Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

SIMULACIÓN DE UNA CÉLULA ROBOTIZADA, CONTROLADA POR UN AUTÓMATA PROGRAMABLE Y GESTIONADA CON UNA INTERFAZ HOMBRE-MÁQUINA, MEDIANTE COMUNICACIONES OPC.

Autor:

Martínez Rodríguez, Alberto

Tutor:

Herreros López, Alberto

Departamento de Ingeniería de Sistemas y Automática

Valladolid, junio de 2022

Agradecimientos

Me gustaría comenzar este proyecto agradeciendo a mi tutor D. Alberto Herreros López, por la confianza y paciencia depositada en mi para la realización de un proyecto de estas características.

A Leoni Systems Spain, por darme la oportunidad de trabajar con ellos y complementar mi formación como profesional.

A mis compañeros y amigos con los que he vivido esta aventura y con los que he pasado unos de los mejores años de mi vida.

Especialmente a mi familia, por apoyarme siempre de forma incondicional en todo lo que haga, en los momentos buenos y sobre todo en los malos.

Resumen

En este Trabajo de Fin de Grado se pretende diseñar y desarrollar una instalación robotizada, controlada por un autómata programable y gestionada a nivel de usuario con una interfaz hombre máquina, mediante la tecnología de comunicaciones industriales OPC.

Se pretende que este proyecto pueda ser aprovechado con fines docentes para la realización de futuros proyectos, así como para el estudio conjunto de las diferentes ramas de la ingeniería que abarca este trabajo.

Para el desarrollo de la estación se ha utilizado el programa de simulación Roboguide proporcionado por la marca FANUC, así como los robots de diferentes gamas de su catálogo, y complementado por el software Autodesk Inventor para el diseño de elementos que componen la célula. El control de la instalación se ha asignado a un PLC SIEMENS S7-1200, simulado desde la herramienta PLCSIM y programado desde TIA Portal. Por último, se añade una interfaz hombre máquina para la gestión y supervisión de la planta, diseñado desde cero en MATLAB.

Palabras Clave

Roboguide (FANUC), TIA Portal (SIEMENS), OPC (KEPServerEX), MATLAB, Automatización.

Abstract

At this final Degree Project, the aim is to design and develop a robotic work cell, controlled by a programmable logic controller and managed at the user level with a human machine interface, using OPC industrial communications technology.

It is intended that this Project can be used for teaching purposes for the realization of future projects, as well as for the joint study of the different fields of engineering that this work covers.

For the development of the process plant, Roboguide, the simulation program provided by the FANUC brand, has been used, as well as the robots from different sections of its catalogue. This simulation program is complemented by the Autodesk Inventor software for the design of the elements that make up the cell. In adition, the control of this work cell has been assigned to a SIEMENS S7-1200 PLC, simulated from PLCSIM tool and programmed with TIA Portal software. Finally, the human machine user interface, which has been designed from scratch in MATLAB, is added to achieve the purpose of management and supervisión of the process plant.

Keywords

Roboguide (FANUC), TIA Portal (SIEMENS), OPC (KEPServerEX), MATLAB, Automation.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

Capítulo 1.1. Motivación, origen y justificación del proyecto

Durante el transcurso de mi etapa académica, el campo de la robótica es el que mayor interés ha despertado en mí. A lo largo de estos últimos años, he adquirido mucha confianza en la realización de proyectos. Así mismo he tomado conciencia de la gran importancia que tiene la robótica en los entornos industriales y la automatización de procesos.

La afición que fui adquiriendo hacia la robótica, me llevo a tomar la decisión de cursar las asignaturas de *prácticas en empresa* y *ampliación de prácticas en empresa* en *Leoni Systems Spain S.L.U*, donde se me enseño como, a nivel laboral, empezar a dominar y desarrollar la versatilidad de trabajar con las principales marcas de robots industriales como FANUC, ABB o KUKA, así como el saber desenvolverme con autómatas programables (PLC) de los principales referentes en la industria como son SIEMENS, ALLEN BRADLEY o el mismo ABB.

Las razones anteriormente expuestas fueron determinantes a la hora de escoger esta temática para la realización de este proyecto, puesto que se ha considerado que uno de los principales objetivos es demostrar las habilidades adquiridas durante el grado, en diferentes ramas de la ingeniería, como son la robótica, el control, la electrónica y la computación entre otras.

Este trabajo surge a raíz de una de las tareas llevadas a cabo durante el período de prácticas, donde trabajando con estaciones robotizadas con un gran número de robots se planteaba la necesidad de intercambiar señales digitales entre los robots, con el objetivo de poder coordinar la célula robotizada que se pretendía desarrollar.

Partiendo de esta necesidad, se plantea como solución añadir un PLC que se encargue de la coordinación entre robots trabajando con las señales digitales necesarias para esta gestión.

La problemática surge a la hora de trabajar con diferentes entornos de simulación, ya que no siempre va a existir una compatibilidad nativa en el software proporcionado por las distintas marcas, pese a que en el entorno de trabajo real la mayoría de los dispositivos implementan protocolos compatibles y existe la posibilidad de trabajar en conjunto con los productos proporcionados por diferentes proveedores.

Por lo tanto, durante el informe relativo al presente proyecto, se van a desarrollar y analizar los mecanismos llevados a cabo para la creación de una célula robotizada en un entorno simulado que trabaje de forma análoga a como la misma célula robotizada lo haría en un entorno industrial real.

Capítulo 1.2. Objetivos

A continuación, se establecen unos objetivos básicos, para delimitar así el proyecto que se va a realizar y sentar unas bases generales por las que se va a regir el trabajo.

Este proyecto ha sido concebido desde un inicio con fines académicos, siendo el principal objetivo proporcionar las herramientas que posibiliten el estudio de células robotizadas en entornos de simulación, trabajando de forma simultánea con software suministrado por diferentes marcas haciendo uso de un estándar de comunicaciones común para todos ellos.

Teniendo en cuenta el objetivo principal, se han establecido una serie de objetivos intermedios que dan cuerpo al proyecto desarrollado, verificando su utilidad como herramienta a nivel académico:

- Estudio y diseño de una célula robotizada, comprendida por una serie de robots que trabajen en un entorno de simulación.
- Diseño de las piezas necesarias en software de diseño 3D de forma que se pueda aportar el mayor realismo posible a la estación.
- Configuración y programación de un autómata que pueda ser simulado y permita la coordinación de la estación de trabajo.
- Diseño y programación de una interfaz de usuario que permita el manejo y control de la instalación de forma dinámica.
- Estudio de un protocolo de comunicaciones que posibilite la comunicación entre los diferentes softwares de simulación.

Capítulo 1.3. Fundamentos previos

Para la elaboración de este proyecto, han sido necesarios una serie de conceptos y conocimientos que se han ido adquiriendo en diferentes asignaturas impartidas durante el grado, así como en las prácticas en empresa, que abarcan los aspectos de la automatización, la robótica y el control, visión artificial y comunicaciones industriales.

Adicionalmente, el uso de diferente software, como Roboguide, TIA Portal o MATLAB, requieren de una formación básica y un estudio autónomo de los mismos que permitan explorar y sacar partido de sus funcionalidades correctamente.

Capítulo 1.4. Alcance del proyecto

Tal y como se ha indicado en previamente, este proyecto comprende un estudio, desarrollo e implementación de una célula robótica haciendo uso de software de simulación de distintos proveedores que no proporcionan un método de conexión directa nativa entre ellos.

Para ello se hará uso de software de terceros, desarrollando así una herramienta de alta utilidad a nivel didáctico. El uso de un software de terceros para la coordinación de las instancias de simulación suministradas por los proveedores de los dispositivos que se van a utilizar nos permite experimentar en un entorno simulado como sería el comportamiento conjunto de estos dispositivos en un entorno real.

Capítulo 1.5. Metodología

Durante el desarrollo del informe, los lectores se van a encontrar con una introducción teórica a las diferentes ramas y principales aspectos que engloba la temática escogida para el proyecto.

Una vez definido el marco teórico y establecido un contexto, se analizará más en profundidad la problemática a la que se pretende dar solución, con el objetivo de proporcionar una serie de soluciones factibles entre las que se encontrará la solución adoptada. Para ello se recurrirá a proyectos pasados realizados en la Universidad de Valladolid para observar las soluciones aportadas por otros autores, con el fin de dar un punto de vista nuevo entorno a la resolución del problema.

Tras definir un marco teórico con las diferentes ramas de conocimiento que abarca el proyecto, se realiza una breve explicación de los diferentes softwares que se han utilizado para que este se pueda llevar a cabo.

A continuación, tras dejar claro que se pretende hacer y cómo trabajar con los programas que se van a utilizar, se realizará un estudio entorno a la normativa vigente para el diseño de instalaciones de este tipo. Seguidamente se explicará el desarrollo del proyecto en sí, hasta llegar a exponer los resultados obtenidos.

Finalmente, para completar este informe, se realizará un análisis económico y se expondrán las principales conclusiones que se han ido extrayendo durante el transcurso del proyecto, dejando una vía abierta a futuros trabajos que puedan llevarse a cabo para completar este proyecto o continuar con el trabajo en otra dirección. Las referencias bibliográficas utilizadas, así como los anexos incluidos se encuentran al final del informe.

Capítulo 1.6. Convenciones

Con el objeto de facilitar la lectura de este informe, se exponen a continuación una serie de normativas de marcado que se han seguido y se encuentran durante los siguientes capítulos.

- Aquellas palabras extranjeras que no han sido traducidas son entrecomilladas y remarcadas en cursiva.
- Las citas que hagan referencia a obras externas se encuentras entrecomilladas y/o referenciadas.
- Seguido al índice general, se encuentra una serie de índices específicos, para figuras y tablas de forma independiente.

CAPITULO 2. MARCO TEÓRICO

Capítulo 2.1. Automatización

La automatización a pequeña escala ha existido durante muchos años, desde el uso de mecanismos simples para desarrollar tareas sencillas de manufactura. Pero este concepto comenzó a ser verdaderamente práctico con la evolución y adición de las computadoras digitales, que permitieron aumentar la flexibilidad para el manejo de cualquier clase de tarea.

Especialmente durante el siglo XX, tras la primera revolución industrial a finales del siglo XVII, cuyo núcleo se centró en la mecanización industrial, comenzó a desarrollarse una diferenciación con lo que se llamó recientemente la segunda revolución industrial, que se centraba en los entornos de automatización de procesos en la industrial. Esto conllevó que las empresas tuvieran que adoptar diferentes políticas para adaptarse a los cambios que esto supuso.

Capítulo 2.1.1. Automatización Industrial

En el contexto actual, la automatización industrial consiste en el uso de distintos sistemas o elementos computarizados, electromecánicos, electroneumáticos y electrohidráulicos con fines industriales.

La tendencia seguida durante las últimas décadas persigue el objetivo de automatizar de forma progresiva todo tipo de proceso productivo, un modelo viable gracias al desarrollo y el abaratamiento de la tecnología de la que se hace uso.

La automatización de procesos de producción propone una serie de objetivos que se han convertido en indispensables para mantener la competitividad:

- Mejora de la calidad.
- Mantener un nivel de calidad uniforme, disminuyendo el número de producción defectuosa.
- Producir una cantidad indicada (necesaria) en el momento preciso.
- Reducir los costes y mejorar la productividad.
- Aumento de la seguridad y precisión del proceso.
- Aumentar la flexibilidad del sistema productivo, facilitando cuando se necesario los cambios en la producción.

Por lo tanto, el aumento de los niveles de automatización en los procesos se ha convertido cada vez más en una necesidad y es indispensable para poder sobrevivir en el mercado actual.

Dentro de la automatización de un proceso productivo, podemos distinguir diferentes niveles de automatización:

1. **Nivel de máquina.** Se considera dentro de este nivel la automatización de una maquina encargada de realizar una tarea simple determinada.
2. **Nivel de célula (o grupo).** Hace referencia al control automatizado de un grupo maquinas que de forma conjunta y coordinada trabajan en la realización de un proceso de producción más complejo.
3. **Nivel de planta.** Se considera en este nivel la automatización de toda la planta de producción que trabaje de forma coordinada para lograr los objetivos de producción global de una fábrica.
4. **Nivel de empresa.** En este nivel se considera la empresa como conjunto (ventas, gestión, producción…).

Capítulo 2.1.2. Definición de automatismo

Se definirá un sistema automatizado, ya se trate tanto de una maquina como de un proceso, como aquel que sea capad de reaccionar de forma automática ante las alteraciones que se produzcan en el mismo, realizando las acciones pertinentes para cumplimentar aquellas funciones para las que ha sido diseñado.

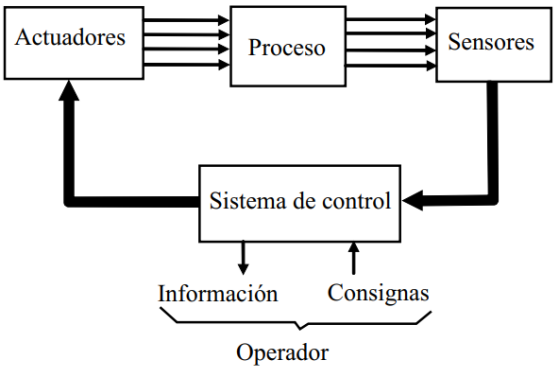


Figura 1. *Estructura de un sistema automatizado.*

La denominación de automatismo se aplica al sistema completo mostrado en la *Figura 1*, aunque el termino suele hacer referencia, fundamentalmente, al sistema de control, ya que es el encargado de producir de forma automática las acciones sobre el proceso partiendo de la información suministrada por los sensores.

Atendiendo a la tecnología empleada para implementar el sistema de control podemos distinguir entre los siguientes tipos de automatismos:

- **Automatismos cableados.** Implementados por medio de conexiones físicas entre los elementos que forman el sistema de control. La estructura del conexionado da lugar a la función lógica que define la señal de salida en función de la entrada.
- **Automatismos programables.** Se implementan mediante un programa, que es ejecutado en un microprocesador, cuyas instrucciones determinan la función lógica que relaciona la salida con las entradas. Existen tres principales formas de implementación:
  - Autómata programable industrial (PLC). Se trata de un equipo electrónico programable, diseñado para controlar en tiempo real procesos secuenciales en ambientes industriales. Hoy en día es el más utilizado para control de máquinas y procesos.
  - Ordenador (PC industrial). Son ordenadores cuyo hardware está diseñado especialmente para ser robusto en entornos industriales.
  - Microcontroladores. Se trata de circuitos integrados programables, comprendidos por un microprocesador, memoria y los periféricos necesarios. Son utilizados mayoritariamente en sistemas de control para máquinas de las cuales se van a fabricar muchas unidades, de forma que se justifique el mayor coste y dificultad del diseño.

Capítulo 2.1.3. Autómata programable industrial

El autómata programable industrial (API), también conocido como PLC (programmable logic controller), es el cerebro de la automatización industrial. Se trata a grandes rasgos de un computador especial, tanto a nivel de software como de hardware.

De acuerdo con la definición más extendida que nos proporciona la NEMA, un PLC es:

[2] "Un dispositivo electrónico operado digitalmente, que usa una memoria programable para el almacenamiento interno de instrucciones para implementar funciones específicas, tales como lógica, secuenciación, registro y control de tiempos, conteo y operaciones aritméticas para controlar, a través de módulos de entrada/salidas digitales (ON/OFF) o analógicos (1-5 VDC, 4-20 mA, etc.), varios tipos de máquinas o procesos".

Atendiendo a la arquitectura interna de un PLC industrial típico, se puede desarrollar esquemáticamente como se detalla en la siguiente figura:

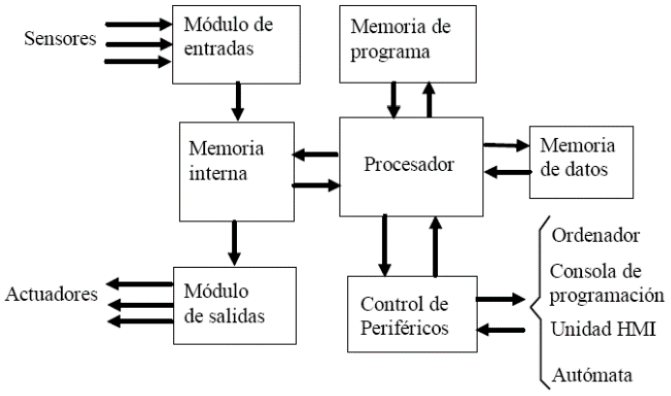


Figura 2. *Arquitectura funcional de un PLC.*

- **Procesador.** Es el encargado de ejecutar en interpretar las instrucciones que hay en el programa almacenado en la memoria de programa, así como monitorizar el estado de las entradas del sistema por medio de los sensores.
- **Bloques de memoria.** Los PLC cuentan con diferentes tipos de memoria entre los que se encuentran memorias de tipo RAM, memorias ROM o EEPROM. A continuación, se exponen los diferentes bloques de memoria que pueden encontrarse en los autómatas programables industriales atendiendo a una clasificación funcional:
  - Memoria de programa. Este módulo contiene el programa con las instrucciones que son ejecutadas por el procesador. Se divide en dos partes, una parte ROM (fija) que contiene el programa monitor encargado de comunicar el autómata con los módulos de programación y una parte RAM con batería (mantiene los datos, aunque se apague la alimentación) donde se almacena el programa de usuario.
  - Memoria interna. Encargada de almacenar los valores de las variables internas del autómata, así como los valores de las entradas y salidas. Aunque son organizadas en bytes, cada variable es de 1 solo bit, por lo que se puede acceder a ellas de forma independiente.
  - Memoria de datos. Almacena los datos de configuración y/o parámetros necesarios para el funcionamiento del autómata y del proceso, así como también contiene datos de propósito general.
- **Módulos Entradas/Salidas.** Son el nexo de conexión entre el autómata y el proceso industrial que se pretende controlar. Permiten la conexión directa con los actuadores del proceso, siendo lo más común referirse a señales de 12V a 24V.

Respecto a la construcción física de los PLC, la mayoría predominante en el mercado son de tipo modular, es decir, estos están formados por una cantidad de módulos que pueden interconectarse entre sí tal y como se muestra en la siguiente figura:

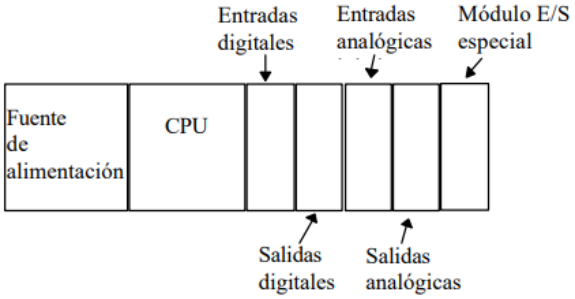


Figura 3. *Esquema a nivel físico de un autómata modular.*

El módulo principal lo constituye la CPU, que contiene tanto el procesador como los diferentes bloques que forman la memoria, así como ciertos controladores de periféricos.

La fuente de alimentación es modulo encargado de suministrar la tensión necesaria para la alimentación de todos los módulos que forman el equipo. CPU y fuente de alimentación forman la estructura mínima necesaria. Los módulos adicionades son conectados al módulo de CPU por medio del bus.

El API se trata de un computador, por lo tanto, a nivel funcional su objetivo consiste en la ejecución de un determinado programa de los que se encuentran en la memoria de programas. Se pueden distinguir a su vez dos modos de funcionamiento: modo de programación y modo de ejecución.

- **Modo de programación.** En este modo el programa monitor se encarga de la comunicación del PLC con el elemento de programación con el fin de traspasar el programa seleccionado del cual se desea ejecutar el algoritmo de control. En este modo de funcionamiento el API no se encuentra controlando el proceso.
- **Modo de ejecución (modo RUN).** En este modo, se ejecuta el programa que implementa el algoritmo de control, pasando así el autómata a controlar el proceso. Al inicializar el autómata en este modo, el programa monitor salta a la dirección donde se encuentra el programa de control para comenzar su ejecución.

La ejecución en modo RUN se realiza de forma cíclica, ejecutando ciclos de ‘scan’ (entendiendo por ‘scan’, o ciclo de trabajo, al grupo de tareas que un autómata ejecuta en un ciclo para llevar a cabo el control del proceso) de forma indefinida. El ciclo de trabajo más común tiene la siguiente estructura:

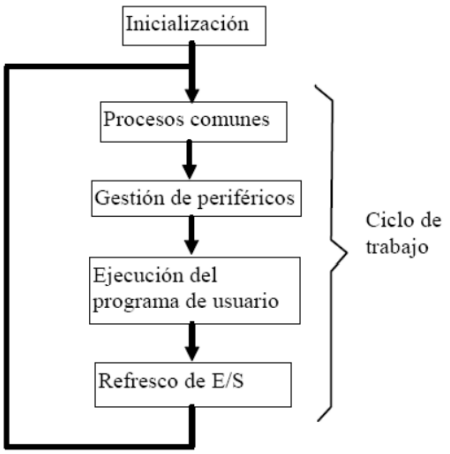


Figura 4. *Esquema a nivel físico de un autómata modular.*

Capítulo 2.1.4. Unidades de programación

Las unidades de programación son aquellos periféricos usados para trabajar con el autómata. Dichas unidades nos permiten cargar el programa de control del proceso en la memoria del PLC. Las más utilizadas son: Consola de programación y ordenador personal.

- **Consola de programación.** Es un dispositivo específico para la programación del autómata. Se comunica con el autómata a través de un puerto serie o de un puerto específico. Resulta útil para pequeñas modificaciones, pero engorroso para introducir programas completos a través de su consola.
- **Ordenador personal (PC).** Es el elemento más común para programar PLCSim, conectado normalmente a través de un puerto serie. Cabe destacar que cada fabricante dispone de su propio software para edición, depuración, compilación y carga del programa en el autómata.
- **Unidades de interfaz hombre-máquina (HMI).** Son dispositivos que, más que para programación al uso, se utilizan con el fin de controlar el proceso o introducir consignas (operaciones tales como marcha, paro, cambio de modo de funcionamiento, etc.). Deben previamente programadas por medio de un PC, una vez realizada su programación, se comunican con el PLC a través del puerto serie.

Capítulo 2.2. Robótica

Durante años, el ser humano ha diseñado y construido diferente maquinas que replicara su comportamiento o incluso partes de su cuerpo. Desde los antiguos egipcios que dotaron de brazos mecánicos en sus estatuas de los dioses, los cuales eran operados por sacerdotes, hasta las construcciones griegas de estatuas que operaban mediante sistemas hidráulicos.

El término 'robot' tiene sus orígenes en la palabra 'robota', procedente del checo, cuyo significado es trabajo forzado o servidumbre. Sin embargo, el origen etimológico de la palabra se remonta a 1920, cuando el dramaturgo de origen Checo Karel Čapek presenta su obra Rossum's Universal Robots (R.U.R). En esta obra aparece la figura de un robot como una herramienta diseñada por una empresa para aligerar la carga de trabajo que recae sobre los humanos.

No fue hasta 1942, cuando se empezó a popularizar el término robótica, acuñado por el escritor Isaac Asimov tras su relato ficticio Runaround, donde se exponen las tres leyes de la robótica:

- **Primera Ley.** Un robot no hará daño a un humano, ni permitirá por inacción que un ser humano sufra algún daño.
- **Segunda Ley.** Un robot debe cumplir las órdenes que un ser humano le dé, a excepción de que dicha orden entre en conflicto con la primera ley.
- **Tercera Ley.** Un robot debe proteger su propia integridad en la medida en que esta protección no entre en conflicto con la primera o segunda ley.

Capítulo 2.2.1. Robótica Industrial

Una vez establecido un precedente histórico que nos lleva al origen de la palabra robot, conviene centrarse en su importancia a nivel de la industria.

En el año 1954 el inventor George Charles Devol desarrollo un primitivo brazo programable que podía ser utilizado para realizar tareas específicas. Este brazo artificial multi-articulado denominado 'Unimate' es considerado el primer robot industrial. Devol fundaría Unimation en 1960, junto a Josep Engelber, empresa dedicada al desarrollo de máquinas con enfoque puramente industrial. Estos dos acontecimientos suponen el origen de la robótica industrial.

El primer robot 'Unimate' fue instalado unos años más tarde, durante la década de los 60, en la empresa 'General Motors' para realizar la carga y descarga de piezas de una máquina de fundición por inyección.

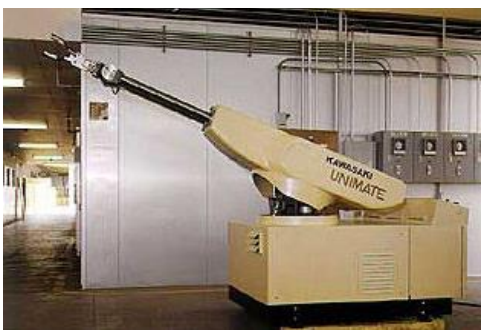


Figura 5. *Unimate robot.*

En 1975 Victor Scheinman, ingeniero mecánico estadounidense, llevó a cabo el desarrollo de un manipulador polivalente conocido como 'brazo manipulador universal programable' (*PUMA*). Este manipulador era capaz de mover un objeto a una posición determinada y colocarlo en cualquier orientación siempre y cuando este punto estuviera dentro de su alcance. El robot *PUMA* es el concepto básico de brazo multi-articulado y en él se basan la gran mayoría de los robots industriales actuales.

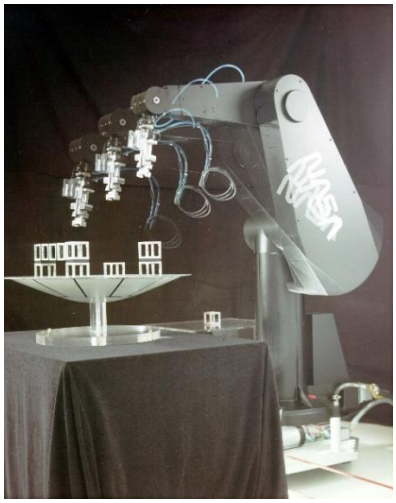


Figura 6. *Robot PUMA.*

Pese al desarrollo y expansión de la robótica en la industria, no existe una definición universal como tal, siendo por lo tanto las diferentes organizaciones nacionales e internacionales quienes imponen sus propias definiciones del concepto de robot industrial. A continuación, se exponen las dos definiciones más comúnmente aceptadas.

- Según la Asociación de Industrias de Robótica (RIA): [8] "Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas, o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas".
- La Organización Internacional de Estándares (ISO) define un robot industrial como: [8] "Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas".

Independientemente de la definición, es común la aceptación de robot industrial como un brazo mecánico que tiene capacidad de manipular objetos y que incorpora un control de mayor o menor complejidad, incluyendo el principio de secuenciación y retroalimentación que permita proporcionar movimientos rápidos y precisos.

La evolución de los robots basados en microprocesadores, así como el empleo de servos en bucle cerrado que permiten establecer con exactitud la posición real de los elementos del robot dan origen a una serie de robot que se detallan a continuación.

- **Manipuladores**. Se trata de sistemas mecánicos multifuncionales, con un sistema de control sencillo, que permiten realizar óptimamente multitud de operaciones básicas, gobernando sus movimientos de forma manual, en secuencia fija o en secuencia variable.
- **Robots de repetición o aprendizaje**. Son robots manipuladores que se limitan a la repetición de una secuencia de movimientos que han sido previamente ejecutados por un operador humano, mediante un controlador manual o un dispositivo auxiliar.
- **Robots de control por computador**. Son robots manipuladores o sistemas mecánicos multifuncionales controlador por un microordenador.
- **Robots inteligentes**. Similares al grupo anterior, con la capacidad además de relacionarse con el mundo que les rodea y tomar decisiones en tiempo real.
- **Micro-Robots.** Utilizados con labor educativa, de entretenimiento o investigación, se trata de robots de formación de precio asequible, cuyo funcionamiento y estructura es similar a los usados en aplicaciones industriales.

Capítulo 2.2.2. Robots manipuladores industriales

Una vez analizada la definición de robot industrial, vamos a centrarnos en la serie de robots manipuladores. Para ello, previamente se deben conocer una serie de características que nos permitan clasificarlos en diferentes grupos.

- **Grados de libertad:** Es la suma de las articulaciones que permiten un movimiento diferente al proporcionado por articulación anterior. Cuanto mayor sea el número de grados de libertad, mayor será su flexibilidad.
- **Área de trabajo:** Región del espacio que es alcanzable por el robot en alguna de sus configuraciones. Esta limitada por las características físicas del robot.
- **Capacidad de carga:** Máximo peso en carga que un robot es capaz de manipular a una velocidad determinada considerando la configuración menos favorable.
- **Capacidad de posicionamiento:** Mide la precisión (grado de exactitud) entre el punto alcanzado en un movimiento y el punto real deseado.
- **Resolución:** Desplazamiento mínimo que es capaz de realizar el robot.

- **Repetibilidad:** Capacidad de alcanzar el mismo punto destino desde el mismo punto inicial de forma sucesiva sin variaciones.

Atendiendo a las características indicadas anteriormente, podemos realizar una clasificación de los diferentes tipos de robots manipuladores desde el punto de vista industrial.

- **Robot cartesiano.** Esencialmente este tipo de robot se caracteriza por su posicionamiento mediante 3 articulaciones de tipo lineal, generando así movimientos perpendiculares entre sí en los 3 ejes cartesianos (X, Y, Z).

Figura 7. *Robot cartesiano.*

- **Robot SCARA.** Este tipo de robot se desplaza en los mismos planos que los robots cartesianos, incorporando al final del brazo robótico (plano Z) un eje que permita girar la herramienta. Este tipo de robot sacrifica flexibilidad en pro de la velocidad dentro de su rango de acción con el fin de satisfacer las tareas que demandan bajos tiempos de ciclo (normalmente '*pick&place*').

Figura 8. *Robot SCARA.*

- **Robot Delta.** Robot paralelo formado por una base fija y una móvil unidas por cadenas cinemáticas seriales. Son utilizados para manipular cargas que requieren movimientos rápidos y precisos. No existen ecuaciones que modelen su configuración, por lo que se debe diseñar un modelo de la dinámica para cada robot.

Figura 9. *Robot Delta.*

- **Robot antropomórfico.** También denominado robot de 6 ejes, son los manipuladores más típicos en el ámbito industrial. Constan de 6 grados de libertad permitiendo situar la herramienta en una posición con 3 diferentes orientaciones, permitiendo una amplia flexibilidad.

Figura 10. *Robots antropomórficos de los principales fabricantes.*

A raíz de este tipo de robot surgen robots de doble brazo que permiten trabajar armónicamente sobre una única pieza de trabajo. Así mismo dentro de este tipo de robot, se están implantando en la industria con gran rapidez una serie de robots colaborativos o cobots, capaces de trabajar junto a un ser humano.

A) **B)**

Figura 11. A) *Robot colaborativo YuMi.*
B) *Robot colaborativo UR3.*

- **Vehículos de guiado automático (AGVs).** Son robots que se comportan como vehículos autónomos, cuya aplicabilidad va ligada a la logística interna de la empresa. Habitualmente sobre ellos se monta un robot colaborativo que por necesidad debe desplazarse de una fase a otra del proceso productivo.

Figura 12. *Robots AGVs.*

Capítulo 2.3. Comunicaciones Industriales

Las redes de comunicación industriales entre los dispositivos que intervienen en el control de entornos de automatización de procesos industriales son clave para permitir la supervisión y el control de estos procesos, asegurando un correcto funcionamiento.

Con el objetivo de lograr el avance hacia la transformación digital de los procesos de las plantas de producción, la arquitectura de los nuevos sistemas de automatización debe basarse en solidas redes de comunicación industrial que utilizan protocolos estandarizados y que actúen como columna vertebral de la interconectividad, proporcionando un poderoso medio para el intercambio de datos y controlabilidad de los mismo, ampliando la flexibilidad para la conexión de varios dispositivos.

Hoy en día se hace uso de un gran número de protocolos de redes de comunicación industrial, ya sean impulsados por fabricantes específicos o definidos como estándares abiertos. Esta tecnología ha permitido la gestión eficiente de los procesos de producción, así como un aumento en la disponibilidad de la información de los dispositivos de campo, de forma centralizada a nivel de planta.

Históricamente, la problemática en la industria se ha dado a la hora de realizar la conexión entre dispositivos de diferentes fabricantes que hacen uso de diferentes protocolos, lo que complicaba el diseño de una red de comunicaciones que sea capaz de compatibilizar la información existente entre los diferentes sistemas.

Diferentes fabricantes y entidades han tratado de imponer un estándar, con el fin de unificar y simplificar las comunicaciones, sin haber alcanzado aún un compromiso pleno.

Capítulo 2.3.1. Niveles de comunicación en una red industrial

El ideal de factoría de procesos completamente automatizada (CIM) se representa en una pirámide estructurada en niveles jerárquicos, donde La información fluye entre los diferentes niveles. Pero no existe una única red de comunicación que satisfaga los requisitos de todos los niveles, por lo que cada nivel suele utilizar diferentes redes en función de sus necesidades, como volumen de datos, velocidad de transmisión, seguridad de los datos, etc.

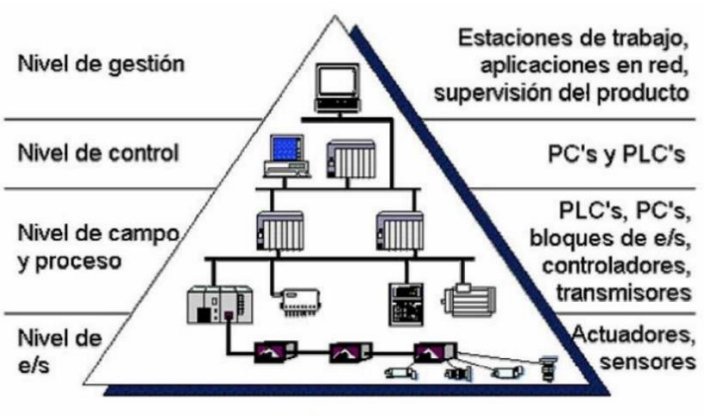


Figura 13. *Pirámide CIM de comunicaciones.*

- **Nivel de Entradas/Salidas.** Lo forman dispositivos de mando (actuadores) y elementos de medida (sensores) que se distribuyen por la línea de producción.
- **Nivel de campo y proceso.** En este nivel se encuentran los elementos que gestionan los actuadores y sensores del nivel anterior, permitiendo que estos trabajen de forma conjunta para controlar el proceso industrial. La tecnología más sofisticada utilizada a nivel de campo es la red de comunicación de bus de campo, pues facilita el control distribuido entre controladores y dispositivos de campo inteligentes.
- **Nivel de control.** Dentro de este nivel se encuentran los sistemas informáticos de automatización que controlan el proceso y que posibilitan la visualización de los procesos que se llevan a cabo en la planta de proceso, así como permiten disponer de un "panel virtual" con la información de alarmas, fallos o alteraciones de los procesos. En este nivel son utilizadas redes de área local (LAN) Ethernet con protocolos TCP/IP para interconectar las unidades de control a ordenadores, aunque también se utilizan buses de control como Profibus y ControlNet.

- **Nivel de gestión.** Es el nivel superior más alejado del proceso productivo, constituido principalmente por computadores. En este nivel adquiere importancia toda la información relativa a la producción y su gestión asociada. Las comunicaciones con este nivel no tienen por qué ser estrictamente de tipo industrial, es decir, robustas y con cortos tiempos de respuesta, sino que adquiere una mayor importancia el volumen de datos que se transmite. Por lo tanto, existen redes a gran escala en este nivel, como puede ser WAN Ethernet.

Capítulo 2.3.2. Tipos de protocolos y redes de comunicación industrial

Un protocolo es un conjunto de reglas utilizadas en la comunicación entre dos o más dispositivos. Sobre las bases de estos protocolos se describen los diferentes tipos de redes de comunicación industrial.

Se describen a continuación algunos de los estándares de comunicación más populares y comunes en la industria.

- **Comunicación Serial.** Sistema de comunicación más básico proporcionado para un controlador. Se implementa a través de protocolos como RS232, RS422 y RS485. Este sistema facilita la comunicación tanto analógica como digital bidireccional al mismo tiempo por un mismo cableado.
- **DeviceNet.** Red a nivel de dispositivo basada en tecnología CAN. Su diseño permite interconectar dispositivos de nivel de campo y controladores de nivel superior adoptando un protocolo CAN básico.
- **Modbus.** Protocolo de sistema abierto que puede funcionar en diferentes capas físicas. Proporciona una relación maestro-esclavo para la comunicación entre dispositivos conectados a una red. Modbus serial con RS232 o RS485 como capa física facilita la conexión de dispositivos en estructura de bus. La nueva versión Modbus TCP/IP utiliza como capa física Ethernet, lo que posibilita el intercambio de datos entre redes distintas.
- **Profibus.** Red de campo abierto ampliamente implementado para la automatización de procesos y fábricas. Adecuado para tareas de comunicación complejas y aplicaciones donde el tiempo es un factor crítico. Existen tres versiones: Profibus-DP, Profibus- PA, Profibus-FMS.
- **Bus de campo (Fieldbus).** Se trata de un protocolo de conexión de instrumentos en instalaciones industriales para su control en tiempo real. Los sistemas FieldBus permiten estandarizar el tipo de interconexiones entre los sistemas de automatización, lo que reduce notablemente los costes de cableado. A su vez este protocolo proporciona la base para integrar estrategias de mantenimiento predictivo.

La disposición de los diferentes elementos que conforman una red de comunicación (nodos, enlaces, etc.) hace que nos podamos encontrar con diferentes estructuras topológicas de una red, definiendo los aspectos lógicos y físicos de la red. La topología física hace referencia a la ubicación de los dispositivos e instalación de los cables, mientras que la topología lógica muestra el flujo de datos dentro la red independientemente del diseño físico.

En las redes industriales encontramos los siguientes tipos de topologías:

- **Redes punto a punto.** Es la topología más simple, con una comunicación directa entre dos dispositivos y protocolos de acceso al medio simples.
- **Topología de bus.** En este tipo de configuración la conexión de dispositivos se realiza en serie a un único bus troncal que actúa de canal de comunicaciones.
- **Topología en estrella.** Distribución basada en la existencia de un nodo o host central que conecta punto a punto con todos los destinos de la red.
- **Topología de árbol.** La conexión entre dispositivos se realiza en forma de árbol como su nombre indica. Puede considerarse una topología en estrella extendida. Cuenta con un cable principal que lleva la comunicación a todos los nodos de la red, compartiendo un mismo canal de comunicación.

Capítulo 2.3.3. Tecnología OPC

La tecnología OPC (OLE for Process Control) es un método de comunicación con una arquitectura cliente/servidor, donde una aplicación actúa como servidor encargado de proporcionar los datos y otra actúa como cliente leyéndolos y manipulándolos. OPC es por lo tanto un estándar para la conectividad, y no un protocolo, que se basa en los estándares más populares del mundo.

Es utilizado con la finalidad de comunicar dispositivos, controladores y/o aplicaciones evitando la problemática de las conexiones basadas en protocolos propietarios. Para lograr una comunicación independiente del fabricante, OPC abstrae los detalles tanto de servidor de datos, como del cliente, permitiendo así que se realice el intercambio de los datos entre ellos sin necesidad de conocer sus protocolos de comunicación nativos ni su organización interna de los datos. Es importante destacar que los protocolos nativos de comunicación de la fuente de datos y el cliente de datos no dejan de ser necesarios, ni son reemplazados en una comunicación mediante OPC.

Conceptualmente, las comunicaciones OPC trabajan de tal forma que se puede representar como una capa de “abstracción” intermedia entre la fuente de datos y el cliente, que permite el intercambio de información sin conocer nada

uno de otro. A nivel funcional, esto se consigue utilizando dos componentes especializadas denominadas 'Servidor OPC' y 'Cliente OPC'.

Un cliente OPC y un servidor OPC de distintos suministradores deben ser capaces de comunicarse entre sí sin problema, siempre y cuando estos cumplan con las mismas especificaciones OPC.

Servidor OPC

Un "OPC Server" es una aplicación software que actúa como un conector entre el entorno OPC con los protocolos nativos de una fuente de datos. Los servidores pueden leer y escribir en una fuente de datos si un cliente OPC así se lo pide, debido a la relación de tipo maestro/esclavo entre Servidor OPC/Cliente OPC.

Existen utilidades diseñadas específicamente para la comunicación entre servidores OPC, ya que el diseño base no está pensado para cumplir con esta función y tan solo permite la comunicación con clientes OPC.

Cliente OPC

Un cliente OPC es un módulo software capaz de comunicarse con un servidor OPC, mediante un servicio de mensajería específico. típicamente los servicios de cliente se encuentran embebidos en aplicaciones de HMIs, SCADAs, etc.

Es importante entender que los clientes OPC tan solo se comunican con servidores OPC por cómo están diseñados, y no con los dispositivos finales fuente de datos, premisa imprescindible para que se cumpla la característica de ser independiente de los protocolos nativos.

Un cliente OPC puede conectarse a tantos servidores OPC como necesite, sin la existencia de un límite teórico. Sin embargo, las comunicaciones cliente-cliente no están definidas, soportando únicamente la arquitectura cliente/servidor.

Capítulo 2.4. Visión artificial

La visión artificial, también conocida como visión por computador o visión computarizada, es una herramienta tecnológica que permite la extracción de información y su análisis mediante el procesamiento de imágenes captadas del mundo real o generadas mediante algún sistema de informático. Esta tecnología abarca varios campos de estudio como la programación, el modelado y las matemáticas.

La información que extraemos de estas imágenes puede ir desde el reconocimiento de modelos 3D hasta la posición relativa de la cámara respecto a un objeto, la búsqueda de contenido, restauración de imágenes, etc.

En numerosas ocasiones, esta visión computarizada trata de imitar el comportamiento de la visión humana, aunque esta suele ser más útil cuando se le da un enfoque más estadístico o de reconocimiento de geometrías.

Uno de los campos que más se ha beneficiado de esta tecnología en los últimos años es el de la robótica, ya que permite dotar de un mayor nivel de autonomía a estas máquinas al permitir reconocer con cierto nivel de precisión la localización de los objetos de su entorno.

Capítulo 2.4.1. Sistema de visión artificial en robótica

La visión artificial aplicada en el campo de la robótica permite dotar a nuestros sistemas de estrategias automáticas para el reconocimiento de patrones que son utilizados en la fabricación, proporcionando la interpretación e inspección de imágenes adquiridas para diferentes aplicaciones.

Esta ciencia, perteneciente a la rama de la ingeniería de sistemas, tarta de combinar las innovaciones existentes para adaptarlas a la resolución de problemas reales. La tecnología aplicada consta de cuatro etapas principales:

- Adquisición de las imágenes.
- Extracción de la información relevante.
- Análisis de la información.
- Comunicación de los resultados al sistema de control.

Existen diferentes tipos de sistemas de visión artificial básicos en función de la demanda de la aplicación que se desea satisfacer. Entre los más comunes se encuentran los siguientes.

- **Sistemas de visión 2D:** Esta técnica proporcionan una instantánea 2D con diferentes resoluciones que permiten el escaneo de líneas, regenerando esta imagen 2D línea por línea.
- **Sistemas de escaneo de área:** Estos métodos requieren de la utilización de cámaras de escaneo de área múltiple para cubrir toda la superficie de un objeto o girar el propio objeto frente a una única cámara para captar la totalidad de la superficie.
- **Sistemas de visión 3D:** En los sistemas de guía robótica este tipo de visión ofrece información de la orientación mediante sistemas multicámara que permiten utilizar la “triangulación” respecto a un punto objetico del espacio tridimensional.

Capítulo 2.4.2. Reconocimiento de objetos

Esta parte de la visión artificial estudia como detectar, en una imagen, objetos sobre la base de su apariencia visual, atendiendo a diferentes aspectos como el tipo de objetos (un coche, una cara, una rueda…) o la instancia del objeto (rueda de una marca X, rueda de una marca Y…). Este proceso de detección suele dividirse en dos partes principales:

- Extracción de características del contenido de una imagen.
- Búsqueda en basada en las características indicadas y las obtenidas de la imagen.

Para clasificar los objetos de la imagen, estos deben ser determinados en cuadros delimitadores que permitan la extracción de las características aplicando modelos matemáticos compactos que "resuman" el contenido de la instantánea tomada para realizar el aprendizaje de los objetos que se desea reconocer.

Capítulo 2.5. Concepto de simulación

La simulación de procesos, en el contexto de ingeniería industrial, sirve como herramienta para reproducir o imitar el comportamiento de un sistema cuando es sometido a una serie de estímulos o cuando se realizan cambios sobre él, pudiendo estudiar su comportamiento y analizar el impacto de las distintas variables que intervienen en el mismo. Esto permite disminuir los riesgos, optimizando la planificación y toma de decisiones en la creación de diseños, análisis y mejora de procesos.

La simulación es por lo tanto una parte muy importante en el desarrollo de sistemas, siendo esta la única técnica que permite que, aún sin disponer del sistema real, este sea programado, permitiendo extraer fallos y analizar el funcionamiento, pudiendo introducir conductas externas.

Dentro de un sistema simulado encontramos diferentes elementos que se asocian a los elementos del sistema real:

- **Entidades.** Conjunto de elementos físicos que representan los elementos simulados que constituyen o fluyen por el sistema. Dependiendo el tiempo que permanecen en el modelo se distingue entre entidades permanentes y temporales.
- **Atributos.** Son los adjetivos que identifican a las entidades, definiendo sus características.
- **Estado.** Situación en la que una entidad se encuentra en un instante determinado de la simulación en el que es observado.

- **Eventos.** Acciones que cambian el estado del sistema. A partir de los instantes elementales se articula la simulación, dando paso al comienzo o final de una actividad y se llega al resultado final.
- **Actividades.** Acciones realizadas por las entidades que suele necesitar del trabajo conjunto de dos o más entidades. Las actividades se sitúan entre dos eventos, uno de ellos precedente y que da paso a la realización de la actividad, y el originado tras la finalización de la actividad.
- **Colas.** Estados pasivos en que se mantiene las entidades hasta el instante que se cumplan las condiciones para que estas cambien de estado.

Capítulo 2.5.1. Ventajas.

- Permite explorar diferentes alternativas, ayudando en la toma de decisiones sobre la estructura y funcionamiento del sistema.
- Ahora de costes y optimización del tiempo en pruebas con el sistema real.
- Aumento de la capacidad de análisis de los puntos críticos del sistema.
- Capacidad de evaluación del diseño de instalaciones para poder adaptarse a la fabricación de nuevos modelos.
- Permiten experimentar con el sistema antes de su implantación física, pudiendo evitar condiciones de peligro.
- Previsualización del proceso y de los resultados obtenidos con el sistema.
- Facilidad de modificación, revisión y optimización de los diseños en tiempo real.

Capítulo 2.5.2. Inconvenientes.

- Resultados imprecisos, lejos de alcanzar el 100% de realismo en las simulaciones.
- Elevado coste y tiempo de desarrollo de los softwares utilizados en simulación.
- Elevado coste computacional, en ocasiones requiere dedicar equipos exclusivamente para esta tarea.
- Existen variables externas que no se pueden simular o no se puede medir su efecto sobre el sistema.
- La toma de decisiones no puede realizarse valiéndose tan solo de la simulación, sino que después deben realizarse una serie de pruebas que garanticen el funcionamiento correcto y así encontrar la solución final.

[Página en blanco por convenio]

CAPITULO 3. ESTADO DEL ARTE

Existen en el mercado diferentes herramientas que permiten la simulación de entornos industriales en una gran cantidad de escenarios, pero la mayoría de estas herramientas no son de uso abierto o libre, siendo generalmente diseñadas por grandes empresas para sus procesos productivos a través de una gran inversión.

En este proyecto se pretende lograr un objetivo similar al que dan estas costas herramientas, haciendo uso de los programas propietarios que proporcionan los proveedores, que por regla general se ponen a la disposición de los clientes al adquirir sus productos.

Capítulo 3.1. Antecedentes

Se han realizado en la Escuela de Ingenierías Industriales de la Universidad de Valladolid diferentes trabajos que hacen uso de estos programas con distintos enfoques para el desarrollo de células robotizadas o instalaciones industriales automatizadas.

[22] En 2016, Rocío Pérez Fernández realizo la simulación de una instalación de un proceso industrial en su trabajo fin de máster, donde se disponía de una isla robotizada simulada en el software de ABB, RobotStudio, y programando el funcionamiento de un PLC y un HMI con las herramientas Step 7 y TIA Portal.

[23] En 2017, Daniel García Fernández basó su proyecto final de máster en el modelado y simulación de una instalación real de fabricación y logística, empelando para ello un simulador de instalaciones industriales, en su caso Factory I/O. Utilizando para el control de la instalación un PLC Siemens y tecnología OPC para las comunicaciones.

[24] En 2019, Carlos Rodríguez-Rabadán Mateos-Aparicio utilizó en su proyecto final de grado algunas funcionalidades del software del proveedor Fanuc, Roboguide, desarrollando una aplicación industrial de seguridad mediante comprobación dual con la funcionalidad DSC (Dual Safety Check).

[25] En 2020, José Delgado García realizó su trabajo fin de grado en el cual realizaba una célula robotizada para la formación basada en equipos Fanuc y Siemens, la creación de esta aula de formación se basó en el montaje físico de la misma, así como la programación del autómata y el robot utilizados.

[26] También en 2020, Luis Ignacio Matía desarrolló, en proyecto fin de grado, una célula robotizada basada en tecnologías de la industria 4.0, haciendo uso

del software Roboguide como pieza fundamental del proyecto y un autómata programable Siemens.

Una vez analizados algunos de los proyectos de fin de estudios realizados en la Universidad de Valladolid, que tengan relación con la robótica, automatización de procesos y comunicaciones entre autómatas y robots, se puede apreciar como la existencia de trabajos realizados suelen basarse en los software de los proveedores ABB y Siemens, RobotStudio y Step7/TIA Portal respectivamente, en gran parte por la facilidad de licencias que proporciona la Escuela de Ingenierías Industriales para el uso de estos programas.

Capítulo 3.2. Análisis del problema

Tras el análisis de los anteriores trabajos, se presentó la posibilidad de trabajar con software Roboguide del proveedor Fanuc, del cual la documentación docente en la Universidad de Valladolid es más escasa, en conjunto con el software de Siemens, con el objetivo de poder desarrollar una herramienta útil a nivel académico, que permita la comunicación de ambos programas en el desarrollo de una célula robotizada, gracias a la tecnología OPC. Se plantea además la posibilidad de desarrollar un IHM (Interfaz Hombre Maquina) con un software adicional que se pueda modificar con facilidad para diferentes proyectos futuros.

La formación en entornos simulados, sin manipulación de los dispositivos físicos hasta fases avanzadas del desarrollo de los proyectos, es cada vez más importante de cara al mundo laboral. Sobre todo, a partir de la situación provocada por el COVID-19 en nuestra sociedad, por lo que la búsqueda de herramientas que faciliten su aprendizaje es cada vez más indispensable.

Capítulo 3.2.1. Análisis de requisitos

Antes de dar comienzo al proyecto han de sentarse una serie de requisitos que deben de cumplirse tras la finalización de este trabajo y que definen el camino que se va a seguir. A continuación, se establece una lista con los requisitos que se consideran necesarios para el alcance del trabajo.

1. Diseño de la célula robotizada, así como los elementos que se van a manipular.
2. Programación de los robots (lógica, configuración…) e investigación acerca de los complementos y opciones de software adicionales.
3. Sincronización y coordinación entre los elementos que forman la isla y las tareas realizadas, mediante el uso de un controlador programable simulado en un entorno diferente al software de simulación robótica.

4. Diseño e implementación de un interfaz hombre máquina para la gestión de la simulación.
5. Establecimiento y adaptación de un protocolo de comunicaciones para el intercambio de información entre los programas.

Capítulo 3.2.2. Soluciones propuestas

Se han estudiado diferentes posibilidades para llevar a cabo un proyecto que cumpliera con los requisitos del capítulo anterior. A continuación, se comentarán estas diferentes propuestas, especificando las razones por las que no se han llevado a cabo finalmente.

1. **Crear una estación de montaje de puertas de un vehículo genérico.**
   En las prácticas curriculares el autor de este proyecto trabajo en el desarrollo de una línea de montaje de puertas en un vehículo genérico. Esta estación era muy interesante pues se trabaja en conjunto con la sincronización entre varios robots. El inconveniente se da desde el punto en el que las señales cruzadas son gestionadas por un robot maestro y se realizan los intercambios sin seguir ningún protocolo.

2. **Paletización de refrescos con clasificación por sabores.**
   Haciendo uso de las funcionalidades Line Tracking e iRVision integradas en el software Roboguide del proveedor Fanuc, se clasificarían en diferentes cintas los paquetes entrantes según el color exterior para un posterior paletizado. Este proyecto se ha valorado varias veces durante la etapa de estudio del software Roboguide, pero la funcionalidad Line Tracking no permite trabajar con dispositivos externos, lo que dificulta la integración de un PLC y una interfaz hombre máquina.

3. **Clasificación de piezas según su geometría para reconducirlas a otra etapa del proceso productivo.**
   Gracias al complemento iRVision se puede hacer una clasificación de piezas, según su geometría, procedentes de un conveyor de entrada. Este proyecto fue descartado, pues únicamente se trabajaría con un robot, además la integración del PLC en este caso no tendría mayor funcionalidad que ejecutar el '*bypass*' de arranque tras la señal del operario.

Capítulo 3.2.3. Propuesta seleccionada

Tras la valoración realizada sobre las propuestas anteriores, se decidió extraer las bondades de cada una de ellas dando como resultado el contenido de este proyecto.

Para ello se hará uso de 3 robots del catálogo del fabricante Fanuc, que realizarán tareas de 'pick&place' con piezas de grandes dimensiones. La estación estará gobernada por un PLC que se encargará de coordinar los robots y las maquinas que la forman. Además, se contará con una interfaz hombre máquina, específicamente diseñada para que el operario encargado de la instalación pueda gestionarla estación y en todo momento observar el estado del proceso y de los robots.

Los robots escogidos del catálogo del fabricante Fanuc son: *R-200iC/125L, M-710iC/45M y R-200iC/165F*, el principal factor que se ha buscado a la hora de escoger estos robots ha sido su rango de alcance, pues las piezas con las que se va a trabajar no son de elevado peso, pero si se requiere de un alcance amplio para llegar sin problema a los puntos de trabajo.

Como autómata programable se va a hacer uso de un *Siemens S7-1214C AC/DC/Relay*, debido al conocimiento por parte del autor de este tipo de autómata por haber trabajado anteriormente con él. En caso de querer llevar este proyecto a cabo de forma física, debería buscarse un modelo que cuente con un mayor número de señales digitales o estudiar el número módulos adicionales de señales digitales.

Para el desarrollo de la interfaz de usuario se va a hacer uso de la utilidad *AppDesigner* de *MATLAB*, ya que la personalización y libertad que proporciona, así como la posibilidad de integración junto con la toolbox de *OPC Communication* son ideales para este proyecto. El uso de este software se debe en gran parte también al enfoque didáctico que se pretende dar a este trabajo, así como las posibilidades de expansión futuras enfocadas a la gestión de más células en distintos softwares de simulación.

El objetivo funcional de la estación que se va a desarrollar es el montaje de un televisor de grandes dimensiones, que son cada vez más comunes en nuestra sociedad. Las grandes dimensiones que estos paneles están alcanzado en los últimos años dificultan que puedan ser manipulados con agilidad por parte de los operarios, por lo tanto, a pesar de estar formados por piezas de no muy elevado peso, se está optando por robotizar y automatizar este tipo de tarea.

Se van a diseñar por lo tanto los principales componentes de un televisor genérico de un tamaño aproximado de 75 pulgadas. Así mismo, las herramientas para la manipulación de las piezas y los elementos de seguridad de la célula serán diseñados específicamente para este proyecto, lo que requerirá un estudio de la normativa en materia robótica actual para el diseño de los elementos de seguridad, así como un estudio de las herramientas comúnmente utilizadas en la industria para las labores que se van a desarrollar.

CAPITULO 4. HERRAMIENTAS SOFTWARE UTILIZADAS

En los siguientes capítulos se realiza una introducción a los programas que han sido utilizados para el desarrollo del proyecto. Puesto que uno de los objetivos principales es hacer valer los conocimientos adquiridos a lo largo del grado, han sido necesarios una serie de softwares de diferentes ramas de la ingeniería que permitan la aportar soluciones en diferentes situaciones al problema en conjunto que se ha descrito en el capítulo anterior.

A continuación, se encuentra un esquema que clarifica los aspectos que han sido trabajados y el respectivo programa que ha sido utilizado para realizarlo:

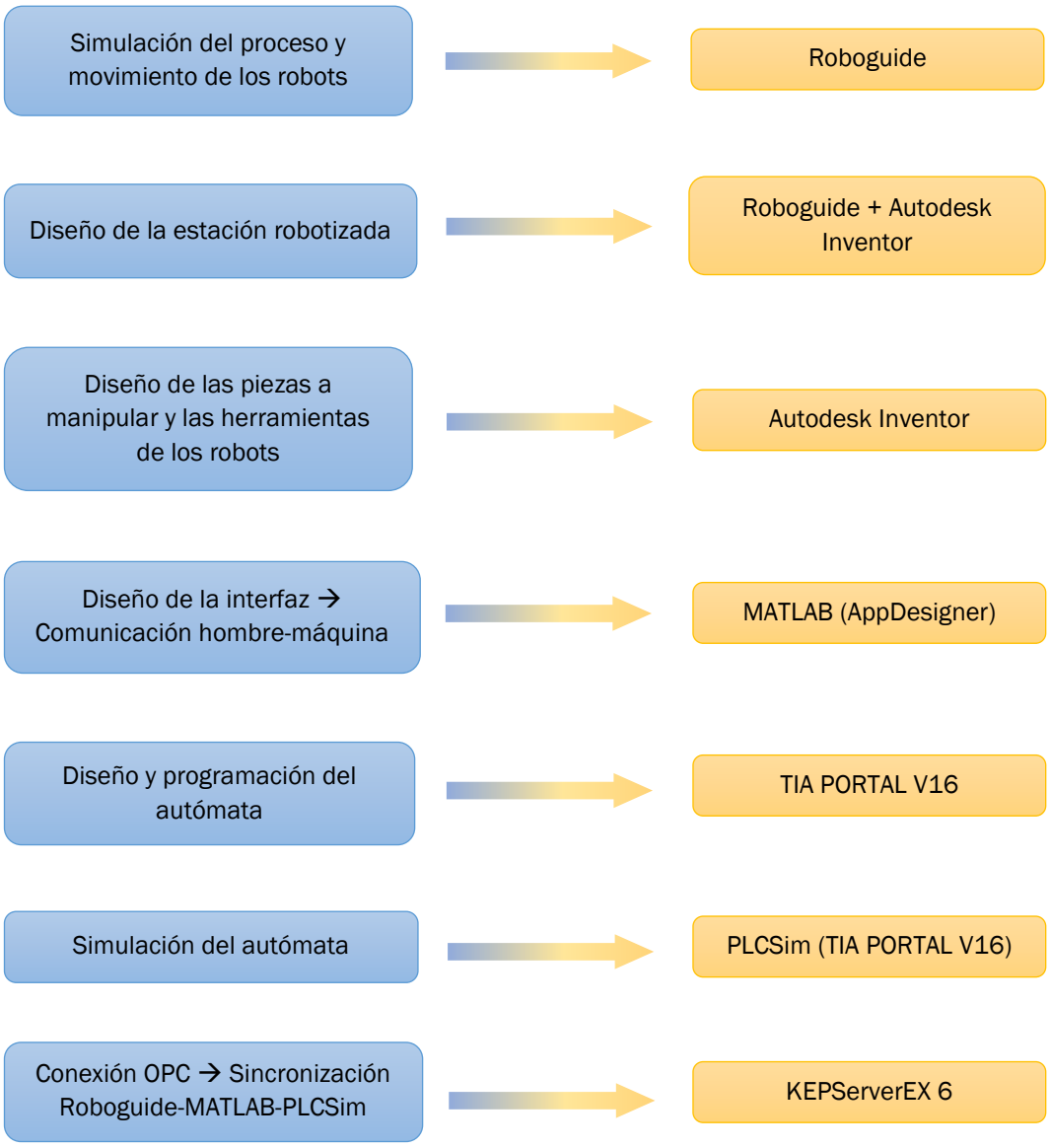


Figura 14. *Esquema representativo del software utilizado y sus aplicaciones en el desarrollo del proyecto.*

Capítulo 4.1. Roboguide

Capítulo 4.1.1. Introducción

Como uno de los principales objetivos de este proyecto es el diseño de una célula robotizada, será necesario un software que nos permita llevar a cabo este diseño del entorno físico de trabajo, así como la programación de los movimientos de los robots para posteriormente realizar una simulación de su funcionamiento.

El programa Roboguide es una herramienta desarrollada por Fanuc que permite la simulación de movimiento, evaluación de células y tiempos de ciclo, así como la programación de cualquier tipo de robot de la marca. Este software está ampliamente extendido y es utilizado por un gran número de empresas para trabajar de forma offline en el modelado de células y selección de los robots más adecuados en proyectos de simulación de procesos industriales.

El trabajo offline y la pre-programación permiten la modificación de parámetros, variables, trayectorias… que serán exportados del programa y cargados en los robots, reduciendo el tiempo que se invierte en realizar cambios de diseño en las instalaciones físicas, garantizando así un impacto mínimo en la producción.

Dentro del propio software Roboguide existen diferentes herramientas, en función de cuál sea el cometido que requiera la aplicación de un robot en específico. Estas herramientas permiten aumentar la productividad eliminando factores como el riesgo de errores y minimizando los tiempos de configuración y ciclo.

- HandlingPRO. Permite simular y probar procesos de manipulación y ‘pick&place’ y realizar un estudio del comportamiento y viabilidad de uso un robot para una aplicación de este tipo según la geometría, volumen y peso de los objetos que se van a utilizar sin necesidad de un prototipo físico.
- ChamferingPRO. Genera y simula trayectorias de desbarbado de forma automática. Los programas con las trayectorias se generan de forma automática delimitando el contorno en el CAD 3D de la pieza.
- OLPCPRO. Complemento de desarrollo y mantenimiento de programación mediante “*teach pendant*” y programación KAREL.
- PaintPRO. Herramienta de aprendizaje de programación de trayectorias y desarrollo de procesos de pintura. Contiene funciones especiales para el control de la herramienta, patrones de pintura, delimitación de zonas a pintar, solapado, velocidad y tiempos de activación de la pistola de pintura
- PalletPRO. Permite crear, depurar y probar aplicaciones de paletizado y des paletizado offline. Los datos creados se pueden cargar en un controlador de un robot real que cuente con la utilidad de paletizado PalletTool.

