UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Aplicación de pick and place de piezas de LEGO con visión artificial en un robot colaborativo

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Aparicio Gonzalez, Javier

Tutor/a: Ivorra Martínez, Eugenio

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO DE FIN DE GRADO

Grado de Ingeniería Electrónica Industrial y Automática

Aplicación de "pick and place" de piezas de LEGO con visión artificial en un robot colaborativo

Septiembre de 2022 Curso 2021/22

Autor: Javier Aparicio González

Tutor: D. Eugenio Ivorra Martínez

Agradecimientos

Este trabajo que ahora presento, es el final de estos 4 años llenos de muchos buenos momentos y algunos que otros no tan buenos, de mucho trabajo y sacrificio. En estos años he aprendido mucho, tanto de los conocimientos que me he ha aportado la carrera que han sido innumerables, como otros muchos conocimientos personales y laborales.

Todos estos años a pesar de todo, han sido maravillosos, ya que he tenido la suerte de poder compartirlos con unos compañeros de clase, que se han convertido en unos compañeros de vida, mis amigos, que hemos superado todos juntos este difícil camino, superando todas las pruebas y ayudándonos los unos a los otros, y también disfrutando de estos años.

Como no, también agradecer a toda mi familia por apoyarme en todo durante estos años, lo primero por permitirme estudiar aquí en Valencia, y por apoyarme y estar en todo a mi lado en estos 4 años.

No me puedo olvidar de agradecer al equipo de Siemens Paterna por haberme permitido desarrollar mis prácticas en Siemens y acogerme tan bien, en especial a mi tutora Eva.

Y por último, a todos aquellos profesores de la UPV que me han hecho mantener vivas las ganas de querer aprender y disfrutar de la electrónica, y sobre todo a Eugenio Ivorra por ayudarme y tutorizar este proyecto.

Resumen

El trabajo desarrollado consiste principalmente en una aplicación de visión artificial desarrollada con el software Merlic de MVTec donde la cámara detecta las piezas de construcción MegaBloks correspondientes para construir cualquier figura con estas.

Para ello las piezas de MegaBlocks pasan por una cinta transportadora, y la cámara, colocada en una posición fija arriba de la cinta transportadora, analiza y clasifica las piezas mediante un programa de aprendizaje profundo, o inteligencia artificial, desarrollado con el software Deep Learning Tool también de MVTec. Comparando esta clasificación de las piezas con una base de datos donde se almacena la información de las piezas necesarias y la posición de estas en el área de montaje, el software manda una señal por TCP/IP (Socket) a un robot colaborativo de Universal Robots, en concreto un UR3e, así este robot coge la pieza de la cinta transportadora y la coloca justo en su sitio para construir la pieza final, que es un puente de unas 10 piezas distintas tanto en forma y color.

Para ello previamente hemos realizado una calibración de la cámara y un preprocesamiento de la imagen con Merlic para el correcto funcionamiento de la clasificación de piezas con el software de aprendizaje.

En cuanto a los resultados obtenidos a la finalización del proyecto son satisfactorios, consiguiendo un correcto funcionamiento de toda la aplicación, tanto en el robot como en la cámara de visión artificial, logrando unos buenos porcentajes de precisión en los modelos en los clasificadores de piezas.

Abstract

The work to be developed consists mainly of an artificial vision application developed with MVTec's Merlic software where the camera will detect the corresponding MegaBloks construction pieces to build any figure with them.

To do this, the MegaBlocks pieces will go through a conveyor belt, and the camera, which will be placed in a fixed position above the conveyor belt, will analyze and classify the pieces through a deep learning program, or artificial intelligence, developed with the Deep Learning Tool software also from MVTec. Comparing this classification of the parts with a database containing the information on the necessary parts and their position in the assembly area, the software will send a signal via TCP/IP (Socket) to a Universal Robots collaborative robot, specifically a UR3e, so that this robot takes the piece from the conveyor belt and places it exactly in its place to build the final piece, which in our case will be a bridge of about 10 different pieces in both shape and color.

For this, it will be necessary to previously calibrate the camera and the robot, in addition to preprocessing the image with Merlic for the correct functioning of the part classification with the learning software.

As for the results obtained at the end of the project, they are satisfactory, achieving a correct operation of the entire application, both in the robot and in the artificial vision camera, achieving good percentages of precision in the part classifiers.

Resum

El treball a desenvolupar consisteix principalment en una aplicació de visió artificial desenvolupada amb el programari Merlic de MVTec on la cambra anirà detectant les peces de construcció MegaBloks corresponents per a construir qualsevol figura amb aquestes.

Per a això les peces de MegaBlocks aniran passant per una cinta transportadora, i la cambra, que estarà col·locada en una posició fixa a dalt de la cinta transportadora, anirà analitzant i classificant les peces mitjançant un programa d'aprenentatge profund, o intelligència artificial, desenvolupat amb el programari Deep Learning Tool també de MVTec. Comparant aquesta classificació de les peces amb una base de dades on estarà la informació de les peces necessàries i la posició d'aquestes en l'àrea de muntatge, el programari manarà un senyal per TCP/IP (Socket) a un robot collaboratiu d'Universal Robots, en concret un UR3e, perquè aquest robot agafe la peça de la cinta transportadora i la colloque just en el seu lloc per a construir la peça final, que en el nostre cas serà un pont d'unes 10 peces diferents punt en forma i color.

Per a això prèviament caldrà fer un calibratge de la cambra i del robot, a més d'un preprocessament de la imatge amb Merlic per al correcte funcionament de la classificació de peces amb el programari d'aprenentatge.

Quant als resultats obtinguts a la finalització del projecte són satisfactoris, aconseguint un correcte funcionament de tota l'aplicació, tant en el robot com en la càmera de visió artificial, aconseguint uns bons percentatges de precisió en els models en els classificadors de peces.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y descripción del problema

En esta pequeña introducción describiremos el problema a solucionar, que es exactamente lo que pretendemos lograr con este proyecto, y como le vamos a poner solución usando la robótica y la visión artificial como nuestras dos armas fundamentales.

El problema que se nos plantea es la construcción de una figura con piezas de construcción, de MegaBloks, parecidas a las de Lego, de manera automática. Esto lo vamos a solucionar como bien dice en el título del TFG, con un pick and place con un robot colaborativo de Universal Robots, en nuestro caso el UR3e. Para ello combinaremos dos grandes ramas de la ingeniería como son la robótica y la visión artificial para identificar las piezas.

Siendo más concretos, construiremos un puente, como el de la siguiente figura. Como se observa, el puente esta formado por 10 piezas de diferentes colores y formas, que tendremos que identificar y colocar en su posición en un orden concreto para así construir la figura. Esto lo haremos con un sistema de visión artificial compuesto principalmente por un clasificador de objetos previamente entrenado con una herramienta de aprendizaje profundo.

Figura 1.1: Figura de montaje del robot: Puente de 10 piezas

Para ello observamos todas las piezas que tenemos para poder entender bien el problema y en estas figuras se muestran todos los tipos de piezas tanto en forma y color.

Figura 1.2: Piezas de construcción, piezas posibles válidas

Figura 1.3: Piezas identificadas como desconocidas

Entre todas estas piezas, identificadas y clasificadas por la aplicación de visión artificial, el programa le mandará esta información al robot y el propio robot, comparará esta información con los datos de las piezas buscadas para comprobar si es válida o no. Si la pieza es válida, el robot irá a por ella y la colocará en la mesa de montaje en su sitio correcto, por el contrario, si la pieza no es la pieza buscada se volverá a activar la cinta desechando así la pieza, y así consecutivamente hasta acabar con el montaje de la figura completa.

Capítulo 2

Relacción del proyecto con los ODS

Hoy en día, en el mundo que vivimos, todos debemos luchar por la sostenibilidad de nuestro planeta, aunque sea con proyectos pequeños como este. En este otro capítulo identificaremos los ODS que están relacionados con este proyecto y que aporta este proyecto a luchar con la sostenibilidad mundial.

Tras la importancia de la sostenibilidad, y buscando un mundo más sostenible, desde el 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. Así nacieron los 17 Objetivos de Desarrollo Sostenible (ODS) cada uno con unas metas específicas para lograr en 15 años.

Figura 2.1: Objetivos de Desarrollo Sostenible

Observando todos estos ODS, aunque el proyecto parece que no está relacionado con ninguno de ellos en su totalidad, si que esta relacionado con los objetivos 8 y 9.

2.1 Relación con el ODS 8: Trabajo decente y crecimiento económico

La relación de este ODS con el proyecto es bastante fuerte, ya que este podría ser un proyecto cualquiera de automatización de cualquier fábrica o puesto de trabajo, que aunque a simple vista, parezca que así se reducen los puestos de trabajo, no siempre es así, si no que se buscan puestos de trabajo más cualificados, buscando más un trabajo decente, como dice este ODS. Esto es así ya que ahora no se necesitará un operario que, como en nuestro caso, monte la figura manualmente, si no que programe el robot, por eso el puesto de trabajo actual es más cualificado y mejor pagado, ayudando a la lucha contra los trabajos precarios.

Además este mismo ODS también habla del crecimiento económico, que obviamente este proyecto también conseguirá aumentar la productividad de cualquier proceso, ya que al automatizarlo consigues más rapidez y eficiencia, ayu-

dando considerablemente al crecimiento económico de la fábrica o empresa a automatizada. Y ya no solo al crecimiento económico de la fábrica, si no también de los trabajadores, ya que al ser puestos de trabajo más cualificados serán mejor pagados y los empleados tendrán más solvencia económica

2.2 Relación con el ODS 9: Industria, innovación e infraestructuras

Es indiscutible que este proyecto, esta muy relacionado con la innovación como así dice este ODS. Este es un proyecto innovador, para que ayude a crecer a las industrias automatizando sus procesos y modernizando sus infraestructuras, haciendo crecer a su empresa. Esta claro, que invertir en innovación y mejorar las infraestructuras es sostenible, y así poder aprovechar al máximo todos los recursos e infraestructuras.

Capítulo 3

Objetivos del proyecto

En el siguiente capítulo, enumeraremos y explicaremos cuales son todos los objetivos que se pretenden lograr con este proyecto.

Hablando de los objetivos del proyecto, además del objetivo principal, que es el proyecto en sí, lo podemos desglosar en otros muchos objetivos que iremos cumpliendo a lo largo de todos los meses de trabajo conforme el proyecto vaya avanzando. Estos objetivos son los siguientes:

- `Desarrollar y entrenar una herramienta Deep Learning`: Para ello, tomamos un gran número de fotos de las piezas y las clasificamos para así entrenar la herramienta de aprendizaje profundo y obtener un clasificador de piezas, tanto como de color y forma.

- `Desarrollar y crear un programa de visión artificial`: El objetivo del programa de visión artificial es primero preprocesar la imagen hasta conseguir una imagen nítida y clara para aplicarle el clasificador previamente entrenado para que así este programa devuelva el tipo de pieza y su posición central. Además también crearemos con el mismo programa una interfaz muy visual para que con un simple vistazo veas que tipo de pieza es.

- `Configurar el robot UR3e con todas sus entradas y salidas`: Definir todas las entradas y salidas con sus correspondientes puertos de conexión, asegurar todas las conexiones,ajustar la carga y centro de gravedad de la herramienta, y todas las configuraciones necesarias para antes de empezar a programar el robot.

- `Crear y conectarse al protocolo de comunicación por Socket`: Conectar al robot y poner las instrucciones necesarias para que cuando esté el coche pasando por el sensor final de carrera de la cinta, avise al programa de visión artificial para que también por socket le devuelva al robot el tipo de pieza y su posición central.
- `Desarrollar y programar el código principal del robot`: Con este objetivo lograremos culminar el proyecto en sí, el robot con todos los datos que le llegan por Socket y comparando con datos fijos predefinidos, decidirá si la pieza es válida y si lo es, la transportará a un punto predefinido para así construir la figura.
- `Redactar la memoria del proyecto en formato LaTeX`: Aunque no sea un objetivo del proyecto en sí, sí que es un objetivo de este TFG para conseguir un documento de mucha más calidad tipográfica.

Capítulo 4

Antecedentes y soluciones comerciales

En este cuarto capítulo, pondremos en antecedentes las dos grandes ramas de conocimiento relacionadas con este proyecto, que son la visión artificial y la robótica, en concreto, de los robots colaborativos de Universal Robots.

4.1 Antecedentes de la visión artificial

La visión artificial, o también llamada visión por computadora, es la disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica, para que puedan ser tratados por un ordenador. En la actualidad, muchos campos se han visto beneficiados por estas técnicas, siendo el que más, el campo de la robótica, para poder darle autonomía a estos y por ejemplo como en nuestro caso, que puedan reconocer con precisión la localización de objetos.

La historia de la visión artificial se remonta a la década de los años 60 con el origen de un prototipo automatizado basado en cámaras de visión artificial y sistemas de procesamiento de las imágenes captadas para así poder analizarlas. Pero realmente, la historia de la visión artificial marca su máximo esplendor en la década de los 80 con el desarrollo de la ingeniería informática y la creación de procesadores más sofisticados y rápidos, dando lugar a microprocesadores

más avanzados capaces de captar, procesar y reproducir imágenes tomadas por una cámara que podría estar conectada de forma remota.

Esta tecnología se fue desarrollando en todas sus aplicaciones desde hace décadas y mejorando su uso en aplicaciones tan variadas como en los controles de calidad, ya sea en el sector químico alimentario, sanitario y en muchos más. Hoy en día, la visión artificial amplía su potencial con la industria 4.0 y el empleo de esta tecnología en la automatización de cualquier sistema, como por ejemplo, la utilización de la inteligencia artificial, una herramienta que tiene la capacidad de aprender y tomar decisiones según la información que capta y procesa.

4.1.1 La herramienta de Deep Learning o aprendizaje profundo

Esta aplicación dentro de la visión artificial hace posible que el sistema opere como si se tratara de un ojo humano y cerebro humano por su capacidad de analizar, evaluar y tener en cuenta todas sus variables, pero con una mayor capacidad de respuesta por la posibilidad de procesas datos masivos (Big data) y analizarlos en tiempo récord. Es decir, el Deep Learning, o el aprendizaje profundo, se define como un algoritmo automático estructurado o jerárquico que emula el aprendizaje humando, con el fin de obtener información.

La primera vez que apareció el termino de Deep Learning fue en 1974, aunque en ese momento la aplicación no estaba del todo testada, fue gracias a Paul Werbos que dio a conocer la existencia de un entrenamiento de aprendizaje a redes neurales retropropagadas. A esta tesis, le siguió Geoffrey E. Hinton que, tras conocer el algoritmo, intentó aplicarlo en máquinas para simular en ellas el aprendizaje humano. Con otros científicos, consiguieron avanzar en el algoritmo, pero no fue hasta 2010, con la llegada del Big Data, cuando se entendiese el mecanismo.

El mecanismo del funcionamiento del Deep Learning se trata de un algoritmo capaz de aprender el solo como comportarse, es decir, no requiere de reglas programadas previamente, sino que el propio sistema es capaz de "aprender" por sí mismo a través de una fase previa de entrenamiento. También se caracteriza por estar compuesto por redes neuronales artificiales entrelazadas para el procesamiento de información. Los algoritmos que componen un sistema de aprendizaje profundo se encuentran en tres capas:

- `Capa de entrada (input layer)`: Compuesta por las neuronas que recogen los datos de entrada, como una imagen o una tabla con datos numéricos

- `Capa oculta (hidden layer)`: En esta es donde se realizan los cálculos para proceder a ejecutar acciones. A más datos, mayor complejidad.
- `Capa de salida (output layer)`: En la última capa es donde el sistema toma la decisión y aporta los datos que lo justifican.

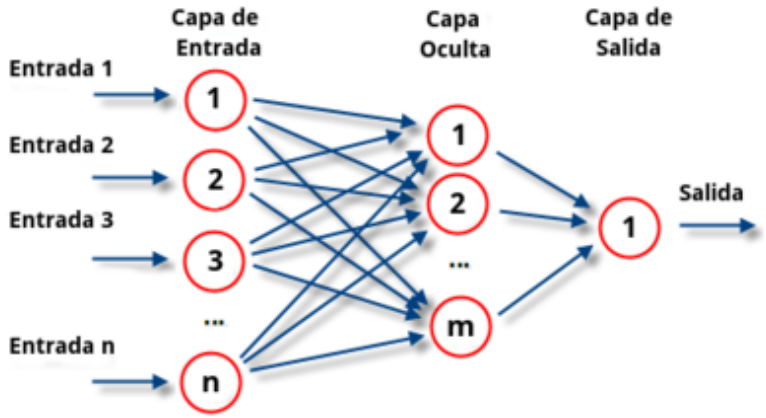


Figura 4.1: Funcionamiento y capas del Deep Learning

4.2 Antecedentes de la robótica

La robótica es la rama de la ingeniería mecánica, electrónica y ciencias de la computación, que se ocupa del diseño, construcción, operación, manufactura y aplicación de los robots. Además la robótica combina diversas disciplinas como la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física.

