la viabilidad de la aplicación.

14 BIBLIOGRAFÍA

1. Finsa. [En línea] [Citado el: 01 de 03 de 2019.] <https://www.finsa.com/>.
2. Berret, Marcus, Ring, Thomas y Kube, Georg. Roland Berger. *Industry 4.0: A real quantum leap*. [En línea] 01 de 12 de 2016. [Citado el: 01 de 03 de 2019.] <https://www.rolandberger.com/en/Point-of-View/Industry-4.0-A-real-quantum-leap.html>.
3. Ude, Aleš. Zums. *Robot Vision*. [En línea] 2010. [Citado el: 01 de 03 de 2019.] <http://zums.ac.ir/files/research/site/ebooks/Robotics/Robot%20Vision.pdf>.
4. de la Fuente Lopez, Eusebio y Trespaderne, Félix Miguel. Revistadyna. *Guiado de robots industriales con visión artificial. Robotización de un proceso de fabricación de radiadores*. [En línea] 10 de 2009. [Citado el: 01 de 03 de 2019.] <https://www.revistadyna.com/busqueda/guiado-de-robots-industriales-con-vision-artificial-robotizacion-de-un-proceso-de-fabricacion-de-rad>.
5. Automated Imaging Association. *Introduction to Machine Vision Lighting*. [En línea] 22 de 07 de 2011. [Citado el: 08 de 03 de 2019.] <https://www.visiononline.org/search-results.cfm>.
6. SlideShare. [En línea] [Citado el: 08 de 03 de 2019.] <https://www.slideshare.net/engineerfriend/machine-vision-lighting>.
7. bcnavision. [En línea] [Citado el: 08 de 03 de 2019.] <http://www.bcnavision.es/blog-vision-artificial/iluminacion-vision-artificial/>.
8. Academico. *Leyes de refracción y lentes*. [En línea] [Citado el: 22 de 02 de 2019.] <http://www.academico.cecyt7.ipn.mx/FisicaIV/unidad2/lentes.htm>.
9. Szeliski, Richard. *Computer Vision: Algorithms and Applications*. [En línea] 03 de 09 de 2010. [Citado el: 20 de 02 de 2019.] http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf.
10. Ikusmen. [En línea] [Citado el: 08 de 03 de 2019.] http://www.ikusmen.com/documentos/descargas/3cbb38_Introduction%20to%20Machine%20Vision.pdf.
11. Čapek, Karel. *R.U.R. (Robots Universales Rossum)*. 1920. 8467208082.
12. Yukou, Lie. *Lie Zi*. 300 a.C. 9787805695211.
13. Martínez, R. y García-Beltrán, A. OCW. *Breve historia de la informática*. [En línea] octubre de 2000. [Citado el: 15 de 03 de 2019.] <http://ocw.upm.es/ciencia-de-la-computacion-e-inteligencia-artificial/fundamentos-programacion/otrosrecursos/brevehistoriainformatica.pdf>.
14. Adelerobots. *Nuka*. [En línea] [Citado el: 10 de 02 de 2019.] <https://www.adelerobots.com/es/nuka/>.
15. Ailverobots. *Nao*. [En línea] [Citado el: 10 de 02 de 2019.] <https://aliverobots.com/nao/>.
16. Thefabricator. *Un robot para soldadura*. [En línea] [Citado el: 10 de 03 de 2019.] <https://www.thefabricator.com/article/automationrobotics/un-robot-para-soldaduraaen-continuo-movimiento>.
17. Sánchez, Carlos. El Confidencial. *La industria ya emplea un ejército de 35.000 robots: dos por cada 1.000 trabajadores*. [En línea] 11 de 01 de 2018. [Citado el: 10 de 03 de 2019.] https://www.elconfidencial.com/economia/2018-01-11/robots-industria-deslocalizacion-automovil-empleos-destruccion-algoritmos-asia-alemania-espana-cotizaciones-impuestos-sueldos-bruegel_1504357/.
18. Chimarro Amaguaña, Juan David y Enríquez Herrera, Andrés David. Universidad de las fuerzas armadas de Ecuador. *Diseño, construcción e implementación de un robot esférico de 4*

grados de libertad para manipulación de objetos utilizando la plataforma robotic operating system (ROS). [En línea] 2015. [Citado el: 20 de 02 de 2019.] <https://repositorio.espe.edu.ec/bitstream/21000/11249/1/T-ESPE-049400.pdf>.

19. Roboticatecnologia. [En línea] [Citado el: 15 de 03 de 2019.] <http://roboticatecnologia2010.blogspot.com/p/generaciones.html>.

20. Sanz Valero, Pedro José. uji. *Introducción a la robótica inteligente.* [En línea] 2006. [Citado el: 15 de 03 de 2019.] <http://www3.uji.es/~sanzp/robot/RobInt-Apuntes.pdf>.

21. Kânge, Frederik. KTH. *Methods for Real-Time Bin-Picking using 2D Vision.* [En línea] [Citado el: 10 de 03 de 2019.] https://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2007/rapporter07/kange_fredrik_07034.pdf.

22. Pinterest. *iRVision from Fanuc.* [En línea] [Citado el: 18 de 07 de 2019.] <https://www.pinterest.cl/pin/294000681908109903/>.

23. Departamendo de ingeniería eléctrica, electrónica y de control. ieec. *Estructura general, componentes. Tipos de PLCs. Funcionamiento.* [En línea] [Citado el: 12 de 04 de 2019.] http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE6_1_1.pdf.

24. ABB. *Especificaciones del producto IRB 1200.* [En línea] 03 de 04 de 2017. [Citado el: 29 de 03 de 2019.] <https://search-ext.abb.com/library/Download.aspx?DocumentID=3HAC046982-005&LanguageCode=es&DocumentPartId=&Action=Launch>.

25. ABB. *Data sheet IRC5 Compact.* [En línea] 08 de 10 de 2018. [Citado el: 29 de 03 de 2019.] <https://search-ext.abb.com/library/Download.aspx?DocumentID=ROB0295EN&LanguageCode=en&DocumentPartId=&Action=Launch>.

26. ABB. *RobotStudio.* [En línea] [Citado el: 29 de 03 de 2019.] <https://new.abb.com/products/robotics/es/robotstudio>.

27. Controller IRC5 with Flexpendant. ABB. [En línea] [Citado el: 29 de 3 de 2019.] <https://library.e.abb.com/public/2b5b950d68a0503cc1257c0c003cb703/3HAC041344-es.pdf>.

28. ABB. *Data sheet RobotWare-Dispense.* [En línea] 11 de 01 de 2019. [Citado el: 03 de 29 de 2019.] <https://search-ext.abb.com/library/Download.aspx?DocumentID=9AKK107046A7582&LanguageCode=en&DocumentPartId=&Action=Launch>.

29. Effilux. *Effi-Flex.* [En línea] [Citado el: 01 de 04 de 2019.] <http://www.effilux.fr/es/products/led-bar/effi-flex/>.

30. Effilux. *Effi-Ring.* [En línea] [Citado el: 26 de 03 de 2019.] <http://www.effilux.fr/es/products/ring/effi-ring/optical-configurations/>.

31. iDS. *UI-5200SE Rev.4.* [En línea] [Citado el: 20 de 03 de 2019.] <https://es.ids-imaging.com/store/ui-5200se-rev-4.html>.

32. MVTec. *Halcon.* [En línea] [Citado el: 03 de 02 de 2019.] <https://www.mvtec.com/products/halcon/>.

33. Intel. *Open CV.* [En línea] 06 de 07 de 2017. [Citado el: 11 de 02 de 2019.] <https://software.intel.com/en-us/articles/what-is-opencv>.

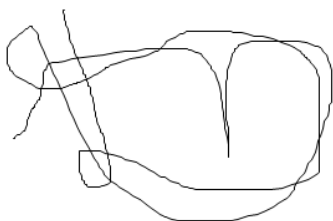
34. Teledyne Dalsa. *Sherlock.* [En línea] [Citado el: 11 de 02 de 2019.] <https://www.teledynedalsa.com/en/products/imaging/vision-software/sherlock/>.

35. iDS. *Camera manager.* [En línea] [Citado el: 28 de 02 de 2019.] <https://es.ids-imaging.com/ids-camera-manager.html>.

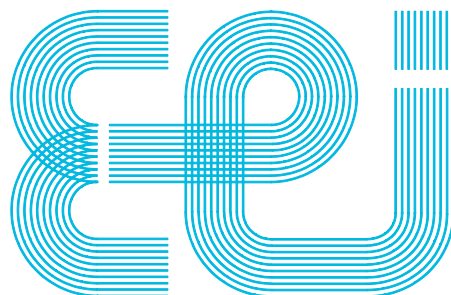
36. Piab. *Kenos*. [En línea] [Citado el: 29 de 03 de 2019.] <https://www.piab.com/es-ES/productos/sistemas-de-sujecion-por-vacio-kenos/kvg/kenos-vacuum-gripper-kvg60/#ordering>.
37. Festo. *Electroválvula neumática MFH-3-1/8*. [En línea] [Citado el: 24 de 04 de 2019.] https://www.festo.com/net/es_es/SupportPortal/default.aspx?cat=1648&q=535897&tab=29&s=t.
38. IMI Norgren. *13J*. [En línea] [Citado el: 20 de 04 de 2019.] <http://pages.norgren.com/es/valves.html?vs=ZjBkYjJmOWQtZDIzOS00NjNmLTgxMDQtYTFhZTRiYTUxZjYyOzsS1>.
39. IFM. *Sensor de distancia O1D100*. [En línea] [Citado el: 02 de 04 de 2019.] <https://www.ifm.com/es/es/product/O1D100?tab=details>.
40. Lupeon. [En línea] [Citado el: 14 de 06 de 2019.] <https://lupeon.com/>.
41. Siemens. *CPU 1513F-1PN*. [En línea] [Citado el: 20 de 04 de 2019.] <https://w3.siemens.com/mcms/programmable-logic-controller/en/advanced-controller/s7-1500/cpu/portfolio/Pages/failsafe-cpus.aspx>.
42. Siemenes. *SITOP PSU200M 10 A*. [En línea] [Citado el: 30 de 04 de 2019.] <https://mall.industry.siemens.com/mall/es/es/Catalog/Product/6EP1334-3BA10>.
43. Siemens. *SCALANCE W774-1*. [En línea] <https://mall.industry.siemens.com/mall/es/es/Catalog/Product/6GK5774-1FX00-0AA0>.
44. Siemens. *IM 155-6 PN HF*. [En línea] [Citado el: 30 de 04 de 2019.] <https://mall.industry.siemens.com/mall/es/es/Catalog/Product/6ES7155-6AU00-0CN0>.
45. Siemens. *DI 16x24VDC ST*. [En línea] <https://mall.industry.siemens.com/mall/es/es/Catalog/Product/6ES7131-6BH01-0BA0>.
46. Siemens. *DQ 16x24VDC/0.5A ST*. [En línea] [Citado el: 30 de 04 de 2019.] <https://mall.industry.siemens.com/mall/es/es/Catalog/Product/6ES7132-6BH01-0BA0>.
47. Siemens. *AI 4xI 2-,4-wire ST*. [En línea] [Citado el: 30 de 04 de 2019.] <https://mall.industry.siemens.com/mall/es/es/Catalog/Product/6ES7134-6GD00-0BA1>.
48. Siemens. *SIMATIC ET 200SP*. [En línea] [Citado el: 30 de 04 de 2019.] <https://mall.industry.siemens.com/mall/es/es/Catalog/Product/6ES7193-6PA00-0AA0>.
49. mercadoactual. *D-Link DES-1005D*. [En línea] [Citado el: 07 de 06 de 2019.] <https://www.mercadoactual.es/redes/switches-hubs/d-link-des-1005d-100986.html>.
50. Manfrotto. *Brazo Fricción Variable sin Soporte Cámara*. [En línea] [Citado el: 26 de 04 de 2019.] <https://www.manfrotto.es/variable-friction-arm-alone>.
51. Manfrotto. *Súper Pinza*. [En línea] [Citado el: 26 de 04 de 2019.] <https://www.manfrotto.es/super-photo-clamp-without-stud-aluminium>.
52. Manfrotto. *Juego de 4 Cuñas para Súper Pinza*. [En línea] [Citado el: 26 de 04 de 2019.] <https://www.manfrotto.es/set-of-4-wedges-for-s-clamp>.
53. Manfrotto. *Espiga para Cámara 1/4" 3/8"*. [En línea] [Citado el: 26 de 04 de 2019.] <https://www.manfrotto.es/camera-stud-superclamp-access>.
54. ABB. Personal.biada. *Manual de referencia técnica*. [En línea] [Citado el: 20 de 02 de 2019.] <http://personal.biada.org/~jhorrrillo/INSTRUCCIONES%20RAPID.pdf>.
55. Personal.biada. *Lenguaje RAPID*. [En línea] 2006. [Citado el: 20 de 02 de 2019.] <http://personal.biada.org/~jhorrrillo/INTRODUCCIO%20RAPID.pdf>.
56. Robotics, BCIT Mechatronics and. youtube. *ABB RAPID programming - Offs versus Reltool*. [En línea] 23 de 10 de 2014. [Citado el: 11 de 03 de 2019.] <https://www.youtube.com/watch?v=e4gkf77uGF0>.

57. ces. *Canny Edge Detection*. [En línea] 23 de 03 de 2009. [Citado el: 29 de 05 de 2019.] <http://www.cse.iitd.ernet.in/~pkalra/col783-2017/canny.pdf>.
58. John Canny, IEEE. citeseerx. *A computational approach to edge detection*. [En línea] 6 de 11 de 1986. [Citado el: 29 de 05 de 2019.] <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf>.
59. Deriche, Rachid. Springer. [En línea] 1987. [Citado el: 31 de 05 de 2019.] <https://link.springer.com/article/10.1007/BF00123164>.
60. Bialkowski, Stephen E. Real-Time Digital Filters: Infinte Impulse Response Filters. *A/C INTERFACE*. [En línea] 1988. [Citado el: 29 de 3 de 2019.] https://pubs.acs.org/doi/pdf/10.1021/ac00157a003?casa_token=E4GazwirmWUAAAAA%3AV_oja8wnjylE9x1xA36zo1sFTDYsGdebf_ekfzLFKZOCuWhKWnzhrjQvAsRGKEnibleySLeVmo6dkg&.
61. cse. *Region Growing*. [En línea] [Citado el: 31 de 05 de 2019.] <https://www.cse.unr.edu/~bebis/CS791E/Notes/RegionGrowing.pdf>.
62. pointclouds. *Region growing segmentation*. [En línea] [Citado el: 31 de 05 de 2019.] http://pointclouds.org/documentation/tutorials/region_growing_segmentation.php.
63. Deering, S. y Hinden, R. tools. *Internet Protocol, version 6 (IPv6)*. [En línea] [Citado el: 07 de 06 de 2019.] <https://tools.ietf.org/html/rfc1883>.
64. Profibus. *Profinet*. [En línea] [Citado el: 14 de 06 de 2019.] <https://us.profinet.com/technology/profinet/>.
65. Universidad Profinet. *Conceptos básicos de los archivos profinet GSD*. [En línea] [Citado el: 12 de 06 de 2019.] <https://profinetuniversity.com/profinet-basico/conceptos-basicos-de-los-archivos-profinet-gsd/>.
66. matrikonopc. *OPC UA (Arquitectura Unificada)*. [En línea] [Citado el: 14 de 06 de 2019.] <https://www.matrikonopc.es/opc-ua/index.aspx>.
67. Owen-Hill, Alex. Robotiq. *Robot Vision vs Computer Vision: What's the Difference?* [En línea] 07 de 07 de 2016. [Citado el: 01 de 03 de 2019.] <https://blog.robotiq.com/robot-vision-vs-computer-vision-whats-the-difference>.