- **iRPickPRO**. Herramienta específica para la programación offline de tareas *pick & place* que requieren de alta velocidad.
- **WeldPRO**. Simulación de procesos de soldadura con arco robotizada.

Adicionalmente, los robots Fanuc, así como su software de simulación, cuentan con una serie de complementos comunes que dotan de mayor flexibilidad a sus robots, como pueden ser herramientas de visión artificial, seguridad o movimiento entre otras. A continuación, se describen las principales.

- **iRVision.** Sistema de reconocimiento de imágenes 2D o 3D integrado en sistemas Fanuc para manipulación en diversas situaciones, como localización de piezas por tamaño, forma o posición, lectura de códigos de barras, ordenar por colores, etc.

Figura 15. *Fanuc iRVision.*

- **Fanuc Dual Check Safety (DCS).** Complemento que ofrece una solución software inteligente para mantener la seguridad de operarios, robots y herramientas, creando zonas de trabajo donde el robot puede o no acceder en función de las señales recibidas.

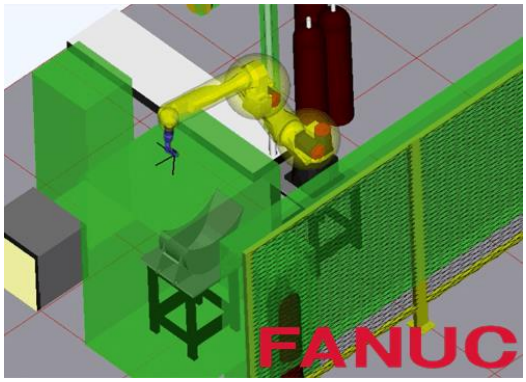


Figura 16. *Restricción de áreas mediante Dual Check Safety.*

- **Fanuc Multi Group Motion.** Complemento que permite el control de un eje auxiliar totalmente integrado y estandarizado, como un rail, una servo-pinza, dos robots trabajando de forma conjunta o un robot y un posicionador.

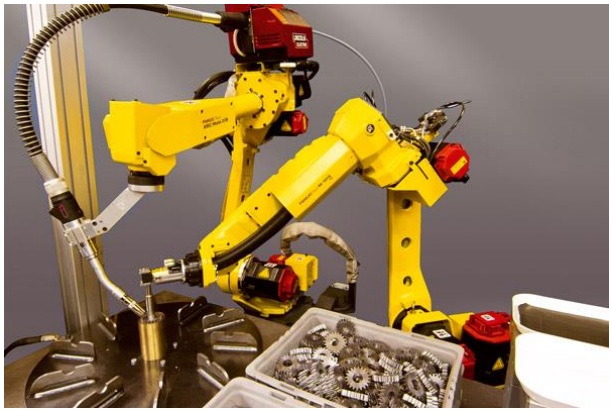


Figura 17. *Trabajo conjunto coordinado mediante Multi Group Motion.*

Capítulo 4.1.2. Descripción del entorno HandlingPro

Roboguide ha sido el software de simulación robótica seleccionado para la realización del proyecto por los conocimientos adquiridos en el periodo de prácticas en empresa de la Universidad de Valladolid.

La interfaz del software presenta el aspecto que se puede observar en la siguiente figura.

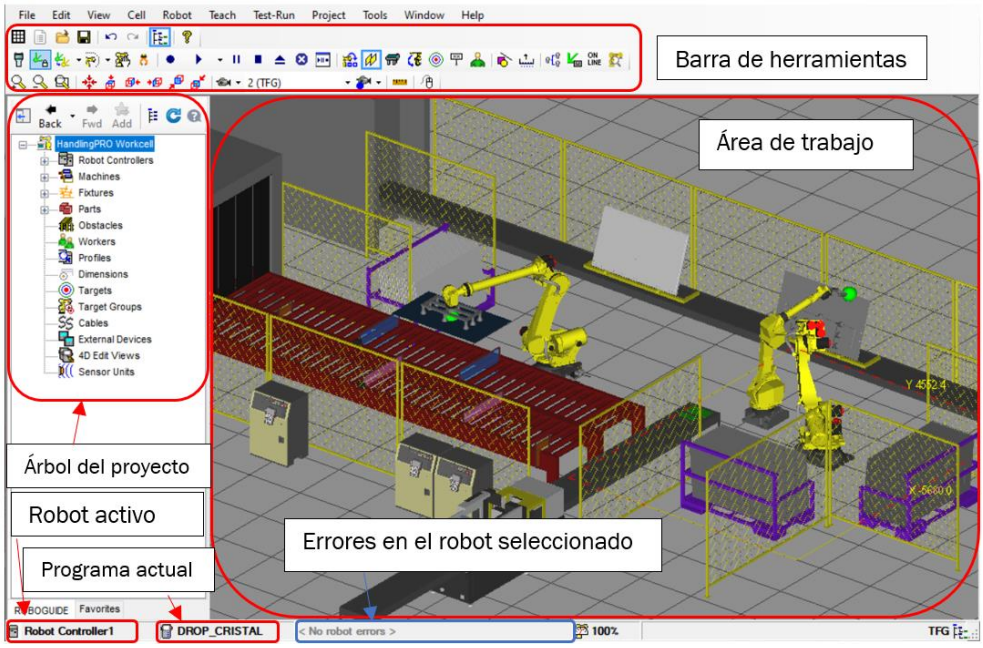


Figura 18. *Software de simulación Roboguide.*

El software de simulación nos permite seleccionar entre diferentes controladores para poder simular las versiones más recientes, así como aquellas más antiguas que aún pueden encontrarse en algunas empresas. En este caso se ha trabajado sobre la herramienta HandlingPro seleccionado la versión V9 que permite trabajar con los controladores más recientes R-30iB Plus, los cuales aportan compatibilidad con todas las herramientas adicionales que se han descrito anteriormente.

Prestando atención a las regiones señaladas en la *Figura 18.* Software de simulación Roboguide. se pueden distinguir las principales divisiones características de un software de este tipo que se estudian a continuación.

En la barra superior sen encuentra un conjunto de menús dedicados a ajustes del entorno de simulación, como número de ventanas, puntos de vista y parámetros de simulación, ajustes de los controladores del robot o de la propia célula robotizada. Debajo de estos menús se encuentra una barra de herramientas con accesos directos a las funcionalidades más comunes.

La región central la abarca el espacio de trabajo donde se muestra en tiempo real la simulación que se está llevando a cabo. En la zona izquierda se encuentra el árbol de trabajo donde se muestran los elementos que se encuentran en el espacio de trabajo y que componen la instalación.

Por último, la barra inferior la abarca información relativa los que está sucediendo en el área de trabajo en cada instante. Encontramos en esta barra contextual el robot que se encuentra activo y el programa que se encuentra seleccionado en dicho robot, así como si existe algún error que deba ser corregido para continuar con la ejecución.

A continuación, se da una breve descripción de las principales funcionalidades de la barra de herramientas.

Figura 19. *Barra de herramientas en Roboguide.*

1. Conjunto de accesos para edición que permiten abrir un nuevo proyecto o uno existente, guardar el proyecto actual y deshacer o rehacer cambios.
2. Acceso directo para ocultar o mostrar el árbol de trabajo con los componentes de la célula actual.
3. Iniciador del Teach Pendant o iPendant. Este icono abre una consola virtual del robot seleccionado en el organigrama anteriormente mencionado. Los movimientos, cambio de variables, definición de trayectorias y programación de los robots se lleva a cabo a través de esta consola.

4. Conjunto de opciones referentes a la herramienta activa en el robot seleccionado en el árbol de trabajo. Con estas herramientas podemos mostrar donde se encuentra el TCP de la herramienta para desplazar u orientar el robot, mostrar el espacio alcanzable por el robot (con y sin la herramienta), mostrar los sentidos de rotación de cada eje del robot y manipular la herramienta entre sus estados activado y desactivado (abierto y cerrado, desplegado o contraído…).
5. Herramientas de control de la simulación, con ellos podemos iniciar o pausar la simulación dinámica, grabar la simulación en tiempo real (según el reloj interno de la simulación) y por lo tanto observar cómo sería el funcionamiento en modo automático.
6. Pulsando en este icono abrimos un menú emergente que permite determinar con que sistema de coordenadas queremos trabajar.
7. Conjunto de herramientas para el trabajo con conveyor, con ellas podemos mostrar las barreras de detección de objetos, así como simular de forma independiente el movimiento de la cinta sin necesidad de una simulación de la célula completa.
8. Este icono despliega un menú emergente con el que se puede mover el robot a un punto (perteneciente a un objeto) que sea alcanzable, señalándolo con el ratón, sin necesidad de conocer las coordenadas de este punto.
9. Herramientas de zoom y centrado del entorno de trabajo.
10. Conjunto de herramientas para la selección de la perspectiva de visualización del entorno de trabajo.

Dentro del árbol nos encontramos con los elementos que se observan con mayor claridad en la siguiente figura.

Figura 20. *Organigrama de elementos que componen la célula.*

- **Robot Controllers.** En este apartado encontramos los accesos a la información referente a los robots que se han cargado en la estación, como

sus programas, herramientas, variables, elementos de visión y line tracking entre otros.

- **Machines.** Dentro de este submenú desplegable se encuentran listados los elementos de la isla robotizada actual que contienen una o más partes móviles.
- **Fixtures.** Apartado formado por elementos, generalmente estáticos, que no suponen un obstáculo para los robots por su localización. Estos elementos suelen contener objetos que van a ser manipulados por el robot. Se definen también en esta categoría los elementos del complemento '*line tracking*'.
- **Parts.** Son los objetos que van a ser manipulados en la simulación, o que pertenecen a un elemento de la categoría anterior (Fixtures)
- **Obstacles.** Submenú constituido por aquellos elementos cuya localización dentro de la célula puede interferir con las trayectorias de los robots y que deben ser evitados, pues la colisión con alguno de estos objetos puede tener graves consecuencias.
- **Workers.** En este apartado se encuentran los complementos que se pueden agregar a la simulación como representación de operarios.
- **External Devices.** Desde este submenú se pueden añadir dispositivos externos, como robots que se encuentren en otra célula simulada (en Roboguide) en una instancia diferente a la actual, máquinas de control numérico CNC, o servidores OPC para el intercambio de información.
- **Sensor Units.** Apartado constituido por los elementos de visión que incorpora la estación. Se incluyen aquí cámaras de visón 2D, 3D laser y sensores de área 3D.

Se incluye en el software una librería con las características geométricas de los robots, así como un repositorio de objetos que se pueden usar en la definición herramientas, obstáculos o accesorios. Igualmente existe la posibilidad de importar archivos CAD que definan de la geometría de los elementos.

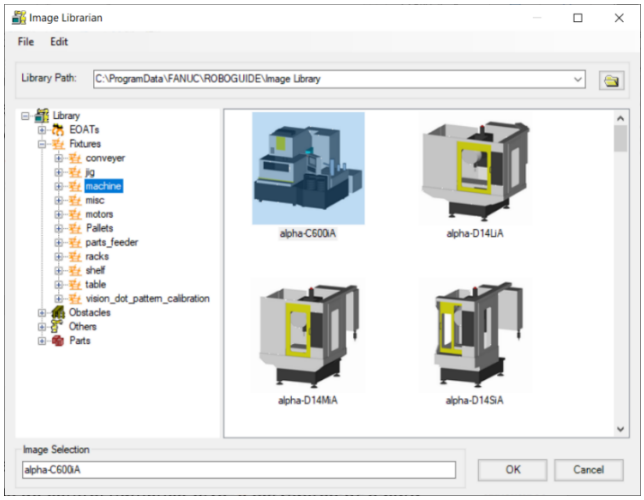


Figura 21. *Librería de objetos CAD en Roboguide.*

Capítulo 4.1.3. Programación del robot

Se puede definir la programación de un robot como un conjunto de órdenes que se ejecutan de forma secuencial para completar el proceso que se desea lograr, para ello los comandos se van ejecutando paso a paso.

El software de simulación Roboguide dispone de dos métodos para el diseño de programas, enfocados a diferentes funciones que se exponen a continuación.

Roboguide TPE (“*Teach Pendant Editor*”).

A través del “*iPendant*”, el cual tiene el mismo software que un “*Teach Pendant*” físico, que se proporciona en la aplicación no solo podemos acceder a los menús de configuración del robot, además, se puede utilizar para crear programas complejos con todas las funcionalidades de los programas TP.

Figura 22. *Virtual Teach Pendant (iPendant).*

Se definen a continuación una serie de instrucciones habituales en la programación TPE de robots Fanuc. En capítulos posteriores se verá el uso de estas instrucciones en programación.

- **Registros (R[n]).** Son variables reales (32 bits) o enteros, de acceso global desde cualquier programa. En ellos es posible almacenar el resultado de una operación aritmética, o introducir un valor ya sea una constante, un valor procedente de alguna señal de entrada-salida, otro registro o un elemento de un registro de posición.
- **Registros de posición (PR[i,j]).** Son puntos en coordenadas 'joint', puntos en coordenadas cartesianas o matrices, igualmente de acceso global ya sea en su conjunto o elemento por elemento.
- **Salto incondicional.** Compuestas por dos elementos, una marca (o etiqueta) de emplazamiento de destino de salto (LBL[n]) y la propia instrucción de salto incondicional (JMP LBL[n]) que permite efectuar un salto (o bucle) a una etiqueta localizada en el mismo programa.
- **Salto condicional (IF/SELECT).** Una instrucción de salto condicional permite realizar un salto (o bucle) a una marca del propio programa solo si se cumple la condición impuesta. En caso contrario se continua la ejecución del programa.
- **Llamadas (CALL).** La instrucción <CALL Programa> permite lanzar un programa secundario, el cual se ejecuta completamente hasta finalizar y se continua la ejecución del programa origen de la llamada.
- **Instrucción de espera (WAIT).** Pausa la ejecución de un programa ya sea durante un tiempo especificado (WAIT [tiempo]) o hasta que se cumpla una condición (WAIT [condición][operando(=,≠)][ON/OFF]).

Igualmente, conviene revisar el formato de las instrucciones de movimiento a las posiciones indicadas para definir las trayectorias deseadas, el cual se ilustra en la siguiente figura.

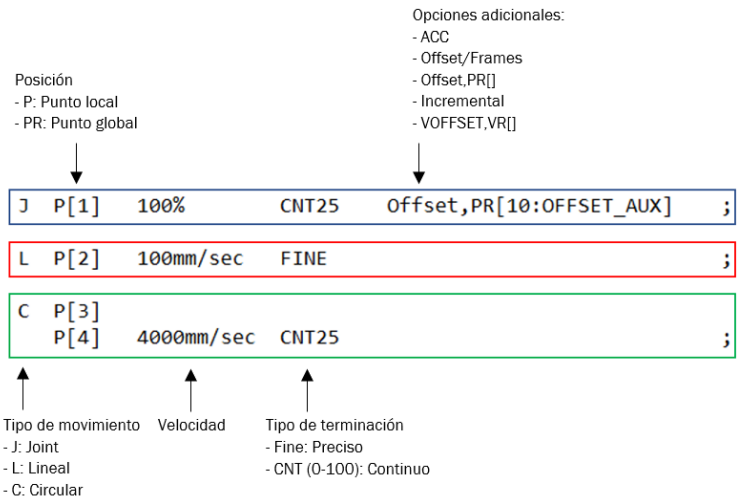


Figura 23. *Formato de instrucciones de movimiento.*

El primero de los campos indica el tipo de movimiento que se va a realizar: lineal, el cual desplaza el TCP de la herramienta en una trayectoria totalmente

lineal desde el punto origen al punto destino; enlace (joint), realiza la trayectoria más conveniente hasta el punto final; circular, define un arco con tres puntos, el punto actual, el punto de paso P[3] y el punto final P[4].

Los siguientes campos establecen el tipo de variable (global o local) donde se almacena la posición del punto del programa que se desea alcanzar y la velocidad de movimiento.

El cuarto campo define el tipo de finalización del movimiento. Este puede ser continuo (CNT) o finalización precisa (FINE).

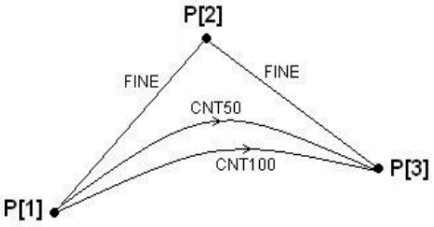


Figura 24. *Tipo de finalización del movimiento.*

El ultimo campo es opcional y define una serie de modificaciones y/o correcciones en el movimiento. Dependiendo del conjunto de funcionalidades adicionales del robot se pueden tener en este campo más o menos opciones.

Roboguide "*Simulation Programm Editor*".

El software Roboguide cuenta con un editor de programas más simple, que permite la programación a través de un entorno gráfico específico para esta tarea. A la hora de ejecutar una simulación, los programas creados con esta herramienta cuentan con la ventaja de simular las animaciones de ciertas acciones como la toma de elementos del entorno, permitiendo evaluar de forma *"offline"* la correcta ejecución de un ciclo de trabajo, controlando factores como la colisión de los objetos tomados con otros elementos del entorno de simulación.

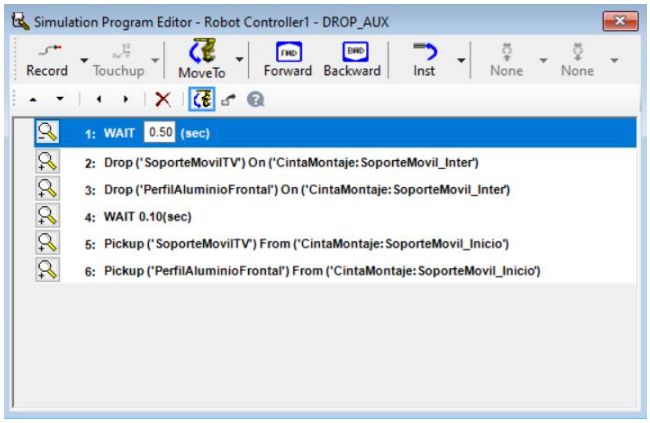


Figura 25. *Ejemplo programa de simulación.*

Capítulo 4.2. TIA Portal, PLCSim y NetToPLCsim

Capítulo 4.2.1. Introducción

En el desarrollo de una estación de trabajo robotizada se busca alcanzar una producción constante que sea capaz de mantener una cierta precisión y cumplir con los estándares de seguridad en la industria. Para ello, los componentes que forman la célula deben trabajar en armonía controlados por un PLC.

Para este proyecto se ha escogido Siemens como el proveedor del autómata programable que gobernará la célula de trabajo, ya que es hoy en día el fabricante más popular en lo referente a la automatización de plantas industriales. Posee diferentes controladores modulares que se clasifican por su grado de compactación y nivel de rendimiento. Igualmente cuenta con un software propio para la configuración y programación de sus autómatas y una herramienta complementaria para la simulación del funcionamiento de estos.

[28] TIA Portal ("*Totally Integrated Automation Portal*") proporciona acceso sin restricciones a la gama completa de servicios de automatización digitalizada de Siemens, desde la planificación digital y la ingeniería integrada hasta la operación transparente.

Al no disponer de los controladores físicos es necesario un software que simule el funcionamiento del autómata y sus conexiones. Para ello se va a utilizar un complemento software llamado PLCSim, diseñado por la misma marca Siemens. Esta herramienta capaz de crear un controlador virtual nos permite cargar y comprobar el funcionamiento de los programas creados de forma sencilla y rápida y lo más importante, sin necesidad de contar con los dispositivos físicos.

Se requiere de una extensión de red TCP/IP para el simulador PLCSim, que permita establecer comunicaciones simuladas con sistemas a través de la misma red. Esta extensión software, denominada NetToPLCsim, nos permite acceder al simulador PLCsim a través de una interfaz de red del PC donde se ejecuta la simulación, permitiendo simular aplicaciones cliente sin necesidad del PLC real.

Capítulo 4.2.2. Descripción del entorno TIA Portal

Para este proyecto se ha escogido la versión V16 del software TIA Portal, con el que se puede gestionar los datos que son necesarios para realizar un proyecto de automatización mediante autómatas programables.

La pantalla inicial con la que nos encontramos al ejecutar el software nos permite la gestión de los proyectos. Desde esta ventana crearemos un nuevo proyecto desde cero, abriremos uno existente o migraremos el proyecto a otras versiones en función de nuestras necesidades.

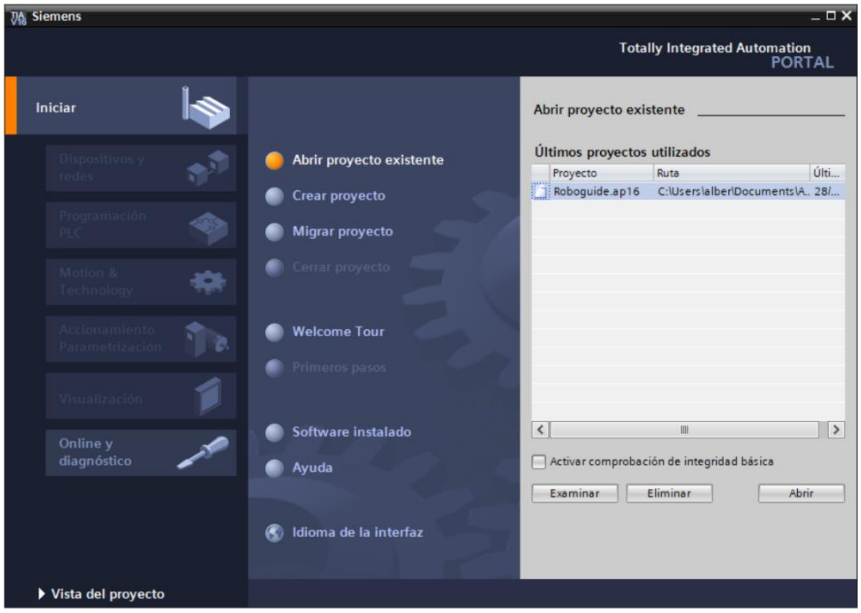


Figura 26. *Vista de portal en TIA Portal V16.*

En este caso se va a crear el proyecto desde cero, pues no se parte de ninguna base de algún trabajo anterior. Una vez creado el proyecto, establecido un nombre y una ruta donde ser guardado, se accede a la pestaña de *'Dispositivos y redes'*, donde podemos escoger entre toda la gama de controladores de Siemens, así como otros dispositivos HMI, sistemas PC y accionamientos propios de la marca que se recogen dentro desde este mismo programa.

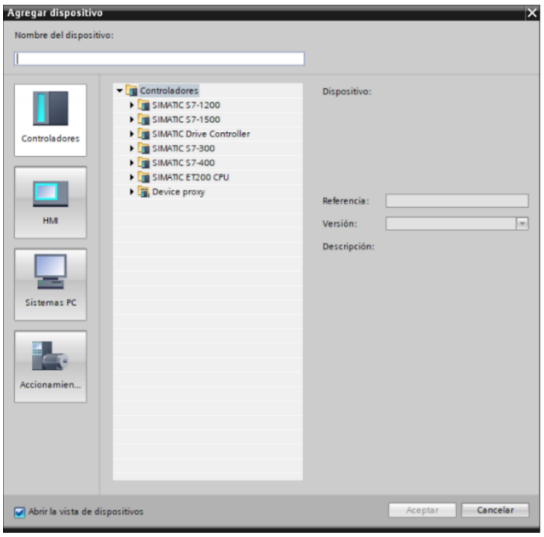


Figura 27. *Pestaña para agregar elementos al proyecto.*

Una vez añadidos los dispositivos necesarios se pasa a la vista de proyecto, con una interfaz propia de un entorno de programación que cuenta con una barra de menús y herramientas desde donde podremos ajustar los parámetros del proyecto y acceder a funcionalidades básicas de forma rápida.

En la zona izquierda se dispone un árbol del proyecto que alberga la información referente a los elementos que se han añadido. Por otro lado, la sección derecha está destinada a una serie de ventanas con herramientas de configuración y modelado del espacio de visualización del proyecto, así como una serie de librerías con funciones de programación específicas.

La región central se encuentra dedicada al área de trabajo, desde donde se pueden programar los dispositivos añadidos al proyecto. En la zona inferior existe una ventana donde podemos encontrar información relacionada a las propiedades y configuración del dispositivo sobre el que se esté trabajando en cada momento.

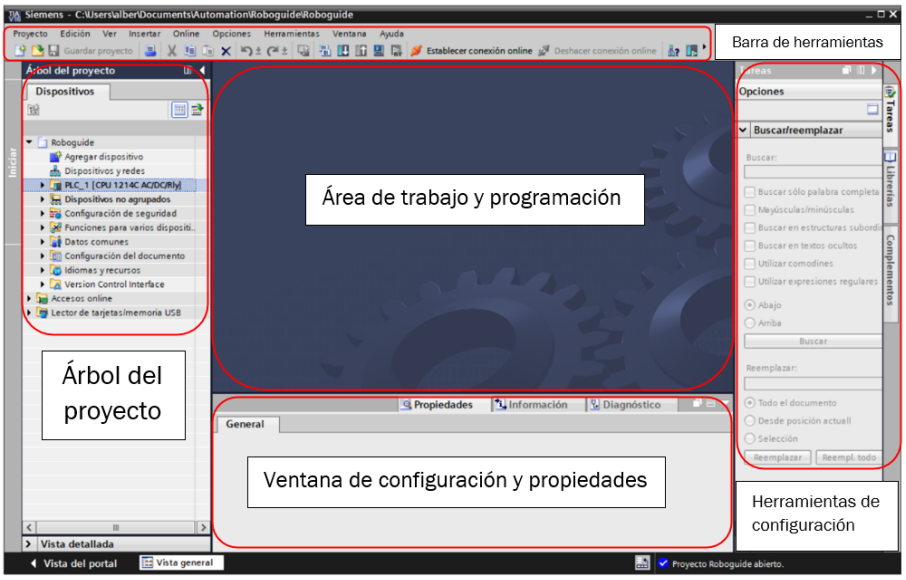


Figura 28. *Vista del proyecto en TIA Portal.*

Seleccionando el/los dispositivos que se hayan añadido previamente en el árbol del proyecto se despliegan una serie de accesos a opciones para la configuración y programación de estos.

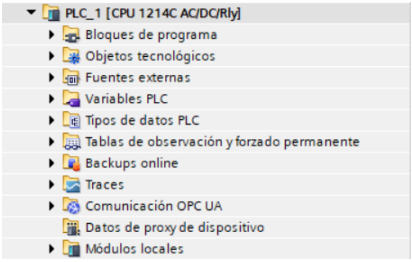


Figura 29. *Vista del controlador en el árbol del proyecto.*

Capítulo 4.2.3. Programación y lenguaje

Dentro del grupo ***Bloques de programa*** se encuentran las diferentes estructuras que permiten la programación de usuario que *TIA Portal* proporciona para gestionar el sistema, organizando de las diferentes funciones que se quieran dar a nuestro dispositivo. Los bloques de programa que se ponen a disposición del usuario se clasifican en 4 tipos:

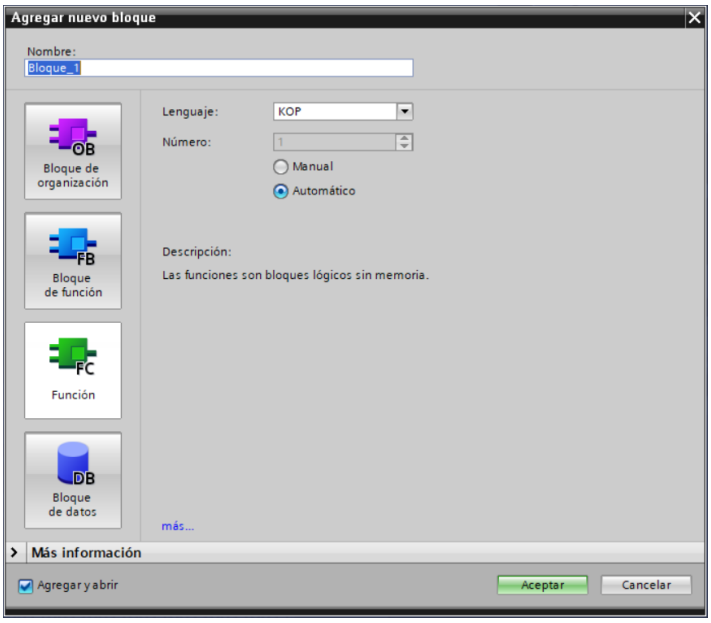


Figura 30. *Bloques de programa en TIA Portal.*

- **Bloque de organización (OB).** Son los bloques lógicos de orden superior (OB1) que se procesan cíclicamente. Desde estos bloques se realizan llamadas al resto de funciones y bloques de función, actúan como "Main" para el resto de los bloques. Igualmente, dentro de este tipo de bloque se encuentran funciones específicas, como la función de arranque (OB100) o las interrupciones cíclicas (OB30) que inician programas de forma periódica con un intervalo de tiempo definido.
- **Bloque de función (FB).** Son bloques programables con memoria, es decir, estos pueden almacenar de forma permanente el valor asignado a una variable en un bloque de datos de instancia (DB) aunque el tratamiento de la FB finalice.
- **Función (FC).** Las funciones son bloques programables sin memoria, por lo tanto, las variables con las que se trabaja son de tipo temporal que se almacenan en una pila de datos local que reinicia los valores cuando la ejecución concluye. En caso de querer almacenar datos de forma permanente, se debe recurrir a los bloques de datos globales.

- **Bloque de datos (DB).** Dentro de estos bloques globales o de instancia se encuentran almacenadas las variables declaradas en los bloques de función.

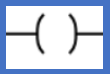
Históricamente, la herramienta STEP 7, desde la que se realizaba la programación de los autómatas, contaba con tres tipos de lenguajes de programación para los bloques OB, FB y FC:

- **KOP.** Lenguaje de programación gráfico basado en esquemas de contactos (*Ladder Diagram* o diagramas de escalera) similar a esquemas de circuitos, donde los elementos de contactos y bobinas se agrupan en segmentos. Uno o varios segmentos constituyen el área de instrucciones de un bloque lógico.
- **AWL.** Es un lenguaje de programación textual, basado en una lista de instrucciones. Se trata de un lenguaje más complejo, orientado contextualmente a la máquina, donde las instrucciones equivalen a los pasos con los que la CPU va ejecutando el programa.
- **FUP.** Es un lenguaje gráfico basado en diagramas de funciones que utilizan álgebra boolena para la representación de la lógica.

Con el paso de la programación de STEP 7 a TIA Portal, se implementó un nuevo lenguaje textual, conocido somo **SCL**, similar a lenguajes de alto nivel como C++ o Python. Este nuevo lenguaje está orientado al acceso y almacenamiento de datos en posiciones de memoria con un código de operación simple.

Cabe destacar que la elección del lenguaje **KOP** permite la utilización conjunta del lenguaje de contactos, con segmentos programados en **AWL** o **SCL**.

El lenguaje normalmente utilizado para la programación de autómatas (y que se va a usar para el desarrollo de este proyecto) es el diagrama en escalera o Ladder, **KOP.** En él se representan las funciones haciendo uso de contactos que permiten realizar operaciones lógicas con bits, entre los principales elementos de contacto de encuentran los siguientes:

SÍMBOLO	NOMBRE	DESCRIPCIÓN
	Contacto Normalmente Abierto	Representa un contacto abierto si la variable asociada tiene el valor lógico 0, que se cierra cuando la variable pasa a valer un 1 lógico.
	Contacto normalmente cerrado	Representa un contacto cerrado si la variable asociada tiene el valor lógico 0, que se abre cuando la variable pasa a valer un 1 lógico.
	Bobina de asignación	Simboliza el valor de una variable de salida asignada a la bobina.

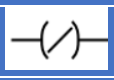
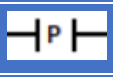
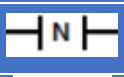
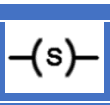
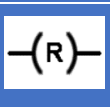
		Cuando se activa la bobina, la variable de salida pasa a tener él valor lógico 1. Si la bobina no se encuentra activa, la variable de salida tendrá el valor lógico 0.
	Bobina negar asignación	Invierte el resultado lógico y lo asigna a la variable indicada.
	Consulta de flanco ascendente	Detecta cuando el estado lógico de una variable pasa de 0 a 1.
	Consulta de flanco descendente	Detecta cuando el estado lógico de una variable pasa de 1 a 0.
	Activación de salida	Activa el operando asignado, estableciendo un 1 lógico cuando la entrada de la bobina es 1.
	Desactivación de salida	Desactiva el operando asignado, estableciendo un 0 lógico cuando la entrada de la bobina es 1.

Tabla 1. *Simbología del lenguaje KOP.*

Capítulo 4.3. MATLAB y App Designer

Capítulo 4.3.1. Introducción

Una vez implementados los robots y un autómata programable a la instalación, surge la necesidad de implementar una herramienta que cumpla la funcionalidad de un interfaz hombre-maquina (HMI), que permita la gestión global de la célula con una interfaz gráfica simple y con rápido acceso a las configuraciones y principales parámetros de la propia instalación. Para ello se ha considerado la opción de desarrollar una aplicación en MATLAB que cumpla con estas premisas.

[31] MATLAB (abreviatura de "*MATrix LABoratory*") es una herramienta matemática de cómputo numérico que combina un lenguaje de programación propio (lenguaje M) con un entorno de desarrollo integrado (IDE).

Su uso está ampliamente extendido en el ámbito universitario y en centros de desarrollo e investigación, gracias a su gran potencial a la hora de trabajar con matrices y algoritmos, representación de datos y funciones, así como la posibilidad de crear interfaces graficas de usuario o GUIs (*Graphical User Interface*) y la capacidad de comunicación con dispositivos hardware y con otras herramientas software con diferentes lenguajes.

Una de las principales características de este software, que lo dotan de una escalabilidad sin precedentes son sus *tooolboxes*, con las que se pueden agregar con facilidad herramientas para el trabajo casi en cualquier campo:

- Matemáticas, Estadística y Optimización.

- Sistemas de control.
- Computación paralela.
- Desarrollo de aplicaciones.
- Robótica.
- Internet de las cosas, IoT (“*Internet of things*”).
- Visión artificial y procesamiento de imagen.
- Redes neuronales.
- Inteligencia artificial y machine learning.
- Mecatrónica.

Capítulo 4.3.2. Descripción del entorno Matlab.

Para este proyecto se cuenta con la versión R2020b de Matlab.

Al ejecutar este programa nos encontramos con una ventana principal por defecto, dividida en diferentes sectores. Desde esta ventana se pude acceder a todas las funcionalidades y toolboxes que MATLAB ofrece.

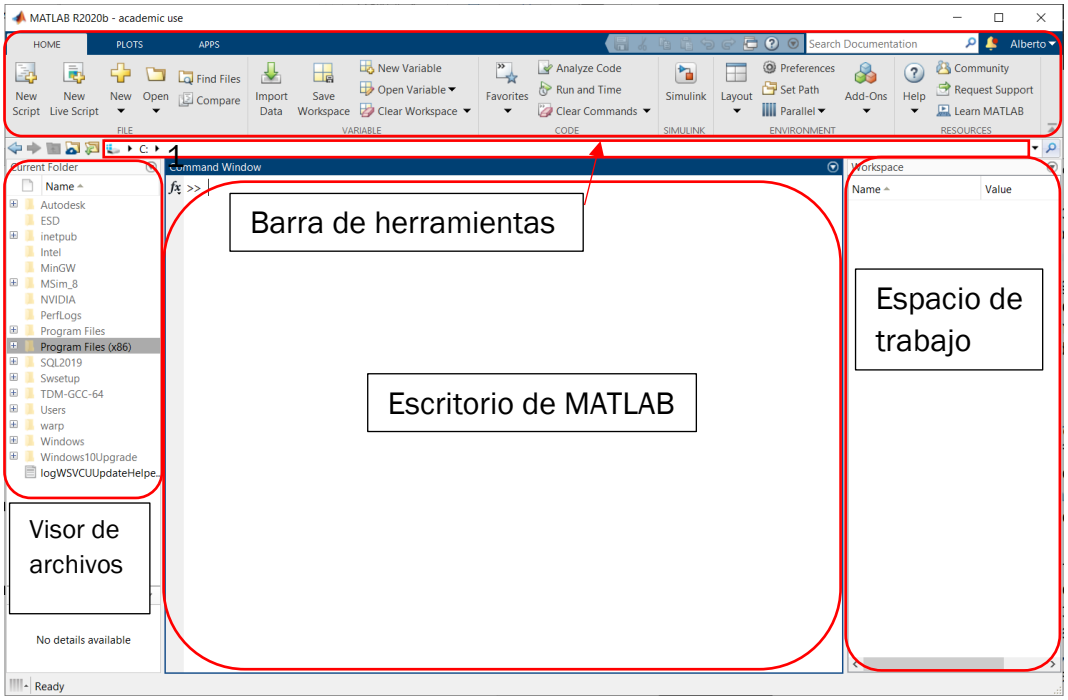


Figura 31. *Venta principal MATLAB.*

Sobre la *Figura 31.* Venta principal MATLAB. se explican las diferentes regiones en que se encuentra dividida esta ventana. Esta ventana puede ser configurada y personalizada por el usuario, por lo que no siempre tiene porque mostrar esta apariencia.

Al apartado principal de esta ventana, situado en el centro, es el escritorio de MATLAB, donde se pueden alojar el resto de las ventanas y componentes, ofreciendo una gran flexibilidad en la forma de organizar este espacio. En este caso se encuentra situado en esta región la ventana de comandos, desde donde se ejecutan interactivamente las instrucciones y donde se muestran los resultados.

En la región derecha se localiza el espacio de trabajo base, donde se encuentran listadas todas las variables que en un instante concreto están definidas en la memoria del programa, así como su valor.

La ventana situada en la zona izquierda contiene el visor de archivos que nos permite navegar por los diferentes directorios de nuestro equipo, permitiendo la posibilidad de agregar al *Path* el directorio desde el que se desea trabajar. Matlab solo puede ejecutar ficheros desde el directorio actual que se encuentra seleccionado (1) o desde el *Path*.

En la zona superior nos encontramos con una barra de herramientas contextual en función de si nos encontramos en la pestaña *HOME, PLOTS o APPS*.

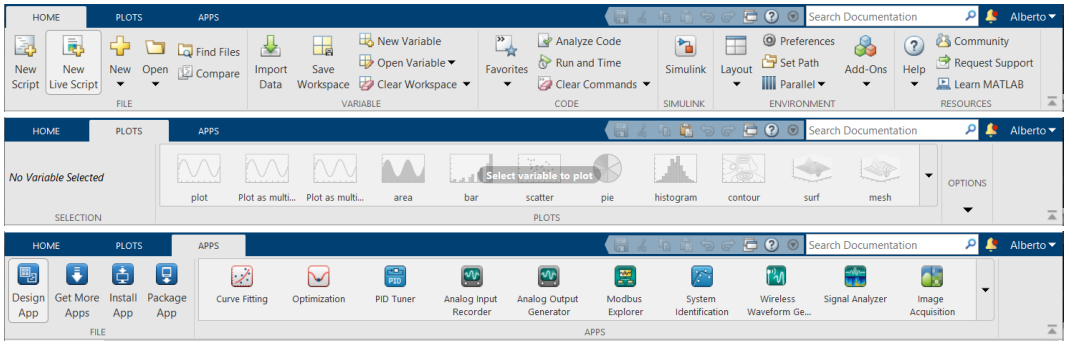


Figura 32. *Pestañas de la barra de herramientas de MATLAB.*

En la pestaña *HOME* se encuentras las principales funcionalidades de creación de archivos y gestión de variables, así como accesos directos a los ajustes del medio como el ajuste de la vista del escritorio de MATLAB, la gestión de *toolboxes* o el acceso directo a una de las herramientas más utilizadas en MATLAB como es *Simulink.*

Dentro del conjunto *PLOTS* se encuentras las opciones comúnmente requeridas para la gestión de gráficas 2D y 3D, que Matlab pone a disposición del usuario.

Por último, se encuentra en la opción *APPS* accesos directos a todas las *toolboxes* que por defecto vienen instaladas con MATLAB, entre ellas se encuentra el acceso rápido a *App Designer*.

Capítulo 4.3.3. Descripción del entorno AppDesigner.

Históricamente el entorno para diseño de interfaces gráficas en MATLAB ha sido la herramienta *GUIDE,* que proporcionaba un método de codificación de funciones que definían el comportamiento de la aplicación, de tal forma que el código y el diseño de la interfaz se mostraban por separado. En la versión R2016a se introdujo la funcionalidad *App Designer* que vinculaba las partes de diseño y código en una versión integrada del editor de MATLAB en un mismo entorno para el desarrollo de interfaces.

[32] *App Designer* permite crear apps profesionales sin necesidad de amplios conocimientos como de desarrollador de software. Esta herramienta integra las dos principales tareas en la creación de aplicaciones: la distribución de elementos gráficos propios de una interfaz de usuario (GUI) y la programación de su comportamiento interactivo.

Existen diferentes métodos de acceso a *App Designer:*

- A través de la ventana de comandos de MATLAB, escribiendo el comando ***appdesigner.***
- En la pestaña HOME de la barra de herramientas a través de la siguiente secuencia: ***New → App.***
- Desde la pestaña Apps de la barra de herramientas pulsando el icono ***Design App.***

Independientemente del método elegido, se despliega una ventana emergente donde se nos dan diferentes posibilidades: abrir una aplicación guardada, escoger una plantilla predefinida o abrir un modelo en blanco para crear la aplicación desde cero.

Una vez abierta la herramienta de *App Designer* el usuario se encuentra con un entorno de trabajo que se divide en dos vistas complementarias mostradas en la *Figura 33* (Vista del editor diseño) y *Figura 34* (Vista del editor de código).

Sobre estas figuras se encuentran remarcados los principales paneles en que se dividen estas pestañas, con los aspectos más relevantes para el uso de esta herramienta.

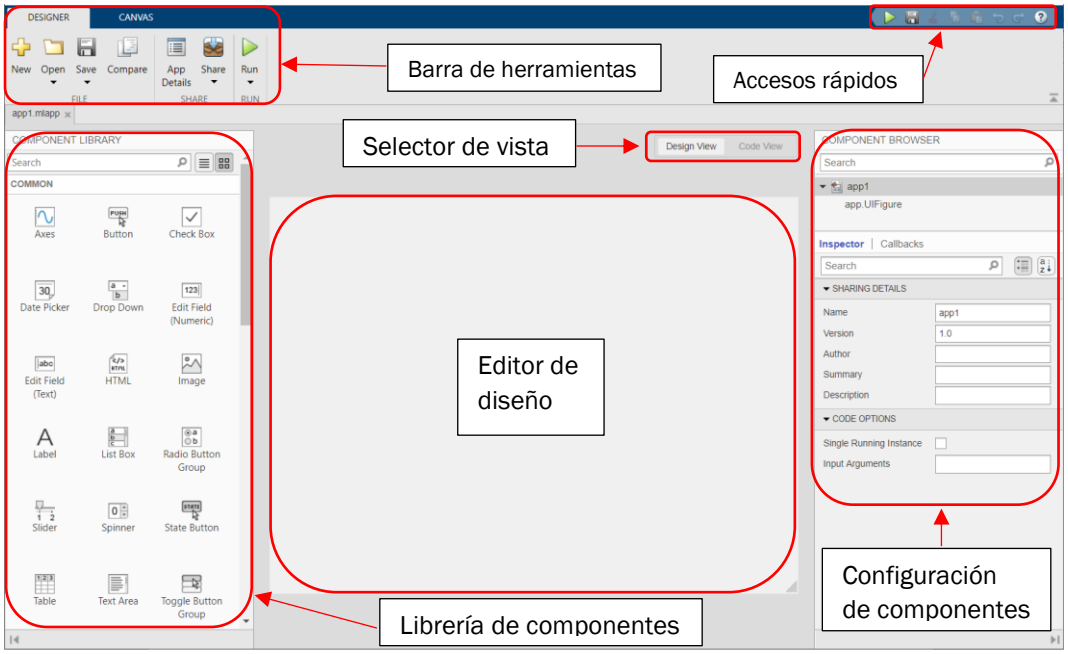


Figura 33. Entorno de trabajo App Designer con la vista de diseño activa.

Paneles con la vista de diseño activa.

1. **Librería de componentes.** Menú subdividido en diferentes áreas donde se encuentran los diferentes componentes predefinidos de los que se dispone para el diseño de la aplicación.
2. **Editor de diseño.** Área central donde se distribuyen los componentes que constituyen el aspecto gráfico de la aplicación de usuario.

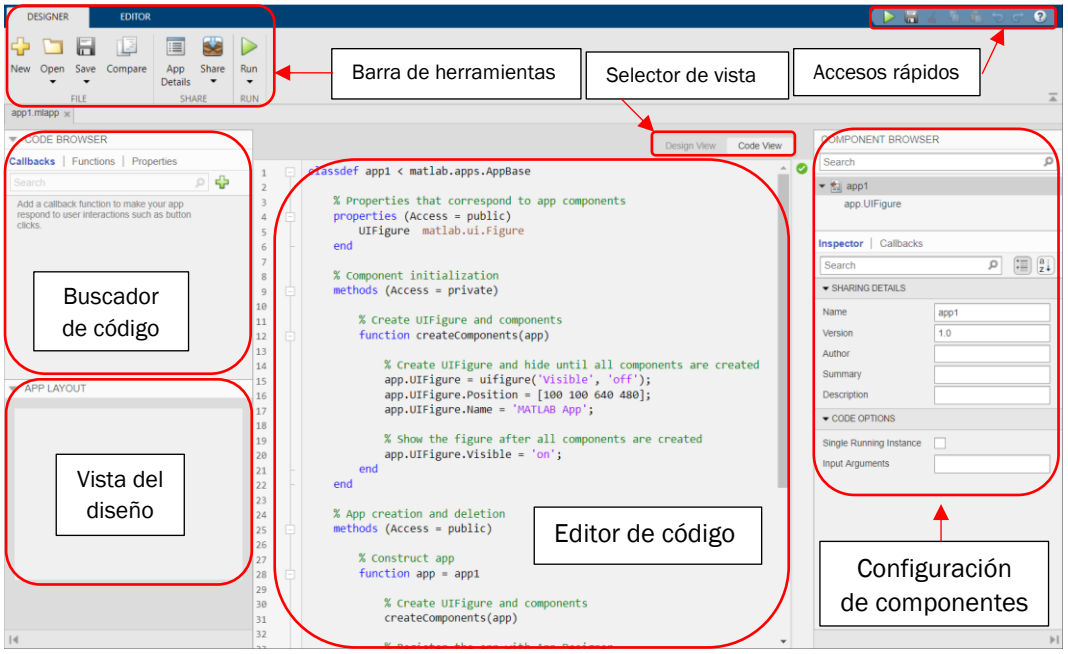


Figura 34. Entorno de trabajo App Designer con la vista de código activa.

Paneles con la vista de código activa.

3. **Editor de código.** Ventana central desde donde se lleva a cabo la programación de callbacks, funciones y propiedades de los diferentes elementos que componen la interfaz.
4. **Buscador de código.** Desde este apartado se accede de forma rápida a los diferentes bloques de programación de funciones, propiedades y callbacks que se encuentran listadas por orden de creación de estos.
5. **Vista del diseño.** En este apartado se encuentra una previsualización de la distribución de los elementos de la interfaz.

Además de los paneles propios de cada una de las vistas, existen algunos paneles comunes a ambas que facilitan el acceso a ciertas funciones.

6. **Configuración de componentes.** Ventana donde se encuentran listados los componentes que se han añadido a la interfaz. Al seleccionar algún componente se despliega en la parte inferior de la ventana una serie de propiedades y características que pueden ser modificadas por el usuario.
7. **Barra de herramientas.** Conjunto de herramientas dividido en dos pestañas, una de ellas (*DESIGNER*) común a las vistas de diseño y de código con funcionalidades para la gestión del archivo sobre el que se está trabajando.
8. **Accesos rápidos.** Formado por las funcionalidades de la pestaña *DESIGNER* de la barra de tareas que proporcionan un cómodo acceso a opciones de guardado, copiado, avance y retroceso entre otras.

La barra de herramientas cuenta con una pestaña independiente en función de si nos encontramos en la vista de edición del diseño o en la edición del código que presentan funcionalidades específicas para cada una de ellas.

El menú *CANVAS* de la barra de herramientas está enfocada a la vista de diseño, se encuentran aquí funcionalidades referentes a la disposición de los componentes y la gestión y organización del espacio disponible.

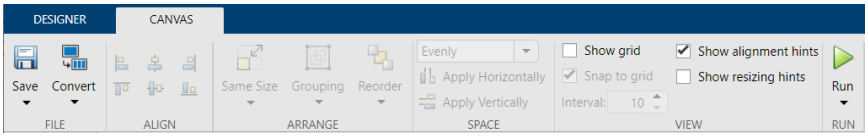


Figura 35. *Barra de herramientas en la vista de edición de diseño.*

Por otro lado, la pesteña *EDITOR* se encarga de agrupar los accesos a funcionalidades enfocadas a la programación de la interfaz.

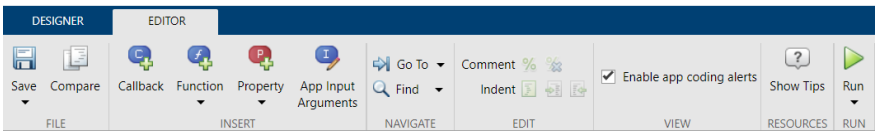


Figura 36. *Barra de herramientas en la vista de edición de código.*

Capítulo 4.3.2. OPC Toolbox

[33] La *toolbox de comunicaciones industriales* proporciona la posibilidad de acceder en tiempo real a datos OPC históricos de plantas industriales, permitiendo la lectura, escritura, y el registro de estos datos de forma directa a través de MATLAB.

Se brinda a través de esta herramienta la posibilidad de trabajar tanto desde MATLAB como Simulink con diferentes estándares de comunicación OPC: OPC DA ("*Data Access*"), OPC HDA ("*Historical Data Access*") y OPC UA ("*Unified Architecture*").

Capítulo 4.4. Autodesk Inventor

Capítulo 4.4.1. Introducción

El diseño de la célula robotizada, así como los objetos a manipular y las herramientas que se van a utilizar en el proyecto deben ser diseñados con una herramienta software de modelado de sólidos en tres dimensiones.

El programa de simulación Roboguide facilita el modelado de su gama de robots, así como una amplia librería con una serie de componentes genéricos de uso extendido en diferentes industrias como pallets, vallas para la instalación, cintas de transporte de objetos (conveyor) de diferentes dimensiones y especificaciones, maquinas CNC, etc. Para diseñar los componentes más específicos necesarios para llevar a cabo la simulación de la tarea que se pretende elaborar, se requiere de una herramienta que nos permita crear estas piezas a medida.

[34] *Autodesk Inventor* ofrece una serie de herramientas, para el modelado paramétrico de sólidos en 3D, elaboración de documentación y simulación de productos, producidas por la empresa de software *Autodesk*.

Este software proporciona una combinación de funciones de diseño mecánico avanzado con modelado paramétrico, directo, de forma libre y basado en reglas. Así mismo, dispone de una serie de herramientas para diseño de estructuras, tubos, tuberías, cables y arneses, simulación por elementos finitos, sistemas de movimiento, diseño de maquinaria, etc. Permitiendo compartir información mediante herramientas de colaboración integradas de forma nativa y satisfacer así la demanda de diseño de productos personalizados.

Con esta herramienta podemos lograr por lo tanto conceptualizar ideas, creando modelos 3D y documentarlos para un uso real, sometiendo el diseño

a una validación virtual de sus parámetros, lo que incurre de forma directa en una reducción de costes en la producción.

Capítulo 4.4.2. Descripción del entorno Autodesk Inventor

En el transcurso de este proyecto se va a hacer uso de la versión 2021 de *Autodesk Inventor Profesional* activado con la licencia para estudiantes de *Autodesk* que proporciona la Escuela de Ingenieros Industriales de la Universidad de Valladolid.

Una vez ejecutado el programa no encontramos en la ventana principal con una barra de herramientas y menús para la gestión de nuevos archivos y proyectos, una ventana para la buscada de proyectos y un buscador de archivos editados recientemente.

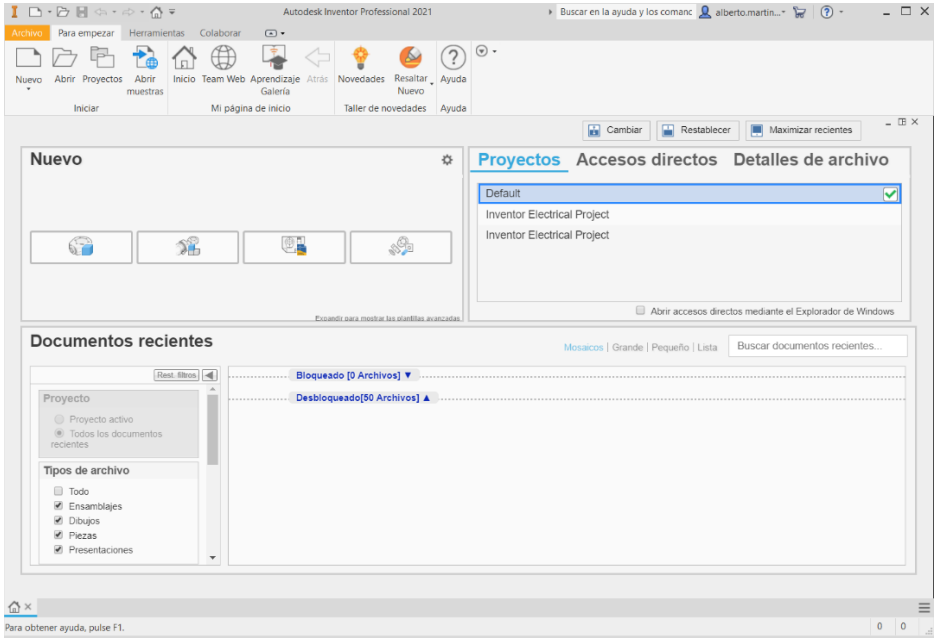


Figura 37. *Ventana principal de Autodesk Inventor.*

Desde el acceso de la barra de herramientas **Iniciar → Proyectos** lanzamos un asistente para la administración de proyectos. En esta ventana desde el botón ***Nuevo → Nuevo proyecto para único usuario*** se puede establecer un nombre para el proyecto en cuestión y seleccionar un directorio existente o crear uno nuevo, estableciendo así la ruta donde se irán guardando los archivos de piezas, ensamblajes, dibujos y presentaciones que se van creando.

Una vez creado un nuevo proyecto con su nombre y ruta asignados, podemos comenzar a diseñar las piezas que comprenden el objeto final deseado. Para ello desde el botón ***Inicio → Nuevo*** de la ventana principal, lanzamos un asistente para la configuración de nuevos archivos.

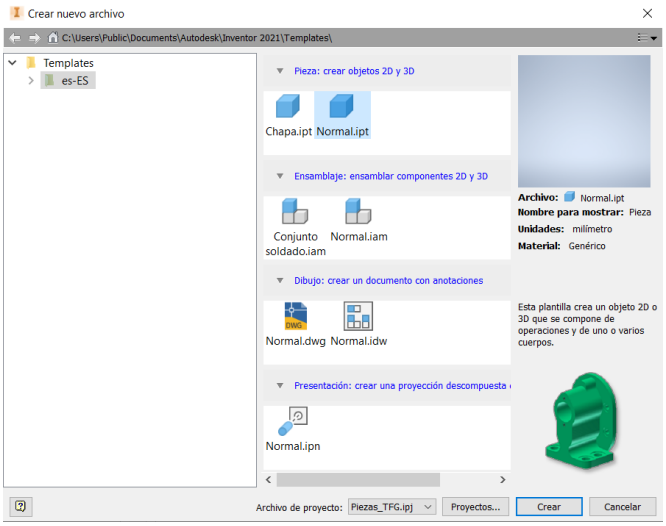


Figura 38. *Asistente de creación de nuevos objetos.*

Para los objetivos marcados y el alcance del trabajo que se va a realizar solo será necesario trabajar con objetos 3D del tipo 'Normal.ipt' y ensamblajes 'Normal.iam', de igual forma los dibujos con anotaciones serán del tipo 'Normal.idw'.

Como se ha comentado en la introducción de *Autodesk Inventor* (véase *Capítulo 4.4.1.* Introducción) este software proporciona innumerables herramientas para el diseño de prácticamente cualquier objeto que se nos pueda ocurrir. Esto ocasiona que existan un gran número de opciones fuera del alcance de este proyecto. Por lo tanto, se describen a continuación únicamente las ventanas, opciones y herramientas necesarias para la realización de este trabajo.

Capítulo 4.4.3. Creación de objetos en 3D ('Normal.ipt')

Es esencial en el diseño de piezas definir los objetos indivisibles que las van a formar. Las piezas simples consistirán en un solo objeto 3D, mientras que muchas de las piezas que se diseñan consisten un ensamblaje de varios de estos objetos.

Sobre la siguiente figura se resaltan y explican las principales secciones, herramientas y opciones para crear objetos 3D en la ventana de modelado de *Inventor.*

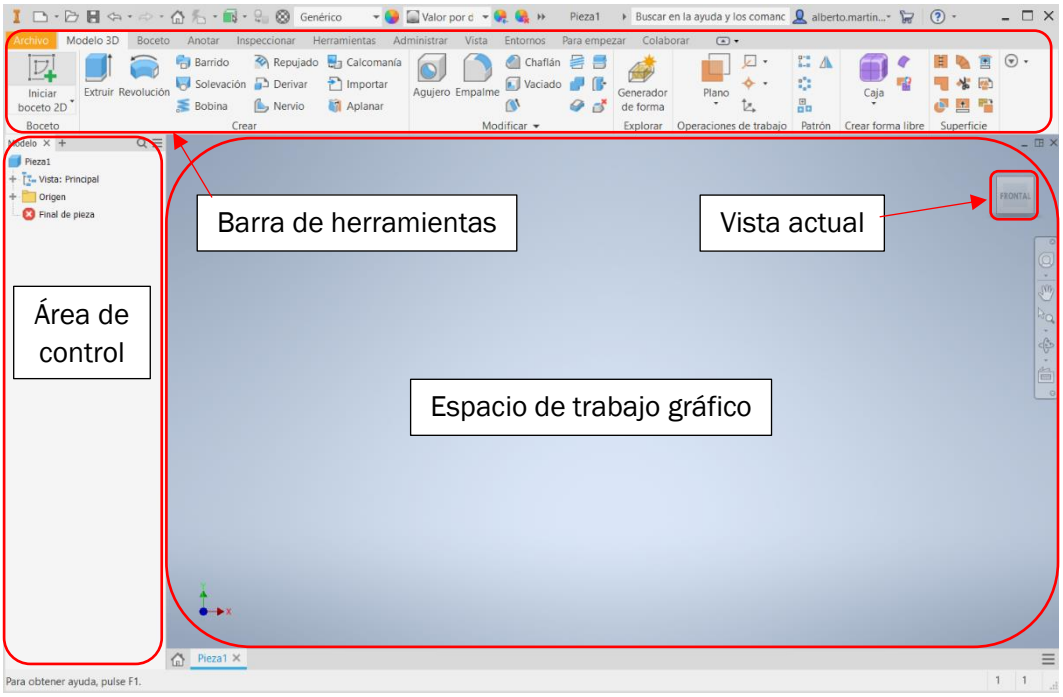


Figura 39. *Ventana principal de modelado de objetos 2D y 3D.*

Como la mayor parte del software utilizado, nos encontramos la ventana de modelado divida en 3 regiones principales: un espacio central de trabajo sobre el objeto actual, un árbol de navegación con el desglose de operaciones que se han ido realizando para conformar el objeto y una barra de herramientas contextual formada por diferentes menús de opciones específicas para las fases del diseño en las que nos podemos encontrar.

El modelado paramétrico de elementos solidos pasa por esbozar un diseño en 2D contenido en un plano, sobre el que se realizan operaciones de creación 3D que agregan volumen a un boceto bidimensional.

Sobre la barra de herramientas en la opción ***Boceto → Iniciar boceto 2D*** se despliegan en el espacio de trabajo los planos cartesianos del triedro directo que podemos escoger para comenzar con el boceto. Una vez seleccionado el plano que define el origen y orientación espacial del objeto que se va a crear, se activa el menú boceto de barra de herramientas, con las opciones que nos permiten comenzar a dibujar ilustradas en la *Figura 40*.

Figura 40. *Barra de herramientas del entorno de modelado 2D y 3D.*

El grupo con la etiqueta ***Crear*** lo forman las opciones propias de dibujado compuestas por líneas, círculos, arcos y rectángulos (cada una de estas

opciones con diferentes opciones de restricción para un trabajo más cómodo en función de nuestras necesidades), así como proyección de geometrías de otros planos (aristas, vértices o superficies). Igualmente se incluyen aquí las opciones de remarcado de puntos y textos y la herramienta de empalme de líneas.

Dentro de los grupos ***Restringir*** y ***Modificar*** se encuentran las funciones que permiten establecer pautas de diseño y modificaciones dimensionales y geométricas acotando debidamente el boceto o realizando modificaciones de recorte, desplazamiento, giro y escalado entre otras. El establecimiento de restricciones aumenta la legibilidad y entendimiento del boceto, pues nos permite reducir el número de cotas necesarias.

Una vez dibujado, acotado y restringido adecuadamente el boceto, pulsando sobre el botón ***Terminar Boceto*** salimos de la ventana de creación bidimensional y se pasa a las herramientas de modelado 3D que agregan la dimensión volumétrica a nuestro diseño.

Figura 41. *Menú de modelado 3D de la barra de herramientas contextual.*

Entre opciones más utilizadas por regla general de este menú encontramos la extrusión, revolución, barrido, solevación, agujero y vaciado.

La herramienta de ***Extrusión*** da la posibilidad de agregar material a un boceto bidimensional formando un sólido, se trata de una operación booleana que permite dotar de profundidad a un perfil en una dirección concreta o quedarnos con la intersección entre dos bocetos.