La historia de la robótica se remonta a la creación de "artefactos" que trataban de materializar el deseo humano de crear seres a su semejanza y que lo descargasen de trabajos tediosos o peligrosos. El ingeniero español Leonardo Torres Quevedo, que construyó el primer mando a distancia para su automóvil mediante telegrafía y otros muchos ingenios, fue el primer ingeniero que acuñó el término de automática.

Más tarde el escritor Karel Čapek, en 1920 acuñó el término robot en una de sus obras a partir de la palabra checa robota que significa servidumbre o

trabajo forzado. Pero realmente el primer ingeniero en acuñar el término de robótica fue Isaac Asimov en 1942, definiéndola como la ciencia que estudia a los robots, Isaac Asimov también creó las tres leyes de la robótica.

Muchos más han sido los ingenieros que han contribuido al avance y desarrollo de esta disciplina, como George Devol, creando el primer robot comercial llamado *Unimate* en 1956, hasta llegar a nuestros días con la creación de los primeros robots humanoides en 2002, ahora ya capaces de desplazarse de forma bípeda y de reconocer y recordar caras o simular expresiones.

4.2.1 Los robots colaborativos y el UR3e

Los robots colaborativos o también llamados cobots, son aquellos robots capaces de trabajar junto a los humanos, por ejemplo, en una cadena de producción. Estos robots colaborativos son capaces de llevar la automatización industrial a nuevas alturas, ya que pueden ocuparse de tareas inaccesibles para los robots cotidianos.

La primera vez que se utilizó el término de robótica colaborativa fue en 1999 por la combinación de las dos palabras colaboración y robots. Sus principales características son estas mismas:

- `Colaboración`: La interacción con los humanos es fundamental, ya que estos robots fueron creados para ayudar a las personas
- `Automatización`: Un cobot es capaz de realizar tareas automatizadas, consiguiendo impulsar la productividad a otros niveles.

En nuestro caso, utilizaremos el robot colaborativo UR3e de Universal Robots. Esta compañía danesa pionera en robots colaborativos fue fundada en el 2005 tras conocerse en la Universidad de Dinamarca, Esben Østergaard, Kasper Støy y Kristian Kassow y en el 2008 se produjo la primera venta de un robot colaborativo, el UR5. En el 2015 se lanza el UR3, y en el 2018 los nuevos robots e-Series, al que pertenece el UR3e, que será el robot que utilizaremos para el proyecto.

Este robot UR3e es el robot evolución del UR3 que mejora la productividad y la calidad del producto respecto a su anterior modelo. El UR3e es un ligero brazo robótico, tan solo de 11,2 kg, de sobremesa que permite la automatización de bancos de trabajo gracias a su tamaño compacto. Además tiene una capacidad de carga de 3 kg, y cuenta con una rotación de 360º en ambos sentidos y en

todas sus articulaciones, excepto en el eje del extremo superior que cuenta con una rotación infinita.

En la siguiente imagen, aparece una vista de este robot, donde se aprecian todas sus articulaciones y grados de libertad:

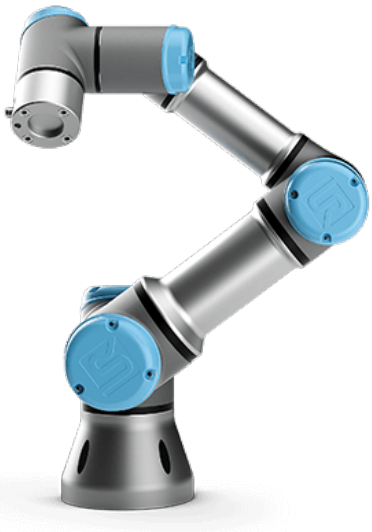


Figura 4.2: Robot colaborativo UR3e

Además cuenta con un sistema de programación que permite que sea muy sencillo de configurar, con soluciones flexibles y cambiantes, minimizando los costes de puesta en marcha. Por último, también permite ajustar hasta 17 opciones de seguridad avanzada, como por ejemplo el límite de fuerza, ya que el robot se detiene al detectar una fuerza de 150 N en cualquier punto del robot, parámetro que se puede cambiar hasta los 50 N.

Capítulo 5

Hardware y software utilizado

En este capítulo, explicaremos todo el montaje realizado, es decir, comentaremos cuales son los programas que hemos utilizado tanto como para la parte de visión artificial como para la programación de la robótica y también todo el equipo utilizado y sus conexiones y comunicaciones con todos los demás elementos para el correcto funcionamiento del robot.

5.1 Software utilizado para la parte de visión artificial

Para la parte de visión artificial hemos utilizados 2 programas, que ambos están diseñados para funcionar complementariamente, ya que son de la misma compañía, que es MVTec Software GmbH, una empresa pionera en software para soluciones con visión artificial. Una de las aplicaciones utilizadas es la herramienta Deep Learning Tool, que la hemos utilizado para crear los clasificadores encargados de identificar las piezas, mientras que la otra aplicación es Merlic, que será la aplicación principal de visión artificial que utilizará estos clasificadores.

5.1.1 La herramienta de Deep Learning Tool

Esta herramienta es la utilizada para entrenar los dos clasificadores que hemos utilizado para clasificar las piezas, uno para la forma y otro para el color. Para ello se meten todas las fotos de todas las piezas y de varias posiciones y orientaciones y se clasifican. Es una aplicación muy sencilla e intuitiva, que una vez que tienes clasificadas todas las fotos divides las fotos en fotos de entrenamiento, validación y test, para así entrenar el clasificador y poder evaluarlo obteniendo la precisión y la matriz de confusión de cada clasificador.

En nuestro caso y el caso más típico es que el porcentaje de fotos de entrenamiento sea mucho más superior que al de validación y test, para así obtener un buen modelo, como por ejemplo, nuestra selección de un 70% para las fotos de entrenamiento y un 15% para las fotos de validación y test.

Una vez entrenado el clasificador se comprueba la precisión del mismo, al igual que su matriz de confusión y si estos datos son aceptables, se puede concluir con que el clasificador es un buen clasificador y que por lo tanto ya se puede usar en la aplicación de visión artificial. Para ello exportamos el modelo o clasificador en formato .hdict para que luego nos lo coja Merlic.

5.1.2 La aplicación de visión artificial Merlic

Esta aplicación sera la aplicación principal de visión artificial y que le dará sentido a los clasificadores creados anteriormente. Merlic será la aplicación encargada de comunicarle al robot que tipo de pieza es, su color y su posición para que así el robot decida si esa pieza es válida para cogerla o no.

Para que todo funcione correctamente, Merlic en primer lugar debe esperar el aviso que el robot le mandará a Merlic por socket para decirle que la pieza ya está al final de la cinta transportadora y que tiene que clasificar, aquí es cuando Merlic cogerá la imagen de la cámara, la calibrará para poder obtener una posición de la pieza lo más exacta posible y después la preprocesará con filtros para en nuestro caso, mejore la calidad de la imagen, aunque según la utilidad del sistema de visión artificial el preprocesamiento puede ser distinto, como por ejemplo convertir la imagen en una imagen binaria, o extraer los distintos colores de una imagen RGB.

Después de la calibración y el preprocesamiento de la imagen, viene el proceso, en nuestro este proceso será doble, en primer lugar clasificar la pieza con los clasificadores que adjuntaremos a Merlic y así obtendremos el tipo de pieza y su color, y por otra parte, obtener la posición de la pieza, mediante una función

específica para esto de Merlic. Finalmente mandaremos toda esta información por socket

Por último Merlic también tiene un opción de crear una interfaz con el usuario o dashboard para mostrar de una manera muy fácil e intuitiva toda la información y poder interactuar con el programa. Nuestro dashboard se compone de la imagen obtenida por la cámara y dos pantallas donde aparecerá el tipo de pieza y su posición, es decir, toda la información que nos puede proporcionar Merlic.

5.2 Software utilizado para la parte de programación del robot. PolyScope

Para la programación del robot se usa el programa de Universal Robots llamado PolyScope, un programa muy intuitivo y fácil de usar. El programa PolyScope se divide en 3 partes: encabezado, pantalla y pie de página. En el encabezado se compone de varios botones para elegir distintas opciones para realizar con el robot. Por ejemplo, en el botón de instalación es donde se configuran todos los ajustes, como entradas y salidas, seguridad, parámetros de la herramienta, etc. En el de E/S para activar o desactivar estas de manera manual, el de mover para moverlo de forma manual, y el más importante el de programa, donde se construirá con todas las operaciones el programa que queremos que ejecute el robot, entre otras muchas más opciones.

Nos centraremos en la parte de la programación de un programa, cuya interfaz general es esta:

Figura 5.1: Pantalla PolyScope para programar un programa

Como se observa, en la columna de la izquierda del todo están todos los comandos que se pueden utilizar para realizar el programa, separados en comandos básicos, avanzados o plantillas. En la columna del medio de la pantalla, aparece el árbol de programa, donde se va creando la estructura del programa según las instrucciones que se vayan añadiendo, y por último en la parte de la derecha en la pestaña Comando ahí es donde aparecerán todos los ajustes de cualquier instrucción o nodo para configurar de forma correcto el árbol de programa.

En cuanto a los comandos o nodos más importantes y que utilizaremos en nuestro programa, vamos a comentar y explicar algunos de ellos:

5.2.1 Comando Mover de PolyScope

Este comando controla el movimiento del robot, a través de los puntos de paso. Existen tres tipos de movimientos:

- `moveJ`: Realiza movimientos calculados en el espacio articular del brazo robótico. Cada junta articular se controla para llegar a la ubicación final deseada, por lo tanto, da lugar a una trayectoria curva. Para realizar este movimiento tendremos que darle como parámetros la velocidad y la aceleración máximas de las juntas. Este movimiento es idóneo cuando queremos que el robot se mueva entre 2 puntos de paso sin tener en cuenta la trayectoria de la herramienta.

- `moveL`: Este movimiento mueve el TCP (Punto Central de Herramienta) linealmente entre 2 puntos de paso, realizando así movimientos más complejos en cada junta para que mantenga la herramienta en una trayectoria recta. En este caso, los parámetros a pasarle son la velocidad y la aceleración de la herramienta, y además una función que determinará en que espacio se posiciona la herramienta de los 2 puntos de paso representados.

- `moveP`: Por último, este movimiento, mueve la herramienta linealmente a una velocidad constante con transiciones circulares. El parámetro a completar es el radio de transición, que cuanto más pequeño hará que la trayectoria sea mucha más brusca. Este movimiento es muy usado para ciertos procesos como el encolado o la dispensación, y también para realizar movimientos circulares a través de un ViaPoint especificado en el arco circular y un EndPoint.

Además de los diferentes tipos de movimiento y sus parámetros también hay que ajustar los puntos de paso, que estos pueden ser fijos, relativos o variables. Estos puntos de paso, independientemente de que tipo sean son la parte más importante del programa del robot, ya que le dicen al brazo robótico donde se tiene que dirigir.

- `Punto de paso fijo`: Para seleccionar los puntos de paso fijos, se pueden escribir en la pantalla o moviendo el robot hasta el punto que queremos, ya sea, moviendo las articulaciones o el TCP desde la pantalla de Mover o pulsando el botón para poder mover el robot manualmente y fijando el punto. Estos puntos de paso reciben automáticamente un nombre distinto a los demás, pero se pueden enlazar varios y así comparten la información.

- `Punto de paso relativo`: Se trata de un punto de paso con una posición relacionada con la posición anterior, es decir, un desplazamiento respecto a la posición anterior.
- `Punto de paso variable`: En este caso, el punto de paso viene dado por una variable, donde la variable tiene que ser una pose. Las variables pose están compuestas de 6 dígitos, donde los tres primeros indican la posición con X, Y, Z y las tres últimas indican la orientación como un vector de rotación dado por el vector Rx, Ry, Rz.

5.2.2 Comando Esperar de PolyScope

El comando Esperar, como dice su nombre, pausa la señal E/S, o la expresión, durante un tiempo definido. Este tiempo definido puede venir dado por muchos casos, ya sea durante un tiempo fijo establecido o hasta que cambie alguna entrada digital o analógica o incluso hasta que se cumpla alguna expresión.

5.2.3 Comando Ajustar y Asignación de PolyScope

El comando Ajustar sirve simplemente para ajustar salidas digitales o analógicas para un valor dado, es decir, para cambiar las salidas de valor. Además de cambiar o ajustar las salidas, también sirve para ajustar la carga del brazo robótico, y así si cambia el peso de la carga, evitar que se active una parada de protección cuando este peso difiera de la carga prevista. Por otro lado, el comando Asignar es para asignar valores a variables, ya sea un valor fijo o un valor dado por una expresión. Es decir, el comando Ajustar asigna valores para las salidas, mientras que el comando Asignación a las variables

5.2.4 Comando Subprograma y Subproceso de PolyScope

Es ideal para albergar partes de un programa que se vayan a ejecutar en varias partes del programa, o quizás para ocultar partes de un programa y así protegerlo de cambios involuntarios. Para invocar este subprograma, se hará con el comando Invocar, que ejecutará las líneas del subprograma y luego se regresará a la línea siguiente.

Mientras el comando Subproceso se utiliza igual que el comando Subprograma pero en este caso, para ejecutar a la vez varias líneas de programa, es decir, ejecutar en paralelo el programa principal y el subprograma. Este subproceso se puede comunicar con el programa principal mediante variables y señales de salida.

Por último cabe destacar, que hay muchos más comandos, que quizás son más conocidos, o más comunes como por ejemplo los bucles, o el If y todas sus variables como el Else o el ElseIf. O otros comandos simples como Aviso, que hace que aparezca un mensaje de aviso en la pantalla, o Detener, que simplemente detiene el programa. Para finalizar, comentar, que también existe la opción del uso de Scripts para realizar programas de mayor complejidad.

5.3 Hardware. Montaje de la instalación

Para el desarrollo del proyecto, antes de hacer nada, habrá que montar la celda o célula robótica, es decir, el conjunto de máquinas y dispositivos que se organizan dentro de un mismo espacio con un robot. En nuestro caso, esta celda robótica estará compuesto por una cámara de visión artificial, o en su defecto, una cámara cualquiera, un ordenador para el programa de visión artificial donde se mostrará la interfaz del programa y donde veremos las imágenes captadas por la cámara y las piezas que identifica con toda su información, y por último, una cinta transportadora, equipada con un sensor detector de objetos justo al final de la cinta transportadora, para ver si la pieza ya ha llegado al final de ésta y es válida que la coja el robot. Por último, pero más importante, el robot también forma parte del hardware, como ya se ha comentado a lo largo del proyecto, éste será el robot UR3e equipado con una ventosa de herramienta, para que así pueda coger las piezas, ya que estas piezas son ligeras y tienen la suficiente superficie para ser cogidas con esta ventosa.

Además de todos estos dispositivos, en el laboratorio también es necesario que haya conexión a internet, para que el robot y el ordenador se conecten a la misma red, y sea posible de una manera cómoda y sencilla la comunicación de ambos dispositivos por socket.

A continuación, hablaremos y explicaremos su función más detenidamente de los dispositivos más importantes:

5.3.1 Cámara de visión artificial

En nuestro caso, al no tener ninguna cámara de visón artificial, utilizaremos una cámara normal, que utilizaremos una webcam. La principal diferencia entre utilizar una cámara de visión artificial o no, es la calidad de la imagen obtenida, ya que una cámara de visión artificial ofrece un completo control de los tiempos y señales, la velocidad de obturación y otros factores fundamentales

que hacen que obtengas una imagen de muy alta calidad según el objetivo del proyecto.

Esta webcam estará situada justo arriba de la cinta transportadora, más bien al final de ésta, ya que el sistema de visión artificial se activará cuando la pieza este al final de la cinta y sea detectada por el sensor que está al final de la cinta. La webcam utilizada, es una Deluxe Optical Glass de la marca Hércules, como la mostrada en las siguientes imágenes, tanto como sin montar, como ya montada en la bandeja metálica perforada que esta instalada en la parte superior del laboratorio:

Figura 5.2: Webcam Hércules Deluxe Optical Glass

Figura 5.3: Montaje de la Webcam en el laboratorio

Respecto a la cámara es importante que tampoco se situé muy lejos y que cubra toda la cinta transportadora, buscando la mejor imagen posible sin que se quede ninguna pieza fuera de su visión. Para captar una buena imagen, aparte de la cámara, la iluminación también es un factor determinante, en nuestro caso, el laboratorio donde se ha realizado el montaje es una sala muy luminosa y no es necesario aportar ningún foco de iluminación artificial o especial.

