industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Sistema integrado de pick & place

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Riera Abellán, Luis Alfonso**
Director: Feliu Batlle, Jorge Juan
Codirector: Martínez Ruiz, Pablo Alejandro

Cartagena, 08 de Junio de 2021

Universidad Politécnica de Cartagena

Agradecimientos:

Quisiera agradecer este proyecto a mis directores *Jorge Juan Feliu Batlle* y *Pablo Alejandro Martínez Ruíz* por brindarme la oportunidad de realizar un proyecto tan apasionante. Mi más sincero agradecimiento por todas las facilidades que me han proporcionado, por la atención en todas las consultas que les he realizado, y por la ayuda y colaboración recibida para solucionar todos los problemas que han ido surgiendo durante el transcurso del proyecto.

También quisiera agradecer todo el apoyo y ánimo que me han otorgado mi familia y amigos, sin ellos no hubiera sido posible superar todas las dificultades que han aparecido en este proyecto y en toda esta etapa universitaria.

A nivel personal, este proyecto me ha supuesto un reto interesante y a pesar de las dificultades que han ido surgiendo, pienso que ha sido un acierto la elección del mismo. Me siento afortunado de que mis profesores y la universidad me hayan dado la posibilidad de realizar un trabajo de estas características y creo que el hecho de haber afrontado un proyecto práctico como este, me ha proporcionado una formación diferente a la que había recibido a lo largo del grado y con ello he conseguido reforzar mis conocimientos en robótica, automatización, visión artificial y programación.

Índice:

Tabla de ilustraciones:

1. Introducción

1.1. Resumen

En el presente trabajo fin de grado se pretende implementar un proceso formado por un brazo robótico y un sistema de visión artificial con la finalidad de conformar un sistema integrado de pick & place.

El proceso comienza con la captación de imágenes mediante una cámara, dicha imagen se enviará a la computadora donde esté implementado el sistema de visión artificial, el cual se encargará del procesamiento de la imagen.

El sistema de visión artificial, enviará la posición de los objetos detectados y mandará esta información al robot por medio de un socket, en el que el robot actuará como servidor y el sistema de visión será el cliente.

La parte final del proceso será la ejecución del movimiento pertinente por parte de nuestro robot IRB 120.

1.2. Justificación

Este proyecto ha sido planteado con el principal fin de introducirnos en los sistemas robotizados que incorporan visión artificial. Gracias a las diferentes ventajas y posibilidades que estos sistemas integrados nos ofrecen, se busca obtener una base de conocimiento que nos permita realizar futuros proyectos en la industria de sistemas, consiguiendo realizar procesos automáticos sin necesidad de intervención humana.

La ejecución de este tipo de proyectos en la práctica, nos permite enfrentarnos directamente a un caso real y afrontar las dificultades e imprevistos que puedan surgir y que no suelen ser tratados en el estudio teórico de la materia.

1.3. Objetivos

El objetivo principal será coordinar e integrar el sistema de visión artificial y la programación del robot para conseguir la operación de pick & place. Para que el proceso sea automático se deberá desarrollar un sistema de comunicación y coordinación entre los sistemas utilizados, de forma que sean capaces de interpretar los datos obtenidos.

Para llevar a cabo el objetivo propuesto, se han realizado las siguientes tareas a modo de subobjetivos:

- Estudio de las condiciones del entorno y objetos.
- Elaboración de un sistema de captación adecuado para las condiciones dadas.

- Elaboración del sistema de procesamiento de la imagen.
- Elaboración del sistema de reconocimiento de objetos.
- Extracción de datos relevantes de los objetos.
- Análisis de las características y limitaciones del robot.
- Estudio teórico del robot.
- Configuración del robot.
- Programación del robot para la resolución del problema.
- Estudio y comparativa de buses de campo.
- Selección del protocolo de comunicación.
- Transformación de datos.

1.4. Contextualización

Para la realización de este proyecto se disponía de una celda de trabajo ubicada en el área de Ingeniería de Sistemas y Automática de la Universidad Politécnica de Cartagena, concretamente en el laboratorio de automatización. Ahí es donde se encontraba instalado nuestro robot IRB 120 de la marca ABB.

En cuanto a la experiencia en la materia y los conocimientos previos con los que afrontaba el proyecto podríamos describir:

- Buena base teórica en sistemas robotizados, aunque muy poca experiencia en la programación de los mismos.
- Poco conocimiento en los sistemas de comunicación y sockets.
- Gran experiencia en la resolución de problemas de visión artificial.

1.5. Aplicaciones

Los sistemas integrados de pick & place que conforman la robótica y la visión artificial son cada vez más utilizados sobre todo en la industria para todo tipo de procesos.

Estos sistemas ofrecen, en la mayoría de ocasiones, una mayor velocidad y precisión en ciertos procesos. Además, también son muy interesantes para aplicaciones en las que es necesario trabajar en entornos con materiales peligrosos o con sustancias perjudiciales para el ser humano.

Una de las principales aplicaciones de estos sistemas es en la industria alimentaria para aplicaciones de control de calidad, por ello, en este proyecto, nos hemos centrado en simular y representar parte de un proceso de una empresa de almacenamiento de frutas y verduras.

El proceso consistirá en la detección de frutas de un determinado color, pues suele ser un distintivo de que la maduración es correcta, y tras ello se colocarán ordenadamente en lotes para su posterior comercialización. De esta manera, solo las frutas que

interesan para el consumo serán distribuidas y el resto serán desechadas o aprovechadas para otros fines.

2. Visión Artificial

2.1. Conceptos generales

2.1.1. Historia de la Visión Artificial

La vista es uno de los sentidos más importantes en nuestra vida, a través de ella somos capaces de extraer mucha información de nuestro entorno como la luz, el color, la forma, la distancia, la posición o el movimiento de aquello que nos rodea. Por ello, antes de adentrarnos en la historia de la visión artificial debemos estudiar la evolución que ha realizado el ser humano en cuanto al conocimiento del sentido de la vista.

Los primeros estudios sobre la visión surgen en la Grecia clásica de mano de Thales de Mileto, Pitágoras, Aristóteles y Euclides, quienes empezaron a comprender el sentido de la vista como una proyección geométrica de rayos de luz.

Con el paso de los años avanzábamos más en esta materia y si nos fijamos, ya en el renacimiento, se observa una evolución en la pintura relacionada con la perspectiva y la proyección de las imágenes.

La primera cámara aparece en 1545, cuando se presenta la cámara oscura, creada por el astrónomo Regnier Gemma Frisius (1508-1555). Como se puede apreciar en la ilustración 1, en esta cámara la luz entraba por un orificio muy pequeño y proyectaba la imagen invertida del mundo exterior. La historia de la fotografía da comienzo en 1826 cuando el químico francés Niepce (1765-1833) toma la primera foto, colocando una superficie fotosensible en el interior de la cámara oscura creada por Frisius. Unos años después, en el año 1838 otro químico francés llamado Daguerre (1787-1851) usó una placa fotográfica que era revelada con vapor de mercurio fijada con trisulfato de sodio sobre esta cámara.

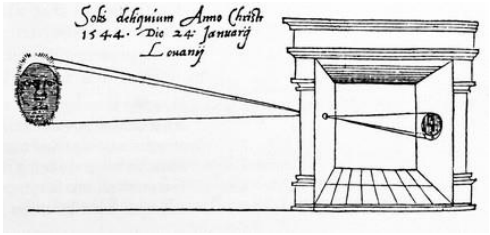


Ilustración 1. Cámara oscura

En la década de los años 20, aparecen diversos avances en relación con la visión digital, por ejemplo, con la transmisión de fotografías marítimas mediante el cable interoceánico. Gracias al empleo de las primeras técnicas en el procesado de imágenes que las codificaban a cinco niveles de grises, se consiguió que estas pudieran ser transmitidas por teléfono.

Avanzando temporalmente en la historia de la visión, llegamos hasta 1961, año en el cual Larry Roberts crea el primer programa de visión artificial denominado "el mundo de microbloques"

Posteriormente en 1964, en el programa espacial de la NASA, tenemos un ejemplo claro de lo que hoy conocemos como visión artificial. Un satélite dirigido a Marte, utilizaba distintas cámaras digitales que enviaban la información binarizada.

El procesamiento digital de las imágenes enviadas es casi idéntico al aplicado hoy en día en el campo de la visión artificial.

Todo lo que hasta ahora se conocía como visión artificial, en la década de los 70, recibe un cambio en su denominación y pasa a llamarse "Visión por Computador". Con ello, se formulan objetivos cada vez más ambiciosos, teniendo como prioridad la representación del espacio tridimensional a través de las imágenes, lo que nos permitirá la detección de objetos de un determinado entorno real mediante imágenes en estéreo, análisis de sombras, extracción de bordes…

Es entonces, en la década de los 80, cuando se inicia la visión artificial gracias a la revolución electrónica con las cámaras CCD y microprocesadores.

Seguidamente, en el año 1990, comienza a manifestarse la visión por computador aplicada a la industria. Con ello se inaugura la llegada de empresas dedicadas a la comercialización tanto de hardware como de software de visión artificial.

Ya en el siglo XXI, la investigación se ha centrado sobre todo en las técnicas basadas en la extracción de características para el reconocimiento de objetos, características basadas en puntos de interés y regiones o contornos. En la última década, la visión por computador está evolucionando incluyendo técnicas propias de la inteligencia artificial.

2.1.2. Elementos de un sistema de Visión Artificial

a) Iluminación

En muchas ocasiones el entorno en el que se encuentran las fábricas industriales no posee una iluminación adecuada y a la hora de incorporar sistemas de visión artificial se hace necesario el estudio y el control de la iluminación para evitar imágenes de bajo contraste, reflexiones especulares, destellos y sombras.

El diseño de un sistema de iluminación apropiado según el entorno, es una parte fundamental de la visión artificial, ya que nos permite una mayor calidad de imagen, que se traducirá en facilidad a la hora de procesar la información necesaria del entorno como la detección de objetos.

- Tipos de iluminación artificial

En cuanto a los tipos de iluminación que se aplican en la visión artificial, existen diversos tipos en función de características como: la temperatura máxima del foco, su vida útil o el rango de longitudes de onda en la que emite la luz.

En función del fenómeno físico que produce la luz podemos distinguir los siguientes tipos de iluminación:

-Luz incandescente:

Posiblemente no sea la iluminación más adecuada para aplicaciones de visión artificial pues no ofrece un espectro de colores demasiado amplio, se centra en el rojo siendo deficiente para azules, verdes o amarillos. Otro inconveniente a tener en cuenta es que desprenden una gran cantidad de calor.

La luz halógena estaría incluida en la luz incandescente, pues sigue el mismo principio, sin embargo, esta ofrece un mayor rendimiento luminoso y una mejor reproducción de colores.

-Luz fluorescente:

La iluminación mediante fluorescentes proporciona una luz brillante y sin sombras. Se utiliza en aquellos entornos en los que se necesita mucha iluminación y sin sombras, por esto es una de las más utilizadas en la visón por computador aplicada a la industria.

-Led:

La iluminación mediante diodos led tiene la ventaja de que se pueden construir fuentes de luz de diversas formas, empleando iluminación difusa, directa o en anillo.

-Láser:

Este tipo de iluminación se utiliza en un entorno específico de aplicación en el cual se necesita conocer mediante mediciones la profundidad del medio. Resulta especialmente interesante para aplicaciones en superficies irregulares. Gracias al acoplamiento de ópticas concretas para esta iluminación, se pueden obtener diversos tamaños y formas.

-Fibra óptica:

La iluminación mediante fibra óptica se caracteriza por no presentar sombras y por generar un haz de luz uniforme. Se utiliza sobre todo en aplicaciones en los que se trabaja con objetos pequeños.

Ilustración 2. Tipos de iluminación

- Técnicas de iluminación

Para conseguir una imagen apropiada para ser procesada, además del tipo de iluminación, es imprescindible utilizar una técnica de iluminación adecuada. Una imagen preparada para procesarla debe presentar una diferencia notable entre los pixeles que componen los objetos a detectar y el resto de pixeles, esto se consigue mediante la luminosidad.

Utilizando una iluminación correcta se evita que la imagen presente zonas oscuras o sombras que nos impidan diferencial los objetos y la información que queremos sacar de cada imagen. Otra razón por la que es imprescindible cuidar la iluminación del entorno, es por la precisión a la hora de realizar medidas, ya que las sombras causan falsas detecciones de contornos. También se debe considerar que un alumbrado escaso, aumenta el ruido de la imagen y por tanto disminuye la calidad de la misma. Asimismo, una iluminación no uniforme puede además complicar considerablemente las operaciones de segmentación.

Para seleccionar una iluminación acertada es indispensable conocer el papel que desempeña cada elemento del sistema de visión en la imagen a realizar. Cada componente influye en la cantidad de luz que llega al sensor y en consecuencia en la calidad de la imagen capturada.

Por ejemplo, la apertura del diafragma de la óptica influye directamente a la cantidad de luz que llega al sensor de la cámara, de manera que la intensidad de la luz tiene que ser incrementada cuanto menor sea la apertura del diafragma.

Por otro lado, es necesario considerar el área con de los objetos con los que se está trabajando, así como su color y brillo, pues son factores que influyen en la reflexión de la luz. De esta forma, una superficie grande refleja más luz que una pequeña y esto si debe estudiar al diseñar la iluminación del entorno.

Por último, el tipo de cámara utilizada también se debe tener en cuanta, a la hora de determinar nuestra iluminación, pues la sensibilidad de la cámara y la ganancia de la misma son factores importantes a tener en cuenta en la iluminación.

- Filtros y difusores

Un filtro es un instrumento hecho de un material determinado que interactúa con las ondas del espectro visible, atenuando los rayos de luz que no cumplan con ciertas propiedades de longitud de onda.

Ilustración 3. Filtros de iluminación

Las propiedades por las cuales se distinguen los filtros entre sí son:

1. El color que trasmiten
2. El componente del que están formados
3. La montura que integran
4. El factor de opacidad del filtro

Incluso hay otras clases de filtros no coloreados, como los filtros infrarrojos.

Un difusor es un instrumento que refleja la luz procedente de una determinada dirección y la divide en muchas direcciones consiguiendo que la luz llegue muy atenuada al objeto con el que se trabaja y disminuyendo así la reflexión de luz que se pueda generar. Además, reducen la densidad de las sombras.

b) El sistema óptico

- Lentes

Una lente corresponde a un instrumento óptico que enfoca o divide la luz que lo atraviesa mediante refracción. Aunque alejándose de esta definición, se encuentran las lentes de Fresnel que dispersan la luz por medio de difracción. Las lentes están compuestas de un medio transparente definido por dos superficies siendo curva al menos una de ellas y suelen estar constituidas de materiales como el vidrio o el plástico.

Hablamos de lentes simples cuando corresponden a una sola superficie de material transparente mientras que las lentes compuestas están formadas por dos o más lentes simples.

Las lentes de una cámara de visión capturan la foto y la mandan al sensor de imagen de la cámara. Una lente influye directamente en la calidad y la resolución de la imagen que se realiza.

Una lente está determinada por un índice de refracción, un índice de reflexión y un índice de dispersión. El índice de refracción n, nos indica que capacidad tiene la lente de disminuir la velocidad de los rayos de luz al atravesarla. La fórmula matemática define el índice de refracción como el cociente entre la velocidad de la luz en el vacío y la velocidad de la luz del medio en nuestro caso la lente.

índice de refracción

$$n = \frac{c}{v}$$

velocidad de la luz en el vacio

velocidad de la luz en el medio

Ilustración 4. Índice de refracción

Las lentes que presentan un índice de refracción pequeño tienen implícito tener un índice de reflexión menor pues estos dos coeficientes están directamente relacionados. Pero con el fin de obtener una lente con un índice de reflexión bajo sin que disminuya el índice de refracción se aplican unos recubrimientos en la lente. La reflexión es el suceso que se ocasiona en el momento en el que la luz cambia de medio y una parte de esta es retornado al medio de origen. Cuando un haz de luz procedente del aire atraviesa el vidrio de la lente se suelen ocasionar unas pérdidas del 10% por reflexión, la cantidad de pérdidas que se producen dependen de la longitud de onda.

Como se ha mencionado, es habitual el uso de recubrimientos en la lente y con esto se consiguen disminuir las pérdidas hasta un 0.2%. Cabe mencionar que el recubrimiento anti reflexión es diseñado para un determinado rango de longitudes de onda y que fuera de este no se puede asegurar que el efecto sea el deseado.

Por último, el índice de dispersión o índice de Abbe nos indica la diferencia entre el coeficiente de refracción con respecto a la longitud de la onda luminosa que traspasa la lente. Este valor nos concede la posibilidad de averiguar la separación de colores que se ocasiona en el instante en el que un haz de luz atraviesa la lente. Cuando este valor es reducido la dispersión será elevada y por ello habrá una divergencia de colores considerable, mientras que conforme la separación de colores y el índice de dispersión sean más bajos el índice de Abbe será mayor.

Línea de Fraunhofer	Color	Longitud de onda (nm)	Vidrio Crown	Sílex Extra denso
			Índice de refracción	
F	Azul (hidrógeno)	486,1	1,5293	1,7378
D	Amarillo (sodio)	589,3	1,5230	1,7200
C	Rojo (hidrógeno)	656,3	1,5204	1,7130
$v = \frac{(n_D - 1)}{(n_F - n_C)}$ = Número de Abbe			valor v	
			59	29

Ilustración 5. Cálculo del índice de Abbe

- Óptica

La óptica de una cámara es un conjunto de lentes dispuestas en un cilindro metálico que se integra a la cámara por medio de una rosca que usamos para ajustar la distancia focal. Se usan para mandar la luz al sensor de la cámara consiguiendo enfocar así los objetos de la escena deseados.

Ilustración 6. Ópticas de visión artificial

Para seleccionar una óptica se deben tener en cuenta los siguientes aspectos:

i) Distancia focal: es la longitud que hay entre el centro óptico de la lente y el punto focal donde se proyecta la imagen. Está directamente relacionada con el tamaño del sensor, con la distancia al objeto e y con el tamaño del objeto enfocado.

$$f = \frac{b * D * c}{B}$$

f= Distancia Focal, b= Tamaño del sensor, D= Distancia de trabajo

B= Anchura del objeto, c= Factor de conversión del tamaño del sensor

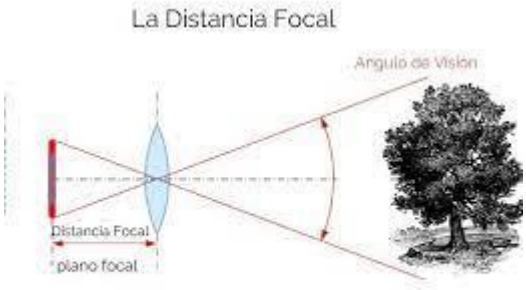


Ilustración 7. Distancia focal

ii) Montura: es el lugar en el cual se comunican la lente de la óptica y el cuerpo de una cámara. Podemos encontrar lentes con montura C, lentes con montura CS y lentes con montura F.

iii)Sensor: es un dispositivo que nos permite captar la presencia de cualquier cuerpo a una distancia determinada. Como se ha mencionado el tamaño del sensor es una magnitud directamente proporcional a la distancia focal.

iv) Apertura del diafragma: el diafragma modera la proporción de luz que llega al sensor y así como según la apertura de esta se vaya modificando, también variará la profundidad de campo. Se representa con F y sus valores denominados puntos del diafragma son 1, 1.4, 2, 2.8, 4, 5.6, 8, 11, 16, 22, 32, siendo F1 la apertura más abierta y F32 la más cerrada y por ello la que menos cantidad de luz dejará pasar al objetivo. F es el coeficiente entre la distancia focal f y el diámetro efectivo de la óptica.

Ilustración 8. Apertura del diafragma

Cuando solo están formadas por una única lente la distancia focal del objetivo es la misma que la de la lente. Sin embargo, al usar varias lentes esto no ocurre. En las ópticas conformadas por dos o más lentes, la distancia focal que depende de la distancia focal de cada lente que posee. Variando la distancia focal se varía el tamaño en la imagen del objeto al que se enfoca, es lo que comúnmente conocemos por zoom. Una forma de clasificar es según la distancia focal, y en este apartado encontramos ópticas de focal corta, ópticas de focal normal y ópticas de focal larga. Lo que llamamos ópticas de focal normal son aquellas que toman las imágenes a la misma o muy parecida distancia focal a la de la vista humana, es decir utilizando un zoom x1.