Por su parte las opciones de ***Revolución, Barrido*** y ***Solevación*** son operaciones booleanas de mayor complejidad que permiten crear cuerpos ya sea revolucionando uno o más perfiles de un boceto alrededor de un eje en cualquier ángulo (hasta 360 grados), arrastrando uno o varios perfiles (de un mismo boceto) a lo largo de una trayectoria que atraviese el plano de dichos perfiles, o creando perfiles a partir de la transición entre dos o más bocetos mediante operaciones de suavizado o tangencia entre estos bocetos.

Las funcionalidades de ***Agujero*** y ***Vaciado*** otorgan por su parte la posibilidad de retirar material de un sólido a partir de operaciones booleanas, de forma complementaria a las operaciones anteriores agregan nuevos niveles de profundidad a los diseños.

Capítulo 4.4.4. Creación de ensamblajes ('Normal.iam')

Una vez finalizado el diseño de todos nuestros objetos tridimensionales que conforman la pieza final que se busca crear, el paso siguiente consiste en ensamblarlas para que formen un único elemento con las restricciones de movilidad pertinentes.

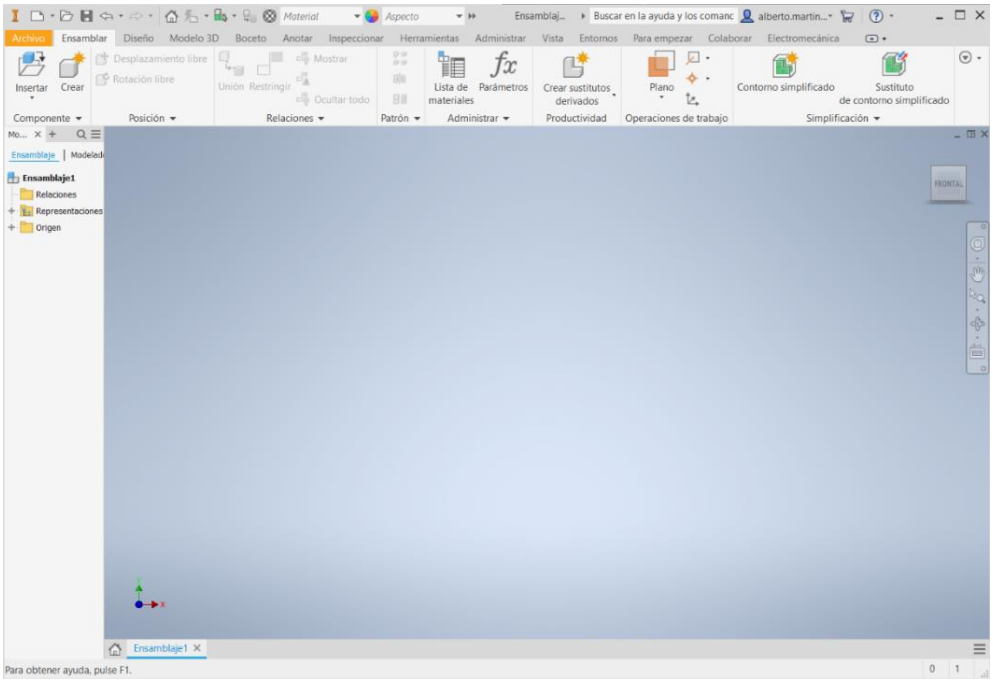


Figura 42. *Ventana de ensamblado de objetos.*

Una vez creado un nuevo archivo de ensamblaje, nos encontramos la ventana principal de ensamblado que se muestra en la *Figura 42*, y que como se puede observar el prácticamente idéntica a la ventana del entorno de creación de modelos 2D y 3D (véase en *Figura 39*), con la salvedad de que en la barra de herramientas encontramos en esta ocasión un nuevo menú específicamente orientado al ensamblaje de objetos.

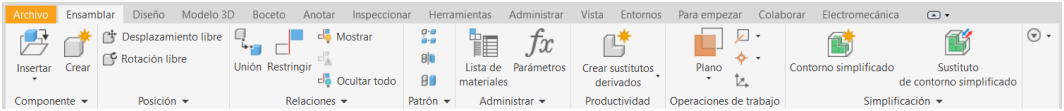


Figura 43. *Barra de herramientas del entorno de ensamblado.*

Para comenzar a añadir objetos al diseño, se dispone del icono ***Componente → Insertar*** que despliega una serie de opciones para agregar elementos que conformen nuestro ensamblaje. Estos elementos pueden ser objetos 3D previamente diseñados o ensamblajes individuales que se comportan como un único sólido.

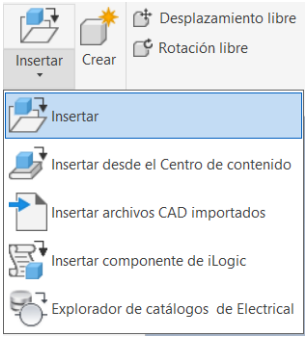
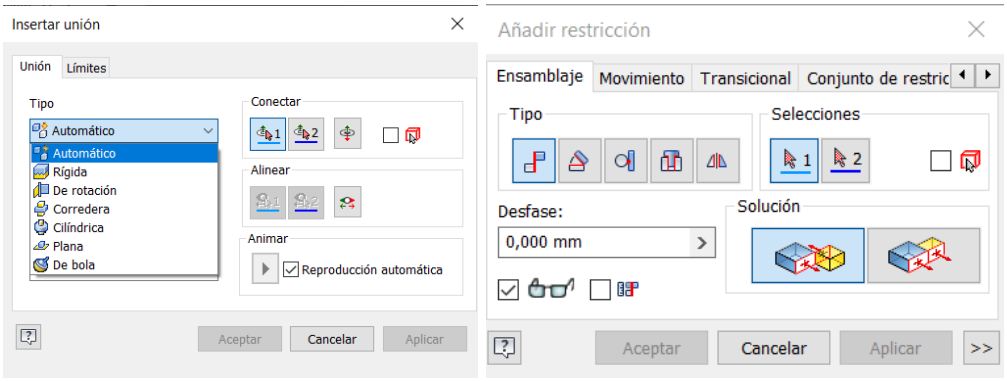


Figura 44. *Insertar nuevo objeto.*

Tras insertar nuevos elementos, estos poseen de 6 grados de libertad, tres movimientos axiales y tres rotaciones sobre sus ejes, por lo que suele definirse uno de los elementos como fijo, lo que permite que el ensamblaje se realice en torno a este elemento.

Para posicionar el resto de los elementos que forman la pieza o mecanismo final contamos con las herramientas de ***Desplazamiento libre*** y ***Rotación libre*** del apartado ***Posición*** de la barra de herramientas, que permiten aproximar los diferentes objetos a su posición final.

Dentro del grupo ***Relaciones*** se encuentran las funcionalidades que permiten definir las zonas de contacto entre diferentes objetos, así como las limitaciones de movimientos entre ellas.



A) **B)**

Figura 45. A) *Ventana de Unión.*
B) *Ventana de Restricciones.*

En la ventana de configuración de ***Restricciones*** se encuentran las opciones que permiten limitar los grados de libertad de forma individual, tanto aspectos posicionales entre elementos como movimientos y transiciones. Por su parte la herramienta ***Unión*** combina de forma simultánea dos o más restricciones de la ventana anteriormente descrita, definiendo de esta forma un rango de movilidad entre los elementos que se van a unir, lo que facilita el trabajo.

Ambos tipos de relaciones cuentan con un apartado en sus ventanas que permite especificar si se desea una distancia o un ángulo de desfase en la operación que se va a realizar con componentes seleccionados.

Capítulo 4.5. KEPServerEX 6

Capítulo 4.5.1. Introducción

La necesidad de comunicación entre los diferentes softwares de simulación que se van a utilizar requiere de una herramienta que permita la manipulación de la información basada en tecnologías OPC.

[35] KEPServerEx es una solución de conectividad que adquiere cualquier dato procedente de un proceso industrial y los deja a disposición de plataformas de supervisión, monitorización, control o análisis de información en un formato estándar y seguro.

La disponibilidad de múltiples drivers integrados en el programa permite una alta escalabilidad, pudiendo conectar casi cualquier dispositivo o sistema según se van incrementando las necesidades del proyecto. Todo ello con un interfaz común, lo que aumenta su usabilidad.

Este software es capaz de proveer la información de diferentes protocolos de comunicación como OPC UA, OPC DA, MQTT ("*Message Queing Telemetry Transport*") o REST ("*REpresentational State Transfer*") entre otros.

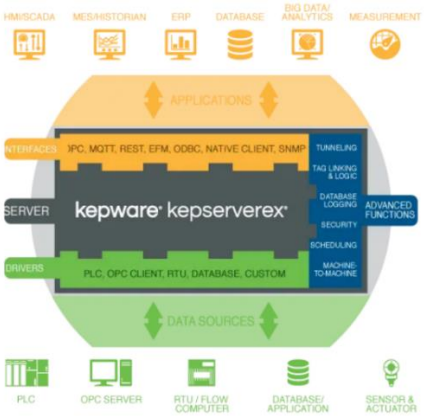


Figura 46. *Esquema de comunicación mediante KEPServerEx.*

Capítulo 4.5.2. Descripción del entorno KEPServerEX

Para el desarrollo de este proyecto se ha utilizado una versión de prueba gratuita de KEPServerEX 6 disponible en la web del proveedor, la cual cuenta con una única limitación referente al tiempo máximo que el servidor se

encuentra activo de forma continuada. El tiempo límite de uso continuado es de dos horas, pero una vez expirado este tiempo se puede reiniciar el servidor y continuar utilizándolo.

En la siguiente figura se observa la ventana principal del software.

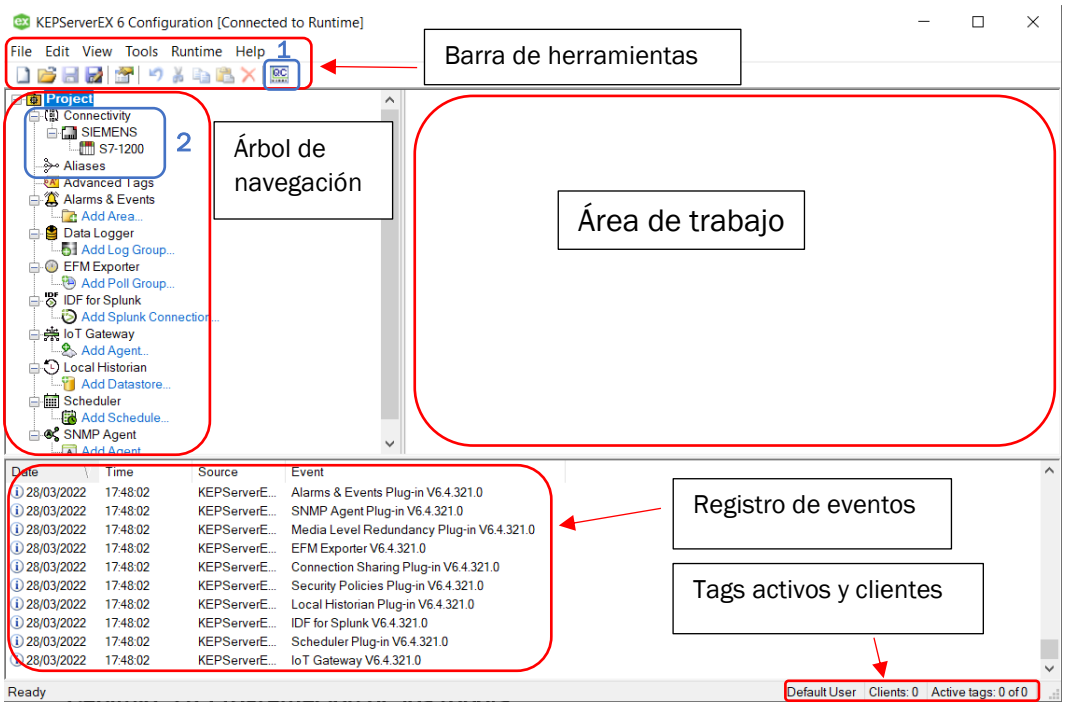


Figura 47. *Ventana principal KEPServerEx.*

Sobre esta figura se muestran y explican los aspectos que componen esta ventana, además se resaltan y enumeran aquellas áreas que presentan una relevancia mayor para el uso propuesto.

En la zona superior se encuentra algunos menús clásicos para la gestión del proyecto abierto y una barra de herramienta con accesos rápidos a las principales funciones que nos encontramos en los menús. La opción más relevante en este caso (1) se trata de un cliente OPC que integra el propio software y que permite observar en tiempo real el correcto funcionamiento del servidor, así como el valor de los datos y señales en cada instante.

La parte izquierda la ocupa el árbol de navegación del proyecto donde se sitúan diferentes aspectos de configuración del servidor, donde el más importante es el aspecto de conectividad (2), desde donde se añaden y gestionan los drivers y tags de los dispositivos que se deseen configurar.

La región central está comprendida por el área de trabajo, donde en función de que opción del árbol del proyecto tengamos seleccionada nos encontráramos diferente información y elementos relevantes para la configuración del servidor en estos aspectos.

Por último, en la zona inferior se dispone de un registro de eventos que permite conocer en todo instante el estado del servidor, así como los posibles fallos con una breve descripción del problema. Además, en la región inferior derecha se sitúa un contador del número de clientes conectados al servidor y las etiquetas (tags) activas en cada momento.

Capítulo 4.5.3. Configuración del servidor

Una vez instalado el software KEPServerEX, se dispone dos dependencias, KEPServerEX Configurator y KEPServerEX Administrator. Al ejecutar esta última se agrega un incono con el logo de KEPServer en la barra de herramientas de Windows. Pulsando con el botón derecho del ratón sobre este incono se puede acceder a la configuración OPC-UA del servidor.

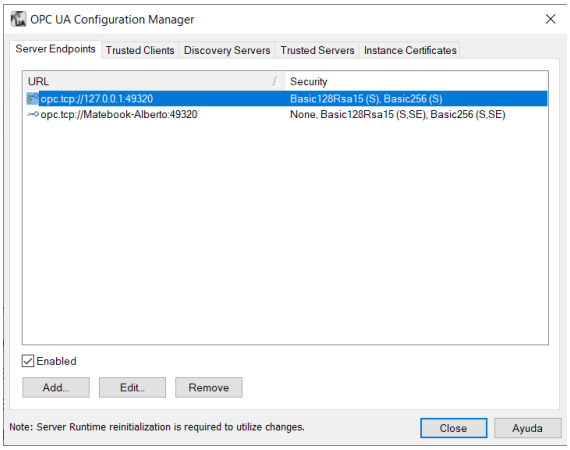


Figura 48. *Configuración del servidor OPC-UA.*

Se muestra en esta ventana aquellas direcciones por las que se puede acceder al servidor OPC. La primera de ellas es la dirección de autorreferencia y la segunda la dirección de la maquina donde se está ejecutando el servidor. Existe la posibilidad de agregar direcciones de red a mayores para el acceso al servidor desde diferentes IP.

Para las conexiones con el servidor se usará la dirección de acceso **opc.tcp://Matebook-Alberto:49320**, ya que todas las instancias que se van a conectar van a ser ejecutadas desde la misma máquina.

Con una doble pulsación sobre esta dirección se accede a la configuración de parámetros que se muestran en la *Figura 49*, donde se encuentra el puerto de acceso y los protocolos de seguridad que deben cumplir los clientes que quieran acceder al servidor.

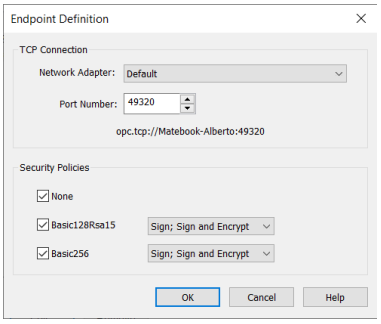


Figura 49. *Configuración de la dirección de acceso y protocolos de seguridad.*

Capítulo 4.5.4. Canales de comunicación

En la pestaña de conectividad del árbol de navegación resaltado en el punto (2) de la *Figura 47,* se encuentran los canales de comunicación que han sido definidos para el servidor. Estos canales representan la comunicación entre los dispositivos deseados y el ordenador, por lo tanto, deben definirse los drivers correctos en la configuración del canal para que esta comunicación sea exitosa.

Para crear estos canales se debe pulsar el botón derecho del ratón sobre la pestaña ***Conectivity → New Channel***. Esta acción despliega una ventana emergente para la configuración de un nuevo canal, donde se debe seleccionar el controlador del dispositivo que se va a utilizar.

En este caso como se dispone del PLC simulado con un puente TCP/IP generado a través del programa NetToPLCSim se selecciona ***Siemens TCP/IP Ethernet***. La configuración final del canal se muestra en la *Figura 50*.

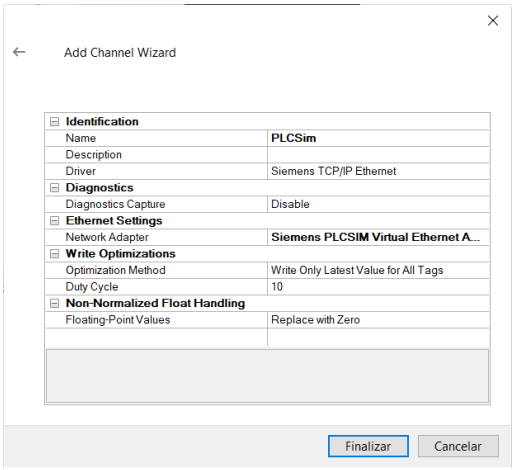


Figura 50. *Configuración del canal para PLCSim.*

En esta ventana se muestran las principales características del canal configurado para la comunicación con el simulador PLCSim:

- Se define un identificador único (***Name***) para los dispositivos que se incorporan un mismo canal de comunicaciones.
- El apartado del controlador (***Driver***) se define en función del tipo de dispositivos asociados al canal.
- En el adaptador de red se selecciona el adaptador virtual del simulador PLC que tiene su dirección IP y puerto de comunicación especifico definidos.
- El método de escritura debe estar abierto a todos los dispositivos para que los 3 clientes puedan acceder a las variables del PLC. Si este apartado se configura de otra forma, solo podrá ser modificado el valor por el último dispositivo que haya modificado la etiqueta.

Capítulo 4.5.5. Definición de los Tags

Una etiqueta o tag representa la información relativa a una variable del PLC.

Una vez agregados los dispositivos al canal de comunicaciones, sobre el área de trabajo, pulsando el botón derecho se muestra la opción ***NewTag***. Esta opción despliega una ventana para la configuración de tags que se muestra en la *Figura 51*.

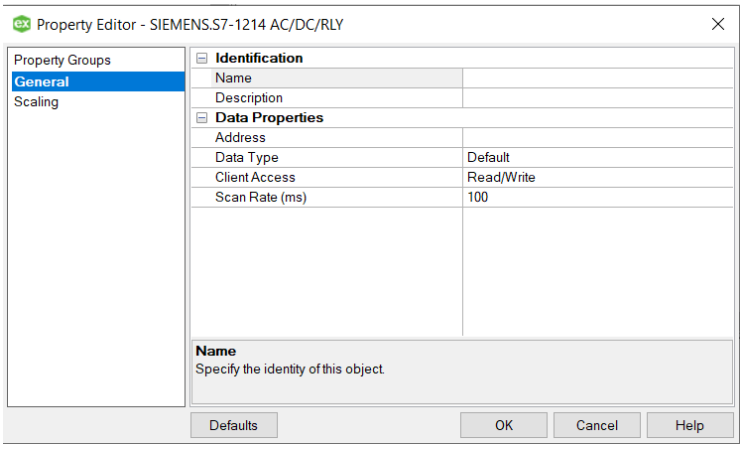


Figura 51. *Configuración de Tags.*

El primer parámetro (***Name***) es el identificador de la variable para el dispositivo seleccionado. No tiene por qué coincidir con el nombre de la variable que se haya establecido en TIA Portal.

La dirección (***Address***) y el tipo de dato (***Data Type***) deben ser establecidos conforme a que variable del PLC queremos asignar a esta etiqueta. Debe establecerse permiso de acceso de los clientes (***Client Access***) en lectura y escritura (***Read/Write***) para que los clientes puedan escribir los valores correspondientes en cada momento. El tiempo de escaneado (***Scan Rate***) se define como el tiempo entre actualizaciones consecutivas del valor de la señal.

[Página en blanco por convenio]

CAPITULO 5. DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN ADOPTADA

El trabajo con software de simulación, y no con dispositivos físicos, ofrece la posibilidad de crear desde cero un nuevo espacio para la instalación de la estación robotizada, seleccionar los dispositivos más convenientes para cada tarea y situar los elementos que forman la célula como al autor le parezca más adecuado; pues no existe un espacio predefinido sobre el que se deba trabajar o al que se deba adecuar la estación, ni una serie de dispositivos que se deban reutilizar para abaratar costes. Todo esto dota de una gran libertad a la hora de trabajar de forma creativa en el diseño, sin perder nunca el objetivo de realizar un proyecto lo suficientemente realista para poder así mostrar el potencial de las herramientas con las que se va a trabajar.

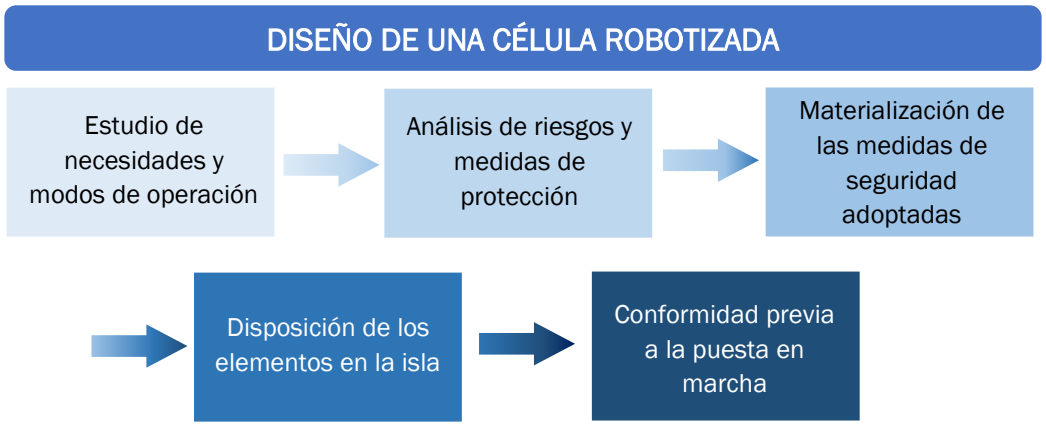


Figura 52. *Diagrama de fases de diseño.*

Durante los siguientes capítulos se realizará una descripción de la metodología de trabajo en el diseño de los elementos que forman la estación y su implementación enmarcada en la normativa de seguridad vigente para entornos robotizados.

Capítulo 5.1. Estudio de seguridad en entornos robotizados

Antes de comenzar con el diseño de la célula robotizada es necesario realizar un estudio sobre la normativa que regula el trabajo con robots en materia de seguridad y los riesgos que entraña no cumplir con la normativa. Puesto que los robots son maquinas que a menudo suelen combinar su ciclo de trabajo con operarios por lo que es indispensable garantizar una serie de medidas para evitar accidentes.

Capítulo 5.1.1. Riesgos laborales en el uso de robots industriales

Con el objetivo de aumentar la comprensión de la normativa de seguridad en el uso de robots industriales, se van a evaluar a continuación los principales riesgos específicos que dan lugar a accidentes en este campo.

Riesgo por contactos mecánicos.

Se da cuando se accede a regiones peligrosas mientras el robot se encuentra en modo de funcionamiento automático, o si el robot excede el área restringida si no se ha limitado el alcance.

También puede ocurrir este tipo de accidente durante el periodo de ajuste y programación, en la realización de pruebas, si las protecciones se encuentran abiertas o son inefectivas. Igualmente pueden darse debido a un arranque intempestivo durante tareas de mantenimiento.

En la colaboración directa con robots, cuando operario y robot acceden al mismo espacio de trabajo al mismo tiempo, se deben aumentar las precauciones y las medidas de seguridad debido a la elevada exposición a riesgos de este tipo.

Riesgo de atrapamiento.

Los movimientos del robot, tanto en modo manual como automático pueden atrapar a los trabajadores, entre el propio brazo y los elementos de la isla que lo rodean.

Riesgo de proyección de materiales.

Se debe disponer de protecciones resistentes ante impactos, no solo del propio robot, si no de materiales o piezas que puedan salir eyectadas durante el transcurso del proceso. Igualmente se deben aplicar medidas para contrarrestar el levantamiento de partículas de polvo, chispas, etc.

Riesgos térmicos.

La existencia de atmosferas inflamables o explosivas por disolventes, así como superficies calientes o el polvo metálico requieren de sistemas de protección frente a quemaduras por contacto, control de llama, etc.

Riesgos eléctricos.

Debe cumplirse la normativa de seguridad eléctrica por posibles contactos con regiones activas mal aisladas durante labores de mantenimiento. Igualmente, la existencia de circuitería, condensadores y sistemas que trabajan con alto voltaje requieren de aplicar las precauciones pertinentes.

[37] Las especiales características de trabajo de los sistemas robotizadas y su funcionamiento automático habitualmente no requieren de presencia humana, por ello, aproximadamente un 90% de los accidentes que se dan en las líneas robotizadas ocurren durante las operaciones de mantenimiento, ajuste o programación, mientras que tan solo un 10% ocurre durante el funcionamiento normal.

Capítulo 5.1.2. Normativas de seguridad en entornos industriales robotizados

A partir de los años 90, con la creciente expansión de la robótica en todos los ámbitos comenzaron a surgir las primeras normativas legales de seguridad para la regulación en el uso de robots y sistemas robotizados, con el fin de garantizar la seguridad de las personas y reducir al máximo los peligros.

Las principales normativas que surgieron durante esta época son:

- Normativa internacional **ISO 10218:1992**.
- Normativa americana **ANSI/RIA R15.06-1992**.
- Normativa europea **EN 775** y española **UNE-EN 775**.

A grandes rasgos, estas normativas de carácter general recogían una sección de análisis de seguridad, definición de riesgos e identificación de fuentes de peligro y/o accidentes.

Se introduce a continuación la evolución de las normativas europea y española, pues son las que conciernen a este proyecto, en concreto el aspecto que atañe a las medidas que se deben tomar en las fases de diseño.

[38] La directiva de referencia es la ***Directiva 2006/42/CE*** del Parlamento Europea y del Consejo, del 17 de mayo de 2006, relativa al uso de maquinaria y según la cual: “los riesgos que conlleva la utilización de las máquinas cubiertas por la presente Directiva, conviene establecer los procedimientos de evaluación de la conformidad con los requisitos esenciales de salud y seguridad. Estos procedimientos deben diseñarse de acuerdo con la importancia del peligro inherente a tales máquinas”.

Toda maquinaria que cumpla los requisitos técnicos mínimos especificados en esta directiva irá acompañada del marcado europeo, necesario para su comercialización.

CE

Figura 53. *Marca de certificación o marcado europeo.*

Bajo esta directiva europea se armonizan las diferentes normativas estatales de referencia en nuestro país, las cuales tratan de dar solución a los riesgos presentes en el uso de robots y sistemas industriales robotizados. Las versiones más actualizadas de estas normas son las siguientes:

[39] **UNE-EN ISO 10218-1:2012**

Esta primera parte de la norma recoge una serie de especificaciones sobre los requisitos y directrices para llevar a cabo un diseño inherentemente seguro, las medidas de seguridad pertinentes, así como las pautas e instrucciones para un correcto uso de robots industriales.

[40] **UNE-EN ISO 10218-2:2011**

Partiendo de la norma ISO 10218 se especifican los requisitos para la correcta integración de sistemas robotizados industriales según se define en la primera parte de la norma para una o varias celdas de robots industriales. Dentro del apartado integración se incluyen aspectos informativos de diseño, fabricación, instalación funcionamiento, mantenimiento y retirada de servicio de robots o celdas robotizadas.

Capítulo 5.1.3. Medidas de seguridad en fase de diseño de instalaciones industriales robotizadas

La directiva de maquinaria del parlamento europeo, así como la normativa UNE reúnen una serie de requisitos en materia de seguridad y salud, indispensables para poder certificar que la celda robotizada y las maquinas que la componen cumplen con la normativa de seguridad en el momento de la puesta en marcha de la instalación. De forma generalizada se pueden citar las siguientes:

Barreras materiales y mecanismos de acceso.

Con el objetivo de restringir el acceso a la célula de personal no autorizado, se debe limitar el espacio de trabajo mediante barreras que abarquen el perímetro de la célula. Tal y como se muestra en la *Figura 54.*

Estas protecciones se encuentran reguladas según la norma **UNE 81-600-85**, donde se indican las dimensiones mínimas (altura mínima de 1,80 metros) que deberían tener todas las barreras siempre que sea posible, así como la especificación del mallado que deben tener para evitar el acceso de las extremidades de los operarios.

Figura 54. *Vallado de seguridad en una instalación industrial.*

Para reforzar la seguridad, es indispensable instalar dispositivos acceso mediante código, detección fotoeléctrica del personal, sensores de presencia o proximidad, u otras tecnologías que garanticen una parada segura inmediata de los elementos que se encuentren en movimiento en este espacio.

Figura 55. *Mecanismo de acceso magnético en una célula robotizada.*

Barreras inmateriales.

Cuando las necesidades de la instalación no permitan la instalación de barreras materiales, ya sea porque su utilización fuera ineficiente o ineficaz o por imposibilidad para el cumplimiento de la norma en su dimensionado, se optará por la instalación de dispositivos laser y fotoeléctricos que sean capaces de interrumpir el suministro eléctrico de la maquinaria.

La instalación de estos dispositivos debe realizarse de tal forma que no puedan ser eludibles, es decir, siempre que se desee entrar a la instalación se deberá obstruir el camino de uno o varios rayos luminosos (visibles o invisibles) que

forman la barrera, de tal forma que sea imposible el acceso sin la detección por parte de esta cortina de luz para que el mecanismo sea seguro y eficaz.

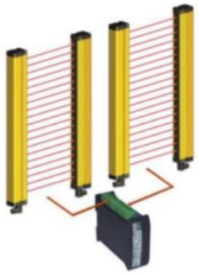


Figura 56. *Barreras de seguridad fotoeléctricas.*

Estas cortinas fotoeléctricas deben contar con mecanismos de auto chequeo del circuito para corroborar la alineación y existencia de suciedad en las lentes, para evitar cortes del suministro eléctrico por una falsa detección.

Señalización de la instalación

Dotando a la estación de una correcta señalización según indica la normativa se incrementa la seguridad, indicando el estado en que se encuentra la instalación, de tal forma que el operario pueda conocer en todo momento (mediante diferentes señales, luminosas y/o acústicas) si el robot se encuentra en funcionamiento automático.

Dispositivos de intercambio.

Cuando los operarios deban acceder al espacio de trabajo del robot para recoger o depositar piezas se den habilitar ciertas áreas que permitan que este intercambio se realice de forma segura mediante mesas giratorias, cintas, etc.

Si los objetos que se van a manipular son de grandes dimensiones y el operario debe entrar completamente al área de carga y descarga, deben utilizarse dispositivos de protección vertical, donde cobran un gran protagonismo los sistemas inmateriales anteriormente mencionados.

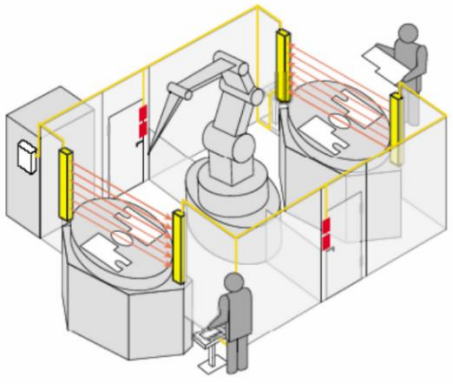


Figura 57. *Dispositivos de intercambio de piezas.*

Zonas de reparación.

Se deben habilitar zonas de reparación y mantenimiento fuera de la región de trabajo del robot, pero que sean alcanzables por él. Esta región tendrá que estar asegurada frente a movimientos automáticos del robot mediante diferentes dispositivos.

Movimiento condicionado

En aquellos casos en los que el operario deba entrar a la instalación, situándose en el área de acción del robot durante el funcionamiento automático (para alimentar de nuevas piezas al robot, por ejemplo), se debe realizar la programación de tal forma que no se efectúe ningún movimiento durante ciertos instantes de tiempo.

Sistemas de mando y órganos de accionamiento.

Es esencial dotar a la instalación de elementos de mando para el operador alrededor de la célula de trabajo con el fin de evitar situaciones peligrosas y permitan evitar posibles riesgos, respondiendo ante las premisas básicas de seguridad y fiabilidad.

Estos dispositivos deben ser diseñados de forma que un error en alguno de estos sistemas no comprometa la seguridad de la célula impidiendo la parada de la maquinaria. Igualmente, los elementos de mando no deben ser alcanzables por ningún elemento o piezas de las maquinas.

Se debe disponer además de dispositivos de accionamiento situados de forma visible y reconocible, que permitan el arranque de la instalación de manera totalmente segura y que no se puedan accionar por error.

Parada

Todas las maquinas que formen la instalación deben contar con un sistema que permitan la parada total de su funcionamiento de forma prioritaria a cualquier otra orden que se le dé. Se pueden distinguir diferentes tipos de parada:

- Normal: paradas segura de alguna (o todas) las funciones de la máquina, sin corte del suministro eléctrico en el instante de realizar la parada, pero posibilitando cortar la alimentación una vez realizada la parada. Enfocada a labores de mantenimiento.
- Operativa: parada segura durante un tiempo determinado sin corte de energía.

- Emergencia: parada del proceso para evitar crear algún tipo de riesgo. Conlleva la activación de algún protocolo de seguridad para poder rearmar la instalación.

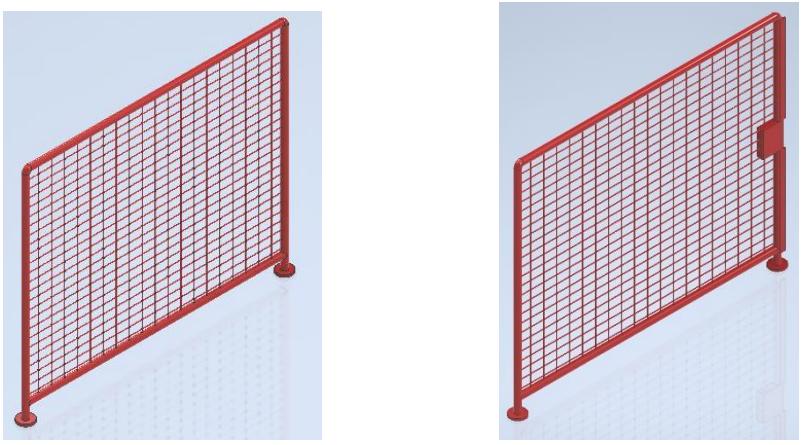
Cuando el dispositivo accionador de la parada cese su acción el sistema no reemprenderá su trabajo de forma automática, si no que se permitirá que mediante otro mecanismo de acción se pueda volver a arrancar la instalación.

Capítulo 5.2. Descripción y diseño de los elementos de seguridad

Se ha creado desde cero un modelo de vallado conforme a la maquinaria de la que se quiere proteger, con un estilo clásico y simétrico, cumpliendo con la normativa de seguridad vigente.

Conforme a las fichas técnicas de los robots estos pueden alcanzar una cota máxima en Z de 3490mm en caso del modelo R-2000iC/125L, por lo que para aplicaciones donde esta altura vaya a encontrarse en un punto de paso, se debe implementar un vallado capaz de cubrir la proyección de un objeto desde dicho punto.

El modelo diseñado alcanza una cota máxima de 2000mm, ya que las tareas que van a desempeñar las máquinas que componen la instalación, no llegan a superar nunca esta altura. Adicionalmente, para que los operarios puedan supervisar la instalación durante su funcionamiento, se cuenta con un mallado lo suficientemente compacto como para que no entren las extremidades de los operarios y a su vez que no exista la posibilidad de que un objeto proyectado lo atraviese.



A) B)

Figura 58. A) *Vallado de seguridad perimetral.*
B) *Vallado con anclaje para puerta.*

Como se puede observar en la *Figura 58* existe una variante del propio vallado que se ha diseñado específicamente para incluir un cierre magnético y un refuerzo en la posición donde irá la puerta de acceso a la instalación.

La geometría base de este vallado es sencilla, consta de perfiles cilíndricos con codos a 90º para su unión y una barra horizontal inferior soldada a los perfiles verticales. El mallado por su parte, son cruces de elementos verticales y horizontales que forman un ángulo de 90º entre sí.

Siguiendo esta misma línea de diseño, se ha creado una puerta de acceso a la instalación, que en este caso tendrá una altura algo menor (la mínima exigida por la normativa, de 1800mm) y que cuenta con el mecanismo de acceso y cierre electromagnético.

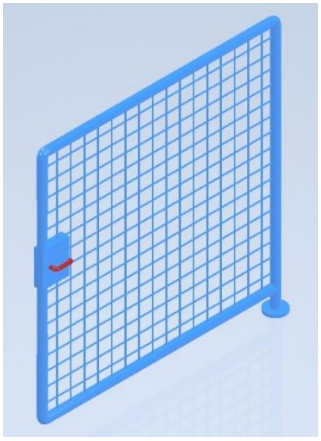


Figura 59. *Puerta de acceso a la instalación.*

Estos elementos llevan una capa de pintura reflectante especialmente llamativa y visible, delimitando sin lugar a duda el área no accesible de la instalación durante el funcionamiento automático.

Capítulo 5.3. Descripción y diseño de objetos que se van a manipular

Se realizará en el presente capítulo una descripción de las piezas diseñadas que van a ser manipuladas durante la simulación de célula para llegar al objeto final una vez sean ensambladas.

Como ya se ha comentado con anterioridad (véase *Capítulo 3.2.3. Propuesta seleccionada*), la célula que se va a diseñar tiene una funcionalidad enfocada al montaje de televisores de grandes dimensiones, por lo que es necesario tener un modelo tridimensional que incluya las suficientes piezas, para dotar a la instalación de un cierto nivel de semejanza a la realidad. Para ello el autor de este proyecto ha diseñado un modelo genérico de televisor de 75 pulgadas,

en base a las dimensiones estandarizadas que un modelo de relación de aspecto 16:9 tiene para estas pulgadas.

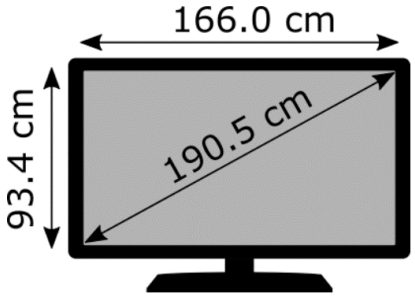


Figura 60. *Medidas estándar televisor 75 pulgadas (16:9).*

Con el diseño en Autodesk Inventor no se busca la exactitud total de las piezas creadas, si no que estas cumplan con las medidas generales del conjunto y sean fácilmente manipulables con herramientas no demasiado complejas.

Se muestra en la siguiente figura una vista explosionada del modelo que se ha diseñado, sobre el que se marcan y describen brevemente cada una de las piezas que lo forman.

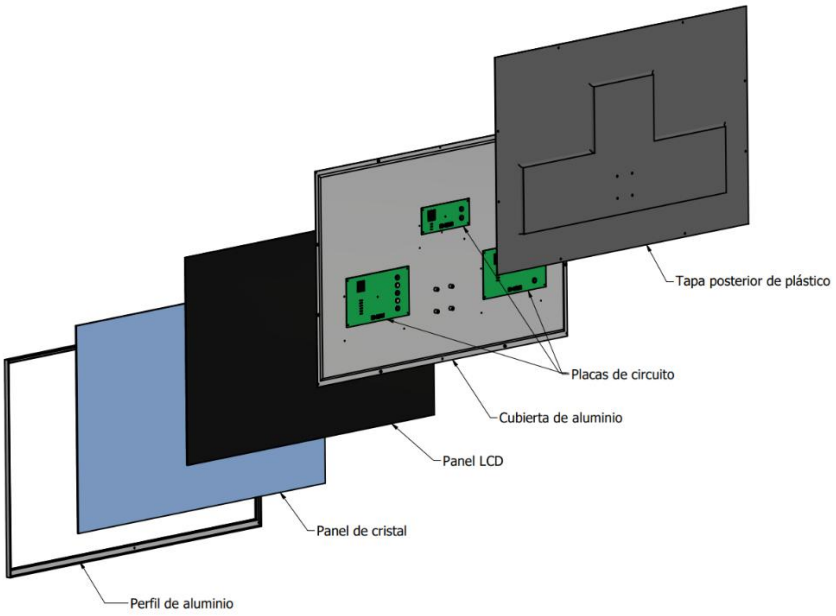


Figura 61. *Vista descompuesta de las piezas del modelo genérico de televisor creado por el autor del proyecto A. Martínez.*

Perfil de aluminio

Esta primera pieza define el espacio sobre el que se irán montando el resto de los componentes, debe ser de un material ligero pero resistente, para dotar de solidez al conjunto.

Cuenta con un espacio definido para que los paneles de cristal y LCD queden perfectamente ajustados en su posición, al igual que la cubierta posterior de

aluminio. Adicionalmente, cuenta con unos agujeros roscados, con los que todo el conjunto, una vez montado queda fijado.

Panel de cristal y Panel LCD

Para que se puedan ver imágenes, ya sean fijas o en movimiento, se cuenta con un panel de cristal líquido o LCD (Liquid Crystal Display).

Este tipo de panel lo forman un gran número de pixeles contenidos entre dos conjuntos de electrodos transparentes y dos filtros de polarización. Esto permite que ajustando la dirección particular de cada píxel gracias a los electrodos se deje pasar una cierta cantidad de luz.

Para garantizar la durabilidad del panel LCD se cuenta con un fino panel de cristal totalmente transparente, que en este caso se ha tintado con un color azul para que se distinga con mayor facilidad en la simulación.

Cubierta de aluminio

Este componente incluye en la región frontal una matriz de luces LED para retroiluminar y servir como fuente de luz al panel LCD. Además, en la parte posterior de la propia cubierta, con unos enganches de plástico se sujetan las placas ce circuito electrónico del televisor.

El conjunto perfil frontal y cubierta posterior de aluminio, dotan de resistencia al conjunto.

Placas de circuito

Se han diseñado tres modelos diferentes de forma aproximada a los que serían las placas de circuito que tiene una televisión actual, donde se encuentran los módulos de alimentación y potencia, o la gestión de los puertos de conexión entre otros.

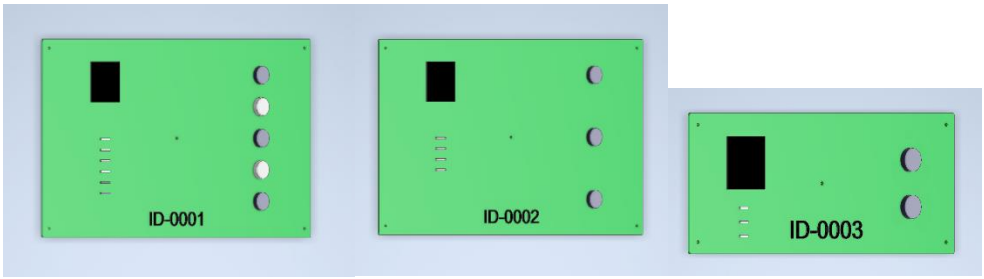


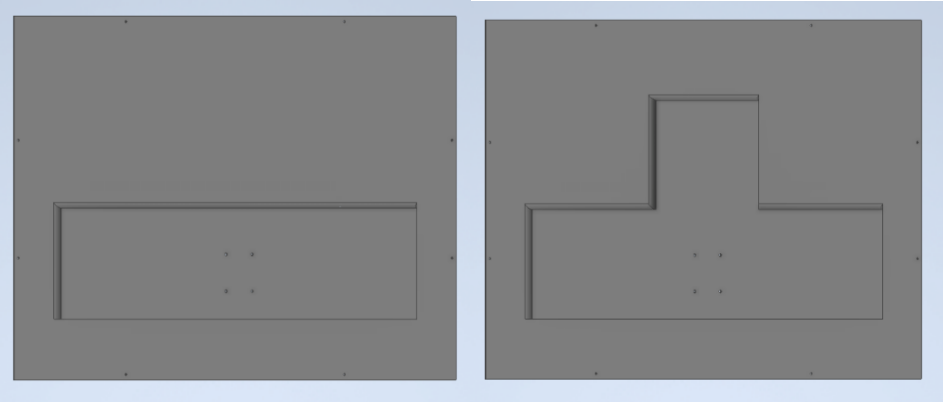
Figura 62. *Placas de circuito diseñadas por el autor del proyecto A. Martínez.*

En estos modelos no se ha buscado el realismo en su representación, si no que cuenten con elementos distintivos entre cada placa. Para ello se incluye un código de identificación (ID).

No todos los televisores que se ensambles en la instalación incluirán las tres placas, si no que, en función del modelo, se incluirán las placas con ID 0001 e ID 0002 o las 3.

Tapa posterior de plástico

Para proteger los componentes electrónicos, se tiene un último componente en la parte posterior, en este caso diseñado en plástico, pues se trata de una pieza de grandes dimensiones, por lo tanto, su construcción en este material reduce el peso de esta considerablemente.



A) B)

Figura 63. A) *Tapa posterior modelo 1.*
B) *Tapa posterior modelo 2.*

Al tener dos modelos de televisor ensamblados en la misma línea, se ha diseñado una tapa posterior para cada modelo en función del número de placas de circuito con las que cuenta cada modelo.

Soporte móvil

Adicionalmente a los componentes que se van a ensamblar, se ha diseñado un soporte que se irá desplazando por una cinta de transporte hasta las diferentes posiciones de trabajo de los robots.

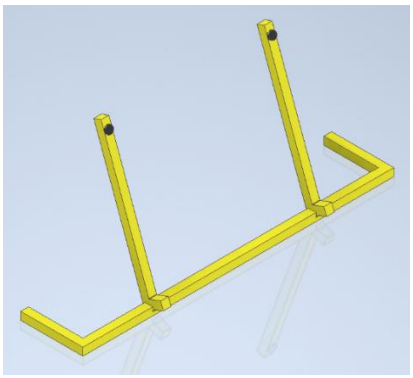


Figura 64. *Soporte móvil diseñado por el autor de proyecto A. Martínez.*

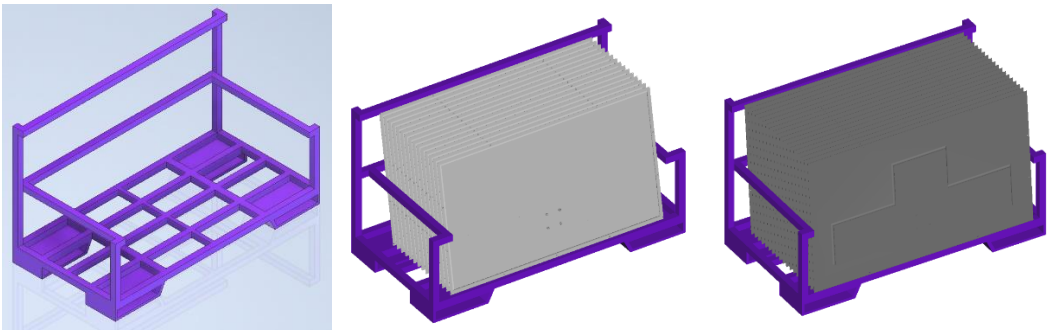
La geometría de esta pieza permite que el proceso de ensamblaje sea claramente visible a través de la rejilla de las vallas de seguridad y controlable por los operarios que se encuentran fuera de la instalación sin necesidad de cámaras para observar el proceso.

Así mismo, tal y como se puede observar en la *Figura 64,* el diseño cuenta con una prolongación horizontal que evita que el propio soporte pueda volcar durante su movimiento o cuando los robots depositen las piezas en el mismo.

Contenedores

Algunos de los componentes llegaran a través de conveyors o cintas de transporte y otros se dispondrán en la estación en contenedores situados en una posición específica para que los robots puedan acceder a las piezas que estos contienen.

Las tapas de plástico, así como la chapa posterior de aluminio son los componentes que se han seleccionado para disponer en los contendores. Para ello se han creado dos tipos de contenedor, siguiendo la misma línea de diseño, pero variando las medidas para ajustarse a cada objeto.



A) *B)* *C)*

Figura 65. A) *Contenedor para piezas diseñado por el autor A. Martínez.*
B) *Contendor con chapa posterior de aluminio.*
C) *Contenedor con tapas de plástico modelo 2.*

Capítulo 5.4. Descripción y diseño de las herramientas de los robots

Cuando un robot industrial, debido a las formas irregulares del objeto que va a manipular, unas dimensiones muy pequeñas o grandes, un bajo peso o un peso muy elevado de la pieza, se tiene acceso complicado al propio objeto, por lo tanto, es necesario diseñar una herramienta específica para su manipulación.

En concreto para este proyecto, donde se van a manipular piezas de elevadas dimensiones, pero bajo paso para las capacidades de carga para las que se han diseñado los robots que se van a utilizar, se ha diseñado una herramienta de ventosas de vacío.

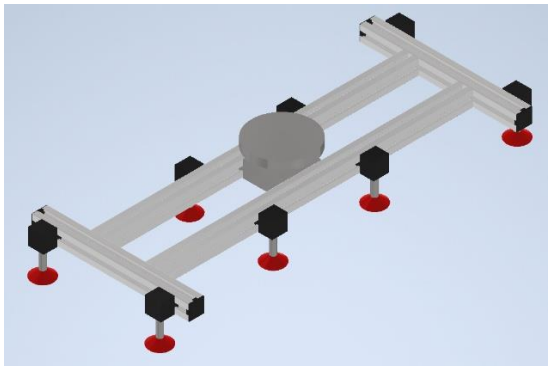


Figura 66. Herramienta de ventosas de vacío diseñada por el autor de proyecto A. Martínez.

Este tipo de herramienta utiliza la presión negativa de aire para adherirse a superficies, creando un vacío parcial que se traduce en una fuerza de succión. La diferencia entre la presión de la atmosfera y la baja presión en la cavidad interior de la ventosa mantiene adherido el objeto a la ventosa.

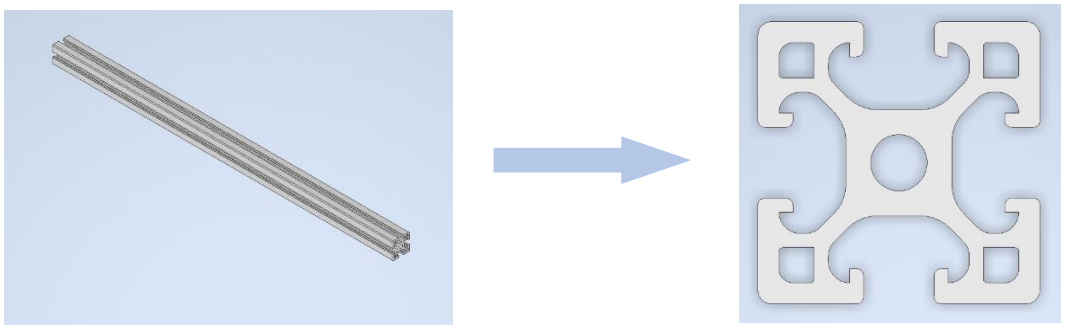


Figura 67. Perfil de aluminio 40x40mm diseñado por el autor del proyecto A. Martínez.

Para crear esta pieza se han diseñado unos perfiles de 40x40 mm que unidos entre si constituyen el cuerpo de la herramienta. En estos perfiles se acoplan las ventosas planas cóncavas, así como una pieza especialmente diseñada para ser acoplada en el extremo del último eslabón del robot.

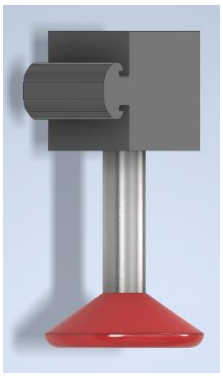


Figura 68. Acople y ventosa plana cóncava de vacío diseñada por el autor del proyecto A. Martínez.

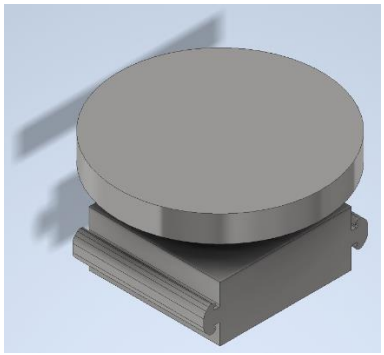


Figura 69. *Adaptador de perfiles 40x40mm para robots diseño por el auto del proyecto A. Martínez.*

Esta será la herramienta utilizada en dos de los robots para manipular, los paneles de cristal y LCD, así como la chapa de aluminio con uno de ellos, y las tapas posteriores de plástico con otro de los robots.

Adicionalmente, para el último robot se ha diseñado otra herramienta a medida con dos funcionalidades, una de ventosa de vacío para manipular las placas de circuitos electrónicos y otra para el atornillado de las tapas posteriores de plástico, que garantice la fiabilidad y velocidad de esta operación.

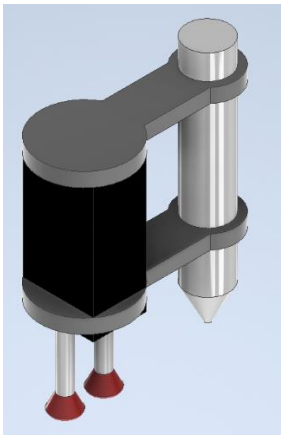
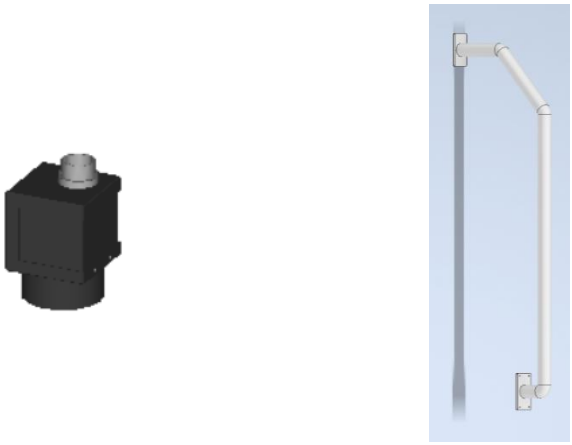


Figura 70. *Herramienta de manipulación y atornillado diseñada por el autor A. Martínez.*

Esta herramienta tiene la particularidad de que las ventosas de vacío deben ser retractiles para no interferir en la tarea de atornillado.

El robot que trabaja con la herramienta mostrada en la *Figura 70.* Herramienta de manipulación y atornillado diseñada por el autor A. Martínez. tiene la peculiaridad de estar apoyado por visión artificial para el reconocimiento de las placas de circuito. Por lo tanto, la propia cámara, así como el soporte donde va a ir anclada la cámara son dos objetos CAD adicionales. La cámara se trata de uno de los modelos 2D proporcionados en el propio software de simulación Roboguide, mientras que el soporte ha sido diseñado específicamente para ser anclado a la cinta de entrada de placas.



A) B)

Figura 71. A) Modelo CAD de la cámara KOWA SC130EF2 B/W proporcionado por FANUC. B) Soporte para cámaras diseñado por el autor del proyecto A. Martínez.

Capítulo 5.5. Diseño final de la estación en Roboguide

Para completar el diseño de la instalación se va a recurrir a la librería de elementos de la herramienta HandlingPRO de Roboguide (véase 'Capítulo 4.1. Roboguide') donde encontramos multitud de elementos diseñados y optimizados para la simulación. Pues es importante reseñar que los elementos diseñados externamente, pese a que se pueden importar sin problemas en diferentes formatos (.igs, .stl, etc.) estos ralentizan la simulación, al estar formados por multitud de caras y aristas sobre los que se va a aplicar una simulación dinámica.

En este conjunto de librerías encontramos un apartado de cintas de transporte con diferentes dimensiones. Estos componentes se añaden a la instalación para simular la llegada de las piezas que componen el televisor que se va a ensamblar.

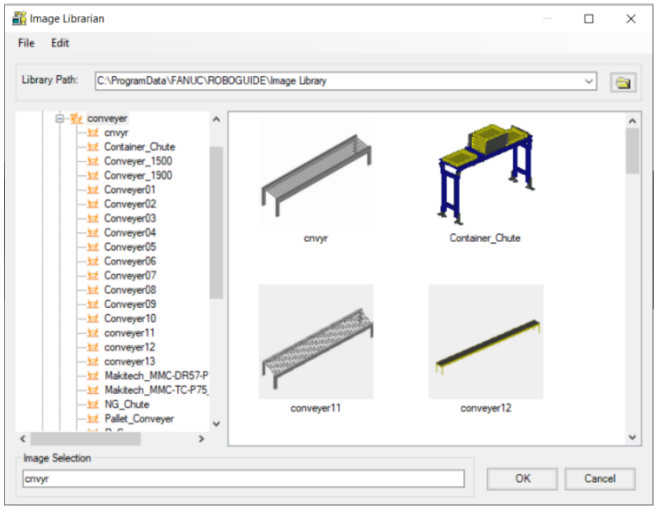


Figura 72. Librería de cintas de transporte.

En específico se van a añadir de esta librería dos conveyor, desde los que llegaran los paneles de cristal y LCD y dos cintas de transporte, una para la llegada de las placas de circuito y una a mayores donde irá el soporte móvil sobre el que se ensambla el televisor.

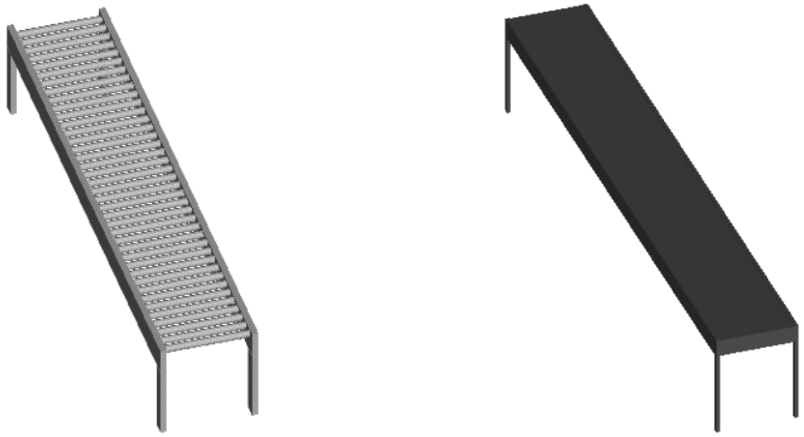


Figura 73. A) *Conveyor de llegada de componentes.*
B) *Cinta de transporte de piezas.*

Estos elementos tienen unas dimensiones específicas con las que han sido diseñados que pueden no encajar con las tareas que se van a desarrollar, por lo tanto, se debe acceder al menú de propiedades de cada una de las cintas y conveyor, para ajustar la escala en cada uno de los ejes cartesianos, de forma que las dimensiones finales sean optimas en relación con los objetos que transportan y la estación que se va a diseñar.

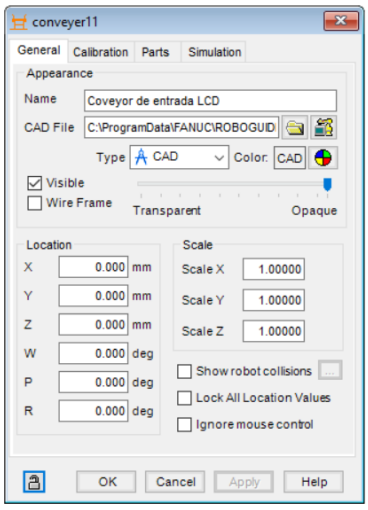


Figura 74. *Ventana de propiedades de los componentes.*

Los elementos fundamentales para realizar el proyecto son los robots que se van a utilizar en la estación. En este caso, para añadirlos, desde el menú ***Cell*** de la barra superior del software (véase *Figura 18. Software de simulación*

Roboguide.) se debe seleccionar la opción ***Add Robot → Single Robot – Serialize Wizard.***

Esta opción despliega un menú emergente de configuración de robots, donde se pueden modificar multitud de propiedades del robot que se desea añadir a la estación.

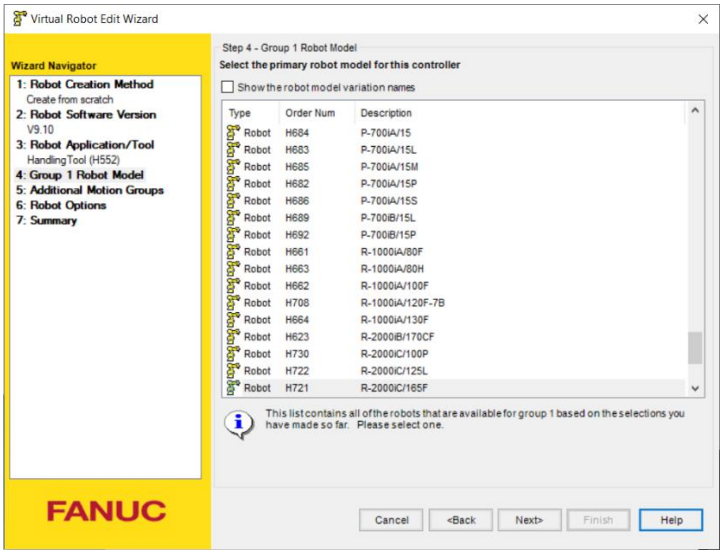


Figura 75. *Menú de selección del modelo de robot.*

Las propiedades configurables van desde la versión del software que deseamos que tenga el robot (en este caso será V9.10 R-30iB Plus para todos los robots), pasando por el tipo de herramienta por defecto y el modelo de robot que se va a añadir hasta la lista de paquetes de software con funcionalidad adicionales con las que pueden contar los robots de la marca.

