Universidad Zaragoza
1542

Proyecto Fin de Carrera

Integración de los componentes y sistemas de percepción de una célula de fabricación robotizada

José Ignacio Abad Martínez

Directores:

Gonzalo López-Nicolás

José Jesús Guerrero Campo

Ingeniería Industrial

Automatización Industrial y Robótica

Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

Abril de 2014

Integración de los componentes y sistemas de percepción de una célula de fabricación robotizada.

RESUMEN.

El presente proyecto describe el trabajo desarrollado para realizar la integración de los diferentes componentes que forman parte de una célula industrial robotizada disponible en el laboratorio L0.6 del edificio Ada Byron de la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza. Además de una puesta a punto, también se busca proporcionar funcionalidad adicional (tanto operativa como docente) a la configuración actual de dicha célula mediante el uso de visión por computador y de simuladores comerciales de robótica industrial.

El trabajo realizado tiene como denominador común el empleo de gran cantidad de documentación técnica referente a los componentes de la célula de fabricación, la implementación de soluciones funcionales y operativas en el momento actual, y la documentación exhaustiva de todo el trabajo realizado. Fundamentalmente, el proyecto se ha desarrollado a lo largo de los siguientes bloques temáticos:

1. Integración de los diferentes componentes de la célula industrial mencionada. Los aspectos más relevantes, entre muchos otros, que se han trabajado han sido: La programación de un robot industrial modelo «Fanuc Arcmate 50 IL»; El desarrollo del sistema de comunicaciones entre dicho robot y un autómata industrial modelo «Modicon M340»; El control de los diversos automatismos mecánicos mediante la programación de dos autómatas adicionales; El establecimiento de comunicaciones y jerarquías de control entre los diferentes autómatas; Y el desarrollo de aplicaciones de control mediante terminales visuales Hombre-Máquina, concretamente sobre plataformas «Magelis» presentes en el laboratorio.

2. Ampliación de la funcionalidad tradicional de las células de fabricación mediante el empleo de algoritmos de visión para cámaras RGB-D. Para ello se han desarrollado algoritmos en lenguaje de programación C++, para uso con cámaras RGB-D que permiten la localización de objetos predefinidos, obteniendo del algoritmo información tal como el tipo de objeto localizado, su posición, orientación y color. Adicionalmente, los algoritmos también permiten realizar un control de seguridad sobre la presencia de objetos extraños en el espacio de trabajo.

3. A un nivel superior en el diseño de una célula de fabricación, se encuentra la fase de simulación. También se ha abordado esta fase y se ha realizado un estudio y análisis de los diferentes simuladores comerciales de robótica industrial disponibles en el mercado. Basándose en el análisis y comparación de opciones realizado, se seleccionó el simulador «ABB Robot-Studio». Adicionalmente, se han realizado diversos programas demostrativos, así como una guía sobre el programa, varios vídeos tutoriales y varias propuestas de guiones de prácticas sobre dicho programa.

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación

La automatización industrial es una rama de la ingeniería que si bien se encuentra ya muy desarrollada e implantada en gran parte de las industrias de los países desarrollados, todavía se encuentra en un proceso de mejora, perfeccionamiento y aumento de sus capacidades.

Dentro de la automatización industrial, uno de los aspectos más importantes radica actualmente en la integración de los diferentes componentes automatizados de una planta de producción. Esta integración es muy importante para el funcionamiento de la planta como conjunto, y es frecuentemente el punto donde más dificultades se encuentran.

El contexto en el que se desarrolla este proyecto parte de la existencia en los laboratorios de la Escuela de Ingeniería y Arquitectura, de una célula de fabricación industrial, con todos los elementos comunes a una industria de manufactura. Más adelante se detallarán sus componentes, pero a modo de ilustración, dispone de: Autómatas programables, un robot industrial, buses de comunicaciones, estaciones de trabajo, etc.