En Santiago, el 1 de agosto de 2019



Daniel Lamas Novoa



Escuela de Ingeniería Industrial

TRABAJO FIN DE MÁSTER

*Diseño de una estación de despaletizado mediante un sistema
de visión robótica*

Máster en ingeniería industrial

Documento

PRESUPUESTO

Universidade de Vigo

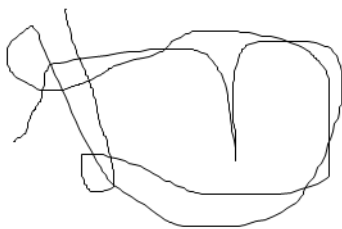
Referencia	Udes.	Precio (€)	Amortización (años/años)	Coste (€)
Meses de trabajo de ingeniero	6	700	-	4.200
Sistema de visión artificial		5.312,47		265,62
Cámara IDS-AB02278	1	2.660,00	0,5/10	133,00
Cable Gige 5m RJ45	1	64,19	0,5/10	3,21
Cable con conector HR8 5m	1	85,23	0,5/10	4,26
Óptica 12mm CBC-IPY-V1228	1	560,00	0,5/10	28,00
Filtro luz roja MID-IBP635-M40.5	1	125,71	0,5/10	6,29
Iluminación EFF-FL25-625-TR-P3	2	1.125,00	0,5/10	56,25
Ventana difusora media EFF-O-FL-SD-15	2	44,00	0,5/10	2,20
Ventana difusora opalina EFF-O-FL-OP-15	2	44,00	0,5/10	2,20
Cable pow 5m para iluminación	2	70,00	0,5/10	3,50
Brazo soporte iluminación MF244N	2	102,80	0,5/10	5,14
Pinza soporte iluminación MF035	4	25,31	0,5/10	1,27
Juego de 4 cuñas para pinza de soporte iluminación MF035WDG	1	6,23	0,5/10	0,31
PC Lenovo ThinkPad E130	1	400,00	0,5/10	20,00
Software Halcon V13 desarrollo	1	0,00	0,5/10	0,00
PLC y periferia distribuida		6.198,32		309,92
Alimentación	1	115,32	0,5/10	5,77
CPU 1515F 2PN	1	2.173,55	0,5/10	108,68
Perfil soporte S7-1500, 482 mm, para armarios de 19"	1	30,00	0,5/10	1,50
Memory Card, 2 GB	1	464,10	0,5/10	23,21
DI 16x24VDC ST	1	62,47	0,5/10	3,12
DQ 16x24VDC/0,5A ST	1	72,50	0,5/10	3,62
AI 4xI 2/4 hilos ST	1	125,67	0,5/10	6,28
F-DI 8x24VDC HF	1	157,83	0,5/10	7,89
F-DQ 4x24VDC/2A PM HF	1	182,81	0,5/10	9,14
IM 155-6 PN HF incluido módulo servidor	1	209,61	0,5/10	10,48
Adaptador de bus 2xFC	1	40,28	0,5/10	2,01
Tipo BU A0, 16 Push-In, 10 AUX, 2 bornes aliment.	1	14,67	0,5/10	0,73
Tipo BU A0, 16 Push-In, 10 AUX, 2 bornes aliment.	1	22,83	0,5/10	1,14
Tipo BU A1, 16 Push-In, 12 bornes aliment.	1	18,79	0,5/10	0,94
Tipo BU A1, 16 Push-In, 12 bornes aliment.	1	27,12	0,5/10	1,36
Perfil soporte 35 mm, longitud: 483 mm, para armarios de 19"	1	20,47	0,5/10	1,02
Cable cat5 verde	1	0,51	0,5/10	0,03
Conector profinet RJ45	1	11,97	0,5/10	0,60
Software de desarrollo del PLC	1	2.447,83	0,5/10	122,39
Sistema del robot		31.125,00		1.089,50
Manipulador		19.825,00		284,50
IRB 1200-5/0.9	1	18.935,00	0,5/10	946,75

Cableado usuario de 7 m	1	890,00	0,5/10	44,50
Armario de control		6.180,00		309,00
Tarjeta de 16 entradas / 16 salidas	1	490,00	0,5/10	24,50
Tarjeta Encoder Interface (externa)	1	505,00	0,5/10	25,25
DeviceNet Single Ch maestro / esclavo	1	740,00	0,5/10	37,00
Software PROFINET IO m/s	1	1.265,00	0,5/10	63,25
Safety Module	1	780,00	0,5/10	39,00
Profisafe -FDevice	1	1.225,00	0,5/10	61,25
Safemove Pro	1	1.175,00	0,5/10	58,75
Software		3.065,00		393,25
Paquete Multifuction	1	855,00	0,5/10	42,75
Conveyor TrACKing	1	1.475,00	0,5/10	73,75
PC Interface	1	735,00	0,5/10	36,75
RobotStudio 6.08	1	1.200,00	0,2/1	240,00
Mesa		2.055,00		102,75
Estructura	1	2.000,00	0,5/10	100,00
Lona negra	1	5,00	0,5/10	0,25
Tablero alta densidad negro 500x300	1	50,00	0,5/10	2,50
Herramienta del robot		11.991,79		2.251,53
Brida de unión con el robot	1	55,00	0,5/10	2,75
Union cámara y sensor de distancia	1	70,00	0,5/10	3,50
Unión garra de vacío	1	285,00	0,5/10	14,25
Software Solidworks	1	10.950,00	0,2/1	2.190,00
Garra de vacío Kenos KVG.200.60.N213.CVL.S2	1	631,79	0,5/10	31,59
Sensor de distancia IFM O1D100		188,84	0,5/10	9,44
Materiales para conexiones		275,72		275,72
CONEXION RECTA SMC KQB2H0602S 6 - 1/4"	1	1,28	1	1,28
RODAMIENTO FAG 6312 2ZC3	1	19,59	1	19,59
CONECTOR HEMBRA IFM EVC04A M12 5 PINES	1	10,71	1	10,71
CONEXION RECTA SMC KQB2H1202S 12 - 1/4"	1	4,35	1	4,35
CONEXION RECTA SMC KQB2H0602S 6 - 1/4"	1	2,55	1	2,55
MANGUITO SMC SA013 1/4"	1	1,01	1	1,01
VENTOSA C/MUELLE 2038752921	1	52,93	1	52,93
ELECTROVALVULA FESTO MFH 3 1/8" 7802	1	72,66	1	72,66
TUERCA BOBINA FESTO 209407	1	0,66	1	0,66
BOBINA NORGREN QM/48/13J/21 * QM/48A/10/	1	10,88	1	10,88
TUERCA REDUCCION 3/4" - 1/2" MH FIG241 G	1	1,37	1	1,37
ADAPTADOR MH SMC SA016 1/8 1/4	1	0,46	1	0,46
TUBO NORGREN 7 X 10 POLIURETANO PU205101	1	4,06	1	4,06
CONEXION RECTA SMC KQB2H1004S 10 - 1/2"	1	10,86	1	10,86
CONEXION RECTA SMC KQB2H0802S 8 - 1/4"	1	2,85	1	2,85

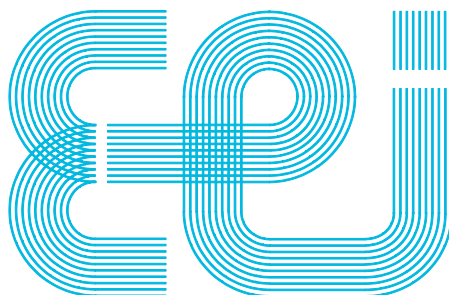
CABLE GENERAL CABLE 54314A1VDP CAT.6 FTP	1	5,66	1	5,66
CONECTOR RJ 45 APANTALLADO	1	1,99	1	1,99
CONEXION RECTA SMC KQB2H0802S 8 - 1/4"	1	1,42	1	1,42
CONEXION RECTA SMC KQB2H0801S 8 - 1/8"	1	5,70	1	5,70
TUBO SMC 5 X 8 POLIURETANO TU0805BU100 A	1	11,91	1	11,91
CONECTOR NORGREN M/P19063	1	2,69	1	2,69
CODO ORIENTABLE SMC KQB2L06M5 6 - M5	1	2,40	1	2,40
CONEXION RECTA SMC KQB2H0601S 6 - 1/8"	1	1,02	1	1,02
TUBO NORGREN 4 X 6 POLIURETANO PU0506100	1	0,82	1	0,82
CONECTOR HEMBRA ACODADO LUMBERG R KWT 4-	1	6,97	1	6,97
BASE UNEX 1256 ADHESIVA 30 X 30 MM	1	1,32	1	1,32
CINTA ESPIRAL CEN 12	1	2,60	1	2,60
D-Link DES-1005D	1	35,00	1	35,00
Materiales para conexiones		275,72		
TOTAL				8.433,32

El presupuesto total del proyecto es de ocho mil cuatrocientos treinta y tres € con treinta y dos céntimos sin IVA.

En Santiago, el 1 de agosto de 2019



Daniel Lamas Novoa



Escuela de Ingeniería Industrial

TRABAJO FIN DE MÁSTER

*Diseño de una estación de despaletizado mediante un sistema
de visión robótica*

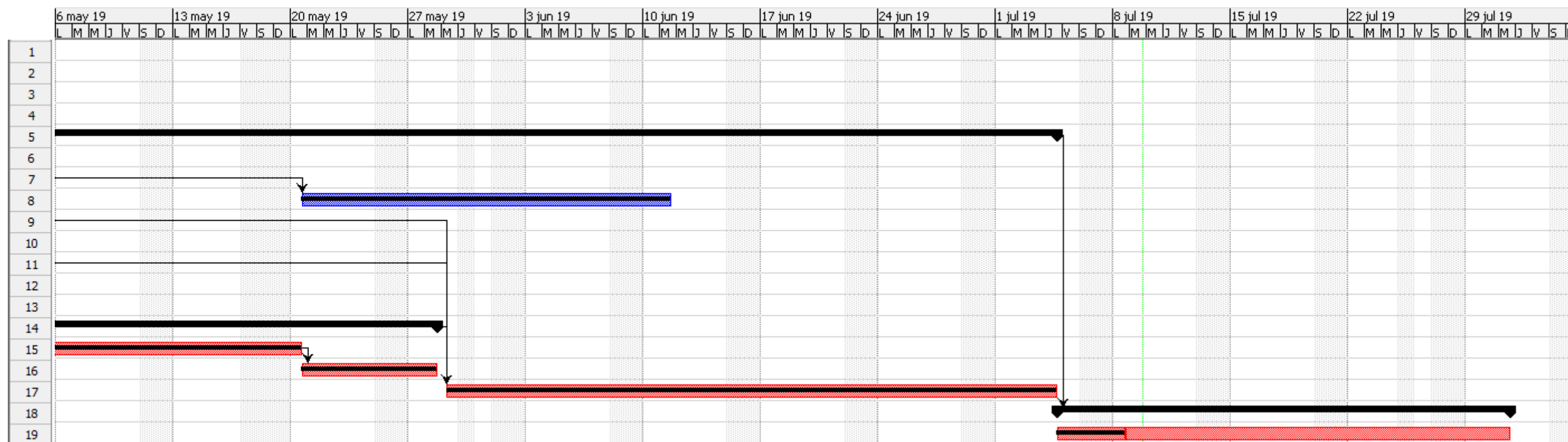
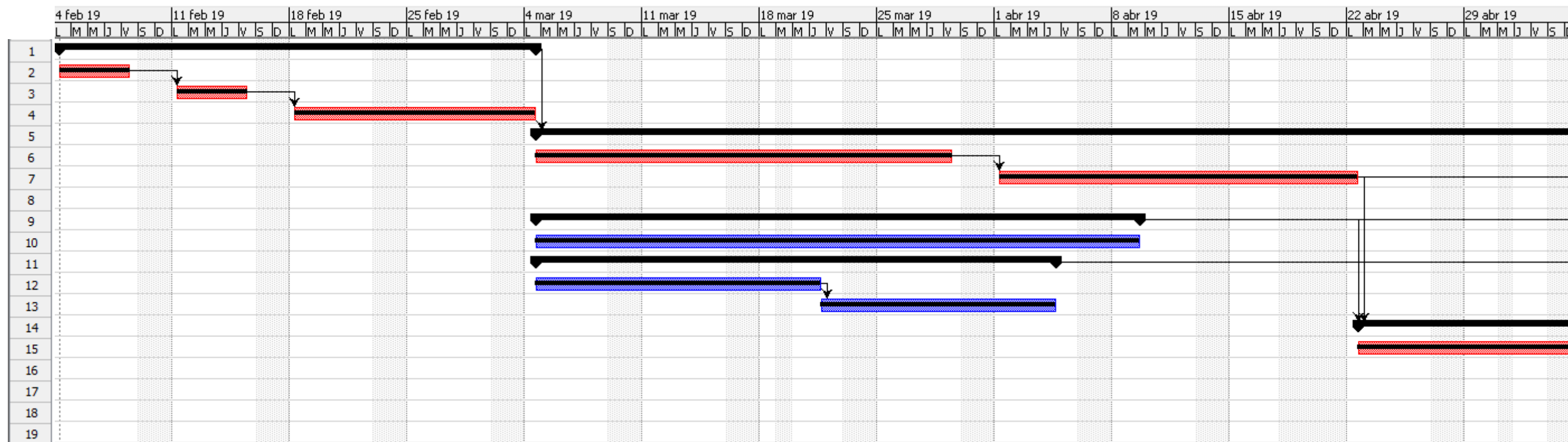
Máster en ingeniería industrial

Documento

DIAGRAMA DE GANTT

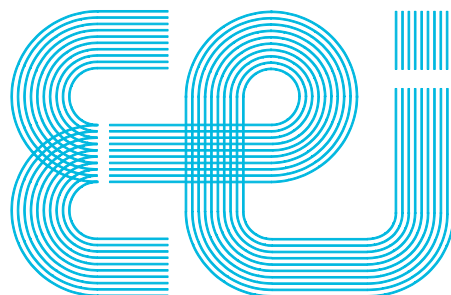
UniversidadeVigo

Nombre		Duración (días)	Inicio	Terminado
1	Inicio	21	04/02/19	04/03/19
2	Jornada de presentación	5	04/02/19	08/02/19
3	Estudio en campo	5	11/02/19	15/02/19
4	Arquitectura y presupuesto	11	18/02/19	04/03/19
5	Desarrollo	83	04/03/19	04/07/19
6	Elección y compra de herramientas	18	04/03/19	29/03/19
7	Diseño garra	14	01/04/19	22/04/19
8	Montaje físico	15	20/05/19	11/06/19
9	Visión artificial	25	04/03/19	09/04/19
10	Desarrollo software	25	04/03/19	09/04/19
11	Robot	22	04/03/19	04/04/19
12	Desarrollo virtual	12	04/03/19	21/03/19
13	Desarrollo en el robot del laboratorio	10	21/03/19	04/04/19
14	PLC	24	22/04/19	28/05/19
15	Comunicaciones	18	22/04/19	20/05/19
16	Programa principal	6	20/05/19	28/05/19
17	Calibraciones y pruebas	25	29/05/19	04/05/19
18	Cierre	18	04/07/19	31/07/19
19	Redacción de documentos	18	04/07/19	31/07/19