También encontramos las denominadas ópticas telecéntricas que son aquellas que solo dejan pasar a los rayos paralelos correspondientes a la escena. Esto presenta como ventaja eliminar errores de perspectiva al minimizar la distorsión, lo que les permite ofrecer una excelente calidad de imagen y por ello son muy útiles para aplicaciones de medición. Sin embargo, presenta el inconveniente de que la lente tendrá que ser al menos igual en tamaño que el objeto a capturar esto multiplica bastante el tamaño de las mismas y por tanto su precio final.

Las ópticas se pueden clasificar según el ángulo de visión que cubren. Los teleobjetivos corresponden a las ópticas que abarcan un campo de visión más pequeño. Por el contrario, las ópticas que abarcan más campo se les denomina como gran angular.

c) Cámaras

Una cámara es un aparato que, a través de un objetivo compuesto de lentes y diafragma, proyecta una imagen sobre el plano del sensor compuesto de elementos fotosensibles, se digitaliza dicha información en el caso de las cámaras digitales y se le manda a la tarjeta de adquisición.

Las cámaras de visión artificial son más sofisticadas que las convencionales y por ello presentan una serie de características específicas como puede ser el

control del disparo de la imagen el cual está preparado para captar imágenes en movimiento y en una posición concreta determinada con elevada precisión.

Las cámaras se dividen en dos grandes grupos dependiendo del tipo del sensor que utilicen para transformar la luz en una señal eléctrica, encontramos las CCD y las CMOS. Un sensor del tipo CCD (Charge Coupled Device) es recomendado principalmente para aplicaciones donde se requiere una mayor calidad de imagen. En cambio, CMOS (Complementary Metal Oxide) es utilizado en aplicaciones de espacio reducido y donde se busca el mínimo consumo de energía.

Comparando las características de estos sensores podríamos concluir en que:

-El nivel de ruido es mayor en una cámara CMOS.

-El rango dinámico es mayor en el caso de una cámara del tipo CCD.

-El tiempo de respuesta de los sensores CMOS es menor que CCD lo que los hace más rápidos.

Ilustración 9. Diferencia entre CCD y CMOS

Hasta hace solo unos años la tecnología de los sensores CCD se imponía sobre CMOS debido a que los sensores CCD presentaban diversas ventajas propias de su diseño. En primer lugar, no tenían que implementar la electrónica dentro del propio pixel y esto permitía que cada pixel pudiera recibir más luz, mejor señal, menos ruido y por ende mayor calidad.

Sin embargo, los sensores CCD tienen una gran desventaja y es que el proceso de fabricación de estos dispositivos electrónicos, es totalmente diferente al que poseen los microchips. Esto es un gran inconveniente, ya que esta tecnología no se aprovecha directamente de los continuos avances en el desarrollo y en la fabricación de microchips. A la vez, necesitan de un circuito integrado aparte en donde poder implementar la electrónica encargada de transformar la señal a digital, lo que supone un aumento en el volumen del sensor. Asimismo, los sensores CCD consumen más energía que los CMOS. Estos tres últimos puntos son los que han producido que CMOS en la actualidad se imponga por delante de CCD, además de que, al mismo tiempo, algunas de las desventajas de CMOS se han ido difuminando poco a poco.

Las diferencias de calidad que aventajaban claramente a los sensores CCD sobre CMOS a día de hoy son mínimas, ya que se ha conseguido mejorar la relación señal-ruido de CMOS, con lo cual se pueden alcanzar ISOs más altos sin pérdida de calidad.

Actualmente la tecnología CMOS nos ofrece una velocidad de ráfaga más alta, precios más bajos y una mayor autonomía en las baterías, todo esto sin una diferencia notable en la calidad con respecto a su competidor CCD.

También se clasifican las cámaras según la disposición física del formato del sensor, donde encontramos dos tipos: sensor de área o lineal. Los sensores de área son de mayor tamaño y tienen forma rectangular, además son económicamente más asequibles que los lineales y su configuración es más sencilla. Los sensores lineales capturan una línea de pixeles dependiendo la resolución de la cámara del tamaño de esta línea, es decir, realizan un barrido lineal del objeto con un desplazamiento longitudinal por el mismo. Una de sus mayores aplicaciones es para capturar fotos de objetos de gran tamaño, porque la preparación de la iluminación con estos sensores resulta más sencilla ya que solo se necesita iluminar una línea de la escena.

Los parámetros principales a valorar para realizar una elección adecuada de una cámara son:

-Distancia de trabajo

-Campo de visión

-Resolución

-Profundidad de campo

Las cámaras digitales presentan diversas opciones de transmisión, por ejemplo, el puerto Firewire con un solo cable conector se consigue obtener una salida de video y alimentación lo que nos permite disminuir los componentes de visión artificial. La velocidad de transferencia es un parámetro que nos indica el número de imágenes (frames) por segundo que se pueden realizar y también influye en la resolución y la profundidad de campo, es decir, en la cantidad de bits que utiliza cada pixel.

- Tarjetas de adquisición

Las tarjetas de adquisición son las responsables de percibir la señal digital de video o imagen, así como de almacenarla en memoria. Comúnmente también se le conoce como Frame Grabber

Las funciones principales que realizan las tarjetas de adquisición son:

- Adquisición de secuencias de imágenes.

- Aceleración del proceso de visualización en pantalla.
- Procesado de imágenes.

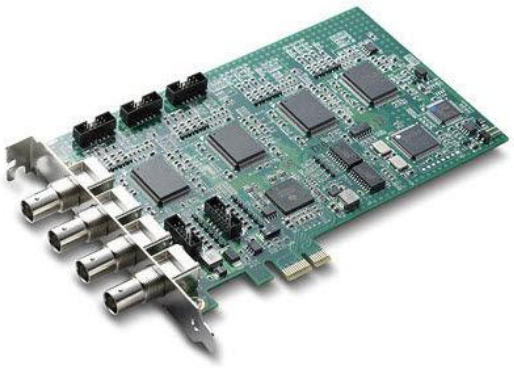


Ilustración 10. Tarjeta de adquisición

2.2. Implementación del sistema de Visión Artificial

2.2.1. Entorno y Objetos

Para implementar el sistema de visión artificial lo primero será estudiar y definir el entorno con el que se va a trabajar. En nuestro caso es sencillo definirlo pues consistirá solamente en adaptar nuestro sistema de visión artificial al entorno donde esté instalado nuestro brazo robótico.

Aspectos como la iluminación y la cámara no han sido optimizados al máximo en nuestro proyecto, pues el objetivo de este, tan solo era el de simular y estudiar un caso aplicable a un entorno real. Aun así, como hemos visto en el apartado anterior, se ha realizado un estudio teórico de estos elementos para poder afrontar cualquier proyecto futuro.

Los elementos que integran nuestro entorno de visión artificial serán:

- Iluminación:

La iluminación del laboratorio conformada por tubos fluorescentes ha sido suficiente para realizar las pruebas que se requerían. Como queda reflejado en el apartado anterior la luz fluorescente nos proporciona una imagen sin sombras, esto es imprescindible para nuestra solución ya que, de esta forma, los pixeles que conforman los objetos a manipular presentarán la misma tonalidad y esto nos permitirá detectar adecuadamente la posición de los objetos.

- Cámara:

Una cámara web estándar ha bastado para detectar los objetos sin demasiados problemas. Concretamente se ha usado la “Logitech HD Webcam C270”. A la hora de definir la posición de la cámara se ha tenido en cuenta que esté en una posición lo más elevada posible con el objetivo de minimizar los errores de perspectiva que se puedan presentar, puesto que lo ideal es que la imagen sea vista desde planta. Se ha usado un soporte de cámara con el fin de poder modificar el ángulo sin necesidad de tener que recolocar continuamente la posición de la cámara, el cual se ha adherido provisionalmente a la pared de la forma más sencilla posible.

Ilustración 11. Cámara

- Área de detección:

Se ha definido un rectángulo donde se colocarán nuestros objetos y se ha buscado que la imagen de área de detección encaje perfectamente con la perspectiva de la cámara, de forma que la imagen esté totalmente horizontal y recta en el plano de trabajo.

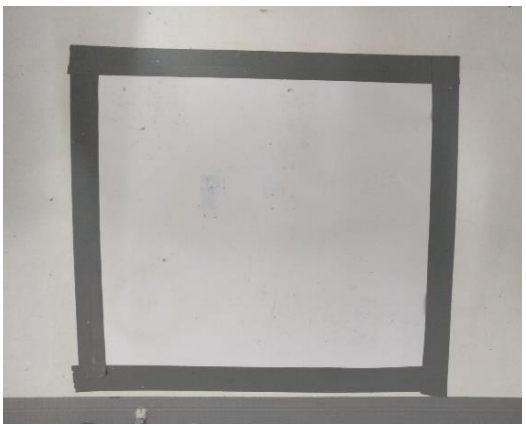


Ilustración 12. Área de detección

- Objetos:

Como se ha mencionado en otros apartados, este proyecto trata de representar un proceso de Pick & Place de una empresa de almacenamiento de frutas, en la que la fruta, será el objeto a detectar por nuestro sistema de visión artificial y posteriormente el objeto a manipular por nuestro brazo robótico.

Para ello, hemos simulado este proceso por medio de unas velas de colores que representan la fruta y que encajan perfectamente con el tamaño y peso de los objetos con los que puede trabajar la pinza de nuestro robot.

Ilustración 13. Objetos

2.2.2. Filtrado y máscara

Las técnicas de procesado tienen como misión mejorar las propiedades de la imagen con el fin de poder realizar operaciones de visión artificial más cómodamente.

De la misma manera que ocurre con las señales unidimensionales, las imágenes pueden ser filtradas con diferentes tipos de filtros, filtros pasa banda, filtros pasa bajo, filtros pasa alto…

En este proyecto se ha utilizado la técnica del filtrado, procedimiento empleado para transformar la imagen realzando ciertos aspectos de la misma o minimizando y eliminando otros. Normalmente esta técnica incluye fases del preprocesado como el suavizado (eliminación de ruido), la detección de bordes y el aislado (la nitidez).

Para solucionar nuestro problema de visión artificial se ha optado por emplear un filtro gaussiano que sirve para eliminar el ruido y a diferencia de otros filtros como el de la media, este produce un suavizado más uniforme. Como afirman los autores Sucar, L. E., & Gómez, G. en su libro *Visión Computacional*: "El Filtro Gaussiano, en general, da mejores resultados que un simple promedio o media y se argumenta que la vista humana hace un filtrado de este tipo."

Por último, se aplicará una máscara de detección de colores, en nuestro caso, se definirá un rango entre el azul claro y el azul oscuro detectando así solo los objetos azules que son los que nos interesan para nuestra simulación. Al aplicar esta máscara, la imagen se transformará a binario, quedando como pixeles negros los colores que estén fuera del rango determinado, distinguiéndolos de los objetos azules que quedarán en color blanco.

En la ilustración 20 del apartado 2.2.4 (página 31), se puede observar cómo queda la imagen tras pasar por nuestra máscara de colores.

2.2.3. Contornos

Una vez nuestra imagen ha sido preparada por medio de nuestro filtro y máscara para acondicionar los objetos deseados, la detección de contornos será el método que emplearemos para localizar y determinar la posición de nuestros objetos en el espacio de 2D. El contorno de un objeto es sencillamente una línea circunscrita que une todos los puntos continuos a lo largo del borde del objeto.

La última operación de nuestro sistema de visón será calcular el centroide del contorno. Obtener el punto conocido como centro de masas de nuestro objeto será la mejor manera de determinar la posición final del mismo. Se ha de tener en cuenta que el centroide es un punto real del objeto en espacio tridimensional, por ello no se puede hallar dicho punto exacto con una sola imagen, sin embargo, hallar el centroide del "área lateral" del contorno será suficiente para obtener un punto de referencia aproximado a la posición del centroide del objeto.

En el capítulo 5 se explicará por qué no es necesario obtener el punto exacto del centroide para la solución de nuestro problema.

A continuación, se demostrará el método matemático que se aplica para hallar el centroide de nuestro objeto que será por medio de los momentos.

El momento de imagen mide el grado de dispersión de una abundante serie puntos en el espacio, en nuestro caso pixeles. La fórmula matemática que define esta función es:

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y)$$

La intensidad de los pixeles se representa con la letra I, siendo 0 la ausencia de intensidad de color, es decir el negro y siendo 1 el color blanco y en este rango se encontrarán las distintas tonalidades de gris. Las coordenadas de cada pixel será x,y. Las siglas i,j muestran el orden.

Para calcular el centroide solo usaremos momentos de orden 0 y 1, es decir, M00, M01 y M10.

M00 nos indica el número de píxeles de valor 1, es decir, el área de píxeles blancos. M10 será la suma de las coordenadas x de los píxeles blancos, mientras que M01 será la suma de las coordenadas y de los pixeles negros.

Llegados a este punto ya podríamos calcular el centroide siendo:

-Su componente Xc la relación entre M10 y M00:

$$X_C = \frac{M_{10}}{M_{00}}$$

-Su componente Yc la relación entre M01 y M00:

$$Y_C = \frac{M_{01}}{M_{00}}$$

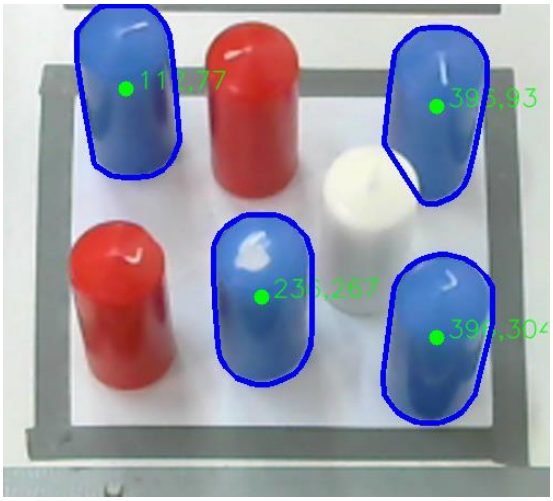


Ilustración 14. Contornos y Centroide

2.2.4. Programación

A la hora de resolver un problema de visión artificial es importante valorar los distintos lenguajes de programación que se pueden utilizar, y escoger el que mejor se adapte a las necesidades del usuario.

Posiblemente podamos resolver un problema como el nuestro utilizando cualquier tipo de lenguaje de programación, sin embargo, para soluciones de visión artificial resulta más interesante usar lenguajes de programación de nivel alto como Matlab, Python o Java aunque también sea posible con lenguajes con un nivel de abstracción medio como C o C++.

El motivo es sencillo, estos programas disponen de herramientas que facilitan la tarea del programador como puede ser el acceso a la biblioteca OpenCV. Esta librería dispone de miles de algoritmos muy interesantes para aplicaciones de visión artificial.

Valorando los diferentes lenguajes se optó por utilizar Python, ya que, investigando en la materia, se observó que era uno de los más empleados para aplicaciones de visión artificial. Además, en mi breve experiencia resolviendo problemas de visión artificial no había trabajado con este lenguaje y creí interesante aprenderlo y aplicarlo.

A continuación, se van a explicar las funciones más importantes que han sido utilizadas:

- cv2.resize: se encarga de recortar la imagen capturada por nuestra cámara enfocando solo el área de detección con el que se va a trabajar.

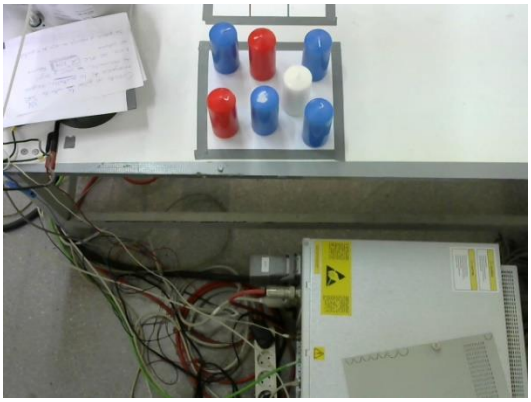


Ilustración 15. Vista de la cámara

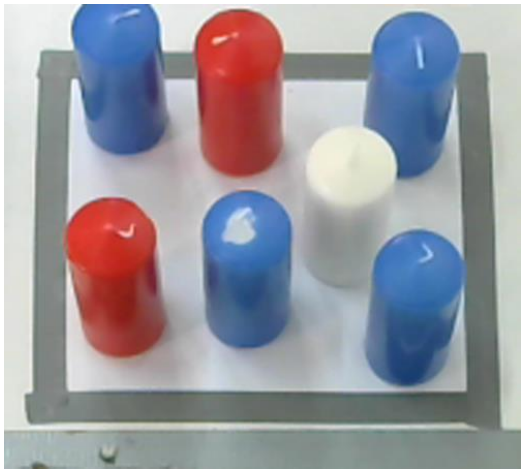


Ilustración 16. Imagen Recortada

- cv2.GaussianBlur: se aplica la operación de filtrado por medio de un filtro Gaussiano. Visualmente no se puede apreciar ninguna diferencia al aplicar este filtro en nuestra imagen a color.

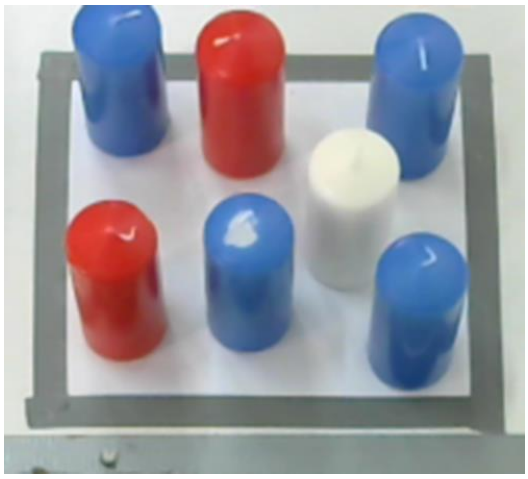


Ilustración 17. Imagen filtrada

- cv2.cvtColor: transforma el espacio de color de RGB a HSV, OpenCV por defecto trabaja con RGB, sin embargo, utilizar el modelo HSV resulta más sencillo a la hora de definir el rango de colores de la máscara por ello realizamos esta transformación.

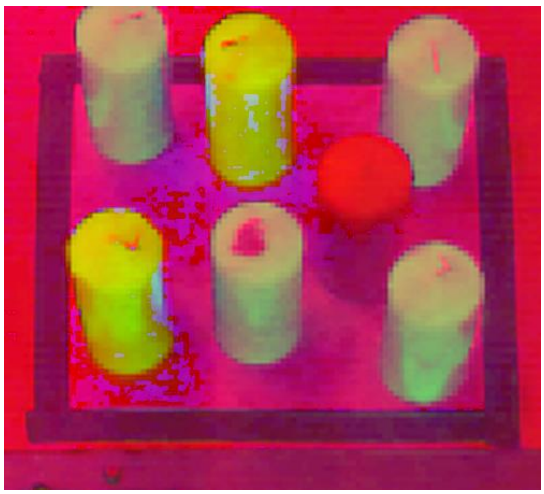


Ilustración 18. Imagen transformada a HSV

Como se puede apreciar al hacer la transformación de colores lo que en nuestra imagen era azul ahora es un verde pistacho.

- cv2.inRange: esta función de OpenCv se usará para definir el rango de colores de nuestra máscara. Solo filtrará los colores que se encuentren entre un verde claro y un verde oscuro.

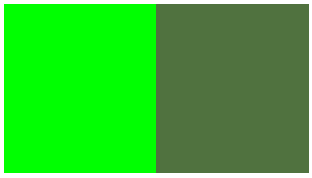


Ilustración 19. Rango de verdes máscara

Código:

```
#rango1=verdeoscuro
rango1 = np.array([100, 45, 45])
#rango2=verdeclaro
rango2 = np.array([120, 255, 255])
mascara=cv2.inRange(hsv,rango1, rango2)
```

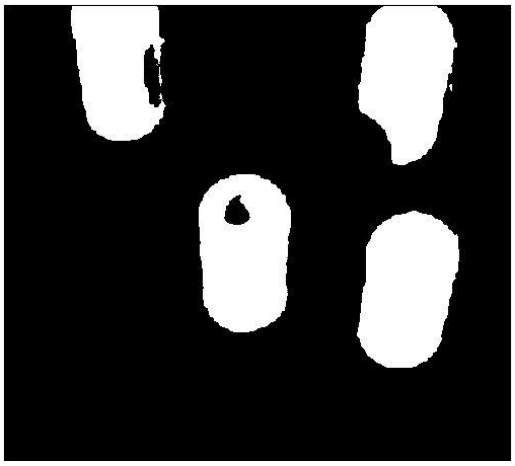


Ilustración 20. Máscara

- cv2.findContours: detecta los contornos de los objetos.
- cv2.drawContours: dibuja dichos contornos.
- cv2.moments: calcula el momento de los contornos realizado.