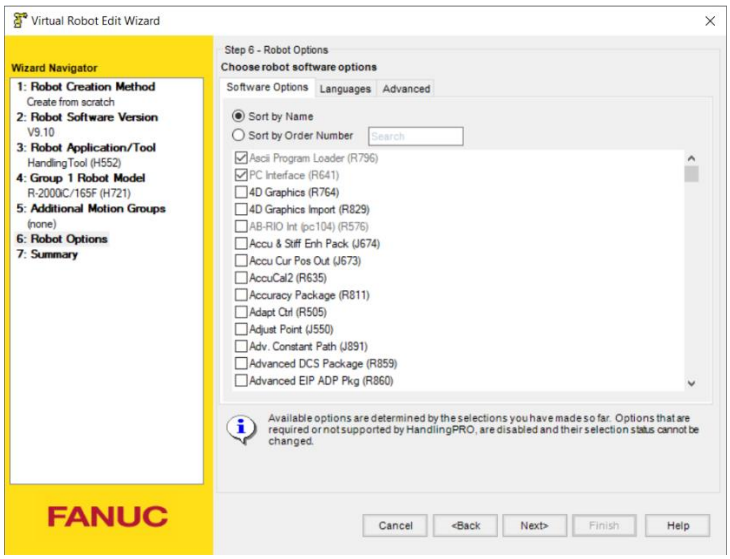
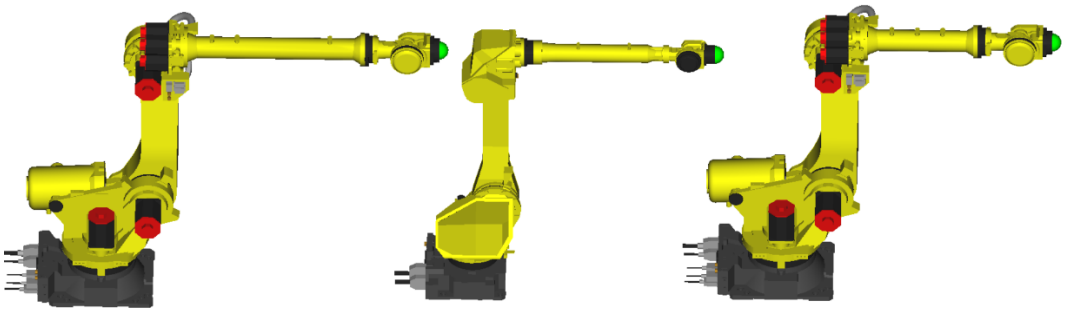


Figura 76. *Menú de funcionalidades software adicionales.*

Una vez finalizada la configuración de los robots estos se añaden a la estación con sus modelos CAD diseñados y optimizados por el propio fabricante.



A) *B)* *C)*

Figura 77. A) *Modelo CAD del robot R-2000iC/125L*
B) *Modelo CAD del robot M-710iC/45M*
C) *Modelo CAD del robot R-2000iC/165F*

Una vez revisados y analizados los elementos principales que forman parte de la instalación, se muestra a continuación como estos se han distribuido en el espacio de trabajo en el que se desarrolla el proceso de montaje.

En la siguiente figura se muestra una recopilación de imágenes con diferentes vistas de la estación. En estas imágenes el lector puede observar una serie de elementos que han sido revisados con anterioridad, como pueden ser los edificios situados a la entrada y/o salida de las cintas de transporte, o los controladores de los robots entre otros, pues estos elementos tienen únicamente un propósito estético y no funcional.

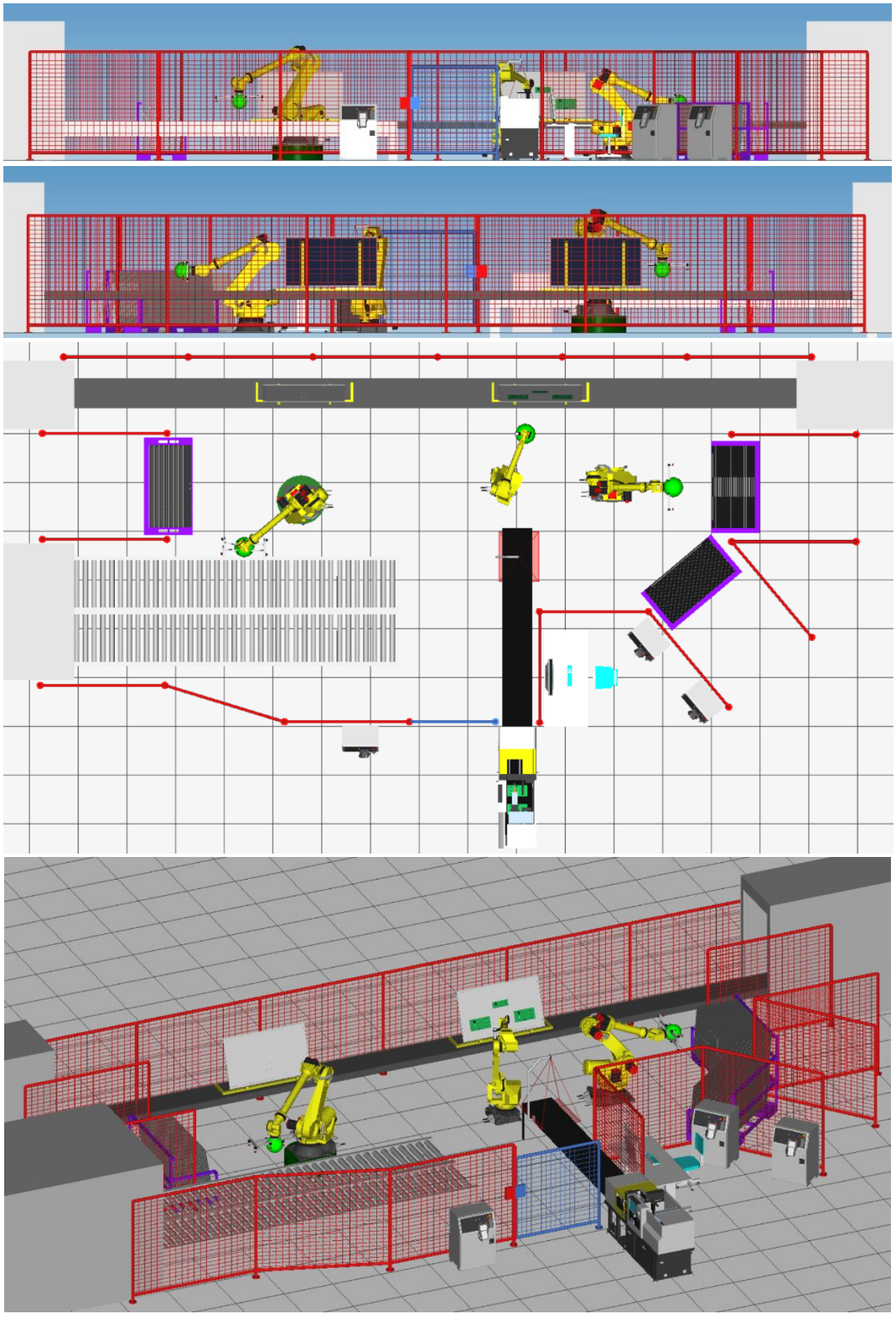


Figura 78. Estación de trabajo diseñada en Roboguide.

CAPÍTULO 6. PROGRAMACIÓN Y CONFIGURACIÓN DE LOS ENTORNOS DE TRABAJO

Una vez definido y analizado el diseño de la instalación, se desarrollará en los siguientes capítulos como se han programado y configurado los diferentes componentes de la estación, así como el PLC en cargado de controlar la estación y la interfaz hombre-maquina desde la que el usuario puede gestionar las funcionalidades que se proporcionan.

Como ya se ha comentado en capítulos previos, los softwares que se van a utilizar dan un gran abanico de posibilidades, pero requieren de un alto nivel de conocimiento para su utilización correcta en la realización de proyectos. En el *CAPITULO 4. HERRAMIENTAS SOFTWARE UTILIZADAS* se realizó una descripción introductoria de cada una de estas aplicaciones, sobre las que se profundizará a continuación.

Cabe destacar que en este informe se han dividido las explicaciones detalladas del desarrollo del proyecto por cada software que se ha utilizado, con el fin de facilitar la comprensión por parte del lector. Pero a la hora de ser desarrollado, se ha ido realizando la programación y configuración de las aplicaciones de forma paralela para ir comprobando el correcto funcionamiento del conjunto.

Capítulo 6.1. Desarrollo en Roboguide

La realización de este proyecto, partiendo desde cero sin ninguna base previa, requiere que se realicen una serie de configuraciones en las maquinas que se van a utilizar, así como en los robots, y no solo la programación de trayectorias para la simulación de movimientos.

En primer lugar, se deben definir las herramientas de los robots, utilizadas para la manipulación de los objetos y la labor de montaje. Al ser estas herramientas diseñadas por el autor y por lo tanto conocidos al detalle aspectos como dimensiones y geometría, su definición y ajuste será esencialmente sencilla.

Una vez definidas las herramientas, se deben ajustar los sistemas de referencia o bases de trabajo que van a ser utilizadas, sobre los que la herramienta se puede posicionar y reorientar.

Con las herramientas y sistemas de usuario correctamente configurados, se puede pasar a la programación de sentencias y trayectorias del robot para la labor o labores que van a desarrollar durante su funcionamiento.

Todos los robots que componen la célula son gobernados por un autómata conectado a la simulación mediante el protocolo de comunicaciones OPC por

lo que los módulos de entradas y salidas digitales que se van a utilizar deben ser configurados adecuadamente.

Respecto a las maquinas que compone la célula, se deben definir algunos parámetros como la velocidad de movimiento, las partes que se contienen o transportan y los tiempos de simulación entre otros.

Capítulo 6.1.1. Definición de los puntos centrales de las herramientas

Para posicionar el robot en un punto del espacio alcanzable, si este está definido en coordenadas cartesianas, las cotas de dicho punto serán las del TCP (‘*Tool Center Point*’) de la herramienta activa en el momento de realizar el movimiento respecto al origen del sistema de coordenadas cartesianas activo en este momento. Por defecto se define el TCP del robot en el extremo del eslabón del eje más externo del robot, conocido como TCP_0.

El TCP, también conocido como ‘*User tool*’ (UTOOL) en Fanuc, es por lo tanto el punto de origen de referencia de la herramienta activa. En función del eje de ataque (orientación del TCP respecto al UTOOL_0), se pueden distinguir dos tipos de herramienta.

- **Herramienta simple.** Tipo de herramienta en la cual el eje de ataque es paralelo al eje Z del TCP por defecto.

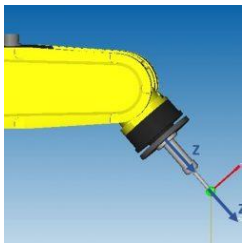


Figura 79. *Ejemplo de herramienta simple.*

- **Herramienta compleja.** En este caso el eje de ataque de la herramienta no es paralelo al eje Z de la herramienta por defecto, si no que se encuentra desplazado y su orientación redefinida.

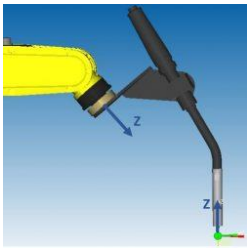


Figura 80. *Ejemplo de herramienta compleja.*

Todas las herramientas utilizadas para este proyecto son herramientas simples, cuyo método de definición usado es la entrada directa, puesto que como se ha indicado anteriormente, todos sus parámetros y geometría son conocidos, lo que facilita esta labor.

Para definir una nueva herramienta en un robot se accede al menú de configuración de herramientas EOAT ('*End Of Arm Tool*') a través del árbol del proyecto en la opción ***Tooling*** de cada uno de los robots utilizados.

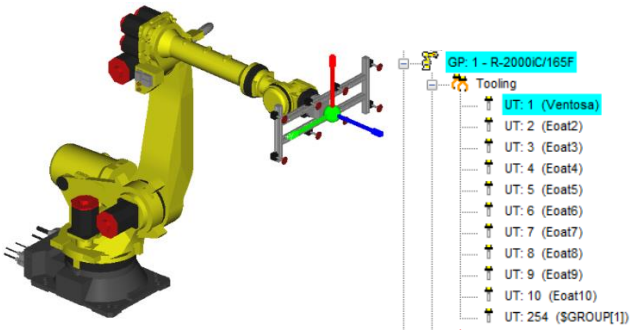


Figura 81. *Menú Tooling del software de simulación.*

En la *Figura 82* se muestra la configuración de la garra con ventosas de vacío para el robot R-2000iC/165F.

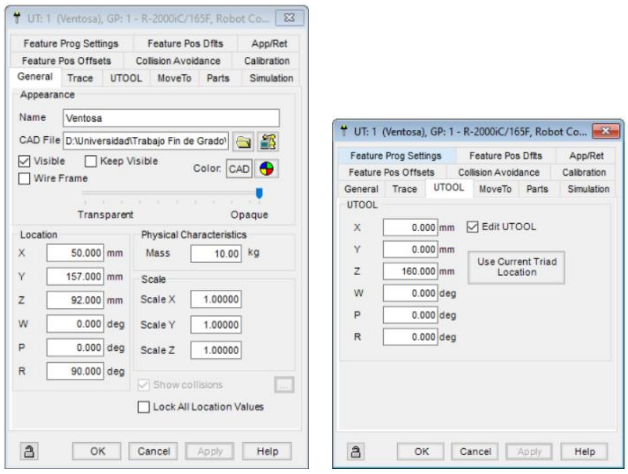


Figura 82. *Configuración de la garra ventosa de vacío en el robot R-2000iC/165F.*

En el menú ***General*** se establece un nombre con el que se identificará la herramienta, así mismo se carga el archivo CAD y se establece su localización respecto al UTOOL_0 del último eslabón del robot. Este menú permite también la configuración de la masa de la herramienta y la escala de su representación en el entorno gráfico del software.

Una vez definidos los aspectos generales, en la pestaña ***UTOOL*** se establece el punto de actuación de la herramienta. En este caso como ya se ha indicado previamente, se van a introducir estos parámetros de forma directa. Si las

dimensiones de la herramienta no fueran conocidas, la forma de definir el punto de actuación de la herramienta debiera ser el método de los tres puntos, disponible a través del iPendant.

Cada una de las partes que esta herramienta vaya a utilizar debe haber sido añadida en el menú ***Parts*** del organigrama del proyecto. Para configurar su posición respecto a la herramienta se debe definir un '*offset*', para ello se accede a la pestaña ***Parts,*** desde donde se pueden escoger las piezas que se van a manipular, así como la orientación y posición relativas de este objeto respecto a la herramienta cuando se actuando sobre este.

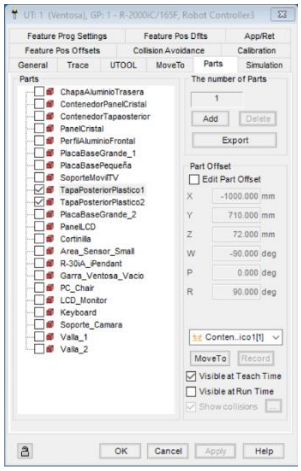


Figura 83. *Definición de las piezas que se van a manipular con el robot R-2000iC/165F.*

El ejemplo de configuración de herramientas seguido para el robot R-2000iC/165F se reproduce de forma análoga en el resto de los robots, definiendo para cada uno de ellos el elemento CAD que corresponda, así como las piezas que va a manipular cada herramienta de forma independiente.

En el caso del robot R-2000iC/125L, la configuración de la herramienta es principalmente la misma, con la salvedad de los objetos que se manipulan.

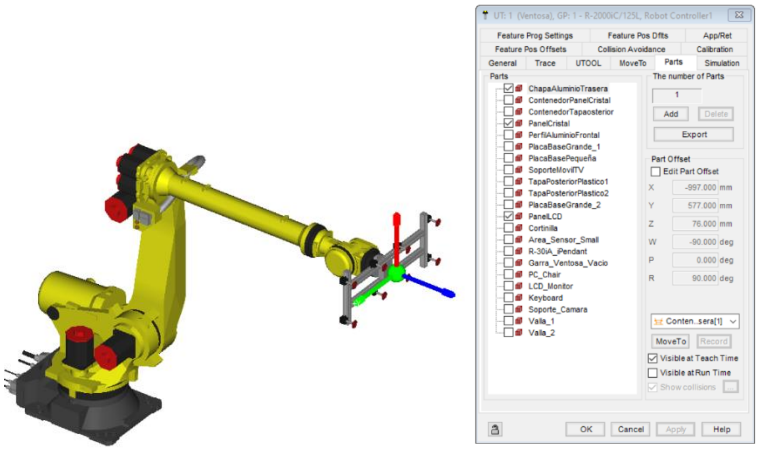


Figura 84. *Configuración herramienta de ventosa de vacío en el robot R-2000iC/125L.*

Para el caso del robot M-710iC/45M se deben configurar dos herramientas, una de ellas para la manipulación de placas de circuito que se compone de dos ventosas de vacío, cuya configuración se muestra en la siguiente figura.

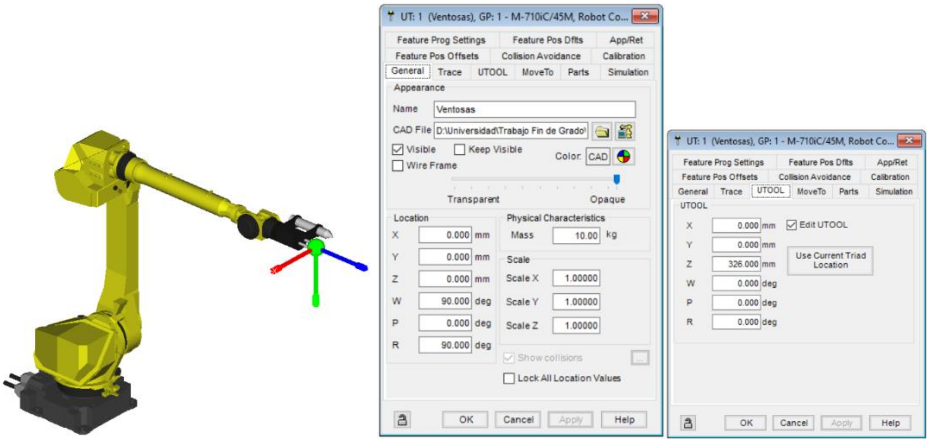


Figura 85. *Configuración de la garra ventosa de vacío en el robot M-710iC/45M*

En el caso de la herramienta de atornillado que porta este mismo robot para realizar el atornillado de tapas posteriores de plástico, el eje ataque se encuentra desplazado respecto al eje Z de la herramienta por defecto tal y como se muestra en la *Figura 86.* Este factor será importante a la hora de planificar las trayectorias con esta herramienta activa, considerando posibles colisiones y singularidades que se puedan dar.

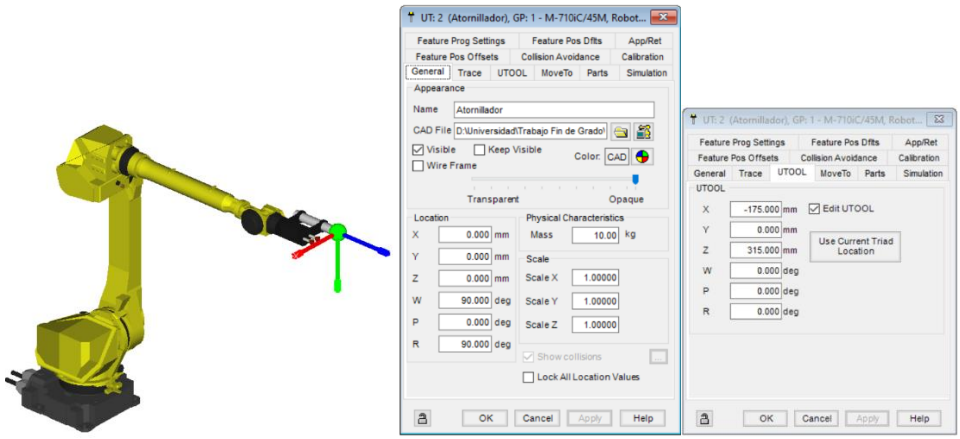


Figura 86. *Configuración de herramienta de atornillado en el robot M-710iC/45M*

Capítulo 6.1.2. Definición de los sistemas cartesianos de referencia

Un 'User Frame' (UFRAME) es un sistema de referencia tridimensional sobre el cual se posicionan los puntos de paso de una trayectoria definida en un programa de un robot. En caso de no encontrarse definido por el usuario ningún

sistema de referencia se establece por defecto el sistema de coordenadas WORLD, cuyo origen se encuentra en el centro de la base del robot.

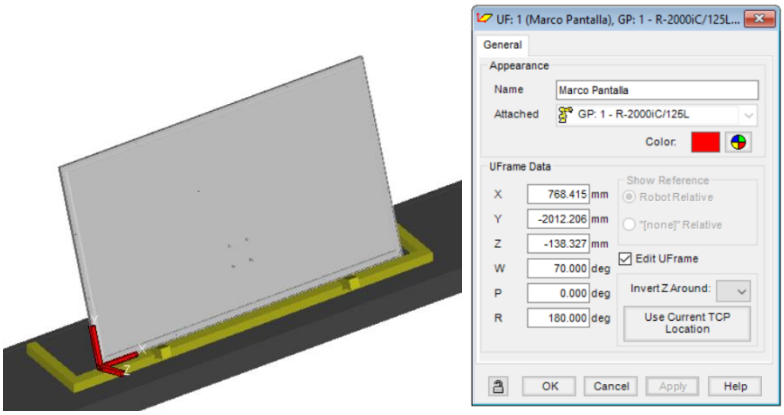


Figura 87. *Ejemplo de sistema de referencia de usuario en el robot R-2000iC/125L.*

Cabe destacar que los sistemas de referencia se asignan a cada robot, puesto que se definen en función del sistema de coordenadas WORLD de cada uno de ellos. Por lo tanto, las coordenadas del sistema de referencia *Marco Pantalla,* que se muestra en el ejemplo de la *Figura 87,* serán diferentes para cada robot.

Para establecer la localización del sistema de referencia en este caso, al no ser conocida la posición exacta se debe usar el método de los tres puntos a través del iPendant. Para ello se debe seguir el siguiente atajo de teclas y opciones desde el '*Teach Pendant*' para acceder a este menú de configuración:
MENU → 6 SETUP → 4 Frames → OTHER → 3 User Frame.

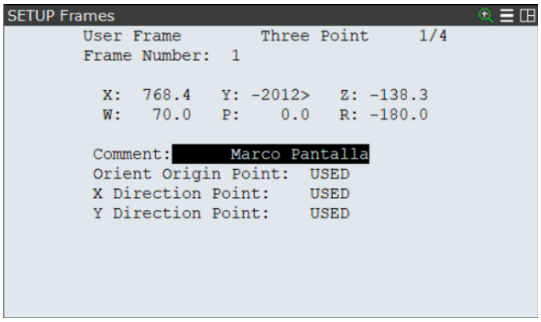


Figura 88. *Menú de configuración de sistemas de referencia de usuario.*

Con este método definimos dos rectas que se cruzan, correspondientes a los ejes X e Y, el punto de cruce será el origen del plano y perpendicular al plano formado, a través del punto origen se sitúa el eje Z. Este método se sigue para cada sistema de referencia que se desea asignar a cada uno de los robots.

Para el caso del sistema de referencia para la calibración de la cámara de visión artificial, se especificará en detalle en el *Capítulo 6.1.5. Configuración iRVision*.

Capítulo 6.1.3. Configuración de los componentes de la estación

Para los componentes diseñados, o añadidos de las librerías proporcionadas en el software, que forman la estación, es necesario definir su comportamiento dinámico con el fin de lograr una recreación de la actuación de sus mecanismos de forma realista durante la simulación.

Para definir el comportamiento, en primer lugar, se debe catalogar cada componente dentro de cada una de las categorías marcadas en la siguiente figura dependiendo de la función que va a desempeñar.

Figura 89. *Campos de estudio señalados en el organigrama de Roboguide.*

Machines

La categoría machines agrupa dispositivos de movimiento generalmente servo motorizados con diferentes funcionalidades. Este tipo de dispositivo puede referirse a un eje externo de movimiento de uno de los robots o a sistemas externos, no relacionados con el robot, dedicados al movimiento de otros componentes que forman parte de la instalación.

Figura 90. *Submenú Machines del organigrama del proyecto.*

- **Cinta de entrada de cristales.** El principio de funcionamiento de este componente se basa en un conveyor lineal capaz de desplazar objetos en la dirección definida por el eje Z del motor que proporciona el movimiento.

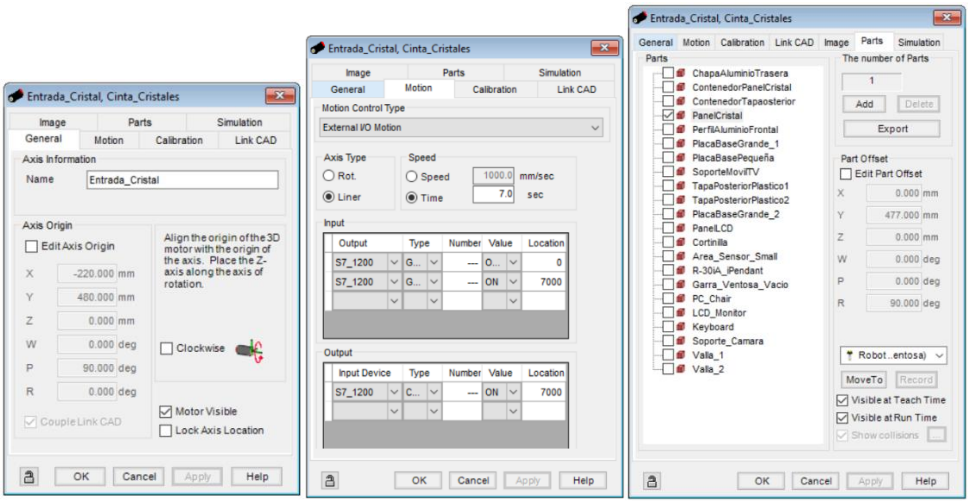


Figura 91. *Configuración conveyor de entrada de cristales.*

Entre las configuraciones principales de este tipo de componente se encuentra la pestaña ***Motion*** desde donde se definen los parámetros de movimiento, así como el dispositivo encargado de gestionar las señales que indican el inicio de un nuevo desplazamiento, hasta el tipo de movimiento (lineal o de rotación) y la velocidad de ejecución del mismo.

La pestaña ***Parts*** permite seleccionar los objetos manipulables por los robots que se van a mover (paneles de cristal para esta máquina), así como el ajuste del 'Offset' de estas piezas respecto al punto origen del movimiento.

- **Cinta de entrada de paneles LCD.** Esencialmente esta máquina consiste en un duplicado del conveyor anterior con la salvedad del posicionamiento del objeto estático (estructura del conveyor) y el objeto que desplaza que se trata de un panel LCD en esta ocasión.

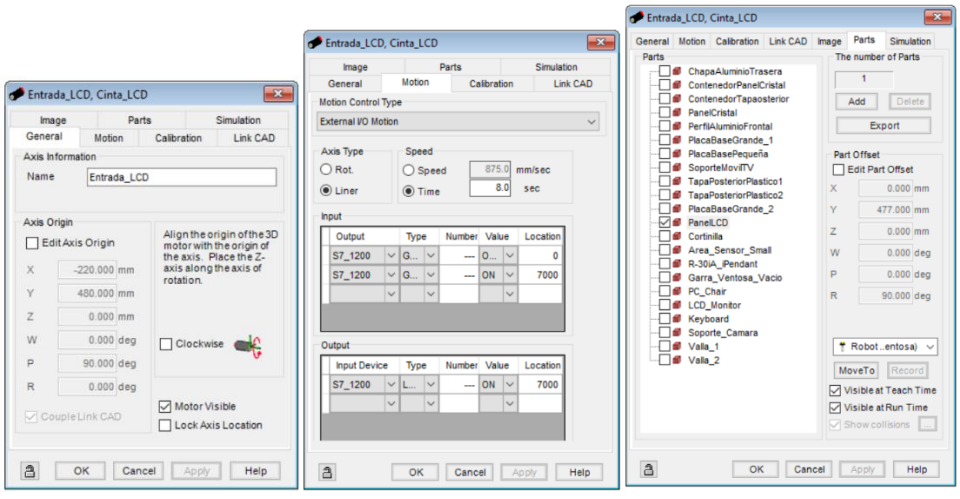


Figura 92. *Configuración conveyor de entrada de paneles LCD.*

- **Cinta de entrada de placas de circuito.** Se concibe este elemento nuevamente como una cinta de transporte lineal, principalmente de la misma naturaleza que las anteriores.
  En este caso se cuenta con dos elementos '*Link*' encargados de la transferencia de movimiento que tendrán asignados diferentes objetos. El primero de ellos se encarga del transporte de las placas de circuito de mayores dimensiones, mientras que el segundo transporta únicamente la placa de menor tamaño.

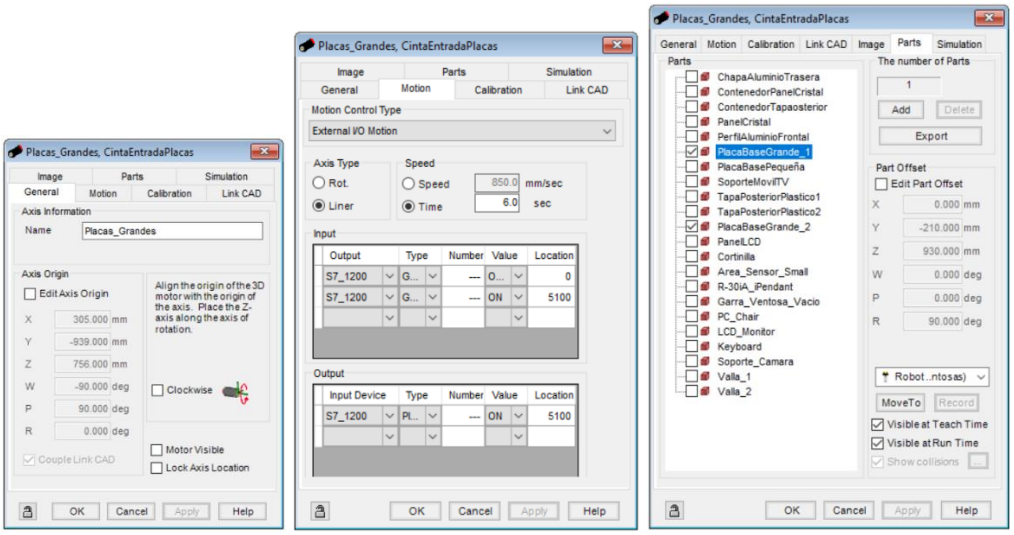


Figura 93. *Configuración de cinta de entrada para placas de circuito ID 0001 e ID 0002.*

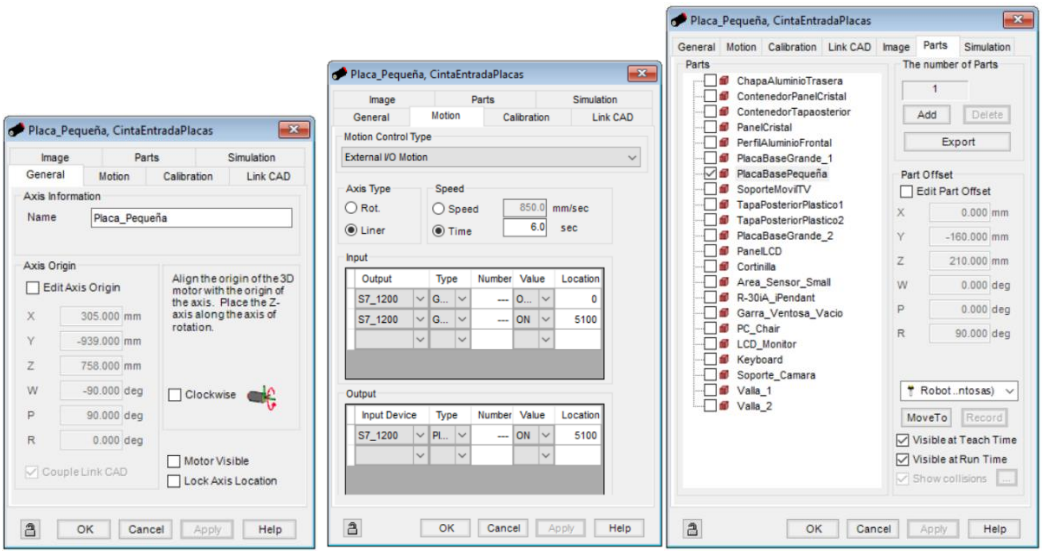


Figura 94. *Configuración cinta de entrada de placas de circuito ID 0003.*

Este cambio respecto a otras máquinas tiene su origen y justificación en la índole del proyecto, puesto que para realizar una simulación dinámica donde se pueda ensamblar en una misma estación televisores que contengan diferente número de placas de circuito se requiere recurrir a esta técnica, que permite simular la llegada hasta el puesto de visión 2 o 3 placas de circuitos

según la situación indicada por el autómata y no siempre 3 placas teniendo que ser desechada una de ellas.

- **Cinta de montaje.** Nuevamente se cuenta con una cinta transportadora cuyo movimiento es lineal. En este caso, se cuenta con 3 objetos '*Link*' que permiten simular las diferentes paradas del soporte móvil, en los puntos de entrada y salida de la estación, así como los puntos de trabajo de los robots en modo automático, y un cuarto elemento '*Link*' para la simulación en modo manual de la instalación.

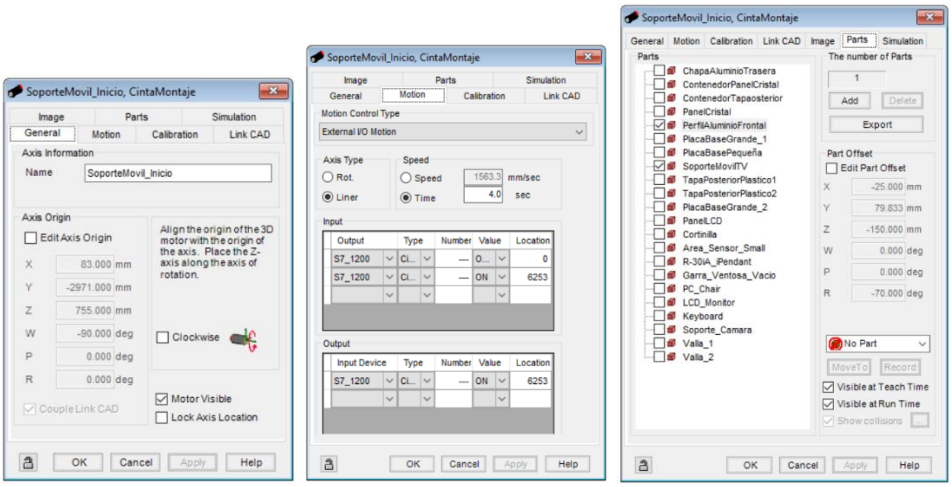


Figura 95. Configuración tramo inicial del recorrido en modo automático.

Al simular cada uno de los elementos '*Link*' la parada del soporte en las posiciones de trabajo de los diferentes robots, cada elemento contara con sus propias señales de activación e indicativo de posición alcanzada. Así mismo será necesario programar la transferencia de los objetos que se transportan desde un elemento '*Link*' al posterior para dotar de continuidad y fluidez a la simulación.

En el ejemplo de la *Figura 95* se muestra la configuración para uno de los elementos móviles de los tres que comprenden el modo automático de funcionamiento de la instalación. La configuración de los otros dos elementos restantes es íntegramente análoga, cambiando los puntos de partida y parada. Si el lector desea profundizar en su análisis puede recurrir a los Anejos de la memoria, donde en la carpeta "Roboguide (FANUC)" encontrará la configuración completa de estos elementos.

Respecto al elemento que permite la simulación del movimiento del soporte en modo manual, su principal característica es que este está dotado de todas las partes que forman el televisor completamente ensamblado, así como cada una de las paradas en los diferentes puntos de trabajo de los robots, todo integrado en un mismo elemento 'Link'.

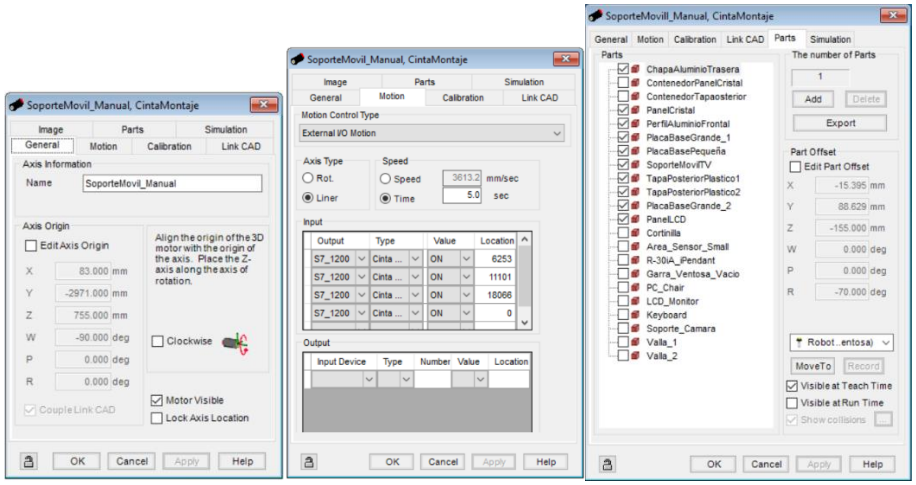
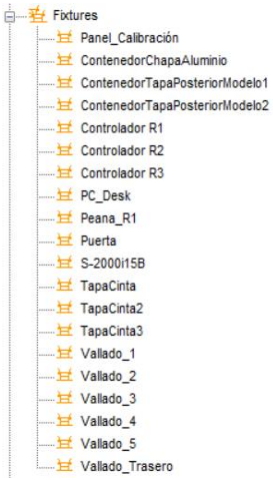


Figura 96. *Configuración cinta de transporte del soporte móvil para el montaje en modo manual.*

Fixtures



Dentro de este menú se encuentran elementos que forman parte de la estación usados para soportar piezas del apartado de ***Parts***, como contenedores de objetos.

Así mismo se sitúan aquí elementos que sirven para guardar posiciones o definir sistemas de referencia de usuario.

Se pueden encontrar en esta categoría componentes dedicados a la funcionalidad 'Line tracking' si algún robot cuenta con este complemento.

Figura 97. *Submenú Fixtures del organigrama del proyecto*

Para poder realizar una correcta calibración de la cámara de visión artificial que utilizará el robot M-710iC/45M para el reconocimiento de las placas de circuito, es necesario incluir un panel de calibrado. Este panel se incluye en el apartado ***Fixtures*** por la facilidad para crear un sistema de referencia ajustado en el punto central del panel tal y como se muestra en la siguiente figura.

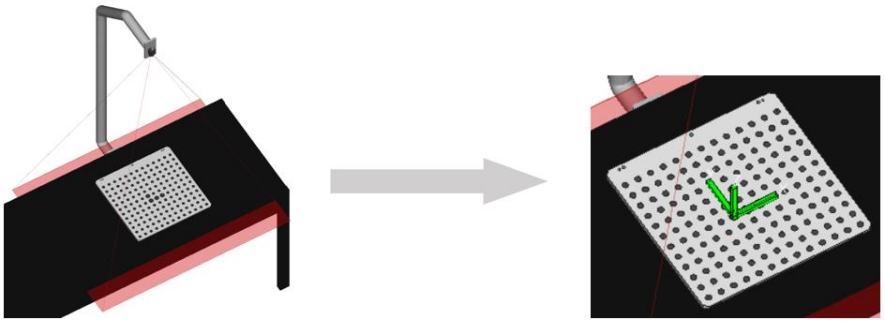


Figura 98. *Panel de calibración para cámaras de visión artificial con iRVision.*

Respecto a los contendores de objetos (véase en *Figura 65*), cada uno de ellos está diseñado para incluir una gran cantidad de piezas en su interior, pero añadir cada una de ellas de forma individual a la instalación supondría una gran sobrecarga en la simulación. Para solventar esta situación se cuenta con opción de poder definir una matriz de objetos en las opciones del contenedor tal y como se ilustra en la siguiente figura.

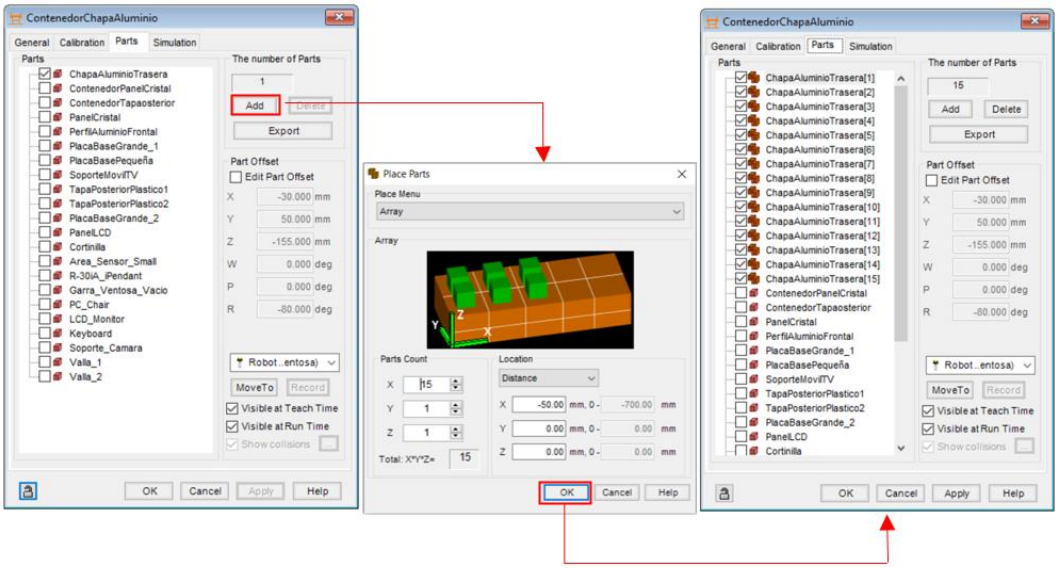


Figura 99. *Matriz de componentes en contenedor de chapas de aluminio.*

El ejemplo mostrado en la figura es el procedimiento realizado para el contendor de chapas posteriores de aluminio. Este mismo procedimiento se replica para ambos contenedores de tapas de plástico, quedando de esta forma completos con todas las partes que pueden ser manipuladas durante la simulación.

Respecto al resto de elementos que se observan en la *Figura 97*, todas ellas tienen una finalidad estética con el fin de dotar a la estación de una apariencia realista que mejora su apreciación durante la simulación.

Parts

Se disponen en este apartado las piezas que forman parte de la simulación del proceso como objetos manipulables o que se desplazan a través del sistema, así como piezas fijas que pertenecen a un conjunto de accesorios para formar un elemento mayor, habitualmente con un propósito estético.

De forma independientes estos objetos no se pueden utilizar dentro de la estación, deben ser añadidos a como accesorios dentro de los apartados Fixtures o Machines.

Figura 100. *Submenú Parts del organigrama del proyecto.*

External Devices

El software de simulación HandlingPRO proporciona la posibilidad de conectar dispositivos externos a la simulación que se va a realizar en la instancia del proyecto abierto. Para ello desde el menú ***External Devices → Add Device*** se debe seleccionar la opción ***OPC Server*** que despliega una ventana de configuración desde el que se establecen los parámetros para la conexión al servidor OPC que se va a utilizar para que el autómata pueda controlar la estación.

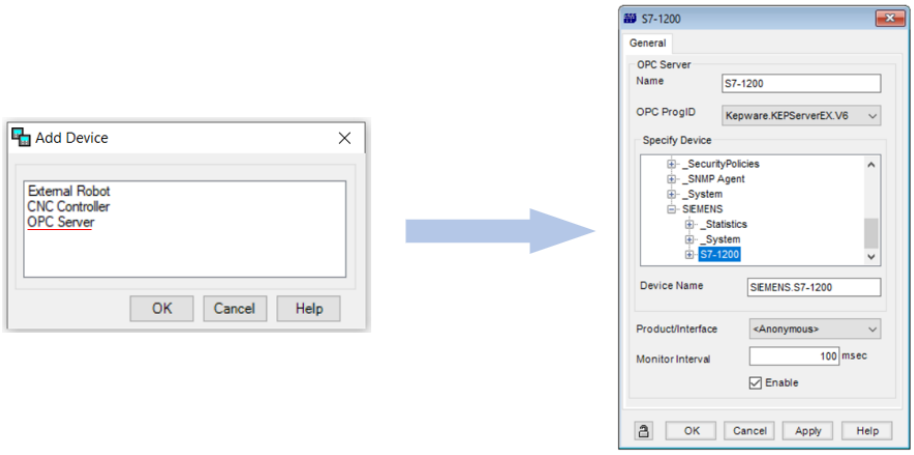


Figura 101. *Configuración del servidor OPC.*

En esta ventana de configuración se tiene que indicar el ID del servidor activo en la maquina en la que se va a realizar la simulación, en este caso ***Kepware.KEPServerEX.V6*** es el identificador del software que alberga el servidor OPC que se va a utilizar. Dentro de este software se debe seleccionar el canal que se crea para los dispositivos SIEMENS y en concreto el dispositivo S7-1200 que identifica al autómata S7-1214 ac/dc/rly que se va a utilizar.

Una vez establecido el dispositivo, si este ofrece una interfaz de las reconocidas por Fanuc se selecciona en el desplegable de la opción ***Product/Interface*** al no estar incluida ninguna opción especifica que se adapte a las necesidades del proyecto se establece la opción ***<Anonymous>.*** Por último, se selecciona el intervalo de muestro y se habilita el dispositivo.

Sensor Units

Dentro de este apartado podemos añadir a la simulación una serie de sensores en función de los complementos software que se hayan incluido a los robots de la estación del proyecto actual. En este caso al contar con los paquetes de la funcionalidad ***iRVision*** se cuenta con una serie de dispositivos apropiados para el reconocimiento mediante cámaras de visón y sensores de área.

Concretamente para este proyecto se va a utilizar una cámara de visión de las proporcionadas en las librerías del software, en concreto el modelo ***KOWA***

SC130EF2 B/W, que se trata de un modelo para labores de visión en dos dimensiones con imagen en blanco y negro. Este dispositivo se ha escogido en concreto por la resolución que ofrece, mucho mayor a las cámaras de captación bidimensional con espectro del espacio de colores RGB, pues la tarea que se va a desarrollar consiste en el reconocimiento de geometrías y posiciones, por lo tanto, la funcionalidad de reconocimiento de color es prescindible en favor de la resolución.

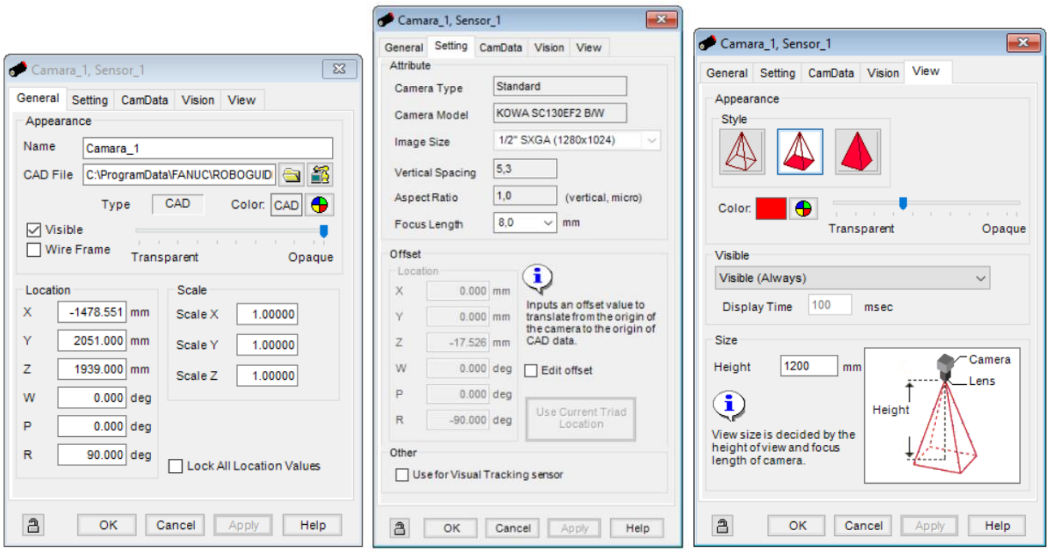


Figura 102. *Ventanas de configuración cámara de visión artificial.*

La anterior figura recoge las principales pestañas de la ventana de configuración de la cámara escogida. Entre los parámetros de la pestaña ***General*** destaca la localización y escala del dispositivo dentro del entrono gráfico del software, respecto al Menú ***Settings*** el principal atributo que se puede modificar es la distancia focal del sensor (***Focus Length***) que afecta directamente al tamaño del campo de visión (mayor cuanto menor es la distancia focal). Por último, en el apartado ***View*** se modifican aspectos referentes a las propiedades del cono de visón, desde su aspecto y estilo, nivel de opacidad y distancia de dibujado.

Capítulo 6.1.4 Configuración de los robots

Antes de comenzar con la programación de los robots utilizados en la estación es necesario realizar una serie de configuraciones previas, tanto en el software como en los propios robots para que la simulación se puede llevar a cabo.

Muchas de estas configuraciones son idénticas en los tres robots utilizados, en los casos en que esto ocurra tan solo se definirá la configuración en uno de los robots, quedando indicado que, para el resto de los robots, la configuración es una réplica exacta.

Con la definición de herramientas, sistemas de referencia y el resto de los elementos que componen la estación completamente configurados, se debe establecer el método de arranque de los robots, que en este caso se realizará de forma remota, a través del menú ***Test-Run → Run Configuration.***

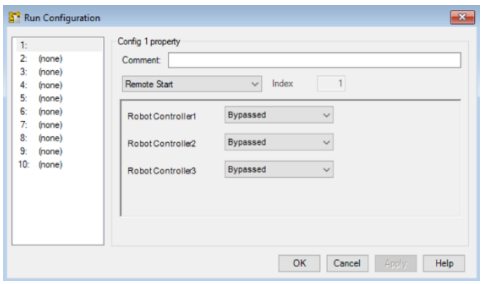


Figura 103. *Menú de configuración de arranque.*

Una vez definido el modo de arranque, a través del iPendant y el menú de señales externas (***Tools → External I/O Connection***) se establecerá la relación de las señales que provienen del autómata para la gestión tanto del propio arranque, como la pausa de programas, parada completa de la ejecución y selección del programa que se desea ejecutar en el arranque remoto.

Desde la siguiente selección de botones ***MENU → I/O → UOP → IN***, del iPendant, se accede al menú de señales que permiten comandar el robot a distancia.

Figura 104. *Configuración de señales de entrada UOP.*

Dentro de este menú se localizan algunas de las principales funciones que se abordan en el proyecto como la pausa controlada (*UI[2]: HOLD*) o el reinicio de la ejecución del programa (*UI[6]: START*) entre otras. Estas señales deben ser asignadas en el menú me conexiones externas a las variables del servidor OPC para que puedan ser comandades estas acciones por el autómata.

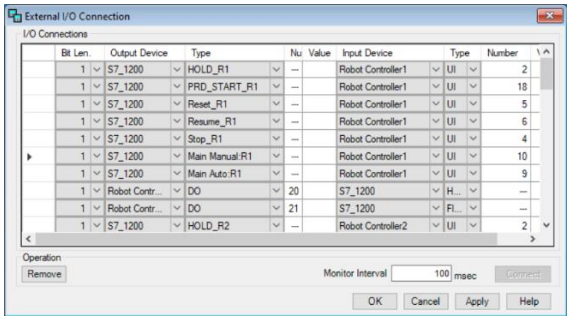


Figura 105. *Menú de conexión de señales externas.*

Las señales *UI[9-16]* que se pueden observar en la *Figura 104* son las que permiten el poder controlar el robot con un PLC externo que comande la ejecución de los programas a través de estas señales.

En este proyecto se van a utilizar dos programas principales, para el control manual y el funcionamiento automático, ambos comandados y arrancados externamente mediante señales digitales. No se desea la interrupción de un programa en ejecución cuando llega la señal de activación de otro programa, si no que se espera a la ejecución del programa actual para comenzar la ejecución del siguiente, por lo tanto, de los métodos disponibles para realizar el arranque externo, el más adecuado es el conocido como '*Robot Service Request*' (RSR).

Para realizar la configuración de arranque de los programas se accede desde el iPendant a través de la siguiente secuencia: ***MENU → SETUP → Prog. Select.*** Dentro de este menú se debe escoger RSR como modo de selección *[1]* y UOP como método de arranque *[2]*.

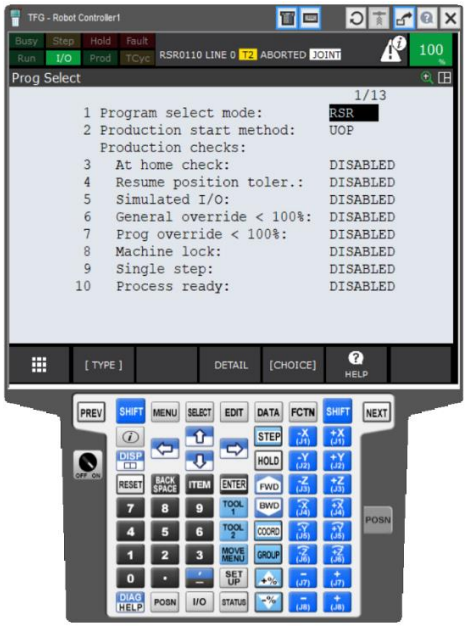


Figura 106. *Menú de configuración de selección de programas.*

Desde el apartado ***Detail*** de este mismo menú se accede a una serie de configuraciones adicionales. El método de selección RSR permite establecer 8 identificadores a los programas, que se corresponden con 8 bits de entrada, por lo tanto, desde esta pantalla de configuración se debe establecer el bit correspondiente a cada número de señal de entrada. Adicionalmente se puede agregar una base sobre la que se agrega el número de bit establecido a cada programa, así como el prefijo de las tareas.

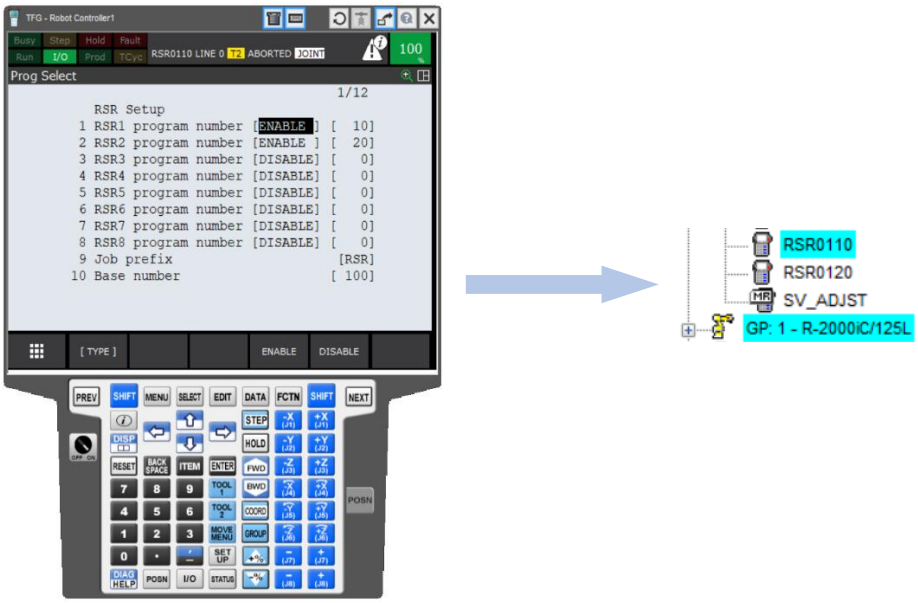


Figura 107. *Configuración detallada de selección de programas.*

De esta forma queda establecido que la entrada RSR1 (asignada a la señal externa *Main Auto:R1* como se puede ver en la *Figura 105*) ejecutará el programa RSR0110 (base 100 + valor de bit RSR1 → 10) y la entrada RSR2 ejecutará la tarea RSR0120.

Este proceso se debe replicar en cada robot. Al contar cada robot con su controlador propio, podría asignarse la señal de entrada RSR1 de cada uno de los robots al bit 10, pero se ha decido establecer en cada robot unos bits diferentes para facilitar la comprensión del conjunto.

Tras definir el método de arranque, selección de programas y asignadas las variables de control del robot, es necesario configurar el resto de las señales digitales para la gestión de las tareas a realizar por cada robot. Para ello se accede desde el iPendant al siguiente menú: ***MENU → I/O → Digital → Config.***

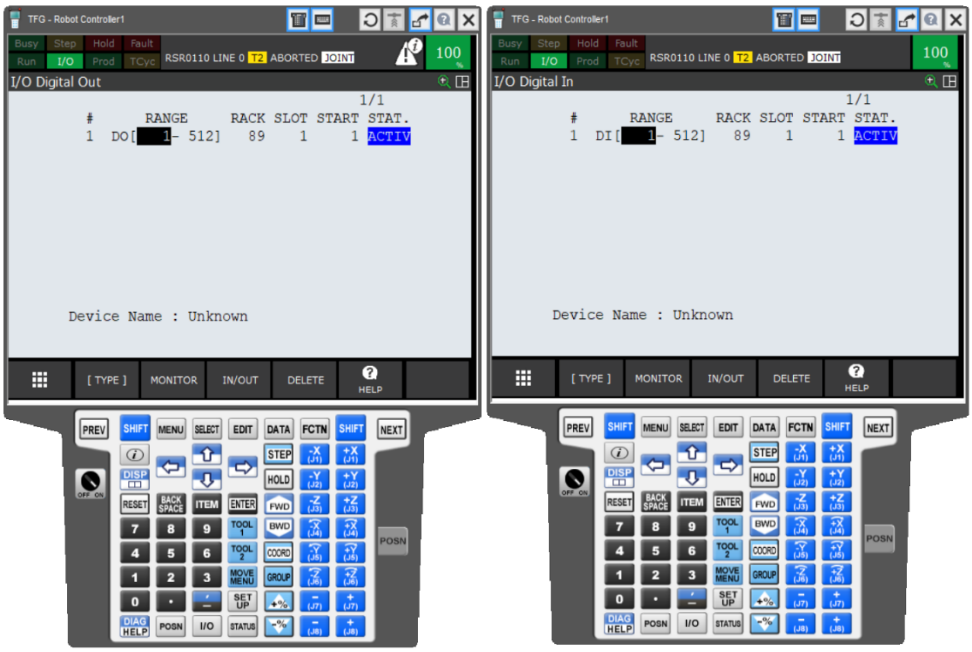


Figura 108. *Menú de configuración de entradas y salidas digitales del robot.*

Estas pantallas permiten establecer el mapeado de un rango de señales que se van a asignar a un '*Rack*' y '*Slot*' determinados. Es importante que esta configuración no se replique para otro tipo de señales digitales (como UOP anteriormente explicadas) pues entonces el funcionamiento será errático ya que se estarán superponiendo asignaciones diferentes en las mismas señales.

En este caso, para todo el rango disponible de señales digitales de entrada y salida, se ha establecido la configuración '*Rack*' 89 y '*Slot*' 1. Esta configuración es la indicada en los manuales del fabricante para la conexión de señales mediante tecnología Ethernet. La configuración normalmente para un proyecto de simulación se realiza con '*Rack*' 0 y '*Slot*' 1, pero esta es la configuración utilizada para las señales de entrada *UOP,* por lo tanto, se ha decidido cambiar la configuración para evitar superposición de señales.

Capítulo 6.1.5. Configuración iRVision

Uno de los robots utilizados en el proyecto, concretamente el robot ***M-710iC/45M*** *(R2),* encargado de la manipulación de las placas de circuitos, va a estar apoyado por la funcionalidad de visión artificial proporcionada por el propio software de simulación iRVision.

El propósito de uso de esta herramienta reside en la necesidad de reconocimiento de tres tipos diferentes de placas de circuitos que en función de su ID deben ir posicionadas en el lugar diseñado para cada una de ellas. Adicionalmente, se puede dar el caso de que estas placas no siempre lleguen

perfectamente alineadas respecto a la posición de llegada guardada, por lo tanto, se debe reconocer el desplazamiento respecto a esta posición previamente definida.

El primer paso para la configuración es definir un sistema de referencia cartesiano que se ajuste a la referencia central del panel de calibración (véase *Figura 98*). Es importante tener en cuenta que este sistema de referencia, además de para la calibración de la cámara, será el utilizado para dar la referencia del desplazamiento y rotación de las placas respecto a la posición predefinida.

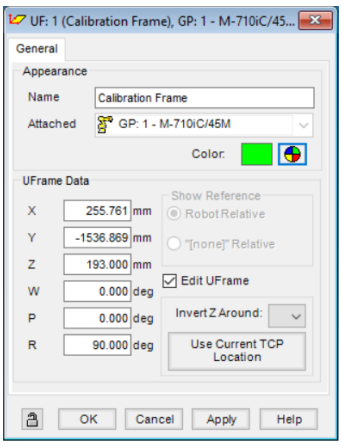


Figura 109. *Definición del sistema de referencia para iRVision.*

Definidos la posición del panel de calibración y el sistema de referencia, se debe asignar el sensor añadido (véase en el apartado '*Sensor Units*' del *Capítulo 6.1.3. Configuración de los componentes de la estación*) a la estación al robot que va a utilizar esta herramienta de reconocimiento.

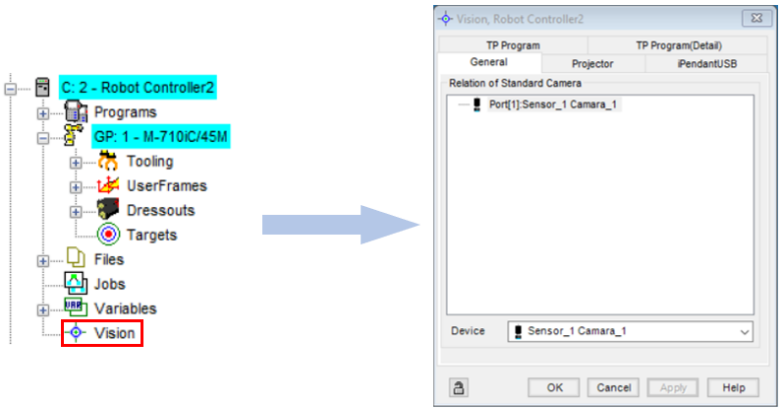


Figura 110. *Asignación del sensor de visión al robot.*

Una vez asignado el dispositivo de visión al robot, se procede a su calibración y configuración. Desde el menú ***Robot → Internet Explorer (o Web Browser)*** del software, se accede a una ventana de configuración de todas las funcionalidades adicionales del robot seleccionado.