En Santiago, el 1 de agosto de 2019

Daniel Lamas Novoa



Escuela de Ingeniería Industrial

TRABAJO FIN DE MÁSTER

*Diseño de una estación de despaletizado mediante un sistema
de visión robótica*

Máster en ingeniería industrial

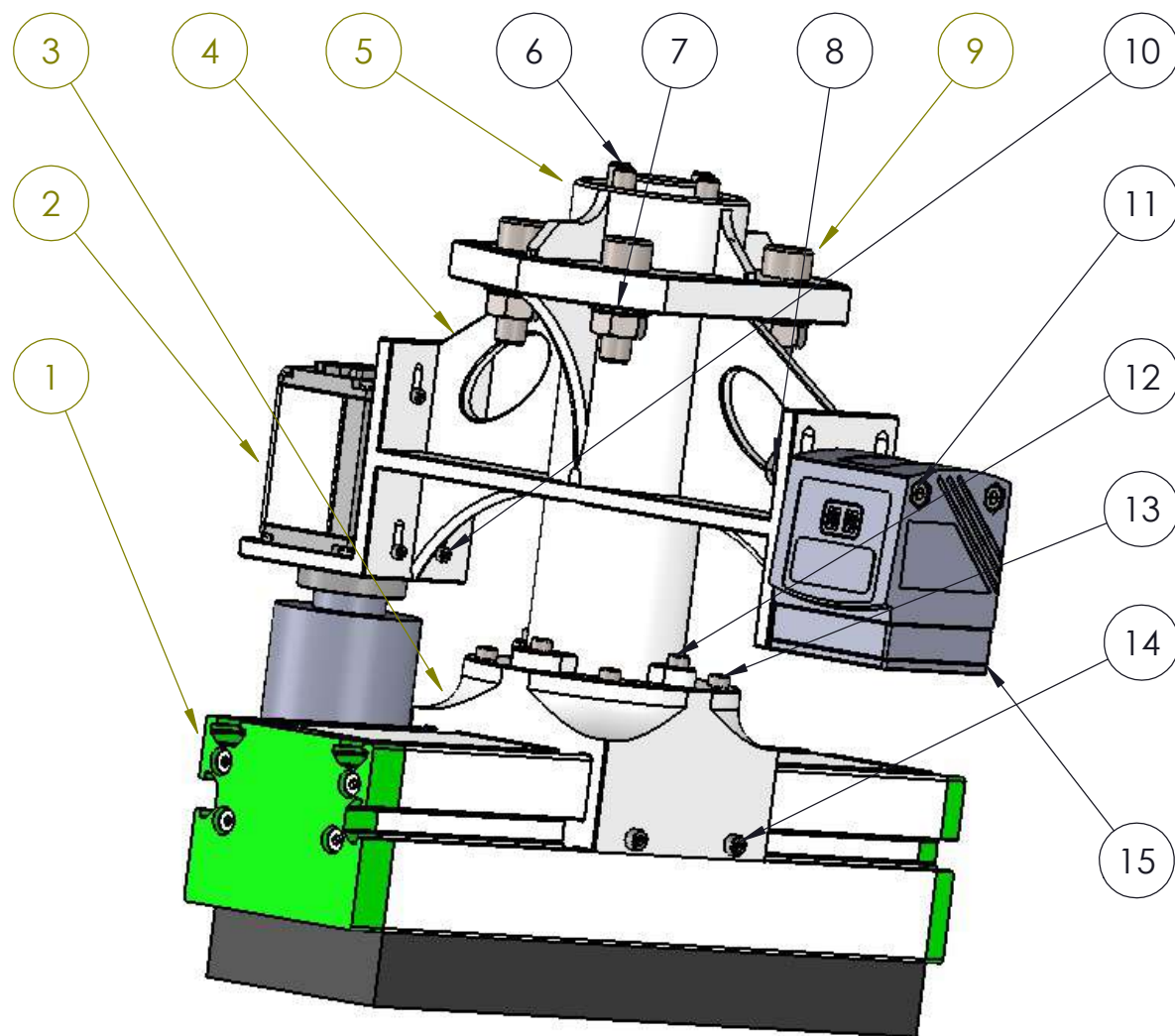
Documento

PLANOS



UniversidadeVigo

ÍNDICE

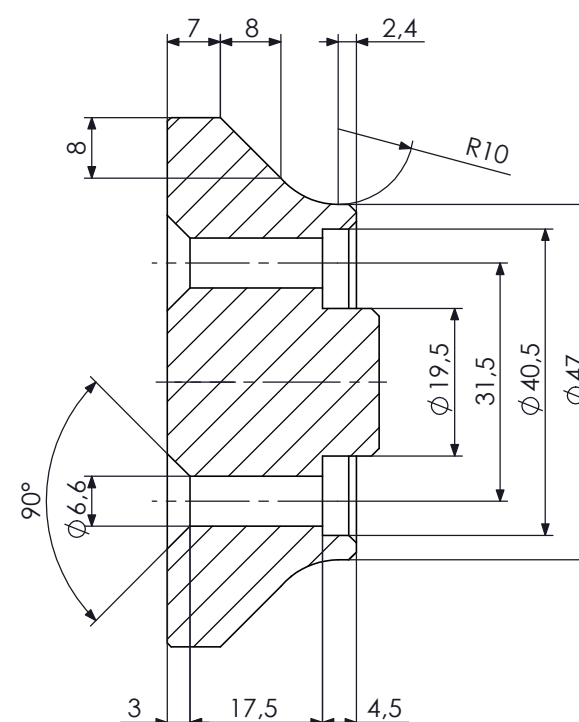
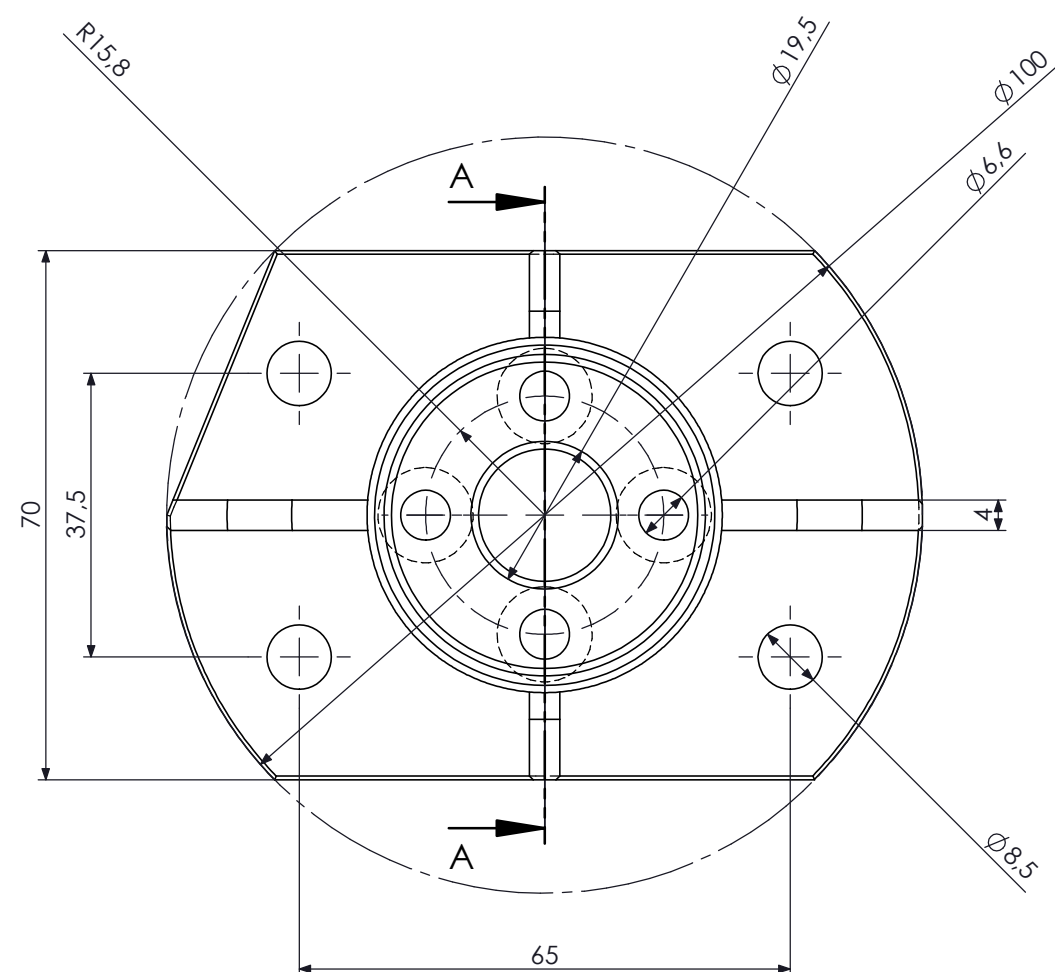
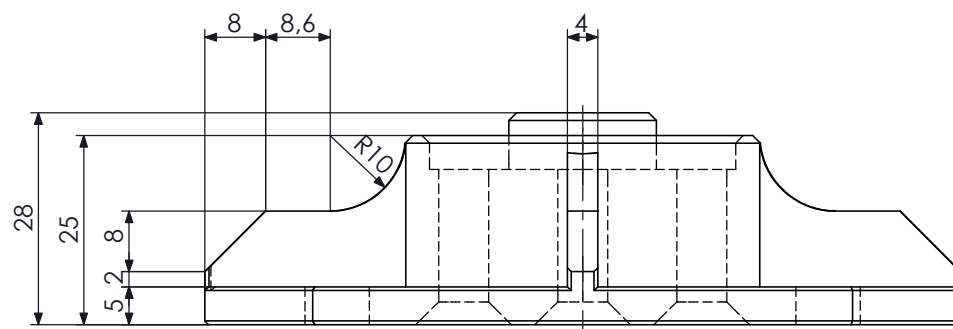
1. Ensamblaje
2. Brida
3. Unión cámara y sensor de distancia
4. Unión garra de vacío



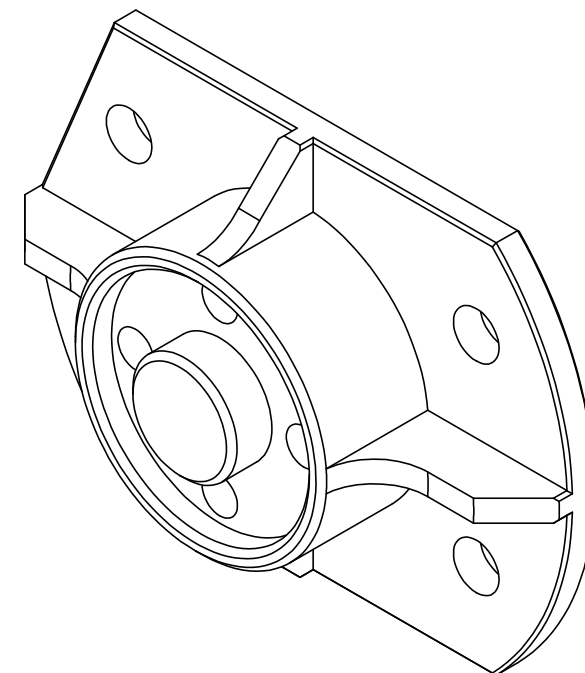
Nº ELEMENTO	PIEZA	CANTIDAD	PLANO
1	kenos KVG200.60	1	
2	iDS UI-5200SE Rev.4	1	
3	unión garra vacío	1	4
4	unión cámara y sensor de distancia	1	3
5	brida	1	2
6	DIN 7991 M6 x 30	4	
7	Hexagon Flange Nut DIN 6923 - M8	4	
8	Hexagon Flange Nut DIN 6923 - M4	2	
9	EN ISO 4762 M8 X 25	4	
10	EN ISO 4762 M2 x 6	4	
11	EN ISO 4762 M4 x 50	2	
12	EN ISO 4762 M4 x 10	4	
13	EN ISO 4762 M4 x 30	4	
14	EN ISO 4762 M4 x 20	4	
15	IFM OD100	1	


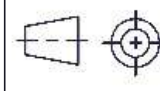
Nº PIEZAS 1		DESCRIPCIÓN Plano del conjunto de la garra del robot			
INGENIERO  Daniel Lamas Novoa		FECHA 27/06/2019		PROYECTO PROYECTO: Diseño de una estación de despaletizado mediante un sistema de visión robótica PETICIONARIO: E.E.I. Universidad de Vigo Financiera Maderera S.A.	
		ARCHIVO ensamblaje		PLANO Ensamblaje	
					
				ESCALA 1:2 Nº DE PLANO 1	



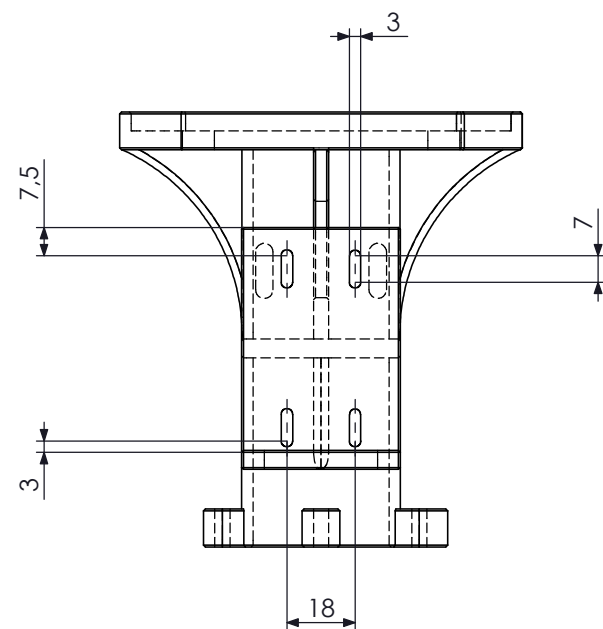
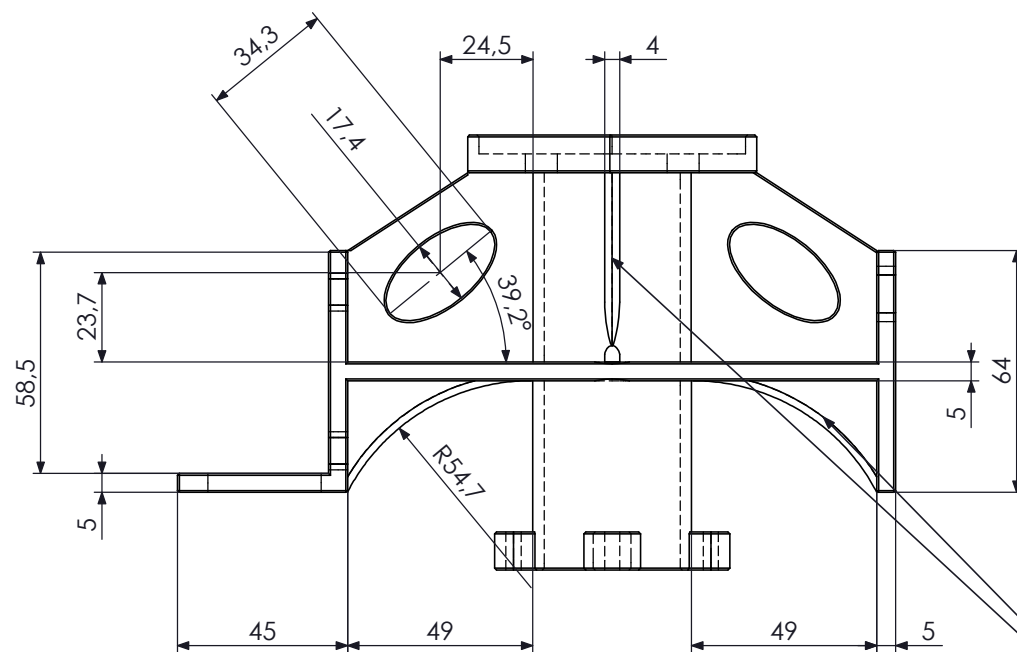
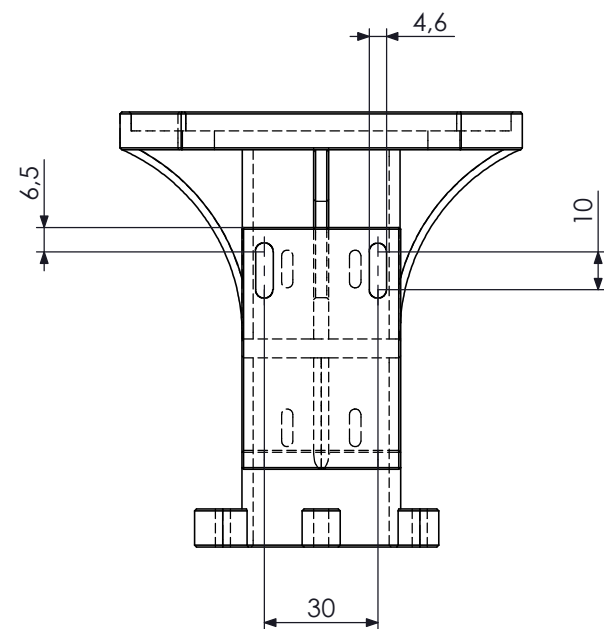
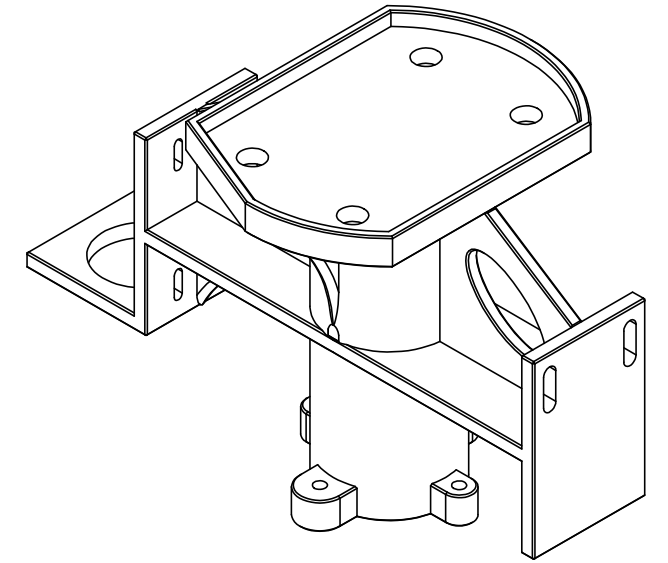
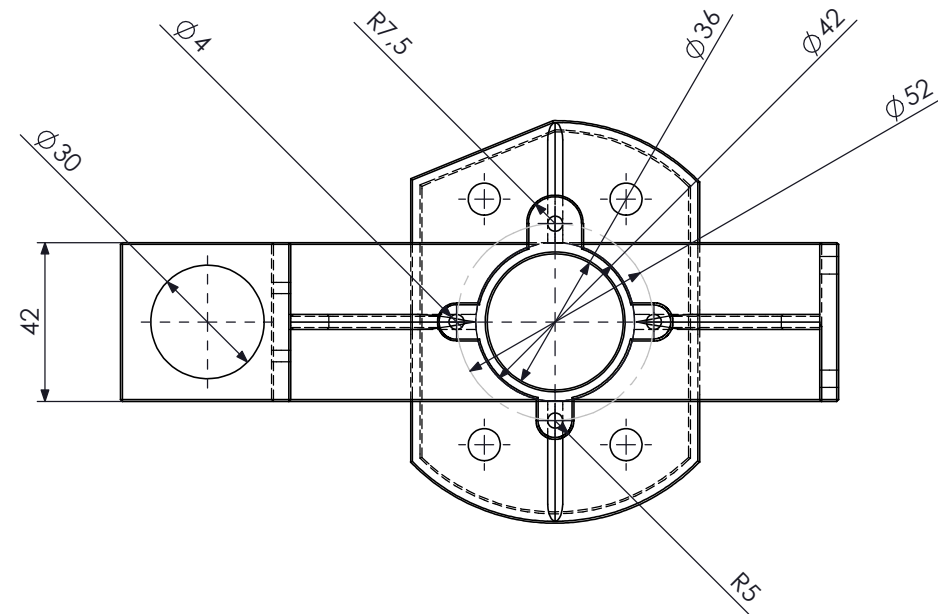