El código del programa realizado en Python se encuentra en el Anexo 8.5.

3. Brazo robótico

3.1. Introducción a la robótica industrial

3.1.1. Definiciones

Antes de hablar del origen y en la evolución de lo que hoy conocemos por robótica, tenemos que precisar qué es exactamente la robótica y cuál es la definición de robot industrial.

Siguiendo la definición de la Asociación de Industrias de Robótica (RIA), "un robot industrial es un manipulador multifuncional reprogramable diseñado para desplazar materiales, piezas, herramientas o dispositivos especiales, mediante movimientos variables programados para la ejecución de una diversidad de tareas".

Por otro lado, un robot industrial, acorde con la definición que nos aporta la norma ISO 8373 es: "Manipulador multifuncional, controlado automáticamente, reprogramado en 3 o más ejes, que puede estar fijo o móvil para uso en aplicaciones de automatización industrial".

3.1.2. Evolución histórica

Desde el origen del ser humano los trabajos y tareas que se han llevado a cabo, han ido evolucionando con el objetivo siempre de mejorar la eficiencia de los mismos. En la prehistoria se creaban herramientas con palos y piedras para realizar tareas de forma más cómoda y rápida.

Siglos después se fueron desarrollando muchas técnicas que permitían reducir el esfuerzo humano al realizar un trabajo como por ejemplo las poleas.

Fue entonces, a mediados del siglo XX, con la llegada de la automatización cuando nuestros sistemas de producción se revolucionaron completamente. Los avances ya no solos consistían en desarrollar herramientas que facilitaban las tareas, sino que, gracias a la llegada de las máquinas, ciertos trabajos ya no requerían de la intervención humana.

En nuestros días, la parte más visible de la automatización se puede encontrar en la robótica industrial y el auge de esta nos ha aportado grandes beneficios en la industria en cuanto a productividad, seguridad, calidad y flexibilidad.

Elektro es el nombre del que es considerado como el primer robot de la historia que fue diseñado en 1939 por Joseph Barnett, un ingeniero de la "Westinghouse Electric Corporation". Este robot humanoide tenía un tamaño elevado en

comparación con los androides de hoy en día, midiendo unos dos metros de altura y pesando hasta 120 kg. Entre las actividades que podía realizar se encontraban caminar, hablar y mover los brazos.

Para hablar del primer robot industrial tendremos que avanzar hasta el año 1954 donde se asientan las bases del primer robot industrial "Unimate", creado por el ingeniero norteamericano George Devol en la fábrica de General Motors de Trenton. La función de esta máquina era la de llevar las piezas fundidas en molde hasta la cadena de montaje y soldarlas al chasis del automóvil. Fue revolucionario este robot ya que realizaba una tarea que era peligrosa para un ser humano debido a que los gases que se producían en este trabajo eran perjudiciales para la salud. Gracias a su éxito, pronto fue producido en masa.

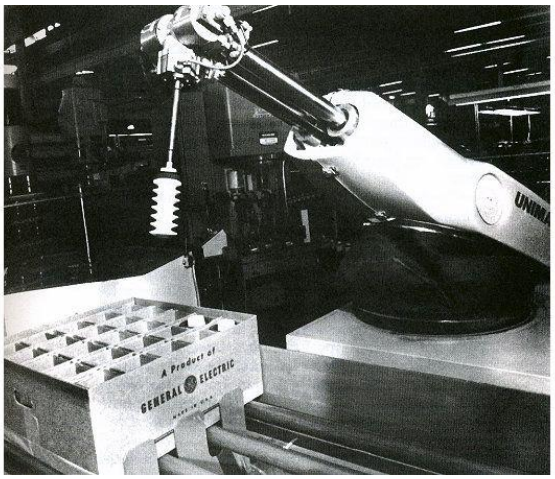


Ilustración 21. Primer robot industrial "Unimate"

Posteriormente, ASEA (Compañía Eléctrica General Sueca) construyó por el año 1973, el primer robot con accionamiento eléctrico, denominado IRb6.

Por último, hay que destacar en la evolución de la robótica industrial lo que a día de hoy es la base de los robots industriales. En 1975, PUMA (Brazo Articulado Universal Programable por sus siglas en inglés), fue diseñado por el ingeniero Victor Scheinman y se trataba de un brazo robótico capaz de realizar tareas de Pick & Place, es decir, de llevar un objeto de un lugar a otro.

En el último siglo la robótica ha ido evolucionando cada vez más, se han desarrollado nuevos métodos de sincronización de robots, métodos para cooperar con los operarios, nuevos sistemas de manipulación y agarre, técnicas de detección de fallos, mejora de interfaces…

3.1.3. Elementos de un robot

Los elementos que conforman un robot son:

-Actuadores: son aquellos dispositivos que dotan de movimiento a las articulaciones del robot, por medio de sistemas hidráulicos, neumáticos o eléctricos, siendo este el último el más utilizado gracias a los servomotores.

-Brazo manipulador: es el elemento que conforma la mecánica del robot. Está compuesto por elementos rígidos denominados eslabones que unidos entre sí forman las articulaciones. El objetivo de este elemento será el de posicionar y orientar el último eslabón de nuestro robot, que posee la herramienta con la que se va a trabajar.

-Sensores: su misión es comunicar cuál es el estado de nuestro robot manipulador. Se pueden diferenciar en función de la posición en la que se encuentren con respecto al robot en sensores internos o externos. Un ejemplo de sensores internos de posición son los encoders, encargados de medir y controlar la posición angular de las articulaciones. Mientras que un sensor de proximidad, que manda la información de parar al robot antes de colisionar con un objeto, sería un sensor de tipo externo.

-Controladora: está constituida por uno o varios microcontroladores que supervisan y controlan el movimiento del robot. Este dispositivo dispone en su implementación de toda la información necesaria acerca de la cinemática directa e inversa y gracias a esto es capaz de procesar y regular todo el movimiento del robot.

3.2. Robot IRB 120

3.2.1. Consola FlexPendant

El FlexPendant de nuestro robot corresponde a la unidad de programación y esto no es otra cosa que un instrumento con el que programar el movimiento del robot, es decir, es un sistema HMI (“Human Machine Interface”) que nos permite la comunicación directa con el robot.

A continuación, se procede a explicar las funcionalidades de la consola tal y como aparecen reflejadas en el manual.

“La Unidad de Programación (denominada en ocasiones FPU o TPU) es un dispositivo que maneja muchas de las funciones relacionadas con el uso del sistema de robot, como ejecutar programas, mover el manipulador, crear y editar programas de aplicación, etc.”

"Se compone de elementos de hardware, como botones y un joystick, y de software y es un ordenador completo en sí. Se conecta al controlador a través de un cable con conector integrado."

Ilustración 22. Partes del FlexPendant

A	"Cable conexión."
B	"Pantalla táctil."
C	"Pulsador de paro de emergencia."
D	"Joystick."
E	"Conector USB para los sticks de memoria para guardar o recuperar los programas."
F	"Dispositivo de habilitación."
G	"Puntero."
H	"Pulsador de Reset."

"La Unidad de Programación cuenta con varios botones de hardware dedicados. Cuatro de ellos son programables."

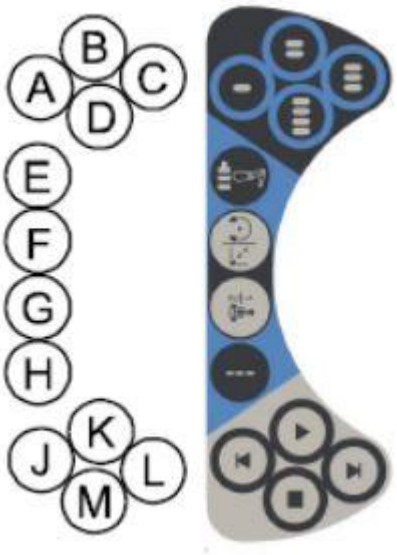


Ilustración 23. Botones FlexPendant

A,B,C,D	"Teclas programables."
E	"Seleccionar una unidad mecánica: Unidad de Robot / Ejes Externos."
F	"Botón acceso rápido a selección de movimientos: Reorientación / Lineal,"
G	"Botón acceso rápido a selección de movimientos: Ejes 1, 2,3 / Ejes 4, 5,6"
H	"Botón acceso rápido a selección de movimientos: Activar / Desactivar Incrementos."
J	"Botón INICIAR. Inicia la ejecución del programa."
K	"Botón RETROCEDER un paso. Ejecuta el programa una instrucción hacia atrás"
L	"Botón AVANZAR un paso. Ejecuta el programa una instrucción hacia delante"
M	"Botón DETENER. Detiene la ejecución del programa."

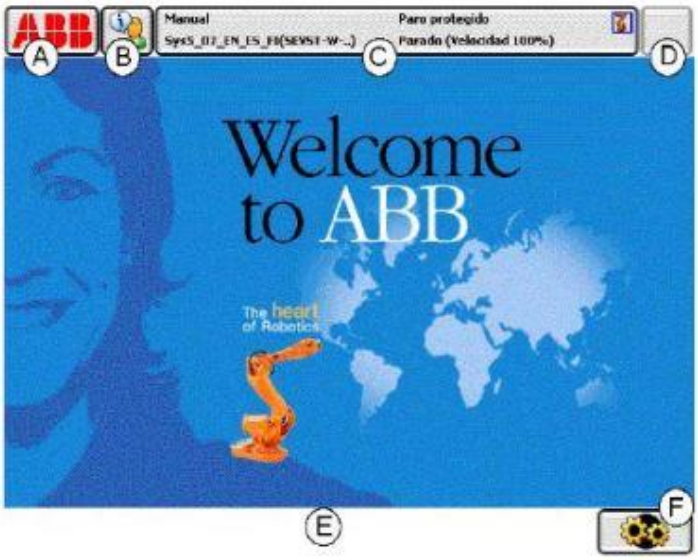


Ilustración 24. Pantalla táctil FlexPendant

A	"Menú ABB."
B	"Ventana de operador."
C	"Barra de estado."
D	"Botón Cerrar."
E	"Barra de tareas."
F	"Menú de configuración rápida."

"Menú ABB: es el menú principal, permite visualizar las ventanas y aplicaciones principales."

Ilustración 25. Menú ABB FlexPendant

"Ventana de operador: la ventana de operador muestra los mensajes del programa. Suelen aparecer cuando el programa necesita alguna respuesta por parte del usuario."

"Barra de estado: la barra de estado muestra información sobre el estado del sistema y permite acceso a los mensajes de error."

"Botón Cerrar: al tocar el botón Cerrar se cierra la vista o aplicación que esté activa actualmente."

"Barra de tareas: se pueden abrir varias vistas desde el menú ABB, pero solo se puede trabajar con una de ellas en cada momento. La barra de tareas muestra todas las vistas y aplicaciones abiertas y permite cambiar entre ellas."

"Menú de configuración rápida: el menú de configuración rápida contiene opciones sobre movimientos y ejecución de programas."

3.2.2. Controladora IRC5

Como se ha explicado en el capítulo 3.1.3, la controladora es el dispositivo capaz de procesar y regular todo el movimiento del robot.

En nuestro caso, la controladora IRC5, es el módulo que utiliza ABB para sus robots de pequeño tamaño como (IRB120, IRB1520, IRB1600, IRB360)."

Tal y como queda reflejado en el manual, "el controlador tiene dos partes: el módulo de control y el módulo de accionamientos."

- "El módulo de control contiene todos los elementos electrónicos de control, como el ordenador principal, las tarjetas de E/S y la unidad de almacenamiento."
- "El módulo de accionamiento contiene todos los elementos electrónicos de alimentación que proporcionan la alimentación a los motores del robot. Un módulo de accionamientos"

Como se puede observar en la siguiente imagen, nuestra controladora dispone de un panel de control con el que podemos realizar diferentes acciones.

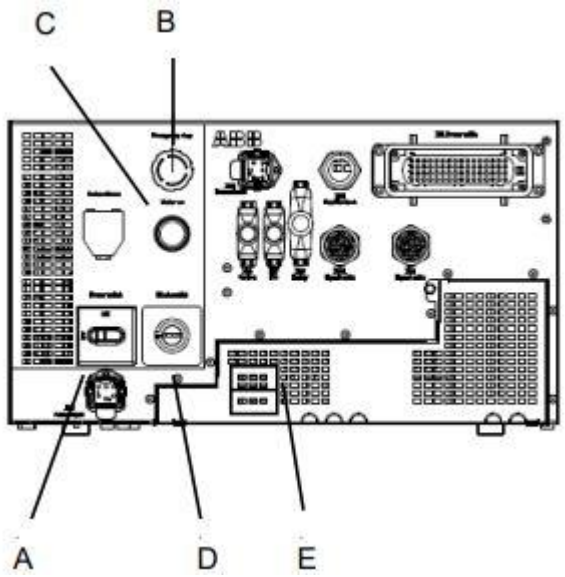


Ilustración 26. Panel de control armario Compact

Parte	Descripción	Función
A	"Interruptor principal"	"Interruptor de encendido/apagado para el apagado del sistema."
B	"Pulsador paro emergencia"	"Pulsador de paro del robot cuando hay una emergencia. Desconecta la alimentación de los motores del robot."
C	"Pulsador Motor ON"	"Pulsador petición motores ON y rearmar paro emergencia. Lámpara fija estado de motores ON, intermitente estado de motores OFF."
D	"Selector de modo"	"Selector para trabajar en modo Automático o Manual."
E	"Puerto Ethernet"	"Para conectarse con el sistema para tareas de Servicio, mediante la aplicación RobotStudio en modo Online."

Por último, hay que mencionar los dos modos de operación que presenta nuestro controlador:

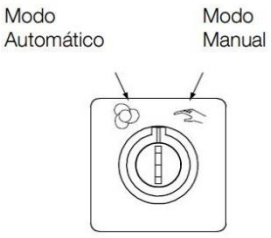


Ilustración 27. Modos de operación

Modo de operación	Descripción
"Automático"	"-Modo de producción, se usa para ejecutar los programas. -El robot se moverá a la máxima velocidad programada. -El dispositivo de habilitación está desconectado, de forma que el robot pueda moverse sin intervención humana. -Todos los mecanismos de protección (GS, AS y SS) están activos durante el funcionamiento."
"Manual"	"Pulsador paro emergencia"

3.2.3. Tarjeta interna DSQC652

Nuestra controladora tiene incluida en el armario Compact una tarjeta interna de E/S, la "DSQC652". Contiene una serie de clemas verdes que están conectados con la tarjeta interna DSQC652.

En este proyecto solo se ha utilizado esta tarjeta para alimentar la electroválvula que controla la apertura de la pinza del robot, utilizando la bornera de salida. Aun así, se ha dejado preparada la bornera de entrada por si en futuros proyectos fuese necesario incorporar entradas.

Ilustración 28. Cableado en borneras de la tarjeta DSQC652

Se dispone de una alimentación externa a través de una fuente de alimentación de 24V. Dicha alimentación se conecta a nuestra bornera de salida (XS14), los 24V de la fuente al pin 10 de la bornera y los 0V al pin 9.

En la bornera de entrada (XS12) lo que se ha hecho ha sido simplemente puentear los 0V hasta el pin 9 de dicha clema.

Finalmente se han conectado los pines 1 y 2 de la bornera XS14 que son las salidas que se utilizan para accionar la electroválvula de la pinza, una salida para abrir y otra para cerrar ya que es una válvula de doble efecto. La tierra de la electroválvula ira conectada al pin 9 XS15 o XS14, realmente son el mismo punto ya que internamente están unidos.

En el anexo 8.3 se puede ver el esquema eléctrico de la tarjeta interna E/S DSQC652.

Finalmente, en el apartado 3.2.5 de configuración se explicará como activar esta tarjeta desde el FlexPendant.

3.2.4. La herramienta "Pinza"

Una de las grandes ventajas que tiene el uso de robots industriales es que se le pueden acoplar casi cualquier tipo de herramientas. A continuación, se van a nombrar algunos ejemplos de herramientas que se les pueden acoplar:

- Pistolas de pintura
- Herramientas de soldadura
- Herramientas de mecanizado
- Pinzas
- Ventosas

En nuestro caso para realizar la operación de pick & place la herramienta utilizada ha sido una pinza, la "SCHUNK SGB50", tal y como esta descrita en su página web consiste en: "una pequeña pinza angular de plástico, con pistón de simple efecto y retorno por muelle".

A ella se le han añadido dos piezas rectangulares a cada uno de los lados, a modo de extensión, para que sea más sencillo manipular objetos cilíndricos.

Esta herramienta ha sido proporcionada por el Dpto. de Automática, Ingeniería eléctrica y Tecnología electrónica de la Universidad Politécnica de Cartagena (UPCT).

Ilustración 29. Herramienta "Pinza"

Por último, nuestra pinza consta de una parte, la cual ha sido diseñada mediante una impresora 3D, que es la que se encaja a nuestro robot.

Debido a que el accionamiento de nuestra pinza es neumático, se ha requerido de una electroválvula que controle el fujo de aire de forma que se accione pinza abierta o pinza cerrada. Como se ha mencionado en capítulos anteriores, esta herramienta se acciona desde la tarjeta interna de E/S del robot. También se debe conocer que podría ser accionada por medio de un PLC.

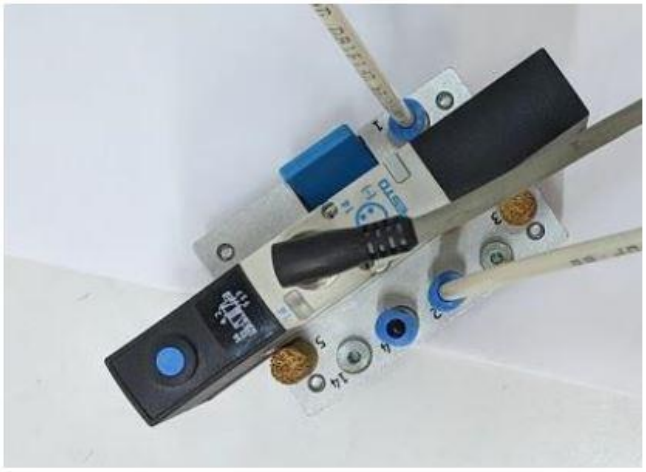


Ilustración 30. Electroválvula

Como se observa en la ilustración 30, el pin 1 de nuestra electroválvula corresponde al pin de entrada que se conecta a nuestro sistema de aire a presión. El pin 2 es un pin de salida que se conecta a la entrada de la pinza y es el que determina si la pinza está en estado abierto o cerrado. El pin 4 también sería de salida, pero no se utilizará para nuestra pinza porque posee sistema de retorno por muelle.

En lo referente al cableado eléctrico que une la bornera con la electroválvula, se han instalado tres cables, dos de color negro y uno de color azul.

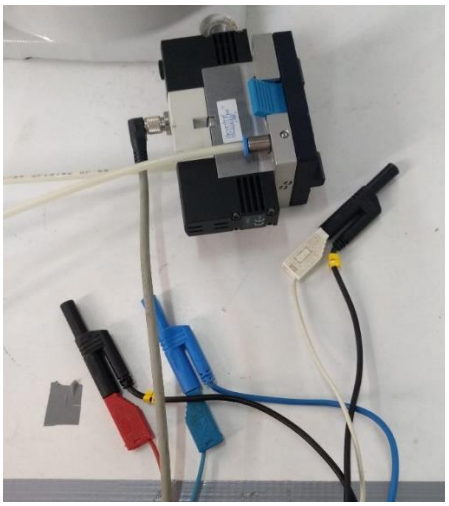


Ilustración 31. Cables Salida Electroválvula

Los que presentan el mismo color, se extenderán con otro cable y se conectan a los pines 1 y 2 de la clema XS14, es decir, la bornera de salida que nos proporciona los 24V. En nuestro caso el rojo irá al pin 1 del XS14 y el cable blanco al pin 2.