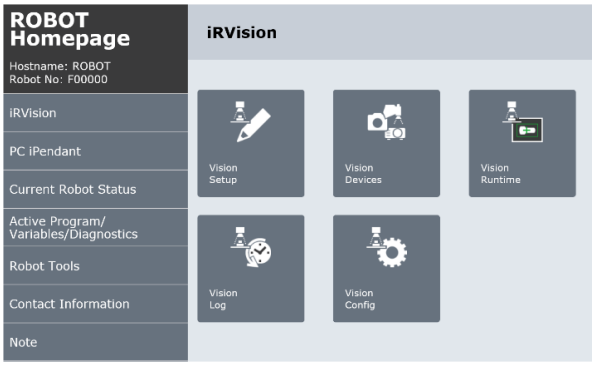


Figura 111. *Configuración de funcionalidades del robot.*

Desde esta ventana podemos acceder a diferentes aspectos de configuración, reporte de errores y visión en tiempo real de ejecución de los diferentes dispositivos asignados al robot. En este caso se centrará en estudio únicamente en la función ***Vision Setup.***

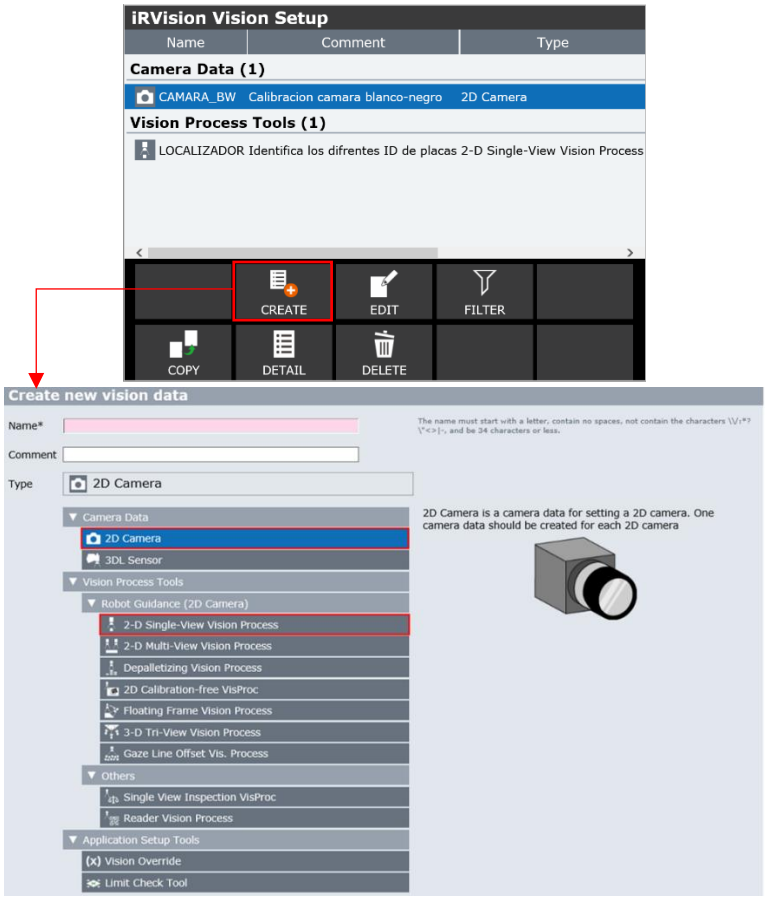


Figura 112. *Configuración cámara iRVision.*

Sobre esta figura se pueden observar el amplio abanico de posibilidades que da esta herramienta. Para la realización de este proyecto únicamente se requieren las opciones que se ven remarcadas en la ilustración: *2D Camera* y *2-D Single-View Vision Process*.

2D Camera

Dentro de esta función se encuentran las herramientas necesarias para realizar el calibrado de la cámara de visión artificial. En primer lugar, se debe establecer el dispositivo de visión sobre el que se debe realizar la calibración, así como el método que se va a emplear.

Como ya se ha indicado anteriormente el método de calibración se basa en un panel con una matriz de agujeros, sobre el que se ha definido un sistema de referencia que servirá de guía para el propio calibrado. Definir el espaciado entre el centro de los agujeros de la matriz (30mm para el panel elegido) facilita el proceso, así como el tipo de vista (ortogonal en este caso). Una vez definidos estos parámetros, el programa pedirá situar el robot en el sentido que se desea definir como eje X para el calibrado y posteriormente el mismo proceso para el eje Y, con estos valores establecidos la calibración se realiza automáticamente.

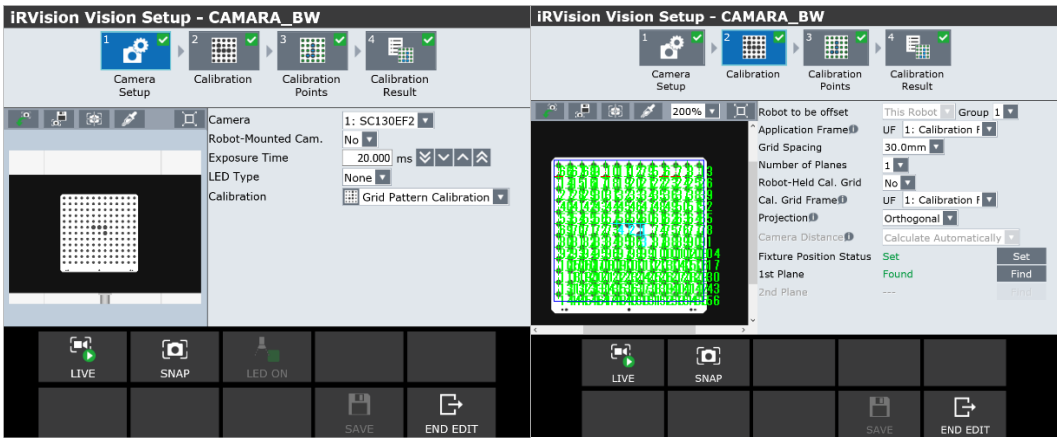


Figura 113. *Calibración del dispositivo de visión.*

2-D Single-View Vision Process

Tras realizar la calibración de la cámara de visión se debe configurar el proceso de visión. En este caso se trata de un reconocimiento en dos dimensiones mediante una sola cámara, para establecer un '*offset*' a los movimientos que va a realizar el robot.

El primer paso dentro de la configuración de esta funcionalidad es asignar la calibración de la cámara realizada anteriormente, así como ajustar el espacio sobre el que se va a realizar la tarea (no necesariamente todo el espacio visible por la cámara). Se debe indicar también si la cámara se encuentra fija en una posición o montada en la herramienta del robot (para este proyecto la posición es fija) así como el sistema de referencia que se va a utilizar y el número de piezas a localizar.

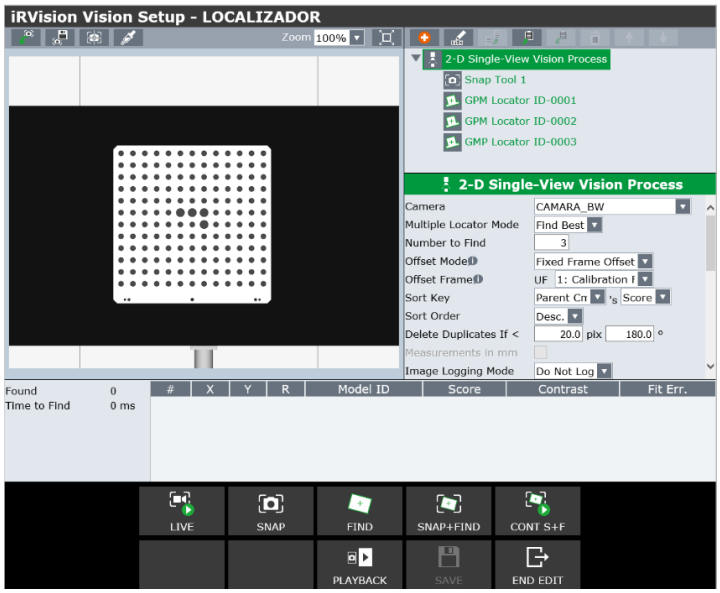


Figura 114. *Configuración del proceso de visión.*

Existen también otros parámetros de configuración menos relevantes para la realización de este proyecto como el registro de fallos del proceso, eliminar detecciones duplicadas, etc.

Una vez establecidos los parámetros de configuración, se escoge el método de reconocimiento que se desea utilizar, en este caso se identificaran las piezas en base a un modelo base que se guarda como referencia. Para cada una de las piezas diferentes que se desean detectar se debele incluir un proceso de este tipo.

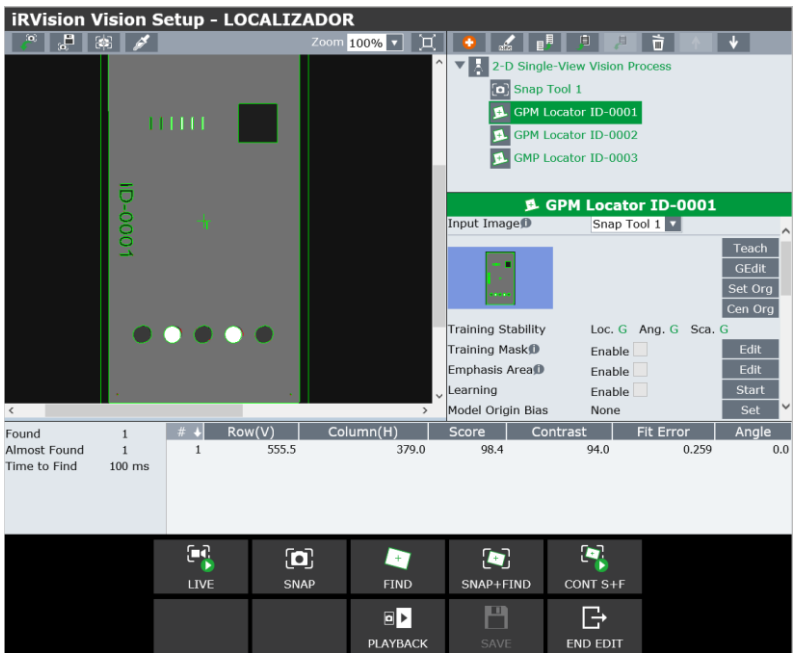


Figura 115. *Definición de elemento que se desea reconocer.*

Una vez definidos todos los modelos contemplados, se deben situar todos en el campo de visión de la cámara para establecer a cada uno de ellos un identificador que se utilizará desde el programa de visión para asignar el '*offset*' que corresponda en cada caso.

Debido al gran parecido entre las tres placas se ha establecido que el porcentaje de coincidencia de las piezas reconocidas respecto al modelo de referencia sea superior al 98%. De esta forma se asegura que no vaya a existir ningún error en el reconocimiento y se evita que una pieza sea asignada a un ID que no le corresponda.

Capítulo 6.1.6. Programación robot manipulador R-2000iC/125L

La estación diseñada cuenta con varios robots, cada uno de ellos con una serie de programas que componen su funcionamiento y tareas en el ensamblado. En el presente capítulo, así como en los específicos para cada robot que forma parte de la instalación, se explicarán las partes fundamentales del código desarrollado que permitan la comprensión del conjunto.

Antes de comenzar con la explicación detallada del código, se define un diagrama de flujo con las posibilidades contempladas en cada modo de actuación (manual o automático). A continuación, se muestran las representaciones gráficas más comunes, cuyo uso identifica las acciones llevadas a cabo durante el proceso.

Figura 116. *Simbología normalizada para la realización de diagramas de flujo.*

Diagrama de flujo R1 → R-2000iC/125L

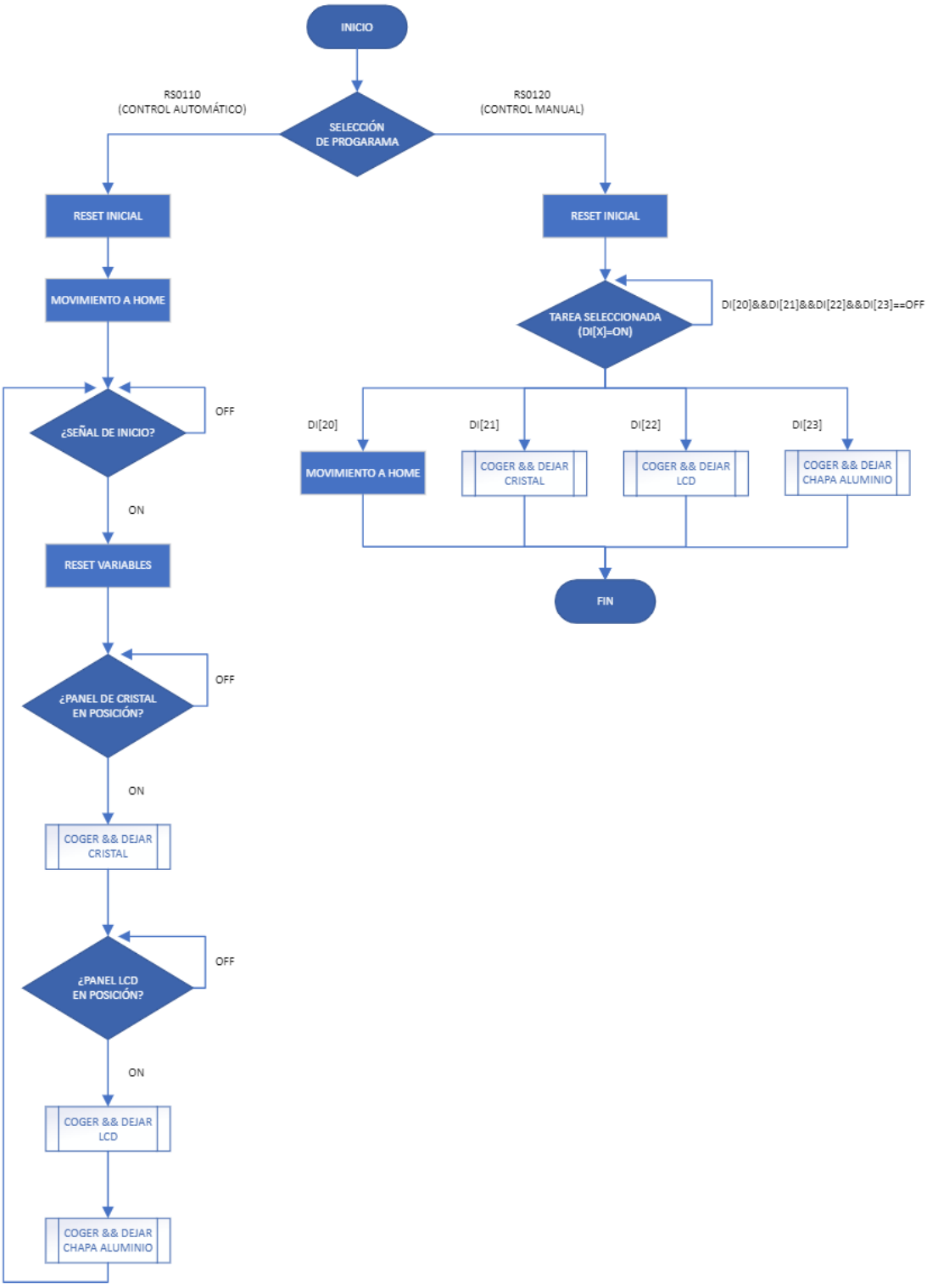


Figura 117. *Diagrama de flujo (código) robot R-2000iC/125L.*

Tal y como se había comentado en capítulos anteriores, el funcionamiento de los robots se basa en dos programas principales (RSR0110 y RSR0120 para

R1), asignados a los modos de funcionamiento automático y manual respectivamente, cuyo arranque se realiza de forma remota.

Como se puede observar en la anterior ilustración, la principal diferencia radica en la ejecución cíclica continua del programa automático, de forma que la única forma de finalizar la ejecución del programa es abortar o detener el funcionamiento global a de la estación a través del autómata. Por su parte, el programa de funcionamiento manual finaliza al terminar una vez que la tarea asignada haya sido realizada.

El estudio del código desarrollado se va a basar en el funcionamiento en modo automático (RSR0110), teniendo en cuenta que las subrutinas o programas independientes a los que se llama la ejecución de las tareas de *'Pick&Place'* son las mismas que para el modo manual. Por otro lado, si el lector desea profundizar en el funcionamiento en modo manual, puede acudir a los anejos a la memoria, donde encontrará todos los programas.

Cabe destacar que todas las tareas, a excepción del posicionamiento en *HOME*, son lanzadas desde uno de los programas principales, ya sea el destinado al funcionamiento automático o al manual.

Conviene antes de comenzar la lectura de los siguientes epígrafes, retomar la explicación relativa a la programación, principales instrucciones y formato de las instrucciones de definición de posiciones realizada en el *Capítulo 4.1.3. Programación del robot.*

RSR0110 – Funcionamiento en modo automático

Al inicio del procedimiento principal de funcionamiento automático se realiza una llamada a un programa se reseteo inicial de señales e inicialización de registros. Este procedimiento se realiza solamente una vez al seleccionar el modo automático de funcionamiento para establecer las condiciones iniciales de la simulación, con el fin de corregir posibles estados anteriores que se hayan quedado grabados de ejecuciones anteriores.

Con la primera etiqueta *LBL[10]* se establece la primera pieza para generar el buble infinito de ejecución que se completa al final del propio programa, una vez ejecutadas todas las instrucciones con un *JMP LBL[10].* Una vez iniciado el bucle se debe asegurar que la herramienta y sistema de referencia seleccionados son los correctos, con las sentencias UFRAME_NUM=0 (selección de sistema de referencia) y UTOOL_NUM=1 (selección de la herramienta), para seguidamente mover el robot a la posición *HOME* guardada e indicar al autómata que el robot se encuentra listo para iniciar sus tareas.

```
/PROG  RSR0110
/MN
  1:  !////////////////////////////////;
  2:  !CONTROL AUTOMATICO DE R1;
  3:  !////////////////////////////////;
  4:  ;
  5:  !RESET INCIAL;
  6:  CALL RESET_INICIAL    ;
  7:  ;
  8:  LBL[10] ;
  9:  !HERRAMIENTA Y BASE DE TRABAJO;
 10:  UFRAME_NUM=0 ;
 11:  UTOOL_NUM=1 ;
 12:  ;
 13:  !MOVIMIENTO A HOME;
 14:J PR[1:HOME] 100% FINE    ;
 15:  WAIT    .10(sec) ;
 16:  DO[20:R1 EN HOME]=ON ;
 17:  ;
```

Una vez recibida la señal de inicio de ciclo por parte del autómata, se ejecuta un reseteo de las señales que puedan haber quedado activas de un ciclo anterior.

Los siguientes pasos serán esperar la señal de confirmación de llegada de panales de cristal y LCD para proceder con las respectivas maniobras de manipulación. Una posicionados ambos paneles, se toma la chapa de aluminio que cubre dichos elementos y se posiciona sobre el marco del televisor. Estas maniobras de manipulación de objetos se detallarán más adelante.

Tras finalizar las tareas secundarias de toma y depósito de piezas en su posición final, se aumenta el contador de ciclo y se realiza una comprobación dual de fin de ciclo, basado en una señal de salida para indicar la finalización y otra de entrada para la confirmación por parte del autómata.

```
 18:  !ESPERA SENIAL DEL PLC;
 19:  WAIT DI[20:INICIO DE CICLO]=ON    ;
 20:  ;
 21:  !RESET VARIABLES Y SENIALES;
 22:  CALL RESET_SENIALES    ;
 23:  ;
 24:  WAIT DI[21:COGER CRISTAL]=ON    ;
 25:  ;
 26:  !MOV. COGER CRISTAL DE CONVEYOR
 27:  CALL COGER_CRISTAL    ;
 28:  ;
 29:  !MOV. DEJADA DE CRISTAL EN POS.;
 30:  CALL DEJAR_CRISTAL    ;
 31:  ;
 32:  WAIT DI[22:COGER LCD]=ON    ;
 33:  ;
 34:  !MOV. COGER LCD DE CONVEYOR;
 35:  CALL COGER_LCD    ;
 36:  ;
 37:  !MOV. DEJADA DE LCD EN POS.;
 38:  CALL DEJAR_LCD    ;
 39:  ;
 40:  !MOV. COGER CHAPA DE CONTENEDOR;
 41:  CALL COGER_CHAPA    ;
 42:  ;
 43:  !MOV. DEJADA DE CHAPA EN POS.;
 44:  CALL DEJAR_CHAPA    ;
 45:  ;
 46:  R[10:CNT_CICLO]=R[10:CNT_CICLO]+1    ;
 47:  ;
 48:  !INDICA FIN Y ESPERA CONFIRMACION;
 49:  DO[21:FIN R1]=ON ;
 50:  WAIT DI[20:INICIO DE CICLO]=OFF    ;
 51:  DO[21:FIN R1]=OFF ;
 52:  ;
 53:  JMP LBL[10] ;
 54:  ;
```

Las tareas encargadas de coger del conveyor de llegada y dejar en su posición final los paneles de cristal y LCD son esencialmente iguales, por lo tanto, con el fin de no alargar más la extensión del informe se va a comentar solo la maniobra realizada para uno de los paneles.

COGER_CRISTAL – Movimiento a la posición de cogida del panel

Se inicia la maniobra de cogida de cristales con la selección del sistema de referencia (base de trabajo), así como la asignación de la velocidad de movimiento según el valor establecido por el usuario desde la interfaz. Seguidamente, por seguridad, se realiza un reseteo del valor del registro de posición auxiliar con el objetivo de eliminar cualquier posible valor que haya podido quedar almacenado en él.

La secuencia comienza con un movimiento de aproximación y reorientación del robot de modo que la garra de ventosas quede correctamente posicionada respecto al conveyor de llegada de piezas. A continuación, se realiza un movimiento hasta la posición de cogida, con un offset en el eje Z (vertical) de 300mm con el fin de no golpear la pieza, y se finaliza con un movimiento lineal a menor velocidad hasta el panel.

Para la simulación de la acción de succión de la ventosa para que pueda extraerse la pieza del conveyor y llevarla así hasta su posición, se hace una llamada a un programa de simulación de dicha acción, el cual se ilustra en la siguiente figura.

Figura 118. *Simulación de succión del panel con la garra de ventosas de vacío.*

Al finalizar el programa de simulación se establece una espera de 500ms con el fin de solventar posibles travas en la ejecución de la propia simulación que lleven a un resultado poco vistoso.

Con el panel ya adherido a la garra de ventosas, se extra del conveyor con un nuevo '*offset*' de 500mm y se desplaza el robot hasta una nueva posición de paso intermedia que asegura que la orientación del robot es la adecuada para la siguiente tarea.

```
/PROG  COGER_CRISTAL
/MN
  1:  !/////////////////////////////////;
  2:  !COGIDA DE CRISTAL DE CONVEYOR;
  3:  !/////////////////////////////////;
  4:  ;
  5:  !SELECCION BASE DE TRABAJO;
  6:  UFRAME_NUM=0;
  7:  ;
  8:  !ESTABLECE VELOCIDAD DE HMI;
  9:  OVERRIDE=R[1:VELOCIDAD_OPC];
 10:  ;
 11:  !RESET PR AUX;
 12:  PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX]   ;
 13:  ;
 14:  !MOVIMIENTO DE APROXIMACION;
 15:J P[1] 100% CNT100    ;
 16:  ;
 17:  !SEGUIMIENTO DE CRISTAL;
 18:  PR[10,3:PR_AUX]=300    ;
 19:  ;
 20:L PR[2:TP_Cristal] 2000mm/sec CNT100 Offset,PR[10:PR_AUX]
 21:  ;
 22:L PR[2:TP_Cristal] 500mm/sec FINE    ;
 23:  !SIMULACION DE COGER CRISTAL;
 24:  CALL PICK  CRISTAL    :
```

DEJAR_CRISTAL – Movimiento hasta la posición final de depósito del panel

Al igual que en el caso anterior, el programa comienza con la selección el sistema de trabajo y establecimiento de la velocidad de movimiento, así como el reseteo del registro de posición auxiliar.

El movimiento nuevamente consta de un punto de aproximación hasta la posición donde va a ir a situado el panel. Una vez orientado el robot, se realizan en este caso dos movimientos de ajuste con diferentes '*offset*' con el fin de afinar el posicionamiento dentro del marco del televisor.

```
/PROG  DEJAR_CRISTAL
/MN
  1:  !/////////////////////////////////;
  2:  !DEPOSITO DEL CRISTAL DENTRO DELO;
  3:  !/////////////////////////////////;
  4:  ;
  5:  !ESTABLECIMIENTO DE VEL. DE HMI;
  6:  OVERRIDE=R[1:VELOCIDAD_OPC];
  7:  ;
  8:  !SELECCION DE BASE DE TRABAJO;
  9:  UFRAME_NUM=1;
 10:  ;
 11:  !RESET PR AUX;
 12:  PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX]   ;
 13:  ;
 14:  !MOVIMIENTO DE APROXIMACION;
 15:J P[1:Aproximacion 1] 100% CNT100    ;
 16:  ;
 17:  !AJUSTE OFFSET Y MOV. FINAL;
 18:  PR[10,2:PR_AUX]=50    ;
 19:  PR[10,3:PR_AUX]=100    ;
 20:L PR[5:DP_Cristal] 2000mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
 21:  ;
 22:  PR[10,2:PR_AUX]=0    ;
 23:  PR[10,3:PR_AUX]=50    ;
 24:L PR[5:DP_Cristal] 500mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
 25:  ;
 26:L PR[5:DP_Cristal] 100mm/sec FINE    ;
 27:  ;
```

Cuando el panel está posicionado correctamente se debe retirar la garra, para ello el primer paso es finalizar la succión de las ventosas. En este proyecto al tratarse de una simulación, se cuenta con diferentes soportes de montaje, para el funcionamiento manual y automático. Por lo tanto, finalizar con la succión se

simula de dos formas diferentes en función del modo de funcionamiento, por lo que es necesaria una variable, procedente del autómata que indique en qué modo se está realizando esta tarea (*DI[28:MODO AUTOMÁTICO]*).

A) *B)*

Figura 119. ***A)*** *Simulación de depósito en modo automático.*
B) *Simulación de depósito en modo manual.*

Una vez finalizada la simulación, con el panel ya situado en su posición final y desenganchado de la herramienta, se procede a retirar la garra y posicionar el robot en una posición adecuada para continuar con las siguientes tareas.

```
28:  !DEJADA DE CRISTAL ;
29:  IF DI[28:MODO AUTOMATICO]=OFF,JMP LBL[10] ;
30:  CALL DROP_CRISTAL_AUTO    ;
31:  WAIT    .20(sec) ;
32:  JMP LBL[20] ;
33:  LBL[10] ;
34:  CALL DROP_CRISTAL_MANUAL    ;
35:  WAIT    .20(sec) ;
36:  LBL[20] ;
37:  ;
38:  !RETIRADA DE GARRA ;
39:L PR[5:DP_Cristal] 500mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
40:  ;
41:J P[2:Aproximacion 2] 100% CNT100    ;
42:  ;
```

Tras finalizar la manipulación de los paneles de cristal y LCD (cuya maniobra es íntegramente igual a la descrita), se procede a la maniobra de manipulación de chapas de aluminio.

COGER_CHAPA – Movimiento de extracción de chapas del contenedor

La base del programa para la extracción de las chapas del contenedor se establece conforme a lo visto para el movimiento de cogida de paneles, de tal forma que los primeros pasos vuelven a ser la selección del sistema de referencia, establecimiento de la velocidad de movimientos y reseteo del registro de posición auxiliar.

Seguidamente se encuentra la peculiaridad de este programa, el cálculo de la distancia hasta la chapa que se debe coger. Para dicho cálculo se utilizan tres registros de datos, uno de ellos auxiliar para almacenar el resultado de la operación matemática entre los otros dos, el segundo se trata de un valor constante de separación entre chapas dentro del contenedor el cual se multiplica por el número de chapas de aluminio que ya han sido extraídas. De esta forma se obtiene el valor del '*offset*' que se debe aplicar al valor guardado de la posición de la primera de las chapas.

```
/PROG  COGER_CHAPA
/MN
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
 !EXTRACCION CHAPA DE CONTENEDOR ;
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  ;
 !SELECCION BASE DE TRABAJO ;
 UFRAME_NUM=0 ;
  ;
 !ESTABLECE VELOCIDAD DE HMI ;
 OVERRIDE=R[1:VELOCIDAD_OPC] ;
  ;
 !RESET PR AUX ;
 PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX]    ;
  ;
 !CALCULO POS. CHAPA ACTUAL ;
 R[12:AUX_CHAPA]=R[11:SEP_CHAPA]*R[15:CNT_CHAPA] ;
  ;
```

Una vez realizado el cálculo se realizan una serie de movimientos de aproximación y orientación de la herramienta para encararse frente al contendor y poder realizar la extracción de las chapas.

Cuando la herramienta se encuentra posicionada en la chapa correspondiente según el ciclo en que se encuentre el robot, se simula la succión por parte de las ventosas generando la adhesión de la pieza a la garra de forma análoga a lo visto en la *Figura 118* para los paneles de cristal. Con la chapa ya adherida a la herramienta, se procede a su extracción mediante movimientos ajustados a partir de '*offset'* en los diferentes ejes para evitar colisiones y finalizando con un movimiento al punto de partida que facilita la reorientación hacia el posterior programa de depósito de las piezas en su posición final.

```
 !MOVIMIENTO DE APROXIMACION ;
J P[1:Aproximacion] 100% CNT100    ;
 !MOVIMIENTO A POS. CALCULADA ;
 PR[10,1:PR_AUX]=(-500)    ;
L PR[4:TP_Chapa_Alumino] 2000mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
  ;
 PR[10,1:PR_AUX]=(-50)+R[12:AUX_CHAPA]    ;
L PR[4:TP_Chapa_Alumino] 2000mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
  ;
 PR[10,1:PR_AUX]=R[12:AUX_CHAPA]    ;
L PR[4:TP_Chapa_Alumino] 500mm/sec FINE Offset,PR[10:PR_AUX]    ;
  ;
 !COGIDA DE CHAPA ;
 CALL PICK_CHAPA_ALUMINIO    ;
 WAIT    .10(sec) ;
  ;
 !MOVIMIENTO DE EXTRACCION ;
 PR[10,3:PR_AUX]=20    ;
L PR[4:TP_Chapa_Alumino] 500mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
  ;
 PR[10,1:PR_AUX]=0    ;
L PR[4:TP_Chapa_Alumino] 500mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
  ;
 PR[10,1:PR_AUX]=(-500)    ;
L PR[4:TP_Chapa_Alumino] 2000mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
  ;
L P[1:Aproximacion] 2000mm/sec CNT100    ;
  ;
 R[15:CNT_CHAPA]=R[15:CNT_CHAPA]+1    ;
  ;
```

La maniobra de dejada de la chapa de aluminio sobre el marco del televisor es esencialmente igual a la descrita para el panel de cristal. Por lo tanto, no se comentará de nuevo, ya que la única diferencia existente en ambos programas es el punto de dejada, siendo idéntica la estructura y funcionamiento del proceso a realizar.

Capítulo 6.1.7. Programación robot manipulador M-710iC/45M

Diagrama de flujo R2 → M-710iC/45M

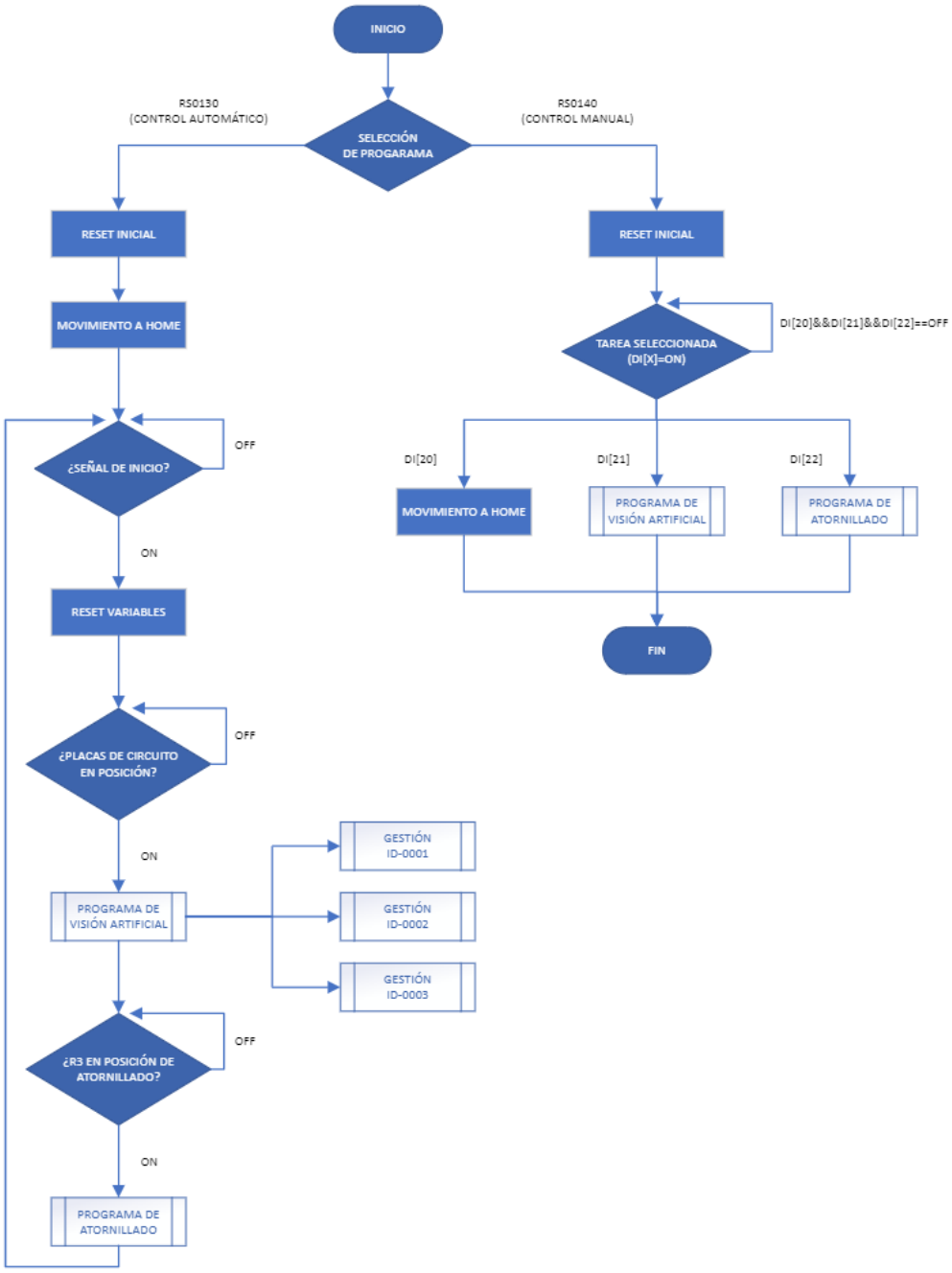


Figura 120. *Diagrama de flujo (código) robot M-710iC/45M.*

Tal y como puede observarse en el diagrama, el funcionamiento del robot se basa en lo visto anteriormente, se selecciona en el arranque uno de los dos programas principales enfocados en dos modos funcionamiento, automático y manual, que a su vez lanzan las diferentes tareas que componen la maniobra del robot.

La estructura de ambos programas principales es idéntica a la realizada en el primer robot. Por ello, se va a omitir el estudio del programa principal de gestión en modo automático, haciendo una breve alusión en este capítulo a la estructura del programa en modo manual, así como a las subrutinas que conforman la maniobra del robot.

RS0140 – Funcionamiento en modo manual

El programa comienza con un restablecimiento inicial de valores de señales y registros que puedan haber quedado guardados de una ejecución anterior, seguido de la selección del sistema de referencia y herramienta por defecto.

A continuación, se entra en un bucle infinito del que se sale cuando llega una señal de selección de la tarea que se va a realizar. A partir de aquí se lanza el procedimiento a realizar y una vez finalizado se saltan el resto de las acciones para finalizar el programa.

```
/PROG  RSR0140
/MN
  1:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  2:  !CONTROL MANUAL CON EL HMI ;
  3:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  4:  ;
  5:  !RESET INCIAL ;
  6:  CALL RESET_INICIAL   ;
  7:  ;
  8:  !HERRAMIENTA Y BASE DE TRABAJO ;
  9:  UTOOL_NUM=1 ;
 10:  UFRAME_NUM=0 ;
 11:  ;
 12:  LBL[10] ;
 13:  ;
 14:  IF DI[24:MANUAL HOME]=ON,JMP LBL[20] ;
 15:  IF DI[25:MANUAL PORG VISION]=ON,JMP LBL[30] ;
 16:  IF DI[26:MANUAL ATORNILLADO]=ON,JMP LBL[40] ;
 17:  ;
 18:  JMP LBL[10] ;
 19:  ;
 20:  !MOVIMIENTO A HOME ;
 21:  LBL[20] ;
 22:J PR[1:HOME] 100% FINE   ;
 23:  JMP LBL[50] ;
 24:  ;
 25:  !!INICIA PROG. DE VISION ;
 26:  LBL[30] ;
 27:  CALL PROG_VISION   ;
 28:  JMP LBL[50] ;
 29:  ;
 30:  !!INICIA SECUENCIA DE ATORNILLADO ;
 31:  LBL[40] ;
 32:  CALL ATORNILLADO   ;
 33:  JMP LBL[50] ;
 34:  ;
 35:  LBL[50] ;
 36:  ;
```

PROG_VISION – Reconocimiento de placas mediante iRVision

Este programa es el atractivo principal de la instalación, que dota a este robot de una funcionalidad adicional para la realización de tareas '*pick&place*'. Cabe destacar que sin una correcta calibración y configuración de la cámara de visión (realizada en el *Capítulo 6.1.5. Configuración iRVision*), todo lo que se va a exponer a continuación no sería funcional.

La tarea comienza estableciendo la herramienta de ventosas como el útil activo y el sistema de referencia de calibración de la cámara, que como ya se

indicó anteriormente es el que se ha utilizado para establecer las posiciones de referencia de llegada de placas.

Comienza a continuación el escaneo mediante visión artificial, con la instrucción '*VISION RUN_FIND 'LOCALIZADOR''* se toma una instantánea del campo de visión definido en la configuración '*LOCALIZADOR*' establecida para esta tares. Seguidamente, el comando '*VISION GET_NFOUND 'LOCALIZADOR' R[10]*' almacena en el registro R[10] el valor del número de piezas encontradas (que coincidan al menos en un 98% con alguno de los modelos establecidos) sobre la instantánea anteriormente tomada.

En el registro R[11] se realiza una copia del número de placas encontradas y se realiza una comprobación sobre el número de elementos localizados. En caso de ser 0 el valor de R[10] lo que indicaría que no se ha detectado ninguna placa, finalizaría la tarea, realizando un salto hasta el final del programa.

```
/PROG  PROG_VISION
/MN
  1:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  2:  !VISION IRVISION DETECTA ID PLACA ;
  3:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  4:  ;
  5:  !HERRAMIENTA Y BASE DE TRABAJO ;
  6:  UTOOL_NUM=1 ;
  7:  UFRAME_NUM=1 ;
  8:  ;
  9:  !INICIO ESCANEO POR VISION ;
 10:  VISION RUN_FIND 'LOCALIZADOR'    ;
 11:  VISION GET_NFOUND 'LOCALIZADOR' R[10]    ;
 12:  ;
 13:  R[11:Copia num placas]=R[10:Placas encontras]    ;
 14:  ;
 15:  IF R[10:Placas encontras]=0,JMP LBL[100] ;
 16:  ;
```

A continuación, se realiza un pequeño bucle, cuyo número de iteraciones depende del total de placas encontradas. Dentro de este bucle se almacena, en unos registros de posición reservados a la funcionalidad iRVision (VR[n]), la diferencia entre la posición y orientación observada, respecto a la definición del modelo de referencia, de las placas que se han localizado anteriormente con la instrucción '*VISION GET_OFFSET*'.

```
 17:  LBL[10] ;
 18:  ;
 19:  !GUARDA EL OFFSET EN VR ;
 20:  VISION GET_OFFSET 'LOCALIZADOR' VR[R[10]] JMP LBL[100] ;
 21:  ;
 22:  R[10:Placas encontras]=R[10:Placas encontras]-1    ;
 23:  ;
 24:  IF R[10:Placas encontras]<>0,JMP LBL[10] ;
 25:  ;
```

Una vez finalizado el bucle de almacenamiento de '*offset*', se realiza un nuevo bucle con la copia del número de placas encontrado (R[11]).

En primer lugar, se copia en un nuevo registro (R[12]) el valor del identificador de la placa cuyo offset se ha almacenado el registro VR[R[11]]; este valor se evalúa mediante sentencias condicionales (IF) y en función del ID se realiza la manipulación de la placa correspondiente.

Al finalizar la tarea de '*pick&place*' de la placa correspondiente se reduce en una unidad el valor del registro R[11] y se evalúa si el nuevo valor es diferente de 0, en cuyo caso se continua con el bucle de manipulación de placas según sus identificadores.

En caso de que el valor del registro R[11] sea cero al reducir su valor tras una tarea de manipulación, significaría que ya se han colocado todas las placas detectadas. Por lo tanto, se salta al final del programa, donde se le indica al autómata que el programa de visión artificial ha finalizado y seguidamente se vuelve al programa principal.

```
LBL[20] ;
 ;
R[12]=VR[R[11]].MODELID ;
 ;
 ;
IF R[12:ID de placa]=3,JMP LBL[40] ;
IF R[12:ID de placa]=2,JMP LBL[30] ;
 ;
!ID 0001 RECONOCIDO ;
CALL ID_0001    ;
R[11:Copia num placas]=R[11:Copia num placas]-1    ;
IF R[11:Copia num placas]<>0,JMP LBL[20] ;
JMP LBL[100] ;
 ;
!ID 0002 RECONOCIDO ;
LBL[30] ;
CALL ID_0002    ;
R[11:Copia num placas]=R[11:Copia num placas]-1    ;
IF R[11:Copia num placas]<>0,JMP LBL[20] ;
JMP LBL[100] ;
 ;
!ID 0003 RECONOCIDO ;
LBL[40] ;
CALL ID_0003    ;
R[11:Copia num placas]=R[11:Copia num placas]-1    ;
IF R[11:Copia num placas]<>0,JMP LBL[20] ;
 ;
LBL[100] ;
 ;
```

ID_0001 – Maniobra de manipulación de placas

Se describe en este apartado la manipulación de placas de circuito describiendo en caso concreto del modelo ID 0001, para el resto de los modelos la maniobra es idéntica cambiando la referencia de posiciones donde coger y dejar las placas.

El programa comienza con la asignación de la velocidad establecida a través de la interfaz por el operario. Seguidamente se reinicia el registro de posición auxiliar y se establecen los parámetros para el movimiento de aproximación, así mismo se establece el sistema de referencia con el que han sido grabados los registros de posición para evitar singularidades en el movimiento.

```
/PROG  ID_0001
/MN
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
 !MANIPULACION PLACAS CON ID-0002 ;
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  ;
 !ESTABLECE VEL. DEL HMI ;
 OVERRIDE=R[1:VELOCIDAD_OPC] ;
  ;
 !RESET PR AUX ;
 PR[10:OFFSET_AUX]=PR[10:OFFSET_AUX]-PR[10:OFFSET_AUX]    ;
  ;
 PR[10,2:OFFSET_AUX]=(-350)    ;
 PR[10,3:OFFSET_AUX]=250    ;
  ;
 !CAMBIO DE BASE DE TRABAJO ;
 UFRAME_NUM=1 ;
  ;
```

La maniobra comienza con un movimiento de reorientación a una posición auxiliar de paso que sirve tanto si el robot parte desde la posición *HOME,* es decir esta la primera de las placas en ser manipulada, como si el punto de partida es el de retirada de la herramienta tras dejar en su posición final una placa anterior.

Como se puede observar en las líneas 21, 24 y 25 del siguiente fragmento del código, los movimientos de aproximación son los que están asignados con la instrucción '*Offset,PR[]*' previo a la instrucción '*VOFFSET,VR[]*' que, como se comentó anteriormente, asigna la diferencia observada por la cámara de posición y orientación respecto a la posición guardada.

La simulación de toma de placas es esencialmente la misma vista para los paneles de cristal y LCD, así como la extracción de chapas de aluminio del contenedor. Una vez la herramienta se encuentra en posición, se debe simular la succión por parte de las ventosas para que la placa se adhiera a la garra de ventosas de vacío, esta acción se simula con la llamada ('*CALL*') al programa *'PICK_ID_0001'.* Una vez adherida la placa esta se extrae de la cinta.

```
17:  !MOVIMIENTO AUX DE RETIRADA ;
18:J PR[8:AUX_1] 100% CNT100    ;
19:  ;
20:  !MOVIMIENTO A POSICION DE COGIDA ;
21:L PR[5:TP_REF_ID_0001] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
22:  ;
23:  PR[10,2:OFFSET_AUX]=0    ;
24:L PR[5:TP_REF_ID_0001] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
25:  ;
26:L PR[5:TP_REF_ID_0001] 500mm/sec FINE VOFFSET,VR[R[11]]    ;
27:  ;
28:  !COGE PLACA ;
29:  CALL PICK_ID_0001    ;
30:  WAIT    .50(sec) ;
31:  ;
32:  !EXTRACCION PLACA ;
33:L PR[5:TP_REF_ID_0001] 500mm/sec CNT25 Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
34:  ;
35:  PR[10,2:OFFSET_AUX]=(-350)    ;
36:L PR[5:TP_REF_ID_0001] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
37:  ;
```

Después de extraer la placa de la cinta de llegada, se realiza un nuevo movimiento de reorientación y aproximación, a través del primer punto de paso descrito para la aproximación a la cinta, pero esta vez en la dirección contraria. Seguidamente, al tratarse de una nueva operación (a pesar de estar dentro del mismo programa), se comprueba la velocidad asignada por el operario y se establece en caso de que haya cambiado.

Se debe cambiar nuevamente la base de trabajo, pues la nueva tarea se realiza con puntos tomados respecto al marco del televisor y no con el sistema de referencia de configuración de la cámara.

```
38:  !MOVIMIENTO AUX A POS. DEJADA ;
39:J PR[8:AUX_1] 100% CNT100    ;
40:  ;
41:  !ESTABLECE VEL. DEL HMI ;
42:  OVERRIDE=R[1:VELOCIDAD_OPC] ;
43:  ;
44:  !CAMBIO DE BASE DE TRABAJO ;
45:  UFRAME_NUM=2 ;
46:  ;
47:  !MOVIMIENTO A POS. DEJADA ;
48:  PR[10,2:OFFSET_AUX]=0    ;
49:J PR[2:DP_ID_0001] 100% CNT25 Offset,PR[10:OFFSET_AUX]    ;
50:L PR[2:DP_ID_0001] 500mm/sec FINE    ;
51:  ;
```

Una vez establecido el nuevo sistema de referencia para esta tarea, se procede a la aproximación y depósito de la placa en su posición final. Como ocurría con el robot R1, para que la simulación de depósito de placas se realice correctamente, se debe conocer si esta operación se está realizando en modo manual o automático para seleccionar el programa de simulación de dejada adecuado.

Tras depositar la placa en su posición final, la garra se retira y se vuelve al programa de visión.

```
52:  !DEJA PLACA ;
53:  IF DI[27:MODO AUTOMATICO]=OFF,JMP LBL[10] ;
54:  CALL DROP_ID_0001_AUTO    ;
55:  WAIT    .50(sec) ;
56:  JMP LBL[20] ;
57:  LBL[10] ;
58:  CALL DROP_ID_0001_MANUAL    ;
59:  WAIT    .50(sec) ;
60:  LBL[20] ;
61:  ;
62:  !RETIRADA DE LA HERRAMIENTA ;
63:L PR[2:DP_ID_0001] 500mm/sec CNT25 Offset,PR[10:OFFSET
/POS
/END
```

ATORNILLADO – Maniobra de atornillado de tapas

En el caso que se está describiendo en este capítulo (funcionamiento manual), al finalizar el programa de visión se volvería al programa RSR0140 y finalizaría el control del robot, teniendo que ser indicada desde el panel de gestión la siguiente tarea. En caso de encontrarnos en el modo automático, al volver al programa principal (RSR0130) se pasaría directamente al programa de atornillado.

Este programa basa su funcionamiento en dos movimientos de aproximación y un movimiento de retirada de la herramienta, entorno a diferentes puntos, con una espera intermedia entre los dos últimos movimientos. Durante este tiempo de espera, que pretende representar la introducción de tornillos y su propio atornillado, no se depositará ninguna pieza en los huecos destinados a esta función, con el objetivo de no sobre cargar la simulación.

Se inicia la tarea estableciendo la velocidad asignada desde la interfaz, así como la herramienta (en este caso la diseñada específicamente para esta tarea) y el sistema de referencia del marco del televisor con la tapa ya colocada.

La secuencia comienza con un movimiento a una posición segura, donde no interfiera con el posicionamiento de la tapa, seguido de la activación de la señal que indica al robot R3 que puede proceder con la colocación de la tapa para su atornillado.

Seguidamente se reinicia el valor del registro de posición auxiliar para posteriormente establecer el '*offset*' en el eje Z (respecto al sistema de referencia seleccionado).

```
/PROG  ATORNILLADO
/MN
   1:  !////////////////////////////// ;
   2:  !ATORNILLADO DE TAPAS ;
   3:  !////////////////////////////// ;
   4:  ;
   5:  !ESTABLECE VEL. DEL HMI ;
   6:  OVERRIDE=R[1:VELOCIDAD_OPC] ;
   7:  ;
   8:  !HERRAMIENTA Y BASE TRABAJO ;
   9:  UTOOL_NUM=2 ;
  10:  UFRAME_NUM=2 ;
  11:  ;
  12:  !MOVIMIENTO DE APROXIMACION ;
  13:J PR[9:AUX_2] 100% FINE    ;
  14:  DO[22:R2 POS. PRE-ATORNILLADO]=ON ;
  15:  ;
  16:  !RESET PR AUX Y OFFSET_Z ;
  17:  PR[10:OFFSET_AUX]=PR[10:OFFSET_AUX]-PR[10:OFFSET_AUX];
  18:  PR[10,3:OFFSET_AUX]=50    ;
  19:  ;
```

La secuencia de atornillado como tal da comienzo tras la espera de la señal de confirmación de parte del robot R3 de que la tapa ya está posicionada y lista para ser atornillada.

Una vez recibida esta señal, se comienza con la secuencia que se puede observar en el siguiente fragmento de código. Un movimiento de aproximación al punto de atornillado el '*offset*' de 50mm establecido, una aproximación lineal lenta hasta tocar con la tapa y una espera que simula el atornillado finalizando con un movimiento lineal de retirada de la herramienta.

```
  20:  !ESPERA R3 EN POSICION ;
  21:  WAIT DI[22:R3 POS. PRE-ATORNILLADO]=ON ;
  22:  ;
  23:  !INICIO DEL ATORNILLADO ;
  24:J P[1] 100% CNT25 Offset,PR[10:OFFSET_AUX];
  25:L P[1] 100mm/sec FINE    ;
  26:  WAIT    .50(sec) ;
  27:L P[1] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
  28:  ;
  29:L P[2] 3000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
  30:L P[2] 100mm/sec FINE    ;
  31:  WAIT    .50(sec) ;
  32:L P[2] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
                                  :
                                  :
  43:  ;
  44:L P[5] 3000mm/sec CNT100    ;
  45:J P[11] 100% CNT100    ;
  46:  ;
  47:L P[6] 3000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]  ;
  48:L P[6] 100mm/sec FINE    ;
  49:  WAIT    .50(sec) ;
  50:L P[6] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]    ;
                                  :
                                  :
```

Como se puede observar en el fragmento de código anterior, en las líneas 44 y 45 se han tenido que añadir algunos movimientos de reorientación posteriores

para evitar posibles colisiones con del robot con la tapa o con el robot que la sostiene en su posición.

Dos de los puntos, de los ocho puntos de atornillado existentes, no son fácilmente accesibles con el robot R3 en su posición, por lo tanto, es necesario que se retire el robot encargado de sujetar la tapa, cuando los seis puntos previos ya hayan sido atornillado y este asegurada la sujeción de la propia tapa en su posición. Para ello, con las dos señales cruzadas entre los robots R2 y R3 se hace una comprobación dual que confirme que R2 está en una posición segura para la retirada de R3 sin colisiones, y otra que confirme que R3 ya ha sido retirado.

Con el robot R3 ya retirado, se finaliza el atornillado de los dos puntos restantes.

```
59:  !R2 EN POSICION SEGURA ;
60:  DO[23:R2 POS. SEGURA PARA R3]=ON ;
61:  ;
62:  !Espera retirada de R3 ;
63:  WAIT DI[22:R3 POS. PRE-ATORNILLADO]=OFF  ;
64:  ;
65:J P[8] 100% CNT25 Offset,PR[10:OFFSET_AUX] ;
66:L P[8] 100mm/sec FINE    ;
67:  WAIT   .50(sec) ;
68:L P[8] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]    ;
69:  ;
70:L P[9] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]    ;
71:L P[9] 100mm/sec FINE    ;
72:  WAIT   .50(sec) ;
73:L P[9] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]    ;
74:  ;
75:J P[10] 100% CNT100    ;
76:  ;
```

Todos los puntos, ya sean de atornillado o de paso para reorientar la herramienta y evitar problemas de colisión se almacenan el final de cada programa (en el caso de puntos locales P[]). El siguiente fragmento de código ejemplifica el caso de alguno de los puntos tomados de forma local para la tarea de atornillado.

```
/POS
P[1]{ GP1: UF : 2, UT : 2,       CONFIG : 'N U T, 0, 0, 0',
X =15.457  mm, Y = 648.643  mm, Z =2.983  mm,
W =-179.983 deg, P = -.834 deg, R = -179.696 deg
};
                                  .
                                  .
                                  .
P[11]{ GP1: UF : 2, UT : 2,      CONFIG : 'N U T, 0, 0, 0',
X=262.462  mm,  Y=15.277  mm,  Z=353.911  mm,
W=180.000 deg,   P=0.000 deg,   R=0.000 deg
};
/END
```

Estos puntos pueden ser grabados en configuración cartesiana o por ejes, quedando reflejado como tal en el código.

Capítulo 6.1.8. Programación robot manipulador R-2000iC/165F

Diagrama de flujo R3 → R-2000iC/165F

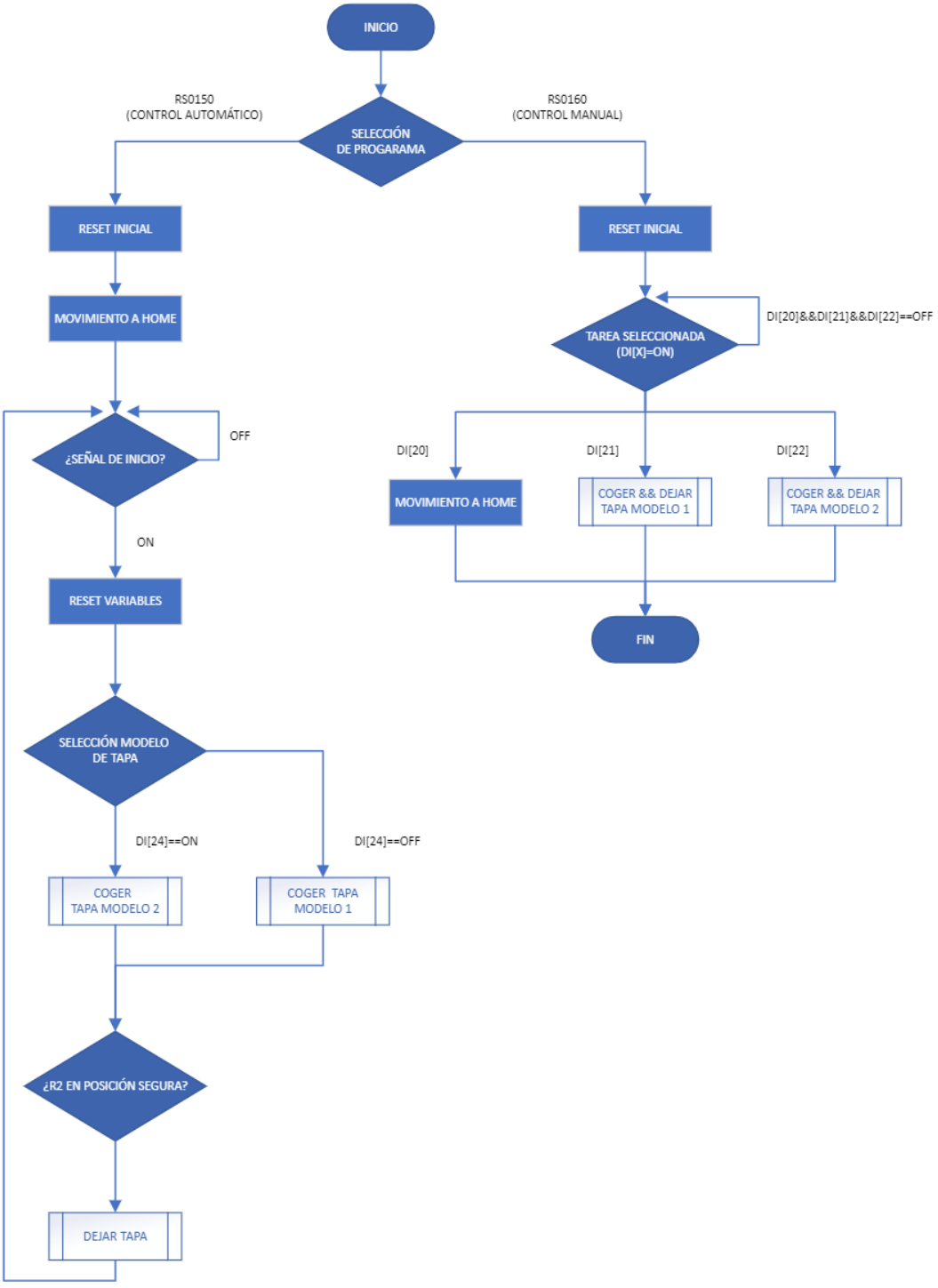


Figura 121. *Diagrama de flujo (código) robot R-2000iC/165F.*

Como ya se ha indicado con anterioridad, y a su vez se ha analizado ya en dos ocasiones, el funcionamiento generalizado se divide en dos modos, automático y manual. Ambos programas principales ya han sido analizados previamente,

por lo tanto, ninguno de ellos se va a estudiar en este epígrafe, quedando recogidos en la carpeta dedicada al software Roboguide en los anejos a la memoria, donde se pueden encontrar todos los programas de los robots.

Se analiza por lo tanto la maniobra de manipulación de una de las tapas, pues la estructura para ambos modelos es idéntica, cambiando los sistemas de referencia (el de cada contendor de los diferentes modelos de tapa) y así mismo los puntos de cogida y dejada por la geometría de cada modelo.

COGER_MODELO_1 – Movimiento de cogida de tapas modelo 1

Esta tarea está basada en el procedimiento de extracción de chapas de aluminio del contenedor del robot R1. Como ya se vio en el programa mencionado, los primeros pasos consisten en comprobar parámetros como la velocidad de trabajo seleccionada a través del autómata e indicada al robot a través del autómata, establecimiento del sistema de referencia que se va a utilizar, y por último el reinicio de los registros auxiliares para el cálculo de la posición de la tapa que se va a manipular en cada ciclo.

```
/PROG  COGER_MODELO_1
/MN
  1:  !////////////////////////////// ;
  2:  !EXTRACCION DE TAPAS MODELO 1 ;
  3:  !////////////////////////////// ;
  4:  ;
  5:  !ESTABLECE VELOCIDAD CON HMI ;
  6:  OVERRIDE=R[1:VELOCIDAD_OPC] ;
  7:  ;
  8:  !SELECCION BASE TRABAJO ;
  9:  UFRAME_NUM=1 ;
 10:  ;
 11:  !RESET PR AUX ;
 12:  PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX]   ;
 13:  ;
 14:  !CALCULO POSICION DE TAPA ACTUAL ;
 15:  R[14:Z_OFFSET]=R[14:Z_OFFSET]-R[14:Z_OFFSET]   ;
 16:  R[14:Z_OFFSET]=R[11:CNT_MODELO1]*R[13:SEP_TAPAS]   ;
 17:  ;
```

Una vez calculado el '*offset*' que se debe aplicar en el eje Z (respecto al sistema de referencia activo), para alcanzar correctamente la tapa indicada, comienza la secuencia de movimiento con un punto de aproximación y reorientación hacia el contendor de tapas. Una vez orientado el robot se realizan los movimientos de aproximación a la tapa, así como la simulación de adhesión de la tapa a la ventosa de vacío como se ilustra a continuación.

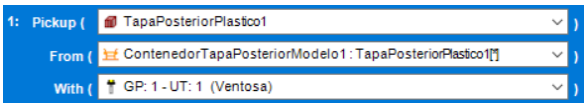


Figura 122. *Simulación 'Pickup Modelo 1'.*

Una vez adherida la tapa a la garra de ventosas, en primer lugar, se aumenta el contador de chapas extraídas para el cálculo del siguiente ciclo, seguido de un ajuste del valor de los valores del '*offset*' que se van a aplicar para los movimientos de retirada de la tapa del contenedor.

En último lugar, se vuelve a la posición grabada como punto *HOME,* pasando previamente por el punto de aproximación y reorientación que se utilizó al inicio de este mismo programa. Estos dos movimientos son estrictamente necesarios debido a las dimensiones de las tapas para evitar posibles colisiones con los contenedores, dejando el robot en una posición y configuración óptima para la siguiente tarea.

```
18: !MOVIMIENTO DE APROXIMACION ;
19:J P[1] 100% CNT100    ;
20:  ;
21: !MOVIMIENTO HACIA PTO. DE COGIDA ;
22:  PR[10,3:PR_AUX]=300    ;
23:J PR[2:TP_Modelo1] 100% CNT100 Offset,PR[10:PR_AUX]    ;
24:  ;
25:  PR[10,3:PR_AUX]=0-R[14:Z_OFFSET]    ;
26:L PR[2:TP_Modelo1] 500mm/sec FINE Offset,PR[10:PR_AUX]    ;
27:  ;
28: !COGER TAPA ;
29:  CALL PICK_MODELO_1    ;
30:  WAIT    .50(sec) ;
31:  R[11:CNT_MODELO1]=R[11:CNT_MODELO1]+1    ;
32:  ;
33: !RETIRA TAPA DE CONTENEDOR ;
34:  PR[10,2:PR_AUX]=50    ;
35:  PR[10,3:PR_AUX]=0    ;
36:L PR[2:TP_Modelo1] 500mm/sec CNT25 Offset,PR[10:PR_AUX]    ;
37:  ;
38:  PR[10,3:PR_AUX]=300    ;
39:L PR[2:TP_Modelo1] 1000mm/sec CNT100 Offset,PR[10:PR_AUX]    ;
40:  ;
41: !APROXIMACION DE SALIDA ;
42:L P[1] 3000mm/sec CNT100    ;
43:  ;
44: !VUELTA A HOME ;
45:J PR[1:HOME] 66% CNT100    ;
46:  ;
```

DEJAR_MODELO_1 – Movimiento de depósito de tapas modelo 1

En caso de encontrarse el sistema, y por lo tanto el robot en modo de funcionamiento automático, el acceso a este programa irá precedido de una espera hasta que una señal procedente del robot R2 indique que este ya se encuentra preparado, y en una posición segura, alejada del rango de actuación de R3, para que esta tarea se lleve a cabo.