SECCIÓN A-A

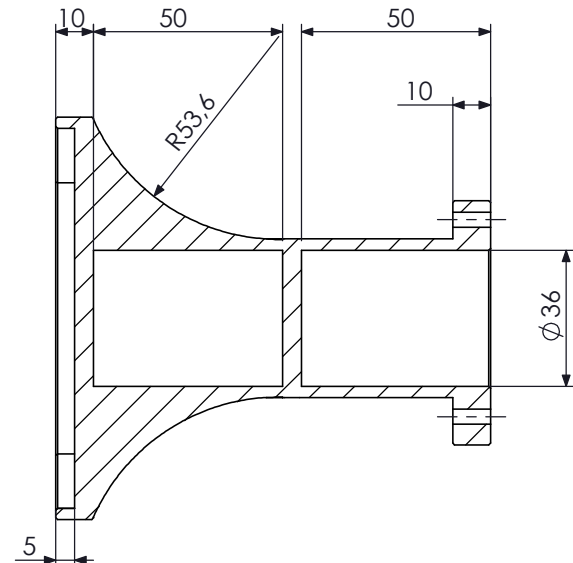


Nº PIEZAS 1	DESCRIPCIÓN Pieza de soporte de la cámara y el sensor láser	CHAFLANES 1x45°	UNIDADES mm y ángulos
INGENIERO  Daniel Lamas Novoa	FECHA 27/06/2019 	PROYECTO PROYECTO: Diseño de una estación de despaletizado mediante un sistema de visión robótica PETICIONARIO: E.E.I. Universidad de Vigo Financiera Maderera S.A.	
TOLERANCIAS ±0,1 mm	ARCHIVO union_camara	PLANO Brida	
MATERIAL PA12 - Poliamida 2200			ESCALA 1:1 Nº DE PLANO 2

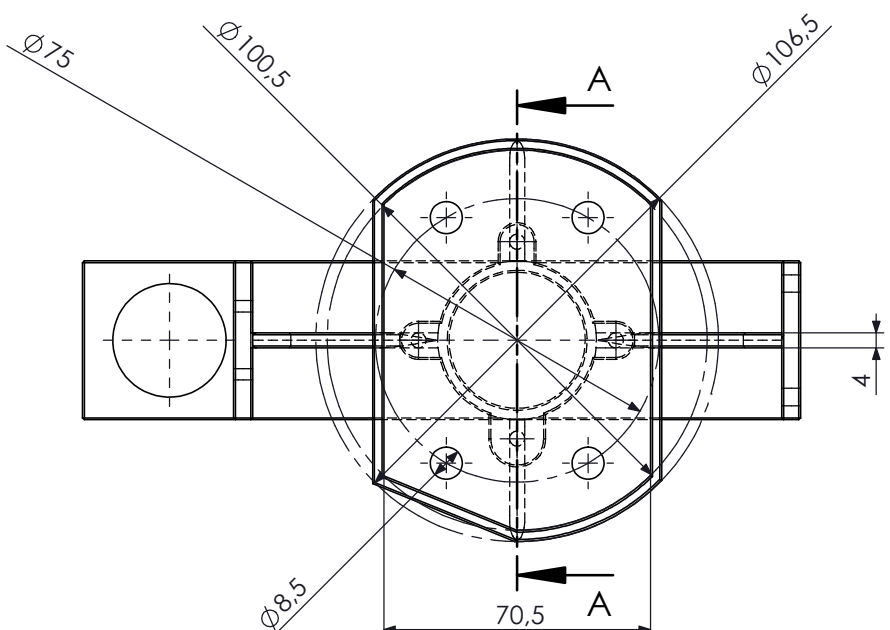


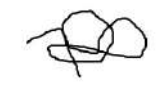
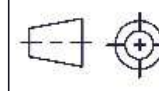



Reedondeo de nervios 2mm

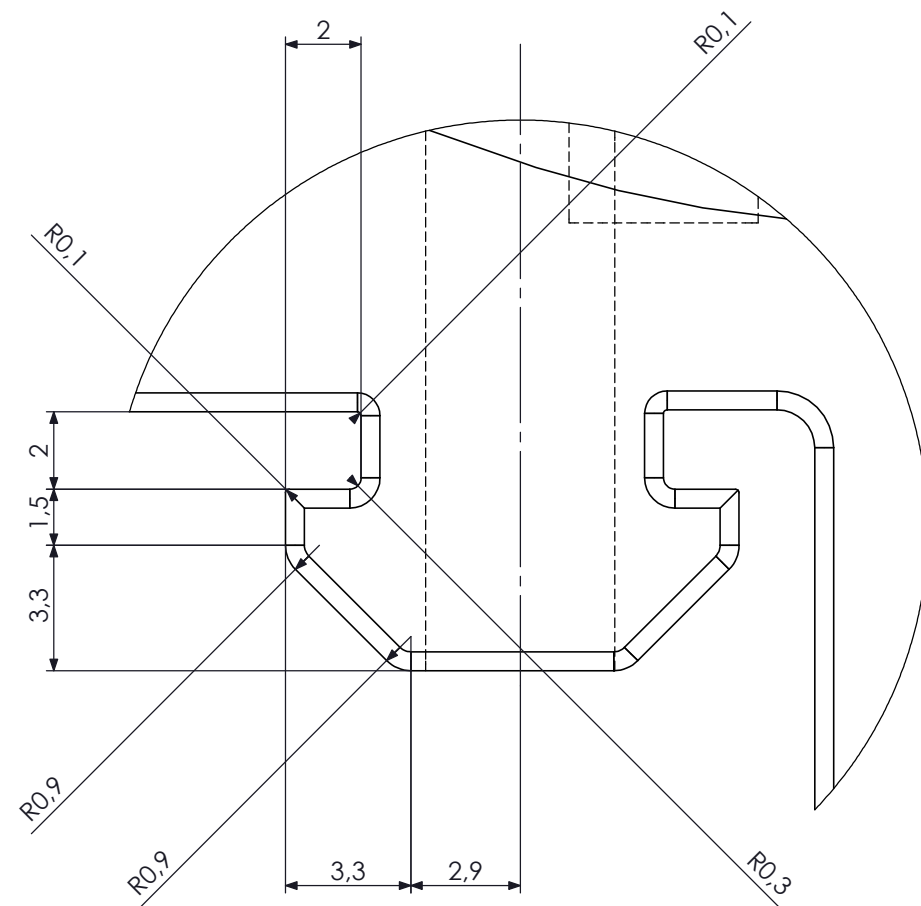
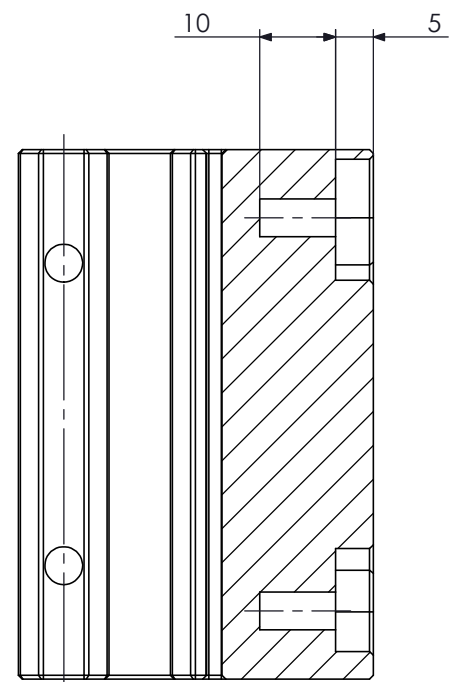



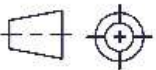


SECCIÓN A-A

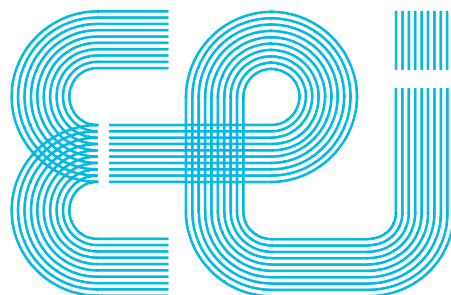


Nº PIEZAS 1	DESCRIPCIÓN Pieza de soporte de la cámara y el sensor láser	CHAFLANES 1x45°	UNIDADES mm y ángulos
INGENIERO  Daniel Lamas Novoa	FECHA 27/06/2019 	PROYECTO PROYECTO: Diseño de una estación de despaletizado mediante un sistema de visión robótica PETICIONARIO: E.E.I. Universidad de Vigo Financiera Maderera S.A.	
TOLERANCIAS ±0,1 mm	ARCHIVO union_camara	PLANO Unión camara y sensor de distancia 	
MATERIAL PA12 - Poliamida 2200	ESCALA 1:2		Nº DE PLANO 3





Nº PIEZAS 1	DESCRIPCIÓN Pieza de soporte de la garra de vacío	CHAFLANES 1x45°	UNIDADES mm y ángulos
INGENIERO  Daniel Lamas Nova	FECHA 27/06/2019 	PROYECTO PROYECTO: Diseño de una estación de despaletizado mediante un sistema de visión robótica PETICIONARIO: E.E.I. Universidad de Vigo Financiera Maderera S.A.	 Universidade de Vigo
TOLERANCIAS ±0,1 mm	ARCHIVO union_ventosa	PLANO Unión garra de vacío 	ESCALA 1:1 Nº DE PLANO 4
MATERIAL PA12 - Poliamida 2200			



Escuela de Ingeniería Industrial

TRABAJO FIN DE MÁSTER

*Diseño de una estación de despaletizado mediante un sistema
de visión robótica*

Máster en ingeniería industrial

Documento

ANEXO I: PROGRAMAS DE LA ESTACIÓN

UniversidadeVigo

ÍNDICE

1. Visión_artificial_despaletizado
2. Robot
 - 2.1. CalibData
 - 2.2. Funciones
 - 2.3. MainModule
 - 2.4. Points
3. PLC
 - 3.1. Actualizacion_entradas
 - 3.2. Actualizacion_salidas
 - 3.3. F_Vision_Artificial
 - 3.4. F_robot
 - 3.5. Main
 - 3.6. Modo_Funcionamiento

vision_artificial_despaletizado.hdev

main (: : :)

```
* CONEXION CON LA CAMARA
* Image Acquisition 01: Code generated by Image Acquisition 01
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', 8, 'default', -1, 'false',
'default', '1', 0, -1, AcqHandle)
set_framegrabber_param (AcqHandle, 'exposure', 64)
grab_image_start (AcqHandle, -1)

* datos de calibracion
* CameraParameters :=
['area_scan_division',0.0136259,-69.9262,3.45507e-006,3.45e-006,2016.29,1488.89,4104,30
06]
* CameraPose := [0.240531,-0.0155894,0.682759,1.28585,358.866,89.1027,0]

* calibracion := 240/1445.59
calibracion := 1/5.80935

* CONEXION CON EL PLC
* IP y puerto opc ua del PLC
MyOpcUaIP := '172.16.131.228'
MyOpcUaPortNumber := 4840
MyOpcUaServer := 'opc.tcp://' + MyOpcUaIP + ':' + MyOpcUaPortNumber

* Open the connection to the server
open_io_device ('OPC_UA', MyOpcUaServer, [], [], IoDeviceHandle)
*
*****
***
* Metodo de comunicaciones. Abrir canal --> leer/escribir variable --> cerrar canal
* Multiplico por 100 los datos en mm para guardarlos en el PLC como DInt y conservando
2 decimales

* abro comunicaciones
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."A"."inicializacion"', [], [],
inicializacion)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."A"."reconoce_pale"', [], [],
reconoce_pale)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."A"."reconoce_piezas"', [], [],
reconoce_piezas)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."A"."envia_datos"', [], [],
envia_datos)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."A"."prepara_datos"', [], [],
prepara_datos)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."A"."finaliza"', [], [],
finaliza)

open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."ESTADOS"."inicializado"',
[], [], inicializado)
open_io_channel (IoDeviceHandle,
'ns=3;s="OPC_VISION"."DE"."ESTADOS"."pale_reconocido"', [], [], pale_reconocido)
open_io_channel (IoDeviceHandle,
'ns=3;s="OPC_VISION"."DE"."ESTADOS"."piezas_reconocidas"', [], [], piezas_reconocidas)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."ESTADOS"."finalizado"', [],
[], finalizado)
```

```

open_io_channel (IoDeviceHandle,
'ns=3;s="OPC_VISION"."DE"."ESTADOS"."datos_actualizados"', [], [], datos_actualizados)
open_io_channel (IoDeviceHandle,
'ns=3;s="OPC_VISION"."DE"."ESTADOS"."datos_preparados"', [], [], datos_preparados)

open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."distancia_vision"', [], [],
distancia_vision)

open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PALE"."Y"', [], [], paleY)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PALE"."X"', [], [], paleX)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PALE"."C"', [], [], paleC)

open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PIEZAS".no_piezas', [], [],
no_piezas)

open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PIEZAS"."X"', [], [],
piezasX)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PIEZAS"."Y"', [], [],
piezasY)
open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PIEZAS"."C"', [], [],
piezasC)

open_io_channel (IoDeviceHandle, 'ns=3;s="OPC_VISION"."DE"."PIEZAS".numero', [], [],
numero_piezas)

while (true)

    * Captura imagen de la camara. Lo hago aqui fuera porque si lo hago despues de
    recibir una orden, no coge la imagen actualizada
    grab_image_async (Image, AcqHandle, -1)

    * dev_display (Image)
    * read_image (Image, 'C:/Users/DLN/Documents/Vision artificial/Imagenes/Tablas')
    * centro de la imagen. Las coordenadas de los puntos se envian relativos a este
    centro
    area_center (Image, Area1, cam_row, cam_colum)
    gen_cross_contour_xld (Cross_center, cam_row, cam_colum, 64, 0.5)
    dev_display (Cross_center)

    * Lectura ordenes del PLC
    * inicializacion

    read_io_channel (inicializacion, DI_inicializacion, Status)
    * close_io_channel (inicializacion)

    * reconoce_pale

    read_io_channel (reconoce_pale, DI_reconoce_pale, Status)
    * close_io_channel (reconoce_pale)

    * reconoce_piezas

    read_io_channel (reconoce_piezas, DI_reconoce_piezas, Status)
    * close_io_channel (reconoce_piezas)

    * envia_datos

    read_io_channel (envia_datos, DI_envia_datos, Status)
    * close_io_channel (envia_datos)