Esta cámara se conectará a un ordenador mediante un puerto USB, este otro dispositivo, se situará fuera de la celda robótica, y será el encargado de ejecutar el programa de visión artificial, por lo tanto, es necesario que esté instalado en

él el programa Merlic. Una vez se este ejecutando el programa, en la pantalla del ordenador se mostrará el dashboard creado y aunque el usuario no puede interaccionar en nada con él, si que es una muy buena fuente de información para saber que tipo de pieza es y su posición.

5.3.2 Cinta transportadora con sensor óptico final de carrera

Por la cinta transportadora o *conveyor* será por donde vayan entrando las piezas al sistema, esta solo se moverá en un sentido, ya que irá colocando las piezas una a una un operario al principio de la cinta hasta que lleguen al final, es decir, hasta que las detecté el sensor final de carrera, ahí será cuando se active la parte de visión artificial para decidir si cogerla o no cogerla, si la pieza no es válida y no la tiene que coger el robot, habrá que desecharla, que para desecharla se volverá a activar la cinta dejándola caer en un cesto que se habrá colocado debajo del final de ésta, dando así paso a la siguiente cinta hasta el sensor.

La cinta estará colocada a un lado del robot, lo suficientemente cerca para que el robot llegue a coger las piezas, y el otro lado del robot, será la zona de montaje. Esta cinta, es una cinta pequeña para colocarla encima de la mesa de trabajo donde está el robot, y además es de color blanco, ideal para que el sistema de visión artificial distinga bien las piezas, ya que no hay ninguna pieza de color blanco y por lo tanto, en todas las fotos, el fondo será de un color distinto al de la pieza. Para el funcionamiento de la cinta transportadora, cuenta con un motor de corriente continua, y un variador de frecuencia.

La cinta, cuenta con 2 sensores final de carrera, uno a la entrada y otro a la salida, ambos idénticos el uno al otro, aunque nosotros solo utilizaremos el de la salida de la cinta. Este es un sensor fotoeléctrico, que puede funcionar de diversas maneras (modelo reflectivo, modelo de barrera o modelo retroreflectivo), pero además del sensor, enfrente de él hay un espejo que actúa como reflector, por lo tanto, nos indica que el sensor esta funcionando como un modelo retroreflectivo. Este sensor funcionando en modo retroreflectivo, funciona de la siguiente manera: El sensor se compone de 2 partes, del elemento emisor de luz y del receptor de luz, el emisor de luz emite una luz, aunque invisible al ojo humano, y si este rayo de luz llega al reflector y por lo tanto, lo refleja hasta el emisor de luz, será que no hay ningún objeto entre el sensor y el reflector, mientras que si se sitúa un objeto entre el sensor y el reflector, el rayo de luz nunca llegará al receptor de luz.

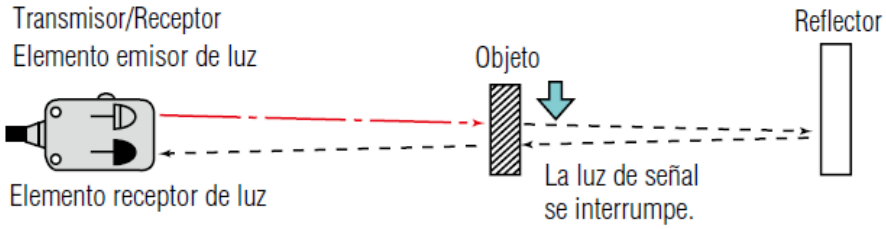


Figura 5.4: Funcionamiento del sensor mediante el modelo retroreflectivo

El sensor, cuando detecte algún objeto devolverá un nivel alto, es decir, se activará la entrada y mientras que no detecte ningún objeto, no se activará la entrada, esta entrada digital será la número 6 del robot UR3e. En concreto, se trata del sensor E3F1-RP11, un sensor fotoeléctrico en carcasa M18 compacta, como se muestra en las siguientes imágenes, antes y después del montaje de él en la cinta:

Figura 5.5: Sensor fotoeléctrico E3F1-RP11

Figura 5.6: Montaje del sensor en la cinta transportadora

En cuanto al motor de la cinta transportadora, es un motor reductor, en concreto el BWQ40, conectado a un variador de frecuencia, del tipo CFW80. Este montaje ya estará realizado previamente al proyecto, aquí se muestran unas imágenes del variador de frecuencia y el motor ya instalados correctamente en la cinta transportadora. Por último, indicar que este motor con su correspondiente variador de frecuencia se ha conectado a la salida digital 1 del robot, que será la encargada de mover la cinta transportadora.

Figura 5.7: Motor reductor BWQ40

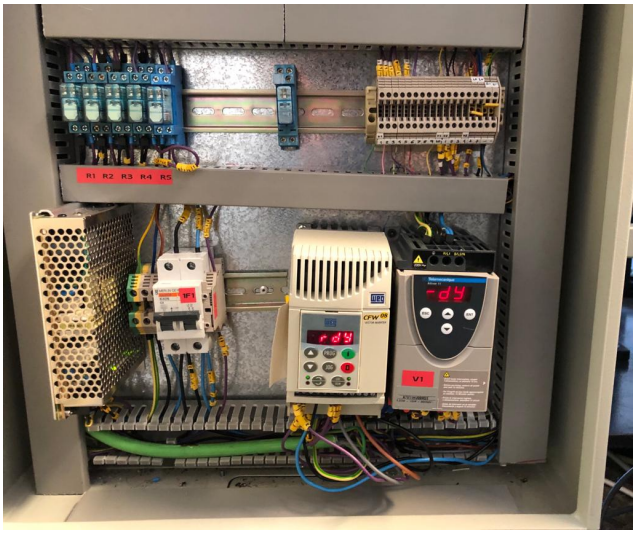


Figura 5.8: Cuadro eléctrico del variador de frecuencia CFW08

5.3.3 El robot UR3e con una ventosa como herramienta

El robot estará situado en medio de la celda robótica, es decir, entre la cinta transportadora y la zona de montaje, y será el encargado de transportar las piezas válidas de la cinta a la zona de montaje, y colocarlas en su sitio correcto para así ir montando la figura. Este robot irá equipado con una ventosa que irá conectada a la salida 5 del robot, y cada vez que se active esta salida, la ventosa hará vació con el objeto que tenga delante, y así será como el robot cogerá las piezas. Esta ventosa con la que contamos, esta sujeta a una pieza imprimida con una impresora 3D que a su vez, esta atornillada a un *quick changer* de OnRobot, que este *quick changer*, o cambiador rápido, permitirá un cambio rápido de herramienta, aunque para nuestro caso no es necesario, ya que solo utilizamos esta ventosa como herramienta. Este cambiador de herramientas,

como máximo soporta 10 kg y pesa unos 0,2 kg, dato que junto con el peso de la ventosa tendremos que tener en cuenta para configurar la ventosa como TCP adecuadamente, al igual que el peso de la ventosa. A continuación se muestran unas fotos de esta herramienta:

Figura 5.9: Herramienta del robot: Ventosa

Seguidamente también se muestran otras fotos del robot con la herramienta acoplada sin la necesidad de utilizar el *quick changer* con el que en las anteriores fotos de la ventosa si que aparece, y otra foto de la caja de control del robot, donde se ven todas estas conexiones a todos los periféricos que controla el robot, la correcta conexión a internet para su comunicación por socket y donde esta conectado también la consola portátil para el manejo y programación del robot.

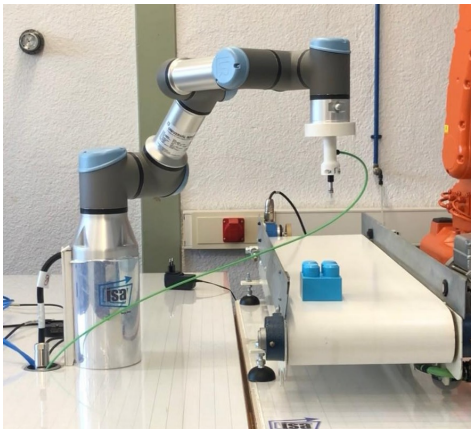


Figura 5.10: UR3e con ventosa acoplada

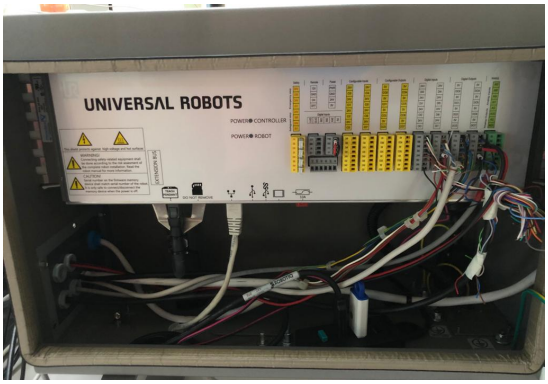


Figura 5.11: Caja de control del UR3e

5.3.4 Celda robótica completa

Para finalizar este apartado, mostraremos una imagen de la celda robótica al completo, donde se aprecie exactamente donde estará colocado el robot, la cinta transportadora y la zona de montaje de la figura. La cámara de visión artificial estará justo arriba de donde esta tomada la foto, es decir, arriba de la cinta transportadora para que pueda captar bien todas las piezas:

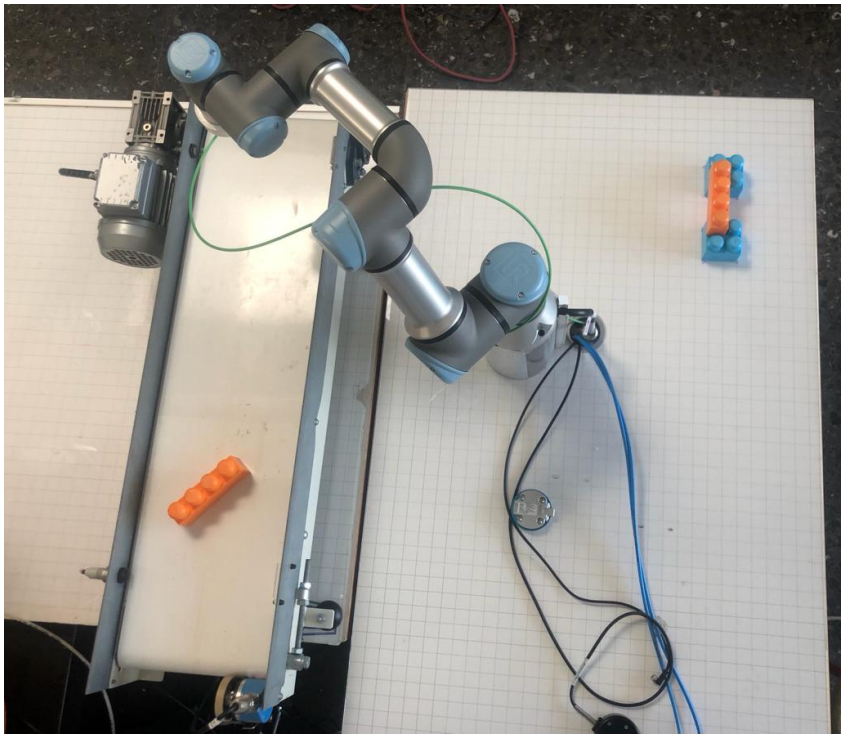


Figura 5.12: Montaje de la celda robótica

Capítulo 6

Descripción de la solución realizada

Este capítulo se puede considerar el más importante de la memoria, aquí será donde detallaremos paso a paso todos los pasos y procesos seguidos hasta dar con la solución final. Este capítulo se dividirá en varios apartados explicando con todo detalle como se han conseguido lograr todos los objetivos enumerados en el anterior capítulo 3.

Dividiremos todo el trabajo en varios apartados y con un orden de ejecución a seguir. En primer lugar, configuramos toda la aplicación de visión artificial, con sus clasificadores y todo lo necesario, y seguidamente la configuración y programación del robot.

6.1 Descripción de la solución general. Diagrama de flujo

Para el correcto funcionamiento del programa, antes de empezar a configurar los programas tanto como de visión artificial como el del robot, tenemos que tener claro el funcionamiento global de todos los dispositivos que interactúan en el proyecto. Para ello realizamos el siguiente flujograma:

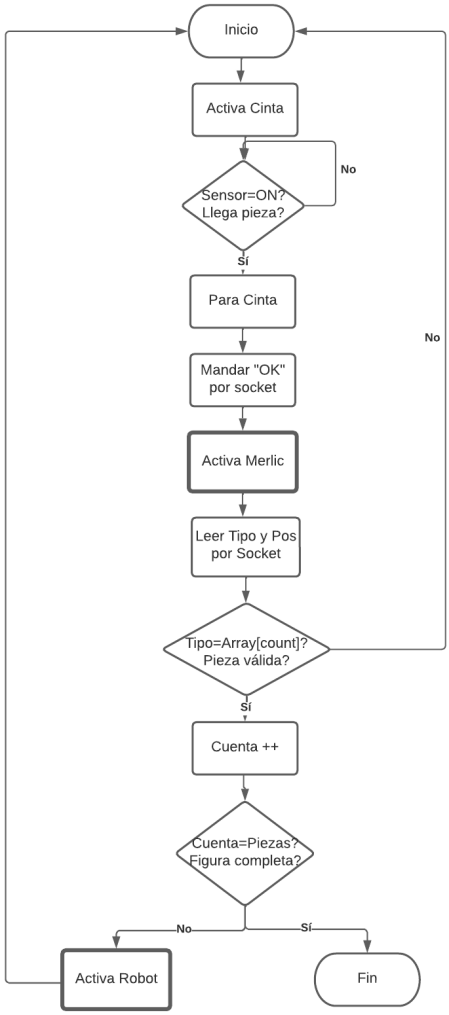


Figura 6.1: Flujograma general

Este es el flujograma general, donde realmente esta incompleto, ya que las acciones de Activa Merlic y Activa Robot, dentro de ahí hay otras muchas acciones, donde realizar otro flujograma que complete este.

Detallando un poco más este flujograma, vemos que lo primero se activará la cinta para que vayan pasando las piezas, hasta llegar al sensor del final de la cinta, donde la cinta se parará. En este punto, mandaremos una señal por socket para activar el programa de visión artificial que identificará la pieza y lo leerá el robot por socket. Aquí comprobaremos si la pieza coincide con la pieza que estamos buscando, y si es así la cogerá el robot, incrementando la cuenta de piezas cogidas y comprobando que no sea la última pieza, si esta pieza no es válida, volveremos a activar la cinta transportadora y repetir todo el proceso hasta completar la figura.

Para completar el flujograma anterior, este siguiente flujograma es el flujograma que corresponde con la acción de Activa Merlic, es decir, el flujograma de la parte de visión artificial.

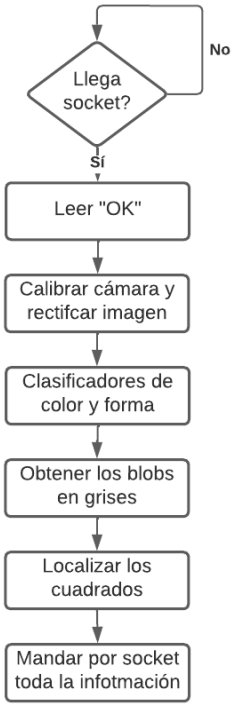


Figura 6.2: Flujograma de la parte de visión artificial

Esta otra parte del flujograma se activará seguidamente de mandar el socket para que se active la aplicación de visión artificial, entonces esta aplicación estará esperando la llegada del socket, que una vez se produzca se iniciará esta aplicación. Lo primero es calibrar la cámara, tanto como para rectificar la imagen y para obtener las posiciones en unidades reales y sobre el mismo sistema de coordenadas del robot. Esta imagen ya calibrada será la que se utilizará para que los clasificadores indiquen de que tipo es la pieza y para calcular la posición de cuadrado de la pieza. Finalmente esta información codificada de la manera que lo entienda el robot se le mandará por socket.

Así, únicamente para explicar todo el flujograma completo del robot solo falta, el último flujograma que se correspondería al bloque de Activa Robot, donde se detallará todo el proceso del robot para coger la pieza con todos los moveL o moveJ que realiza el robot.