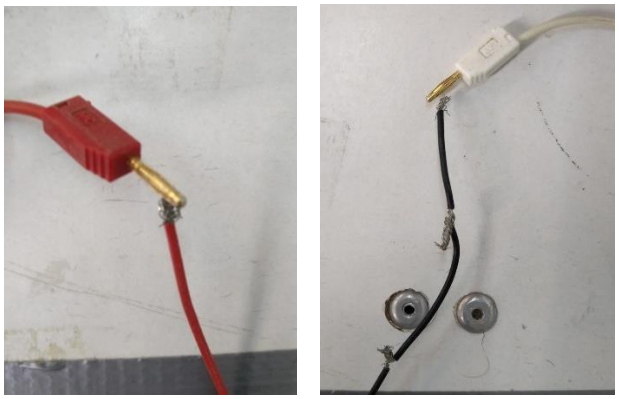


Ilustración 32. Cables bornera de salida XS14

Mientras que el cable de distinto color se conecta a tierra que como ya se ha dicho valdría tanto en el pin 9 de la bornera XS14 como en la XS15.

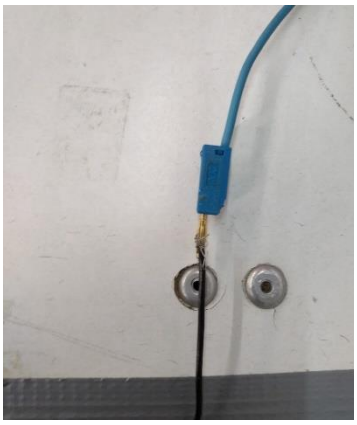


Ilustración 33. Cable a tierra

Por último, es importante acordarse de abrir la llave del sistema de aire comprimido que hay instalado en nuestro laboratorio.

3.2.5. Configuración

En este apartado se va a resumir toda la configuración que se ha realizado en lo referente al robot. Puesto que se partía con la configuración ya realizada por el alumno Santiago Martínez Nicolás, en su proyecto: "*Automatización de una planta robotizada para operaciones de almacenaje*", se van a realizar muchos pasos descritos en dicho proyecto.

-Configuración de la IP del Robot.

Es de suma importancia establecer una dirección IP correcta para que los dispositivos en red sean capaces de establecer comunicación.

Para establecer o modificar la dirección IP en nuestro Robot ABB debemos hacer uso del FlexPendant. En él, abriendo la pestaña de opciones, accedemos a reiniciar, dentro de reiniciar, accedemos a avanzadas… marcamos la opción Boot Aplication y damos en Siguiente.

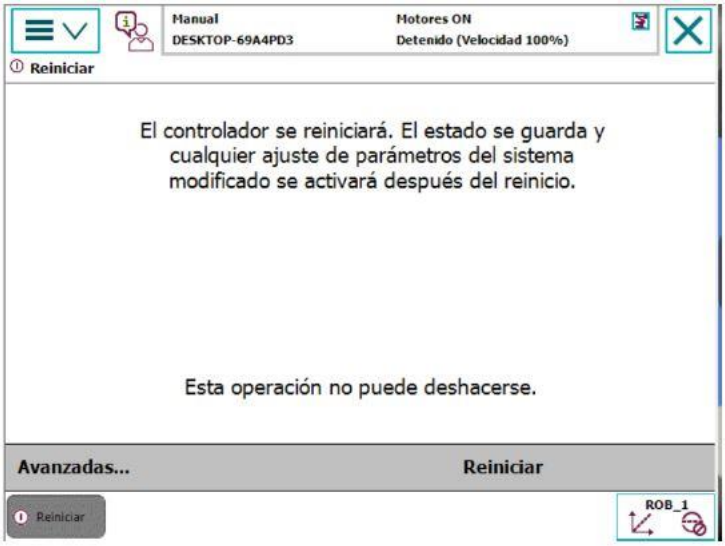


Ilustración 34. Ventana emergente al pulsar en Reiniciar en FlexPendant.

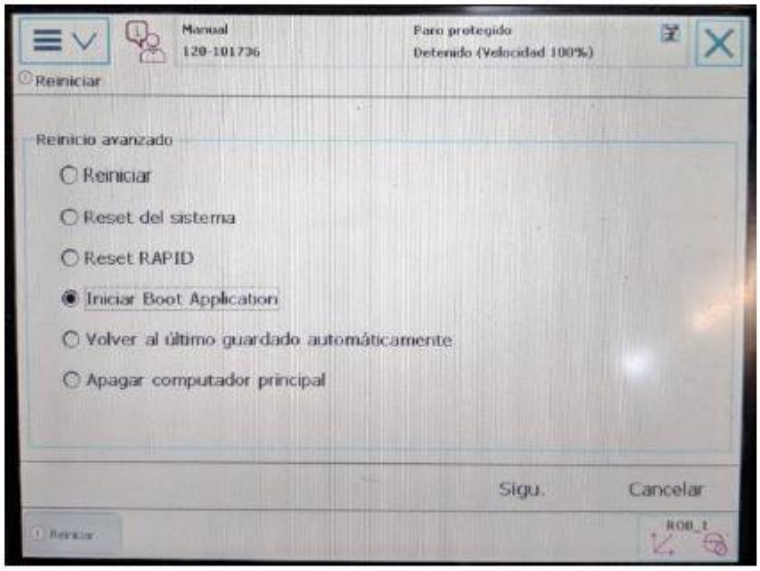


Ilustración 35. Pantalla de avanzadas… en FlexPendant

Una vez se haya reiniciado el sistema, se habrá accedido a al menú de configuración avanzado.

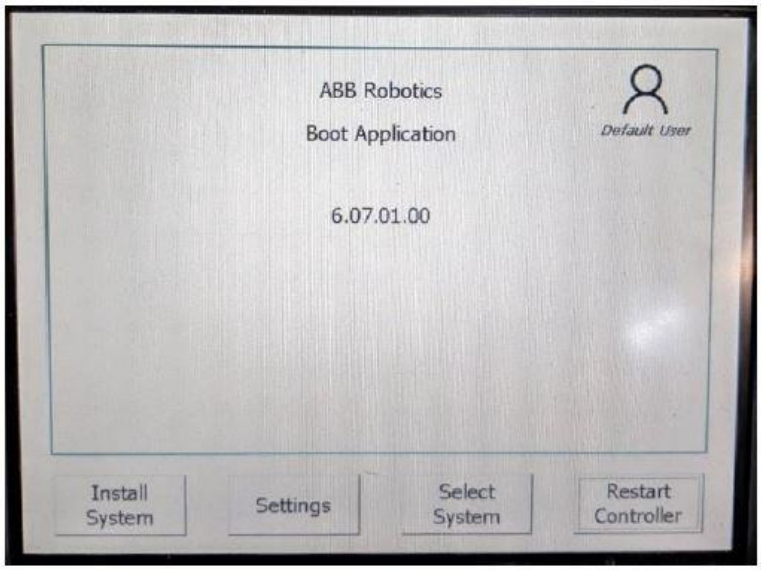


Ilustración 36. Boot Aplication

Continuamos el proceso pulsando en la opción Select System que nos aparece y le damos a Seleccionar.

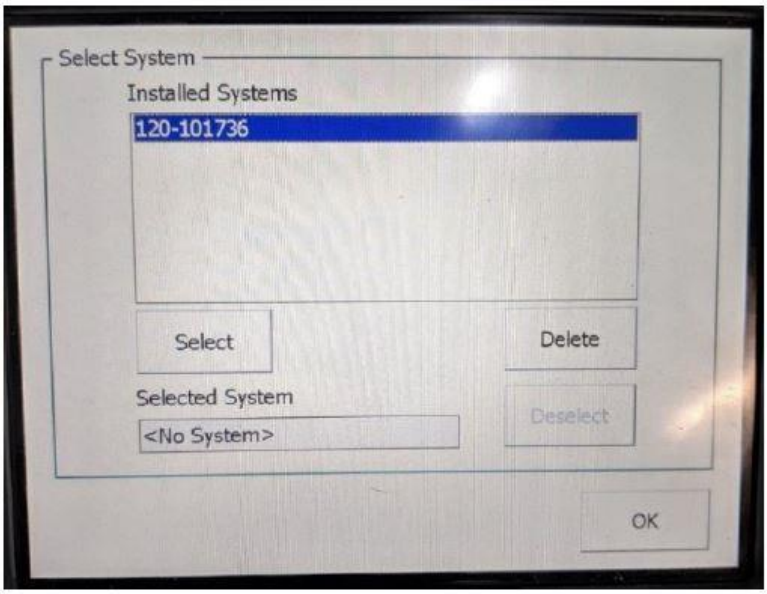


Ilustración 37. Ventana emergente de Select System

Como podemos observar en la figura siguiente, aparece una ventana emergente y tras leer la advertencia se presionará OK.

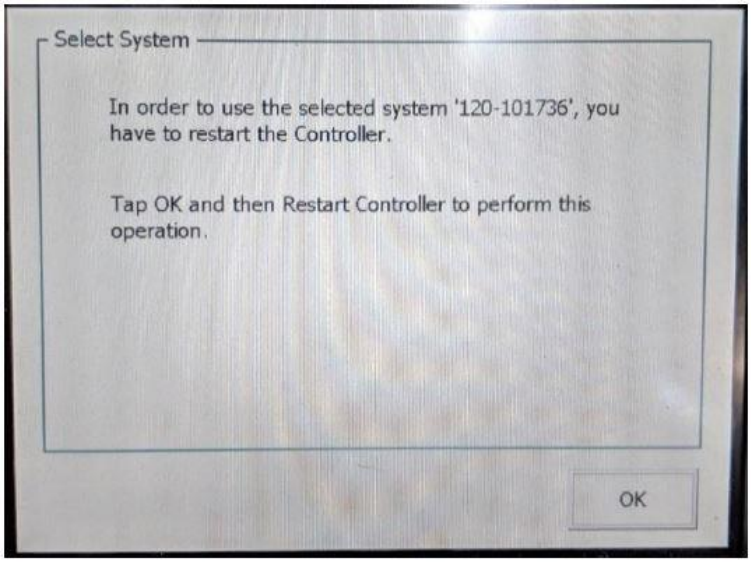


Ilustración 38. Ventana emergente al pulsar OK

Seguidamente, nos aparece la ventana Settings la cual, podemos analizar en la figura siguiente, en la que introducimos tanto la dirección IP que le queramos dar "192.168.1.10" (pulsando en use the following IP settings), como la máscara de subred (255.255.255.0) y la puerta de enlace (192.168.1.0).

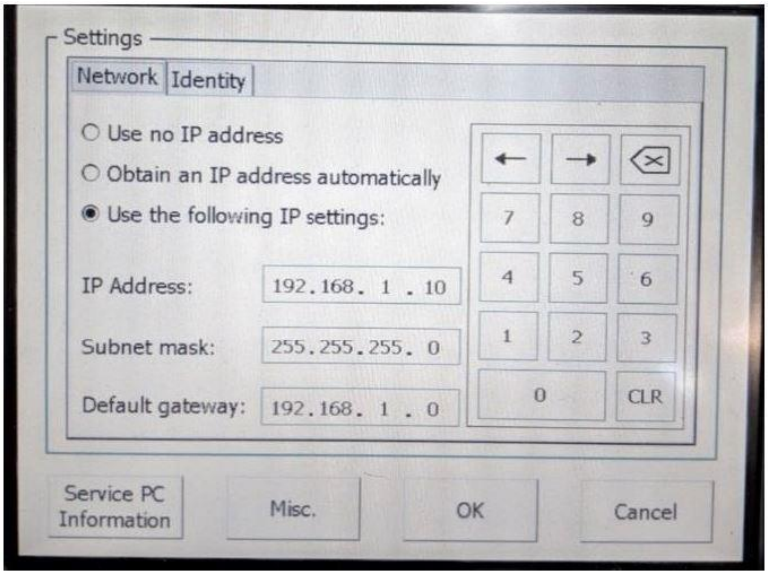


Ilustración 39. Configuración IP del robot.

En la zona superior pulsamos identity, esta opción permite modificar el nombre del robot, para el caso se dejará el mismo nombre que poseía.

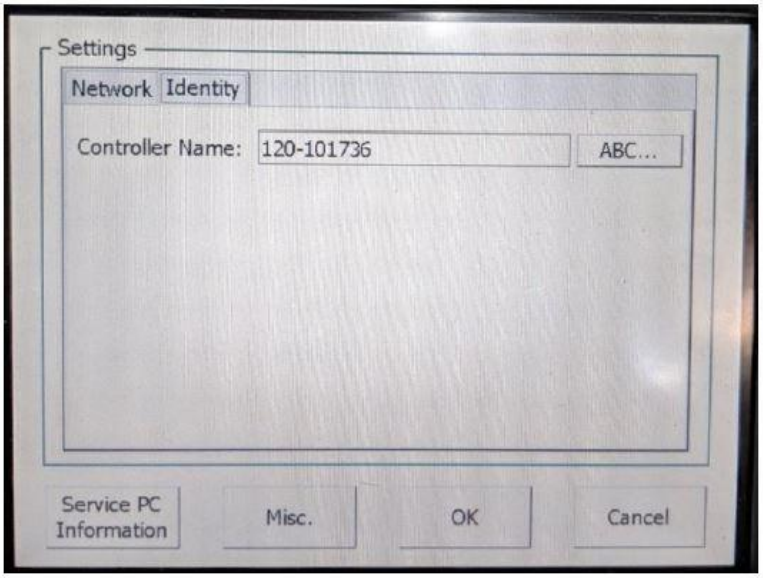


Ilustración 40. Configuración del nombre de la controladora.

Finalmente, se presiona OK, a continuación, restart controller y así, se reinicia el sistema y se aplica la configuración insertada.

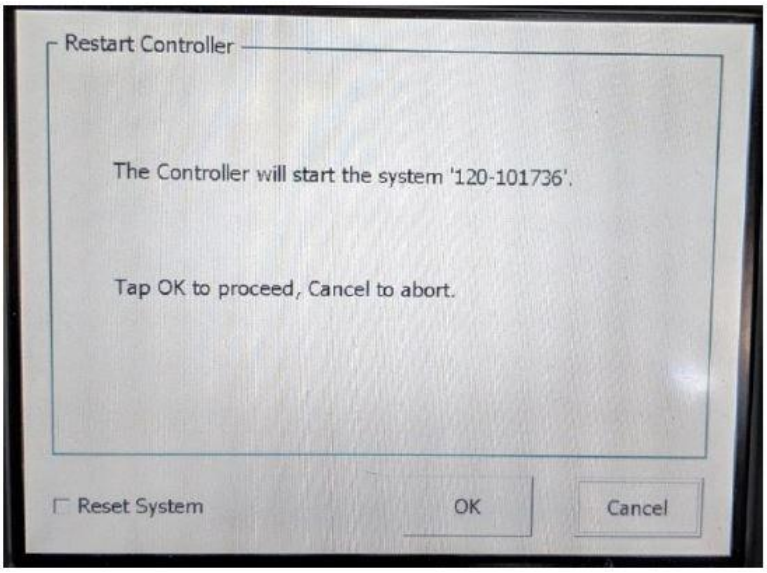


Ilustración 41. Finalización de la configuración IP del Robot.

-Activación de tarjeta interna de entradas y salidas DSQC652.

Para proceder a su activación, en primer lugar, desde el FlexPendant se accede a Panel de control > Configuración.

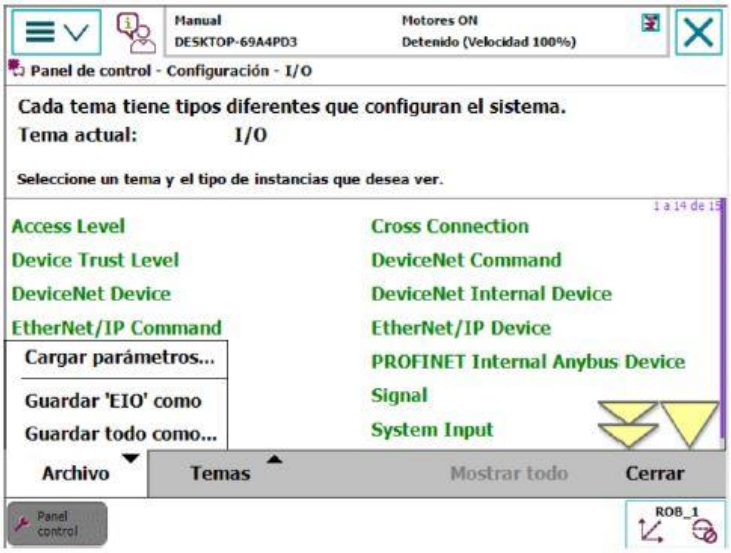


Ilustración 42. Configuración de la tarjeta DSQC652.

Dentro de esta nueva ventana, como podemos observar en la figura anterior, se presionará en el desplegable, Archivo y dentro de este desplegable se presiona en la opción Cargar parámetros…

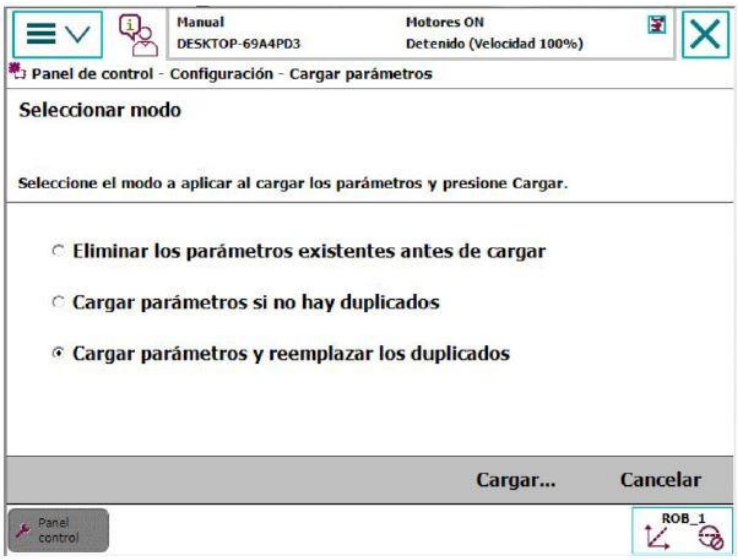


Ilustración 43. Ventana previa a la carga de parámetros de la tarjeta DSQC652.

Nuevamente, aparecerá otra ventana en la que dará 3 opciones a elegir. Se selecciona la tercera opción, Cargar parámetros y reemplazar los duplicados, y a continuación Cargar…

Después de realizar estos pasos, se abrirá un explorador de archivos. Es entonces cuando se deberá buscar el directorio DeviceNet donde estarán todos los dispositivos de entrada y salida que se pueden instalar.

En el caso que acontece este proyecto, el directorio se encontraba en la ruta: /hd0a/120-101736/Products/RobotWare_6.07.1011/utility/service/ioconfig/ DeviceNet

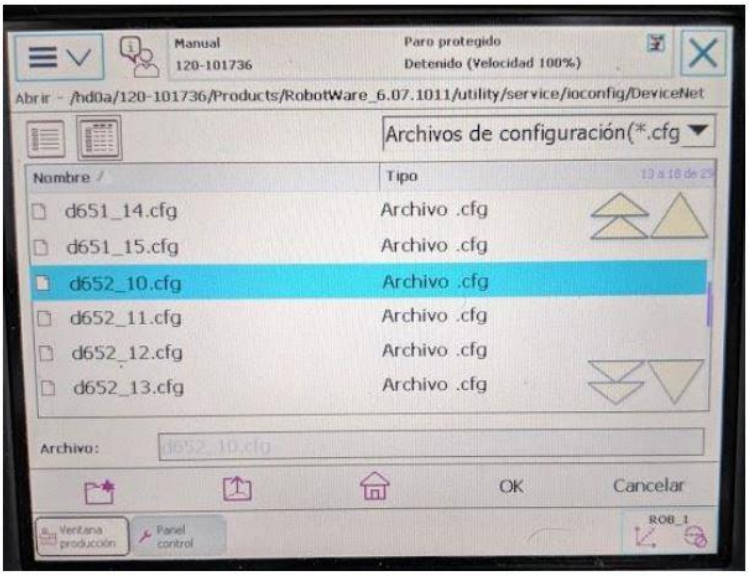


Ilustración 44. Archivo de configuración de la tarjeta DSQC652.