```



```

* prepara_datos

read_io_channel (prepara_datos, DI_prepara_datos, Status)
* close_io_channel (prepara_datos)

* finaliza

read_io_channel (finaliza, DI_finaliza, Status)
* close_io_channel (finaliza)

*
*****
*****

* el plc manda una orden. Tengo que poner en la comparacion 'true' porque es asi como
la guarda. Si pongo true o no pongo nada o 1 no funciona
if (DI_inicializacion='true' or DI_reconoce_pale='true' or DI_reconoce_piezas='true'
or DI_envia_datos='true' or DI_prepara_datos='true' or DI_finaliza='true')
    * Pongo a 0 todos los estados, indicandoselo al PLC
    grab_image_async (Image, AcqHandle, -1)
    * inicializado

    write_io_channel (inicializado, 0, Status)
    * close_io_channel (inicializado)

    * pale_reconocido

    write_io_channel (pale_reconocido, 0, Status)
    * close_io_channel (pale_reconocido)

    * piezas_reconocidas

    write_io_channel (piezas_reconocidas, 0, Status)
    * close_io_channel (piezas_reconocidas)

    * finalizado

    write_io_channel (finalizado, 0, Status)
    * close_io_channel (finalizado)

    * datos_actualizados

    write_io_channel (datos_actualizados, 0, Status)
    * close_io_channel (datos_actualizados)

    * datos_preparados

    write_io_channel (datos_preparados, 0, Status)
    * close_io_channel (datos_preparados)

*
*****
*****

if (DI_inicializacion='true')

    * parte grafica
    dev_clear_window ()
    * ajuste el tamaño de la pantalla
    * dev_resize_window_fit_image (Image, 0, 0, -1, -1)

```

```

* inicio unos vectores para usarlos en reconocer piezas
tuple_gen_const (0, 0, filas)
tuple_gen_const (0, 0, cols)
tuple_gen_const (0, 0, orient)

* distancia a la que poisiocnar la camara

write_io_channel (distancia_vision, 591*100, Status)
* close_io_channel (distancia_vision)

* COMUNICO CON EL PLC.Pongo a 1 inicializado

write_io_channel (inicializado, 1, Status)
* close_io_channel (inicializado)

elseif (DI_reconoce_pale='true')
    grab_image_async (Image, AcqHandle, -1)
    dev_clear_window ()

    * Busco el pale
    Reconocer_pale (Image, pale, pale_row1, pale_column1, pale_row, pale_colum,
pale_angulo)
    dev_display (Image)
    dev_display (Cross_center)
    Dibujar_contorno (pale)

    * coordenadas con respecto a la camara en mm
    * image_points_to_world_plane (CameraParameters, CameraPose, pale_row,
pale_colum, 'mm', World_Row_pale, World_Col_pale)
    pale_row := pale_row-cam_row
    pale_colum := pale_colum-cam_colum

    World_Row_pale := calibracion*pale_row
    World_Col_pale := calibracion*pale_colum

    * COMUNICACOINES CON EL PLC
    * pale_row

    write_io_channel (paleY, World_Row_pale*100, Status)
    * close_io_channel (paleY)

    * pale_colum

    write_io_channel (paleX, World_Col_pale*100, Status)
    * close_io_channel (paleX)

    * angulo pale

    write_io_channel (paleC, -pale_angulo*100, Status)
    * close_io_channel (paleC)

    * pongo a 1 pale_reconocido

    write_io_channel (pale_reconocido, 1, Status)
    * close_io_channel (pale_reconocido)

elseif (DI_reconoce_piezas='true')
    grab_image_async (Image, AcqHandle, -1)
    dev_clear_window ()

```

```

* Busco el pale
Reconocer_pale (Image, pale, pale_row1, pale_column1, pale_row2, pale_column2,
phi)
dev_display (Image)
* reduzco la zona de busqueda al pale. ROI en el pale
reduce_domain (Image, pale, ImageReduced)

* BUSCO LAS PIEZAS
* metodos para el reconocimiento de piezas
regiongrowing (ImageReduced, piezas, 1, 1, 3, 100000)
* regiongrowing_n (ImageReduced, Regions3, 'n-norm', 0.1, 6, 400000)
* regiongrowing_n (ImageReduced, Regions4, 'n-norm', 1, 9, 400000)

* filtra por forma
select_shape (piezas, piezas, 'rectangularity', 'and', 0.7, 1)
* filtro por tamaño
* select_shape (piezas, piezas, 'area', 'and', 100000, 1000000)
* cuenta las piezas
count_obj (piezas, num_piezas)

* pongo a 0 los vectores en los que se guradaron las filas y columnas de los
centros
tuple_gen_const (0, 0, filas)
tuple_gen_const (0, 0, cols)
tuple_gen_const (0, 0, orient)

* recorre todas las piezas. Dibuja su contorno y guarda su centro
for Index := 1 to num_piezas by 1

    * selecciono una pieza del vector de regiones
    select_obj (piezas, ObjectSelected, Index)
    * la rectangularizo. Esto lo hago porque se que son rectangulares. Así tengo
menos error en el calculo de su centro
    shape_trans (ObjectSelected, pieza_rect, 'rectangle2')
    * Busco su centro
    area_center (pieza_rect, Area, Row, Column)
    * Y su orientacion
    orientation_region (pieza_rect, radianes_medidos)
    grados_medidos := radianes_medidos*360/(2*3.141592)
    if (grados_medidos > 90)
        Phi := grados_medidos-180
    elseif (grados_medidos<-90)
        Phi := grados_medidos+180
    else
        Phi := grados_medidos
    endif

    * coordenadas con respecto a la camara en mm
    * image_points_to_world_plane (CameraParameters, CameraPose, Row, Column, 'mm',
World_Row_pieza, World_Col_pieza)

    Row := Row-cam_row
    Column := Column-cam_column

    World_Row_pieza := calibracion*Row
    World_Col_pieza := calibracion*Column

```

```

        * guardo la posicion del centro en los vectores

        tuple_insert (filas, 0, World_Row_pieza, filas)
        tuple_insert (cols, 0, World_Col_pieza, cols)
        tuple_insert (orient, 0, Phi, orient)

        Dibujar_contorno (pieza_rect)

    endfor

    * Pongo el valor del indice en 0. Esto se usa para ir mandando los datos de los
    vectores
    I := 0

    * indico si se reconocio alguna pieza o si no hay ninguna
    if (num_piezas>0)
        * numero piezas que faltan por sacar

        write_io_channel (no_piezas, 0, Status)
        * close_io_channel (no_piezas)
    else

        write_io_channel (no_piezas, 1, Status)
        * close_io_channel (no_piezas)
    endif

    * COMUNICACION CON EL PLC
    * pongo a 1 piezas_reconocidas

    write_io_channel (piezas_reconocidas, 1, Status)
    * close_io_channel (piezas_reconocidas)

elseif (DI_prepara_datos='true')

    * Resto 1 a las piezas que faltan por enviar
    num_piezas := num_piezas-1

    * actualizo el indice del vector
    I := I+1

    * pongo a 1 datos_preparados

    write_io_channel (datos_preparados, 1, Status)
    * close_io_channel (datos_preparados)

elseif (DI_envia_datos='true')

    * si quedan piezas por mandar mando la siguiente
    if (num_piezas>0)

        * pieza_row

        write_io_channel (piezasX, cols[I]*100, Status)
        * close_io_channel (piezasX)

        * pieza_colum

        write_io_channel (piezasY, filas[I]*100, Status)
        * close_io_channel (piezasY)

        * pieza_angulo

```

```

    write_io_channel (piezasC, -orient[I]*100, Status)
    * close_io_channel (piezasC)

    * sino quedan piezas por mandar no mando nada
else
endif

* pongo a 1 datos_actualizados

write_io_channel (datos_actualizados, 1, Status)
* close_io_channel (datos_actualizados)

* numero piezas que faltan por sacar

write_io_channel (numero_piezas, num_piezas, Status)
* close_io_channel (numero_piezas)

elseif (DI_finaliza='true')

    * pongo a 1 finalizado

    write_io_channel (finalizado, 1, Status)
    * close_io_channel (finalizado)
endif

endif

endwhile

* Cierra comunicacion con el PLC y con la camara
* Esto solo se ejuctaria al salir del bucle. Nunca se sale de el.
close_io_device (IoDeviceHandle)
close_framegrabber (AcqHandle)

```

Procedimientos utilizados

dev_resize_window_fit_image (Image : : Row, Column, WidthLimit, HeightLimit :)

Breve Descripción: Changes the size of a graphics window with a given maximum and minimum extent such that it preserves the aspect ratio of the given image

Capítulos: Develop

Público Procedimiento externo: C:/Program
Files/MVTEC/HALCON-13.0/procedures/general/dev_resize_window_fit_image.hdevp

```

* This procedure adjusts the size of the current window
* such that it fits into the limits specified by WidthLimit
* and HeightLimit, but also maintains the correct image aspect ratio.
*

```

```

* If it is impossible to match the minimum and maximum extent requirements
* at the same time (f.e. if the image is very long but narrow),
* the maximum value gets a higher priority,
*
* Parse input tuple WidthLimit
if (|WidthLimit| == 0 or WidthLimit < 0)
    MinWidth := 500
    MaxWidth := 800
elseif (|WidthLimit| == 1)
    MinWidth := 0
    MaxWidth := WidthLimit
else
    MinWidth := WidthLimit[0]
    MaxWidth := WidthLimit[1]
endif
* Parse input tuple HeightLimit
if (|HeightLimit| == 0 or HeightLimit < 0)
    MinHeight := 400
    MaxHeight := 600
elseif (|HeightLimit| == 1)
    MinHeight := 0
    MaxHeight := HeightLimit
else
    MinHeight := HeightLimit[0]
    MaxHeight := HeightLimit[1]
endif
*
* Test, if window size has to be changed.
ResizeFactor := 1
get_image_pointer1 (Image, Pointer, Type, ImageWidth, ImageHeight)
* First, expand window to the minimum extents (if necessary).
if (MinWidth > ImageWidth or MinHeight > ImageHeight)
    ResizeFactor := max([real(MinWidth) / ImageWidth, real(MinHeight) / ImageHeight])
endif
TempWidth := ImageWidth * ResizeFactor
TempHeight := ImageHeight * ResizeFactor
* Then, shrink window to maximum extents (if necessary).
if (MaxWidth < TempWidth or MaxHeight < TempHeight)
    ResizeFactor := ResizeFactor * min([real(MaxWidth) / TempWidth, real(MaxHeight) / TempHeight])
endif
WindowWidth := ImageWidth * ResizeFactor
WindowHeight := ImageHeight * ResizeFactor
* Resize window
dev_set_window_extents (Row, Column, WindowWidth, WindowHeight)
dev_set_part (0, 0, ImageHeight - 1, ImageWidth - 1)
return ()

```

Reconocer_pale (Image : pale_rect :: Row1, column1, Row, Column, Phi)

Breve Descripción:

Capítulos:

Procedimiento local

```

* con esto funciona pero es mas rapido por grises
* regiongrowing (Image, pale_rect, 10, 10, 5, 1000000)
* busca el pale por nivel de gris ya que el fondo es negro
threshold (Image, Regions, 70, 255)

```

```

* abro de forma rectangular
opening_rectangle1 (Regions, pale_rect, 100, 100)

* rectangularizo la forma detectada
shape_trans (pale_rect, pale_rect, 'rectangle2')

* calculo sus medidas
area_center (pale_rect, Area, Row, Column)

orientation_region (pale_rect, radianes_medidos)
grados_medidos := radianes_medidos*360/(2*3.141592)
if (grados_medidos > 90)
    Phi := grados_medidos-180
elseif (grados_medidos<-90)
    Phi := grados_medidos+180
else
    Phi := grados_medidos
endif

smallest_rectangle1 (pale_rect, Row1, Column1, Row2, Column2)

return ()

```

Dibujar_contorno (region : : :)

Breve Descripción:
Capítulos:
Procedimiento local

```

*
*
* orientation_region (region, Phi)
* area_center (region, Area, Row, Column)
* gen_contour_region_xld (region, Contours, 'border')

* me quedo con sus contorno
gen_contour_region_xld (region, Contours, 'border')
* busco el centro
area_center (region, Area, Row, Column)
* genero una X en el centro
gen_cross_contour_xld (Cross, Row, Column, 64, 0.5)
*
* Ahora dibujo
* modifico el ancho de linea
dev_set_line_width (4)
* cambio el color del contorno
dev_set_color ('red')
* dibjo el contorno
dev_display (Contours)
* cambio el color de la X
dev_set_color ('green')
* dibujo la X
dev_display (Cross)
*
*

```



```
return ()
```

```

MODULE CalibData
  !POSICION DE CARGA Y DESCARGA
  TASK PERS wobjdata pale:=[FALSE,TRUE,"",[[344.8,116.872,21.612],
[0.000039496,-0.000013233,-1,0.000083955]],[[0,0,0],[1,0,0,0]]];    !posicion pale leida por la
camara
  TASK PERS wobjdata descarga:=[FALSE,TRUE,"",[[346.861,115.463,54.8953],
[2.87687E-05,4.15206E-05,1,-1.0278E-05]],[[0,0,0],[1,0,0,0]]];    !datos del lugar en el que se
descargan las piezas

  !PIEZA
  TASK PERS loaddata pieza:=[1,[10,0,0],[1,0,0,0],0,0,0];    !datos de la pieza

  !HERRAMIENTAS. Todas las herramientas estan montadas a la vez. Los unicos datos que varian entre
ellas son los de la poiscion del tcp
  TASK PERS tooldata ventosa:=[TRUE,[[2.50494,-0.694668,218.922],
[0.924843,-0.00331838,-0.00807098,-0.380249]],[1,[0,0,1],[1,0,0,0],0,0,0]];    !datos de la
herramienta
  TASK PERS tooldata camara:=[TRUE,[[ -64.8213,-67.6793,156.417],
[0.924843,-0.00331838,-0.00807098,-0.380249]],[1,[0,0,1],[1,0,0,0],0,0,0]];    !datos de la camara
  TASK PERS tooldata sensor:=[TRUE,[[74.2693,63.2177,111.275],
[0.924843,-0.00331838,-0.00807098,-0.380249]],[1,[0,0,1],[1,0,0,0],0,0,0]];    !datos de la sensor

  !TASK PERS tooldata sensor:=[TRUE,[[68.2693,68.2177,111.275],[0.91706,0,0,-0.398749]],[1,
[0,0,1],[1,0,0,0],0,0,0]];    !datos de la sensor

  !PARAMETROS
  CONST speeddata vel_rapido:=v800;    !velocidad rapida
  CONST speeddata vel_lento:=v100;    !velocidad lento
  CONST zonedata zona:=z10;    !ajuste puntos

  !DATOS
  VAR num sentido:=1;
  VAR num margen:=50; !distancia entre los puntos coger/dejar y sus superiores en mm
  VAR num espuma:=17;    !ancho de la espuma
  VAR num tiempo_espera:=1;    !tiempo en segundos que el robot espera para que la ventosa cargue o
descargue la pieza
ENDMODULE

```

MODULE Funciones

!sacados de la camara

!distancias en mm y angulos en grados. NO SE SI LOS MILIMETROS NO LOS MARCA

BIEN????????????????

```
VAR num I_pale_x:=0;
VAR num I_pale_y:=0;
VAR num I_pale_z:=0;
VAR num I_pale_a:=0;
VAR num I_pale_b:=0;
VAR num I_pale_c:=0;
```