Cuando la pieza sea válida y el robot tenga que coger esta pieza, lo primero el robot se moverá a la posición de la pieza pero una distancia más elevada, en nuestro caso unos 200 mm, donde ahí el robot podrá llegar a través de cualquier trayectoria, es decir, con un moveJ, una vez ahí, si que bajará mediante un moveL hasta la posición exacta de la pieza, mediante un movimiento relativo de unos 200 mm para abajo. En este punto, se activará la ventosa y volveremos a levantar el brazo robótico estos 200 cm con un moveL con la ventosa activada y por lo tanto, con la pieza. Después el robot leerá la posición de montaje en el vector donde estarán todas estas posiciones almacenadas y que definirán la pieza a montar y llegará a esta posición igual que a la posición de la pieza, es decir, mediante un moveJ hasta, en este caso, unos 250 mm arriba y seguidamente con un moveL hasta la posición exacta. Aquí desactivaremos la ventosa, esperaremos un pequeño tiempo y levantaremos el robot con un moveL hasta la posición de arriba, por lo tanto, ya estará la pieza colocada en su sitio, y en esta posición el robot se parará a la espera de ir a recoger la siguiente pieza.

Para acabar de comentar el flujograma, indicar que el proceso vuelve hasta el inicio otra vez, donde se vuelve a activar la cinta hasta que llegue la siguiente pieza válida y la coja el robot para colocarla en su posición hasta completar la figura de 10 piezas.

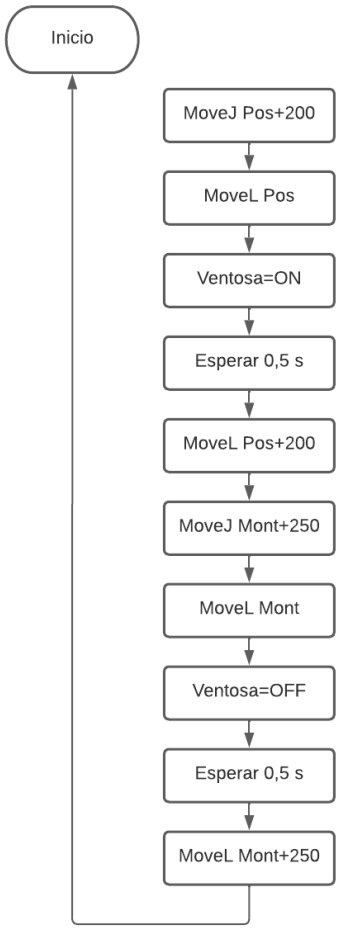


Figura 6.3: Flujograma de la parte del robot

6.2 Entrenamiento de los clasificadores

Como ya hemos comentado en el capítulo de Hardware y software utilizado, el entrenamiento de los clasificadores lo haremos con la herramienta de MVTec llamada Deep Learning Tool, y como ya hemos explicado su funcionamiento anteriormente, clasificaremos una a una las 265 fotos tomadas, indicando de que tipo son. Aunque antes que clasificar las piezas, para evitar distorsiones y objetos a los que no nos referimos, recortaremos todas las fotos dejando únicamente en la foto la superficie de la cinta, ya que será en el único sitio por donde aparecerán las piezas. Para recortar las fotos lo haremos con un pequeño script de Matlab donde dentro de un bucle que irá recorriendo todas las fotos, lee las fotos una a una, las recorta con unas dimensiones predefinidas, que se corresponden con la superficie de la cinta, y por último renombra la foto con su correcta numeración y la guarda.

Una vez tenemos las fotos recortadas y renombradas por orden, entrenamos los clasificadores. Primero vamos a entrenar el clasificador correspondiente al color. Para ello las clasificaremos de la siguiente manera:

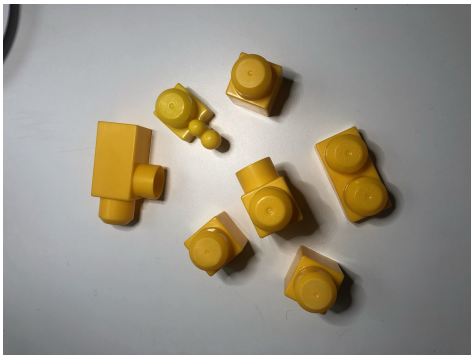


Figura 6.4: Piezas de color amarillo

Figura 6.5: Piezas de color azul

Figura 6.6: Piezas de color naranja

Figura 6.7: Piezas de color rojo

Figura 6.8: Piezas de color verde

Figura 6.9: Piezas de color desconocido

En nuestro caso, con este clasificador, una vez tenemos las fotos clasificadas, hemos elegido la siguiente división de fotos para entrenar, validar y probar este clasificador: un 72% las fotos de entrenamiento, unas 188 imágenes, un 14% (38 imágenes) para validar el clasificador y un 14% (39 imágenes) para testearlo.

Comprobamos la precisión y la matriz de confusión de este clasificador y obtenemos los siguientes resultados

	Class	FP	Precision [%]	FN	Recall [%]	F1-Score [%]	Total	Predicted
1	Amarillo	0	100	0	100	100	7	7
2	Azul	0	100	0	100	100	8	8
3	Naranja	0	100	0	100	100	5	5
4	Rojo	0	100	0	100	100	9	9
5	Verde	0	100	0	100	100	7	7
6	Desconocido	0	100	0	100	100	3	3

Figura 6.10: Parámetros del clasificador de color

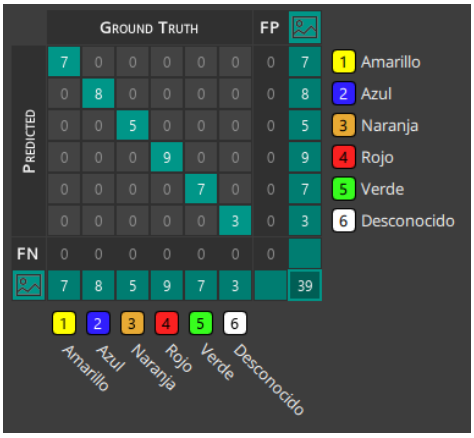


Figura 6.11: Matriz de confusión del clasificador de color

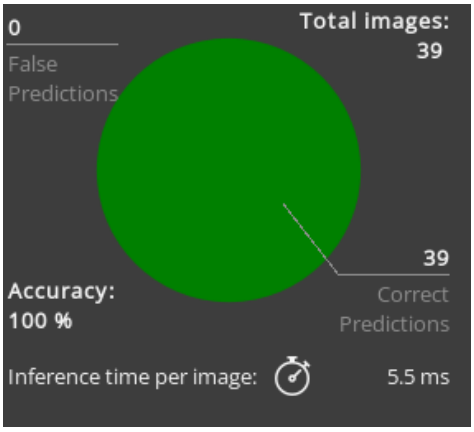


Figura 6.12: Precisión del clasificador de color

Concluimos con que este clasificador es perfecto e ideal para distinguir los colores de las piezas, ya que hemos obtenido un 100% de precisión global, aunque no es la única solución. Por lo tanto, cabe destacar que otro método para identificar los colores de las piezas, hubiera sido ya directamente en Merlic

haciendo un preprocesamiento de la imagen obteniendo los valores de cada canal de los colores RGB y así identificando el color.

Para el otro clasificador, en un principio planteamos la solución de hacer un detector de objetos, que aparte de identificar el tipo de pieza, que es lo que haría un clasificador normal, también te devuelve la posición y orientación de la pieza. Para esto, habría que clasificar todas las fotos, pero ya no solo eso, indicar donde esta la pieza en la foto y decir su orientación. Esta opción que hubiera sido la más adecuada no fue posible, ya que el programa aun no contemplaba esta opción, si no que es un proyecto futuro. Por lo tanto, tuvimos que hacer otro clasificador normal, simplemente clasificando las piezas con el tipo de pieza.

En cuanto a los tipos de piezas, las clasificamos de la siguiente forma, dejando todas las piezas que no utilizamos para la construcción de la figura y que además tienen una forma rara en una clase o tipo llamado desconocidas.

Figura 6.13: Piezas de forma de 1 cuadrado

Figura 6.14: Piezas de forma de 2 cuadrados

Figura 6.15: Piezas de forma de 3 cuadrados

Figura 6.16: Piezas de forma de 4 cuadrados en cuadrado

Figura 6.17: Piezas de forma de 4 cuadrados larga

Figura 6.18: Piezas de forma desconocida

Tomando los mismos valores para la división de las fotos (72% para entrenar el clasificador, 14% para la validación del clasificador y otro 14% para testearlo) y marcando que queremos un análisis mejorado, ya que la clasificación de estas piezas es más compleja que el anterior clasificador del color. Aún así obtenemos un 97,4% de precisión, con los siguientes parámetros y matriz de confusión:

	Class	FP	Precision [%]	FN	Recall [%]	F1-Score [%]	Total	Predicted
1	1	1	90.91	0	100	95.24	10	11
2	2	0	100	0	100	100	10	10
3	3	0	100	0	100	100	1	1
4	4 largo	0	100	0	100	100	4	4
5	4 cuadrado	0	100	0	100	100	4	4
6	Desconocido	0	100	1	88.89	94.12	9	8

Figura 6.19: Parámetros del clasificador de forma

Figura 6.20: Matriz de confusión del clasificador de forma

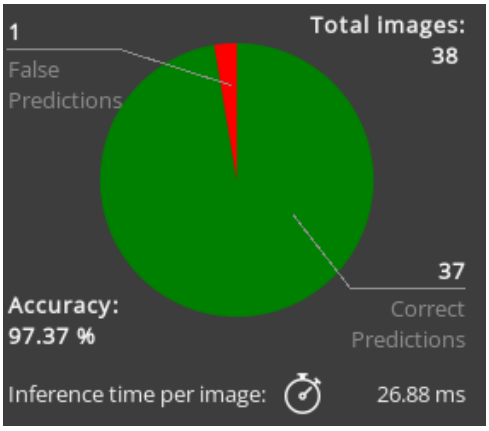


Figura 6.21: Precisión del clasificador de forma

Con este otro clasificador, vemos que no llegamos al 100% de precisión, ya que es más complicado identificar la forma que el color, además también influye la calidad de la imagen, que no es muy buena. Observando más detenidamente la matriz de confusión vemos que donde más errores se producen es con las piezas de un solo cuadrado y las de forma desconocida, probablemente por el pequeño tamaño de las piezas de uno y la variedad de todas las piezas desconocidas. Aun así, podemos concluir con que es un clasificador válida al tener una precisión de un 97,37%

6.3 Desarrollo de la aplicación de visión artificial

El desarrollo de la aplicación de visión artificial lo haremos con Merlic, que ya hemos hablado anteriormente un poco de este programa. En este programa, para crear la aplicación, vas arrastrando los comandos o instrucciones que quieras usar creando una especie de flujograma, desde que adquieres las imágenes tomadas por la cámara, hasta que mandas la información útil al robot. Antes de explicar la estructura del programa creado en Merlic vamos a explicar como hemos calibrado la cámara, mediante unas 5 instrucciones que más abajo se detallan en el esquema.

Para la calibración de la cámara, primeramente haremos la conversión de píxeles a unidades del mundo real, para ello con la instrucción *Determinate Pixel Size* e introduciéndole una distanciada marcada en una imagen es suficiente. Nosotros lo haremos con la siguiente imagen de la izquierda, que le pasaremos a esta instrucción con la instrucción de *Acquire Imagen from File* y con una ROI linear marcaremos el metro indicando que mide unos 940 mm. Para cambiar el sistema de coordenadas, que por defecto Merlic lo sitúa en el centro de la imagen, se hace simplemente con la instrucción *Set Calibration Origin* y con una ROI de punto indicando donde se quiere situar este sistema de coordenadas, como se muestra en la imagen de la derecha. También indicar que para colocar de manera exacta este sistema de coordenadas se tomo de referencia la posición de una pieza.

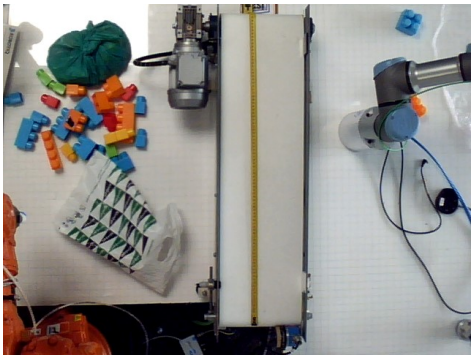


Figura 6.22: Imagen utilizada como medidor para la calibración de la cámara

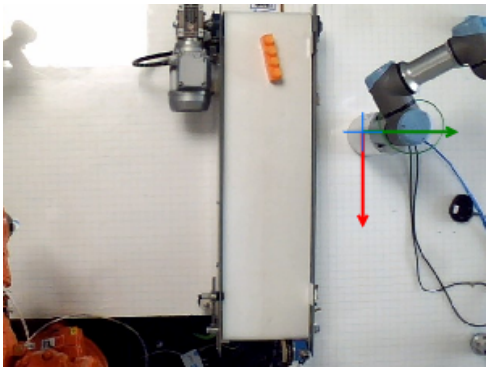


Figura 6.23: Colocación del sistema de coordenadas para la calibración de la cámara

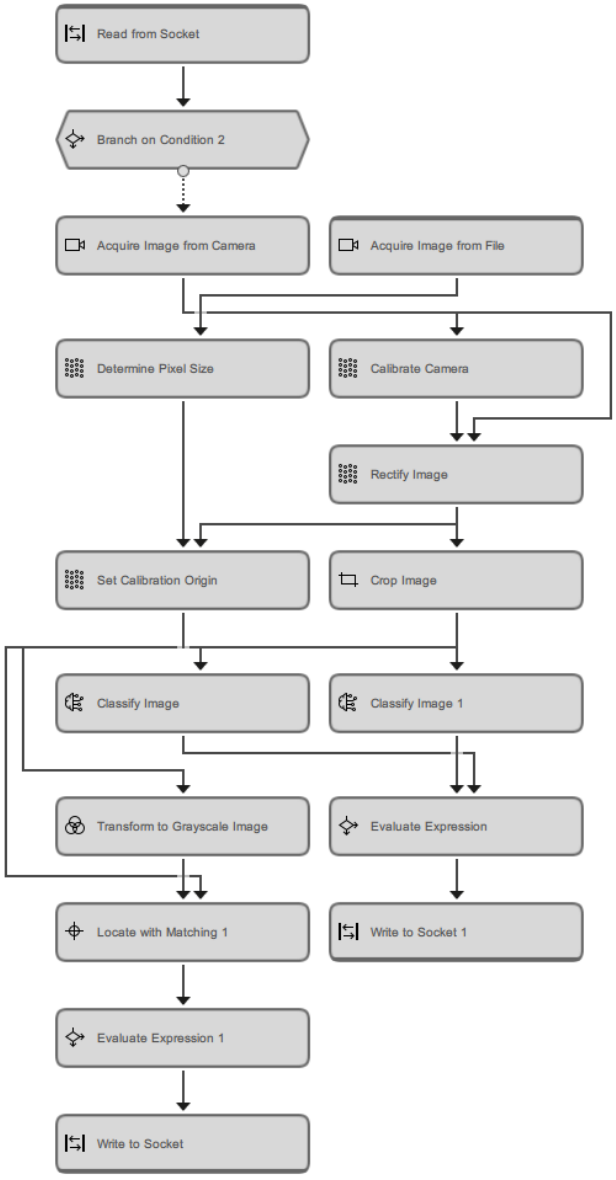


Figura 6.24: Esquema de trabajo de la aplicación Merlic

Para acabar correctamente la calibración de la cámara nos faltaría corregir los fallos de desviaciones por la cámara, para ello entrenamos la instrucción *Calibrate Camera* con una imagen tomada con la cámara de un patrón de calibración (en nuestro caso uno de 160 mm) y la propia instrucción crea unos datos de calibración con los que la instrucción *Rectify Image* rectificará la imagen corrigiendo estas desviaciones y curvaturas.

Una vez explicada la calibración de la cámara, el esquema de trabajo empieza con la lectura del mensaje por socket, para ello la instrucción *Read from Socket* que se configura con la dirección IP del robot e indicando que el puerto es el número 21 y en la siguiente instrucción se comprueba que esta información leída coincida con un 'OK', y entonces será cuando le daremos paso a la imagen de la cámara, que se calibrará como hemos explicado.