Seleccionamos el archivo de configuración d652_10.cfg, a continuación, se pulsa en OK y se finaliza reiniciando el sistema para que se guarde la configuración. Una vez la controladora se ha reiniciado ya aparecerá el dispositivo de entradas y salidas con todas sus entradas y salidas mapeadas en el sistema.

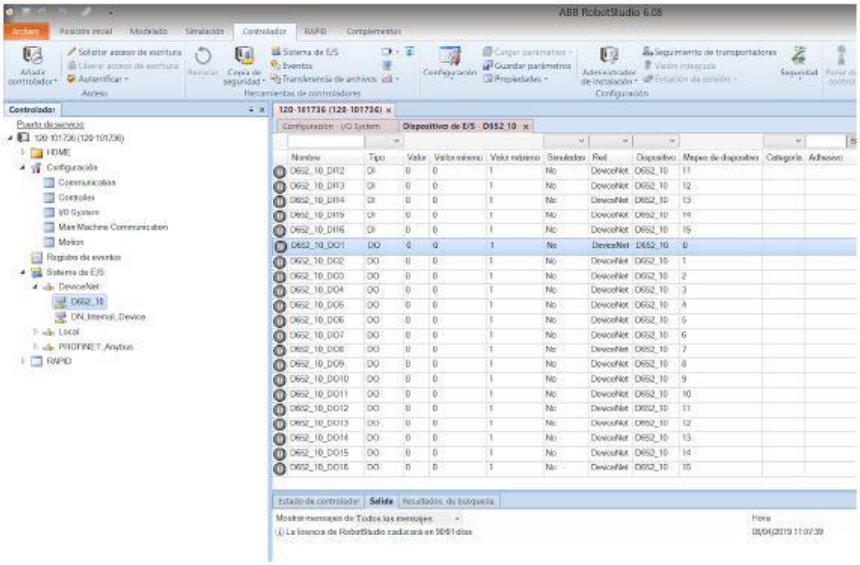


Ilustración 45. Visualización de entradas y salidas mapeadas en el dispositivo DSQC652.

Viendo el mapeo y observando el esquema eléctrico se puede observar cuál es la entrada o salida correspondiente a cada acceso eléctrico de los borneros, ya que los números en el esquema eléctrico se corresponden con los números de mapeo en el dispositivo.

-Definición de la herramienta de trabajo (TCP).

Cuando una herramienta es instalada en el robot es importante definirla en el sistema, puesto que, en ese momento el sistema de referencia del extremo del robot ya no estará en la brida, sino que estará en el extremo de la herramienta o en el elemento de actuación que tenga. Definir la herramienta es interesante para ejecutar movimientos de reorientación, en la que se observa como la herramienta se orienta con respecto a un punto fijo en el extremo de aplicación de esta.

Para definir la herramienta, se accede desde el FlexPendant a Datos de programa y a continuación, se selecciona el tipo de dato tooldata.

Accediendo a tooldata, se muestra una ventana en la que aparece una herramienta ya definida, la tool0, cuyo sistema de referencia está posicionado en la brida del robot, es decir, no hay ninguna herramienta instalada.

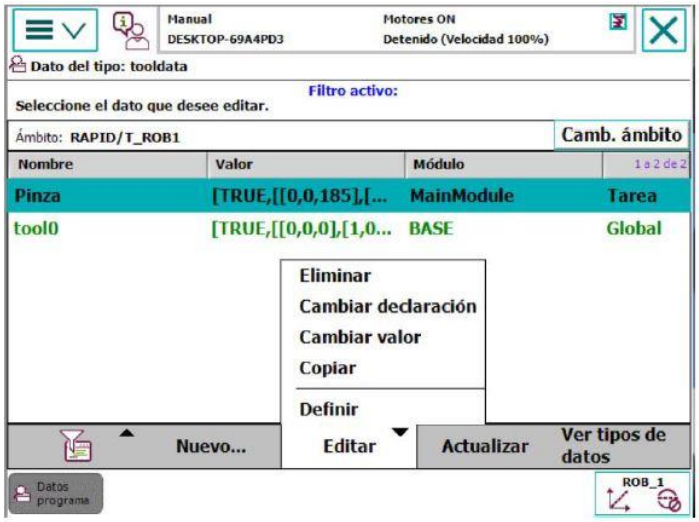


Ilustración 46. Configuración de la pinza en el robot.

Pulsando en Nuevo…, como se observa en la figura anterior, automáticamente se crea una nueva herramienta vacía, la cual, deberá ser definida. Podemos modificar el nombre de la herramienta accediendo a Editar y a continuación Cambiar declaración, entonces se mostrará una ventana como la de la figura siguiente.

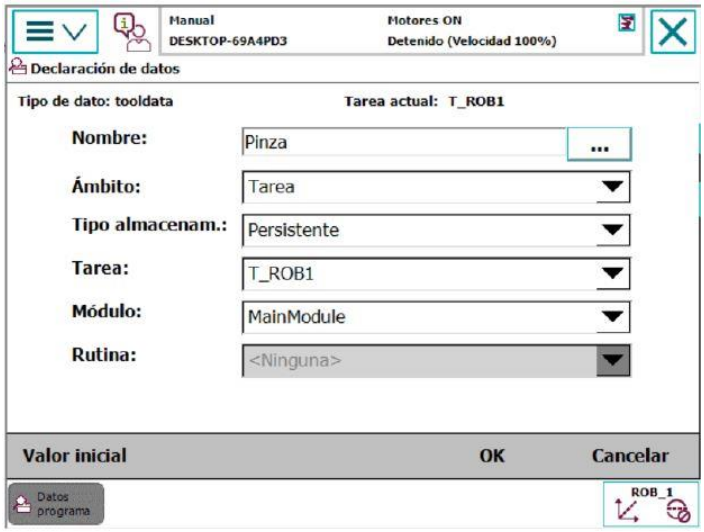


Ilustración 47. Creación del tipo de dato de información de la pinza.

En la casilla correspondiente a Nombre, pulsando en el icono con tres puntos suspensivos, se accede al teclado del FlexPendant para modificar el nombre de la herramienta. Realizado el cambio de nombre y pulsando en OK se guardará la nueva configuración de nombre de la herramienta (TCP).

Ahora, solo queda definir las características de la herramienta. En el caso objeto de este proyecto, el extremo de la herramienta se situaba justo 185 mm desplazado en el eje Z tomando como referencia la brida o la tool0 que en este caso sería el mismo.

Existen dos modos de definir la herramienta: por puntos o por introducción directa de datos.

En este caso es más sencillo introducción directa de datos, ya que tiene una geometría sencilla y tiene un peso lo suficientemente pequeño como para despreciarlo y junto con él, la inercia.

Para establecer los valores de la herramienta, basta con acceder en Editar y seguidamente acceder a la opción Cambiar valor.

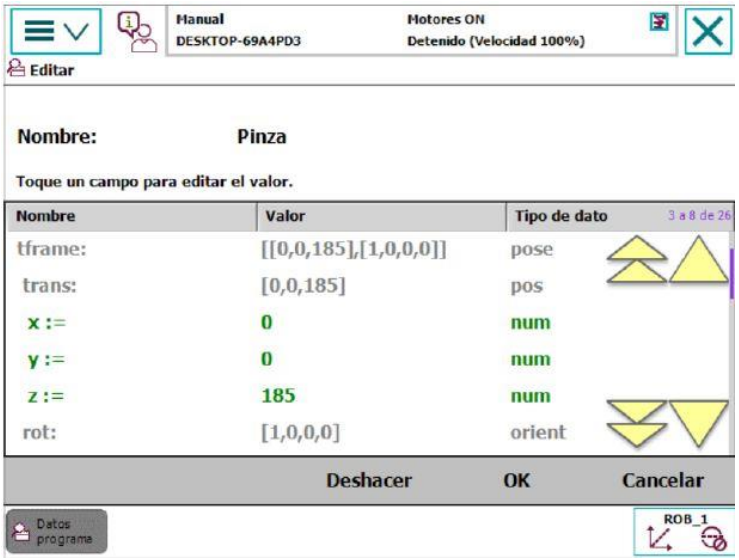


Ilustración 48. Inserción manual de datos de configuración de la pinza.

Conociendo que el extremo de la herramienta se sitúa justo a 185 mm en el eje Z del centro de la brida, se debe declarar las coordenadas en esta ventana, siendo las coordenadas.

Aunque se ha dicho anteriormente que la masa de la herramienta es despreciable, es interesante que esta no tenga configurado un valor negativo, puesto que en ocasiones el sistema, al ser un peso muy ligero, no consigue detectarlo e inserta automáticamente el valor -1. Esto ocurre sobre todo en la definición de la herramienta por puntos.

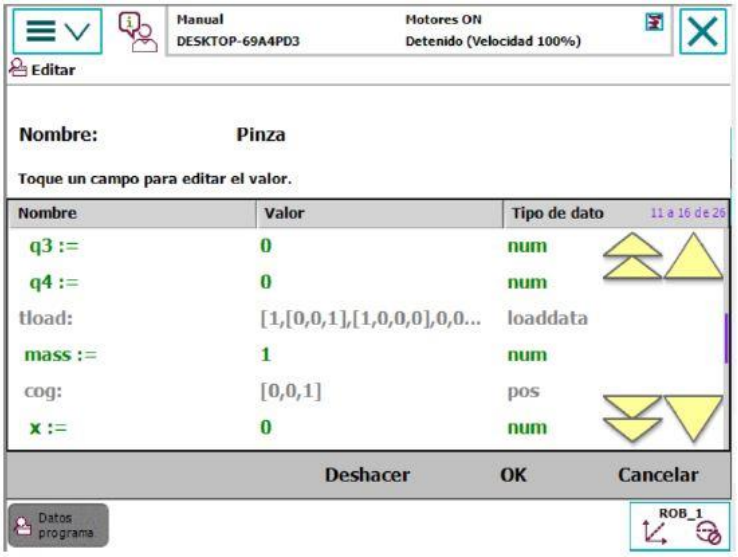


Ilustración 49. Inserción del valor de masa de la pinza.

Por lo tanto, podemos modificar este valor, en el apartado mass como: Mass := 1.

Este valor se define en kilogramos.

Si por el contrario se quisiera configurar la herramienta por puntos, este procedimiento se realizaría accediendo en Editar y Definir. A continuación, en la ventana emergente se deberá indicar el número de puntos con los que se desea definir la herramienta, a mayor número de puntos mayor precisión a la hora de definir la herramienta.

El procedimiento consiste en ir almacenando puntos posicionando la herramienta sobre un mismo punto, pero con la herramienta orientada en diferentes direcciones como se puede observar en la figura siguiente.

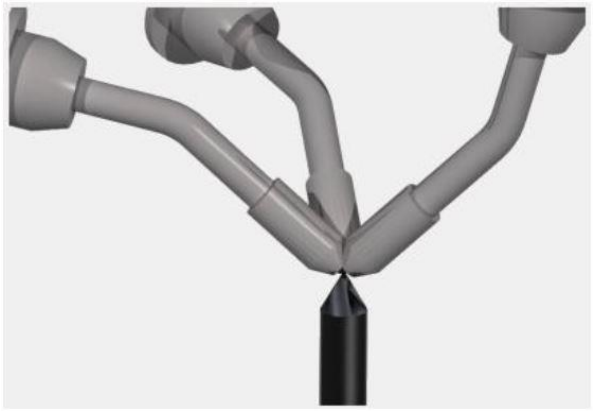


Ilustración 50. Método de 3 puntos.

-Pulsadores personalizables del FlexPendant para manejo de la pinza.

La consola FlexPendant dispone de teclas personalizables gracias a las cuales podemos asignarles salidas de nuestro robot a dichos botones.

En este proyecto se ha configurado una tecla para realizar la acción de abrir y/o cerrar la pinza, cambiando la posición en la que se encuentre nuestra herramienta. De forma que si está abierta se cerrará y viceversa.

Ilustración 51. Teclas personalizables FlexPendant

A continuación, se explicará cómo se ha configurado esta opción. En primer lugar, hay que dirigirse al Panel de Control de nuestra consola.

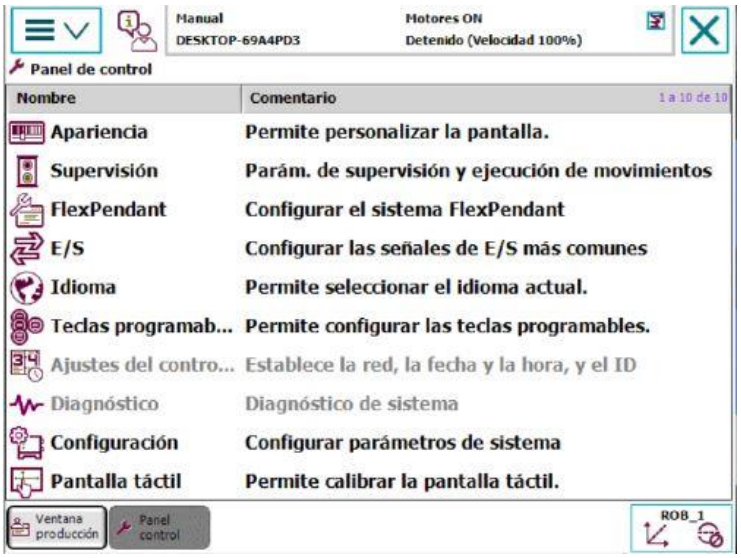


Ilustración 52. Panel de Control

En este menú seleccionamos la opción de teclas programables y nos aparecerá la siguiente pantalla:

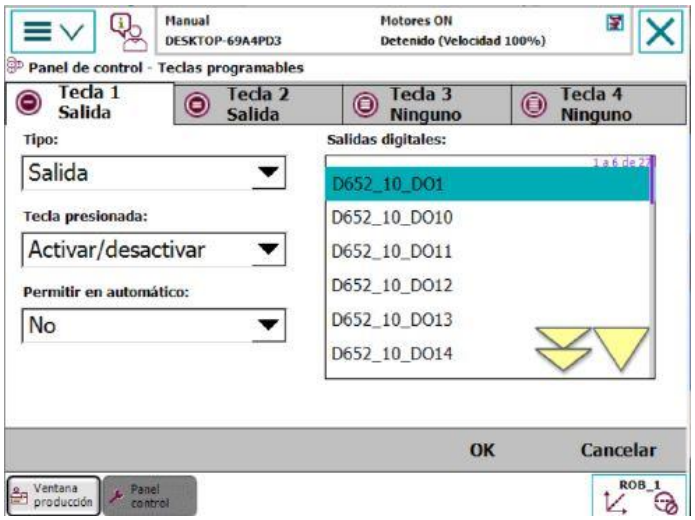


Ilustración 53. Configuración Salida Pinza

Para configurar la salida que controlará la posición de la electroválvula que hace que se abra o se cierre la pinza, debemos seleccionar la salida digital de la tarjeta DSQC652 correspondiente a la herramienta que será D652_10_D01.

3.2.6. Programación

Los robots industriales ABB utilizan un lenguaje de programación único de esta marca, RAPID. Este lenguaje de programación se considera de alto nivel de abstracción.

Por otro lado, encontramos Robotstudio, es el software de programación y simulación de robots ABB. Una de las ventajas que nos ofrecen los robots ABB, gracias a Robotstudio, es la herramienta de simulación, que fuera del entorno industrial, nos da la posibilidad de probar el funcionamiento y comprobar cómo se comportará en robot en el entorno real. De esta manera, se pueden corregir errores antes de su puesta en marcha y evitar problemas que podrían suceder, como roturas en las herramientas.

En nuestro caso hemos utilizado RobotStudio para la escritura del programa en RAPID, mientras que la consola FlexPendant solo se ha utilizado para definir los puntos mediante robtargets.

A continuación, se van a explicar las funciones más importantes de nuestro programa en rapid:

- Robtarget:

  La manera de definir puntos en el espacio tridimensional de nuestro entorno, es a través de la función robtarget.

  Un ejemplo de la estructura en código RAPID de esta función es:

  CONST robtarget home := [[321.03,-4.86,355.09], [0.0108168,-0.349829,-0.93675,-0.00151423], [-1,0,0,0],
  [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

```
Datos de posición
RECORD robtarget
   pos trans;
   orient rot;
   confdata robconf;
   extjoint extax;
ENDRECORD
```

Ilustración 54. Esquema de configuración de robtarget con RobotStudio

  Desde la consola FlexPendant resulta más sencillo programar estos puntos, ya que tan solo ubicando el robot en la posición deseada podemos guardarla y se genera el código en RAPID correspondiente al punto sin tener que definirlo

de forma manual. A continuación, se va a poner un ejemplo de cómo se configura un punto con el FlexPendant.

Lo primero será colocar el robot en la posición que queremos definir, para ello haremos uso del joystick mientras se mantiene pulsado el dispositivo de habilitación. Se debe recordar que el robot consta de 6 articulaciones y con el joystick solo se pueden mover 3 a la vez. Un eje se moverá acorde con el movimiento del joystick arriba y abajo, otro de derecha e izquierda y un tercer eje controlado con el movimiento de rotación del joystick. Para cambiar de ejes se dispone de una tecla que nos permite cambiar entre mover ejes 1-2-3 y ejes 4-5-6 (consultar apartado 3.2.1 para encontrar el botón).

Cuando el robot se encuentre ubicado en la posición a definir, accedemos al apartado de Datos de programa del menú principal de la consola y marcamos robtarget.

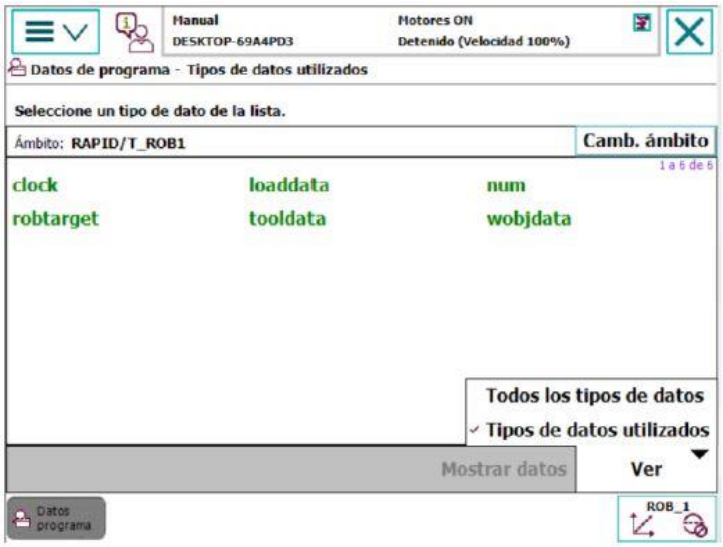


Ilustración 55. Tipos de datos

En el caso de que no nos salga, debemos seleccionar en la opción Ver, que se encuentra abajo a la derecha, y marcar Todos los tipos de datos.

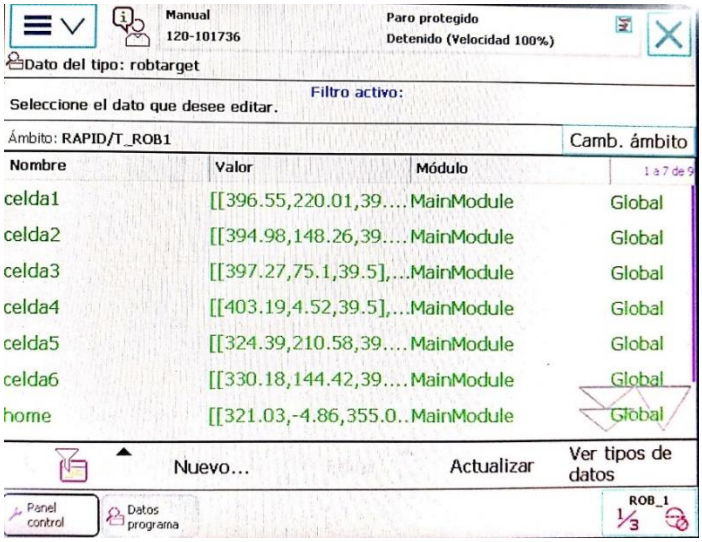


Ilustración 56. Vista de todos los datos (robtarget) definidos

En la ventana de robtarget podremos ver los diferentes puntos que han sido definidos. Seleccionando la opción "Nuevo…" podemos definir una nueva posición a la que le podremos asignar el nombre y el tipo de variable.