En este contexto, la motivación de este proyecto tiene dos claras vertientes. Por un lado, el interés del proyectando tanto en la robótica como en la automatización, que hacía que este proyecto quedara perfectamente integrado en su formación.
Por otro lado, el interés en generar una nueva configuración y programación integrada de la célula de fabricación presente en el laboratorio L 0.6 del edificio Ada Byron, con el objeto de poder servir como soporte para el desarrollo de futuras prácticas para nuevas asignaturas de Grado.

Sin salir del campo más práctico, y como mejora directa del potencial que ofrecen las células automatizadas de fabricación, se hizo el planteamiento de realizar algunos algoritmos para la utilización de una cámara RGB-D. Con ello se busca indagar en el interés que pueden tener estas cámaras como complemento a los sistemas de percepción más tradicionales de las células de fabricación, dado que ya han tenido resultados muy positivos en campos como el entretenimiento.

En estrecha unión con la parte más práctica y aplicada del proyecto, se propuso una más teórica consistente en la selección e implantación de un simulador de robótica industrial. El interés de esta parte se centra fundamentalmente en la necesidad docente de poder enseñar un simulador completo de robótica. Un simulador que incluya no solo los tradicionales aspectos de cinemática de un robot de seis ejes, sino también la posibilidad de interactuar con un entorno industrial complejo. Una vez seleccionado un simulador, también era necesaria la elaboración de documentación y simulaciones de prueba.

1.2. Objetivos y alcance

El objetivo principal de este proyecto es la puesta en funcionamiento, de un modo completamente operativo, de una nueva configuración para la célula de fabricación industrial mencionada, de tal forma que sirva mejor a los más recientes objetivos docentes. Incluyendo así mismo ampliaciones de sus funcionalidades mediante el empleo de la visión por computador, y el uso de un simulador comercial de robótica industrial en un contexto docente.

En particular, los objetivos iniciales de este proyecto Fin de Carrera son:

- Conseguir una integración segura y robusta de los diferentes componentes de la estación del robot, la estación de transporte y la estación de expedición de palets de la célula de fabricación antes mencionada.
- Programación del robot, de los autómatas, y de otros componentes que pudieran ser necesarios.
- Desarrollo de un programa demostrativo que permitiera mostrar el potencial de la célula.
- Desarrollo de algoritmos para cámaras RGB-D que aportaran valor a la supervisión y al control de una célula de fabricación.
- Selección y estudio de un programa de simulación de robots que permitiera la simulación completa tanto de un robot industrial en su aspecto cinemático, como de un entorno completo de trabajo con el que pudiera interactuar.
- Realización de manuales de uso y de utilización tanto de la parte empleada de la célula de fabricación como del simulador a buscar.
- Documentación de todo el trabajo desarrollado, así como de los manuales realizados.

Respecto al alcance final del proyecto, que se irá desgranando a lo largo de los diferentes capítulos, se destacan los siguientes hitos:

- Se ha efectuado un desarrollo a todos los niveles para lograr la correcta integración de todos los componentes propuestos de la célula de fabricación mencionada. En concreto, se han integrado para permitir su funcionamiento como un conjunto:

  - El robot industrial.
  - El autómata asociado al robot.
  - La estación de expedición de palets de la célula de fabricación.
  - La estación de transporte de la célula de fabricación.

- Se han programado tanto el robot industrial como los autómatas programables, de tal manera que puedan funcionar de manera individual, y también como un conjunto integrado.

- Se ha desarrollado un programa (uno por cada componente programable) que permite ejecutar una demostración de las capacidades de la célula de fabricación mediante la integración en un conjunto de sus componentes.

- Se han implementado dos aplicaciones para control y supervisión de la célula. Dichas aplicaciones se utilizan en los terminales «Magelis» [21] , disponibles en el laboratorio L 0.6 del edificio Ada Byron.

- Se han desarrollado diferentes algoritmos para la cámara RGB-D disponible [3]. Dichos programas han sido desarrollados con el objeto de mostrar el posible aumento de las capacidades de una célula de fabricación con el uso del tipo de cámaras mencionado.
  En concreto, se ha desarrollado un programa que supervisa la zona de trabajo del robot y alerta de posibles objetos extraños. Y también otro programa que permite identificar una serie de objetos previamente establecidos, y da información cualitativa y cuantitativa sobre ellos (Tipo de objeto, color básico, tamaño y coordenadas).