Como viene siendo habitual, el programa comienza con la selección de la base de trabajo (en este caso el marco del televisor para un posicionamiento preciso), establecimiento de la velocidad de actuación y reinicio del registro de posición auxiliar.

```
/PROG  DEJAR_MODELO_1
/MN
   1:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
   2:  !DEPOSITO TAPA MODELO 1 EN POSICN ;
   3:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
   4:   ;
   5:  !ESTABLECE VELOCIDAD CON HMI ;
   6:  OVERRIDE=R[1:VELOCIDAD_OPC] ;
   7:   ;
   8:  !SELECCION BASE DE TRABAJO ;
   9:  UFRAME_NUM=3 ;
  10:   ;
  11:  !RESET PR AUX ;
  12:  PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX]    ;
  13:   ;
  14:  !MOVIMIENTO DE APROXIMACION ;
  15:J PR[7:AUX_1] 100% CNT100    ;
  16:   ;
```

Seguidamente se procede a un movimiento de aproximación y reorientación previo, recordando que se parte de la posición *HOME* donde se finalizó la extracción de la tapa, para finalizar el posicionamiento con tres movimientos

previos de aproximación al marco del televisor. Una vez posicionada la tapa se indica al robot R2 que puede proceder con la tarea de atornillado.

```
17:  !MOVIMIENTO A POSCION DEJADA ;
18:  PR[10,3:PR_AUX]=(-250)   ;
19:  PR[10,2:PR_AUX]=50   ;
20:L PR[4:DP_Modelo1] 2000mm/sec CNT100 Offset,PR[10:PR_AUX]
21:  ;
22:  PR[10,3:PR_AUX]=(-20)   ;
23:  PR[10,2:PR_AUX]=50   ;
24:L PR[4:DP_Modelo1] 500mm/sec CNT100 Offset,PR[10:PR_AUX]   ;
25:  ;
26:  PR[10,2:PR_AUX]=0   ;
27:L PR[4:DP_Modelo1] 100mm/sec CNT100 Offset,PR[10:PR_AUX]   ;
28:  ;
29:L PR[4:DP_Modelo1] 100mm/sec FINE   ;
30:  !R3 EN POSICION DE ATORNILLADO ;
31:  DO[22:R3 EN POS. ATORNILLADO]=ON ;
32:  ;
```

Tal y como ha ocurrido con todas las tareas de depósito de piezas en sus posiciones finales, la simulación de dejada depende de una señal indicativa del modo de funcionamiento contando con dos programas para esta labor que evitan incoherencias en la simulación global, prevenir que puedan desaparecer o superponerse piezas.

```
33:  !DEJAR TAPA ;
34:  IF DI[28:MODO AUTOMATICO]=OFF,JMP LBL[10] ;
35:  CALL DROP_MODELO_1_AUTO   ;
36:  WAIT    .50(sec) ;
37:  JMP LBL[20] ;
38:  LBL[10] ;
39:  CALL DROP_MODELO_1_MANUAL   ;
40:  WAIT    .50(sec) ;
41:  LBL[20] ;
42:  ;
```

Como se vio en la tarea de atornillado del robot R2, los dos últimos puntos de atornillado no son alcanzables si el presente robot (R3) se encuentra sujetando la tapa. Por lo tanto, cuando los seis puntos previos de atornillado se hayan realizado, el robot R2 se coloca en una posición segura, indicando a R3 que este puede retirarse.

La retirada consta de un movimiento lineal lento de retirada de la garra previo a la vuelta a la posición *HOME*, con un punto de reorientación intermedio para evitar cualquier posible singularidad.

```
43:  !ESPERA A R2 EN POSCION FINAL ;
44:  WAIT DI[23:R2 POS. SEGURA]=ON   ;
45:  ;
46:  !RETIRADA DE GARRA ;
47:  PR[10,2:PR_AUX]=0   ;
48:  PR[10,3:PR_AUX]=(-250)   ;
49:L PR[4:DP_Modelo1] 2000mm/sec CNT100 Offset,PR[10:PR_AUX]   ;
50:  ;
51:J PR[7:AUX_1] 100% CNT100   ;
52:  ;
53:  DO[22:R3 EN POS. ATORNILLADO]=OFF ;
54:  ;
55:J PR[1:HOME] 100% FINE   ;
56:  ;
```

Como se puede observar en el código anterior, después del paso por el punto de reorientación (línea 51) se le indica al robot R2 que el robot ya ha sido retirado hacia una posición donde no interfiera, para que este pueda finalizar el atornillado de los dos puntos restantes.

Capítulo 6.2. Desarrollo en TIA Portal

Una vez realizado el estudio sobre la programación de los robots que componen la célula se procede a explicar cómo se ha organizado y programado el control de esta haciendo uso de un autómata programable de la gama S7-1200 de Siemens, en concreto una CPU S7-1214c.

La programación se ha llevado a cabo a través del software TIA Portal, implementando el control mediante una estructura modular, según la cual, se cuenta con un bloque principal general que a su vez realiza llamadas a otros bloques para la ejecución de tareas específicas.

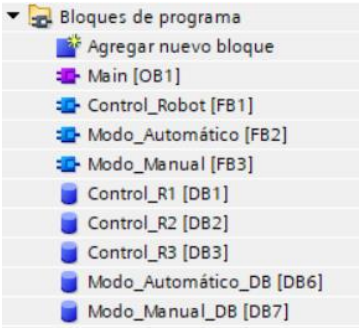


Figura 123. *Bloques de programa TIA Portal.*

Capítulo 6.2.1. Main [OB1]

Se trata del bloque principal del programa. Este es ejecutado de forma cíclica y en cada ciclo que se ejecutan secuencialmente los segmentos con las instrucciones asignadas como se muestra en la siguiente figura.

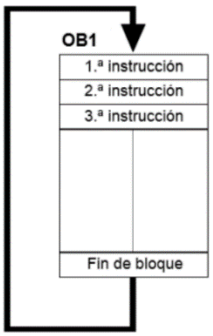


Figura 124. *Ejecución cíclica del programa principal.*

Este OB está formado por ocho segmentos con los que se gestionan las llamadas a los bloques de función para el control de los robots individualmente y los diferentes modos de funcionamiento de la instalación, buscando que el programa principal sea lo más organizado posible.

Todos los bloques de función que son llamados desde el programa principal cuentan con una serie de condiciones que deben cumplir para que no sean llamados en cada ciclo, sino que solo sean llamados cuando deban ejecutarse. Así mismo, estos bloques pueden contar con entradas y salidas, así como asignaciones de activación sobre variables cuando el bloque se encuentra activo.

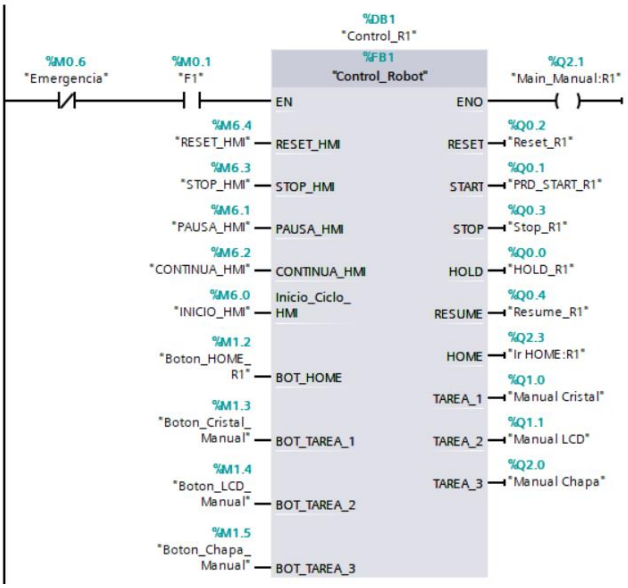


Figura 125. Segmento de control individual del robot R1.

Desde este bloque de programa principal se gestionan también algunas acciones que deben ser comprobadas en cada ciclo de ejecución como la velocidad asignada por el usuario desde el HMI para el movimiento de los robots,

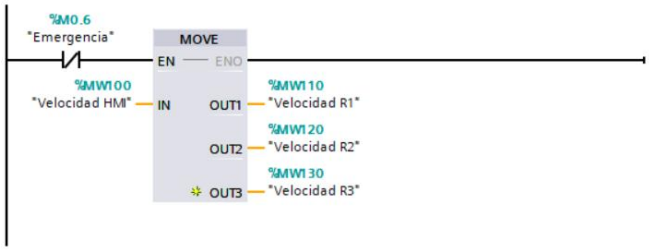


Figura 126. Segmento de asignación de velocidad en OB1.

Es imprescindible también que en cada ciclo se compruebe el estado de la variable de entrada, procedente del HMI, de señalización de emergencia. En cuyo caso, se desactivará cualquier bloque de función que se encontrara en ejecución e indicará a los robots que paren su ejecución. A su vez, al contar con un botón de rearme en la interfaz de usuario, este debe ser programado en el autómata, restableciendo el estado de la señal de parada de los robots.

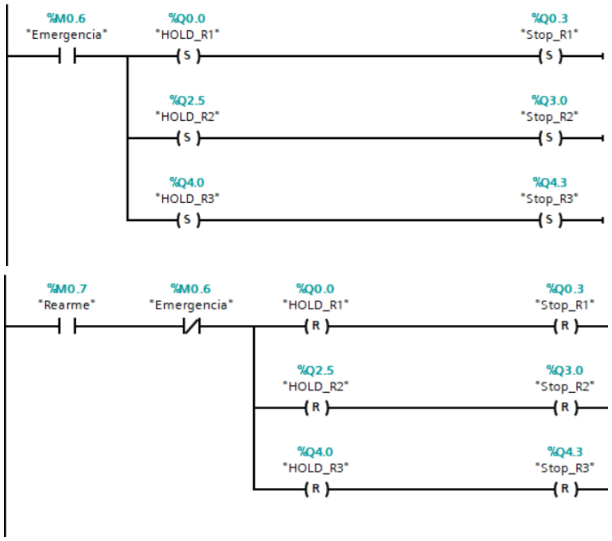


Figura 127. *Segmentos de estado de emergencia y rearme de la estación.*

Capítulo 6.2.2. Control_Robot [FB1]

Este bloque de función gestiona el control individual de los robots. Al tratarse de un bloque con memoria, cada llamada desde el bloque principal debe realizarse indicado el bloque (DB) de datos asignado para cada robot.

Los segmentos que se encargan de la gestión del estado de funcionamiento del robot, es decir, realizar una pausa, continuar con un programa pausado, restablecer los errores del robot, finalizar un programa antes de que termine su ejecución, etc. Son segmentos simples, es decir, el cumplimiento de una condición (botón pulsado en el HMI) activa una variable de salida del autómata (señal de entrada al robot).

Para la asignación de tareas al robot, se deben cumplir una serie de condiciones excluyentes, basadas principalmente en la selección de una de las tareas desde la interfaz de usuario y que no este comanda la ejecución de ninguna de las tareas restantes.

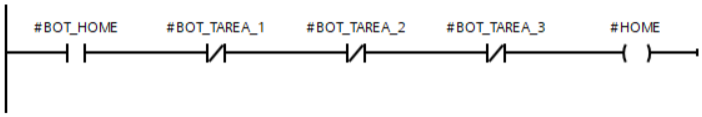


Figura 128. *Asignación de tarea 'movimiento a posición HOME'.*

Capítulo 6.2.3. Modo_Automático [FB2]

El desarrollo de este proyecto se ha realizado en torno a este modo de funcionamiento. Es por lo tanto es más extenso y complejo, por la cantidad de

variables que se deben coordinar, para asegurar el correcto funcionamiento de los robots, así como del proceso. En la siguiente figura se muestra la llamada a esta función desde el bloque de organización principal.

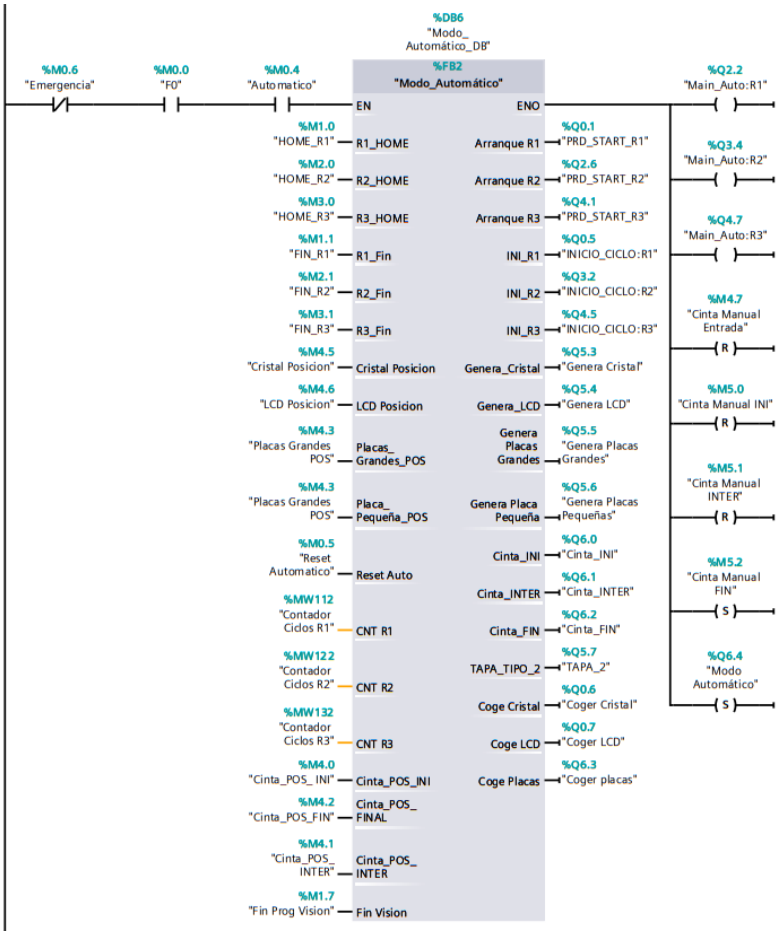


Figura 129. *Bloque de función modo automático.*

En caso de que el lector lo desee, puede acudir al apartado *Modo Automático (FB2)* perteneciente al *ANEXO A. Bloques de programa TIA Portal* para consultar la documentación gráfica de los segmentos a los que se hace alusión a lo largo del presente capítulo, pues no serán representados durante el estudio y análisis de estos por su extensión.

El primer segmento de este bloque consiste en una comprobación de primer ciclo donde se realiza un reinicio de los valores de algunas de las principales variables de salida, con el objetivo de asegurar que todas las condiciones iniciales se cumplen y así evitar posibles estados donde el sistema pueda quedarse paralizado por el incumplimiento de las condiciones necesarias para continuar con el proceso.

Seguidamente, el segundo segmente se encargar de comprobar que el primer robot se encuentre en su posición HOME y los soportes que se desplazan en

torno a la posición de trabajo de R1 en sus posiciones iniciales para comandar la entrada del soporte móvil sobre el que se realiza el montaje a la estación.

Una vez se encuentre el soporte en la posición de trabajo de R1, este comienza su ciclo de trabajo (tercer segmento). En el cuarto segmento, el inicio del ciclo de R1 supone el intercambio del soporte móvil a otro elemento de simulación (necesario para dotar de continuidad y realismo al proceso) y la entrada de los paneles de cristal y LCD al sistema. Cuando estos paneles se encuentren en su posición de manipulación, se debe indicar esta situación a R1 (quinto segmento).

El robot R1 continuará con sus tareas has finalizarlas, instante en que se debe reestablecer la posición inicial de los paneles para que la simulación realmente otorgue al usuario la sensación de proceso continuo y no de un conjunto de eventos (sexto segmento). Cabe destacar, como ya se ha hecho en otras ocasiones que, en una programación de un proyecto real, instrucciones de este tipo no sería necesarias por tratarse realmente de un proceso continuo.

En el séptimo segmento, la condición de finalización de las tareas de R1 que coincida con el estado de los robots R2 y R3 en sus respectivas posiciones *HOME,* y simultáneamente con los estados iniciales de los soportes que se desplazan entorno a la posición de trabajo de estos dos últimos robots mencionados activan el movimiento del soporte que se encuentra actualmente en la posición de trabajo de R1 y el reinicio de la variable de inicio de ciclo de este robot.

Cuando el soporte alcance la posición de trabajo, estando R2 en posición *HOME*, se habilita el inicio de ciclo de este robot (segmento 8).

El siguiente segmento realiza un cálculo en función del valor de la variable que almacena el número de ciclo del robot R2. Con el valor obtenido se definirá el número de placas que tendrá el televisor y por lo tanto el modelo de tapa que se debe montar.

En el décimo segmento, igual que sucedía con el cuarto, el comienzo de ciclo de R2 y la desactivación de la variable de posición *HOME* supone que la simulación de cambio de soporte ya ha sido completada, lo que permite restablecer la posición del soporte intermedio a su estado inicial. A su vez, el inicio de ciclo de R2 supone activar la señal de movimiento de placas, en función de lo anteriormente calculado, así como la señal de selección del modelo de tapa trasera (en caso de que proceda según lo calculado anteriormente) y, por lo tanto, en este instante se puede dar paso al inicio de ciclo del robot R3 (undécimo segmento).

Una vez las placas hayan llegado a la posición de visión artificial debe ser comunicado al robot R2 (segmento 12). Seguidamente, si la labor de visión ha finalizado, se reinicia el estado de posición de las placas de circuito (segmento 13), para que la simulación sea coherente y en el siguiente ciclo no se generen las placas ya en la posición final.

Los dos últimos segmentos hacen referencia al movimiento de salida de del soporte móvil de la instalación (decimocuarto segmento) una vez ambos robots, R2 y R3, hayan finalizado sus tareas, Así como al reinicio del valor por defecto del propio soporte cuando este ha sido eliminado de la simulación, instante en que se produce de forma simultánea el reinicio de las variables de comienzo de ciclo de ambos robots (decimoquinto segmento).

Cabe destacar que las variables (*INI_R1, INI_R2, INI_R3*) encargadas de indicar a sus respectivos el inicio de ciclo cuando se dan las condiciones para que estas se activen (paso de 0 a 1 lógico), son las mismas variables que permiten realizar la comprobación dual de finalización de ciclo, de la que se habló en los capítulos de programación de los robots, cuando estas se desactivan (paso de 1 a 0 lógico).

Capítulo 6.2.4. Modo_Manual [FB3]

El objetivo de este bloque de función es comandar las indicaciones realizadas por el usuario a través de la interfaz a las maquinas (no robots) que forman la instalación.

Por lo tanto, desde este bloque de función se indican movimientos de avance y retroceso del soporte móvil, movimientos de los paneles a las posiciones de manipulación y vuelta a sus puntos de partida, desplazamiento de las placas de circuito a la posición de reconocimiento mediante visión artificial y asignación del modelo de tapa trasera que se debe montar.

Avance y retroceso del soporte móvil

Como se analizó en el *Capítulo 6.1.3. Configuración de los componentes de la estación,* observable en la *Figura 96,* el soporte asignado al funcionamiento en modo manual cuenta con cuatro posibles posiciones, asignadas a su vez a cuatro variables del autómata. Por lo tanto, el movimiento del robot a cada una de las posiciones contempladas se rige según la siguiente tabla de la verdad.

	Señal posición de entrada	Señal posición inicial	Señal posición intermedia	Señal de posición final
Pos. Entrada (0)	1	0	0	0
Pos. Inicial (6253)	0	1	0	0
Pos. Intermedia (11101)	0	0	1	0
Pos. Final (18066)	0	0	0	1

Tabla 2. *Movimiento del soporte móvil en modo manual.*

Para programar estos movimientos, se han utilizado tres variables auxiliares de cambios de posición, entrada-inicial, inicial-intermedio e intermedio-final.

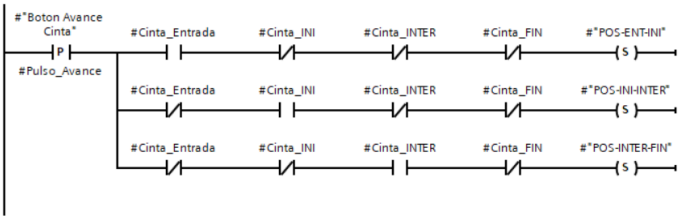


Figura 130. *Activación de cambios de posición con flanco ascendente.*

Con este planteamiento el funcionamiento es el siguiente: si el soporte se encuentra en la posición de entrada y se recibe pulso ascendente de la señal de avance, se activará la señal entrada-inicial, que en un segmento posterior activará la señal de posición inicial y desactivará la señal de posición de entrada.

Figura 131. *Movimiento entre posiciones de entrada e inicial.*

En caso de selección de un movimiento de retroceso del soporte, el funcionamiento es el análogo, se reestablece el estado de la variable entrada-inicial, restableciendo el valor de la variable de posición inicial y estableciendo como activa la variable de posición de entrada.

El lector puede consultar la programación de este apartado con mayor lujo de detalle en el *ANEXO A. Bloques de programa TIA Portal* dentro de la sección *Modo Manual (FB3)*.

Movimiento de los paneles de cristal y LCD

El funcionamiento de estos dos apartados es eminentemente más simple que lo visto para los movimientos de avance y retroceso del soporte móvil. En este

caso, el cumplimiento de una condición simple de entrada que indica al autómata la selección del botón de entrada de paneles activa la salida que controla el movimiento de estos paneles en la instalación. En caso de que se desenclave el botón pulsado (se debe hacer así por tratarse de un proyecto de simulación) los paneles vuelven a su posición de partida.

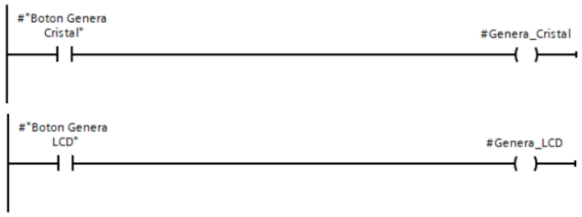


Figura 132. *Movimiento de paneles en modo manual.*

Movimiento de placas de circuito y selección del modelo de tapa.

El funcionamiento es muy simular al visto para los paneles. En este caso, se ha optado por dotar a la interfaz de dos botones de selección del modelo de televisor que se desea ensamblar. El primer modelo requiere tan solo las placas de tamaño mayor y la tapa posterior es la utilizada por defecto. Para el segundo modelo se requiere de una placa de menores dimensiones adicional, así como una señal indicativa para que se coja la tapa adecuada.

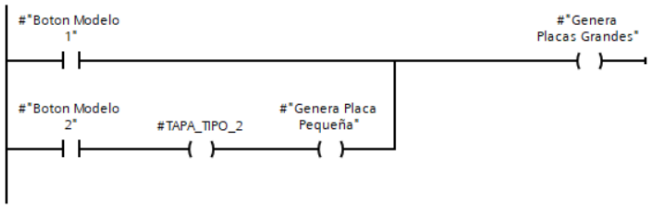


Figura 133. *Movimiento de placas de circuito y selección del modelo de tapa posterior.*

Capítulo 6.3. Desarrollo en MATLAB

Tal y como se había adelantado en capítulos previos, el desarrollo de una interfaz hombre-maquina (IHM) se va a llevar a cabo en el programa MATLAB, concretamente en el entorno de desarrollo de aplicaciones gráficas AppDesigner.

La dificultad en el desarrollo del código se ve reducida con el uso de esta herramienta, que genera de forma automática los fragmentos de código que definen los componentes utilizados de su librería. Este código autogenerado no se mostrará en las descripciones sobre la programación de este capítulo. Si el lector desea profundizar en dicho código puede acudir a los Anejos de la memoria en la carpeta MATLAB, donde encontrar los códigos de todas las ventanas programadas.

Antes de comenzar con la explicación detallada del código desarrollado, se ilustra un diagrama de bloques donde se muestra el funcionamiento de la interfaz.

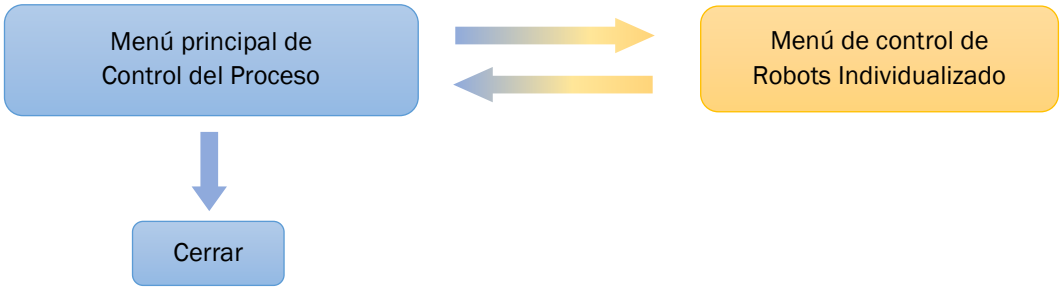


Figura 134. *Diagrama de bloques funcionamiento de la interfaz hombre-maquina.*

Capítulo 6.3.1. Menú principal de control del proceso

Esta pantalla de la interfaz es la principal y, por lo tanto, la que deben ejecutarse al inicio, para cargar los valores por defecto. En este menú se encuentran una serie de opciones con las que el usuario encargado de la gestión puede controlar las maquinas del proceso de forma manual, o establecer el modo de funcionamiento automático. Entre las opciones disponibles se tiene:

- Selector de modo de funcionamiento (manual, automático) con luz de indicación.
- Botones de gestión del soporte para el montaje a través de la instalación, control de avance y retroceso.
- Botones de control de elementos de simulación (paneles de cristal y LCD y placas de circuito), generar nueva pieza o restablecer.
- Slider y ventana de control de la velocidad de funcionamiento de los robots
- Botones de parada de emergencia y rearme de la estación.
- Botones de selección de pestañas para control de robots individualmente.

Adicionalmente, se cuenta con una serie de imágenes y textos que facilitan la usabilidad de la propia interfaz.

El siguiente diagrama de flujo muestra de forma resumida el manejo de la interfaz, indicado los grupos de botones y opciones con los que se cuenta.

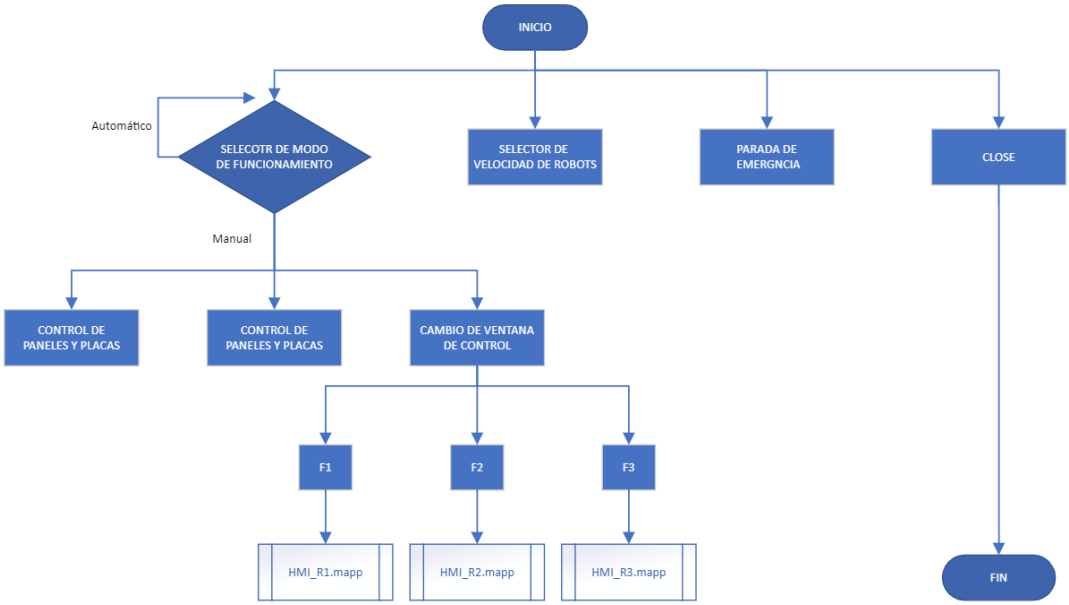


Figura 135. *Diagrama de flujo interfaz HMI_PROCESO.mapp.*

Una vez definido el funcionamiento del menú principal de la interfaz, se procede a detallar el proceso de programación exclusivamente, para observar el resultado gráfico de la interfaz se debe acudir al *CAPITULO 7. PRUEBAS Y RESULTADOS*.

Al ejecutar el código de la interfaz se deben definir una serie de propiedades de acceso público que puedan ser utilizadas por todas las funciones de la interfaz. Estas variables no van a ser definidas como tipos de datos, por lo tanto, al asignar por primera vez un valor a cada una de las variables se

```
properties (Access = public)
    Cliente_HMI_Proceso;        Proceso;
    Automatico;         Reset_Automatico;
    Avance;         Retroceso;
    Entrada_Cristal;          Entrada_LCD;
    TV_Modelo_1;         TV_Modelo_2;
    Velocidad;
    Emergencia;         Rearme;
    F0;         F1;         F2;         F3;
end
```

Para acceder desde una función a estas variables, se utiliza el siguiente comando:

‘app.NombreVariable.CaracterísticaVariable’

Function startupFcn(app)

Esta función se ejecuta automáticamente al inicializar la interfaz, se encarga de definir los valores por defecto del estado de los elementos presentes en la aplicación, así como la definición de características, generales o específicas, de las variables que se utilizan.

En el siguiente fragmento de código se observan los aspectos que se han comentado sobre la definición inicial de valores.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Función inicial de arranque de la aplicacion HMI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Activación de los botones pulsables al iniciar la interfaz
app.SELECTORDEMODOSwitch.Enable='On';
app.SELECTORDEMODOSwitch.Value='Manual';
app.MODOMANUALACTIVOLamp.Enable='On';
app.MODOMANUALACTIVOLamp.Color='Green';
app.MODOAUTOMATICOACTIVOLamp.Enable='Off';
app.MODOAUTOMATICOACTIVOLamp.Color='Red';
                            :
                            :
%Se establece conexión con el servidor OPC al iniciar la
%aplicacion
opcreset;
app.Cliente_HMI_Proceso = opcda('localhost','Kepware.KEPServerEX.V6')
connect(app.Cliente_HMI_Proceso)

%Se asigna cada varaibles global con su correspondiente
%variable definidda en el servidor
app.Proceso = addgroup(app.Cliente_HMI_Proceso);

Cinta_Manual = additem(app.Proceso,'SIEMENS.S7-1200.Cinta Manual Entrada');
write(Cinta_Manual,1);
                            :
                            :
app.Velocidad = additem(app.Proceso,'SIEMENS.S7-1200.Velocidad_HMI');
write(app.Velocidad,100);
```

Las siguientes funciones son las denominadas '*Callback*', estas funciones se ejecutan al interactuar con alguno de los componentes de la interfaz (botones, sliders, cuadros numéricos o de texto…), en función del tipo de elemento que se trate, la llamada a la función tendrá diferentes argumentos y definiciones de los valores de las variables que se van a tratar.

Dentro de cada función se pueden establecer variables locales, por lo que a lo largo de los siguientes puntos se observará como en diferentes '*Callback*s' existen variables con el mismo nombre.

CallBack: Cerrar Interfaz

Cuando el usuario desee cerrar por completo la interfaz, deberá recurrir al botón *Cerrar (Close)* situado en la esquina superior derecha de la aplicación (como todas las ventanas en el sistema operativo Windows).

En primer lugar, para proceder al cerrado de la aplicación, se debe eliminar la conexión con el servidor OPC, para ello se ejecuta el comando (*opcreset*) de la librería OPCToolbox encargado de reestablecer todos los valores referentes a cualquier conexión OPC que MATLAB esté estableciendo.

Una vez eliminados los valores de conexión, se procede a cerrar la aplicación.

```
% Close request function: UIFigure
function UIFigureCloseRequest(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Desconexion y Borrado de la comunicación OPC.
    % Limpieza de variables OPC
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Se elimina la conexión OPC y el grupo de variables
    opcreset

    % Se elimina la aplicación
    delete(app.UIFigure)
    delete(app)
```

CallBack: Selector de modo de control (manual. automático)

Para la selección del modo de control se dispone de un elemento con dos posibles posiciones, una para cada modo de funcionamiento contemplado en la instalación. El fragmento de código que se muestra a continuación muestra como se ha programado la secuencia de acción de este componente.

```
% Value changed function: SELECTORDEMODOSwitch
function SELECTORDEMODOSwitchValueChanged(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Selector de modo de funcionamiento de la instalacion
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    value = app.SELECTORDEMODOSwitch.Value;

    switch value
        case 'Manual'
            %Se establece el modo manual
            write(app.Automatico,0);
            write(app.Reset_Automatico,0);

            % Se habilitan los componentes del control manual
            app.MODOMANUALACTIVOLamp.Enable='On';
            app.MODOMANUALACTIVOLamp.Color='Green';
            app.MODOAUTOMATICOACTIVOLamp.Enable='Off';
            app.MODOAUTOMATICOACTIVOLamp.Color='Red';
                                      ⋮
            % Se reestablecen las condiciones iniciales
            write(app.Avance,0);
            write(app.Retroceso,0);
            write(app.Entrada_Cristal,0);
            write(app.Entrada_LCD,0);
            write(app.TV_Modelo_1,0);
            write(app.TV_Modelo_2,0);

        case 'Automatico'

            %Se establece el modo automatico
            write(app.Automatico,1);
            write(app.Reset_Automatico,1);

            % Se habilitan los componentes del modo automático
            app.MODOMANUALACTIVOLamp.Enable='Off';
            app.MODOMANUALACTIVOLamp.Color='Red';
            app.MODOAUTOMATICOACTIVOLamp.Enable='On';
            app.MODOAUTOMATICOACTIVOLamp.Color='Green';
                                      ⋮
            % Se reestablecen las condiciones iniciales
            write(app.Avance,0);
            write(app.Retroceso,0);
            write(app.Entrada_Cristal,0);
            write(app.Entrada_LCD,0);
            write(app.TV_Modelo_1,0);
            write(app.TV_Modelo_2,0);
    end
```

Se debe asegurar en cada cambio de modo la escritura de la variable 'Automatico' con el valor que le corresponda en cada caso, para que el autómata sea capaz de interpretar que bloque de función se está solicitando. Así mismo se ajusta la visibilidad de los grafismos de la aplicación, así como los botones que se deben encontrar habilitados en el caso de control manual o deshabilitados en el control automático, para que no exista posibilidad de incoherencia en el funcionamiento.

CallBacks: Control individual de robots (botones F1, F2 y F3)

Siempre y cuando se encuentre el control de la estación en modo manual, los botones para lanzar las aplicaciones de control individualizados de los robots se encontrarán activos. El siguiente fragmente de código muestra el procedimiento al pulsar el botón F1.

```
% Button pushed function: F1Button
function F1ButtonPushed(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Se cambia a la ventana de control de R1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    app.F0Button.Enable='On';
    app.F1Button.Enable='Off';
    app.F2Button.Enable='Off';
    app.F3Button.Enable='Off';

    write(app.Reset_Automatico,0);
    write(app.Automatico,0);

    HMI_R1;
```

El procedimiento en caso de los botones F2 y F3 será análogo, con la salvedad de la aplicación que se ejecuta que serán HMI_R2 y HMI_R3 respectivamente.

Adicionalmente se escribe el valor de la variable de reinicio de modo automático a 0 así como la propia variable de estado automático, por redundancia en el código, para evitar posibles errores.

El caso del botón F0 en la venta de control de proceso es utilizado únicamente cuando el control individualizado de los robots ha finalizado para reactivar el estado de los botones.

CallBacks: Botones de control del soporte móvil (avance y retroceso)

Se cuenta para la gestión en modo manual del movimiento del soporte sobre el que se van a ensamblar los televisores con dos botones con enclavamiento, que escriben en sus correspondientes variables el estado de los botones en cada cambio.

```
% Value changed function: AVANCEButton
function AVANCEButtonValueChanged(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Se da un pulso de avance al autómata
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    value = app.AVANCEButton.Value;
    if value==0
        app.RETROCESOButton.Enable='On';
    elseif value==1
        app.RETROCESOButton.Enable='Off';
    end
    write(app.Avance,value);
```

```
% Value changed function: RETROCESOButton
function RETROCESOButtonValueChanged(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Se da un pulso de retroceso al autómata
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    value = app.RETROCESOButton.Value;
    if value==0
        app.AVANCEButton.Enable='On';
    elseif value==1
        app.AVANCEButton.Enable='Off';
    end
    write(app.Retroceso,value);
```

Adicionalmente, para evitar que ambos variables (Avance y Retroceso) se encuentre a nivel alto simultáneamente, se desactiva el botón contrario cuando uno de ellos se encuentra pulsado.

CallBacks: Entrada de Cristal, Reset Cristal, Entrada de panel LCD y Reset LCD

Para la gestión de entrada al sistema de paneles de cristal y LCD se ha recurrido a utilizar en este caso de botones sin enclavamiento que se complementan con un botón de reinicio para cada botón de entrada.

De esta forma se logra un efecto visual diferencia respecto al movimiento del soporte móvil. A parte de la componente estética, la programación se ha tenido que realizar de esta forma por la naturaleza del proyecto, pues la simulación no funcionaría correctamente sin el botón de reseteo de los paneles hasta la posición de entrada.

Los siguientes fragmentos de código se muestran cómo se ha realizado la programación para los botones de entrada de paneles de cristal, así como el reseteo hasta la posición de partida. Para el caso de los botones relativos a los paneles LCD el procedimiento sería análogo, cambiando la variable que se escribe (véase *ANEXO C*. Ejemplos de código en la interfaz).

```
% Button pushed function: ENTRADACRISTALButton
function ENTRADACRISTALButtonPushed(app, event)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Se genera un cristal en la instalacion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

write(app.Entrada_Cristal,1);
app.ENTRADACRISTALButton.Enable='Off';
app.Reset_CristalButton.Enable='On';
```

```
% Button pushed function: Reset_CristalButton
function Reset_CristalButtonPushed(app, event)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reset de la funcion de generar un cristal en la instalación
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

write(app.Entrada_Cristal,0);
app.ENTRADACRISTALButton.Enable='On';
app.Reset_CristalButton.Enable='Off';
```

CallBacks: Gestión de placas de circuito y tapas de plástico.

El modelo de televisor que se va a ensamblar viene definido en función del número de placas de circuito que se van a instalar. Cuando se cuente con 2 placas de circuito, se tratará del modelo denominado 'Modelo 1' y cuando se monten 3 placas de circuito se tratará del 'Modelo 2'.

Para gestionar el número de placas de circuito que llegan a la instalación, así como la indicación al robot de manipulación de tapas de plástico que modelo debe seleccionar, cuando la estación está siendo controlada en modo manual, se han dispuesto 3 botones sin enclavamiento. Dos de estos botones están dedicados a la selección del modelo de televisor que se va a ensamblar y por lo tanto el número de placas que se deben generar y el tercero actúa como reseteo a la posición inicial de las placas.

En los siguientes fragmentos de código se muestra como se ha llevado a cabo la programación, recurriendo a dos variables para la selección del modelo deseado.

```
% Button pushed function: TVMODELO1Button
function TVMODELO1ButtonPushed(app, event)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Se genera placas para TV modelo 1 en la instalación
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

write(app.TV_Modelo_1,1);
write(app.TV_Modelo_2,0);
app.TVMODELO1Button.Enable='Off';
app.TVMODELO2Button.Enable='On';
app.Reset_TVButton.Enable='On';
```

```
% Button pushed function: TVMODELO2Button
function TVMODELO2ButtonPushed(app, event)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Se genera placas para TV modelo 2 en la instalación
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

write(app.TV_Modelo_1,0);
write(app.TV_Modelo_2,1);
app.TVMODELO1Button.Enable='On';
app.TVMODELO2Button.Enable='Off';
app.Reset_TVButton.Enable='On';
```

A continuación, se puede observar como el botón de reseteo, como en anteriores ocasiones, se encarga de devolver al sistema al estado por defecto.

```
% Button pushed function: Reset_TVButton
function Reset_TVButtonPushed(app, event)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reset de la funcion de generar un cristal en la instalacion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

write(app.TV_Modelo_1,0);
write(app.TV_Modelo_2,0);
app.TVMODELO1Button.Enable='On';
app.TVMODELO2Button.Enable='On';
app.Reset_TVButton.Enable='Off';
```

CallBacks: Control de la velocidad de trabajo de los robots

Dentro de la ventana de control del proceso, se cuenta con un slider y un campo numérico editable que permiten seleccionar la velocidad de trabajo de los robots, tanto en modo de funcionamiento manual como automático.

Estos campos están enlazados de tal forma que el valor en ambos debe ser siempre el mismo y estar comprendido siempre entre los valores 0 y 100, refiriéndose al porcentaje que se aplica a la velocidad programada. Es decir, indicar desde el HMI que se desea una velocidad del 100% no hará que cada movimiento se realice a la máxima velocidad permitida por el robot, sino a la máxima programada. De esta forma se asegura que un movimiento que por seguridad se ha limitado a una cierta velocidad, esta nunca pueda ser excedida, pero si pueda reducirse para observar un movimiento concreto, o una trayectoria, con mayor detenimiento.

```
% Value changed function: VELOCIDADSlider
function VELOCIDADSliderValueChanged(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Se cambia de forma conjunta la velocidad de todos los robots
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    value = int16(app.VELOCIDADSlider.Value);
    app.VELOCIDADField.Value=app.VELOCIDADSlider.Value;

    write(app.Velocidad,value);
```

```
% Value changed function: VELOCIDADField
function VELOCIDADFieldValueChanged(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Se cambia la velocidad de los robots
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    value=int16(app.VELOCIDADField.Value);
    % Se cambia el valor del slider y posteriormente se escribe en
    % las variables del servidor el valor
    app.VELOCIDADSlider.Value=app.VELOCIDADField;
    % Se mantienen los limites del slider
    if value>100
        value=100;
    elseif value<0
        value=0;
    end
    write(app.Velocidad,value);
```

CallBacks: Parada de emergencia y rearme

Para finalizar con la ventana de control, se ha incluido un botón que permite avisar al autómata, desde la propia interfaz, de un funcionamiento anómalo, o el deseo de abortar el funcionamiento en modo automático, con el fin de cumplir con las medidas de seguridad requeridas por la instalación.

Una vez corregida la anomalía se podrá reactivar la estación a través del botón de rearme, siempre en modo manual.

```
% Button pushed function: Seta_Emergencia
function Seta_EmergenciaButtonPushed(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Se establece el modo de emergencia en la instalacion
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    write(app.Automatico,0);
    write(app.Emergencia,1);
    app.VELOCIDADSlider.Value=0;

    % Se desactivan los botones de control
    app.SELECTORDEMODOSwitch.Enable='Off';
    app.ENTRADACRISTALButton.Enable='Off';
    app.ENTRADALCDButton.Enable='Off';
    app.TVMODELO1Button.Enable='Off';
    app.TVMODELO2Button.Enable='Off';
    app.Reset_CristalButton.Enable='Off';
    app.Reset_LCDButton.Enable='Off';
    app.Reset_TVButton.Enable='Off';
    app.Seta_Emergencia.Enable='Off';
    app.MODOMANUALACTIVOLamp.Enable='Of';
    app.MODOMANUALACTIVOLamp.Color='Red';
    app.MODOAUTOMATICOACTIVOLamp.Enable='Off';
    app.MODOAUTOMATICOACTIVOLamp.Color='Red';

    %Se activa el boton de rearme
    app.REARMEDELAESTACINButton.Enable='On';
```

```
% Button pushed function: REARMEDELAESTACINButton
function REARMEDELAESTACIONButtonPushed(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Se elimina el estado de emergencia de la instalacion
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    write(app.Reset_Automatico,0);
    write(app.Emergencia,1);

    % Se activan los botones de control
    app.SELECTORDEMODOSwitch.Enable='Off';
    app.ENTRADACRISTALButton.Enable='On';
    app.ENTRADALCDButton.Enable='On';
    app.TVMODELO1Button.Enable='On';
    app.TVMODELO2Button.Enable='On';
    app.Reset_CristalButton.Enable='On';
    app.Reset_LCDButton.Enable='On';
    app.Reset_TVButton.Enable='On';
    app.Seta_Emergencia.Enable='On';
    app.MODOMANUALACTIVOLamp.Enable='On';
    app.MODOMANUALACTIVOLamp.Color='Green';
    app.MODOAUTOMATICOACTIVOLamp.Enable='Off';
    app.MODOAUTOMATICOACTIVOLamp.Color='Red';

    %Se desactiva el boton de rearme
    app.REARMEDELAESTACINButton.Enable='Off';

    %Se reactiva la instalacion en modo manual
    app.SELECTORDEMODOSwitch.Value='Manual';
```

Capítulo 6.3.2. Menú control de robots individualizado

Para poder controlar cada uno de los robots de forma individualizada, se han desarrollado 3 aplicaciones adicionales. Desde el punto de vista funcional se tratan de ventanas secundarias, pero han sido programadas como aplicaciones independientes para evitar escrituras incorrectas de variables que pudieran causar un mal funcionamiento, por lo que cada aplicación supone un nuevo cliente conectado al servidor.

Para facilitar la interacción del usuario, no se permite tener abiertas simultáneamente dos ventanas de control de robots simultáneamente, por lo tanto, cuando se cambia desde la ventana de control de uno de los robots a la ventana de control de otro de ellos, la ventana original desde la que se comanda el cambio es cerrada y eliminado su cliente.

Desde el punto de vista funcional, todas las ventanas cuentan con los mismos componentes, entre los que se encuentran:

- Botones de gestión del estado del robot, incluyendo una parada de emergencia (STOP) y un botón dedicado a eliminar fallos del robot.
- Botones de cambio entre pestañas de control de robots y botón de regreso a la ventana principal.
- Botón de retorno a la posición HOME
- Ilustración referente al robot que está siendo controlado.

Se ilustra a continuación un diagrama de flujo explicativo de las funcionalidades que se encuentran disponibles en esta ventana de control individualizado de los robots.

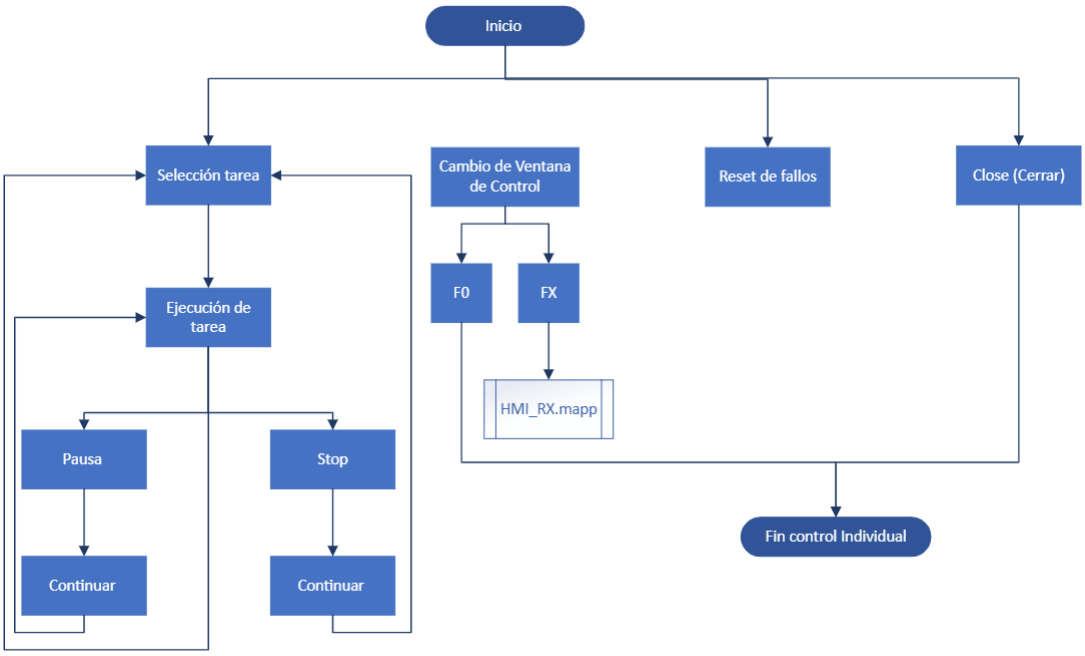


Figura 136. *Diagrama de flujo de control de robot individualmente.*

Respecto al desarrollo del código, se va a mostrar cómo se ha programado la interfaz para el control del primer robot (R-2000iC/125L). Para profundizar en el código, el lector puede acudir a los anejos a la memoria, donde encontrará el código de las ventanas de control de los dos robots restantes, las cuales no han sido añadidas a la memoria, por ser esencialmente idénticas a la ventana que se va a detallar, con las variables que se encargan de la gestión de cada robot.

La aplicación de control de robots inicia, al igual que en el caso anterior, con la definición de una serie de variables globales para la interfaz seguida de una función de arranque encargada de la inicialización de la interfaz, donde se establece tanto el estado de los componentes de la interfaz, como el valor inicial de las variables.

CallBacks: Tareas a realizar por el robot

Cada uno de los robots contará con más o menos botones en función de las tareas que se le hayan asignado, estos botones son botones con enclavamiento que funcionan de forma que cuando se encuentran activos, la variable de la acción correspondiente se escribe a nivel alto para que el autómata pueda interpretar que acción realizar. Adicionalmente, se desactivan el resto de los botones de selección de tarea cuando uno de ellos se encuentra activo.

Independientemente del número de tareas que lleve a cabo el robot, todos ellos cuentan con la opción de movimiento a la posición asignada como HOME en cada uno de ellos.

```
% Value changed function: MOVIMIENTOAHOMEButton
function MOVIMIENTOAHOMEButtonValueChanged(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Movimiento del robot a posicion HOME
    % Se deshabilitan los botones del resto de tareas
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    value = app.MOVIMIENTOAHOMEButton.Value;
    write(app.Cristal,0);
    write(app.LCD,0);
    write(app.Chapa,0);
    write(app.HOME,value);

    switch value
        case 0
            app.MANIPULACHAPAButton.Enable='On';
            app.MANIPULACRISTALButton.Enable='On';
            app.MANIPULALCDButton.Enable='On';

        case 1
            app.MANIPULACHAPAButton.Enable='Off';
            app.MANIPULACRISTALButton.Enable='Off';
            app.MANIPULALCDButton.Enable='Off';
            write(app.START,1);
            pause(0.5);
            write(app.START,0);
    end
```

Como se puede observar en el anterior fragmento de código, cuando se desea realizar alguna función, además de establecer el valor de la tarea (en este caso HOME) a nivel alto, se debe dar un pulso (superior a 250ms según el manual del fabricante) en la variable de inicio del programa seleccionado en el robot (en este caso el indicado para el control manual del robot).

CallBack: Pausa el movimiento del robot

Si durante la ejecución de alguna trayectoria, el usuario desea pausar el movimiento, de tal forma que este pueda ser retomado tras la parada, se cuenta en esta venta con un botón simple, sin enclavamiento, dedicado a esta función, el cual establece a nivel alto la variable de entrada del autómata que indica que este comandar la parada al robot.

```
% Button pushed function: PAUSAButton
function PAUSAButtonPushed(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Se establece una pausa controlada en el robot
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    app.CONTINUAButton.Enable='On';
    app.PAUSAButton.Enable='Off';
    write(app.PAUSA,1);
```

CallBack: Parada total de la ejecución del robot

En el caso de que el usuario desee abortar completamente el movimiento del robot, sin importar que el movimiento no se pueda retomar tras la parada, pues la situación en que se encuentra es crítica, se ha dotado a la aplicación de un botón encargado de esta función.

```
% Button pushed function: STOPButton
function STOPButtonPushed(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Se realiza una parada abrupta abortando en programa MAIN
    % No se retomara el movimiento desde el punto de parada
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    app.CONTINUAButton.Enable='On';

    app.MOVIMIENTOAHOMEButton.Enable='Off';
    app.MANIPULACHAPAButton.Enable='Off';
    app.MANIPULACRISTALButton.Enable='Off';
    app.MANIPULALCDButton.Enable='Off';

    app.STOPButton.Enable='Off';
    app.PAUSAButton.Enable='Off';

    write(app.START,0);
    write(app.PAUSA,1);
    write(app.PAUSA,0);

    write(app.STOP,1);
    pause(0.5);
    write(app.STOP,0);
```

Se deben dar una serie de condiciones para poder abortar la ejecución. En primer lugar, se debe establecer a nivel bajo la variable de arranque, seguidamente se instaura el estado de pausa para detener el movimiento y a continuación, con un pulso de al menos 250ms se aborta la ejecución con la variable asignada a esta función. Adicionalmente se desactivan todas las funcionalidades de la ventana salvo la de continuar con el control y eliminar fallos del robot.

CallBack: Reanuda el control

El botón 'CONTINUAR' contempla dos opciones en función del estado del robot. En caso de pulsar este botón después de una pausa controlada, simplemente se reinicia la ejecución del programa y por tanto el movimiento, para ello se establece a nivel bajo la variable de pausa y se da un pulso en la variable '*RESUME*'.

Cuando el estado precedente se tratase de una interrupción abrupta del programa, se deben eliminar posibles fallos y a continuación restablecer los botones de selección de tareas.

```
% Button pushed function: CONTINUAButton
function CONTINUAButtonPushed(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Se reanuda el movimiento desde la parada
    % Se continua con la trayectoria previa a la parada
    % si esta ha sido pausada o se restablecen los valores
    % tras abortar el programa MAIN
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Se leen el estado de las variables de STOP y PAUSA
    valor_PAUSA=read(app.PAUSA);

    if valor_PAUSA.Value==1
        app.PAUSAButton.Enable='On';
        write(app.PAUSA,0);
        write(app.RESUME,1);
        pause(0.5);
        write(app.RESUME,0);
    else
        app.MOVIMIENTOAHOMEButton.Enable='On';
        app.MANIPULACHAPAButton.Enable='On';
        app.MANIPULACRISTALButton.Enable='On';
        app.MANIPULALCDButton.Enable='On';
        app.STOPButton.Enable='On';
        app.PAUSAButton.Enable='On';
    end
```

CallBack: Elimina fallos de robot

Esta funcionalidad simple, diseñada con un botón sin enclavamiento, se encarga de eliminar fallos del robot que impidan la ejecución o continuación de un programa, como singularidades o errores de programación que han de ser corregidos previamente, para ello se da un pulso de 100ms en la variable asignada.