A la imagen ya calibrada, antes de aplicarles los clasificadores se recortan con una ROI fija que abarque la zona del sensor del final de la cinta, ya que todas las piezas se pararán ahí para su clasificación. Y a esta imagen donde solo aparece la pieza se le aplican los dos clasificadores, tanto el de color como el de forma, con las instrucciones de *Classify Image* importándole el modelo previamente entrenado con Deep Learning. Estos clasificadores nos devolverán el tipo de pieza codificado, para así facilitar la programación para la comprensión del robot que tipo de pieza es. La codificación utilizada es la mostrada en las siguientes tablas:

CODIFICACIÓN PARA LA FORMA	
Tipo de pieza	**Código**
1	1
2	2
3	3
4 cuadrado	4
4 larga	5
Desconocido	6

Tabla 6.1: Codificación de las piezas según la forma

CODIFICACIÓN PARA EL COLOR	
Tipo de pieza	**Código**
Azul	1
Verde	2
Amarillo	3
Rojo	4
Naranja	5
Desconocido	6

Tabla 6.2: Codificación de las piezas según el color

Por lo tanto el tipo de pieza codificado será el número que junta los dos números de estas tablas, siendo primero el número de la forma y después el color, por ejemplo, una pieza azul de 3 cuadrados será la número 31.

La instrucción que le sigue a la salida de estos clasificadores crea la expresión de este código entre corchetes, que así es como lo requiere el robot, y se manda por socket.

Para a continuación mandar la posición de la pieza, se transforma esta imagen recortada a escala de grises con la instrucción *Transfrom to Grayscale Image*, para que así sea más fácil localizar un único cuadrado para coger la pieza, ya que así todos tendrán un color similar y que se distingue con bastante claridad respecto al color de la cinta transportadora. Y así con la imagen en escala de grises, se le aplica la instrucción *Locate with Matching*, que hay que entrenar con una imagen cualquiera de las piezas válidas, seleccionando con una ROI un único cuadrado por donde el robot pueda coger la pieza.

Una vez esta instrucción nos devuelva la posición, con las coordenadas X e Y y la orientación, creamos otro *Evaluate Expression* donde creamos el vector de la posición de la pieza y posteriormente lo mandamos por socket, después de enviar primeramente el tipo de pieza codificado, ya que el robot esperará esta información en este orden.

6.4 Desarrollo del programa principal del robot

Este al apartado, lo separaremos en otros 4 aparatados, en los que explicaremos la comunicación por socket, el subprograma de la cinta transportadora, y el programa principal del robot de transporte de la pieza y un último apartado donde mostraremos todo el código.

6.4.1 La comunicación por socket

La comunicación por socket es imprescindible en nuestro proyecto, ya que esta comunicación se va a dar en estos 2 casos: el primero será siempre que llegue una pieza desde la cinta transportadora al final de esta cuando el sensor la detecte, que además de pararse la cinta, se mandara un "OK" para que así se active el programa de Merlic, es decir, será una comunicación del robot al ordenador donde se estará ejecutando la aplicación de Merlic. Y el segundo caso de comunicación, será de Merlic al robot, cuando se ejecute todo el programa de Merlic, este por socket mandará toda la información, tanto como el tipo y la posición de esta pieza.

Para ello en la programación del robot, antes de empezar el programa tendremos que establecer esta comunicación, que lo haremos con la instrucción *socket_ open()* que además la repetiremos dentro de un bucle hasta que nos aseguremos que esta conexión se ha producido con éxito.

Los parámetros que requiere esta instrucción es la dirección IP del ordenador y puerto, es decir, los parámetros que definen un socket. Para identificarlos es importante saber de que tipo de socket se trata, si de Stream o Datagram, es decir, por TCP o UDP, en nuestro caso es por TCP (Protocolo de control de trnasmisión). Por lo tanto, para obtener la dirección IP correcta, el ordenador con el que se vaya a comunicar el ordenador tiene que estar conectado a la misma red que el robot, y entonces configurar la comunicación por socket del robot con la dirección IP que la red ha asignado al ordenador. El otro parámetro que es el puerto, siempre será el número 21, ya que es el puerto reservado para la comunicación por FTP basada en web (Protocolo de transferencia de archivos). Con todo esto ya tenemos abierta la comunicación por socket, ahora solo hay que utilizar las instrucciones de leer o mandar sockets para así poder comunicarnos.

Por orden correlativo del programa, primero mandaremos el mensaje de "OK" del robot al ordenador, que para ello utilizaremos la instrucción *socket_ send_ string()*, donde el único parámetro es el mensaje a mandar, en nuestro caso, "OK".

Para el segundo caso de la comunicación por socket, que se da del ordenador al robot, como antes hemos configurado en el programa de Merlic, está comunicación se da en 2 pasos. El primero para que el robot reciba el número que codifica el tipo de pieza y el vector de los 6 dígitos que definen la posición de la pieza. Para recibir la información, es de manera similar que para mandarla, se hace con la instrucción *socket_ascii_float()* e indicándole como parámetro la longitud del vector que va a leer, es decir, para el caso del tipo codificado un 1 y para la posición un 6. Destacar que cuando se leen estos vectores también se almacena generando la posición 0 del vector con la variable que define la longitud del vector. Esta información que nos dice la longitud del vector nos será muy útil para asegurarnos que llegue la información correctamente repitiendo la instrucción de leer el socket en un bucle que no sea cierto hasta que este vector sea de la longitud requerida.

6.4.2 El subprograma de la cinta transportadora

Para mover la cinta transportadora, únicamente hay que activar la salida digital a la que esté conectado el variador de frecuencia que mueve el motor y a su vez, a la cinta transportadora, esta salida digital en nuestro robot es la salida número 1. Para activar y desactivar la salida digital se hace con la instrucción Ajustar.

Con el manejo de la cinta, en este subprograma también tenemos que incluir el sensor de la cinta transportadora y la comunicación por socket explicada en el anterior apartado. El programa empezará con la ejecución de este subprograma, ya que la primera instrucción del programa principal será llamar a este subprograma, y este subprograma lo primero que tendrá que hacer es activar la cinta hasta que la pieza llegue al sensor y entonces, desactivar la cinta y mandar el socket a Merlic. Seguidamente, vienen las instrucciones de esperar la llegada de los sockets de la respuesta de Merlic y entonces también dentro de este subprograma compararemos la información de la pieza que está en la cinta con la información de las piezas buscadas, que esta información estará almacenada en un vector con el orden de piezas correcto que declararemos en el apartado de antes de comenzar el programa, junto con todas las demás inicializaciones de las variables utilizadas, igual que la variable contador de piezas que nos será útil para saber cuantas piezas llevamos ya colocadas y que pieza es la que estamos buscando dentro del vector de piezas válidas.

Para comparar la información enviada por Merlic con la del vector de piezas válidas, lo haremos con un If y si esta información coincide, entonces colocaremos el vector de la posición de la pieza que nos ha devuelto Merlic en un

vector posición (vector que incluye una p antes del vector) mediante un bucle y un contador. También actualizaremos el valor de la variable válida a TRUE, que esta variable será la que dará paso al programa principal del robot para que coja la pieza y la coloque en su sitio, además también actualizaremos el valor del contador de piezas colocadas, sumándole una pieza más.

6.4.3 El programa principal con el movimiento del robot

El programa principal del robot será el encargado del movimiento del robot, para ello antes que eso nos tenemos que asegurar que ya está la pieza válida al final de la cinta transportadora y con la posición que nos ha mandado Merlic, por eso, la primera instrucción será esperar a que la variable válida sea verdadera, y hasta que así no sea se estará llamando al subprograma de la cinta transportadora. Cuando esto así sea se implementaran todos los movimientos que anteriormente en el apartado de los flujogramas hemos visto con el flujograma de la parte del robot.

Estos movimientos consistirán primero en un moveJ a la posición de arriba de la pieza, que será la posición que Merlic nos ha mandado y nosotros en el subprograma de la cinta hemos almacenado en un vector posición, por lo tanto, este moveJ será con un punto de paso variable. El siguiente movimiento será un moveL ya que ya se está dirigiendo a la posición exacta de la pieza, que será un movimiento relativo de unos 200 mm para abajo, aquí será cuando se active la ventosa, con la instrucción Ajustar y marcando la salida digital de la ventosa a nivel alto, y esperaremos un pequeño tiempo para asegurarse que se ha cogido bien la pieza, después otro moveL de movimiento relativo con otros 200 mm hacia arriba, a lo largo del eje Z. Ahora llevamos la pieza a la zona de montaje, par ello repetimos la secuencia de estos movimientos, pero antes tenemos que extraer la posición de la pieza del vector donde se almacena todas las posiciones de las piezas, para ello en una nueva variable cogemos el vector que corresponda con el número de pieza que sea.

Una vez que ya tenemos la posición de donde debe ser colocada la pieza, vamos a esa posición a través de un moveJ también con la posición de la variable que antes hemos creado, que realmente esta posición es la posición de arriba de donde realmente va la pieza, por eso ahora se baja unos 250 mm con un moveL relativo, y una vez que ya está la pieza en su sitio se apaga la ventosa y se espera un pequeño tiempo, seguidamente otro moveL con el punto de paso relativo de otros 250 mm para arriba y ahí es donde se acaba el programa y se vuelve a llamar al subprograma de la cinta transportadora para que sigan

pasando piezas por la cinta transportadora y repetir todo el proceso hasta que se complete la figura de las 10 piezas.

6.4.4 El programa completo

En este último apartado de este extenso capítulo, mostraremos la programación de todo el programa, que en cuanto a programación, se divide en tres subprogramas. El primero de ellos, lo que se ejecuta una única vez antes de empezar el programa, donde están las declaraciones de las variables y vectores, inicializaciones y la apertura de la comunicación. Y los otros dos subprogramas son el programa principal que es lo que se ejecuta a continuación y se repite continuamente y el subprograma de la cinta transportadora que se ejecuta siempre que se ejecute en el programa principal la instrucción de llamar a este subprograma

A continuación se mostrarán estos 3 subprogramas, mostrando así el código completo del robot.

Figura 6.25: Programación subprograma de antes de comenzar

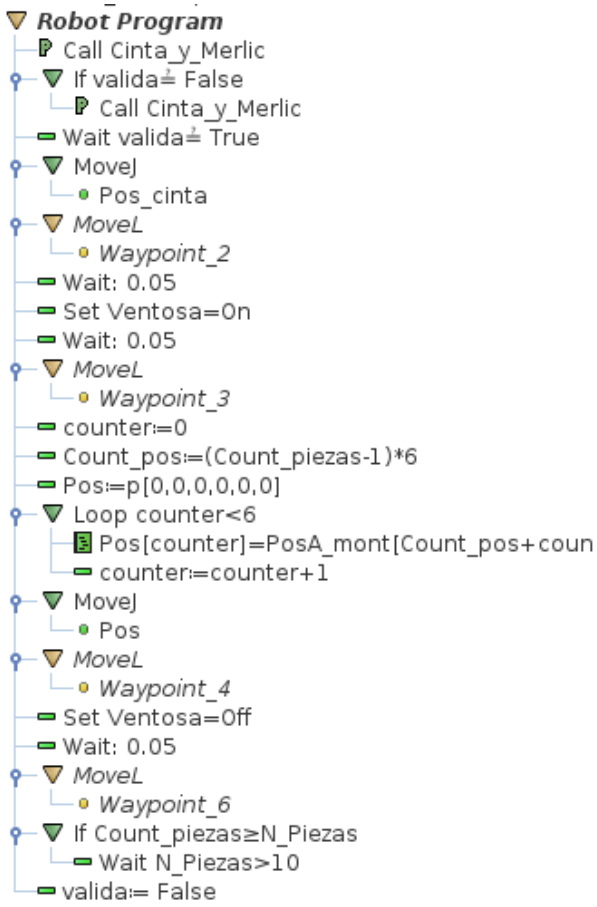


Figura 6.26: Programación del programa principal

Figura 6.27: Programación del subprograma de la cinta transportadora

Para concluir este apartado añadir que en los Anexos se muestra la programación completa en forma de Script.

Capítulo 7

Resultados

Después de la descripción detallada de la solución, en este otro apartado, mostraremos el correcto funcionamiento del robot. Ejecutaremos el programa anteriormente descrito, para mostrar de la manera más gráficamente posible todo el proceso,con todos los casos que se pueden dar, hasta que finalmente el robot monte el puente.

Para mostrar los resultados del funcionamiento del robot, pondremos en robot en marcha he iremos fotografiando todo el flujograma que recorre el programa, a la vez que se comentan todas las fotografías con el proceso a seguir.

También hay que desatacar que para tomar todas estas fotografía el programa se ha modificado la opción de velocidad del robot, ejecutándose el programa a un porcentaje bajo de velocidad respecto a las velocidades configuradas en el programa, así nos ha resultado más fácil poder tomar las fotografías de todos los casos.

En estas dos primeras imágenes en la primera foto, el robot se acaba de iniciar y esta parado esperando que la pieza llegue al sensor con la cinta encendida, y en la segunda foto vemos como cuando al llegar la pieza al sensor la cinta se desactiva:

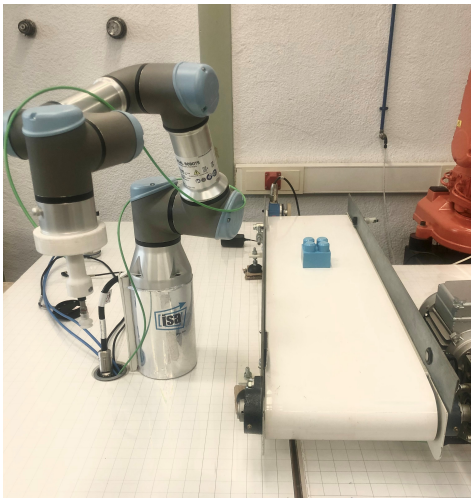


Figura 7.1: Robot esperando la llegada de la pieza al sensor

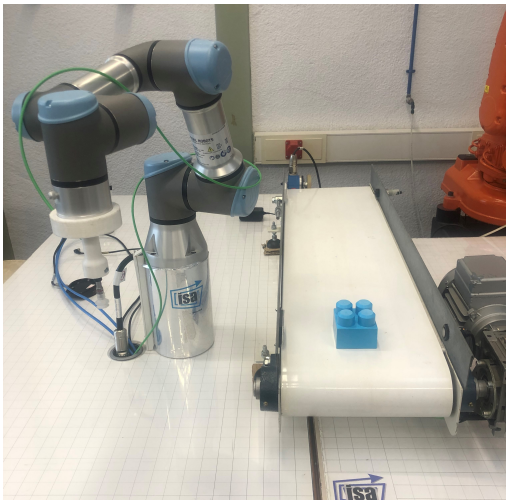


Figura 7.2: La pieza ya ha llegado al sensor

En esta situación el robot manda un socket activando la aplicación de Merlic, donde en estas dos fotos se ve como llega el "OK" en la aplicación de Socket Test que hemos utilizado para probar la comunicación, mientras que en la siguiente imagen se muestra la interfaz de Merlic ejecutándose donde aparece la pieza al final de la cinta transportadora con toda la información que Merlic nos devuelve, como el tipo de pieza, si el sensor esta activo o no y la posición exacta de la pieza.