- Se han estudiado diferentes simuladores de robótica industrial, realizando una selección en favor de «ABB Robot Studio» [1], ya que era el más completo de todos los analizados,. Adicionalmente, se obtuvieron 100 licencias gratuitas para uso docente por parte de la Universidad.
  Dentro de este hito, hay que añadir que han sido realizadas diferentes pruebas, así como un conciso manual de uso, diversos vídeos tutoriales, y diferentes escenarios para su uso en prácticas.

- Para las tareas realizadas y los programas desarrollados, se han realizado diversos manuales e informes técnicos. Son los siguientes:

  - Manual de configuración de un módulo «ADVANTYS» para su funcionamiento con un autómata «MODICON»
  - Guía de utilización de entradas y salidas mediante red Ethernet para su uso con un autómata «M340»
  - Manual de iniciación a «RobotStudio»

- Respecto a la documentación del trabajo realizado, además de los manuales, se encuentra todo el contenido de esta memoria y sus apéndices. Así como todos los programas debidamente comentados.

1.3. Herramientas y entorno de trabajo

El entorno de trabajo principal se ha localizado en el laboratorio L0.6 del edificio Ada Byron. La maquinaria y principales elementos físicos disponibles son:

- Robot industrial modelo «Fanuc Arcmate 50IL».
- Autómatas programables con conexión Ethernet modelo «Modicon M340».
- Terminales de diálogo táctiles «Magelis».
- Módulos de entradas y salidas distribuidos mediante bus de datos «CANopen».
- Cilindros neumáticos, electro-válvulas, cintas transportadoras, etc.

Aparte de los elementos físicos disponibles en el laboratorio L 0.6, cabe destacar la utilización de:

- Lenguaje de programación de robots «KAREL»
- Software de programación de autómatas «UNITY», mediante los lenguajes «SFC» y «ST» (Sequential Function Chart, y Structured Text)
- Software de programación de terminales de diálogo «Vijeo Designer»
- Para la cámara RGB-D. Lenguaje de programación C++ sobre entorno Windows mediante «MS Visual Studio»
- En concreto, los programas de la cámara se han realizado en base a las librerías PCL (Point Cloud Library) [17].
- Para la comunicación entre la cámara y el computador, se han utilizado las librerías OpenNI [14].
- Simulador de robótica industrial «ABB Robot Studio», incluyendo su lenguaje de programación «Rapid».

1.4. Organización de la memoria

Después de esta visión general de lo que es este proyecto, se va a explicar brevemente la forma en la que está estructurada esta memoria.

En el capítulo 2, Situación previa de la célula industrial, se explicará cómo estaban los equipos utilizados en el momento de plantear este proyecto.

En el capítulo 3, Programación del robot industrial, se abordarán los métodos que se han utilizado para programar dicho robot, algunos detalles del robot, y los programas resultantes de la programación.

El capítulo 4, Control del robot con autómatas, versa sobre la necesidad de dicho control, la manera de realizarlo, cómo se efectúa la comunicación, y cómo se logra la integración del conjunto.

Los capítulos 5 y 6 tratan sobre el porqué de introducir dos elementos tales como los terminales de supervisión y las cámaras RGB-D. El qué aportan, cómo se programan, y qué pueden ofrecer. Además, para la cámara RGB-D, también se detalla la forma en la que se logra la detección de elementos, y su identificación cuando proceda.

El capítulo 7 es un capítulo centrado en un nivel superior de planificación, diseño y simulación, y que contiene información sobre los simuladores analizados, el simulador seleccionado y sobre los tutoriales y escenarios realizados.

Por último, el capítulo 8 recoge las conclusiones del proyecto.

Como uno de los objetivos del proyecto era la obtención de nuevo material docente para las nuevas asignaturas de Grado relacionadas con la temática del mismo, en los capítulos de los que se ha obtenido alguna clase de este material, aparece una sección dedicada a él.