!pieza

```
VAR num I_pieza_x:=0;
VAR num I_pieza_y:=0;
VAR num I_pieza_z:=0;
VAR num I_pieza_a:=0;
VAR num I_pieza_b:=0;
VAR num I_pieza_c:=0;
```

```
VAR num inicializacion:=0;
VAR num sensor_pale_ini:=0;
VAR num camara_pale_ini:=0;
VAR num sensor_pale:=0;
VAR num camara_pale:=0;
VAR num sensor_pieza:=0;
VAR num coge_pieza:=0;
VAR num mueve_pieza:=0;
VAR num deja_pieza:=0;
VAR num finaliza:=0;
```

PROC Iniciar() !incorporo los datos de arranque que el PLC envíe

```
GripLoad load0;
visualizacion:=visualizacion_ini;
sentido:=1;
MoveJ home,vel_rapido,zona,ventosa\WObj:=wobj0;
```

ENDPROC

PROC Posicionar_sensor_pale_ini()

MoveJ visualizacion_ini,vel_rapido,fine,sensor\WObj:=pale; !me posiciono para buscar el pale con el sensor. Con el dato del sensor corrijo altura

WaitTime tiempo_espera; !espero para que el valor del sensor se estabilice

ENDPROC

PROC Posicionar_sensor()

MoveJ visualizacion,vel_rapido,fine,sensor\WObj:=pale; !me posiciono en el punto de visualizacion con el sensor. No hago movimientos relativos porque el punto está actualizado

WaitTime tiempo_espera; !espero para que el valor del sensor se estabilice

ENDPROC

PROC Posicionar_camara()

MoveJ RelTool (visualizacion, I_pale_x, I_pale_y, I_pale_z \Rx:=I_pale_a \Ry:=I_pale_b \Rz:=I_pale_c),vel_rapido,fine,camara\WObj:=pale; !me posiciono para ver las piezas. Corrijo datos de vision

visualizacion:=CROBT(\Tool:=camara \WObj:=pale); !actualizo el punto

WaitTime 2*tiempo_espera; !espero para que el valor del sensor se estabilice

ENDPROC

```
PROC Ir_centro_pieza() !voy al centro de la pieza con el sensor a la altura de visualizacion,
por eso el parametro de avance en Z está a 0
    MoveJ RelTool (visualizacion, I_pieza_x, I_pieza_y, 0 \Rx:=I_pieza_a \Ry:=I_pieza_b
\Rz:=I_pieza_c), vel_rapido,fine,sensor\WObj:=pale; !voy al centro de la pieza a la altura de vision
para guardar a que altura esta cada pieza
    WaitTime tiempo_espera; !espero para que el valor del sensor se estabilice
ENDPROC
```

```
PROC Coger_pieza()
    MoveJ RelTool (visualizacion, I_pieza_x, I_pieza_y, I_pieza_z - margen \Rx:=I_pieza_a
\Ry:=I_pieza_b \Rz:=I_pieza_c), vel_rapido,zona,ventosa\WObj:=pale; !voy encima de la pieza a una
diferencia de altura de margen
    MoveL RelTool (visualizacion, I_pieza_x, I_pieza_y, I_pieza_z + espuma \Rx:=I_pieza_a
\Ry:=I_pieza_b \Rz:=I_pieza_c), vel_lento,fine,ventosa\WObj:=pale; !bajo hasta la pieza la distancia
margen en Z
    !WaitTime tiempo_espera;
ENDPROC
```

```
PROC Mover_pieza()
    GripLoad pieza; !le indico al robot que ahora lleva la pieza. Necesario para que calcule la
dinamica teniendo en cuenta la pieza
    MoveL RelTool (visualizacion, I_pieza_x, I_pieza_y, I_pieza_z - margen \Rx:=I_pieza_a
\Ry:=I_pieza_b \Rz:=I_pieza_c), vel_lento,zona,ventosa\WObj:=pale; !voy encima de la pieza a una
diferencia de altura de margen
    MoveJ esquivar_iluminacion,vel_rapido,zona,ventosa\WObj:=pale;!paso por este punto para no
chocar con la iluminacion
    MoveJ Offs(deja,0,0,-margen),vel_rapido,zona,ventosa\WObj:=descarga; !voy a un punto por
encima del punto de descarga
    MoveL deja,vel_lento,fine,ventosa\WObj:=descarga; !bajo al punto de descarga
    !WaitTime tiempo_espera;
ENDPROC
```

```
PROC Dejar_pieza()
    GripLoad load0; !le indico al robot que descargue
    MoveJ Offs(deja,0,0,-margen),vel_lento,zona,ventosa\WObj:=descarga; !voy a un punto por
encima del punto de descarga
    MoveJ esquivar_iluminacion,vel_rapido,zona,ventosa\WObj:=pale;!paso por este punto para no
chocar con la iluminacion
ENDPROC
```

```
PROC Fin()
    MoveJ home,vel_rapido,zona,ventosa\WObj:=wobj0;
ENDPROC
```

```
!lectura de variables
FUNC dnum leerInt32(dnum value)
    IF value > 2147483647 THEN
        RETURN (value-4294967296);
    ELSE
        RETURN value;
    ENDIF
ENDFUNC
```

ENDMODULE

```

MODULE MainModule
  PROC main()

    !datos de la calibracion de la orientacion de las herramientas
    !camara.tframe.rot:=OrientZYX(-44.7,-1,0);
    !sensor.tframe.rot:=OrientZYX(-44.7,-1,0);
    !ventosa.tframe.rot:=OrientZYX(-44.7,-1,0);

    IF DI_inicializacion=1 OR DI_sensor_pale_ini=1 OR DI_camara_pale_ini=1 OR DI_sensor_pale=1
    OR DI_camara_pale=1 or DI_sensor_pieza=1 OR DI_coge_pieza=1 OR DI_mueve_pieza=1 OR DI_deja_pieza=1
    OR DI_finaliza=1 THEN !reseteo todas las variables de salida

      !Guardo los estados de las variables por si varian en medio de un ciclo
      inicializacion:=DI_inicializacion;
      sensor_pale_ini:=DI_sensor_pale_ini;
      camara_pale_ini:=DI_camara_pale_ini;
      sensor_pale:=DI_sensor_pale;
      camara_pale:=DI_camara_pale;
      sensor_pieza:=DI_sensor_pieza;
      coge_pieza:=DI_coge_pieza;
      mueve_pieza:=DI_mueve_pieza;
      deja_pieza:=DI_deja_pieza;
      finaliza:=DI_finaliza;

    !Pongo a 0 todas las salidas
    Reset DO_inicializado;
    Reset DO_sensor_pale_ini;
    Reset DO_camara_pale_ini;
    Reset DO_sensor_pale;
    Reset DO_camara_pale;
    Reset DO_sensor_pieza;
    Reset DO_pieza_cogida;
    Reset DO_pieza_movida;
    Reset DO_pieza_dejada;
    Reset DO_finalizado;

    I_pieza_x:=DnumToNum(LeerInt32(GInputDnum(GI_pieza_x)))/100;
    I_pieza_y:=DnumToNum(LeerInt32(GInputDnum(GI_pieza_y)))/100;
    I_pieza_z:=DnumToNum(LeerInt32(GInputDnum(GI_pieza_z)))/100;
    I_pieza_a:=DnumToNum(LeerInt32(GInputDnum(GI_pieza_a)))/100;
    I_pieza_b:=DnumToNum(LeerInt32(GInputDnum(GI_pieza_b)))/100;
    I_pieza_c:=DnumToNum(LeerInt32(GInputDnum(GI_pieza_c)))/100;

    I_pale_x:=DnumToNum(LeerInt32(GInputDnum(GI_pale_x)))/100;
    I_pale_y:=DnumToNum(LeerInt32(GInputDnum(GI_pale_y)))/100;
    I_pale_z:=DnumToNum(LeerInt32(GInputDnum(GI_pale_z)))/100;
    I_pale_a:=DnumToNum(LeerInt32(GInputDnum(GI_pale_a)))/100;
    I_pale_b:=DnumToNum(LeerInt32(GInputDnum(GI_pale_b)))/100;
    I_pale_c:=DnumToNum(LeerInt32(GInputDnum(GI_pale_c)))/100;

    IF inicializacion=1 THEN
      Iniciar;
      Set DO_inicializado;
    
```

```

ELSEIF sensor_pale_ini=1 THEN !pone el sensor sobre donde supone que debe estar el pale
    Posicionar_sensor_pale_ini;
    Set DO_sensor_pale_ini;      !indica al PLC que la camara esta en posicion

ELSEIF camara_pale_ini=1 THEN      !pone la camara a la altura correcta para buscar el pale
    Posicionar_camara;
    Set DO_camara_pale_ini;

ELSEIF camara_pale=1 THEN      !pone la camara sobre el pale esta vez con los datos de su
centro
    Posicionar_camara;
    Set DO_camara_pale;

ELSEIF sensor_pieza=1 THEN      !pone el sensor sobre el pale
    Ir_centro_pieza;
    Set DO_sensor_pieza;

ELSEIF coge_pieza=1 THEN !coge la pieza
    Coger_pieza;
    Set DO_pieza_cogida;
    !WaitTime 0.1*tiempo_espera; !espero a que la ventosa se active y espero a que se
hagan las comunicaciones con el PC. Sino, el PLC enciende y apaga una orden demasiado rapido

ELSEIF mueve_pieza=1 THEN      !mueve la pieza hasta el sitio de descarga
    Mover_pieza;
    Set DO_pieza_movida;
    !WaitTime 0.1*tiempo_espera; !espero a que se desactive la ventosa y se hagan las
comunicaciones con el PC

ELSEIF deja_pieza=1 THEN !deja la pieza en el sitio de descarga
    Dejar_pieza;
    Set DO_pieza_dejada;

ELSEIF finaliza=1 THEN      !acaba el pale
    Fin;
    Set DO_finalizado;

ELSEIF DI_sensor_pale=1 THEN      !pone el sensor sobre el centro del pale
    Posicionar_sensor;
    Set DO_sensor_pale;
ENDIF

ENDIF

ENDPROC

ENDMODULE

```



```

MODULE Points
    CONST robtarget home:=[[-457.71,129.67,909.64],[0.548925,-0.112085,-0.824308,-0.0814577],
[1,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]; !posicon de inicio
    CONST robtarget deja:=[[-343.71,-51.44,326.26],[0.00436709,-0.0100748,0.00438916,-0.99993],
[0,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]; !punto donde se deja la pieza
    VAR robtarget visualizacion_ini:=[351.91,474.44,-603.27],
[0.704796,-0.00470417,-0.0103333,0.709319],[1,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]; !
punto de visualizacion con respecto al pale
    VAR robtarget visualizacion:=[351.91,474.44,-603.27],[0.704795,-0.00470493,-0.0103332,0.70932],
[1,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]; !punto de visualizacion con respecto al pale
    VAR robtarget esquivar_iluminacion:=[49.12,379.20,-686.98],
[0.438332,-0.0124791,-0.0171508,0.898563],[0,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]; !
punto de visualizacion con respecto al pale

    PROC borrar()
        MoveL deja,v1000,z100,ventosa\WObj:=descarga;
        MoveL home,v1000,z100,ventosa\WObj:=wobj0;
        MoveL visualizacion,v1000,z100,ventosa\WObj:=pale;
    ENDPROC
!variable para modificar el wobj pale. Igualo el wobj a este punto
ENDMODULE

```

estacion_despaletizado / plc_vison_robotica [CPU 1513F-1 PN] / Bloques de programa

Actualizacion_entradas [FC1]

Actualizacion_entradas Propiedades					
General					
Nombre	Actualizacion_entradas	Número	1	Tipo	FC
Idioma	SCL	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizada	

Actualizacion_entradas			
Nombre	Tipo de datos	Valor predet.	Comentario
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
Actualizacion_entradas	Void		

```
0001 //TARJETAS
0002 //sensor de distancia
0003 "TARJETAS".ENTRADAS.ANALOGICAS.distancia := SCALE_X_REAL(MAX := 2000, MIN :=
100, VALUE := NORM_X_REAL(MAX := 27623, MIN := 0, VALUE := "AI_distancia"));
0004 "HMI".DE_HMI.inicializacion := "inicializacion";
0005 "HMI".DE_HMI.pale_colocado := "pale_colocado" XOR 1;
0006 //////////////////////////////////////
0007
0008
0009
0010 //ROBOT
0011 //BUS DE ROBOT --> DB Robot
0012 "ROBOT".DE.camara_pale := "I_Robot".DI_camara_pale;
0013 "ROBOT".DE.sensor_pale_ini := "I_Robot".DI_sensor_pale_ini;
0014 "ROBOT".DE.finalizado := "I_Robot".DI_finalizado;
0015 "ROBOT".DE.inicializado := "I_Robot".DI_inicializado;
0016 "ROBOT".DE.pieza_cogida := "I_Robot".DI_pieza_cogida;
0017 "ROBOT".DE.pieza_dejada := "I_Robot".DI_pieza_dejada;
0018 "ROBOT".DE.pieza_movida := "I_Robot".DI_pieza_movida;
0019 "ROBOT".DE.sensor_pale := "I_Robot".DI_sensor_pale;
0020 "ROBOT".DE.sensor_pieza := "I_Robot".DI_sensor_pieza;
0021 "ROBOT".DE.camara_pale_ini := "I_Robot".DI_camara_pale_ini;
0022
0023 //////////////////////////////////////
0024
0025
0026
0027 //VISION
0028 //DB OPC_VISION --> DB VISION
0029 "VISION".DE := "OPC_VISION".DE;
```

Símbolo	Dirección	Tipo	Comentario
"AI_distancia"	%IW2	Int	lectura del sensor de distancia

Totally Integrated Automation Portal			
Símbolo	Dirección	Tipo	Comentario
"HMI".DE_HMI.inicializacion		Bool	
"HMI".DE_HMI.pale_colocado		Bool	
"I_Robot".DI_camara_pale	%I100.3	Bool	
"I_Robot".DI_camara_pale_ini	%I100.2	Bool	
"I_Robot".DI_finalizado	%I101.1	Bool	
"I_Robot".DI_inicializado	%I100.0	Bool	
"I_Robot".DI_pieza_cogida	%I100.5	Bool	
"I_Robot".DI_pieza_dejada	%I100.7	Bool	
"I_Robot".DI_pieza_movida	%I100.6	Bool	
"I_Robot".DI_sensor_pale	%I100.4	Bool	
"I_Robot".DI_sensor_pale_ini	%I100.1	Bool	
"I_Robot".DI_sensor_pieza	%I101.0	Bool	
"inicializacion"	%I0.0	Bool	
"OPC_VISION".DE		Struct	
"pale_colocado"	%I0.1	Bool	
"ROBOT".DE.camara_pale		Bool	
"ROBOT".DE.camara_pale_ini		Bool	
"ROBOT".DE.finalizado		Bool	
"ROBOT".DE.inicializado		Bool	
"ROBOT".DE.pieza_cogida		Bool	
"ROBOT".DE.pieza_dejada		Bool	
"ROBOT".DE.pieza_movida		Bool	
"ROBOT".DE.sensor_pale		Bool	
"ROBOT".DE.sensor_pale_ini		Bool	
"ROBOT".DE.sensor_pieza		Bool	
"TARJETAS".ENTRADAS.ANALOGICAS.distancia	%DB1.DBDO	Real	sensor de distancia
"VISION".DE	P#DB3.DBX0.0	Struct	

estacion_despaletizado / plc_vison_robotica [CPU 1513F-1 PN] / Bloques de programa

Actualizacion_salidas [FC2]

Actualizacion_salidas Propiedades					
General					
Nombre	Actualizacion_salidas	Número	2	Tipo	FC
Idioma	SCL	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizada	