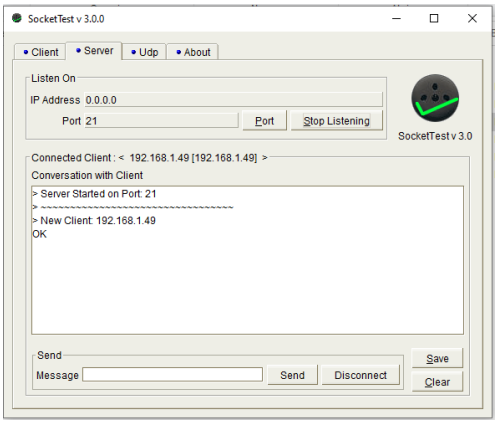


Figura 7.3: Captura de Socket Test cuando llega la pieza al sensor

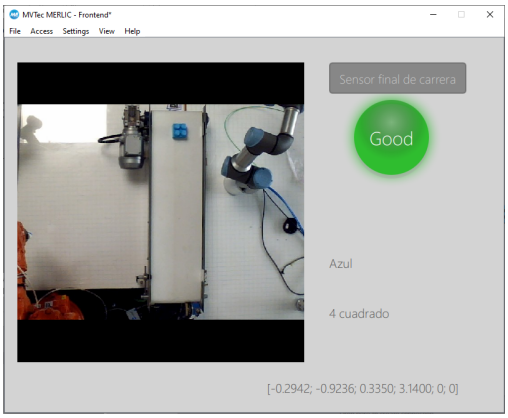


Figura 7.4: Dashboard de Merlic en funcionamiento clasificando la primera pieza

Mientras tanto el robot sigue parado a la espera de que Merlic mande el tipo de pieza y su posición, en este caso la primera pieza es correcta y entonces el robot se activa para coger la pieza, primero va hacia la posición de arriba de la pieza mediante un moveJ y baja hasta ella con un moveL para evitar chocar con la cinta transportadora. En estas dos fotos se muestran estos 2 movimientos:

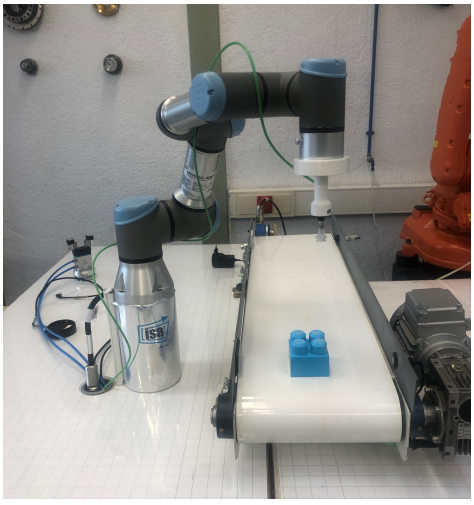


Figura 7.5: MoveJ hasta arriba de la pieza

Figura 7.6: MoveL acercándose a coger la pieza

Una vez el robot esta abajo, se activa la ventosa y coge la pieza, repitiendo la secuencia de los movimientos anteriores, un moveL de unos 200 mm para arriba para levantar la pieza en línea recta para evitar colisiones y un moveJ hasta unos 250 mm arriba de la posición exacta para el montaje de la figura como se muestra en estas dos siguientes fotografías:

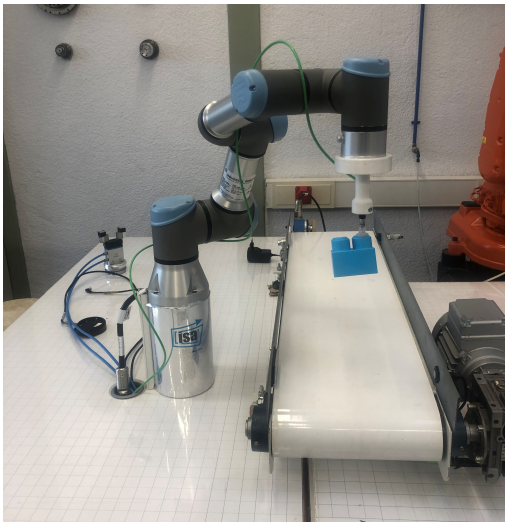


Figura 7.7: MoveL para levantar la pieza

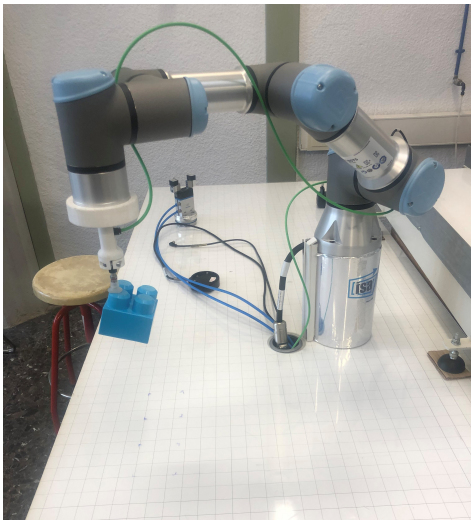


Figura 7.8: MoveJ hasta la zona de montaje

Por último para colocar la primera pieza el robot baja con un moveL estos 250 mm sobre el eje Z y desactiva la ventosa, después de una pequeña espera para asegurarse de que la pieza se suelte de la ventosa vuelve a hacer este moveL hacia arriba hacia un lugar seguro donde esperar para recoger la segunda pieza. En este punto se vuelve a activar la cinta a la espera de que llegue una nueva pieza y se repita todo este ciclo. En estas 2 imágenes se muestran estos 2 últimos movimientos del primer ciclo para dejar la primera pieza:

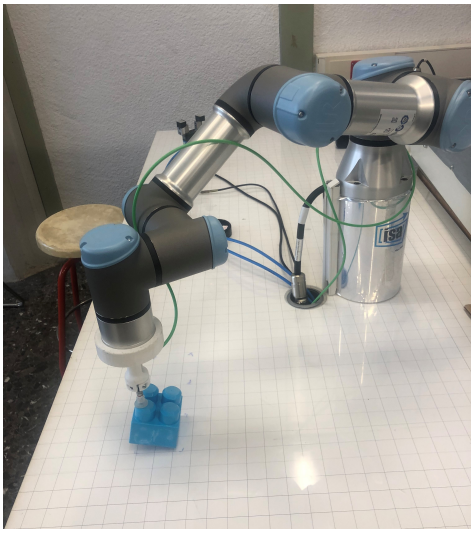


Figura 7.9: MoveL para dejar la pieza en su sitio de montaje

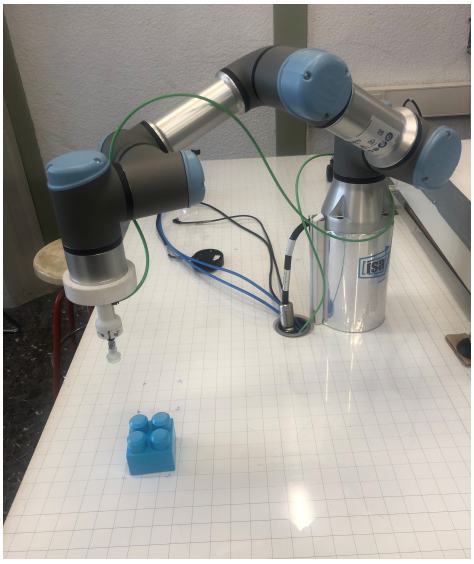


Figura 7.10: MoveL hacia arriba

La segunda correcta es otra pieza igual que la primera, es decir, una pieza de 4 cuadrados en modo cuadrado y de color azul, pero para comprobar que el robot rechaza las piezas inválidas y funciona correcta el sistema de visión artificial, ahora ponemos en la cinta una pieza verde de un cuadrado. Como vemos en la primera foto la interfaz visual de Merlic la identifica y clasifica correctamente y el robot activa la cinta transportadora desechando la pieza como se ve en la segunda imagen:

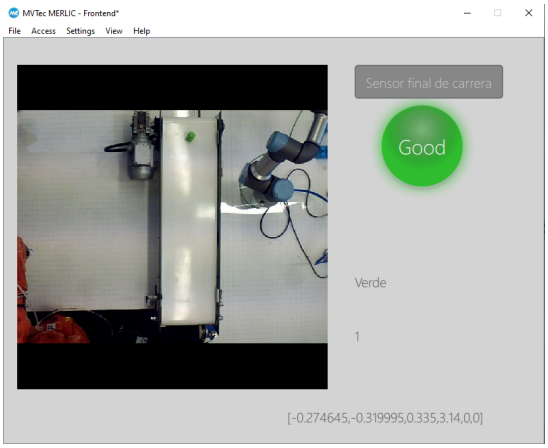


Figura 7.11: Dashboard de Merlic clasificando una pieza inválida

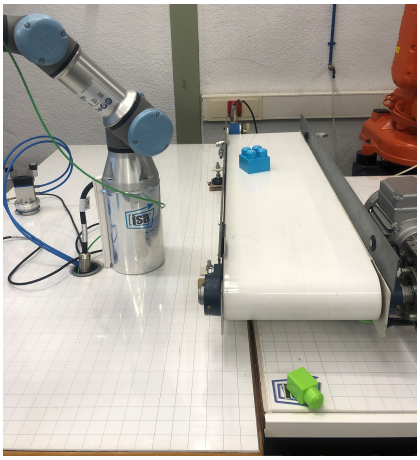


Figura 7.12: El robot desecha esta pieza

Y así repite este ciclo hasta con 10 piezas que completan la figura. Como dato a los resultados añadir que el robot tarda apenas unos 6 segundos en hacer el pick and place completo de una pieza, es decir, desde su posición de espera (desde la posición de arriba de la zona de montaje, o si es su primera pieza, desde la posición en la que se encuentre al iniciarse el robot) hasta que deja a la pieza en su sitio y sube hasta su posición de espera.

Capítulo 8

Conclusiones y dificultades

Con este apartado, acabamos la parte de la memoria del proyecto, la parte en la que hemos descrito todo el proyecto en sí, para acabar esta memoria, las conclusiones. En este último capítulo, analizaremos todo el proyecto, las complicaciones que hemos encontrado y como las hemos solventado, futuras líneas para acabar y mejorar el proyecto y propuestas de cambio para mejorar o simplificar el proyecto.

8.1 Complicaciones y dificultades

A continuación enumeraremos las complicaciones que hemos ido desarrollando a lo largo del proyecto.

Las mayores complicaciones que hemos ido experimentando desarrollando el proyecto se han dado en el laboratorio, como la de conocer el funcionamiento del laboratorio, ya que al estar el robot ya montado y configurado me tocó descubrir cuales eran las conexiones del robot a la ventosa, al sensor o a la cinta transportadora. Dándome la sorpresa de que la cinta transportadora solo estaba conectado al robot industrial ABB que también está en el laboratorio, y por lo tanto, la salida que correspondía a la cinta transportadora de nuestro UR3e estaba conectado al robot ABB siendo una entrada digital del ABB. Así me tocó configurar un simple programa que comprobará el estado de esta entrada digital que provenía del UR3e y según esta entrada digital activara o desactivara la cinta transportadora mediante una salida digital del ABB. Todo

esto aunque ha ido implícito en el proyecto, al ser un caso concreto de este laboratorio y realmente al no pertenecer en el proyecto no se ha considerado en la memoria del proyecto, simplemente se comenta en las conclusiones como una complicación extra del proyecto.

Otra complicación por la utilización de este laboratorio ha sido el tiempo que lo he tenido disponible para mi, que por desgracia, no ha sido mucho, ya que al ser un laboratorio docente no solo estaba a mi disposición y lo he tenido que compartir con otros compañeros mios que estaban también realizando su TFG o TFM. También hay que tener en cuenta de que en agosto son vacaciones y tampoco he podido acceder al laboratorio durante otra semana más, ya que antes de agosto cambiaron el robot de un UR3 que estaba antiguamente por el actual robot UR3e y durante esta semana que lo configuraban no se podía utilizar. Todos estos inconvenientes me han hecho tener que trabajar desde mi casa creando el programa mediante un máquina virtual con la aplicación de PolyScope y llevando los programas al laboratorio para corregirlo y ajustar posiciones.

Otra complicación que se pueda destacar, a parte de los otros muchos problemas que he ido experimentando con la programación o dudas técnicas que he solucionado informándome o haciendo cursos de Universal Robot o Merlic, es que el programa Merlic es de pago y he tenido que utilizar varias licencias de prueba, que no he podido renovar en el mismo ordenador más de 2 veces, ya que hay disponible 2 versiones del programa y estaba el mes de prueba disponible en ambas versiones. Por lo tanto, para no contar con el programa solo 2 meses me han hecho falta dos ordenadores para desarrollar el proyecto.

Por último, otro imprevisto que hicimos frente a él fue el de como coger las piezas con el robot, ya que en un principio el proyecto se planteó hacerlo con piezas Lego, y tuve que buscar unas piezas más grandes como las utilizadas de Megabloks, que a pesar de ser más grandes en principio la ventosa no las cogía bien por el relieve que estas piezas tienen en la parte superior de la pieza, y que por lo tanto, tuve que lijar todas las piezas por la parte superior para así quitarles este relieve.

8.2 Futuras líneas de mejora del proyecto

Hay muchas maneras de mejorar o ampliar el proyecto, como por ejemplo hacer el robot mas autónomo y que no siguiera un orden tan estricto para seleccionar las piezas válidas, ya que conforme se va construyendo la figura no solo hay una pieza válida, si no que se pueden poner varias en el mismo punto de construcción, así ahorrando tiempo si la pieza válida tarda más en salir por la cinta. O por ejemplo, otra medida de mejora que se me ocurre es el haberle incluido medidas de protección al robot con más sensores, para que nadie se pueda acercar a tocar el robot cuando este se esta moviendo para evitar accidentes.

Pero sin duda, la mayor medida que mejoraría el proyecto sería la de sustituir la webcam por una cámara de visión artificial, que nos diera mucha más precisión y más prestaciones. Incluso cambiar también el programa de visión artificial por otro con más prestaciones e instrucciones más robustas y potentes, ya que por ejemplo en muchas utilidades como localizar la posición estaba bastante limitado, al no poder jugar mucho con las pocas instrucciones que tiene y además con limitaciones para poder entrenar estas instrucciones, por eso quizás el programa no funciona con la precisión que debería. Además Merlic en su flujograma no deja crear estructuras con realimentación, por lo tanto no podíamos desarrollar contadores, bucles o estructuras condicionales If/Else. Por este motivo, la aplicación de Merlic simplemente informa al robot del tipo de pieza, sin poder tomar ninguna decisión de Merlic que mejoraría el proyecto, como por ejemplo rechazar alguna pieza desde el dashboard de Merlic, e indicar en el dashboard si la pieza es válida o no, ya que esta es una decisión que toma el robot y no Merlic. Además así se simplificaría bastante el programa del robot, siendo mucho mas simple la información que le tendría que dar Merlic al robot.

Por último, otra opción de mejora que se podría implementar para darle mas funcionalidad al robot, es el poder elegir que figura montar entre varias figuras ya definidas, es decir, que lo primero que tuvieras que seleccionar desde el dashboard de Merlic o a través de otra comunicación sea la figura a construir, por ejemplo, una casa, un cubo o cualquier otra cosa, y que estos datos de estas posiciones por ejemplo estuvieran almacenados en ficheros en internet y que el robot tendría que consultar para construir la figura.

8.3 Conclusiones y vida del proyecto

Como conclusión final, me gustaría reflejar que me hubiera gustado más disfrutar de la ejecución del proyecto, ya que es un proyecto que en un principio me parecía muy interesante y que me motivo mucho poder manejar este robot UR3e y aprender sobre él, ya que este TFG me ha aportado muchísimos conocimientos sobre robótica y visión artificial que seguro que aprovechare mucho en mi futuro como estudiante y laboralmente. Pero si que es cierto, que por algunas de las complicaciones nombradas anteriormente, sobre todo por el poco tiempo que he tenido el laboratorio, me ha hecho tomarme este TFG como un proyecto a contrareloj sin haber podido disfrutar de él. Además también cabe indicar que me hubiera gustado, si el tiempo me lo hubiera permitido, seguir con el proyecto implementado le algunas mejoras y nuevas funciones que también he comentado en el apartado anterior.

También indicar que a pesar de todo el esfuerzo e implicación máxima con el proyecto no se ha logrado que el 100% de las veces que el robot dejaba una pieza del segundo piso se encajara del todo bien, ya que las piezas del primer piso se deslizaban sobre la mesa algunas veces.

Por último, también comentar que este proyecto se trata de un proyecto de máxima actualidad, ya que los robots colaborativos son bastantes actuales y muy revolucionarios respecto a los robots tradicionales, por su fácil instalación y programación, poder trabajar junto a los humanos sin una extrema seguridad, y otros muchos más aspectos técnicos. Además también tiene una amplia vida útil de unos 4 años funcionando las 24 horas del día, aunque claro está que esto se trata de una situación extrema. Otro dato bastante relevante, que aunque no se ha comentado en los presupuestos, es para tener muy en cuenta, es que el promedio para recuperar la inversión del robot UR3e es de solo un año. Todo esto indica, que estos robots serán tan revolucionarios que apuntan a un cambio total en los empleos de los seres humanos.

Además es que este proyecto es muy extrapolable a cualquier otro proceso, prácticamente cualquier proceso automatizado tiene aplicaciones de pick and place con visión artificial, ya sea para clasificar objetos o por ejemplo, en controles de calidad.

Capítulo 9

Presupuesto

Al cerrar la parte de la memoria del proyecto, en la que se almacena la mayor parte de información, pasamos a los presupuestos, aquí calcularemos el precio final de este amplio proyecto. Recopilaremos todo lo utilizado, tanto de material (hardware) como de software, con su precio unitario y lo sumaremos hasta llegar al precio final.