1.5. Gráfico resumen del proyecto

Para ilustrar gráficamente los diferentes aspectos abordados durante este proyecto, la relación existente entre ellos y su integración en un único conjunto, se presenta el diagrama de la figura 1.1 a modo de resumen.
En este gráfico se esquematizan los intercambios de información entre los diferentes componentes del proyecto, y se trata de representar el nivel de relación existente entre los mismos.

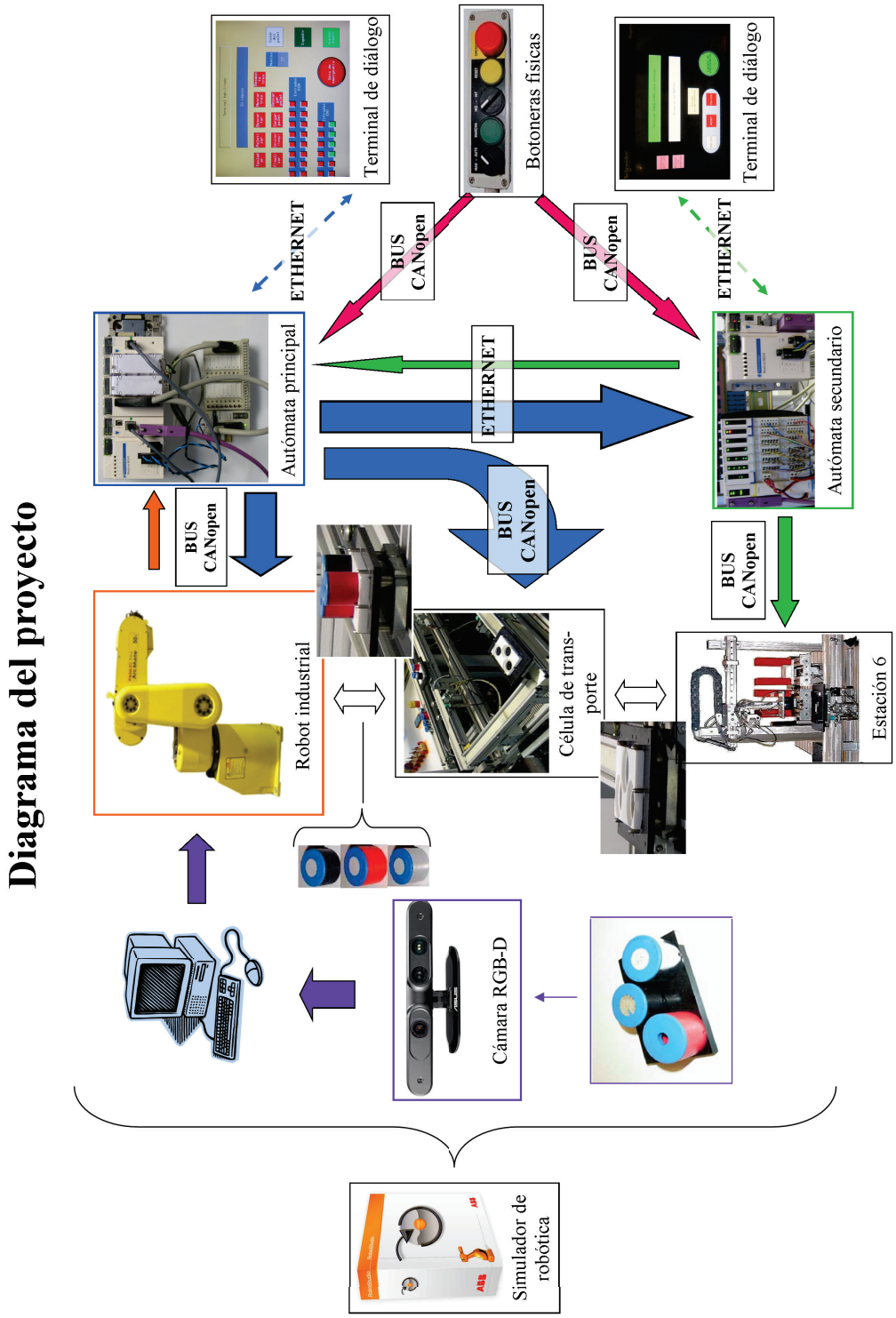

Figura 1.1: Gráfico resumen del proyecto.