Actualizacion_salidas			
Nombre	Tipo de datos	Valor predet.	Comentario
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
Actualizacion_salidas	Void		

```
0001 //TARJETAS
0002 "ventosa":= "TARJETAS".SALIDAS.ventosa;
0003
0004 //DB ROBOT -->Bus Robot
0005 //Ordenes
0006 "O_Robot".DO_camara_pale := "ROBOT".A.ORDENES.camara_pale;
0007 "O_Robot".DO_sensor_pale_ini := "ROBOT".A.ORDENES.sensor_pale_ini;
0008 "O_Robot".DO_sensor_pieza := "ROBOT".A.ORDENES.sensor_pieza;
0009 "O_Robot".DO_mueve_pieza := "ROBOT".A.ORDENES.mueve_pieza;
0010 "O_Robot".DO_deja_pieza := "ROBOT".A.ORDENES.deja_pieza;
0011 "O_Robot".DO_inicializacion := "ROBOT".A.ORDENES.inicializacion;
0012 "O_Robot".DO_coge_pieza := "ROBOT".A.ORDENES.coge_pieza;
0013 "O_Robot".DO_camara_pale_ini := "ROBOT".A.ORDENES.camara_pale_ini;
0014 "O_Robot".DO_sensor_pale := "ROBOT".A.ORDENES.sensor_pale;
0015 "O_Robot".DO_finaliza := "ROBOT".A.ORDENES.finaliza;
0016
0017 //Datos pieza
0018 "O_Robot".GO_pieza_x := "ROBOT".A.PIEZA.X;
0019 "O_Robot".GO_pieza_y := "ROBOT".A.PIEZA.Y;
0020 "O_Robot".GO_pieza_z := "ROBOT".A.PIEZA.Z;
0021 "O_Robot".GO_pieza_a := "ROBOT".A.PIEZA.A;
0022 "O_Robot".GO_pieza_b := "ROBOT".A.PIEZA.B;
0023 "O_Robot".GO_pieza_c := "ROBOT".A.PIEZA.C;
0024
0025 //Datos pale
0026 "O_Robot".GO_pale_x := "ROBOT".A.PALE.X;
0027 "O_Robot".GO_pale_y := "ROBOT".A.PALE.Y;
0028 "O_Robot".GO_pale_z := "ROBOT".A.PALE.Z;
0029 "O_Robot".GO_pale_a := "ROBOT".A.PALE.A;
0030 "O_Robot".GO_pale_b := "ROBOT".A.PALE.B;
0031 "O_Robot".GO_pale_c := "ROBOT".A.PALE.C;
0032
0033 //////////////////////////////////////
```

Totally Integrated Automation Portal			
<div>0034</div> <div>0035 //VISION</div> <div>0036 "OPC_VISION".A := "VISION".A;</div> <div>0037</div> <div>0038 //////////////////////////////////////</div> <div>0039</div> <div>0040 //HMI</div> <div>0041 "HMI".A_HMI.modos := "Bloque de datos".modos;</div>			
Símbolo	Dirección	Tipo	Comentario
"Bloque de datos".modos		Int	
"HMI".A_HMI.modos		Int	
"O_Robot".DO_camarapale	%Q100.3	Bool	
"O_Robot".DO_camarapale_ini	%Q100.2	Bool	
"O_Robot".DO_cogepieza	%Q100.6	Bool	
"O_Robot".DO_dejapieza	%Q101.0	Bool	
"O_Robot".DO_finaliza	%Q101.1	Bool	
"O_Robot".DO_inicializacion	%Q100.0	Bool	
"O_Robot".DO_muevepieza	%Q100.7	Bool	
"O_Robot".DO_sensorpale	%Q100.4	Bool	
"O_Robot".DO_sensorpale_ini	%Q100.1	Bool	
"O_Robot".DO_sensorpieza	%Q100.5	Bool	
"O_Robot".GO_pale_a	%QD138	DInt	
"O_Robot".GO_pale_b	%QD142	DInt	
"O_Robot".GO_pale_c	%QD146	DInt	
"O_Robot".GO_pale_x	%QD126	DInt	
"O_Robot".GO_pale_y	%QD130	DInt	
"O_Robot".GO_pale_z	%QD134	DInt	
"O_Robot".GO_pieza_a	%QD114	DInt	
"O_Robot".GO_pieza_b	%QD118	DInt	
"O_Robot".GO_pieza_c	%QD122	DInt	
"O_Robot".GO_pieza_x	%QD102	DInt	
"O_Robot".GO_pieza_y	%QD106	DInt	
"O_Robot".GO_pieza_z	%QD110	DInt	
"OPC_VISION".A		Struct	
"ROBOT".A.ORDENES.camarapale		Bool	
"ROBOT".A.ORDENES.camarapale_ini		Bool	
"ROBOT".A.ORDENES.cogepieza		Bool	
"ROBOT".A.ORDENES.dejapieza		Bool	
"ROBOT".A.ORDENES.finaliza		Bool	
"ROBOT".A.ORDENES.inicializacion		Bool	
"ROBOT".A.ORDENES.muevepieza		Bool	
"ROBOT".A.ORDENES.sensorpale		Bool	

Totally Integrated Automation Portal			
Símbolo	Dirección	Tipo	Comentario
"ROBOT".A.OR-DENES.sensor_pale_ini		Bool	
"ROBOT".A.OR-DENES.sensor_pieza		Bool	
"ROBOT".A.PALE.A		DInt	
"ROBOT".A.PALE.B		DInt	
"ROBOT".A.PALE.C		DInt	
"ROBOT".A.PALE.X		DInt	
"ROBOT".A.PALE.Y		DInt	
"ROBOT".A.PALE.Z		DInt	
"ROBOT".A.PIEZA.A		DInt	
"ROBOT".A.PIEZA.B		DInt	
"ROBOT".A.PIEZA.C		DInt	
"ROBOT".A.PIEZA.X		DInt	
"ROBOT".A.PIEZA.Y		DInt	
"ROBOT".A.PIEZA.Z		DInt	
"TARJETAS".SALIDAS.ventosa	%DB1.DBX20.0	Bool	
"ventosa"	%Q0.0	Bool	alimentacion de la bobina de la electrovalvula de la ventosa
"VISION".A	P#DB3.DBX58.0	Struct	

Totally Integrated Automation Portal

estacion_despaletizado / plc_vision_robotica [CPU 1513F-1 PN] / Bloques de programa

F_Vision_artificial [FC4]

F_Vision_artificial Propiedades

General

Nombre	F_Vision_artificial	Número	4	Tipo	FC
Idioma	SCL	Numeración	Automático		

Información

Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizada	

F_Vision_artificial

Nombre	Tipo de datos	Valor predet.	Comentario
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
F_Vision_artificial	Void		

```
0001
0002
0003
0004 IF "Bloque de datos".modo = 2 THEN
0005     //Modo automatico
0006     //Borro todas las ordenes
0007     "VISION".A.inicializacion := 0;
0008     "VISION".A.reconoce_pale := 0;
0009     "VISION".A.reconoce_piezas := 0;
0010     "VISION".A.envia_datos := 0;
0011     "VISION".A.prepara_datos := 0;
0012     "VISION".A.finaliza := 0;
0013
0014 IF "HMI".DE_HMI.inicializacion THEN
0015     //iniciar
0016     "VISION".A.inicializacion := 1;
0017
0018 ELSEIF ("VISION".DE.ESTADOS.inicializado AND "ROBOT".DE.camara_pale_ini) THEN
0019     //vision iniciada y camara sobre pale 0 datos actualizados, no quedan pie-
0020     zas en este nivel y la camara esta sobre el pale --> reconoce pale
0021     "VISION".A.reconoce_pale := 1;
0022
0023 ELSEIF ("VISION".DE.ESTADOS.pale_reconocido AND "ROBOT".DE.camara_pale) OR
("VISION".DE.ESTADOS.datos_actualizados AND "VISION".DE.PIEZAS.numero = 0 AND
"ROBOT".DE.camara_pale) THEN
0024     //pale reconocido y la camara sobre el pale 0 datos actualizados, no que-
0025     dan piezas en este nivel y la camara esta sobre el pale --> reconoce las pie-
0026     zas
0027     "VISION".A.reconoce_piezas := 1;
0028
0029 ELSEIF ("VISION".DE.ESTADOS.piezas_reconocidas AND "VISION".DE.PIEZAS.no_pie-
zas=0) OR ("VISION".DE.ESTADOS.datos_preparados AND "TARJETAS".SALIDAS.vento-
sa) THEN
```


estacion_despaletizado / plc_vison_robotica [CPU 1513F-1 PN] / Bloques de programa

F_Robot [FC3]

F_Robot Propiedades					
General					
Nombre	F_Robot	Número	3	Tipo	FC
Idioma	SCL	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizada	

F_Robot			
Nombre	Tipo de datos	Valor predet.	Comentario
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
F_Robot	Void		

```
0001 IF "Bloque de datos".modo = 2 THEN
0002     //Modo automatico
0003     //Borro todas las ordenes
0004     //Los datos de posiciones estan multiplicados por 100 porque el robot no ad-
    mite decimales
0005
0006
0007
0008     "ROBOT".A.ORDENES.inicializacion := 0;
0009     "ROBOT".A.ORDENES.sensor_pale_ini := 0;
0010     "ROBOT".A.ORDENES.camara_pale_ini := 0;
0011     "ROBOT".A.ORDENES.sensor_pale := 0;
0012     "ROBOT".A.ORDENES.camara_pale := 0;
0013     "ROBOT".A.ORDENES.sensor_pieza := 0;
0014     "ROBOT".A.ORDENES.coge_pieza := 0;
0015     "ROBOT".A.ORDENES.mueve_pieza := 0;
0016     "ROBOT".A.ORDENES.deja_pieza := 0;
0017     "ROBOT".A.ORDENES.finaliza := 0;
0018
0019
0020 IF "HMI".DE_HMI.inicializacion THEN
0021     //PULSO INICIALIZACION
0022
0023
0024     //Le pongo todo 0
0025     "ROBOT".A.PALE.X := 0;
0026     "ROBOT".A.PALE.Y := 0;
0027     //"ROBOT".A.PALE.Z := "VISION".DE.PALE.Z;
0028     //"ROBOT".A.PALE.A := "VISION".DE.PALE.A;
0029     //"ROBOT".A.PALE.B := "VISION".DE.PALE.B;
0030     "ROBOT".A.PALE.C := 0;
0031
0032     "TARJETAS".SALIDAS.ventosa := 0;
```

```
0033     "ROBOT".A.ORDENES.inicializacion := 1;
0034
0035     ELSIF ("ROBOT".DE.inicializado AND "HMI".DE_HMI.pale_colocado) THEN
0036         //Estoy en inicializacion y pongo el pale --> coloca sensor sobre el pale
para saber a que distancia esta
0037         //No le paso coordenadas porque el robot va a un punto que tiene guardado
el
0038
0039         "TARJETAS".SALIDAS.ventosa := 0;
0040         "ROBOT".A.ORDENES.sensor_pale_ini := 1;
0041
0042         ELSIF ("ROBOT".DE.sensor_pale_ini) THEN
0043             //El sensor esta sobre el pale pero sin estar centrado --> paso el dato
de la distancia a la que esta el pale y posicoino la camara para buscar el
pale. No estará centrado sobre el pale porque aun no lo he reconocido
0044
0045             //Le pongo todo 0
0046             "ROBOT".A.PALE.X := 0;
0047             "ROBOT".A.PALE.Y := 0;
0048             //"ROBOT".A.PALE.Z := "VISION".DE.PALE.Z;
0049             //"ROBOT".A.PALE.A := "VISION".DE.PALE.A;
0050             //"ROBOT".A.PALE.B := "VISION".DE.PALE.B;
0051             "ROBOT".A.PALE.C := 0;
0052
0053             //Paso dato dedistancia que es el unico que conozco
0054             "ROBOT".A.PALE.Z := REAL_TO_DINT(IN := "TARJETAS".ENTRADAS.ANALOGICAS.dis-
tancia * 100) - "VISION".DE.distancia_vision;
0055
0056             "TARJETAS".SALIDAS.ventosa := 0;
0057             "ROBOT".A.ORDENES.camara_pale_ini:=1;
0058
0059             ELSIF ("ROBOT".DE.camara_pale_ini AND "VISION".DE.ESTADOS.pale_reconocido)
THEN
0060                 //La camara esta sobre el pale pero no centrado, solo con la altura cor-
recta, y reconozco el pale --> posicoino la camara centrada en el pale
0061
0062                 //Mando datos del pale con respecto a desde donde fue visualizado
0063                 "ROBOT".A.PALE.X := "VISION".DE.PALE.X;
0064                 "ROBOT".A.PALE.Y := "VISION".DE.PALE.Y;
0065                 //"ROBOT".A.PALE.Z := "VISION".DE.PALE.Z;
0066                 //"ROBOT".A.PALE.A := "VISION".DE.PALE.A;
0067                 //"ROBOT".A.PALE.B := "VISION".DE.PALE.B;
0068                 "ROBOT".A.PALE.C := "VISION".DE.PALE.C;
0069
0070                 "TARJETAS".SALIDAS.ventosa := 0;
0071                 "ROBOT".A.ORDENES.camara_pale:=1;
0072
0073                 ELSIF ("ROBOT".DE.sensor_pale) THEN
0074                     //el sensor esta centrado sobre el pale --> corrijo la altura y poisciono
la camara para ver las piezas
0075                     //los datos de x e y ya han sido pasado
0076                     "ROBOT".A.PALE.Z := REAL_TO_DINT(IN := "TARJETAS".ENTRADAS.ANALOGICAS.dis-
tancia * 100) - "VISION".DE.distancia_vision;
0077
0078                     "TARJETAS".SALIDAS.ventosa := 0;
0079                     "ROBOT".A.ORDENES.camara_pale := 1;
0080
```

```
0081     ELSIF ("ROBOT".DE.camara_pale AND "VISION".DE.ESTADOS.datos_actualizados AND
"VISION".DE.PIEZAS.numero > 0) OR ("ROBOT".DE.pieza_dejada AND "VI-
SION".DE.PIEZAS.numero > 0) THEN
0082     //la camara esta sobre las piezas y el sistema de vision manda datos 0 se
acaba de dejar una pieza y manda dato de la siguiente pieza --> sensor al cen-
tro de la pieza
0083
0084
0085     "ROBOT".A.PIEZA.X := "VISION".DE.PIEZAS.X;
0086     "ROBOT".A.PIEZA.Y := "VISION".DE.PIEZAS.Y;
0087     //"ROBOT".A.PIEZA.Z := "VISION".DE.PIEZAS.Z;
0088     //"ROBOT".A.PIEZA.A := "VISION".DE.PIEZAS.A;
0089     //"ROBOT".A.PIEZA.B := "VISION".DE.PIEZAS.B;
0090     "ROBOT".A.PIEZA.C := "VISION".DE.PIEZAS.C;
0091
0092     "TARJETAS".SALIDAS.ventosa := 0;
0093     "ROBOT".A.ORDENES.sensor_pieza := 1;
0094
0095     ELSIF "ROBOT".DE.sensor_pieza THEN
0096     //el sensor esta en el centro de la pieza --> coger pieza
0097
0098     //mando distancia a la pieza
0099     "ROBOT".A.PIEZA.Z := REAL_TO_DINT(IN := "TARJETAS".ENTRADAS.ANALOGI-
CAS.distancia*100);
0100
0101     "TARJETAS".SALIDAS.ventosa := 0;
0102     "ROBOT".A.ORDENES.coge_pieza := 1;
0103
0104     ELSIF "ROBOT".DE.pieza_cogida AND "VISION".DE.ESTADOS.datos_preparados THEN
0105     //la ventosa esta sobre la pieza --> activo la ventosa y muevo la pieza
0106
0107     "TARJETAS".SALIDAS.ventosa := 1;
0108     "ROBOT".A.ORDENES.mueve_pieza := 1;
0109
0110     ELSIF "ROBOT".DE.pieza_movida AND "VISION".DE.ESTADOS.datos_actualizados
THEN
0111     //el robot ya ha movido la pieza --> la dejo
0112
0113     "TARJETAS".SALIDAS.ventosa := 0;
0114     "ROBOT".A.ORDENES.deja_pieza := 1;
0115
0116     ELSIF "ROBOT".DE.camara_pale AND "VISION".DE.ESTADOS.piezas_reconocidas AND
"VISION".DE.PIEZAS.no_piezas=1 THEN
0117     //la camara esta sobre las piezas y no se detecta ninguna --> FIN porque
el pale esta vacio
0118
0119     "TARJETAS".SALIDAS.ventosa := 0;
0120     "ROBOT".A.ORDENES.finaliza := 1;
0121
0122     ELSIF ("ROBOT".DE.pieza_dejada AND "VISION".DE.ESTADOS.datos_actualizados
AND "VISION".DE.PIEZAS."numero"=0) THEN
0123     //dejo la pieza y no quedan mas --> llevo el sensor al pale al punto de
vision paracorreger la altura
0124
0125     "TARJETAS".SALIDAS.ventosa := 0;
0126     "ROBOT".A.ORDENES.sensor_pale:=1;
0127     END_IF;
0128
0129 END_IF;
```