Para el cálculo del precio final del proyecto, lo haremos en estos 3 siguientes apartados

9.1 Precios unitarios

En este primer apartado, enumeraremos todos los materiales y software utilizados con su precio unitario, y tambien incluiremos el precio de la hora del ingenierio que realiza el proyecto.Para ello lo haremos en la siguiente tabla:

Concepto	Precio
Cinta transportadora	120,00 €
Sensor fotoeléctrico E3F1-RP11 de Omron	51,96 €
Variador de frecuencia CFW08 de Weg con armario completo	391,90 €
Motor reductor BWQ40 de Brown Group	562,50 €
WebCam Hércules Deluxe Optical Glass	26,46 €
Bolsa de piezas Mega Bloks	32,99 €
Robt UR3e con PolyScope	22.800,00 €
Ventosa con quick changer de Omron	80,00 €
Software Merlic 5 Small (1 cámara)	195,00 €
Ordenador portatil con conexión a internet	372,00 €
Mano de obra por hora	15,24 €

Tabla 9.1: Precios unitarios

Siendo el precio de la hora de la mano de obra, una media entre todos los trabajos realizados, como el de programador, ingeniero o peón.

9.2 Medición

En este otro apartado, simplemente indicaremos la cantidad utilizada en el proyecto de cada concepto indicado en el aprtado anterior, para ello realizaremos otra tabla:

Concepto	Cantidad
Cinta transportadora	1
Sensor fotoeléctrico E3F1-RP11 de Omron	1
Variador de frecuencia CFW08 de Weg con armario completo	1
Motor reductor BWQ40 de Brown Group	1
WebCam Hércules Deluxe Optical Glass	1
Bolsa de piezas Mega Bloks	1
Robt UR3e con PolyScope	1
Ventosa con quick changer de Omron	1
Software Merlic 5 Small (1 cámara)	1
Ordenador portatil con conexión a internet	1
Mano de obra por hora de ingenierio	360

Tabla 9.2: Medición

En cuanto a la medición cabe destacar las 360 horas que aparecen, que simplemente es una estimación de las horas que he empleado en la ejecución de este proyecto, tanto como en el desarrollo de la aplicación, programación, montaje y redacción de este documento.

9.3 Presupuesto de Ejecución Material (PEM)

Por último, con los precios unitarios y la medición hecha, se concluyen los presupuestos con la siguiente tabla:

Concepto	Precio (€)	Cantidad	Total (€)
Cinta transportadora	120,00	1	120,00
Sensor fotoeléctrico E3F1-RP11 de Omron	51,96	1	51,96
Variador de frecuencia CFW08 de Weg	391,90	1	391,90
Motor reductor BWQ40 de Brown Group	562,50	1	562,50
WebCam Hércules Deluxe Optical Glass	26,46	1	26,46
Bolsa de piezas Mega Bloks	32,99	1	32,99
Robt UR3e con PolyScope	22.800,00	1	22.800,00
Ventosa con quick changer de Omron	80,00	1	80,00
Software Merlic 5 Small (1 cámara)	195,00	1	195,00
Ordenador portatil con conexión a internet	372,00	1	372,00
Mano de obra por hora de ingenierio	15,24	360	5.486,40
		TOTAL	**30.119,21**

Tabla 9.3: Presupuesto

Por lo tanto, la cantidad final del presupuesto de este proyecto asciende a **treinta mil ciento diecinueve con veintiuna centésimas**

Capítulo 10

Pliego de condiciones

Este otro documento, aparte de la memoria, tiene como objetivo reflejar todas las condiciones a cumplir entre los posibles contratantes de este proyecto, estas condiciones serán las generales, las económicas y las de ejecución, Además, se citará toda la normativa vigente con la que cumple el robot UR3e y un breve manual de usuario para la seguridad del cliente.

10.1 Condiciones generales

Este proyecto, y más este documento, es de carácter obligado cumplimiento una vez se haya firmado y estén de acuerdo ambas partes en su ejecución, y en el caso de que hubiera modificaciones, es requisito indispensable que sean aprobadas por ambas partes. Todos los materiales, equipos y software utilizados serán los nombrados y presupuestados en el proyecto, y que deben estar en perfectas condiciones y cumpliendo la normativa vigente

El cliente tendrá una garantía de un año tras recibir, montar e instalar todos los equipos. Garantía que incluye el reemplazo de del elemento o equipo dañado si la causa de este desperfecto fuera un defecto de fábrica. Por lo tanto, esta garantía no incluirá los daños ocasionados a los equipos que fueran producidos por un mal uso de ellos.

El contratista debe de facilitar toda la información necesaria para su correcta instalación, como lugar de trabajo, condiciones, horarios, etc. La instalación

será llevada a cabo por personal capacitado para ello, y que una vez instalado se realizarán todas las pruebas pertinentes para comprobar su correcto funcionamiento. Además estas pruebas se deberán de realizar bajo la supervisión del contratista para que ambas partes estén de acuerdo de que la instalación ha sido llevada a cabo de forma adecua cada y sin ningún problema.

10.2 Condiciones económicas

Queda claro que el precio aparecido en el presupuesto es un precio fijo, sin lugar de descuentos ni negociaciones del mismo. El pago se podrá realizar por transferencia bancaria en dos pagos, un primer pago del 75% del presupuesto al firmar el contrato y la adjudicación del proyecto y otro pago del 25% restante en 15 días hábiles tras el montaje del equipo.

En caso de incumplimiento de estos pagos en los plazos establecidos, se cobrará un cargo extra de un 10% sobre el total del presupuesto. También en caso de que no se cumplan los plazos establecidos de montaje e instalación de todos los equipos, el cliente recibirá una compensación de mutuo acuerdo.

El coste de las averías o daños causados de cualquier elemento de la instalación una vez cumplida la garantía repercutirán sobre el cliente. Además el mantenimiento o cualquier modificación del mismo también supondrá un coste al cliente dependiendo de las horas y materiales empleados en ello.

10.3 Condiciones de ejecución

Una vez todos los equipos instalados y con la aplicación en marcha con un correcto funcionamiento, todo funciona como se detalla en este proyecto, para cualquier duda recurrir a él como manual de usuario. Para poner en marcha el funcionamiento se deberá de seguir una serie de instrucciones de puesta en marcha que se detallan a continuación:

- Primeramente habrá que encender el robot UR3e desde el botón superior de la consola y el variador de frecuencia que da señal al motor de la cinta transportadora accionando el interruptor del cuadro eléctrico de este variador.
- Para encender la parte de visión artificial, una vez encendido el ordenador y abierto el archivo de programa de Merlic, habrá que asegurarse que el ordenador este conectado a la misma red a la que esta conectado el robot,

ya sea por cable o conexión Wi-Fi, y este conectado exclusivamente a esta red, ya que si esta conectado a dos redes a la vez no se asignarán correctamente las IP y no se establecerá la comunicación entre ambos dispositivos.

- La cámara también debe estar conectada a este ordenador por cable USB, que esta cámara ya estará instalada y configurada al software de Merlic. Una vez aseguradas todas estas conexiones, se ejecuta el programa a través del dashboard o pantalla gráfica de Merlic, para así una correcta visualización de la cámara con toda la información extraída de Merlic.

- Una vez todo encendido se deberá activar la movilidad del robot en el botón ON de la consola del robot, que habrá que pulsar este botón hasta 2 veces, para así desbloquear los frenos neumáticos que impiden moverse al robot cuando esta apagado.

- Seguidamente habrá que cargar este programa y su archivo de instalación, donde estarán ya todas las entradas y salidas configuradas, igual que la herramienta.

- Antes de ejecutar el programa, chequear que se cumplen todas las medidas citadas en el manual de seguridad que se adjunta en este mismo documento en el punto 10.6

- Por último ejecutar el programa dándole al botón de play de la consola, e indicando la opción de ejecutar el programa desde el principio.

10.4 Normativa

El robot colaborativo usado UR3e cumple con toda la normativa vigente requerida, que a continuación se enumera la más importante, y para ver toda la normativa explicada con más detalle hay que recurrir al manual de usuario del UR3e que sale reflejado en las referencia bibliográficas al final de todo el proyecto:

- **ISO 13849-1:2015** Estipula que el control de seguridad esté diseñado como nivel de rendimiento de acuerdo con las demás normas que rigen el control de seguridad.

- **ISO 13850:2015** Obliga a que la parada de emergencia sea una parada de categoría 1, que según esta norma una parada de categoría 1 es una parada

controlada que se consigue manteniendo la alimentación de los motores hasta conseguir el paro, que es cuando se desconecta la alimentación

- **ISO 12100:2010** Norma de principios generales de diseño con la que se evalúan todos los robots UR
- **ISO 10218-1:2011** Trata sobre la instalación y el diseño de la aplicación robótica
- **IEC61000-6-2:2005** Este conjunto de normas definen los requisitos relativos a las perturbaciones eléctricas y electromagnéticas y garantiza que estos robots tengan buenos resultados en entornos industriales sin perturbar el funcionamiento de otros equipos.
- **IEC 61326-3-1:2008** Define requisitos adicionales sobre inmunidad electromagnética para funciones relacionadas con la seguridad.
- **IEC 61131-2:2007** Normativa que las entradas y salidas del UR3e cumplen con los requisitos, tanto las E/S de 24 V, las normales o las de seguridad, para así garantizar una comunicación fiable con sistemas PLC
- **ISO 14118:2000** Principios de seguridad para evitar el arranque inesperado por diversas causas posibles.
- **IEC 60947-5-5/A1:2005** Mas normativa sobre el botón de parada de emergencia
- **IEC 60529:2013** Esta normativa define requisitos de la carcasa del robot, para protección frente al agua o el polvo y cumplen con un grado de protección IP
- **IIEC 60320-1/A1:2007** El cable de alimentación cumple esta norma.
- **ISO 9404-1:2004** Rige el ajuste del tipo 50-4-M6 que cumple la brida de la herramienta y todas sus herramientas deben cumplir para garantizar un buen ajuste
- **ISO 13732-1:2006** Define el máximo de la temperatura superficial del robot según sus límites ergonómicos definidos en esta norma.
- **IEC 61140/A1:2004** Los robots están constituidos según esta norma para la protección contra las descargas eléctricas, con una conexión tierra/masa obligatoria.

- **IEC 60068-2-1:2007** Definen los métodos de prueba con los que se prueban estos robots
- **IEC 61784-3:2010** Define requisitos de para buses de comunicaciones de seguridad
- **IEC 60204-1/A1:2008** Norma de equipamiento eléctrico
- **IEC 60664-1:2007** Normativa que cumplen los circuitos eléctricos del UR3e

10.5 Manual de usuario de seguridad

Debes tener en cuenta que deberías de leer este breve manual de usuario antes de utilizar esta aplicación y tenerlo a mano para consultarlo cuando sea necesario. En este manual se explicarán. La información incluida en este manual no debe considerarse una garantía de que el manipulador de la aplicación no causará daños, aunque se cumplan todas las instrucciones de seguridad.

Las advertencias detalladas a continuación pueden causar situaciones peligrosas, que si no se evitan, podrían causar lesiones o daños importantes en el equipo. Estos son los símbolos de advertencia que engloban las siguientes advertencias:

Figura 10.1: Señal de advertencia de peligro en general

Figura 10.2: Señal de advertencia por superficies calientes

- Asegúrese de que el brazo robótico y herramienta estén atornillados de manera correcta y segura en su lugar.
- Asegúrese de que el brazo robótico tenga espacio suficiente para funcionar libremente.
- No lleve ropa holgada ni joyas cuando trabaje con el robot. Asegúrese de recogerse el pelo si lo tiene largo cuando trabaje con el robot.
- Nuca utilice el robot si está dañado, por ejemplo, si los tapones de la junta están sueltos, rotos o se han retirado.
- Si el software indica que se ha producido un error, active inmediatamente la parada de emergencia, anote los códigos de error mostrados en la pantalla y póngase en contacto con el proveedor.
- Tenga cuidado con el movimiento del robot cuando utilice la consola portátil.
- Nunca modifique el robot. Las modificaciones podrían crear peligros imprevistos por el integrador.
- Asegúrese de que los usuarios del robot están informados de la ubicación del o los botones de parada de emergencia
- El robot y su caja de controlador generan calor durante su funcionamiento. No manipule ni toque el robot mientras esté en funcionamiento o inmediatamente después de su funcionamiento dado que el contacto prolongado puede causar malestar.
- No introduzca nunca los dedos tras la cubierta interna de la caja del controlador.

El robot cuando con una seta de emergencia en la consola que actúa de botón de parada de seguridad, este botón debe pulsarse inmediatamente cuando se detecte algún fallo o anomalías que puedan dañar a alguien o al equipo. Una vez corregido el fallo, hay que rearmar esta seta de emergencia y volver a activar el robot. El robot también cuenta detrás con otro botón, el botón de movimiento libre, que manteniéndolo pulsado puedes cambiar la posición del robot de forma manual.

Para iniciar la aplicación se debe hacer uso de los consejos e instrucciones que se han detallado en las condiciones de ejecución e indicar que el contratista no se hará responsable de los problemas ocasionados por no seguir estas normas.

Capítulo 11

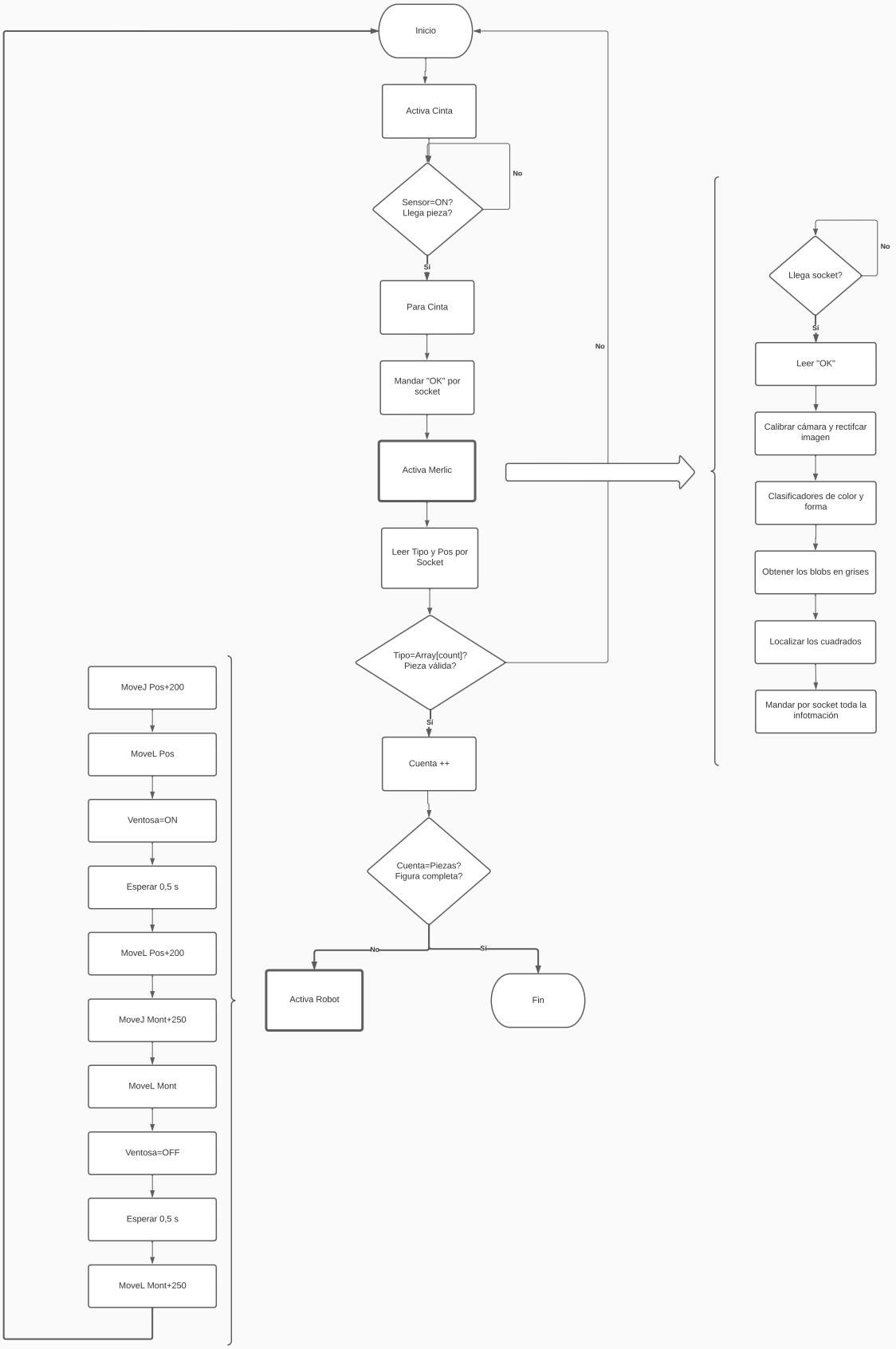
Planos

En esta última parte o documento antes de los Anexos, acabamos con este proyecto, aquí adjuntaremos los planos más relevantes utilizados para la correcta ejecución del proyecto.