Capítulo 2

Situación previa de la célula industrial

En este capítulo se comenta el estado inicial de los equipos y se describen de forma general, indicando sus carencias. Con ello se pretende facilitar la comprensión de los pasos seguidos a lo largo del proyecto algunos de los cuales fueron condicionados por el estado inicial de los equipos. Además, también se da una ligera descripción de los mismos, con especial interés en la célula de fabricación robotizada sobre la que se ha trabajado, la cual es comentada con algo más de detalle.

2.1. Célula de fabricación industrial

La célula de fabricación industrial existente en el laboratorio L 0.6 del edificio Ada Byron, consta en esencia de siete estaciones de trabajo y almacenamiento, y dos estaciones de transporte. Además también posee una estación conteniendo un robot industrial modelo FANUC ARCMATE 50IL.

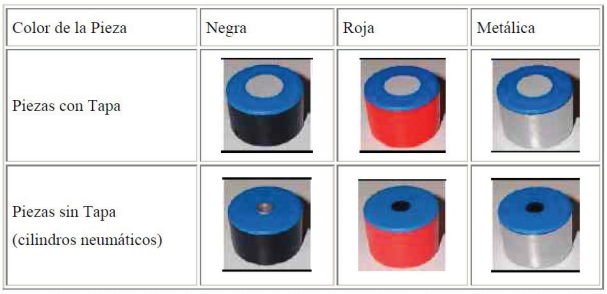

Figura 2.1: Tipos de piezas que se producen en la célula industrial.

El objetivo de esta instalación que simula una fábrica, es la manufactura de unos pequeños cilindros que están formados por una camisa, una tapa, y opcionalmente un émbolo y un muelle.
Particularmente, las estaciones de trabajo son las que a continuación se describen. (Más

adelante se darán más detalles de las que se han usado en este proyecto). Las primeras estaciones corresponden a lo que sería la sub-célula de fabricación, en el sentido de que las piezas se «fabrican» en esta parte. Las restantes estaciones consisten en la parte de la célula dedicada a la preparación de pedidos y transporte.

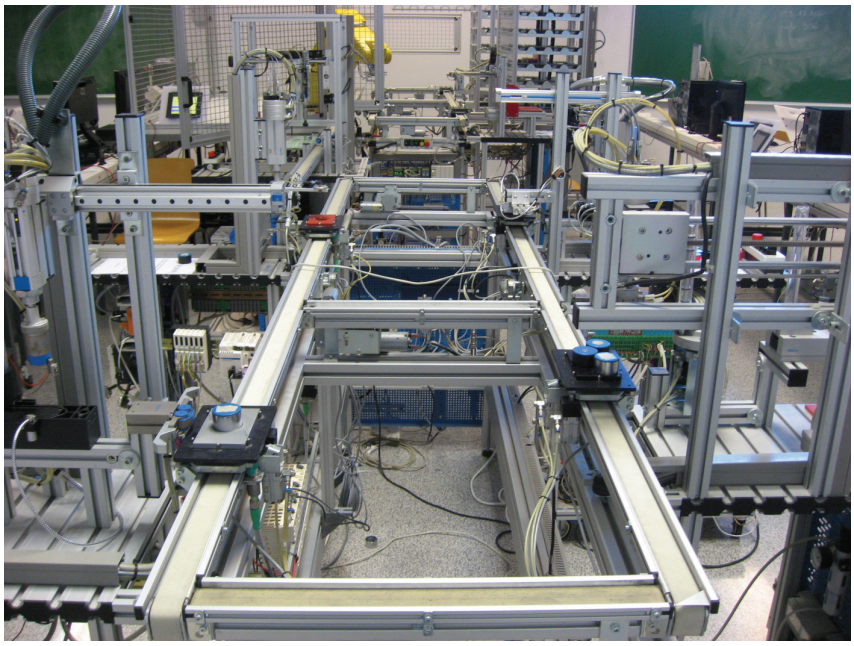

Figura 2.2: Vista general de la célula completa. Se aprecian diferentes estaciones de trabajo, cintas transportadoras, y al fondo el robot industrial

Estaciones de la parte de la célula dedicada a la manufactura de las piezas:

- Transporte 1. Consistente en una serie de cintas transportadoras, desvíos y carros, se encarga de mover las piezas de una estación de trabajo a otra.