```
% Button pushed function: RESETFALLOSButton
function RESETFALLOSButtonPushed(app, event)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Se eliminan los fallos del robot
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    write(app.RESET,1);
    pause(0.1);
    write(app.RESET,0);
```

Capítulo 6.4. Comunicación entre instancias

Una vez definidas las funcionalidades de los robots, así como la programación del autómata para la gestión de la instalación y la planificación de las aplicaciones para la gestión de las funcionalidades de control, es necesario definir la configuración específica del software encargado de mapear las señales para la comunicación entre las diferentes instancias.

El programa KEPServerEX, cuya configuración y usabilidad fue explicada en el *Capítulo 4.5.* KEPServerEX 6*,* es el encargado de mapear las señales que provienen del autómata, a variables cuya referencia pueda ser usada por los clientes en Roboguide y MATLAB.

El primer paso para poder realizar el mapeado de las señales del autómata es definir un nuevo canal de conectividad, para ello se ha elegido el driver de *Siemens TCP/IP Ethernet* proporcionado por el software. Como adaptador de red se selecciona el generado por PLCSIM, con su dirección IP asignada en el equipo.

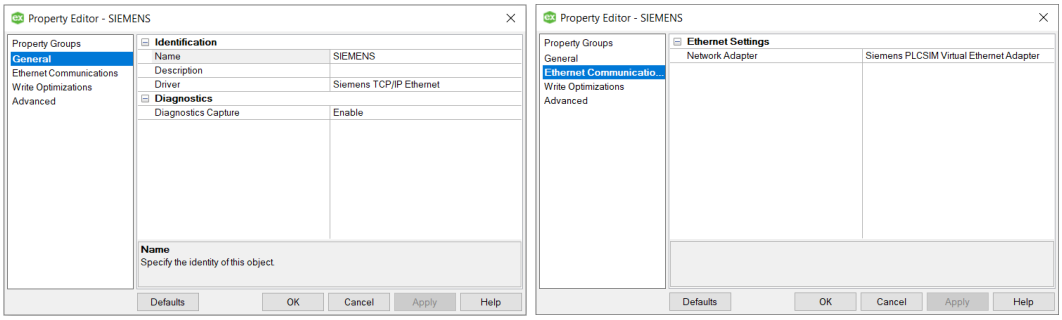


Figura 137. *Configuración de comunicaciones para Siemens TCP/IP Ethernet.*

Una vez definido el canal, se establece la configuración del dispositivo que se va a utilizar. Los parámetros más importantes en este apartado son la dirección

IP del dispositivo y el puerto de comunicación. La dirección IP se establece la misma asignada al simulador PLCSIM (192.168.1.196) y como puerto de comunicación se utiliza el 102, puerto del que se apodera la programa NETtoPLCSim, para establecer un puente directo entre el autómata y el servidor. Entre otras configuraciones, al utilizar para el proyecto un autómata de la gama 1200 de Siemens, el rack de la CPU será el Rack 0 en el Slot 1.

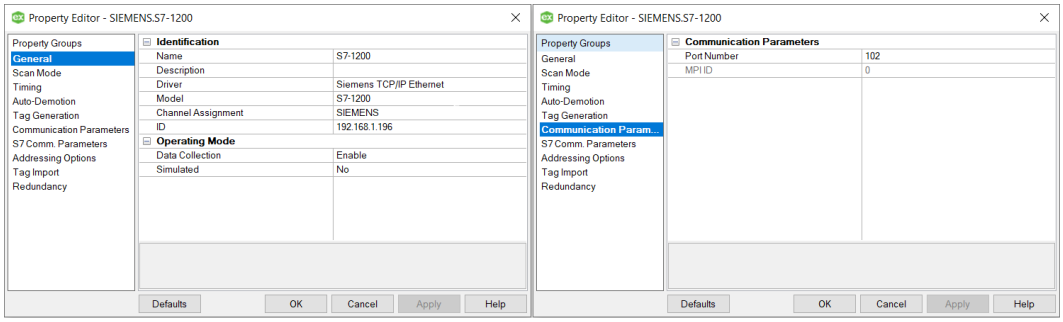


Figura 138. *Configuración PLC de la gama 1200.*

El uso de la dirección IP del adaptador de red del simulador de los autómatas y no la asignada al propio PLC se debe al uso de la aplicación NETtoPLCSIM. A través de esta aplicación se establece un puente de acceso directo con el autómata, a través del puerto 102 del cual se apodera el software al ejecutarse, que da acceso al servidor a sus variables.

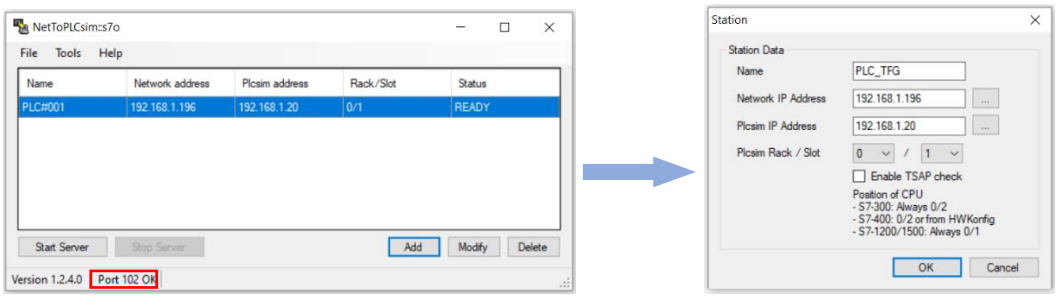


Figura 139. *Configuración NETtoPLCSIM.*

Una vez definida la configuración del servidor OPC, así como el puente de conexión con el autómata, se deben copiar, en el dispositivo creado en la aplicación del servidor, las variables que se van a ponerse a disposición de los clientes.

El nombre que se le dé a cada variable es el que usuran los clientes para comprobar su valor y/o modificarlo, mientras que la conexión con el autómata se realiza en función de la dirección asignada. Cabe destacar que el tipo de dato debe coincidir en el servidor y en el autómata para que no se den errores de escritura.

Tag Name	Address	Data Type	Scan Rate	Scaling	Description
F0	M0.0	Boolean	100	None	
F1	M0.1	Boolean	100	None	
F2	M0.2	Boolean	100	None	
F3	M0.3	Boolean	100	None	
Automatico	M0.4	Boolean	100	None	
Reset Automati...	M0.5	Boolean	100	None	
Emergencia	M0.6	Boolean	100	None	
Rearme	M0.7	Boolean	100	None	
HOME_R1	M1.0	Boolean	100	None	
FIN_R1	M1.1	Boolean	100	None	
Boton_HOME_...	M1.2	Boolean	100	None	
Boton_Cristal_...	M1.3	Boolean	100	None	
Boton_LCD_Ma...	M1.4	Boolean	100	None	
Boton_Chapa_...	M1.5	Boolean	100	None	
Fin Vision	M1.7	Boolean	100	None	
HOME_R2	M2.0	Boolean	100	None	
FIN_R2	M2.1	Boolean	100	None	
Boton_HOME_...	M2.2	Boolean	100	None	
Boton Vision	M2.3	Boolean	100	None	
Boton Atornillado	M2.4	Boolean	100	None	
HOME_R3	M3.0	Boolean	100	None	
FIN_R3	M3.1	Boolean	100	None	
Boton_HOME_...	M3.2	Boolean	100	None	
Boton Tapa 1	M3.3	Boolean	100	None	
Boton Tapa 2	M3.4	Boolean	100	None	
Cinta_POS_INI	M4.0	Boolean	100	None	
Cinta_POS_INT...	M4.1	Boolean	100	None	
Cinta_POS_FIN	M4.2	Boolean	100	None	
Placas Grande...	M4.3	Boolean	100	None	
Placa Pequeña...	M4.4	Boolean	100	None	
Cristal en Posici...	M4.5	Boolean	100	None	
LCD en Posicion	M4.6	Boolean	100	None	
Cinta Manual E...	M4.7	Boolean	100	None	
Cinta Manual INI	M5.0	Boolean	100	None	
Cinta Manual IN...	M5.1	Boolean	100	None	
Cinta Manual FIN	M5.2	Boolean	100	None	
INICIO_HMI	M6.0	Boolean	100	None	
PAUSA_HMI	M6.1	Boolean	100	None	
CONTINUA_HMI	M6.2	Boolean	100	None	
STOP_HMI	M6.3	Boolean	100	None	
RESET_HMI	M6.4	Boolean	100	None	
Avance Cinta	M6.5	Boolean	100	None	
Retroceso Cinta	M6.6	Boolean	100	None	
Avance Cristal	M7.0	Boolean	100	None	
Avance LCD	M7.1	Boolean	100	None	
TV Modelo 1	M7.2	Boolean	100	None	
TV Modelo 2	M7.3	Boolean	100	None	
Velocidad_HMI	MW100	Word	100	None	
Velocidad R1	MW110	Word	100	None	
Contador Ciclos...	MW112	Word	100	None	
Contador Chap...	MW114	Word	100	None	
Velocidad R2	MW120	Word	100	None	
Contador Ciclos...	MW122	Word	100	None	
Velocidad R3	MW130	Word	100	None	
Contador Ciclos...	MW132	Word	100	None	

Figura 140. *Variables definidas en el servidor.*

CAPITULO 7. PRUEBAS Y RESULTADOS

Con el desarrollo que se ha llevado a cabo en el desarrollo del proyecto explicado y detallado en el capítulo anterior, se procede a continuación a mostrar los resultados finales del trabajo que se ha realizado.

En las siguientes subdivisiones del presente capítulo, se mostrarán una serie de capturas representativas que se han podido obtener durante la ejecución de la simulación en los diferentes programas utilizados.

Cabe destacar que los programas se muestran secuencialmente, según el orden de lanzamiento de los mismo, pero realmente la ejecución debe ser paralela para que se produzca la coordinación de señales y se sincronicen las simulaciones. Cualquier secuencia de ejecución que difiera de la que se va a explicar enumerar a continuación no asegura el correcto funcionamiento conjunto de las simulaciones.

1. Servidor OPC

El primer paso que se debe llevar a cabo cuando se quiera ejecutar la simulación debe ser lanzar o reiniciar el servidor OPC encargado al que se conectarán todas las aplicaciones cliente.

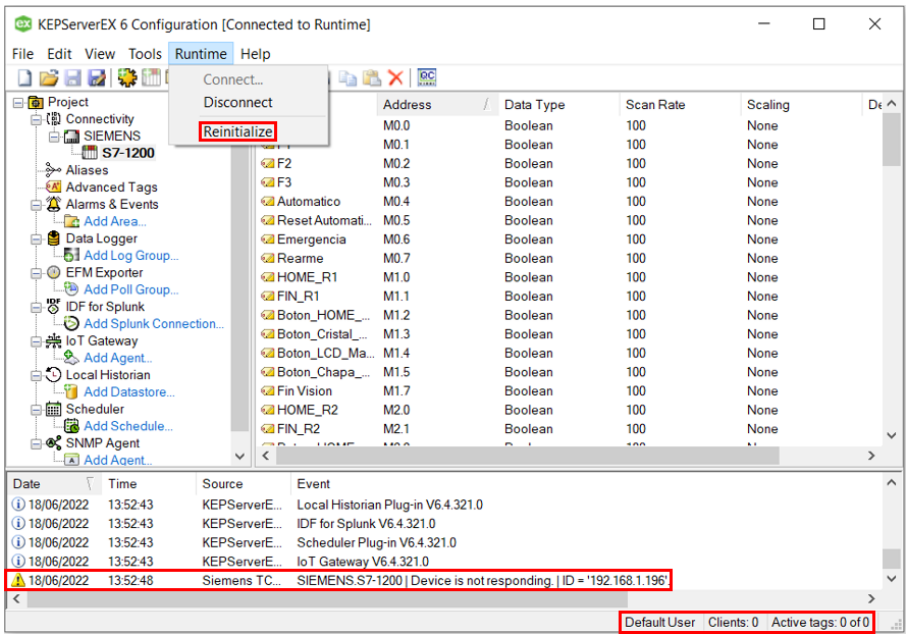


Figura 141. *Reinicio del servidor OPC.*

Una vez reiniciado, al no encontrarse en ejecución el simulador del autómata, el usuario se encontrará con todas las etiquetas inactivas, así como un mensaje de advertencia indicando que el dispositivo encargado de la gestión de las variables no se encuentra activo y no se ha podido establecer la conexión.

El servidor se mantendrá en una búsqueda constante del dispositivo en la dirección y puerto configurados hasta que consiga establecer conexión. Por lo tanto, el usuario no tiene por qué realizar ninguna acción más sobre este programa, más que comprobar que el estado del servidor cambie al lanzar el simulador del autómata.

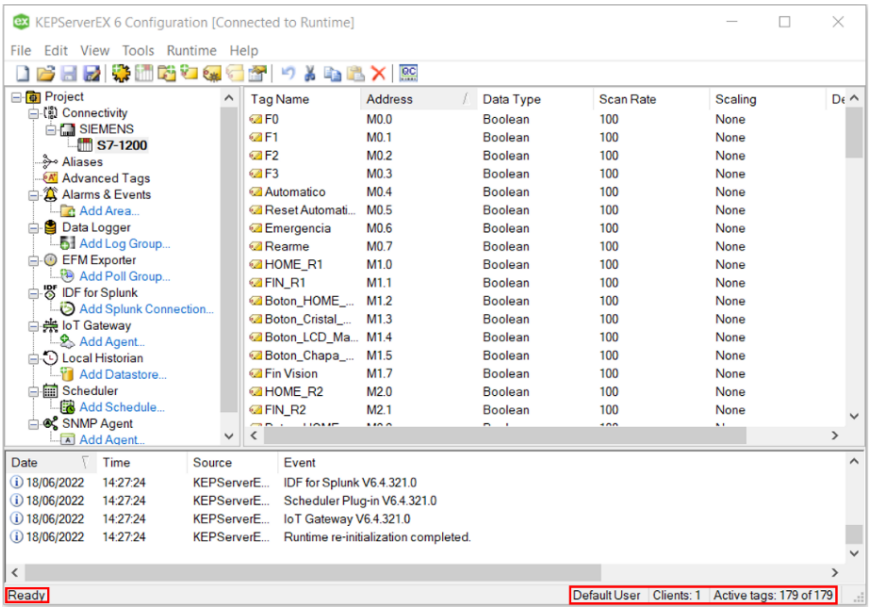


Figura 142. *Estado del servidor con el autómata conectado.*

Adicionalmente, lanzando la funcionalidad ***Quick Client*** del propio software se pueden observar el estado de todas las variables definidas en el servidor.

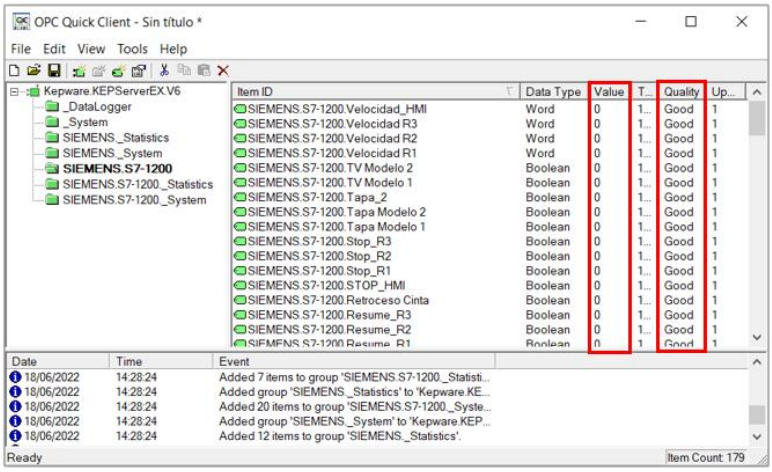


Figura 143. *Funcionalidad 'Quick Client'.*

Si el apartado *Quality* muestra como estado *Good,* así como el valor correcto en el apartado *Value* la conexión habrá sido exitosa.

Debe tenerse en cuenta, que al tratarse con una versión para estudiantes del software que mantiene el servidor OPC conectado, este se mantiene activo un

máximo de dos horas seguidas, desconectándose automáticamente una vez concluye el tiempo.

2. Simulación del autómata

Una vez se haya inicializado el servidor OPC, se procede a lanzar la simulación del autómata y cargar el programa diseñado. A su vez se debe realizar en la aplicación NETtoPLCSIM la configuración vista en la *Figura 139* y comenzar la comunicación (*StartServer*) para que se establezca la conexión del dispositivo con el servidor.

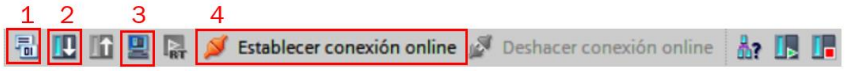


Figura 144. *Herramientas de simulación, conexión y control del autómata.*

El primer paso es compilar la programación realizada [1], con el objetivo de localizar posibles errores o advertencias. En caso de que la compilación no arroje ningún error, se lanza el simulador PLCSim, esto puede hacerse directamente desde la barra de tareas de TIA Portal [3], o abriendo la aplicación de escritorio. Una vez iniciado el simulador, se cargan los bloques de programa anteriormente compilados en el simulador [2].

Una vez lanzado el simulador y cargados los bloques de programa, se puede establecer conexión con el autómata [4] para observar el estado de las variables y los segmentos programados dentro de cada bloque.

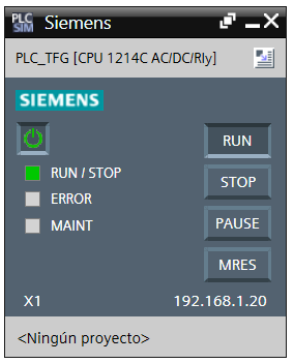


Figura 145. *Simulador del autómata.*

No resulta muy representativo ni aclarativo añadir capturas de instantes de funcionamiento del autómata, pues lo único que el lector va a poder observar es si una variable se encuentra o no activa permitiendo el paso de la corriente, por lo que no se agregarán este tipo de ilustraciones al informe. Sin embargo, esta funcionalidad si es una gran ayuda durante la programación, pues permite controlar en tiempo real lo que está sucediendo en el segmente observado y así comprender donde puede haber un fallo de programación de una forma muy visual.

3. Interfaz hombre-máquina

Una vez el autómata se encuentre conectado al servidor, se pueden ir añadiendo los diferentes clientes. A partir de este punto el orden de lanzamiento de los programas se puede invertir, pero se comenta a continuación el orden que se considera más conveniente.

Con el fin de ajustar algunos valores antes de comenzar con la simulación en Roboguide, se lanza en primer lugar la interfaz de gestión del proceso '*HMI_Proceso.mlapp*'. Esta aplicación cuenta con los botones y funcionalidades necesarias para controlar la simulación y muestra el siguiente aspecto una vez lanzada:

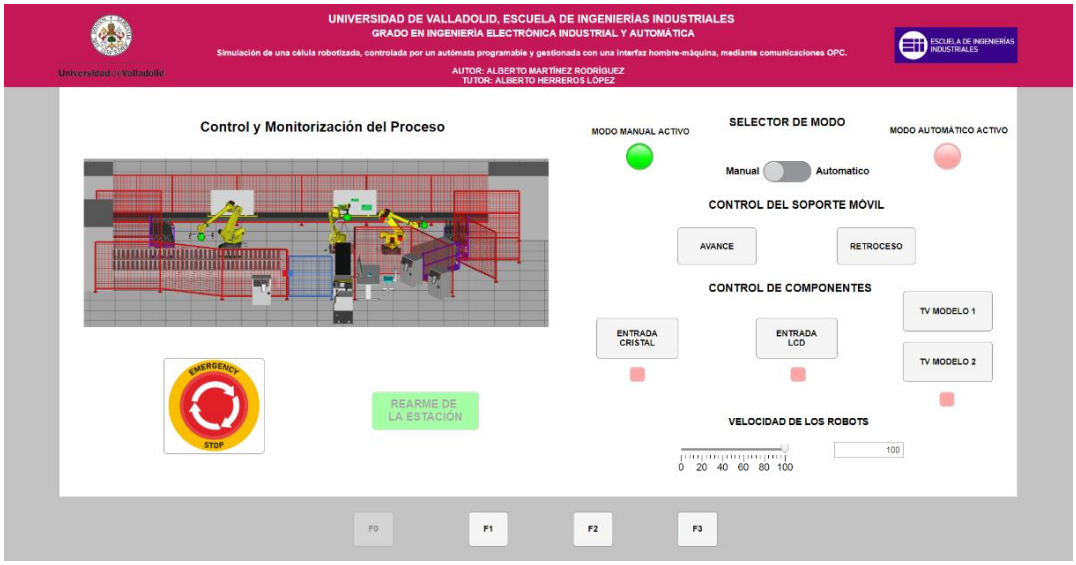


Figura 146. *Menú de gestión del proceso de la interfaz diseñada.*

Este menú principal cuenta con un selector principal de modo de funcionamiento de la instalación, el cual distingue entre dos modos: manual y automático. Como se explicó en el apartado de programación, en función del modo de funcionamiento, se activan o desactivan el resto de las funcionalidades y elementos gráficos como las lámparas.

Concretamente en la figura anterior se observan las funciones activas en el modo manual que se activa por defecto al iniciar la aplicación. Sin embargo, para el modo automático, el usuario de la aplicación se encontrará con que solo se mantienen activos, al cambiar a este modo, los botones de emergencia y el selector de velocidad de los robots.

Respecto a los botones de control de componentes, se observan una serie de botones de menor tamaño de color rojo, que se activan para el restablecimiento de la posición original de cada uno de estos elementos.

Con los botones inferiores de función (F0,F1,F2,F3) el usuario puede lanzar las ventanas secundarias de control individual de cada robot. Por ejemplo, se ilustra a continuación la aplicación emergente al seleccionar el control del robot R-2000iC/125L (R1) con el botón F1. El resto de las aplicaciones de control individual son íntegramente idénticas, por lo que no se mostrarán en el presente informe.

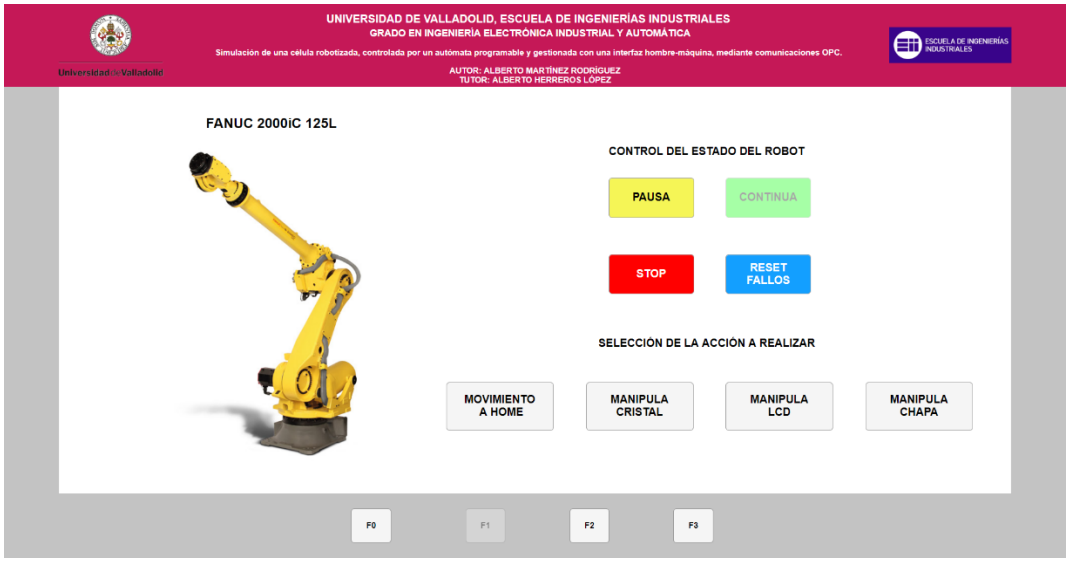


Figura 147. *Ventana de control individual de R1.*

Como se puede observar, esta ventana, pese a ser más simple que la principal, contiene todos los botones para el control del robot, así como la realización de las principales tareas que se pueden desempeñar en función del robot al que esté dirigida la aplicación.

Igual que con la ventana principal, en la parte inferior se encuentran los botones de función, desde donde el usuario puede seleccionar a que ventana desplazarse. No se contempla el control simultaneo de dos robots, pues se considera que el modo manual se utilizará solo para mantenimiento, comprobación de reprogramaciones o labores de índole similar, por lo tanto, la selección de una ventana de control de otro robot cierra la aplicación actual, desconectando su cliente del servidor y finalizando el movimiento del robot.

Para volver desde una ventana de control individual de alguno de los robots se debe pulsar el botón de función F0, el cual cierra la aplicación secundaria, desconectando el cliente y volviendo a mostrar la aplicación principal. Dentro de la aplicación principal se debe pulsar de nuevo el botón F0 como doble factor de verificación para que se activen de nuevo los botones de control manual.

5. Simulación del proceso en Roboguide

Se muestra, para concluir con los resultados, diferentes instantes del proceso de ensamblado en la instalación diseñada en el software de simulación Roboguide. Concretamente se ilustra el proceso de montaje para el modelo 2 de televisor, compuesto por tres placas de circuito, en modo de funcionamiento automático.

Como es difícil mostrar el resultado sin apoyarse en un video, para la elaboración de las siguientes ilustraciones se han tomado instantáneas sobre el funcionamiento de la simulación.

En primer lugar, se muestra en la siguiente figura, representado mediante flechas de diferentes colores, la dirección de llegada de los paneles (azul) en los conveyor de entrada, así como el curso de entrada de las placas de circuito (verde) y el sentido del movimiento del soporte móvil a través de la estación (rojo).

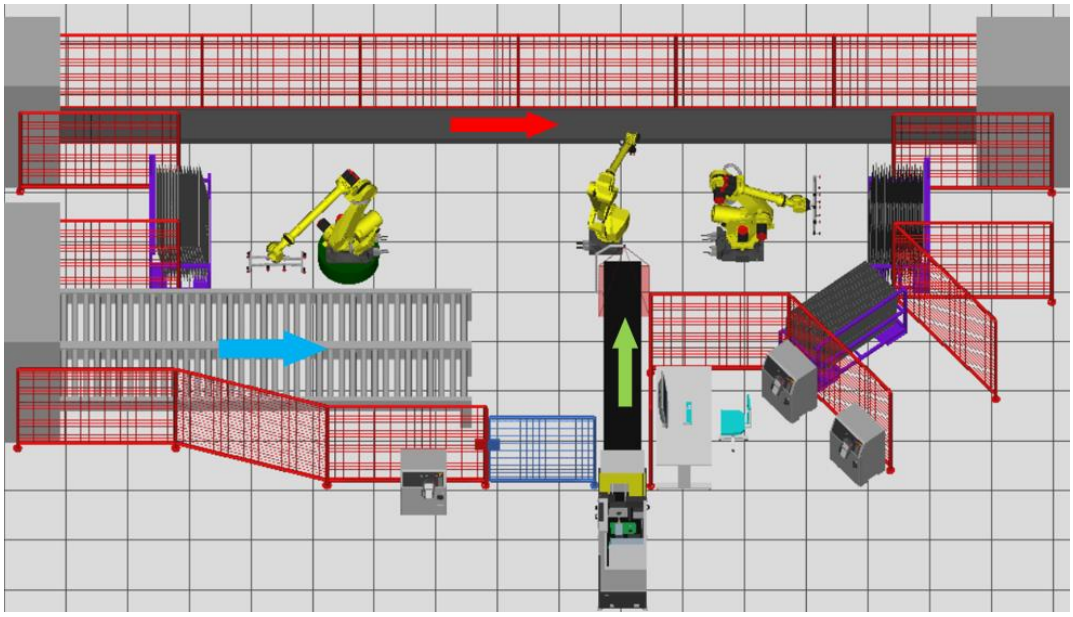


Figura 148. *Detalle de llegada y movimiento de piezas a través de la estación.*

En las siguientes ilustraciones se acota el campo de representación haciendo énfasis en el primero de los robots (R1) R-2000iC/125L y en las tareas que este robot lleva a cabo: manipulación de paneles de cristal y LCD, así como de chapas de aluminio para el montaje de las mismas sobre el marco del televisor a ensamblar.

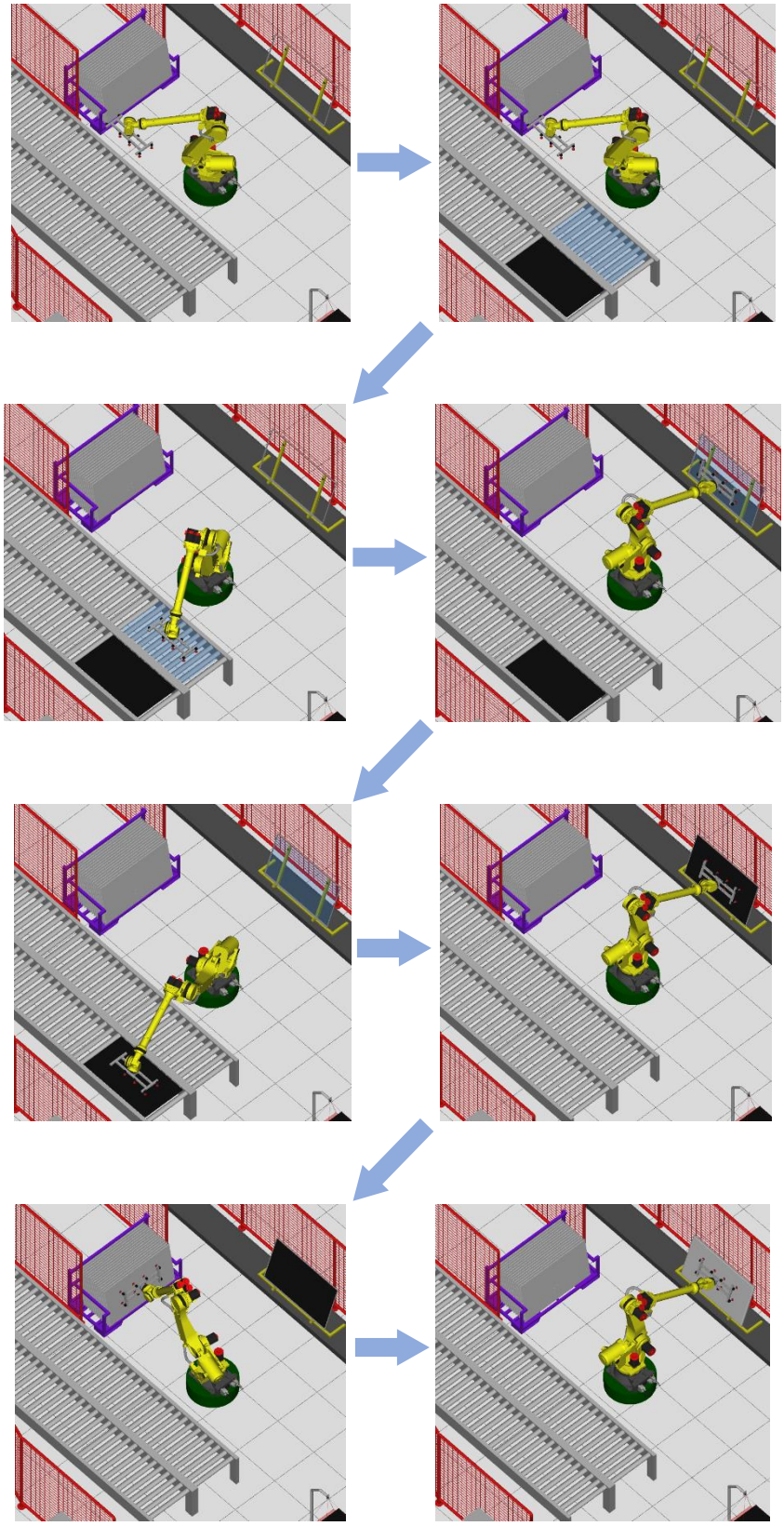


Figura 149. Maniobra de montaje de R1.

Una vez finalizada la maniobra del robot R1, el soporte móvil se desplaza hasta la posición de actuación de los robots R2 y R3 siguiendo la dirección vista en

la *Figura 148*. En la siguiente secuencia de figuras se muestra, por lo tanto, la actuación conjunta de estos dos robots en la labor de montaje.

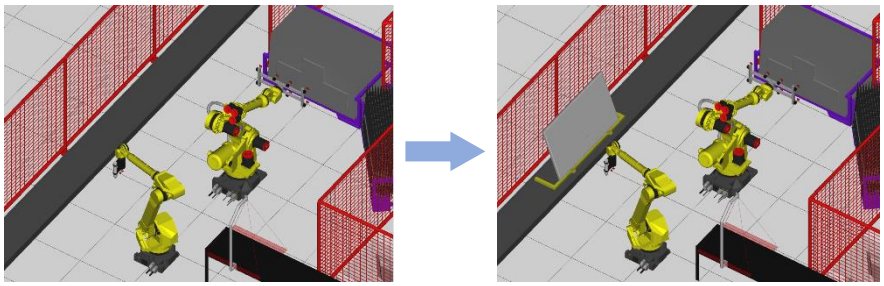
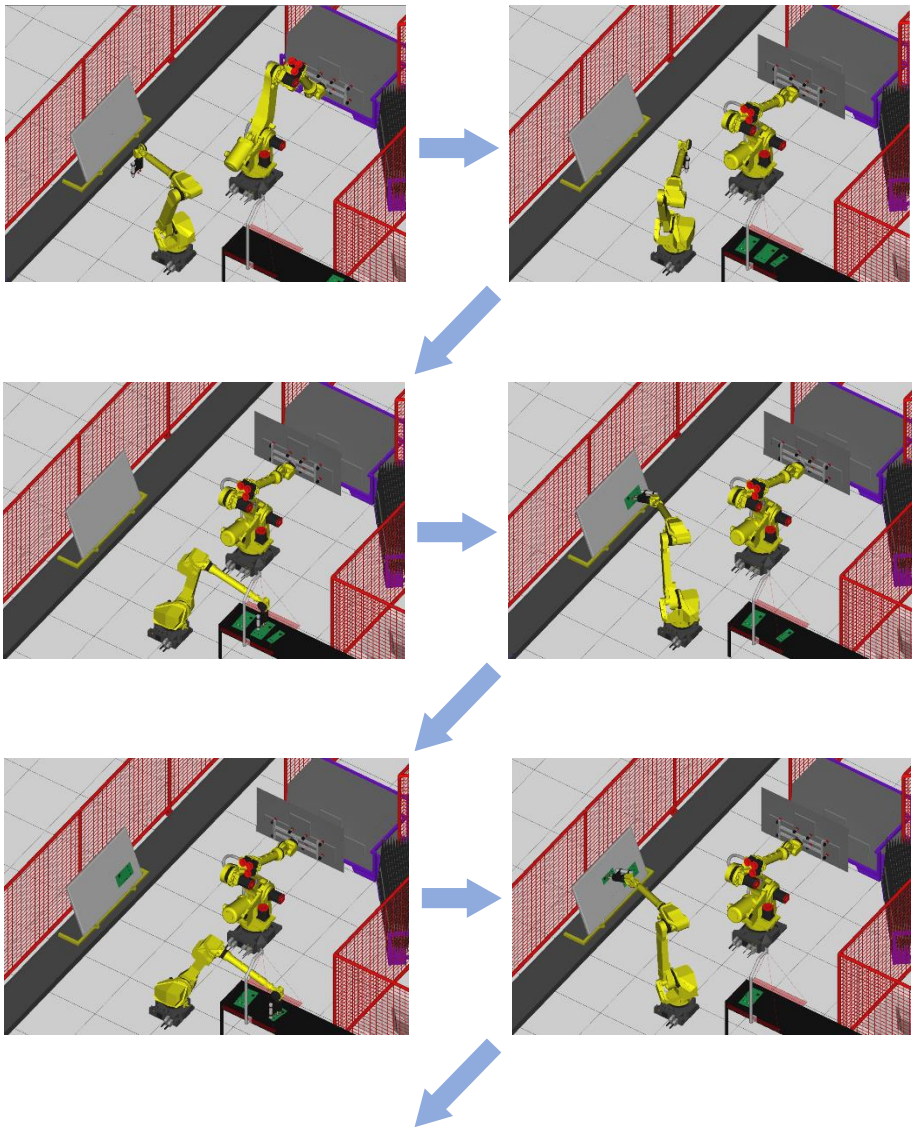


Figura 150. *Movimiento del soporte móvil a la posición de trabajo de R2 y R3.*

El robot M-70iC/45M (R2) apoyado por la funcionalidad iRVision será el encargado de colocar las placas de circuito en su posición, mientras que R3 (R-2000iC/165F) realiza la tarea de extracción de la tapa correspondiente en función del número de placas de circuito generadas (en este caso tapa modelo 2).



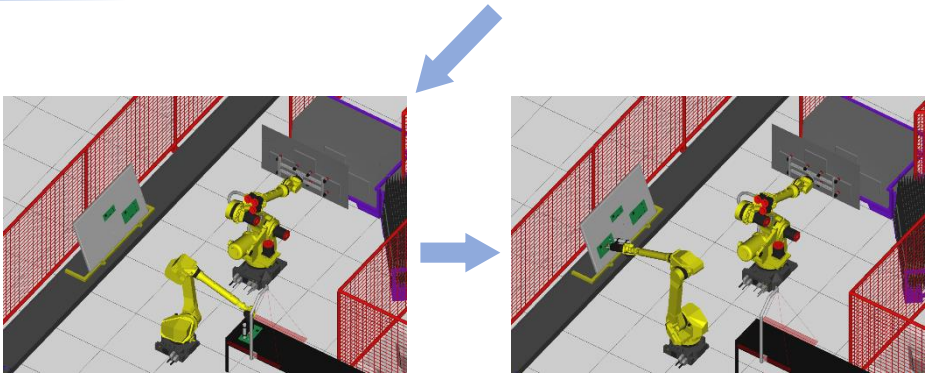


Figura 151. *Secuencia de manipulación de placas de circuito (R2) y extracción de tapa (R3).*

Una vez situadas las placas de circuito, R2 se prepara para el atornillado, tarea que comienza una vez R3 ha situado la tapa en su posición final.

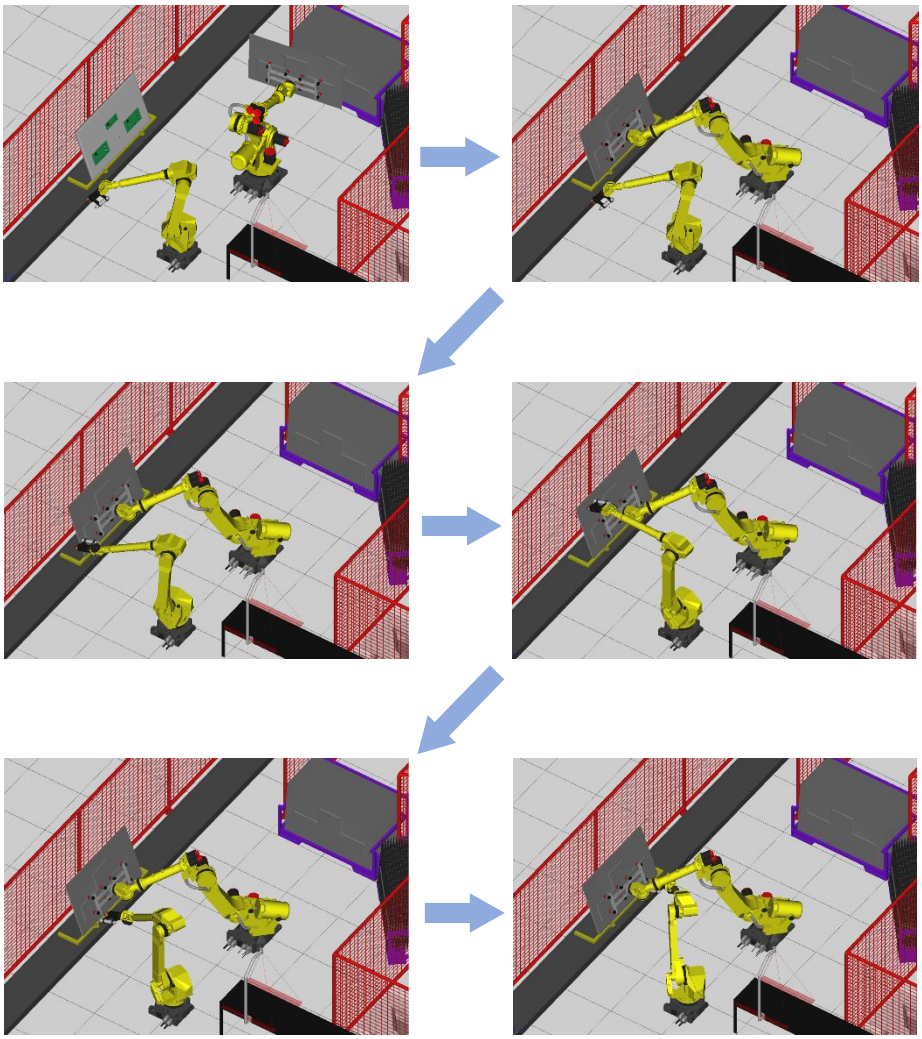


Figura 152. *Primera fase de la secuencia de atornillado.*

Para finalizar, los dos últimos puntos de atornillado deben ser realizados con el robot R3 en una posición que le permita a R2 alcanzar estas posiciones. Para ello en primer lugar R2 se mueve a una posición segura, después del sexto

punto de atornillado; cuando R2 está fuera de la trayectoria de R3, este procede a retirarse.

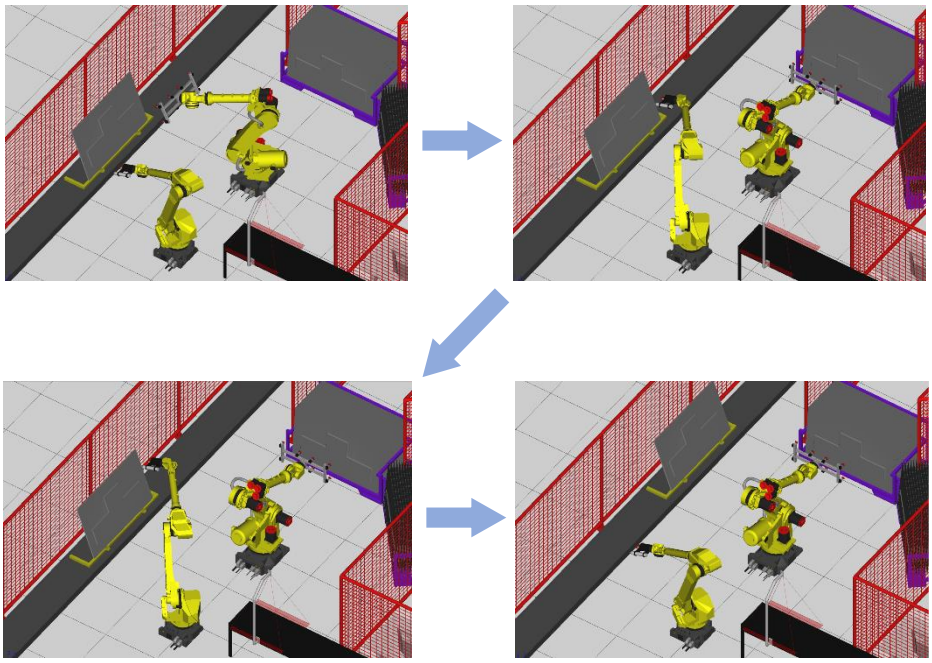


Figura 153. *Finalización de la secuencia de atornillado.*

Una vez finalizada la secuencia de atornillado, se retira el soporte móvil, con el televisor ya ensamblado y los robots R2 y R3 retornan a su posición HOME, donde permanecen a la espera hasta que inicie el siguiente ciclo.

Con esto queda finalizado el apartado de pruebas y resultados, procediendo a continuación a realizar un breve estudio económico de los costes derivados de la realización del proyecto, que a su vez dará paso a poder realizar un análisis sobre los objetivos cumplidos y extraer una serie de valoraciones y conclusiones, así como a una revisión de posibles mejoras o futuros proyectos.

CAPITULO 8. ANÁLISIS ECONÓMICO

En el presente capítulo se realiza un estudio de los costes totales asociados a la elaboración de este proyecto. Se incluyen por lo tanto los precios de las herramientas necesarias, así como el tiempo dedicado al diseño, configuración y programación de todos los elementos necesarios para llevar a cabo la simulación y puesta en funcionamiento de la isla.

Para ello se realizará una división de los costes en función de si son costes directos o indirectos dependiendo de su procedencia. La clasificación de costes en estos grupos se llevará a cabo considerando que el proyecto se diera en un entorno real por un ingeniero industrial para calcular la inversión necesaria.

Estos costes podrían variar si se toman licencias con suscripciones anuales o mensuales, así como si se dispone de descuentos para las mismas por diferentes causas como personal universitario o por incluir dichas licencias como parte de la compra de dispositivos físicos.

Capítulo 8.1. Costes directos

Se asignan los costes directos a los gastos que guardan una relación de procedencia relativa al personal, los materiales y maquinaria necesaria. Es decir, en este caso será aquellos que se asocien de forma directa con la elaboración del proyecto.

Los costes directos se calculan en función del número de horas efectivas de trabajo en un año para así poder calcular la amortización anual resultante. A continuación, se realiza una aproximación del número mínimo de horas efectivas en un año.

NÚMERO DE HORAS EFECTIVAS POR AÑO	
Días en un año	365 días
Días vacacionales por año	30 días
Fines de semana y festivos	109 días
Días de petición extraordinarios	6 días
Horas de trabajo / día	8 horas /día
Total, de horas efectivas al año	1.760 horas / año

Tabla 3. *Horas efectivas anuales.*

Coste del personal

El coste asociado a la mano de obra se estima en función del número de horas trabajadas en un año y el sueldo bruto más incentivos. Esta cuantía puede variar en función de muchos factores como ubicación geográfica, oferta de empleo, experiencia, etc., por lo que se tomará un valor medio.

[41] Además del salario bruto, las empresas deben pagar a la seguridad social el 23,60 % por contingencias comunes y un 0,60 % por formación, más otro 0,20 % de FOGASA y un 5,50 % de la cotización por desempleo si el contrato es fijo o el 6,70 % si el contrato es temporal.

También hay que añadir la cotización por accidentes de trabajo o enfermedad, que varía entre el 1,5 % y el 6,70 % en función del trabajo que se desarrolle.

COSTE SALARIAL DEL PERSONAL	
Concepto	Importe (€/año)
Sueldo bruto + incentivos (Ingenieros técnicos)	26.065,00
Contingencias comunes (23,60%)	6.151,34
Formación (0,60%)	156,39
FOGASA (0,20%)	52,13
Cotización de desempleo (6,70%)	1.746,36
Accidentes de trabajo o enfermedad (1,50%)	390,96
Total, coste personal	34.562,18 €/año

Tabla 4. *Estimación del coste salarial.*

Dividiendo el coste anual del personal entre el número de horas efectivas de trabajo anuales se obtiene el coste horario.

COSTE DEL PERSONAL POR HORA	
Total, coste personal / año	34.562,18 €/año
Numero de hora anuales	1.760 horas
Total, coste por hora	19,63 €/hora

Tabla 5. *Coste del personal por hora.*

Realizando un desglose del número de horas invertidas en cada fase de desarrollo del proyecto se puede calcular el coste del personal en este caso.

COSTE DEL PERSONAL EN EL PROYECTO			
Concepto	Coste por hora (€/hora)	Unidades (horas)	Coste total (€)
Formación y estudio	19,63	120	2355,60
Diseño de la propuesta de proyecto	19,63	20	392,60
Diseño de la célula	19,63	275	5398,25
Programación de robots	19,63	120	2355,60
Programación PLC	19,63	110	2159,30
Programación HMI	19,63	150	2944,50
Total		795 horas	15.605,85 €

Tabla 6. *Coste total del personal en el proyecto.*

Costes materiales amortizables

Son aquellos costes relativos a los equipos informáticos y el software utilizado para la elaboración del proyecto. Se supone un tiempo de amortización tanto para las licencias software como para el hardware de 5 años, lo que supone un porcentaje de amortización anual del 20%.

COSTE SOFTWARE AMORTIZABLE		
Concepto	Importe total	Amortización anual al 20% (€)
SIMATIC STEP 7 V16	2.116,80 €	423,36
PLCSim Advanced 3.0	2.500,00 €	500,00
Roboguide HandlingPro Fanuc	6.000,00 €	1.200,00
KEPServerEx 6	3.714,00 €	742,80
NetToPLCSim	Gratuito	-
Matlab 2021	2.000,00 €	400,00

Autodesk Inventor 2021	2.886,00 €/año	2.886,00
Sistema Operativo Windows 10	136,62 €	136,62
Paquete Microsoft Office 2019	149,00 €	29,80
Total, Coste Software	19.502,42 €	6.208,92 €

Tabla 7. *Estimación de los costes del software.*

COSTE HARDWARE AMORTIZABLE		
Concepto	Importe total (€)	Amortización anual al 20% (€)
Ordenador portátil (Huawei Matebook D15)	649,00	129,80
Monitor (Samsung LF27T350FHRXEN)	152,61	30,52
Periféricos (Teclado, ratón y USB)	25,99	5,20
Total, Coste Hardware	827,60 €	165,52 €

Tabla 8. *Estimación de los costes del hardware.*

En la siguiente tabla se realiza el cálculo de los costes imputables al material amortizable requerido para el proyecto en base al tiempo de amortización estipulado de 5 años.

COSTE DE HARDWARE Y SOFTWARE	
Coste de amortización del software	6.208,92 €/año
Coste de amortización del hardware	165,52 €/año
Total, de horas efectivas por año	1.760 horas/año
Coste por hora del material	3,62 €/hora
Número total de horas del proyecto	795
Total, coste del material	2.879,36 €

Tabla 9. *Costes totales del material amortizable.*

Costes directos totales

En la *Tabla 10* se calculan los costes directos totales como la suma de los costes del personal y los costes del material amortizable.

COSTES DIRECTOS TOTALES	
Concepto	Importe (€)
Costes del personal	15.605,85
Costes del material amortizable	2.879,36
Total, de costes directos	18.485,21 €

Tabla 10. *Costes directos totales del proyecto.*

Capítulo 8.2. Costes indirectos

Se incluyen en este punto los gastos que se generen y que no se puedan atribuir de forma directa a la elaboración del proyecto, pero que son inequívocamente identificables por su sistema de contabilidad. Entre estos gastos encontramos los relativos al suministro eléctrico, internet, servicios administrativos, etc.

COSTES INDIRECTOS TOTALES			
Concepto	Importe por unidad	Número de unidades	Importe total (€)
Consumo eléctrico (estimado)	0,3293 €/kWh	795 horas (aprox. 185kWh)	60,92
Telefonía + Internet (estimado)	30 €/mes	5 meses	150
Gastos administrativos (estimados)	-	-	75
Total, gastos indirectos (estimado)			285,92 €

Tabla 11. *Costes indirectos totales del proyecto.*

Capítulo 8.3. Costes totales

El coste total de desarrollo del proyecto según la filosofía seguida será la suma de costes directos e indirectos calculados anteriormente.

COSTES TOTALES DEL PROYECTO	
Concepto	Importe (€)
Costes directos totales	18.485,21
Costes indirectos totales	285,92
Total	18.771,13 €

Tabla 12. *Costes totales del proyecto.*

CAPÍTULO 9. CONCLUSIONES Y LÍNEAS FUTURAS

Capítulo 9.1. Conclusiones

Se puede concluir tras la finalización de este Trabajo de Fin de Grado que se han cumplido los objetivos que fueron pautados en el Capítulo 1.2. Objetivos.

Comenzando por el software de simulación *Roboguide*, se ha encontrado en este una herramienta muy útil para el aprendizaje en el ámbito de la robótica industrial. Se ha descrito a lo largo del texto como el autor ha desarrollado una célula robotizada orientada al montaje de televisores de grandes dimensiones. El estudio previo de este software ha permitido sacar el mayor partido de la herramienta, añadiendo robots con diferentes funciones y complementos, así como diferentes elementos dinámicos que han permitido realizar una versión digital que posibilita su estudio en diferentes situaciones.

La estación siempre ha estado controlada por un PLC de la marca Siemens, por lo que se ha podido profundizar en el estudio de la herramienta TIA Portal donde se ha realizado la programación y configuración del autómata. Profundizando en la programación KOP y estructurado la programación en bloques funcionales y de datos.

La integración de una interfaz hombre-máquina para la gestión de la célula se ha programado en MATLAB, herramienta ampliamente utilizada por estudiantes para su formación y desarrollo de habilidades de programación, pues sus posibilidades son casi infinitas.

Parte fundamental en este proyecto ha sido encontrar un método que permita comunicar el autómata, HMI y Roboguide en simulación. Para ello se ha recurrido a un protocolo ampliamente extendido como es la comunicación vía OPC y a una herramienta muy polivalente como es KEPServerEX que permite trabajar con diferentes fabricantes y protocolos.

En la realización de este TFG se han analizado diferentes campos muy presentes en la revolución de la industria 4.0, desarrollando conocimientos adquiridos durante el grado, así como con el estudio autónomo de nuevas herramientas que el autor no había utilizado con anterioridad.

Para finalizar, se puede observar en base a los resultados obtenidos como este trabajo asume la función de proporcionar una amplia guía para la realización de futuros proyectos análogos que requieran la utilización de las herramientas descritas a lo largo del informe, o la ampliación del presente proyecto con nuevas funcionalidades, complementando los objetivos principales planteados.

Capítulo 9.2. Líneas futuras

Con el objetivo de dar continuidad al trabajo en la línea propuesta por el autor durante el presente texto, se van a realizar a continuación una serie de propuestas de posibles mejoras o añadidos complementarios que escapan al alcance de este proyecto pero que permiten profundizar en la idea que se ha seguido.

1. Mejora de los conveyor con la herramienta '*Conveyor tracking*'.

Las cintas de llegada de componentes a la estación han sido diseñadas como máquinas para que estas pudieran ser comandadas de forma directa por el autómata. No obstante, existe una funcionalidad interesante en el propio software de simulación Roboguide, para la gestión de cintas transportadoras de objetos que se ajustan mejor a la realidad de un proceso continuo.

Si se desase en un futuro ambientar este proyecto en un entorno más realista, se recomienda profundizar en esta funcionalidad. Para ello, parte de la organización y disposición de la estación debería ser rediseñada.

2. Búsqueda de objetos en los contenedores con la funcionalidad '*Skip*'.

Nuevamente con el objetivo de aumentar el realismo de la estación, se propone para futuros trabajos la utilización de la funcionalidad '*Skip*' para la búsqueda de objetos en los contenedores.

Para la realización de este proyecto se ha supuesto que los contenedores siempre se van a encontrar en la misma posición, lo que facilita la extracción de los objetos conocida la localización del primero de ellos y la separación entre dos piezas contiguas. Pero esto puede no siempre ser así, puede darse el caso de que el operario no posicione siempre el contenedor en el mismo punto, o que la separación entre piezas varie, para ello la funcionalidad '*Skip*' es muy útil, pues con un sensor de presión en la herramienta el robot realiza una búsqueda en la dirección indicada.

3. Añadir nuevas estaciones.

Por último, se propone añadir una nueva estación en un nuevo software de simulación, como podría ser RobotStudio de ABB, en el que se realicen nuevas tareas sobre el televisor, por ejemplo, un empaquetado de los televisores en cajas y un paletizado en grupos de cajas para su salida de la fábrica.

Se considera de especial interés que la estación que se realice este gobernada también por un PLC y gestionada por un HMI, de forma que el funcionamiento conjunto de las dos estaciones quede sincronizado.

BIBLIOGRAFÍA

[1] Sanchis Llopis, R., Romero, Pérez, J. A., Vicent Ariño, C. (2010). *Automatización Industrial. Ingeniería de sistemas industriales y diseño*. Universitat Jaume I.

Recuperado el 25 de enero de 2022 de
http://repositori.uji.es/xmlui/handle/10234/24182

[2] Murillo Sánchez, A. E. (2013, 25 octubre). ¿Qué es un PLC?

Recuperado el 25 de enero de 2022 de
http://www.ctinmx.com/que-es-un-plc/

[3] EDS Robotics (2021, 2 febrero). *Evolución de la robótica*.

Recuperado el 5 de febrero de 2022 de
https://www.edsrobotics.com/blog/evolucion-robotica-industrial/

[4] Vives, Judith (2021, 9 marzo). *La evolución de la robótica, de robots industriales a la IA*.

Recuperado el 5 de febrero de 2022 de
https://www.lavanguardia.com/evolucion-robotica-robots-industriales-ia.html

[5] Ruiz Mitjana, L. (s.f). *Las 3 leyes de la robótica, explicadas.*

Recuperado el 5 de febrero de 2022 de
https://psicologiaymente.com/cultura/leyes-de-robotica

[6] Universidad de Santiago de Chile (s.f). *Desarrollo Histórico y Evolución de la Robótica.*

Recuperado el 6 de febrero de 2022 de
http://www.udesantiagovirtual.cl

[7] British Federal México (2018, 10 octubre). *Los 5 tipos de robots industriales más utilizados por las empresas*.

Recuperado el 7 de febrero de 2022 de
https://www.bfmx.com/automatizacion/tipos-de-robots-industriales-mas-utilizados/

[8] Víctor R. González (2002, marzo). *Robots industriales.*

Recuperado el 8 de febrero de 2022 de
http://platea.pntic.mec.es/robotica/industrial.htm

[9] VLD Engineering (s.f). *Tipos de robots industriales: una pieza clave para ganar competitividad y seguridad laboral.*

Recuperado el 8 de febrero de 2022 de
https://www.vld-eng.com/blog/tipos-de-robots-industriales/

[10] José M.ª Hurtado (s.f). *Introducción a las Redes de Comunicación Industrial*.

Recuperado el 16 de febrero de 2022 de
http://www.infoplc.net/files/documentacion/comunicaciones.pdf

[11] Sicma21 (2021, 22 abril). *Redes de comunicación industrial: todo lo que necesitas saber.*

Recuperado el 16 de febrero de 2022 de
https://www.sicma21.com/que-son-las-redes-de-comunicacion-industrial/

[12] Centro de formación técnica para la industria (s.f). *Qué son las redes de comunicación industrial.*

Recuperado el 16 de febrero de 2022 de
https://www.cursosaula21.com/que-son-las-redes-de-comunicacion-industrial/

[13] Darek Kominek, P. Eng. Alberta (2009). *OPC: ¿De qué se trata, y cómo funciona? "Guía para entender la Tecnología OPC".*

Recuperado el 20 de febrero de 2022 de
https://www.interempresas.net/Guia-para-entender-la-tecnologia-OPC.pdf

[14] Logitek Team (2019, 8 mayo). *Qué es OPC y que es un OPC Server.*

Recuperado el 20 de febrero de 2022 de
https://www.kepserverexopc.com/que-es-opc-y-que-es-un-opc-server/

[15] OPC Foundation, The Industrial Interopability Standard (s.f). *¿Qué es OPC?*

Recuperado el 20 de febrero de 2022 de
https://opcfoundation.org/about/what-is-opc/

[16] Brita Inteligencia Artificial (2021, 10 enero). Visión por computadora o Visión Computación: Guía 2021.

Recuperado el 1 de marzo de 2022 de
https://brita.mx/vision-por-computadora-o-vision-computacion-guia-2021

[17] Esha Chakraborty (s.f). ¿Qué es la visión robótica? Más de 5 aplicaciones importantes.

Recuperado el 1 de marzo de 2022 de
https://es.lambdageeks.com/robotic-vision-important-features/

[18] Atria Innovation (2020, 29 septiembre). Visión artificial y robótica, ¿la pareja perfecta?

Recuperado el 1 de marzo de 2022 de
https://www.atriainnovation.com/vision-artificial-y-robots-la-pareja-perfecta/

[19] VLD Engineering (s.f). *¿Qué entendemos por simulación de procesos industriales?*

Recuperado el 15 de marzo de 2022 de
https://www.vld-eng.com/blog/simulacion-procesos-industriales/

[20] GSL Industrias (2021, 30 agosto). *Simulación Ingeniería Industrial*.

Recuperado el 15 de marzo de 2022 de
https://industriasgsl.com/blogs/automatizacion/simulacion-ingenieria-industrial

[21] Ingenio en marcha Simulación y Analytics (s.f). *Principios de simulación.*

Recuperado el 15 de marzo de 2022 de
https://agiltools.com/blogsp/simulacion/principios/

[22] Pérez Fernández, R. (2016, junio). *Simulación de una instalación de un proceso industrial. PLC, Robot e IHM, mediante OPC* (Trabajo fin de máster). Universidad de Valladolid, Escuela de Ingenierías Industriales. Valladolid.

Recuperado el de 20 marzo de 2022 de
https://uvadoc.uva.es/handle/10324/18998

[23] García Fernández, D. (2017, julio). *Simulación y diagnóstico de una instalación industrial mediante Factory I/O y OPC* (trabajo fin de máster). Universidad de Valladolid, Escuela de Ingenierías Industriales. Valladolid.

Recuperado el de 20 marzo de 2022 de
https://uvadoc.uva.es/handle/10324/24682

[24] Rodríguez-Rabadán Mateos-Aparicio, C. (2019, abril). *Desarrollo y aplicación industrial del software de seguridad de comprobación dual en un robot FANUC* (trabajo fin de grado). Universidad de Valladolid, Escuela de Ingenierías Industriales. Valladolid.

Recuperado el de 20 marzo de 2022 de
https://uvadoc.uva.es/handle/10324/36243

[25] Delgado García, J. (2020, 18 mayo). *Creación de una célula robotizada para formación basada en equipos Fanuc y Siemens* (trabajo fin de grado). Universidad de Valladolid, Escuela de Ingenierías Industriales. Valladolid.

Recuperado el de 20 marzo de 2022 de
https://uvadoc.uva.es/handle/10324/41133

[26] Matía García de los Ríos, L. I. (2020, junio). *Desarrollo de una célula robotizada basada en tecnologías de la industria 4.0* (trabajo fin de grado). Universidad de Valladolid, Escuela de Ingenierías Industriales. Valladolid.

Recuperado el de 20 marzo de 2022 de
https://uvadoc.uva.es/handle/10324/41398

[27] Fanuc (s.f). *Simulación inteligente de robots en 3D.*

Recuperado el 30 marzo de 2022 de
https://www.fanuc.eu/es/es/robots/accesorios/roboguide

[28] Siemens (s.f). *TIA Portal (Totally Integrated Automation Portal).*

Recuperado el 1 abril de 2022 de
https://new.siemens.com/ar/es/productos/automatizacion/software-industrial/tia-portal.html

[29] Siemens (s.f). *Principios básicos de la programación de FC con SIMATIC S7-1200.*

Recuperado el 1 abril de 2022 de
https://www.automation.siemens.com/sce-static/learning-training-documents/tia-portal/basics-programming-s7-1200/sce-031-100-fc-programming-s7-1200-r1709-es.pdf

[30] InfoPLC (2021, 10 enero). *NetToPLCsim Extensión de red para el simulador PLCSim de Siemens.*

Recuperado el 1 abril de 2022 de
https://www.infoplc.net/descargas/107-siemens/software-step7-tiaportal

[31] The MathWorks Inc. (s.f). *Introducción a MATLAB.*

Recuperado el 11 de abril de 2022 de
https://es.mathworks.com/products/matlab/getting-started.html

[32] The MathWorks Inc. (s.f). *Desarrollar apps mediante App Designer.*

Recuperado el 11 de abril de 2022 de
https://es.mathworks.com/help/matlab/app-designer.html

[33] The MathWorks Inc. (s.f). *Industrial Comunication Toolbox.*

Recuperado el 11 de abril de 2022 de
https://es.mathworks.com/products/industrial-communication.html

[34] Autodesk (s.f). *Inventor: software eficaz de diseño mecánico para sus proyectos más ambiciosos.*

Recuperado el 15 de abril de 2022 de
https://www.autodesk.es/products/inventor/

[35] Kepware (s.f). *KEPSERVER: Así es el mejor OPC-UA server*.

Recuperado el 25 de abril de 2022 de
https://www.kepserverexopc.com/kepware-kepserverex-features/

[36] Federación Empresarial Metalúrgica Valenciana, FEMEVAL (s.f). *Robots industriales y cobots en prevención de riesgos laborales.*

Recuperado el 29 de abril de 2022 de
https://www.femeval.es/Guia_Robots.pdf

[37] British Federal México (2019, junio 26). *5 señales de alarma en el manejo de un robot para evitar accidentes.*

Recuperado el 29 de abril de 2022 de
https://www.bfmx.com/automatizacion/senales-de-alarma-manejo-de-robot/

[38] Directiva 2006/42/CE del Parlamento Europeo y del Consejo, de 17 de mayo de 2006, relativa a las máquinas y por la que se modifica la Directiva 95/16/CE (refundición). *Boletín Oficial del Estado,* 157, 9 de junio de 2006.

Recuperado el 9 de mayo de 2022 de
https://www.boe.es/buscar/doc.php?id=DOUE-L-2006-81063

[39] Asociación Española de Normalización (UNE, 2012). *Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 1: Robots.*

Recuperado el 9 de mayo de 2022 de
https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0049289

[40] Asociación Española de Normalización (UNE, 2011). *Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 2: Sistemas Robot e integración.*

Recuperado el 9 de mayo de 2022 de
https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0048668

[41] Ministerio de inclusión, seguridad social y migraciones. Gobierno de España (2022). *Bases y tipos de cotización 2022.*

Recuperado el 30 de mayo de 2022 de
https://www.seg-social.es/CotizacionRecaudacionTrabajadores

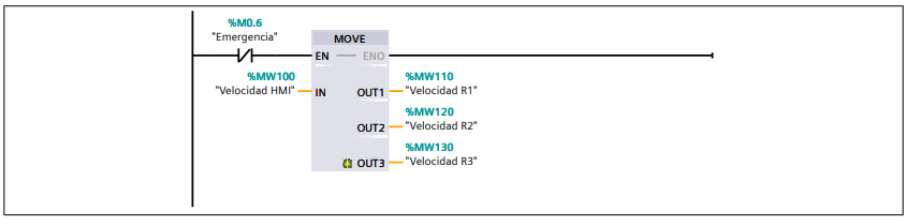
[Página en blanco por convenio]

ANEXOS

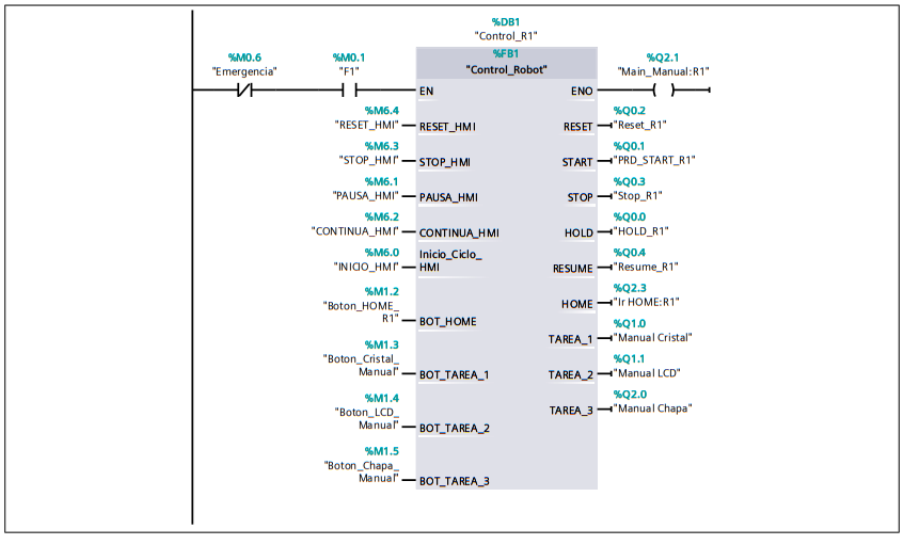
ANEXO A. Bloques de programa TIA Portal

Main (OB1)

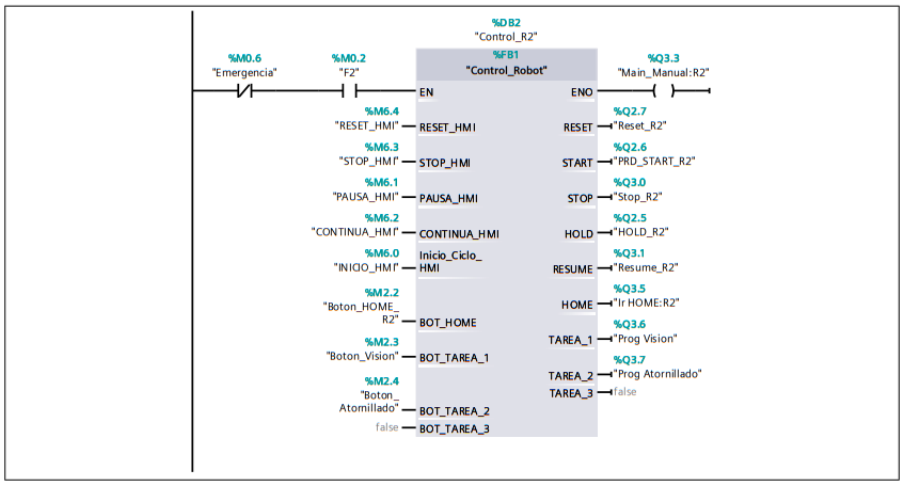
Segmento 1: Establece la velocidad de los robots



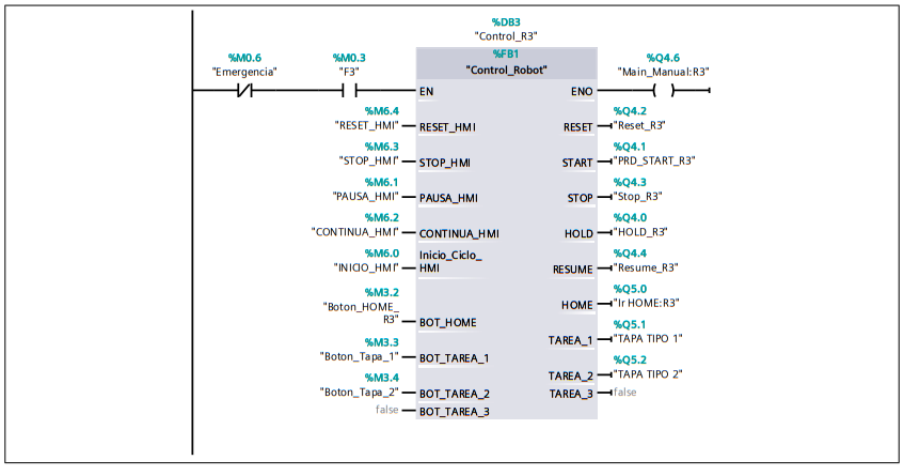
Segmento 2: Control individual de R1 desde HMI



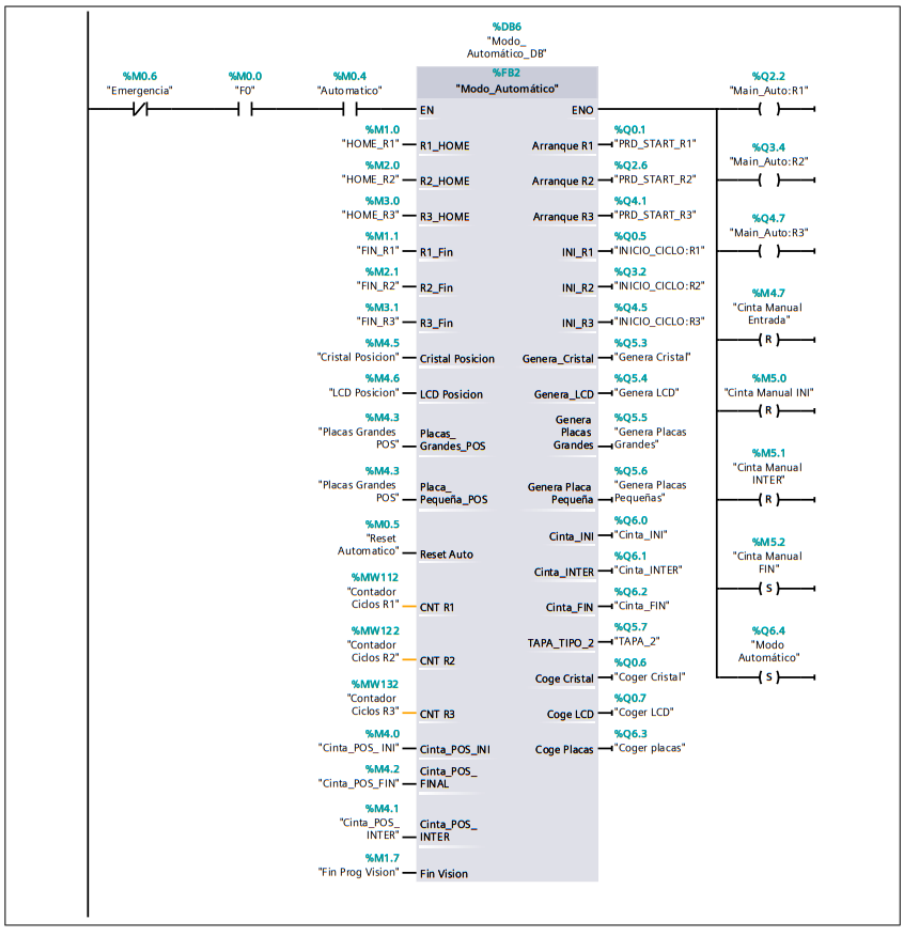
Segmento 3: Ventana Controlador R2



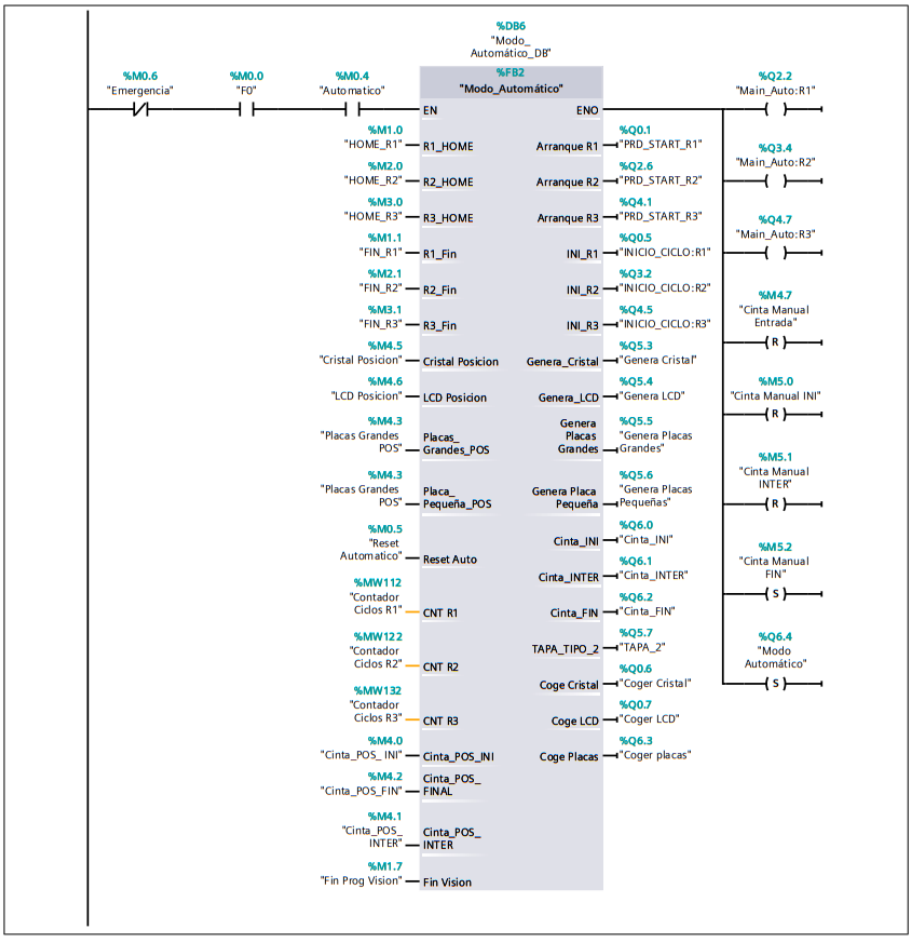
Segmento 4: Control individual de R3 desde HMI



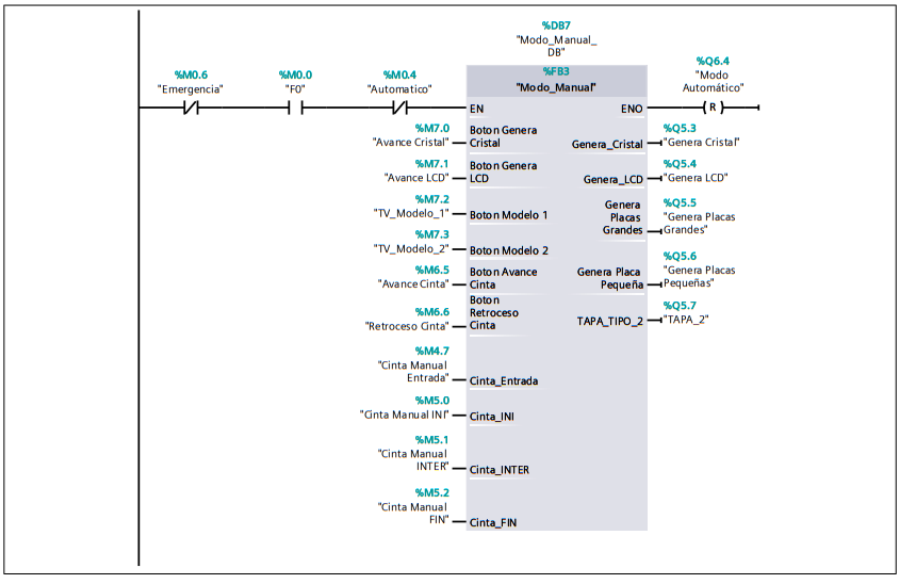
Segmento 5: Modo Automático



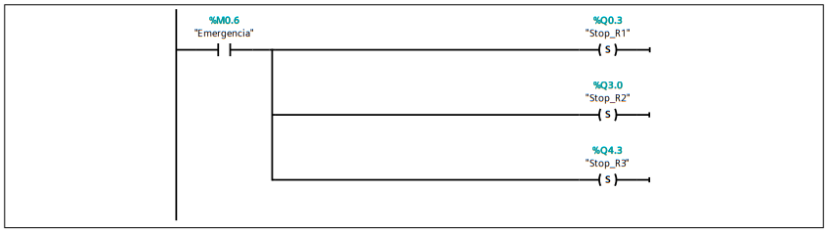
Segmento 5: Modo Automático



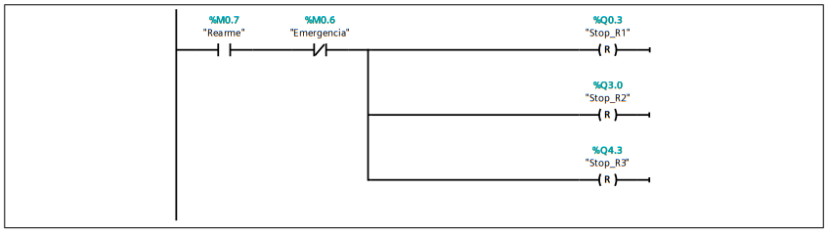
Segmento 6: Modo Manual



Segmento 7: Parada de emergencia

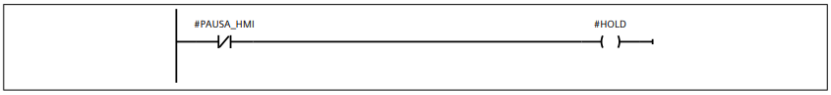


Segmento 8: Rearme de la instalación

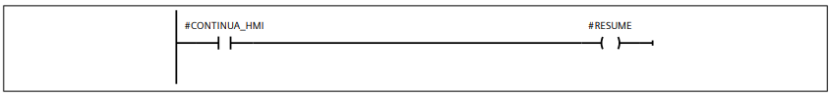


Control Robot (FB1)

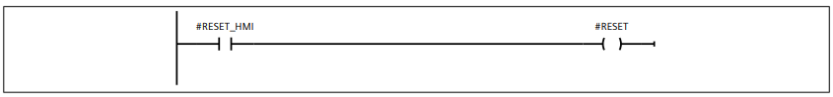
Segmento 1: Establece el estado de pausa en el robot



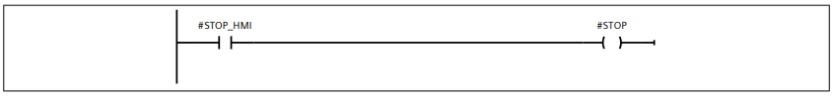
Segmento 2: Reestablece el estado de pausa y continua la marcha



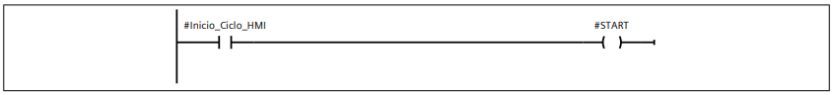
Segmento 3: Elimina un fallo del robot



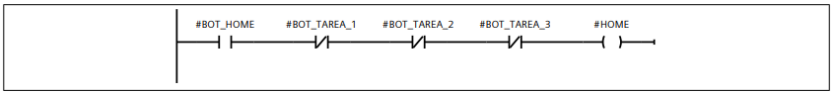
Segmento 4: Parada de emergencia



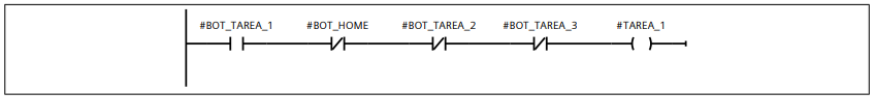
Segmento 5: Inicio de ciclo del programa seleccionado



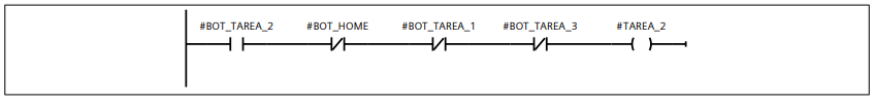
Segmento 6: Selección de movimiento a posición HOME



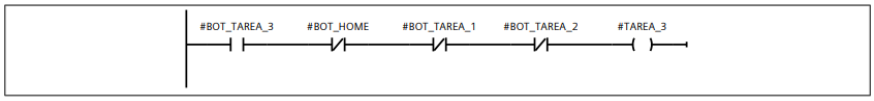
Segmento 7: Selección de la primera tarea asociada al robot



Segmento 8: Selección de la segunda tarea asociada al robot

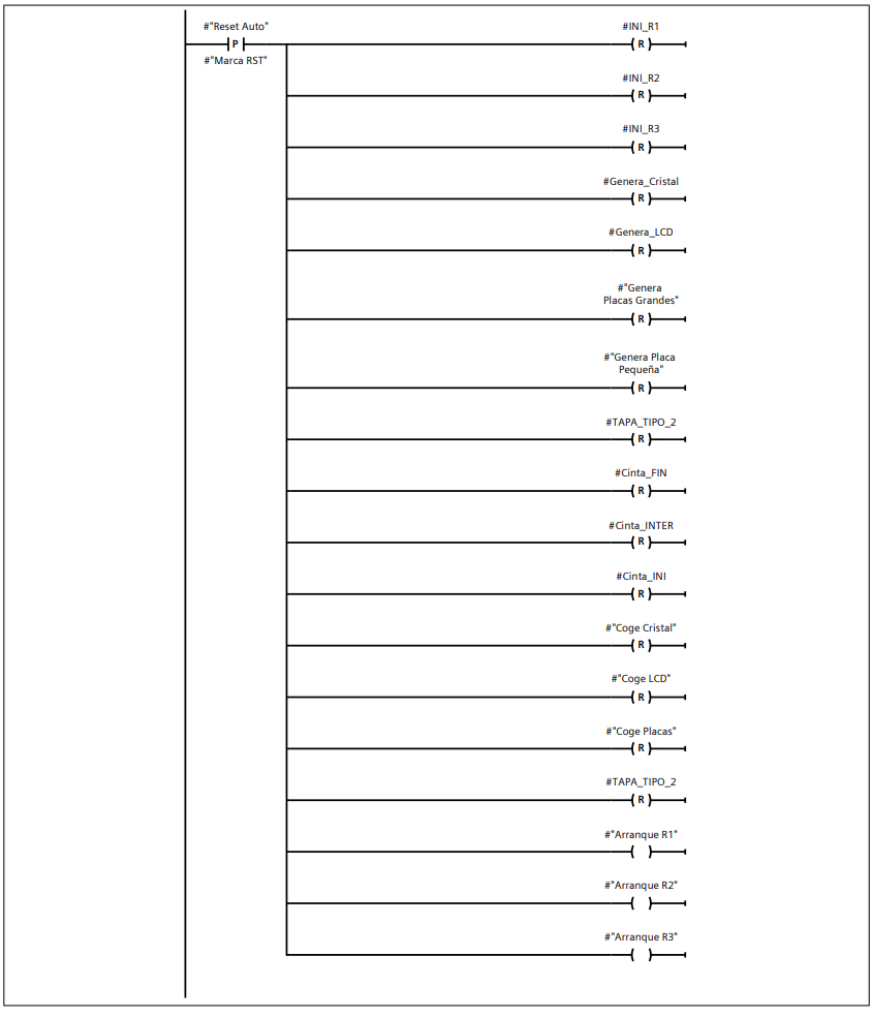


Segmento 9: Selección de la tercera tarea asociada al robot

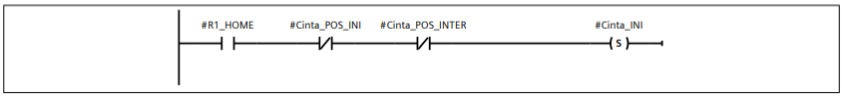


Modo Automático (FB2)

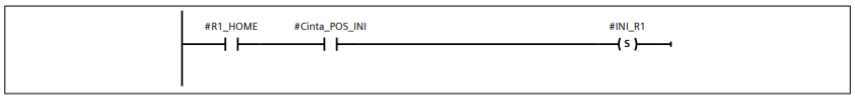
Segmento 1: Comprobacion de primer ciclo y reset de señales



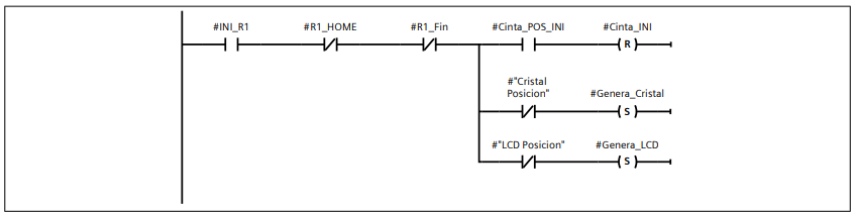
Segmento 2: Movimiento de soporte a posicion de trabajo de R1



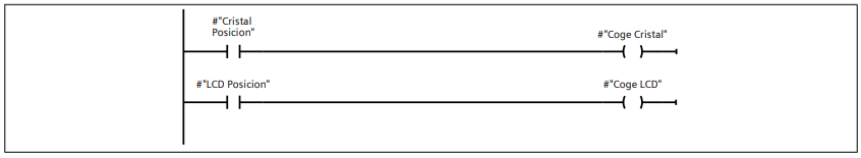
Segmento 3: Inicio de ciclo de R1



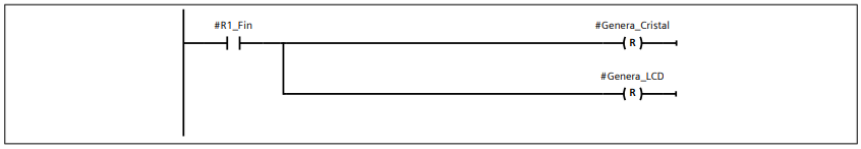
Segmento 4: Genera Cristal y LCD y devuelve el soporte a la posicion de entrada



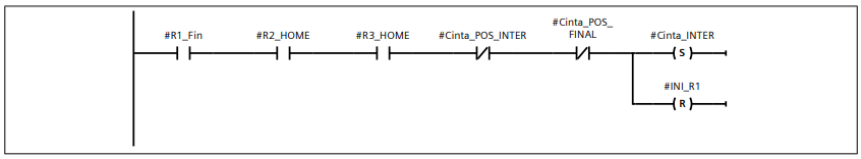
Segmento 5: Autorización para coger Cristal y LCD



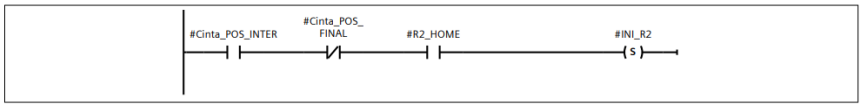
Segmento 6: Fin de ciclo de R1



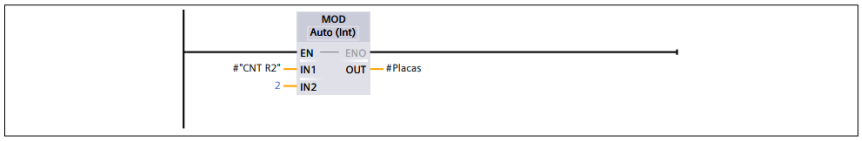
Segmento 7: Movimiento del soporte a la posicion de R2 y R3



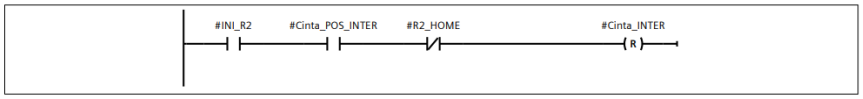
Segmento 8: Inicio de ciclo de R2 y R3



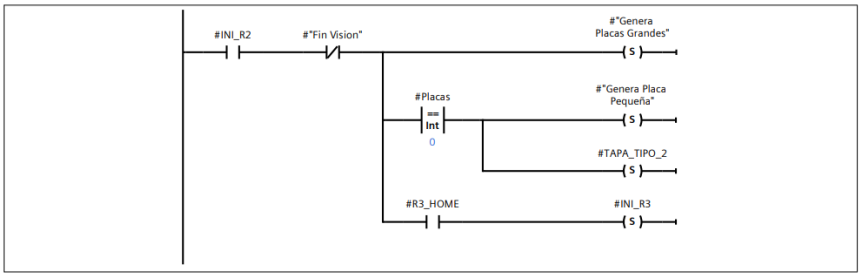
Segmento 9: Cálculo placas de circuito en funcion del ciclo de R2



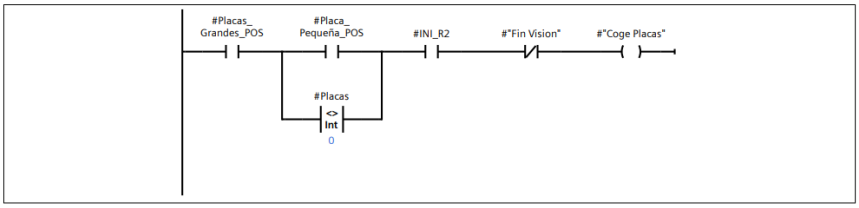
Segmento 10: Devuelve el soporte a la posicion de R1



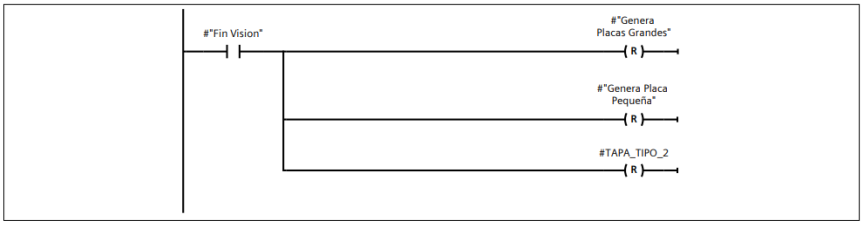
Segmento 11: Genera placas de circuito



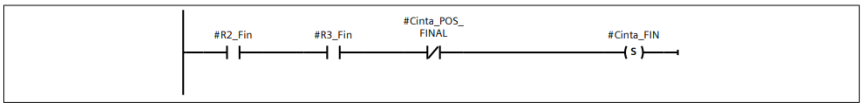
Segmento 12: Autorización para coger placas



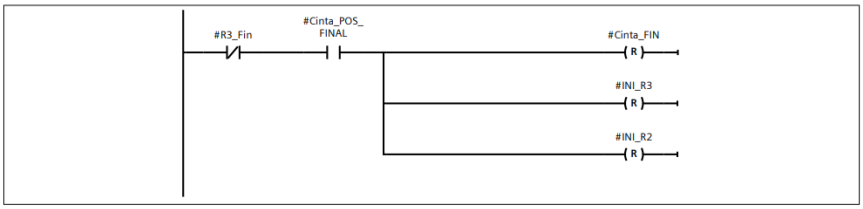
Segmento 13: Devuelve placas al inicio



Segmento 14: Avance hasta la posición de salida del soporte de la cinta móvil

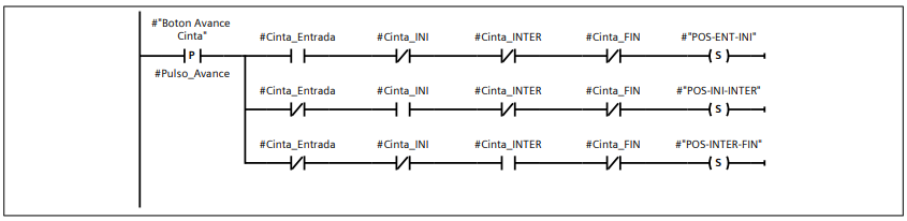


Segmento 15: Reinicio de ciclos R2 y R3

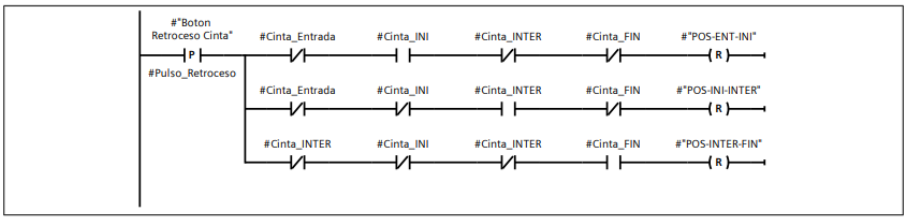


Modo Manual (FB3)

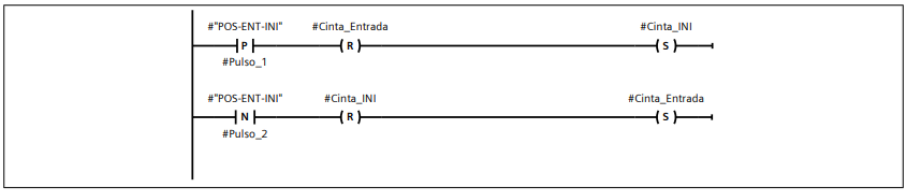
Segmento 1: Selección de posicion de avance según la posicion anterior del soporte



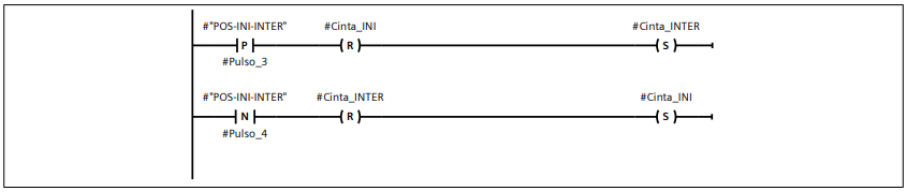
Segmento 2: Selección de posicion de retroceso según la posicion anterior del soporte



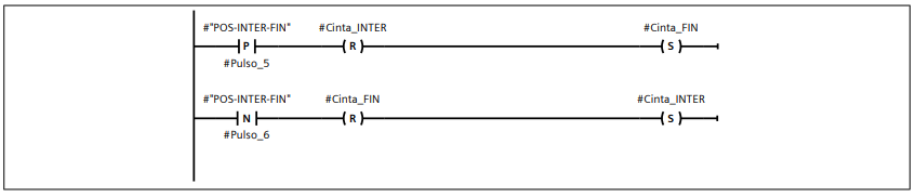
Segmento 3: Movimiento entre posiciones de entrada e inicial



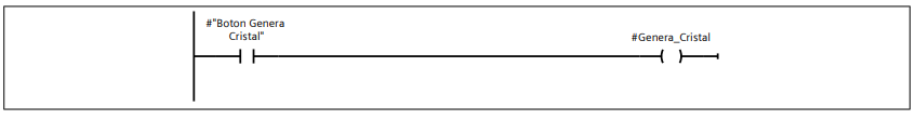
Segmento 4: Movimiento entre posiciones inicial e intermedia



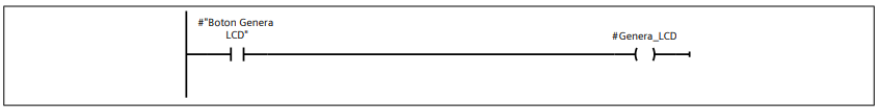
Segmento 5: Movimiento entre posiciones intermedia y final



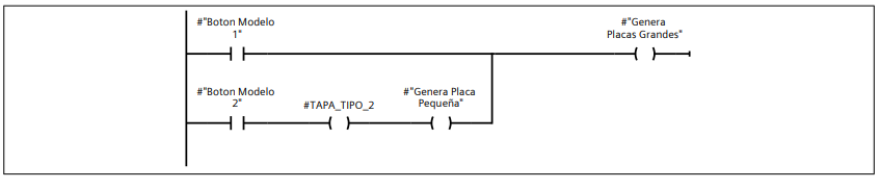
Segmento 6: Desplaza cristal a poscion



Segmento 7: Desplaza LCD a posicion



Segmento 8: Desplaza las placas a posicion en función del modelo seleccionado



ANEXO B. Ejemplos de código en los programas de robots.

Se incluyen en este Anexo algunos ejemplos del código de los programas más relevantes, quedando a disposición del lector la totalidad de los programas desarrollados en archivo comprimido de los anejos a la memoria, concretamente en la carpeta “Roboguide (FANUC)”, donde podrá encontrar los programas organizados por carpetas según a que robot corresponda.

Programa main automático en R-2000iC/125L (RS0110)