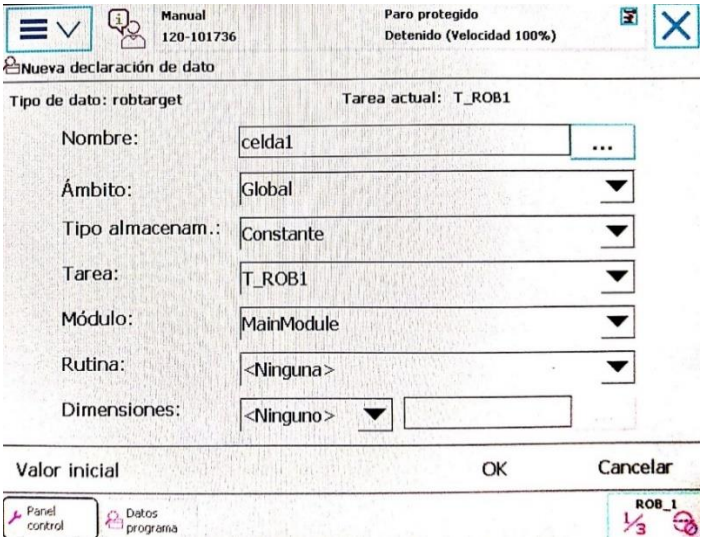


Ilustración 57. Modificación de variables robtarget

Además de asignarle un nombre al punto robtarget, tenemos que seleccionar que tipo de variable de almacenamiento usará. En nuestro caso siempre usaremos almacenamiento de tipo "Constante" para los puntos definidos por medio de robtarget.

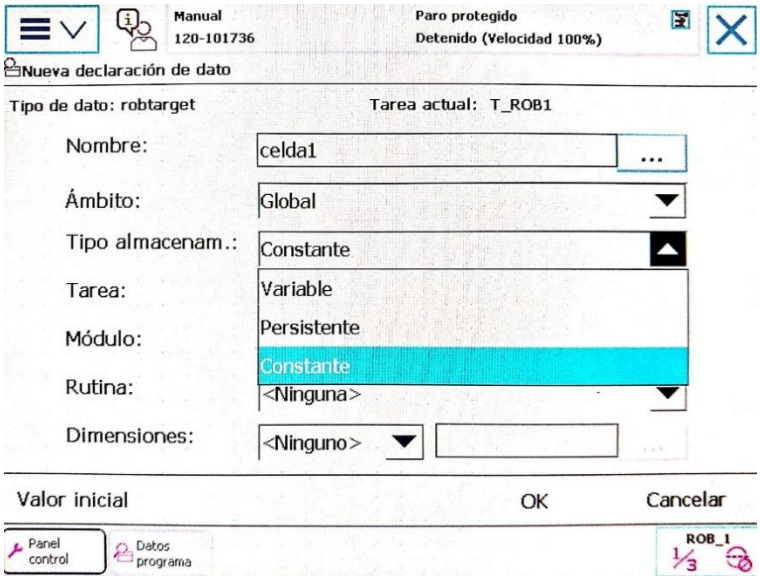


Ilustración 58. Tipo de almacenamiento de la variable robtarget

Para editar cualquier punto que ya haya sido definido nuestra consola nos ofrece la posibilidad tanto de cambiar el nombre como de redefinir el valor de la posición.

Si se desease cambiar el nombre del punto, primero se seleccionaría el punto del que se quiere modificar el nombre y desplegando la pestaña Editar, accedemos a Cambiar declaración. Aparecerá una nueva ventana con distintos apartados de posible configuración. Puesto que el objetivo es modificar el nombre, en el apartado Nombre y presionando sobre el icono con tres puntos suspensivos situado directamente a la derecha, el nombre podrá ser modificado.

Si se considera que el punto no está correctamente definido, se podrá modificar su valor situando previamente el robot en la posición deseada y a continuación, seleccionando el punto que se desea modificar. A continuación, se accederá al desplegable Editar y posteriormente, se presionará en Modificar posición. Inmediatamente la posición y valores de ese punto habrán sido modificados.

- MoveL:

Para definir un movimiento en rapid se pueden utilizar diversas funciones como MoveL, MoveJ o MoveC. Para nuestro problema se ha usado MoveL que define una trayectoria lineal entre la posición en la que se encuentra el robot y la posición final a la que nos dirigimos.

Un ejemplo de esta función es: MoveL celda1, v1000, fine, Pinza;

También se le puede añadir a esta función un offset con respecto al punto definido. Esta posibilidad es muy útil para realizar movimientos de aproximación al objeto. Por ejemplo, en nuestro caso antes de dirigirse a cada objeto se realizaba un movimiento con offset en el eje z dejando un margen de cierta altura para evitar colisionar con otros objetos durante la trayectoria.

Para el caso anterior añadiéndole un offset sería:
MoveL Offs(celda1,0,0,150), v1000, fine, Pinza;

- WaitTime:

WaitTime es una función que como su nombre indica es simplemente esperar un tiempo determinado. Esta función no era imprescindible en nuestro programa, sin embargo, se ha utilizado para observar con mayor claridad las distintas funciones que realizaba nuestro robot. Por ello entre cada línea de código de nuestro programa se ha incluido un waitTime de 0’2 segundos.

- SetDO D652_10_DO1,1 y SetDO D652_10_DO1,0:

Estas funciones corresponden con las acciones de abrir o cerrar la pinza.

El código del programa realizado en Rapid se encuentra en el Anexo 8.4.

4. Protocolo de comunicación

En el presente, se efectúan en un computador muchos procedimientos que precisan de la comunicación con otra computadora para obtener información.

Existen principalmente dos protocolos de comunicación TCP y UDP:

-El protocolo TCP (Transmission Control Protocol) ordena la comunicación de tipo PtP (point to point) entre dos equipos que permite el flujo e intercambio de datos a través de estas dos computadoras. Este protocolo se caracteriza por garantizar que el orden con el que se envían los datos desde un lado de la conexión será el mismo con el que llegarán al otro dispositivo de conexión.

-El protocolo UDP (User Datagram Protocol) se diferencia del TCP en que éste no se considera un protocolo orientado a la conexión. Esto significa que la comunicación mediante este protocolo no puede garantizar la recepción secuencial de los datos. UPD hace la transmisión de los datos sin establecer un conducto de comunicación exclusiva como lo hace TCP.

En el presente proyecto se ha utilizado el protocolo de TCP/IP también conocido como comunicación mediante sockets de flujo por todas las ventajas que este ofrece. Cabe destacar que el término socket de datagrama, por otro lado, hace referencia al protocolo UPD.

4.1. Socket

Ilustración 59. Esquema Protocolo TCP/IP

El término socket hace referencia a un medio de comunicación entre dos procesos, que se ejecutan en una misma computadora o en diferentes computadoras, y que realizan un intercambio de datos fluido.

Los sockets permiten implementar una arquitectura cliente-servidor. Lo que diferencia al cliente del servidor es que el cliente se refiere al programa que inicia la comunicación mientras que el servidor es el que se mantiene a la espera.

Un socket es un procedimiento de comunicación existente entre la máquina cliente y la maquina servidora en la cual, los datos podrán ser intercambiados gracias a las distintas capas de red.

- Dominio de un socket

Una familia o dominio de conexión tiene como misión estructurar los sockets que presentan las mismas características. El dominio de comunicación define el formato de las direcciones y esta forma se consigue distinguir de manera inmediata a los diferentes nodos que soportan la comunicación del socket.

Podemos distinguir dos grandes dominios:

El dominio AF_UNIX (Address Family UNIX): en este caso el cliente y el servidor se sitúan en la misma computadora. Los sockets de dominio AF_UNIX a diferencia de los que veremos a continuación emplean una única dirección como la dirección del archivo local de comunicación.

El dominio de Internet AF_INET (Address Family INET): el cliente y el servidor se encuentran en diferentes equipos de la red de Internet y la comunicación se establece por medio del protocolo TCP/IP. La comunicación de nuestro proyecto es de este tipo, por ello ahondaremos un poco más en este dominio. En este caso para identificar nuestro socket se requiere tanto de una dirección IP como de un puerto serie. Los sockets de internet son la herramienta para la cesión de datos de la tarjeta de red en diferentes hilos de comunicación, que bien pueden ser procedentes de la misma computadora o bien de diferentes computadoras. Por tanto, podemos concluir afirmando que nuestro socket queda establecido con: el protocolo transmisión de datos, la dirección IP tanto local como remota y con los puertos que permiten la comunicación en dichas direcciones.

- Requisitos

Para que se pueda realizar la correcta transmisión de datos entre dos procesos se deben cumplir las siguientes condiciones:

-Que ambos procesos puedan localizarse entre sí.

-Que ambos procesos sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad.

- Medios

Para llevar a cabo la comunicación es imprescindible definir dos recursos de nuestro socket:

-Una dirección IP a cada equipo de la red que reconozca la computadora de origen y la remota.

-Un puerto que identifique los procesos que integran el protocolo de comunicación.

4.1.1. Implementación

A continuación, se va a explicar cómo ha sido implementado en cuanto a la programación nuestro sistema de transmisión de datos entre Python y RAPID.

Las funciones referentes al socket en nuestro programa de Python, es decir, de nuestro cliente son:

- import socket:

Declaramos la librería de sockets que tiene implementada Python para trabajar con las funciones siguientes.

- mi_socket = socket.socket():

Creamos una variable que será nuestro cliente y la nombramos como mi_socket.

- mi_socket.connect(("192.168.125.1" , 8000)):

Declaramos la dirección de IP y el puerto con el que se va a conectar nuestro cliente al servidor.

- mi_socket.send(str(x)+str(y)):

Enviamos dos variables de tipo string a nuestro servidor, que serán el eje x y el eje y del centroide que detectamos.

Las funciones referentes al socket en nuestro programa de Rapid, es decir, de nuestro servidor son:

- VAR socketdev serverSocket:

Variable utilizada para crear el servidor del socket.

- VAR socketdev clientSocket:

Variable utilizada para crear el cliente del socket.

- VAR bool okX y VAR bool okY:

Variables de tipo booleano que se usarán para transformar los datos que llegan como string a variables de tipo numérico.

- VAR string data:

Cadena de caracteres que se usará para almacenar los caracteres que llegan de nuestro cliente.

- VAR num cambioStrX y VAR num cambioStrY:

Variables de tipo numérico ya transformadas, que son las que se usan para indicar la posición de los objetos.

- VAR string XValorString y VAR string YValorString:

Se utilizan para guardar las partes de la cadena de caracteres que interesan.

- SocketCreate serverSocket:

Crea el servidor del socket.

- SocketBind serverSocket, "192.168.125.1", 8000:

Seleccionamos la dirección de IP y el puerto de nuestro servidor.

- SocketListen serverSocket:

Ponemos el servidor en modo escucha.

- SocketAccept serverSocket,clientSocket,\Time:=WAIT_MAX:

Espera un tiempo indefinido hasta que el cliente se conecte al servidor.

- SocketSend clientSocket \str:="Conexion con abb correcta":

El servidor envía el mensaje de conexión correcta al cliente.

- SocketReceive clientSocket \str:=data:

Recibe la cadena de caracteres que le envia el cliente y la almacena en la variable de tipo string data.

- XValorString := StrPart(data,1,3):

Coge los 3 primeros caracteres de la variable data, es decir, de los 6 caracteres que le envía el cliente, coge los que se refieren al eje x y los guarda en una variable de tipo string.

- okX:=StrToVal(XValorString,cambioStrX):

Transforma la coordenada en x del centroide a una variable de tipo numérico para poder trabajar con ella.

- YValorString := StrPart(data,4,3):

Coge los 3 últimos caracteres de la variable data, es decir, de los 6 caracteres que le envía el cliente, coge los que se refieren al eje y y los guarda en una variable de tipo string.

- okY:=StrToVal(YValorString,cambioStrY):

Transforma la coordenada en y del centroide a una variable de tipo numérico para poder trabajar con ella.

5. Integración de datos

5.1. Consideraciones

5.1.1. Perspectiva de la cámara y centroide

A la hora de enviar los datos de nuestros objetos al robot hay que tener varias consideraciones en cuenta.

1. El centroide de la imagen no corresponde con el centro de masas de nuestro objeto en el espacio tridimensional:

   Lo primero es conocer la forma de los objetos con los que vamos a trabajar, en nuestro caso se trata de cilindros. Gracias a la geometría de nuestro objeto con solo una vista de nuestro objeto podemos hallar una referencia muy válida para conocer el centroide del objeto ya sea con vista en planta, alzado o perfil.

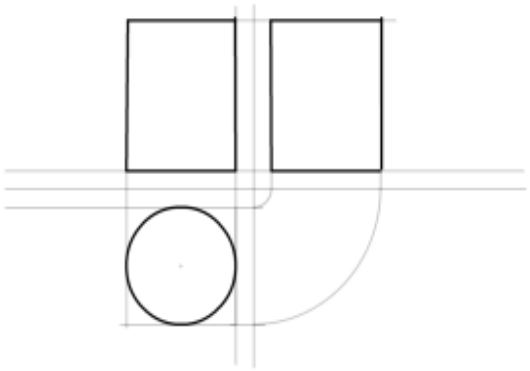


Ilustración 60. Vistas de un Cilindro

   Las vistas alzado y perfil nos darán un rectángulo cuyo centro coincidirá en dos ejes con el del volumen. Sin embargo, no podremos saber la profundidad del objeto, por ello estas vistas no nos servirían para determinar el centroide del objeto con una sola imagen.

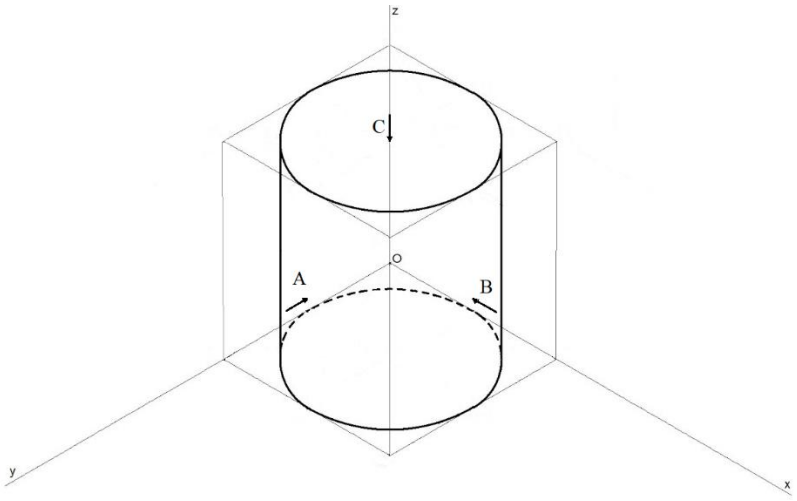


Ilustración 61. Dibujo Cilindro ejes

El centro del rectángulo que podemos ver desde la vista del alzado (A) nos daría las coordenadas del centro de masas de nuestro cuerpo en el eje X y en el eje Z.

El centro del rectángulo que podemos ver desde la vista del perfil (B) nos daría las coordenadas del centro de masas de nuestro cuerpo en el eje Y y en el eje Z.

Mientras que el centro del círculo que podemos ver desde la vista de planta (C) nos daría las coordenadas del centro de masas de nuestro cuerpo en el eje X y en el eje Y.

Conociendo los ejes del círculo que forma el cilindro visto desde planta, ya podríamos obtener el centroide de nuestro cuerpo a pesar de faltarnos un eje. Esto es así porque sabemos que el robot está colocado sobre la mesa y el eje Z del robot coincide con la altura del cilindro, entonces por geometría podemos saber que la mitad de la altura del cilindro será la posición en el eje Z de nuestro centroide.

En cambio, por limitaciones de nuestro entorno, no hemos podido situar la cámara en una posición en la cual se capte la imagen vista desde planta. Aun así, se ha intentado que nuestra perspectiva se le asemeje al máximo y por ello se ha buscado que la cámara se coloque lo más alto posible.

A continuación, se ha realizado un dibujo sobre la imagen creada por nuestro sistema de visión artificial. Se ha marcado en negro el centro aproximado del círculo de la cara superior de nuestro cilindro.

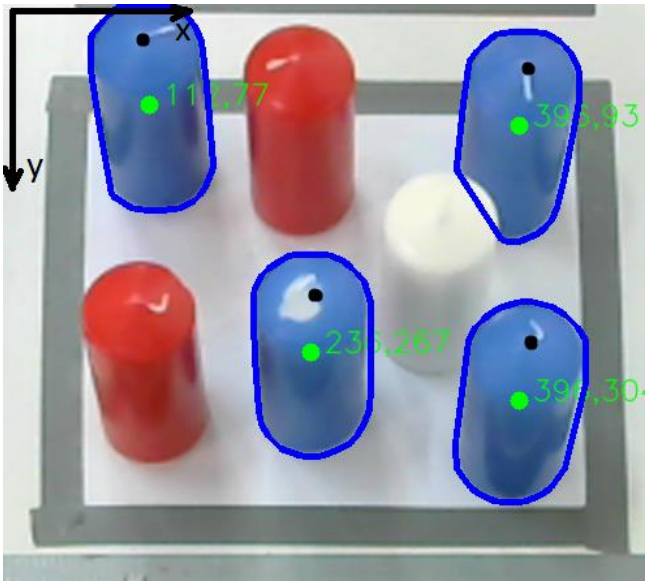


Ilustración 62. Comparación centro buscado con centro obtenido

Como vemos la posición del centro que nos interesa no difiere demasiado con el centro que nos da nuestro sistema de visión. Aplicando un ajuste de perspectiva constante podríamos alcanzar la posición que buscamos y con solo restarle 40 en el "eje y", ya tendríamos una posición más aproximada.

2. El sistema de coordenadas de nuestra imagen no es el mismo que el del robot:

   Como es lógico, los ejes de coordenadas del robot y de la imagen no son coincidentes, pero se ha buscado que si sean paralelos. Aunque no es estrictamente necesario, el hecho de que los ejes sean paralelos facilita mucho los cálculos, ya que evita tener que realizar operaciones matemáticas con matrices de rotación.

   Para hacer coincidentes los ejes en nuestro caso bastará con sumarle o restarle una constante en cada eje.

3. La variación de posición no mantiene la misma magnitud en los dos sistemas de coordenadas:

   Aunque el problema hasta ahora parecía sencillo de resolver aparece un factor importante que condiciona nuestra resolución.

Si nos paramos a pensar, como la vista de la imagen no es en planta, las distancias no van a ser las mismas, es decir, lo que significan 20mm en el eje x de la imagen no van a ser los mismos 20 mm del espacio tridimensional real que es el eje del robot.

Esto nos lleva a que debemos realizar una operación que tenga en cuenta que la variación en los dos ejes no es proporcional.

5.2. Interpolación

Para resolver nuestro problema de perspectiva que provoca que las variaciones en los ejes de la imagen no sean proporcionales a los del entorno, vamos a aplicar una interpolación lineal. Como sabemos este modelo matemático no nos da una solución exacta. Sin embargo, como nuestra área de detección no es demasiado grande, al aplicar esta solución sobre nuestro problema, la aproximación que nos da será muy buena para conseguir la posición final de los objetos que buscamos.

Tendremos que valorar los dos extremos de nuestra posición en la imagen, es decir, el valor mínimo y máximo que pueden tener los objetos.

Como hemos visto en la ilustración 3 del apartado anterior, los ejes de la imagen se encuentran en la esquina superior izquierda. Por ello, con estudiar las siguientes posiciones: esquina superior izquierda y esquina inferior derecha, será suficiente para realizar nuestra operación.

-Robot: Pos inicial [255’27, -128’93]

-Imagen: (Xo,Yo) = (198, 190)

A = 0

B = 0

-Robot: Pos final [425’52, -270’19]

-Imagen: (Xf,Yf) = (513, 427)

A= 170’25

B= -141’26

Solución eje X: (170.25 * (198 – cambioStrX) / (198 - 513))

Solución eje Y: (-141.26 * (190 – cambioStrY) / (190 - 427))

5.3.Problemática con ciertos puntos del plano

Como se ha visto en capítulos anteriores los mensajes que se envían mediante sockets son simplemente cadenas de caracteres.