Estos son los 2 planos que se adjuntan al proyecto:

11.1 Flujograma completo de todo el proyecto

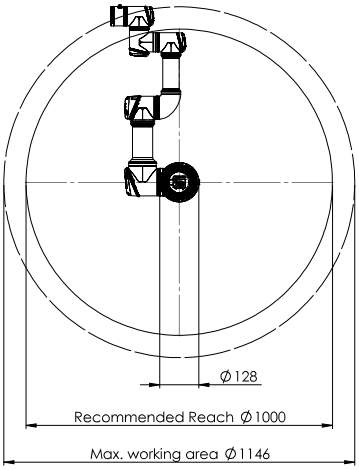
En la siguiente página se muestra el flujograma completo, que en el desarrollo del proyecto ya se ha mostrado por partes y explicado cada una de estas partes.


Inicio
Activa Cinta
Sensor=ON?
Llega pieza?
No
Sí
Para Cinta
Mandar "OK" por socket
Activa Merlic
Leer Tipo y Pos por Socket
Tipo=Array[count]?
Pieza válida?
No
Sí
Cuenta ++
Cuenta=Piezas?
Figura completa?
No
Sí
Activa Robot
Fin
Llega socket?
No
Sí
Leer "OK"
Calibrar cámara y rectifcar imagen
Clasificadores de color y forma
Obtener los blobs en grises
Localizar los cuadrados
Mandar por socket toda la infotmación
MoveJ Pos+200
MoveL Pos
Ventosa=ON
Esperar 0,5 s
MoveL Pos+200
MoveJ Mont+250
MoveL Mont
Ventosa=OFF
Esperar 0,5 s
MoveL Mont+250

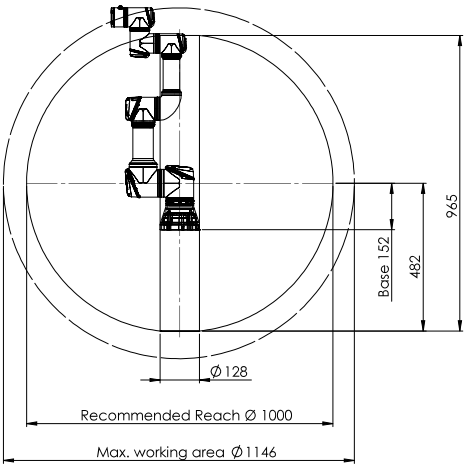
11.2 Planos del UR3e y su área de trabajo

En estos planos, primeramente verémos el área de trabajo del robot, tanto desde una vista superior y frontal o de lado. Mientras que en la siguiente página se muestra un plano del dimensionado del robot UR3e con el que hemos trabajado.

UR3 working area, top view



UR3 working area, side view

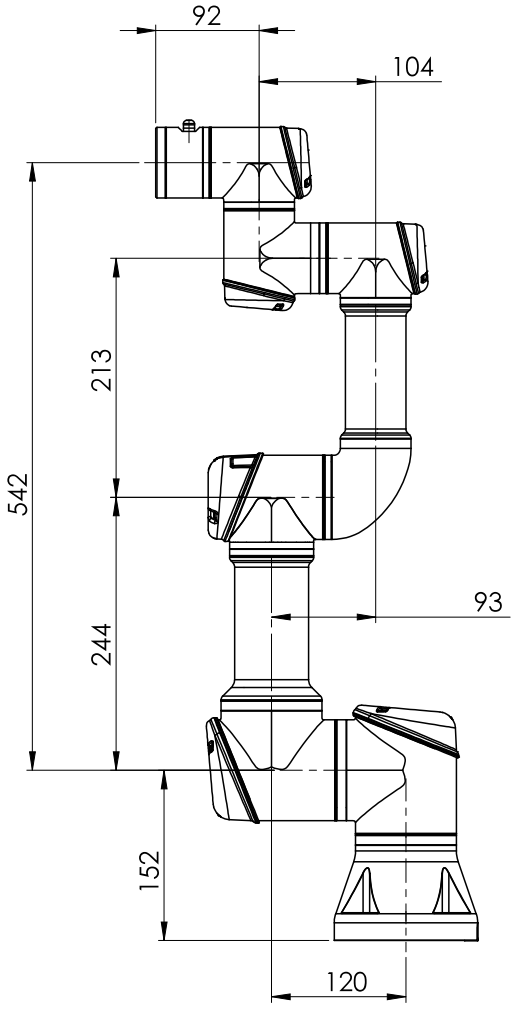


All dimension is in mm
For public use

UNIVERSAL ROBOTS

TEL: +45 89 93 89 89 FAX: +45 38 79 89 89 WEB: universal-robots.com

TITLE:	UR3e Working Area
DATE	28-05-2018
Status change date:	
DWG NO.	1000697
REV.	0



All dimension is in mm
For public use

UNIVERSAL ROBOTS

TEL: +45 89 93 89 89 FAX: +45 38 79 89 89 WEB: universal-robots.com

TITLE:	UR3e Working Area
DATE	28-05-2018
Status change date:	
DWG NO.	1000697
REV.	0

Capítulo 12

Bibliografía

Referencia fundamental para el desarrollo del proyecto en general:

- Manual de usuario del UR3e y PolyScope

Referencias para desarrollar el apartado de Antecedentes:

- Historia de la visión artificial. Infaimon
- Visión artificial. Wikipedia
- UR3. Página oficial de UR
- Robótica. Wikipedia
- Robótica colaborativa. Pagina oficial de UR
- Historia de UR. Página oficial de UR

Otra referencias de utilidad:

- Objetivos de Desarrollo Sostenible

Capítulo 13

Anexos

13.1 Programación de Matlab para recortar las fotos

Este código se utilizó para recortar las fotos todas por el mismo sitio para poder entrenar el clasificador y además renombrarlas.

```
%Script para recortar las fotos en la parte central

for c=1:266 %Recorremos todas las fotos, hasta la 266
    cs=string(c) %Pasamos el número de la foto a string
    pieza='Pieza (' +cs +').jpg' %Construimos el nombre de
    %la foto
    I=imread(pieza); %Leemos la foto
    %imshow(I) %Mostramos la foto
    %Ir=imcrop(I)
    Ir= imcrop(I, [242.5 6.5 127 363]); %Recortamos la foto
    imshow(Ir) %Mostramos la foto recortada
    cont= sprintf('%04d', c) %Numeración con 4 cifras
    cont=string(cont)
    nombre='Pieza' + cont + '.jpg' %Nombre de la foto.
    %Pieza0000
    imwrite (Ir, nombre); %Guardamos la foto
end
```

13.2 Programación del robot

Programación extraída de la consola del robot UR3e que se puede usar como script.

```
def P1pieza():
  global _hidden_verificationVariable=0
  step_count_103ffb1b_abdc_41d1_aa63_bca773b79a0b = 0.0
  thread Step_Counter_Thread_8bdfb6aa_b25b_4ee6_96d1_b12cd7a1f813():
    while (True):
      step_count_103ffb1b_abdc_41d1_aa63_bca773b79a0b =
step_count_103ffb1b_abdc_41d1_aa63_bca773b79a0b + 1.0
      sync()
    end
  end
  run Step_Counter_Thread_8bdfb6aa_b25b_4ee6_96d1_b12cd7a1f813()
  set_safety_mode_transition_hardness(0)
  set_gravity([0.0, 0.0, 9.82])
  set_tcp(p[0.0,0.0,0.0,0.0,0.0,0.0])
  set_target_payload(0.200000, [0.000000, 0.000000, 0.000000], [0.000000, 0.000000,
0.000000, 0.000000, 0.000000, 0.000000])
  set_standard_analog_input_domain(0, 1)
  set_standard_analog_input_domain(1, 1)
  set_tool_analog_input_domain(0, 0)
  set_tool_analog_input_domain(1, 0)
  set_analog_outputdomain(0, 0)
  set_analog_outputdomain(1, 0)
  set_input_actions_to_default()
  set_tool_communication(False, 115200, 0, 1, 1.5, 3.5)
  set_tool_output_mode(0)
  set_tool_digital_output_mode(0, 1)
  set_tool_digital_output_mode(1, 1)
  set_tool_voltage(0)
  global Waypoint_2_from_p=p[-.311915633251, -.288886998752, .335994011243, -
2.864225139027, 1.290249725193, -.000118690983]
  global Waypoint_2_to_p=p[-.311925624220, -.288871992443, .135995924902, -
2.864156038790, 1.290236483418, -.000062639249]
  global Waypoint_3_from_p=p[-.311921167785, -.288876407525, .136001631725, -
2.864165471647, 1.290237405384, -.000041580474]
  global Waypoint_3_to_p=p[-.311905493503, -.288873680801, .335984567501, -
2.864119893402, 1.290103685376, -.000191411093]
  global Waypoint_4_from_p=p[-.338357616777, .363420096612, .221989289230, -
2.864131736497, 1.290227942961, -.000189205803]
  global Waypoint_4_to_p=p[-.338366326744, .363421301462, -.027998978395, -
2.864224991822, 1.290224719269, -.000058373079]
  global Waypoint_6_from_p=p[-.338367054991, .363425116273, -.027990549591, -
2.864182455592, 1.290182611849, -.000027938497]
  global Waypoint_6_to_p=p[-.338373771834, .363417627503, .221990133832, -
2.864239205739, 1.290158616573, -.000023940679]
  def Cinta_y_Merlic():
    $ 48 "Cinta_y_Merlic" "noBreak"
    $ 49 "Set Cinta=On"
    set_standard_digital_out(1, True)
    $ 50 "Wait Sensor=HI"
    while (get_standard_digital_in(6) == False):
      sync()
    end
    $ 51 "Set Cinta=Off"
```

```
set_standard_digital_out(1, False)
$ 52 "sendToServer:='OK'"
global sendToServer="OK"
$ 53 "socket_send_string('OK')"
socket_send_string("OK")
$ 54 "Wait: 0.1"
sleep(0.1)
$ 55 "recibido_tipo:=socket_read_ascii_float(1)"
global recibido_tipo=socket_read_ascii_float(1)
$ 56 "Loop recibido_tipo[0]≠1"
while (recibido_tipo[0] != 1):
 $ 57 "Wait: 0.03"
 sleep(0.03)
 $ 58 "recibido_tipo:=socket_read_ascii_float(1)"
 global recibido_tipo=socket_read_ascii_float(1)
end
$ 59 "recibido:=socket_read_ascii_float(6)"
global recibido=socket_read_ascii_float(6)
$ 60 "Loop recibido[0]≠6"
while (recibido[0] != 6):
 $ 61 "Wait: 0.03"
 sleep(0.03)
 $ 62 "recibido:=socket_read_ascii_float(6)"
 global recibido=socket_read_ascii_float(6)
end
$ 63 "If recibido_tipo[1]≟Piezas_val[Count_piezas]"
if (recibido_tipo[1] == Piezas_val[Count_piezas]):
 $ 64 "counter_1:=1"
 global counter_1=1
 $ 65 "Pos_cinta:=p[0,0,0,0,0,0]"
 global Pos_cinta=p[0,0,0,0,0,0]
 $ 66 "Loop counter_1<7"
 while (counter_1<7):
  $ 67 "Pos_cinta[counter_1-1]=recibido[counter_1]"
  Pos_cinta[counter_1-1]=recibido[counter_1]
  $ 68 "counter_1:=counter_1+1"
  global counter_1=counter_1+1
 end
 $ 69 "Count_piezas:=Count_piezas+1"
 global Count_piezas=Count_piezas+1
 $ 70 "valida:= True "
 global valida=  True
else:
 $ 71 "Else" "noBreak"
 $ 72 "Set Cinta=On"
 set_standard_digital_out(1, True)
 $ 73 "Wait Sensor=LO"
 while (get_standard_digital_in(6) == True):
  sync()
 end
end
$ 74 "sync()"
sync()
```

```
 end
 $ 1 "BeforeStart"
 $ 2 "open:=socket_open('192.168.1.103',21)"
 global open=socket_open("192.168.1.103",21)
 $ 3 "Loop open≟ False "
 while (open ==   False  ):
  $ 4 "open:=socket_open('192.168.1.103',21)"
  global open=socket_open("192.168.1.103",21)
 end
 $ 5 "PosA_mont:=[-0.329,0.388,0.221,2.246,-2.195,-0.02,-0.205,0.382,0.22,2.284,-2.157,0,-
0.305,0.363,0.256,-3.114,0.416,0,-0.305,0.393,0.256,3.114,0.416,0]"
 global PosA_mont=[-0.329,0.388,0.221,2.246,-2.195,-0.02,-0.205,0.382,0.22,2.284,-2.157,0,-
0.305,0.363,0.256,-3.114,0.416,0,-0.305,0.393,0.256,3.114,0.416,0]
 $ 6 "Piezas_val:=[41,41,55,55,25,25,12,13,12,13]"
 global Piezas_val=[41,41,55,55,25,25,12,13,12,13]
 $ 7 "Count_piezas:=0"
 global Count_piezas=0
 $ 8 "N_Piezas:=10"
 global N_Piezas=10
 $ 9 "Set Cinta=Off"
 set_standard_digital_out(1, False)
 $ 10 "Set Ventosa=Off"
 set_standard_digital_out(5, False)
 $ 11 "valida:= False "
 global valida=  False
 $ 12 "counter:=0"
 global counter=0
 $ 13 "Count_piezas:=0"
 global Count_piezas=0
 $ 14 "counter_1:=1"
 global counter_1=1
 $ 15 "Pos:=p[0,0,0,0,0,0]"
 global Pos=p[0,0,0,0,0,0]
 $ 16 "Pos_cinta:=p[0,0,0,0,0,0]"
 global Pos_cinta=p[0,0,0,0,0,0]
 while (True):
  $ 17 "Robot Program"
  $ 18 "Call Cinta_y_Merlic"
  Cinta_y_Merlic()
  $ 19 "If valida≟ False "
  if (valida ==   False  ):
   $ 20 "Call Cinta_y_Merlic"
   Cinta_y_Merlic()
  end
  $ 21 "Wait valida≟ True "
  while (not(valida ==   True  )):
   sync()
  end
  $ 22 "MoveJ"
  $ 23 "Pos_cinta" "breakAfter"
  movej(Pos_cinta, a=1.3962634015954636, v=1.0471975511965976)
  $ 24 "MoveL"
  $ 25 "Waypoint_2" "breakAfter"
```

```
    movel(pose_add(get_target_tcp_pose(), pose_sub(Waypoint_2_to_p,
  Waypoint_2_from_p)), a=1.2, v=0.25)
    $ 26 "Wait: 0.05"
    sleep(0.05)
    $ 27 "Set Ventosa=On"
    set_standard_digital_out(5, True)
    $ 28 "Wait: 0.05"
    sleep(0.05)
    $ 29 "MoveL"
    $ 30 "Waypoint_3" "breakAfter"
    movel(pose_add(get_target_tcp_pose(), pose_sub(Waypoint_3_to_p,
  Waypoint_3_from_p)), a=1.2, v=0.25)
    $ 31 "counter:=0"
    global counter=0
    $ 32 "Count_pos:=(Count_piezas-1)*6"
    global Count_pos=(Count_piezas-1)*6
    $ 33 "Pos:=p[0,0,0,0,0,0]"
    global Pos=p[0,0,0,0,0,0]
    $ 34 "Loop counter<6"
    while (counter<6):
      $ 35 "Pos[counter]=PosA_mont[Count_pos+counter]"
      Pos[counter]=PosA_mont[Count_pos+counter]
      $ 36 "counter:=counter+1"
      global counter=counter+1
    end
    $ 37 "MoveJ"
    $ 38 "Pos" "breakAfter"
    movej(Pos, a=1.3962634015954636, v=1.0471975511965976)
    $ 39 "MoveL"
    $ 40 "Waypoint_4" "breakAfter"
    movel(pose_add(get_target_tcp_pose(), pose_sub(Waypoint_4_to_p,
  Waypoint_4_from_p)), a=1.2, v=0.25)
    $ 41 "Set Ventosa=Off"
    set_standard_digital_out(5, False)
    $ 42 "Wait: 0.05"
    sleep(0.05)
    $ 43 "MoveL"
    $ 44 "Waypoint_6" "breakAfter"
    movel(pose_add(get_target_tcp_pose(), pose_sub(Waypoint_6_to_p,
  Waypoint_6_from_p)), a=1.2, v=0.25)
    $ 45 "If Count_piezas≥N_Piezas"
    if (Count_piezas >= N_Piezas):
      $ 46 "Wait N_Piezas>10"
      while (not(N_Piezas>10)):
        sync()
      end
    end
    $ 47 "valida:= False "
    global valida=  False
  end
end
```