0130

Símbolo	Dirección	Tipo	Comentario
"Bloque de datos".modo		Int	
"HMI".DE_HMI.inicializa- cion		Bool	
"HMI".DE_HMI.pale_colo- cado		Bool	
"ROBOT".A.ORDENES.ca- mara_pale		Bool	
"ROBOT".A.ORDENES.ca- mara_pale_ini		Bool	
"ROBOT".A.OR- DENES.coge_pieza		Bool	
"ROBOT".A.ORDENES.de- ja_pieza		Bool	
"ROBOT".A.ORDENES.fi- naliza		Bool	
"ROBOT".A.ORDENES.ini- cializacion		Bool	
"ROBOT".A.OR- DENES.mueve_pieza		Bool	
"ROBOT".A.OR- DENES.sensor_pale		Bool	
"ROBOT".A.OR- DENES.sensor_pale_ini		Bool	
"ROBOT".A.OR- DENES.sensor_pieza		Bool	
"ROBOT".A.PALE.C		DInt	
"ROBOT".A.PALE.X		DInt	
"ROBOT".A.PALE.Y		DInt	
"ROBOT".A.PALE.Z		DInt	
"ROBOT".A.PIEZA.C		DInt	
"ROBOT".A.PIEZA.X		DInt	
"ROBOT".A.PIEZA.Y		DInt	
"ROBOT".A.PIEZA.Z		DInt	
"ROBOT".DE.ca- mara_pale		Bool	
"ROBOT".DE.ca- mara_pale_ini		Bool	
"ROBOT".DE.inicializado		Bool	
"ROBOT".DE.pieza_cogi- da		Bool	
"ROBOT".DE.pieza_deja- da		Bool	
"ROBOT".DE.pieza_movi- da		Bool	
"ROBOT".DE.sensor_pale		Bool	
"ROBOT".DE.sen- sor_pale_ini		Bool	
"ROBOT".DE.sensor_pie- za		Bool	
"TARJETAS".ENTRA- DAS.ANALOGICAS.dis- tancia	%DB1.DBDO	Real	sensor de distancia
"TARJETAS".SALI- DAS.ventosa	%DB1.DBX20.0	Bool	
"VISION".DE.distancia_vi- sion	%DB3.DBD54	DInt	
"VISION".DE.ESTA- DOS.datos_actualizados	%DB3.DBX0.4	Bool	

Totally Integrated Automation Portal		

Totally Integrated Automation Portal

estacion_despaletizado / plc_vison_robotica [CPU 1513F-1 PN] / Bloques de programa

Main [OB1]

Main Propiedades

General

Nombre	Main	Número	1	Tipo	OB
Idioma	SCL	Numeración	Automático		

Información

Título	"Main Program Sweep (Cycle)"	Autor		Comentario	
Familia		Versión	0.1	ID personalizada	

Main

Nombre	Tipo de datos	Valor predet.	Comentario
▼ Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

0001

//actualizo el valor de las señales de entrada

0002

"Actualizacion_entradas"();

0003

0004

//escojo el modo de funcionamiento

0005

//"Modo_funcionamiento"();

0006

0007

//comunico ordenes a ls procesadoreso

0008

"F_Robot"();

0009

"F_Vision_artificial"();

0010

0011

0012

//actualizo el valor de las señales de salida

0013

"Actualizacion_salidas"();

Símbolo	Dirección	Tipo	Comentario
---------	-----------	------	------------

estacion_despaletizado / plc_vison_robotica [CPU 1513F-1 PN] / Bloques de programa

Modo_funcionamiento [FC5]

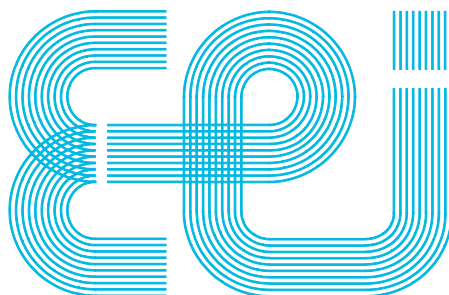
Modo_funcionamiento Propiedades					
General					
Nombre	Modo_funcionamiento	Número	5	Tipo	FC
Idioma	SCL	Numeración	Automático		
Información					
Título		Autor		Comentario	
Familia		Versión	0.1	ID personalizada	

Modo_funcionamiento			
Nombre	Tipo de datos	Valor predet.	Comentario
Input			
Output			
InOut			
Temp			
Constant			
▼ Return			
Modo_funcionamiento	Void		

```
0001 IF ("Bloque de datos".modo = 3 AND "HMI".DE_HMI.SELECCION_MODAL.mantenimiento =
      0) OR ("Bloque de datos".modo = 1 AND "HMI".DE_HMI.SELECCION_MODAL.parada = 1)
      OR ("Bloque de datos".modo = 2 AND "HMI".DE_HMI.SELECCION_MODAL.parada = 1)
      THEN
0002 //modo mantenimiento y desactivo mantenimiento 0 pulso parada desde automa-
      tico o manual --> paro
0003
0004 "Bloque de datos".modo := 0;
0005
0006 ELIF ("Bloque de datos".modo = 0 AND "HMI".DE_HMI.SELECCION_MODAL.parada = 0)
      OR ("Bloque de datos".modo = 2 AND "HMI".DE_HMI.SELECCION_MODAL.automatice = 0)
      THEN
0007 //Modo paro y desenchavo paro 0 modo automatico y desactivo automatico --
      >modo manual
0008
0009 "Bloque de datos".modo := 1;
0010
0011 ELIF "Bloque de datos".modo = 1 AND "HMI".DE_HMI.SELECCION_MODAL.automatice
      AND "HMI".DE_HMI.inicializacion THEN
0012 //modo manual y pulso modo automatico e inicializo --> modo automatico
0013
0014 "Bloque de datos".modo := 2;
0015
0016 ELIF "Bloque de datos".modo = 0 AND "HMI".DE_HMI.SELECCION_MODAL.mantenimiento
      = 1 THEN
0017 //modo paro y pulso mantenimiento
0018
0019 "Bloque de datos".modo := 3;
0020
0021 END_IF;
```

Símbolo	Dirección	Tipo	Comentario
"Bloque de datos".modo		Int	

Totally Integrated Automation Portal			
Símbolo	Dirección	Tipo	Comentario
"HMI".DE_HMI.inicializa- cion		Bool	
"HMI".DE_HMI.SELEC- CION_MODO.automati- co		Bool	
"HMI".DE_HMI.SELEC- CION_MODO.manten- imiento		Bool	
"HMI".DE_HMI.SELEC- CION_MODO.parada		Bool	



Escuela de Ingeniería Industrial

TRABAJO FIN DE MÁSTER

*Diseño de una estación de despaletizado mediante un sistema
de visión robótica*

Máster en ingeniería industrial

Documento

ANEXO II: PROGRAMAS DE DESARROLLO

UniversidadeVigo

ÍNDICE

1. Area_calibracion
2. Calibracion_camara_robot
3. Comunicaciones_opc_ua

area_calibracion.hdev

main (: : :)

```
* Image Acquisition 01: Code generated by Image Acquisition 01
* Image Acquisition 01: be set in a specific order (e.g., image resolution vs. offset).
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', 8, 'default', -1, 'false',
'default', '1', 0, -1, AcqHandle)
set_framegrabber_param (AcqHandle, 'auto_brightness_max_exp', 129.87)
set_framegrabber_param (AcqHandle, 'auto_brightness_max_gain', 95)
set_framegrabber_param (AcqHandle, 'exposure', 50.0602)
set_framegrabber_param (AcqHandle, 'image_height', 3006)
set_framegrabber_param (AcqHandle, 'level_controlled_trigger', 'disable')
* Image Acquisition 01: Attention: The initialization may fail in case parameters need
to
grab_image_start (AcqHandle, -1)

una_vez := 1
while (una_vez=1)
    tuple_gen_const (0, 0, Areas)
    tuple_gen_const (0, 0, pixel_mm)

    grab_image_async (Image, AcqHandle, -1)
    * Image Acquisition 01: Do something
    regiongrowing (Image, Regions, 3, 3, 6, 100000)
    select_shape (Regions, piezas, 'rectangularity', 'and', 0.8, 1)
    count_obj (piezas, Number)

    for Index := 1 to Number by 1

        * selecciono una pieza del vector de regiones
        select_obj (piezas, ObjectSelected, Index)
        * la rectangularizo. Esto lo hago porque se que son rectangulares. Así tengo menos
error
        shape_trans (ObjectSelected, pieza_rect, 'rectangle2')
        * Calculo el area
        area_center (pieza_rect, Area, Row, Column)

        tuple_insert (Areas, 0, Area, Areas)

        * Calculo la relacion entre pixeles y mm conociendo el area de la pieza
        relacion := sqrt(Area/(240*92))

        tuple_insert (pixel_mm, 0, relacion, pixel_mm)
    endfor

    tuple_sum (pixel_mm, Sum)
    pixel_mm_media := Sum/Number

    suma_desviacion := 0
    * calculo de desviacion tipica
    for Index := 0 to Number-1 by 1
        suma_desviacion := suma_desviacion + (pixel_mm[Index]-
pixel_mm_media)*(pixel_mm[Index]-pixel_mm_media)
    endfor

    desviacion_tipica := sqrt(suma_desviacion/Number)
    una_vez := 0
```

```
endwhile  
close_framegrabber (AcqHandle)
```

calibracion_camara_robot.hdev

main (: : :)

```
* CONEXION CON LA CAMARA
* Image Acquisition 01: Code generated by Image Acquisition 01
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', 8, 'default', -1, 'false',
'default', '1', 0, -1, AcqHandle)
set_framegrabber_param (AcqHandle, 'exposure', 85)
grab_image_start (AcqHandle, -1)

while (true)

    * Captura imagen de la camara. Lo hago aqui fuera porque si lo hago despues de
    recibir una orden, no coge la imagen actualizada
    grab_image_async (Image, AcqHandle, -1)
    * dev_display (Image)
    * read_image (Image, 'C:/Users/DLN/Documents/Vision artificial/Imagenes/Tablas')
    * centro de la imagen. Las coordenadas de los puntos se envian relativos a este
    centro
    area_center (Image, Area1, cam_row, cam_colum)
    gen_cross_contour_xld (Cross_center, cam_row, cam_colum, 64, 0.5)
    dev_display (Cross_center)

    gen_cross_contour_xld (Cross_row, cam_row+1000, cam_colum, 64, 0.5)
    dev_display (Cross_row)

    gen_cross_contour_xld (Cross_colum, cam_row, cam_colum+1000, 64, 0.5)
    dev_display (Cross_colum)

endwhile
```

opc_ua_browse_variables_s7_1500.hdev

main (: : :)

```
*
* This example shows how to determine all variable nodes available on an OPC UA server.
*
*
* Please adapt the following lines to specify your configuration:
MyOpcUaIP := '172.16.131.228'
MyOpcUaPortNumber := 4840
*
MyOpcUaServer := 'opc.tcp://' + MyOpcUaIP + ':' + MyOpcUaPortNumber
*
dev_update_off ()
dev_close_window ()
*
* Open the connection to the server.
open_io_device ('OPC_UA', MyOpcUaServer, [], [], IoDeviceHandle)
*
* Use the root node to start browsing.
control_io_device (IoDeviceHandle, 'translate', [], Nodes)
Variables := []
while (|Nodes|)
    Start := Nodes[|Nodes| - 1]
    tuple_remove (Nodes, |Nodes| - 1, Nodes)
    query_io_device (IoDeviceHandle, Start, 'browse', Info)
    for I := 0 to |Info| - 1 by 3
        * Get node information.
        Node := Info[I + 0]
        Type := Info[I + 1]
        Name := Info[I + 2]
        * Due to a limitation in the OPC UA SDK, node IDs are truncated to
        * 255 characters, so any node Id of 255 characters is potentially truncated
        * and thus invalid.
        tuple_strlen (Node, Length)
        if (Length >= 255)
            continue
        endif
        if (Type == 'Variable')
            Variables := [Variables, Node]
        elseif (Type == 'Object')
            Nodes := [Node, Nodes]
        else
            endif
    endfor
endwhile
Variables := sort(Variables)

*
* Open graphics window for variables output:
ShowValue := false
NumWindows := |Variables| / 65 + 1
WindowHandles := []
ImageHeight := 800
for Index := 0 to NumWindows - 1 by 1
    dev_open_window (0, Index * 400, 500, ImageHeight, 'black', CurrentWindowHandle)
```

```

dev_set_color ('white')
set_display_font (CurrentWindowHandle, 11, 'mono', 'true', 'false')
WindowHandles := [WindowHandles, CurrentWindowHandle]
endfor
*
* Display the variables
count := 0
for i := 0 to |Variables| - 1 by 1
  if (i % 65 == 0)
    CurrentWindowHandle := WindowHandles[count]
    dev_set_window (CurrentWindowHandle)
    count := count + 1
  endif
  set_tposition (CurrentWindowHandle, (12 * (i - 65 * (count - 1))) + 2, 10)
  write_string (CurrentWindowHandle, Variables[i])
endfor
*
* Get the display_name and data_type for each variable.
query_io_device (IODeviceHandle, Variables, 'display_name', Displayname)
query_io_device (IODeviceHandle, Variables, 'data_type', DataType)
*
* Close the connection to the device
close_io_device (IODeviceHandle)

```

Procedimientos utilizados

dev_update_off (: : :)

Breve Descripción: Switch dev_update_pc, dev_update_var and dev_update_window to 'off'.

Capítulos: Develop

Público Procedimiento externo: C:/Program

Files/MVTec/HALCON-13.0/procedures/general/dev_update_off.hdev

```

* This procedure sets different update settings to 'off'.
* This is useful to get the best performance and reduce overhead.
*

```

```

dev_update_pc ('off')
dev_update_var ('off')
dev_update_window ('off')
return ()

```

set_display_font (: : WindowHandle, Size, Font, Bold, Slant :)

Breve Descripción: Set font independent of OS

Capítulos: Graphics / Text

Público Procedimiento externo: C:/Program

Files/MVTec/HALCON-13.0/procedures/general/set_display_font.hdev

```

* This procedure sets the text font of the current window with

```

```

* the specified attributes.
*
* Input parameters:
* WindowHandle: The graphics window for which the font will be set
* Size: The font size. If Size=-1, the default of 16 is used.
* Bold: If set to 'true', a bold font is used
* Slant: If set to 'true', a slanted font is used
*
get_system ('operating_system', OS)
if (Size == [] or Size == -1)
    Size := 16
endif
if (OS{0:2} == 'Win')
    * Restore previous behaviour
    Size := int(1.13677 * Size)
else
    Size := int(Size)
endif
if (Font == 'Courier')
    Fonts := ['Courier','Courier 10 Pitch','Courier New','CourierNew', 'Liberation Mono']
elseif (Font == 'mono')
    Fonts := ['Consolas','Menlo','Courier','Courier 10 Pitch','FreeMono', 'Liberation
Mono']
elseif (Font == 'sans')
    Fonts := ['Luxi Sans','DejaVu Sans','FreeSans','Arial', 'Liberation Sans']
elseif (Font == 'serif')
    Fonts := ['Times New Roman','Luxi Serif','DejaVu Serif','FreeSerif',
'Utopia','Liberation Serif']
else
    Fonts := Font
endif
Style := ''
if (Bold == 'true')
    Style := Style + 'Bold'
elseif (Bold != 'false')
    Exception := 'Wrong value of control parameter Bold'
    throw (Exception)
endif
if (Slant == 'true')
    Style := Style + 'Italic'
elseif (Slant != 'false')
    Exception := 'Wrong value of control parameter Slant'
    throw (Exception)
endif
if (Style == '')
    Style := 'Normal'
endif
query_font (WindowHandle, AvailableFonts)
Font := ''
for Fdx := 0 to |Fonts| - 1 by 1
    Indices := find(AvailableFonts,Fonts[Fdx])
    if (|Indices| > 0)
        if (Indices[0] >= 0)
            Font := Fonts[Fdx]
            break
        endif
    endif
endfor
if (Font == '')
    throw ('Wrong value of control parameter Font')
endif
Font := Font + '-' + Style + '-' + Size

```



```
set_font (WindowHandle, Font)
return ()
```