```
1:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
2:  !CONTROL AUTOMATICO DE R1 ;
3:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
4:  ;
5:  !RESET INCIAL ;
6:  CALL RESET_INICIAL    ;
7:  ;
8:  LBL[10] ;
9:  !HERRAMIENTA Y BASE DE TRABAJO ;
10:  UFRAME_NUM=0 ;
11:  UTOOL_NUM=1 ;
12:  ;
13:  !MOVIMIENTO A HOME ;
14:J PR[1:HOME] 100% FINE    ;
15:  WAIT    .10(sec) ;
16:  DO[20:R1 EN HOME]=ON ;
17:  ;
18:  !ESPERA SENIAL DEL PLC ;
19:  WAIT DI[20:INICIO DE CICLO]=ON    ;
20:  ;
21:  !RESET VARIABLES Y SENIALES ;
22:  CALL RESET_SENIALES    ;
23:  ;
24:  WAIT DI[21:COGER CRISTAL]=ON    ;
25:  ;
26:  !MOV. COGER CRISTAL DE CONVEYOR ;
27:  CALL COGER_CRISTAL    ;
28:  ;
29:  !MOV. DEJADA DE CRISTAL EN POS. ;
30:  CALL DEJAR_CRISTAL    ;
31:  ;
32:  WAIT DI[22:COGER LCD]=ON    ;
33:  ;
34:  !MOV. COGER LCD DE CONVEYOR ;
35:  CALL COGER_LCD    ;
36:  ;
37:  !MOV. DEJADA DE LCD EN POS. ;
38:  CALL DEJAR_LCD    ;
39:  ;
40:  !MOV. COGER CHAPA DE CONTENEDOR ;
41:  CALL COGER_CHAPA    ;
42:  ;
43:  !MOV. DEJADA DE CHAPA EN POS. ;
44:  CALL DEJAR_CHAPA    ;
45:  ;
46:  R[10:CNT_CICLO]=R[10:CNT_CICLO]+1    ;
47:  ;
48:  !INDICA FIN Y ESPERA CONFIRMACION ;
49:  DO[21:FIN R1]=ON ;
50:  WAIT DI[20:INICIO DE CICLO]=OFF    ;
51:  DO[21:FIN R1]=OFF ;
52:  ;
53:  JMP LBL[10] ;
```

Programa main manual en R-2000iC/125L (RS0120)

```
1:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
2:  !CONTROL MANUAL CON EL HMI ;
3:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
4:  ;
5:  !RESET INCIAL ;
6:  CALL RESET_SENIALES    ;
7:  ;
8:  LBL[10] ;
9:  ;
10:  IF DI[24:MANUAL HOME]=ON,JMP LBL[20] ;
11:  IF DI[25:MANUAL CRISTAL]=ON,JMP LBL[30] ;
12:  IF DI[26:MANUAL LCD]=ON,JMP LBL[40] ;
13:  IF DI[27:MANUAL CHAPA ALUMINIO]=ON,JMP LBL[50] ;
14:  IF DI[28:MODO AUTOMATICO]=ON,JMP LBL[60] ;
15:  ;
16:  JMP LBL[10] ;
17:  ;
18:  !MOVIMIENTO A HOME ;
19:  LBL[20] ;
20:J PR[1:HOME] 100% FINE    ;
21:  JMP LBL[60] ;
22:  ;
23:  !MANIPULACION DE CRISTALES ;
24:  LBL[30] ;
25:  CALL COGER_CRISTAL    ;
26:  CALL DEJAR_CRISTAL    ;
27:  JMP LBL[60] ;
28:  ;
29:  !MANIPULACION PANELES LCD ;
30:  LBL[40] ;
31:  CALL COGER_LCD    ;
32:  CALL DEJAR_LCD    ;
33:  JMP LBL[60] ;
34:  ;
35:  !MANIPULACION CHAPAS ALUMINIO ;
36:  LBL[50] ;
37:  CALL COGER_CHAPA    ;
38:  CALL DEJAR_CHAPA    ;
39:  JMP LBL[60] ;
40:  ;
41:  LBL[60] ;
```

Programa coger chapas de aluminio en R-2000iC/125L

```
  1: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  2: !EXTRACCION CHAPA DE CONTENEDOR ;
  3: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  4:  ;
  5: !SELECCION BASE DE TRABAJO ;
  6: UFRAME_NUM=0 ;
  7:  ;
  8: !ESTABLECE VELOCIDAD DE HMI ;
  9: OVERRIDE=R[1:VELOCIDAD_OPC] ;
 10:  ;
 11: !RESET PR AUX ;
 12: PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX] ;
 13:  ;
 14: !CALCULO POS. CHAPA ACTUAL ;
 15:
R[12:AUX_CHAPA]=R[11:SEP_CHAPA]*R[15:CNT_CHAPA] ;
 16:  ;
 17: !MOVIMIENTO DE APROXIMACION ;
 18:J P[1:Aproximacion] 100% CNT100    ;
 19: !MOVIMIENTO A POS. CALCULADA ;
 20: PR[10,1:PR_AUX]=(-500)    ;
 21:L PR[4:TP_Chapa_Alumino] 2000mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 22:  ;
 23: PR[10,1:PR_AUX]=(-50)+R[12:AUX_CHAPA] ;
 24:L PR[4:TP_Chapa_Alumino] 2000mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 25:  ;
 26: PR[10,1:PR_AUX]=R[12:AUX_CHAPA]    ;
 27:L PR[4:TP_Chapa_Alumino] 500mm/sec FINE
Offset,PR[10:PR_AUX]    ;
 28:  ;
 29: !COGIDA DE CHAPA ;
 30: CALL PICK_CHAPA_ALUMINIO    ;
 31: WAIT    .10(sec) ;
 32:  ;
 33: !MOVIMIENTO DE EXTRACCION ;
 34: PR[10,3:PR_AUX]=20    ;
 35:L PR[4:TP_Chapa_Alumino] 500mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 36:  ;
 37: PR[10,1:PR_AUX]=0    ;
 38:L PR[4:TP_Chapa_Alumino] 500mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 39:  ;
 40: PR[10,1:PR_AUX]=(-500)    ;
 41:L PR[4:TP_Chapa_Alumino] 2000mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 42:  ;
 43:L P[1:Aproximacion] 2000mm/sec CNT100    ;
 44:  ;
 45: R[15:CNT_CHAPA]=R[15:CNT_CHAPA]+1    ;
 46:  ;
```

Programa dejar chapas de aluminio en R-2000iC/125L

```
  1: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  2: !DEPOSITO DE CHAPA EN POSICION ;
  3: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  4:  ;
  5: !ESTABLECIMIENTO DE VEL. DE HMI ;
  6: OVERRIDE=R[1:VELOCIDAD_OPC] ;
  7:  ;
  8: !SELECCION BASE DE TRABAJO ;
  9: UFRAME_NUM=1 ;
 10:  ;
 11: !RESET PR AUX ;
 12: PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX];
 13:  ;
 14: !MOVIMIENTO DE APROXIMACION ;
 15:J P[1:Aproximacion 1] 100% CNT100    ;
 16: !AJUSTE DE OFFSET Y MOV. FINAL ;
 17: PR[10,2:PR_AUX]=50    ;
 18: PR[10,3:PR_AUX]=100    ;
 19:L PR[7:DP_Chapa_Alumino] 2000mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 20:  ;
 21: PR[10,2:PR_AUX]=0    ;
 22: PR[10,3:PR_AUX]=50    ;
 23:L PR[7:DP_Chapa_Alumino] 500mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 24:  ;
 25:L PR[7:DP_Chapa_Alumino] 100mm/sec FINE    ;
 26:  ;
 27: !DEJADA DE CHAPA ;
 28: IF DI[28:MODO AUTOMATICO]=OFF,JMP LBL[10];
 29: CALL DROP_CHAPA_ALUMINIO_AUTO    ;
 30: WAIT    .20(sec) ;
 31: JMP LBL[20] ;
 32: LBL[10] ;
 33: CALL DROP_CHAPA_ALUMINIO_MANUAL    ;
 34: WAIT    .20(sec) ;
 35: LBL[20] ;
 36:  ;
 37: !RETIRADA DE GARRA ;
 38:L PR[7:DP_Chapa_Alumino] 500mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 39:  ;
 40:J P[1:Aproximacion 1] 100% CNT100    ;
```

Programa de atornillado en M-710iC/45M

```
 1: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
 2: !ATORNILLADO DE TAPAS ;
 3: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
 4:  ;
 5: !ESTABLECE VEL. DEL HMI ;
 6: OVERRIDE=R[1:VELOCIDAD_OPC] ;
 7:  ;
 8: !HERRAMIENTA Y BASE TRABAJO ;
 9: UTOOL_NUM=2 ;
10: UFRAME_NUM=2 ;
11:  ;
12: !MOVIMIENTO DE APROXIMACION ;
13:J PR[9:AUX_2] 100% FINE    ;
14: DO[22:R2 POS. PRE-ATORNILLADO]=ON ;
15:  ;
16: !RESET PR AUX Y OFFSET_Z ;
17: PR[10:OFFSET_AUX]=PR[10:OFFSET_AUX]-
PR[10:OFFSET_AUX];
18: PR[10,3:OFFSET_AUX]=50    ;
19:  ;
20: !ESPERA R3 EN POSICION ;
21: WAIT DI[22:R3 POS. PRE-ATORNILLADO]=ON ;
22:  ;
23: !INICIO DEL ATORNILLADO ;
24:J P[2] 100% CNT25 Offset,PR[10:OFFSET_AUX];
25:L P[2] 100mm/sec FINE    ;
26: WAIT    .50(sec) ;
27:L P[2] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
28:  ;
29:L P[1] 3000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
30:L P[1] 100mm/sec FINE    ;
31: WAIT    .50(sec) ;
32:L P[1] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
33:  ;
34:L P[3] 3000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
35:L P[3] 100mm/sec FINE    ;
36: WAIT    .50(sec) ;
37:L P[3] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
38:  ;
39:L P[4] 3000mm/sec CNT25 Offset,PR[10:OFFSET_AUX] ;
40:L P[4] 100mm/sec FINE    ;
41: WAIT    .50(sec) ;
42:L P[4] 1000mm/sec CNT25
Offset,PR[10:OFFSET_AUX];
43:  ;
44:L P[5] 3000mm/sec CNT100    ;
45:J P[11] 100% CNT100    ;
46:  ;
47:L P[6] 3000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]
;
48:L P[6] 100mm/sec FINE    ;
49: WAIT    .50(sec) ;
50:L P[6] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]
;
 51:  ;
52:L P[7] 3000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]
;
53:L P[7] 100mm/sec FINE    ;
54: WAIT    .50(sec) ;
55:L P[7] 1000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]
;
56:  ;
57:L P[10] 3000mm/sec FINE    ;
58:  ;
59: !R2 EN POSICION SEGURA ;
60: DO[23:R2 POS. SEGURA PARA R3]=ON ;
61:  ;
62: !Espera retirada de R3 ;
63: WAIT DI[22:R3 POS. PRE-ATORNILLADO]=OFF;
64:  ;
65:J P[8] 100% CNT25 Offset,PR[10:OFFSET_AUX];
66:L P[8] 100mm/sec FINE    ;
67: WAIT    1.00(sec) ;
68:L P[8] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]
;
69:  ;
70:L P[9] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]
;
71:L P[9] 100mm/sec FINE    ;
72: WAIT    1.00(sec) ;
73:L P[9] 2000mm/sec CNT25 Offset,PR[10:OFFSET_AUX]
;
74:  ;
75:J P[10] 100% CNT100    ;
76:  ;
```

Programa de visión artificial en M-710iC/45M

```
 1: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
 2: ! IRVISION DETECTA ID PLACA ;
 3: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
 4:  ;
 5: !ESTABLECE VEL. DEL HMI ;
 6: OVERRIDE=R[1:VELOCIDAD_OPC] ;
 7:  ;
 8: !HERRAMIENTA Y BASE DE TRABAJO ;
 9: UTOOL_NUM=1 ;
10: UFRAME_NUM=1 ;
11:  ;
12: !INICIO ESCANEO POR VISION ;
13: VISION RUN_FIND 'LOCALIZADOR'    ;
14: VISION GET_NFOUND 'LOCALIZADOR' R[10];
15:  ;
16: R[11:Copia num placas]=R[10:Placas encontradas]  ;
17:  ;
18: IF R[10:Placas encontras]=0,JMP LBL[100];
19:  ;
20: LBL[10] ;
21:  ;
22: !GUARDA EL OFFSET EN R11 ;
23: VISION GET_OFFSET 'LOCALIZADOR' VR[R[10]] JMP
LBL[100] ;
24:  ;
25: R[10:Placas encontras]=R[10:Placas encontras]-1    ;
26:  ;
27: IF R[10:Placas encontras]<>0,JMP LBL[10];
28:  ;
29: LBL[20] ;
30:  ;
31: R[12]=VR[R[11]].MODELID ;
32:  ;
33:  ;
34: IF R[12:ID de placa]=3,JMP LBL[40] ;
35: IF R[12:ID de placa]=2,JMP LBL[30] ;
36:  ;
37: !ID 0001 RECONOCIDO ;
38: CALL ID_0001    ;
39: R[11:Copia num placas]=R[11:Copia num placas]-1    ;
40: IF R[11:Copia num placas]<>0,JMP LBL[20]; 41:  JMP
LBL[100] ;
42:  ;
43: !ID 0002 RECONOCIDO ;
44: LBL[30] ;
45: CALL ID_0002    ;
46: R[11:Copia num placas]=R[11:Copia num placas]-1    ;
47: IF R[11:Copia num placas]<>0,JMP LBL[20];
48: JMP LBL[100] ;
49:  ;
50: !ID 0003 RECONOCIDO ;
51: LBL[40] ;
52: CALL ID_0003    ;
53: R[11:Copia num placas]=R[11:Copia num placas]-1    ;
54: IF R[11:Copia num placas]<>0,JMP LBL[20] ;
55:  ;
56: LBL[100] ;
57:  ;
58: DO[24:FIN VISION]=ON ;
59:  ;
```

Programa de manipulación de placas con ID 0001 en M-710iC/45M

```
1:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
2:  !MANIPULACION PLACAS CON ID-0001 ;
3:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
4:  ;
5:  !ESTABLECE VEL. DEL HMI ;
6:  OVERRIDE=R[1:VELOCIDAD_OPC] ;
7:  ;
8:  !RESET PR AUX ;
9:  PR[10:OFFSET_AUX]=PR[10:OFFSET_AUX]-
PR[10:OFFSET_AUX]    ;
10:  ;
11:  PR[10,2:OFFSET_AUX]=(-350)    ;
12:  PR[10,3:OFFSET_AUX]=250    ;
13:  ;
14:  !CAMBIO DE BASE DE TRABAJO ;
15:  UFRAME_NUM=1 ;
16:  ;
17:  !MOVIMIENTO AUX DE RETIRADA ;
18:J PR[8:AUX_1] 100% CNT100    ;
19:  ;
20:  !MOVIMIENTO A POSICION DE COGIDA ;
21:L PR[5:TP_REF_ID_0001] 2000mm/sec CNT25
Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
22:  ;
23:  PR[10,2:OFFSET_AUX]=0    ;
24:L PR[5:TP_REF_ID_0001] 2000mm/sec CNT25
Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
25:  ;
26:L PR[5:TP_REF_ID_0001] 500mm/sec FINE
VOFFSET,VR[R[11]]    ;
27:  ;
28:  !COGE PLACA ;
29:  CALL PICK_ID_0001    ;
30:  WAIT    .50(sec) ;
31:  ;
32:  !EXTRACCION PLACA ;
33:L PR[5:TP_REF_ID_0001] 500mm/sec CNT25
Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
34:  ;
35:  PR[10,2:OFFSET_AUX]=(-350)    ;
36:L PR[5:TP_REF_ID_0001] 2000mm/sec CNT25
Offset,PR[10:OFFSET_AUX] VOFFSET,VR[R[11]]    ;
37:  ;
38:  !MOVIMIENTO AUX A POS. DEJADA ;
39:J PR[8:AUX_1] 100% CNT100    ;
40:  ;
41:  !ESTABLECE VEL. DEL HMI ;
42:  OVERRIDE=R[1:VELOCIDAD_OPC] ;
43:  ;
44:  !CAMBIO DE BASE DE TRABAJO ;
45:  UFRAME_NUM=2 ;
46:  ;
47:  !MOVIMIENTO A POS. DEJADA ;
48:  PR[10,2:OFFSET_AUX]=0    ;
49:J PR[2:DP_ID_0001] 100% CNT25 Offset,PR[10:OFFSET_AUX] ;
50:L PR[2:DP_ID_0001] 500mm/sec FINE    ;
51:  ;
52:  !DEJA PLACA ;
53:  IF DI[27:MODO AUTOMATICO]=OFF,JMP LBL[10] ;
54:  CALL DROP_ID_0001_AUTO    ;
55:  WAIT    .50(sec) ;
56:  JMP LBL[20] ;
57:  LBL[10] ;
58:  CALL DROP_ID_0001_MANUAL    ;
59:  WAIT    .50(sec) ;
60:  LBL[20] ;
61:  ;
62:  !RETIRADA DE LA HERRAMIENTA ;
63:L PR[2:DP_ID_0001] 500mm/sec CNT25
Offset,PR[10:OFFSET_AUX]    ;
```

Programa de cogida de tapas modelo 1 en R-2000iC/165F

```
1:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
2:  !EXTRACCION DE TAPAS MODELO 1 ;
3:  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
4:  ;
5:  !ESTABLECE VELOCIDAD CON HMI ;
6:  OVERRIDE=R[1:VELOCIDAD_OPC] ;
7:  ;
8:  !SELECCION BASE TRABAJO ;
9:  UFRAME_NUM=1 ;
10:  ;
11:  !RESET PR AUX ;
12:  PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX]    ;
13:  ;
14:  !CALCULO POSICION DE TAPA ACTUAL ;
15:  R[14:Z_OFFSET]=R[14:Z_OFFSET]-R[14:Z_OFFSET] ;
16:
R[14:Z_OFFSET]=R[11:CNT_MODELO1]*R[13:SEP_TAPAS];
17:  ;
18:  !MOVIMIENTO DE APROXIMACION ;
19:J P[1] 100% CNT100    ;
20:  ;
21:  !MOVIMIENTO HACIA PTO. DE COGIDA ;
22:  PR[10,3:PR_AUX]=300    ;
23:J PR[2:TP_Modelo1] 100% CNT100
Offset,PR[10:PR_AUX]    ;
24:  ;
25:  PR[10,3:PR_AUX]=0-R[14:Z_OFFSET]    ;
26:L PR[2:TP_Modelo1] 500mm/sec FINE
Offset,PR[10:PR_AUX]    ;
27:  ;
28:  !COGER TAPA ;
29:  CALL PICK_MODELO_1    ;
30:  WAIT    .50(sec) ;
31:  R[11:CNT_MODELO1]=R[11:CNT_MODELO1]+1    ;
32:  ;
33:  !RETIRA TAPA DE CONTENEDOR ;
34:  PR[10,2:PR_AUX]=50    ;
35:  PR[10,3:PR_AUX]=0    ;
36:L PR[2:TP_Modelo1] 500mm/sec CNT25
Offset,PR[10:PR_AUX]    ;
37:  ;
38:  PR[10,3:PR_AUX]=300    ;
39:L PR[2:TP_Modelo1] 1000mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
40:  ;
41:  !APROXIMACION DE SALIDA ;
42:L P[1] 3000mm/sec CNT100    ;
43:  ;
44:  !VUELTA A HOME ;
45:J PR[1:HOME] 66% CNT100    ;
46:  ;
```

Programa de cogida de tapas modelo 1 en R-2000iC/165F

```
  1: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  2: !DEPOSITO TAPA MODELO 1 EN POSICION ;
  3: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ;
  4:  ;
  5: !ESTABLECE VELOCIDAD CON HMI ;
  6: OVERRIDE=R[1:VELOCIDAD_OPC] ;
  7:  ;
  8: !SELECCION BASE DE TRABAJO ;
  9: UFRAME_NUM=3 ;
 10:  ;
 11: !RESET PR AUX ;
 12: PR[10:PR_AUX]=PR[10:PR_AUX]-PR[10:PR_AUX]    ;
 13:  ;
 14: !MOVIMIENTO DE APROXIMACION ;
 15:J PR[7:AUX_1] 100% CNT100    ;
 16:  ;
 17: !MOVIMIENTO A POSCION DEJADA ;
 18: PR[10,3:PR_AUX]=(-250)    ;
 19: PR[10,2:PR_AUX]=50    ;
 20:L PR[4:DP_Modelo1] 2000mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 21:  ;
 22: PR[10,3:PR_AUX]=(-20)    ;
 23: PR[10,2:PR_AUX]=50    ;
 24:L PR[4:DP_Modelo1] 500mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 25:  ;
 26: PR[10,2:PR_AUX]=0    ;
 27:L PR[4:DP_Modelo1] 100mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 28:  ;
 29:L PR[4:DP_Modelo1] 100mm/sec FINE    ;
 30: !R3 EN POSICION DE ATORNILLADO ;
 31: DO[22:R3 EN POS. ATORNILLADO]=ON ;
 32:  ;
 33: !DEJAR TAPA ;
 34: IF DI[28:MODO AUTOMATICO]=OFF,JMP LBL[10] ;
 35: CALL DROP_MODELO_1_AUTO    ;
 36: WAIT    .50(sec) ;
 37: JMP LBL[20] ;
 38: LBL[10] ;
 39: CALL DROP_MODELO_1_MANUAL    ;
 40: WAIT    .50(sec) ;
 41: LBL[20] ;
 42:  ;
 43: !ESPERA A R2 EN POSCION FINAL ;
 44: WAIT DI[23:R2 POS. SEGURA]=ON    ;
 45:  ;
 46: !RETIRADA DE GARRA ;
 47: PR[10,2:PR_AUX]=0    ;
 48: PR[10,3:PR_AUX]=(-250)    ;
 49:L PR[4:DP_Modelo1] 2000mm/sec CNT100
Offset,PR[10:PR_AUX]    ;
 50:  ;
 51:J PR[7:AUX_1] 100% CNT100    ;
 52:  ;
 53: DO[22:R3 EN POS. ATORNILLADO]=OFF ;
 54:  ;
 55:J PR[1:HOME] 100% FINE    ;
 56:  ;
```

ANEXO C. Ejemplos de código en la interfaz de usuario

Con el objeto de no alargar la extensión del informe, tan solo se va a añadir al igual que en el apartado anterior, el código que se considera más relevante, pudiendo encontrar el resto del código en los anejos a la memoria, en la carpeta destinada al software MATLAB.

Interfaz control de proceso

Definición de propiedades de acceso publico

```
properties (Access = public)
Cliente_HMI_Proceso;
Proceso;
Automatico;
Reset_Automatico;
Avance;
Retroceso;
Entrada_Cristal;
Entrada_LCD;
TV_Modelo_1;
TV_Modelo_2;
Velocidad;
Emergencia;
Rearme;
STOP_R1;
STOP_R2;
STOP_R3;
F0;
F1;
F2;
F3;
End
```

Función de inicialización de la interfaz

```
function startupFcn(app)
evalin('base', 'clear ALL')
app.SELECTORDEMODOSwitch.Enable='On';
app.SELECTORDEMODOSwitch.Value='Manual';
app.MODOMANUALACTIVOLamp.Enable='On';
app.MODOMANUALACTIVOLamp.Color='Green';
app.MODOAUTOMATICOACTIVOLamp.Enable='Off';
app.MODOAUTOMATICOACTIVOLamp.Color='Red';
app.ENTRADACRISTALButton.Enable='On';
app.ENTRADALCDButton.Enable='On';
app.TVMODELO1Button.Enable='On';
app.TVMODELO2Button.Enable='On';
app.Reset_CristalButton.Enable='Off';
app.Reset_LCDButton.Enable='Off';
```

```
app.Reset_TVButton.Enable='Off';
app.VELOCIDADSlider.Enable='On';
app.VELOCIDADSlider.Value=100;
app.VELOCIDADField.Enable='On';
app.VELOCIDADField.Value=100;
app.Seta_Emergencia.Enable='On';
app.REARMEDELAESTACINButton.Enable='Off';
app.F0Button.Enable='Off';
app.F1Button.Enable='On';
app.F2Button.Enable='On';
app.F3Button.Enable='On';
opcreset;
app.Cliente_HMI_Proceso = opcda('localhost','Kepware.KEPServerEX.V6')
connect(app.Cliente_HMI_Proceso)
app.Proceso = addgroup(app.Cliente_HMI_Proceso);
Cinta_Manual = additem(app.Proceso,'SIEMENS.S7-1200.Cinta Manual Entrada');
write(Cinta_Manual,1);
app.F0 = additem(app.Proceso,'SIEMENS.S7-1200.F0');
write(app.F0,1);
app.Automatico = additem(app.Proceso,'SIEMENS.S7-1200.Automatico');
write(app.Automatico,0);
app.Reset_Automatico = additem(app.Proceso,'SIEMENS.S7-1200.Reset Automatico');
write(app.Reset_Automatico,0);
app.Avance = additem(app.Proceso,'SIEMENS.S7-1200.Avance Cinta');
write(app.Avance,0);
app.Retroceso = additem(app.Proceso,'SIEMENS.S7-1200.Retroceso Cinta');
write(app.Retroceso,0);
app.Entrada_Cristal = additem(app.Proceso,'SIEMENS.S7-1200.Avance Cristal');
write(app.Entrada_Cristal,0);
app.Entrada_LCD = additem(app.Proceso,'SIEMENS.S7-1200.Avance LCD');
write(app.Entrada_LCD,0);
app.TV_Modelo_1 = additem(app.Proceso,'SIEMENS.S7-1200.TV Modelo 1');
write(app.TV_Modelo_1,0);
app.TV_Modelo_2 = additem(app.Proceso,'SIEMENS.S7-1200.TV Modelo 2');
write(app.TV_Modelo_2,0);
app.Emergencia = additem(app.Proceso,'SIEMENS.S7-1200.Emergencia');
write(app.Emergencia,0);
app.Rearme = additem(app.Proceso,'SIEMENS.S7-1200.Rearme');
write(app.Rearme,0);
app.Velocidad = additem(app.Proceso,'SIEMENS.S7-1200.Velocidad_HMI');
write(app.Velocidad,100);
end
```

Selector de modo de funcionamiento

```
function SELECTORDEMODOSwitchValueChanged(app, event)
value = app.SELECTORDEMODOSwitch.Value;
switch value
case 'Manual'
write(app.Automatico,0);
write(app.Reset_Automatico,0);
```

```
app.MODOMANUALACTIVOLamp.Enable='On';
app.MODOMANUALACTIVOLamp.Color='Green';
app.MODOAUTOMATICOACTIVOLamp.Enable='Off';
app.MODOAUTOMATICOACTIVOLamp.Color='Red';
app.ENTRADACRISTALButton.Enable='On';
app.ENTRADALCDButton.Enable='On';
app.TVMODELO1Button.Enable='On';
app.TVMODELO2Button.Enable='On';
app.Reset_CristalButton.Enable='Off';
app.Reset_LCDButton.Enable='Off';
app.Reset_TVButton.Enable='Off';
app.F1Button.Enable='On';
app.F2Button.Enable='On';
app.F3Button.Enable='On';
write(app.Avance,0);
write(app.Retroceso,0);
write(app.Entrada_Cristal,0);
write(app.Entrada_LCD,0);
write(app.TV_Modelo_1,0);
write(app.TV_Modelo_2,0);
case 'Automatico'
write(app.Automatico,1);
write(app.Reset_Automatico,1);
app.MODOMANUALACTIVOLamp.Enable='Off';
app.MODOMANUALACTIVOLamp.Color='Red';
app.MODOAUTOMATICOACTIVOLamp.Enable='On';
app.MODOAUTOMATICOACTIVOLamp.Color='Green';
app.ENTRADACRISTALButton.Enable='Off';
app.ENTRADALCDButton.Enable='Off';
app.TVMODELO1Button.Enable='Off';
app.TVMODELO2Button.Enable='Off';
app.Reset_CristalButton.Enable='Off';
app.Reset_LCDButton.Enable='Off';
app.Reset_TVButton.Enable='Off';
app.F1Button.Enable='Off';
app.F2Button.Enable='Off';
app.F3Button.Enable='Off';
write(app.Avance,0);
write(app.Retroceso,0);
write(app.Entrada_Cristal,0);
write(app.Entrada_LCD,0);
write(app.TV_Modelo_1,0);
write(app.TV_Modelo_2,0);
end
end
```

Botón de entrada de paneles de cristal

```
function ENTRADACRISTALButtonPushed(app, event)
write(app.Entrada_Cristal,1);
app.ENTRADACRISTALButton.Enable='Off';
```

```
app.Reset_CristalButton.Enable='On';
end
```

Botón de reseteo de entrada de paneles de cristal

```
function Reset_CristalButtonPushed(app, event)
write(app.Entrada_Cristal,0);
app.ENTRADACRISTALButton.Enable='On';
app.Reset_CristalButton.Enable='Off';
end
```

Botones de avance y retroceso del soporte móvil

```
function AVANCEButtonValueChanged(app, event)
value = app.AVANCEButton.Value;
write(app.Avance,value);
end

function RETROCESOButtonValueChanged(app, event)
value = app.RETROCESOButton.Value;
write(app.Retroceso,value);
end
```

Interfaz control de robots

Función de inicialización de la interfaz

```
function startupFcn(app)
evalin('base', 'clear ALL')
app.PAUSAButton.Enable='On';
app.RESETFALLOSButton.Enable='On';
app.STOPButton.Enable='On';
app.CONTINUAButton.Enable='Off';
app.MOVIMIENTOAHOMEButton.Enable='On';
app.MANIPULACHAPABButton.Enable='On';
app.MANIPULACRISTALButton.Enable='On';
app.MANIPULALCDButton.Enable='On';
app.F0Button.Enable='On';
app.F1Button.Enable='Off';
app.F2Button.Enable='On';
app.F3Button.Enable='On';
app.CNT=0;
app.Cliente_HMI_R1 = opcda('localhost','Kepware.KEPServerEX.V6')
connect(app.Cliente_HMI_R1)
app.Variables_R1 = addgroup(app.Cliente_HMI_R1);
app.F0 = additem(app.Variables_R1,'SIEMENS.S7-1200.F0');
app.F1 = additem(app.Variables_R1,'SIEMENS.S7-1200.F1');
app.F2 = additem(app.Variables_R1,'SIEMENS.S7-1200.F2');
app.F3 = additem(app.Variables_R1,'SIEMENS.S7-1200.F3');
```

```
write(app.F0,0);
write(app.F1,1);
write(app.F2,0);
write(app.F3,0);
app.PAUSA = additem(app.Variables_R1,'SIEMENS.S7-1200.PAUSA_HMI');
app.START = additem(app.Variables_R1,'SIEMENS.S7-1200.INICIO_HMI');
app.RESUME = additem(app.Variables_R1,'SIEMENS.S7-1200.CONTINUA_HMI');
app.STOP = additem(app.Variables_R1,'SIEMENS.S7-1200.STOP_HMI');
app.RESET = additem(app.Variables_R1,'SIEMENS.S7-1200.RESET_HMI');
app.HOME = additem(app.Variables_R1,'SIEMENS.S7-1200.Boton_HOME_R1');
write(app.PAUSA,0);
write(app.START,0);
write(app.RESUME,0);
write(app.STOP,0);
write(app.RESET,0);
write(app.HOME,0);
app.Cristal = additem(app.Variables_R1,'SIEMENS.S7-1200.Boton_Cristal_Manual');
app.LCD = additem(app.Variables_R1,'SIEMENS.S7-1200.Boton_LCD_Manual');
app.Chapa = additem(app.Variables_R1,'SIEMENS.S7-1200.Boton_Chapa_Manual');
write(app.Cristal,0);
write(app.LCD,0);
write(app.Chapa,0);
end
```

Botón de pausa del movimiento

```
function PAUSAButtonPushed(app, event)
app.CONTINUAButton.Enable='On';
app.PAUSAButton.Enable='Off';
write(app.PAUSA,1);
end
```

Botón de para de ejecución del programa

```
function STOPButtonPushed(app, event)
app.CONTINUAButton.Enable='On';
app.MOVIMIENTOAHOMEButton.Enable='Off';
app.MANIPULACHAPAButton.Enable='Off';
app.MANIPULACRISTALButton.Enable='Off';
app.MANIPULALCDButton.Enable='Off';
app.STOPButton.Enable='Off';
app.PAUSAButton.Enable='Off';
write(app.START,0);
write(app.PAUSA,1);
write(app.PAUSA,0);
write(app.STOP,1);
pause(0.5);
write(app.STOP,0);
end
```

Botón de continuar el movimiento

```
function CONTINUAButtonPushed(app, event)
valor_PAUSA=read(app.PAUSA);
if valor_PAUSA.Value==1
app.PAUSAButton.Enable='On';
write(app.PAUSA,0);
write(app.RESUME,1);
pause(0.5);
write(app.RESUME,0);
else
app.MOVIMIENTOAHOMEButton.Enable='On';
app.MANIPULACHAPAButton.Enable='On';
app.MANIPULACRISTALButton.Enable='On';
app.MANIPULALCDButton.Enable='On';
app.STOPButton.Enable='On';
app.PAUSAButton.Enable='On';
end
end
```

Botón de tarea: movimiento a HOME

```
function MOVIMIENTOAHOMEButtonValueChanged(app, event)
value = app.MOVIMIENTOAHOMEButton.Value;
write(app.Cristal,0);
write(app.LCD,0);
write(app.Chapa,0);
write(app.HOME,value);
switch value
case 0
app.MANIPULACHAPAButton.Enable='On';
app.MANIPULACRISTALButton.Enable='On';
app.MANIPULALCDButton.Enable='On';
case 1
app.MANIPULACHAPAButton.Enable='Off';
app.MANIPULACRISTALButton.Enable='Off';
app.MANIPULALCDButton.Enable='Off';
write(app.START,1);
pause(0.5);
write(app.START,0);
end
end
```