Nuestro robot recibe seis caracteres, de forma que los tres primeros corresponden al eje x del objeto detectado y los tres siguientes al eje y.

Una vez almacenados estos seis caracteres en una variable, se guardan en dos partes según la posición de caracteres, así dividimos el punto del centroide en dos variables una que corresponde a la posición en el eje x y otra en el eje y.

Por último, se realiza cada variable a otra de tipo numérico para poder operar.

Este método de resolución generaba un problema y es que no siempre los puntos del centroide detectado disponían de tres dígitos por eje. Es decir, los puntos entre el 0 y el 99 eran un problema. Por ejemplo:

Si se detectaba un centroide cuyas coordenadas eran (86,204). Nuestro programa transformaba esto como x=862 e y=04.

La fórmula que se ha utilizado para solucionar esta problemática es sencillamente sumar 100 a cada variable antes del envío desde el programa de Python. De esta manera, nos aseguramos de que el valor mínimo siempre tendrá tres dígitos. Posteriormente desde Rapid se deshace la operación, restando 100 a cada variable.

6. Conclusiones

6.1. Conclusiones

Una vez finalizado el proyecto podemos afirmar que se han cumplido los objetivos que se propusieron en un comienzo, logrando realizar diversas tareas en las que cooperan el sistema de visión artificial y el robot IRB120. Se ha conseguido desarrollar un sistema robótico automatizado que a través de un sistema de visión artificial es capaz de seleccionar objetos de un determinado color y tamaño, y trasladarlos mediante una pinza a una nueva ubicación, ignorando cualquier acción posible sobre objetos que no sean los adecuados y no cumplan con las especificaciones programadas.

Aunque el objetivo del proyecto era de carácter didáctico, ya que se ha realizado un estudio completo en la materia, se podría aplicar en cualquier entorno industrial real.

6.2. Problemas durante el desarrollo

Si bien el desarrollo de este proyecto ha sido una experiencia entusiasta, no ha estado exenta de dificultades que han surgido en el transcurso del mismo.

La llegada de la pandemia provocó un inevitable parón en la ejecución del proyecto ya que no se pudo asistir al laboratorio durante varios meses. Además, esto provocó una pérdida de ritmo y de conexión con el trabajo, que hizo que retomarlo fuese casi una vuelta a empezar.

En cuanto a problemas técnicos:

Durante varias semanas se estuvo intentando poner en marcha la pinza del robot que ni se abría ni se cerraba. Se pensó que el problema sería de configuración o de programación, pero finalmente resultó ser un problema de cableado y se demoró mucho tiempo hasta detectar dicho fallo.

Por otro lado, cuando se estaban realizando algunas pruebas con el robot, se produjo una avería en la pinza partiéndose una de sus placas extensoras al colisionar con un objeto. Se pudo reparar rápidamente y continuar con el proyecto.

La última dificultad notable que se presentó en el proyecto fue en lo referente a la transmisión de datos entre el sistema de visión y el robot, dicha problemática se encuentra desarrollada en el apartado 5.3.

6.3. Trabajos futuros

Para futuros trabajos se pueden realizar diversas mejoras de diseño en el entorno que nos permitan alcanzar nuevos objetivos:

-Diseñar un soporte para cámaras fácil de acoplar al robot o a la pared. Esto se podría realizar con una impresora 3D.

-Implementar en el robot un sistema anticolisiones para evitar las averías que se pueden producir en nuestra herramienta al chocar con un elemento externo de nuestro entorno como las piezas a manipular. Una posibilidad sería integrar sensores de proximidad en nuestro robot.

-Diseñar una herramienta doble, de forma que una esté operativa y la otra esté plegada y mediante un mecanismo o un servomotor se pudiera cambiar de herramienta. Esto podría dar lugar a un proyecto en el que según el tamaño y la forma de nuestras piezas unas se recogerán con la pinza y otras con una ventosa. Nuestro sistema de visión enviaría al robot tanto la posición del objeto como la herramienta a utilizar y el robot procedería a realizar la operación de pick & place.

-Añadir una segunda cámara para que la detección de los objetos se pueda obtener de una forma más precisa al incorporar imágenes desde varios ángulos.

-Incorporar un segundo brazo robótico para que se puedan realizar tareas más complejas en las que los robots interactúen de forma coordinada.

Otros proyectos que se podrían realizar con el material que disponemos en el laboratorio serían:

-Incorporar a nuestro entorno una cinta transportadora controlada mediante un PLC. El objetivo sería coordinar los tres sistemas: visón artificial, PLC y robot:

- Un ejemplo de proyecto podría ser el de simular un proceso de etiquetado de un producto, en el que el robot recoge los productos sin etiquetar, los coloca en la cinta (la cinta simulará un proceso externo de etiquetado del producto) y una vez acabado el movimiento de la cinta el robot recogerá el objeto etiquetado para situarlo en una nueva ubicación. Además, se podría añadir un sistema de HMI con el que por ejemplo se pueda variar la velocidad de la cinta.

- Otra idea de trabajo que se podría realizar en relación al anterior, sería realizar la operación de pick & place con distintos objetos en movimiento. Sobre la cinta transportadora se colocarían unos objetos y nuestro sistema integrado tendrá que ser capaz de dar la posición del objeto en tiempo real, de esta forma el robot podrá realizar la operación con las piezas en movimiento.

7. Bibliografía

[1] Página web ABB - https://new.abb.com/es

[2] Corke, P. (2011). *Robotics, Vision & Control.* Springer.

[3] Alegre, E., Pajares, G., & de la Escalera, A. (2016). *Conceptos y Métodos en Visión por Computador.* España: Grupo de Visión del Comité Español de Automática (CEA).

[4] Sucar, L. E., & Gómez, G. (2011). *Visión Computacional.* México.

[5] Página web OpenCV- https://docs.opencv.org/3.4/index.html

[6] Manual de Usuario del Robot IRB120 –

https://new.abb.com/products/robotics/es/robots-industriales/irb-120

[7] Fu, K. S., Lee, C. G., & González, R. (1987). *Robótica: Control, detección, visión e inteligencia.* McGraw-Hill Education.

[8] Barrientos, A., Peñín, L. F., Balaguer, C., & Aracil, R. (1997). *Fundamentos de Robótica.* McGraw-Hill.

[9] Martínez Nicolás, S. (2019): *Automatización de una planta robotizada para operaciones de almacenaje*, Trabajo Fin de Grado, Universidad Politécnica de Cartagena (UPCT).

[11] Senén Estremera, A. (2015): *Diseño de una estación de trabajo para el Robot IRB120. Control de cinta transportadora mediante IRC*, Trabajo Fin de Grado, Universidad de Alcalá (UAH)

[12] Díaz García, L. (2018): *Integración de visión artificial en un robot industrial*, Trabajo Fin de Grado, Universidad Pontificia Comillas (ICAI)

[13] Valera Fernández, Á., Gómez Moreno, J., Sánchez Salmerón…, (2012): *Industrial robot programming and upnp services orchestration for the automation of factories*.

[14] Shanmugasundar, G., Sivaramakrishnan R. (2012). *Software Development for an Inverse Kinematics of Seven-Degrees of Freedom Newly Designed Articulated Inspection Robot*

8. Anexos

8.1. Hoja de características del robot ABB IRB 120

ROBOTICS

IRB 120

ABB's 6 axis robot – for flexible and compact production

The IRB 120 robot is the latest addition to ABB's new fourth-generation of robotic technology. It is ideal for material handling and assembly applications and provides an agile, compact and lightweight solution with superior control and path accuracy.

Compact and lightweight
IRB 120's compact design enables it to be mounted virtually anywhere at any angle without any restriction - for example inside a cell, on top of a machine or close to other robots.

IRB 120 is also the most portable and easy to integrate on the market with its 25 kg weight. The smooth surfaces are easy to clean and the cables for air and customer signals are internally routed, all the way from the foot to the wrist, ensuring that integration is effortless.

Multipurpose
IRB 120 is ideal for a wide range of industries including the electronic, food and beverage, machinery, solar, pharmaceutical, medical and research sectors.

The Food Grade Lubrication (NSF H1) option includes Clean Room ISO Class 5, which ensures uncompromising safety and hygiene for food and beverage applications.

Optimized working range
IRB 120 has a horizontal reach of 580 mm, the best in class stroke, the ability to reach 112 mm below its base and a very compact turning radius.

Fast, accurate and agile
Designed with a light, aluminum structure, the motors ensure the robot is enabled with a fast acceleration, and can deliver accuracy and agility in any application.

IRC5 Compact controller – optimized for small robots
ABB's new IRC5 Compact controller presents the capabilities of the IRC5 controller in a compact format. It brings accuracy and motion control to applications which have been exclusive to large installations and enables easy commissioning through one phase power input, external connectors for all signals and a built-in expandable 16 in, 16 out, I/O system.

RobotStudio for offline programming enables manufacturers to simulate a production cell to find the optimal position for the robot, and provide offline programming to prevent costly downtime and delays to production.

Reduced footprint
The combination of the new lightweight architecture of the IRB 120 with the new IRC5 Compact controller introduces a significantly reduced footprint.

Specification

Robot version	Reach (m)	Handling capacity (kg)	Armload (kg)
IRB 120-3/0.6	0.58	3*	0.30
Number of axes	6		
Protection	IP30		
Mounting	Any angle		
Controller	IRC5 Compact/IRC5 Single Cabinet		
Integrated signal supply	10 signals on wrist		
Integrated air supply	4 air on wrist (5 bar)		

* 4 with vertical wrist

Performance (according to ISO 9283)

	IRB 120
1 kg picking cycle	
25 x 300 x 25 mm	0.58 s
25 x 300 x 25 with 180° axis 6 reorientation	0.92 s
Acceleration time 0-1 m/s	0.07 s
Position repeatability	0.01 mm

Technical information

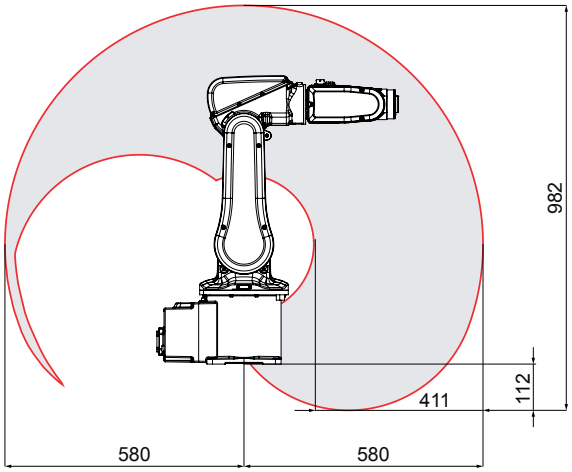
Electrical Connections	
Supply voltage	200-600 V, 50/60 Hz
Rated power transformer rating	3.0 kVA
Power consumption	0.24 kW
Physical	
Robot base	180 x 180 mm
Robot height	700 mm
Robot weight	25 kg
Environment	
Ambient temperature for robot manipulator:	
During operation	+5°C (41°F) to +45°C (113°F)
During transportation and storage	-25°C (-13°F) to +55°C (131°F)
During short periods (max. 24 h)	up to +70°C (158°F)
Relative humidity	Max. 95%
Noise level	Max. 70 dB (A)
Safety	Safety and emergency stops 2-channel safety circuits supervision, 3-position enabling device
Emission	EMC/EMI-shielded
Options	Clean Room ISO class 5 (certified by IPA)**

** ISO class 4 can be reached under certain conditions.
Data and dimensions may be changed without notice.

Movement

Axis movement	Working range	Velocity IRB 120
Axis 1 rotation	+165° to -165°	250°/s
Axis 2 arm	+110° to -110°	250°/s
Axis 3 arm	+70° to -110°	250°/s
Axis 4 wrist	+160° to -160°	320°/s
Axis 5 bend	+120° to -120°	320°/s
Axis 6 turn	Default: +400° to -400° Max. rev: +242 to -242	420°/s

Working range



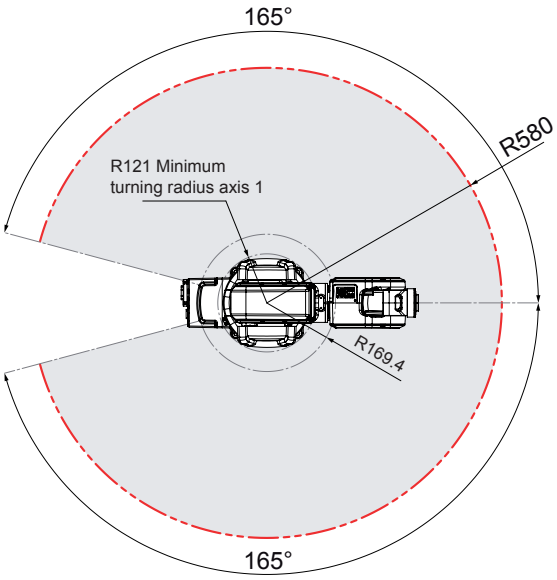


abb.com/robotics

ROBO149EN_D Rev. J November 2019

8.2.Hoja de características controladora IRC5C Compact

ROBOTICS

IRC5

Industrial Robot Controller

Based on more than four decades of robotics experience, the IRC5 is the robotic industry's benchmark in robot controller technology. In addition to ABB's unique motion control it brings flexibility, safety, modularity, application interfaces, multi-robot control and PC tool support. The IRC5 comes in different variants to provide a cost-effective and optimized solutions for every need.

Fast and accurate
The IRC5 gives our robots the ability to perform their tasks in a highly efficient manner. Based on advanced dynamic modelling, the IRC5 automatically optimizes the performance of the robot by reducing cycle times (QuickMove®) and providing precise path accuracy (TrueMove®). Thanks to ABB's IRC5 technology, a robot's motion is predictable and its performance high, with no tuning required by the programmer. What you program is what you get.

Safe
Operator safety is a leading benefit of the IRC5. It fulfills all relevant regulations and is certified by third party inspectors worldwide.

SafeMove2 represent a new generation of safety, enabling more flexible robotic cell safety concepts, e.g. enabling floor space reduction and collaboration between robot and humans.

Compatible
No matter where in the world your robot is located, and regardless of what regulatory standards apply, the IRC5 is up to the task. ABB's controller is compatible with various types of main voltages and can handle a broad spectrum of environmental conditions. IRC5 communicates with other machines in a manufacturing environment; in a safe and predictable way. It supports the majority of all state-of-the-art industrial networks for I/O. Sensor interfaces, remote access and a rich set of programmable interfaces are examples of the IRC5's many powerful networking features.

Programmable
All ABB robot systems are programmed with RAPID™, ABB's flexible, high-level programming language. On the surface RAPID's basic features and functionality are easy to use, but dig deeper and you will find that this programming language allows you to create highly sophisticated solutions. It is a truly universal language on and off the shop floor which supports structured programs, and advanced features. It also incorporates powerful support for the most common robot process applications such as welding and assembly.

Reliable
The IRC5 is practically maintenance free, and its outstanding quality ensures unmatched up-time. Built-in diagnostic functions help ensure fast recovery and production restarts when operations are interrupted on the factory floor.

The IRC5 also comes equipped with remote monitoring technology, ABB Ability Connected Services. Advanced diagnostics allow quick investigation of failures as well as real-time monitoring of the robot condition throughout its lifecycle; all made to increase your productivity.

IRC5 Single Cabinet Controller

- Designed for high IP protection and full expandability.
- Provides a protected environment for axillary equipment in the robot system.
- Capable of control of up to four robots in a MultiMove® setup. Just add a compact drive module to each additional robot.
- MultiMove® opens up previously unthinkable operations, thanks to the perfect coordination of complex motion patterns.

—

Technical information

Electrical Connections	
Supply voltage	3 phase, 200-600 V, 50-60 Hz
Physical	
Dimensions	
Single cabinet	970 x 725 x 710 mm
MultiMove® drive modules	720 x 725 x 710 mm
Weight	
Single cabinet	150 kg
MultiMove® drive modules	130 kg
Environment	
Ambient temperature	0-45°C, optional: 0-52°C
Relative humidity	Max. 95% non condensing
Safety	SafeMove: Supervision of stand-still, speed, position and orientation (robot and additional axes): 8 safe inputs for function activation, 8 safe monitoring outputs
Extended safety	Electronic Position Switches: 5 safe outputs monitoring axis 1-7
Degree of protection	IP54 (cooling ducts IP33)

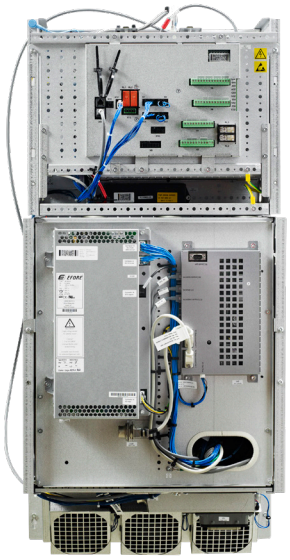
IRC5C Compact Controller

- Offers the capabilities of the powerful IRC5 controller in a compact format.
- Delivers space saving benefits and easy commissioning through one phase power input
- External connectors for all signals and a built in expandable 16 in, 16 out I/O system.

—

Technical information

Electrical Connections	
Supply voltage	Single phase 220/230 V, 50-60 Hz
Physical	
Dimensions	320 x 449 x 442 mm
Weight	28.5 kg
Environment	
Ambient temperature	0-45°C
Relative humidity	Max. 95% non condensing
Degree of protection	IP20



IRC5 PMC Panel Mounted Controller

- Comes without a cabinet
- Can be integrated into any enclosure for customization or for special environmental requirements

—

Technical information

Electrical Connections	
Supply voltage	3 phase 200-600 V,50-60 Hz
Physical	
Dimensions	
Control module	375 x 498 x 271 mm
Drive module small	375 x 498 x 299 mm
Drive module large	685 x 498 x 425 mm
Weight	
Control module	12 kg
Drive module small	24 kg
Drive module large	40 kg
Environment	
Ambient temperature	0-45°C
Relative humidity	Max. 95% non condensing
Safety	SafeMove. Supervision of stand-still, speed, position and orientation (robot and additional axes) 8 safe inputs for function activation, 8 safe monitoring outputs
Extended safety	Electronic Position Switches: 5 safe outputs monitoring axis 1-7
Degree of protection	IP20

IRC5P Paint Controller

- Provides full control of the paint process by integrating hardware and software seamlessly
- Reduces cycle time, saves paint, and is environmentally-friendly.
- Is certified as an associated equipment for interfacing manipulators/equipment for hazardous location.
- The IRC5P FlexPaint Pendant is Ex certified for use in hazardous locations; includes soft keys, a dual joystick, a 3.5 inch back lit screen; supports Asian language characters and has an emergency stop.
- Includes RobView 5, which automatically adapts the robot system to the specific paint application.

—

Technical information

Electrical Connections	
Supply voltage	3 phase 200-600 V, 50-60 Hz
Physical	
Dimensions	1450 x 725 x 710 mm
Weight	180 kg
Environment	
Ex classification:	II (2) G [Ex ib px] IIB T4 II (2) D [Ex pD 21] T65°C FM Class I,II. Div.1, Group C, D, E, F, G 135°C
Ambient temperature	Max. 95% non condensing
Relative humidity	0-48°C
Degree of protection	IP54

RobotStudio Online
RobotStudio Online is a suite of tablet applications for shop floor operations. Utilizing the familiar and user-friendly nature of tablets, these applications make it easy to perform operations such as calibration, editing programs or jogging. Combined with an ABB Jokab Safety three position enabling device safety is not compromised.

FlexPendant
The FlexPendant is characterized by its clean, color touch screen-based design and 3D joystick for intuitive interaction. Powerful support for tailor-made applications, e.g. operator screens.

Technical information, FlexPendant

Environment	
Functions	Graphical color touch screen Joystick Hot plug- Add/remove during operation Membrane keyboard with 12 buttons USB Memory support
Safety functions	Emergency stop 3-position enabling switch (dual circuit)
Degree of protection	IP54

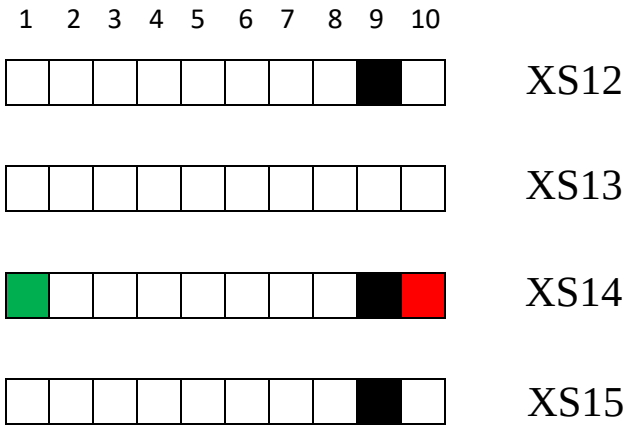
Technical information, JSHD4-3 Three position device

Environment	
Functions	LED status diods
Safety functions	Emergency stop 3-position enabling switch (dual circuit)
Degree of protection	IP65

abb.com/robotics

ROB0295EN Rev. D June 2019

8.3. Esquema eléctrico de la tarjeta interna DSQC652

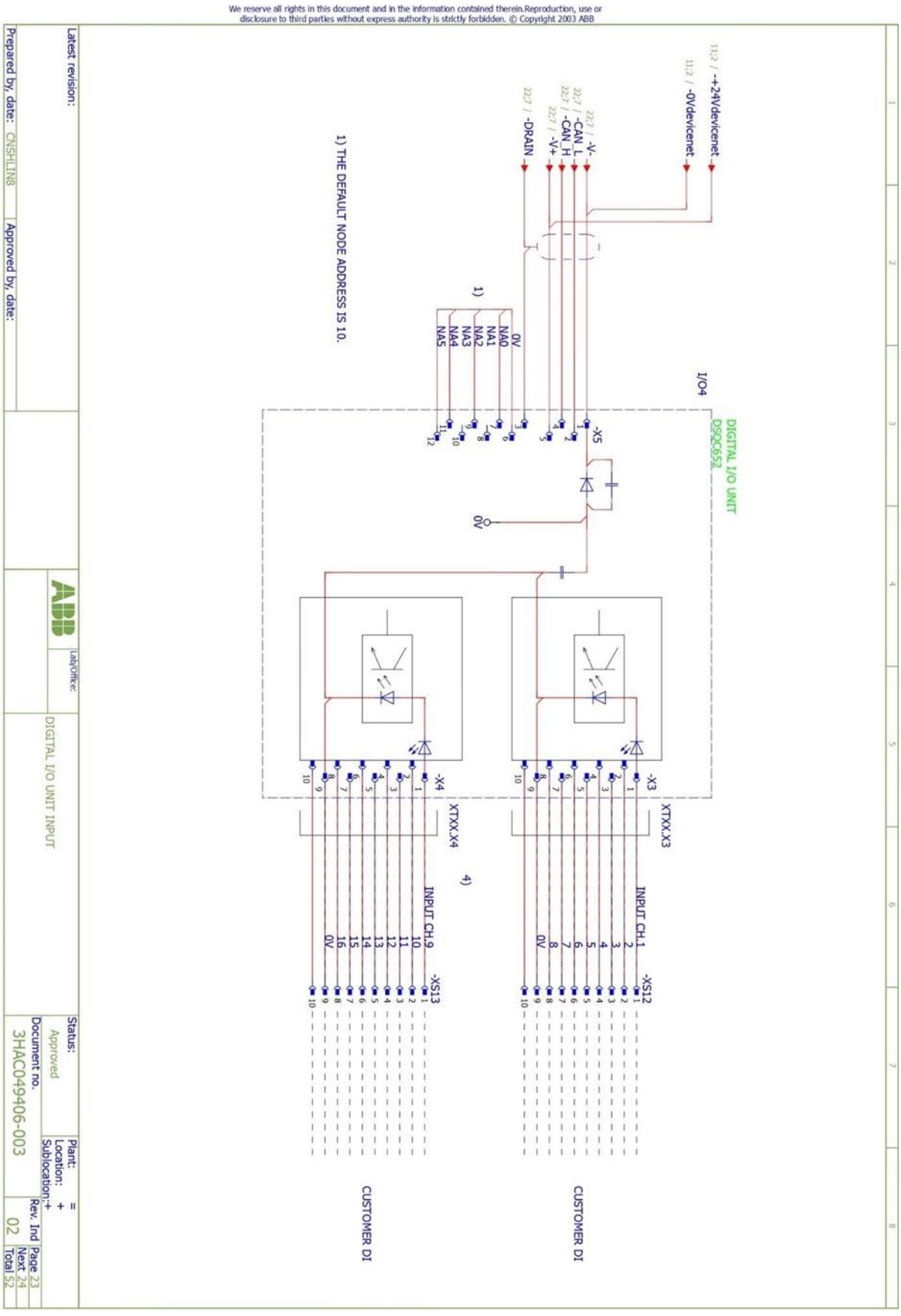


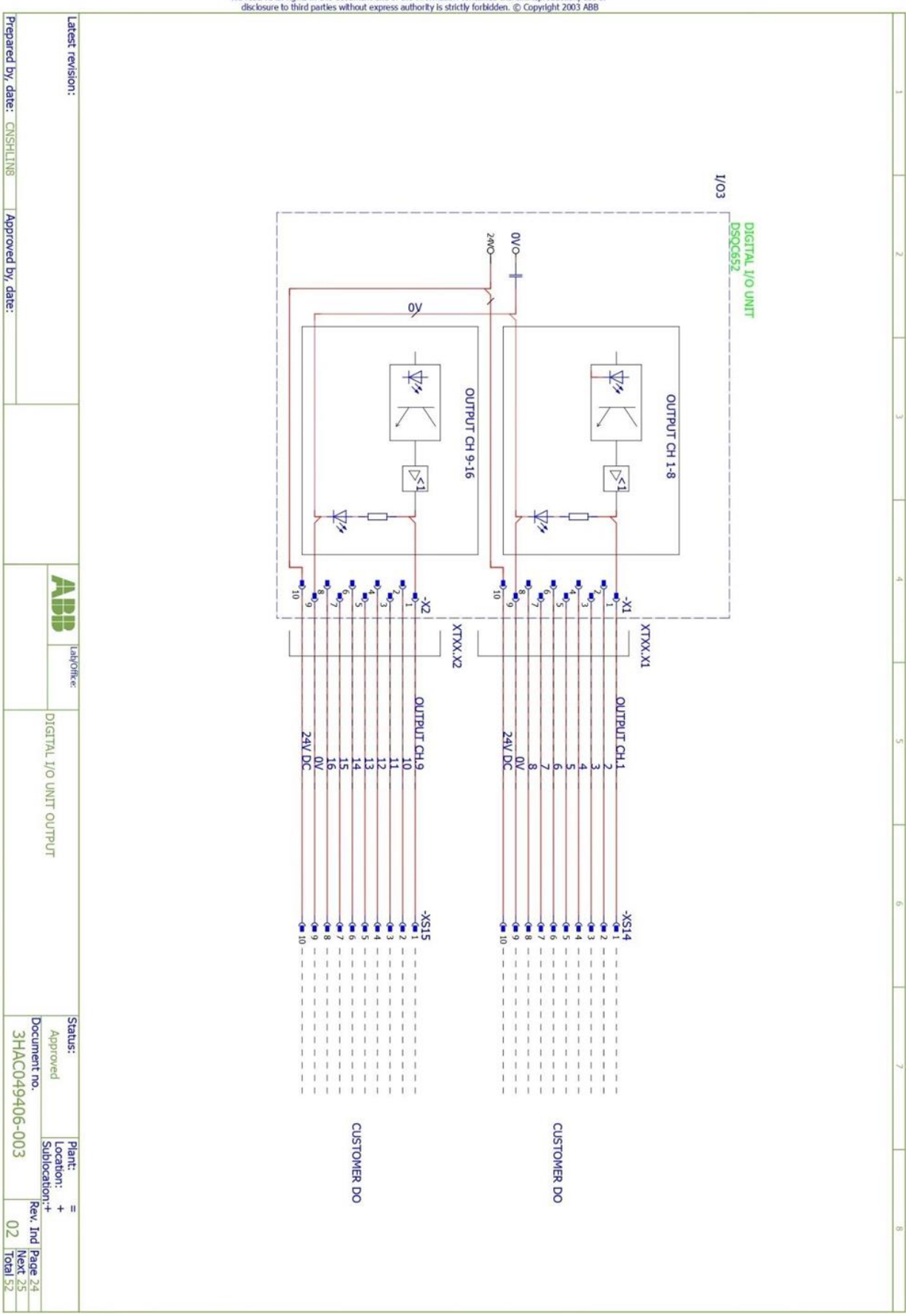
Salida electroválvula

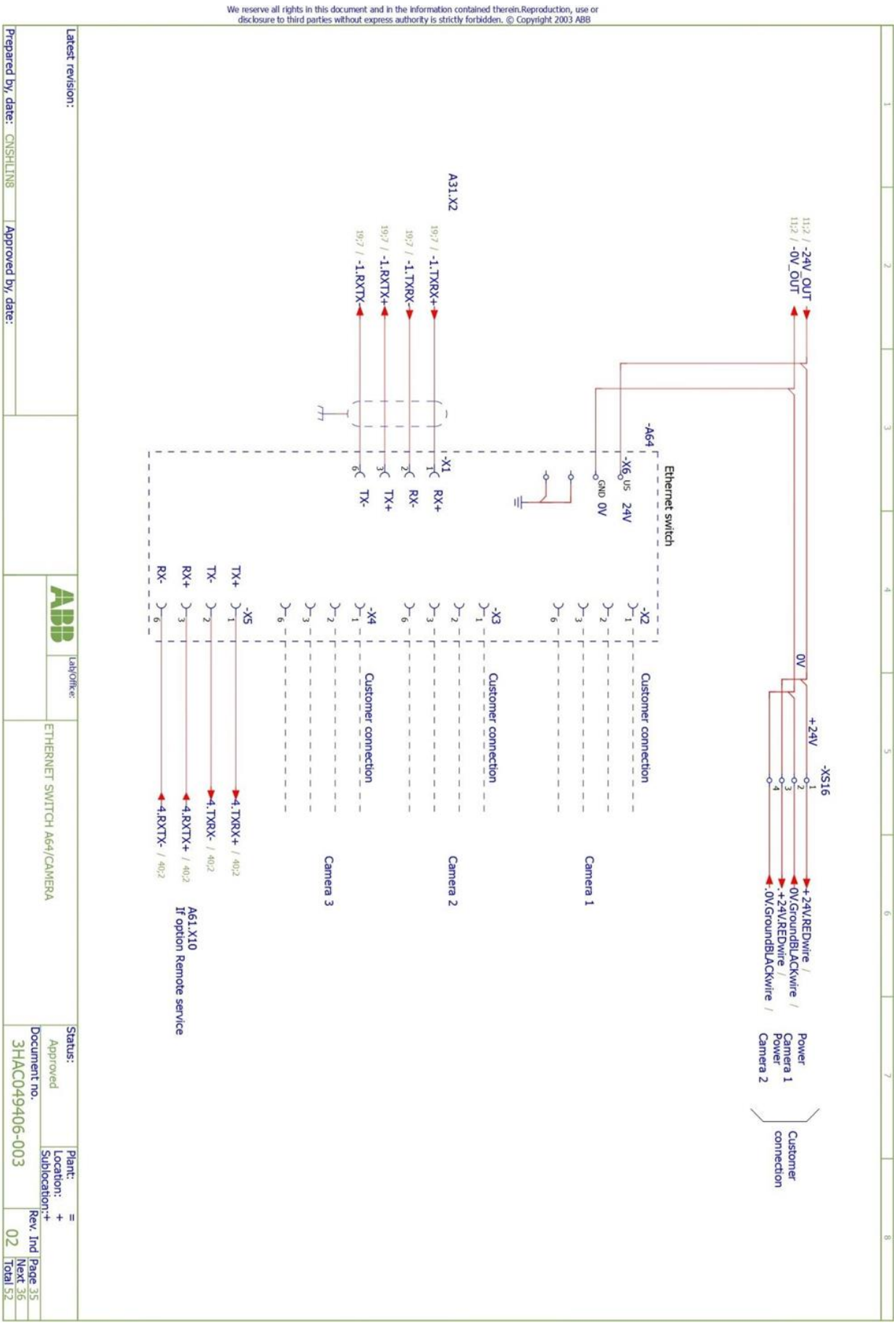
Toma de tierra 0v

Alimentación desde fuente de tensión externa de 24 V

Esquema eléctrico bornero


We reserve all rights in this document and in the information contained therein.Reproduction, use or disclosure to third parties without express authority is strictly forbidden. © Copyright 2003 ABB
11;2 / -+24Vdevicenet
11;2 / -0Vdevicenet
22;7 / -DRAIN
22;7 / -V+
22;7 / -CAN_H
22;7 / -CAN_L
22;7 / -V-
1) THE DEFAULT NODE ADDRESS IS 10.
1)
0V
NA0
NA1
NA2
NA3
NA4
NA5
I/O4
DIGITAL I/O UNIT
DSQC652
-X5
-X3
-X4
XTXX.X3
XTXX.X4
4)
INPUT CH.1
INPUT CH.9
-XS12
-XS13
CUSTOMER DI
CUSTOMER DI
Latest revision:
Prepared by, date: CNSHLIN8
Approved by, date:
LabOffice
DIGITAL I/O UNIT INPUT
Status: Approved
Document no. 3HAC049406-003
Rev. Ind 02
Plant:
Location:
Sublocation:
Page 23
Next 24
Total 52




We reserve all rights in this document and in the information contained therein.Reproduction, use or disclosure to third parties without express authority is strictly forbidden. © Copyright 2003 ABB
Latest revision:
Prepared by, date: CNSHLIN8
Approved by, date:
ABB
LabVIEW
ETHERNET SWITCH A64/CAMERA
Status: Approved
Document no. 3HAC049406-003
Plant:
Location: +
Sublocation:+
Rev. Ind 02
Page 35
Next 36
Total 52
-24V_OUT
-0V_OUT
-XS16
+24V
0V
+24V.REDwire
0V.GroundBLACKwire
+24V.REDwire
0V.GroundBLACKwire
Power Camera 1
Power Camera 2
Customer connection
Ethernet switch
-A64
-X6
US 24V
GND 0V
-X1
RX+
RX-
TX+
TX-
-X2
-X3
-X4
-X5
Customer connection
Camera 1
Camera 2
Camera 3
A31.X2
-1.TXRX+
-1.TXRX-
-1.RXTX+
-1.RXTX-
4.TXRX+
4.TXRX-
4.RXTX+
4.RXTX-
A61.X10
If option Remote service

8.4. Código en Rapid

```
MODULE MainModule

CONST robtarget home:=[[321.03,-4.86,355.09],[0.0108168,-0.349829,-0.93675,-
0.00151423],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

TASK PERS tooldata
Pinza:=[TRUE,[[0,0,185],[1,0,0,0]],[1,[0,0,1],[1,0,0,0],0,0,0]];

VAR socketdev serverSocket;

VAR socketdev clientSocket;

VAR bool okX;

VAR bool okY;

VAR string data;

VAR num cambioStrX;

VAR num cambioStrY;

VAR string XValorString;

VAR string YValorString;

CONST robtarget pos_inicial:=[[255.27,-
128.93,37.64],[0.00600793,0.342599,0.939461,-0.00183555],[-1,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget pos_final:=[[425.52,-270.19,39.50],[0.016342,-0.364199,-
0.930983,0.0190323],[-1,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget celda1:=[[396.55,220.01,39.50],[0.0162963,-0.340619,-
0.93872,0.0501892],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget celda2:=[[394.98,148.26,39.50],[0.0229242,-0.369552,-
0.927443,0.0524945],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget celda3:=[[397.27,75.10,39.50],[0.0191015,-0.35616,-
0.933524,0.0363043],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget celda4:=[[403.19,4.52,39.50],[0.0215309,-0.347544,-
0.936918,0.0305543],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget celda5:=[[324.39,210.58,39.50],[0.0132388,-0.369049,-
0.927582,0.0567312],[0,-1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
CONST robtarget
celda6:=[[330.18,144.42,39.50],[0.00139955,0.349777,0.936554,-0.0228229],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];



PROC main()


    SocketCreate serverSocket;
    SocketBind serverSocket, "192.168.125.1", 8000;
    SocketListen serverSocket;
    SocketAccept serverSocket,clientSocket,\Time:=WAIT_MAX;
    SocketSend clientSocket \str:="Conexion con abb correcta";
    WHILE D652_10_DI6=0 DO
    SocketReceive clientSocket \str:=data;



    XValorString := StrPart(data,1,3);
    okX:=StrToVal(XValorString,cambioStrX);
    YValorString := StrPart(data,4,3);
    okY:=StrToVal(YValorString,cambioStrY);
  SetDO D652_10_DO1,1;
  MoveL Offs(pos_inicial,(170.25*(198-cambioStrX)/(198-513)),(-141.26*(190-
cambioStrY)/(190-427)),150), v1000, fine, Pinza;
    WaitTime(0.2);
  MoveL Offs(pos_inicial,(170.25*(198-cambioStrX)/(198-513)),(-141.26*(190-
cambioStrY)/(190-427)),0), v1000, fine, Pinza;
    WaitTime(0.2);
  SetDO D652_10_DO1,0;
  WaitTime(0.2);
```

```
    MoveL Offs(pos_inicial,(170.25*(198-cambioStrX)/(198-513)),(-141.26*(190-
cambioStrY)/(190-427)),150), v1000, fine, Pinza;
    WaitTime(0.2);
    MoveL Offs(celda1,0,0,150), v1000, fine, Pinza;
    WaitTime(0.2);
    MoveL celda1, v1000, fine, Pinza;
    WaitTime(0.2);
    SetDO D652_10_DO1,1;
    WaitTime(0.2);
    MoveL Offs(celda1,0,0,150), v1000, fine, Pinza;
    WaitTime(0.2);
    MoveL home, v1000, fine, Pinza;

      ENDWHILE

            stop;

    ENDPROC
ENDMODULE
```

8.5. Código en Python

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

mi_socket = socket.socket()
mi_socket.connect(("192.168.125.1" , 8000))



captura=cv2.VideoCapture(1)
Contador = int(0)

while True:

    while(captura.isOpened()):
        ret,imagen=captura.read()
        if ret==True:
            cv2.imshow('video',imagen)
            if cv2.waitKey(1) & 0xFF == ord('s'):
                cv2.imwrite('imgcapturada.png',imagen)
                cv2.destroyAllWindows()
                break

    foto=imagen[25:205,225:425]
    imageOut = cv2.resize(foto,(500,450), interpolation=cv2.INTER_CUBIC)
    gaussiano = cv2.GaussianBlur(imageOut, (5,5), 0)
```

```
    hsv =cv2.cvtColor(gaussiano, cv2.COLOR_BGR2HSV)
    #rango1=negro
    rango1 = np.array([100, 45, 45])
    #rango2=azulclaro
    rango2 = np.array([120, 255, 255])
    mascara=cv2.inRange(hsv,rango1, rango2)
    _, contours, _ = cv2.findContours(mascara, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)


    for c in contours:
        area = cv2.contourArea(c)
        if area > 10000 and area < 100000:
            M=cv2.moments(c)
            if (M["m00"]==0):M["m00"]=1
            x=int(M["m10"]/M["m00"])
            y=int(M["m01"]/M["m00"])
            cv2.circle(imageOut,(x,y),7,(0,255,0),-1)
            font=cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(imageOut,'{},{}'.format(x,y),(x-
40,y+40),font,0.75,(0,255,0),1,cv2.LINE_AA)
            nuevoContorno=cv2.convexHull(c)
            cv2.drawContours(imageOut,[nuevoContorno],0,(255,0,0),3)
            x=(x+100)
            y=(y+100)
            mi_socket.send(str(x)+str(y))
            time.sleep(0.2)
            print(x)
            print(y)
            Contador = (Contador + 1)
```

```
            cv2.imshow('contornos', imageOut)
            cv2.imshow('umbral', mascara)

    if Contador>3:break

cv2.waitKey(0)
cv2.destroyAllWindows()
```