- Estación 1. Esta estación se encarga de colocar en un carro de transporte las camisas del cilindro. Dichas camisas están contenidas en un alimentador, y la pieza se coloca en el carrito mediante la combinación de un empujador y un brazo movido por dos cilindros neumáticos.

- Estación 2. Coloca un émbolo y un muelle dependiendo del tipo de cilindro que le llegue. Para ello la estación se sirve de un brazo con una pequeña garra y un motor paso a paso. Mediante dos operaciones se colocan el émbolo y el muelle.

- Estación 3. Encaja una tapa en las piezas que le llegan. Las tapas se colocan enroscadas en las camisas mediante un alimentador que contiene las tapas, un empujador que las saca y brazo con una garra y movido por una pareja de cilindros neumáticos.

- Estación 4. Verifica si el cilindro fabricado funciona correctamente. Para ello coge las piezas mediante un brazo con una ventosa, las introduce en un cilindro de comprobación, les inyecta aire comprimido, mide la salida del émbolo, y si es satisfactorio las traslada de cinta. Si no son adecuadas, las desecha.

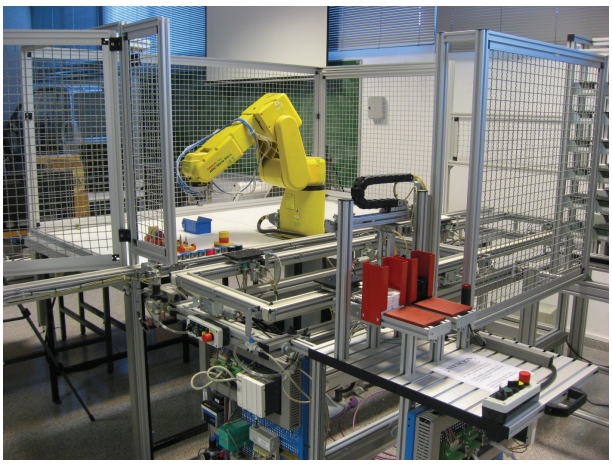

Figura 2.3: Subcélula de preparación y transporte de pedidos

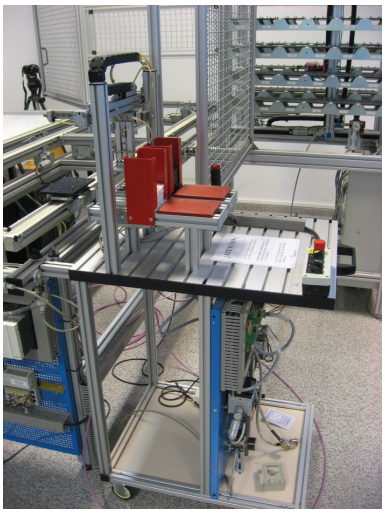

Figura 2.4: Estación expendedora de palets para colocar las piezas.

- Estación 5. Es un almacén intermedio, que es capaz de distinguir las piezas que le llegan y almacenarlas hasta que son solicitadas. Está compuesto de dos motores paso a paso y un cilindro neumático que guían a un brazo con ventosa a lo largo de un almacenamiento de tipo matricial.

A partir de aquí, se pueden considerar las siguientes estaciones como pertenecientes a la sub-célula de gestión de pedidos, que es a la que se refiere este proyecto:

- Transporte 2. Se encarga de transportar los carritos en los que se sitúan los pedidos a lo largo de la sub-célula de gestión de pedidos. Consta de seis cintas transportadoras, un desvío intermedio, diversos sensores de detección (inductivos y capacitivos) y cuatro carritos transportadores. Los diversos carritos pueden circular entre las diferentes estaciones, existiendo desvíos para que no sea necesario que pasen por todas para llegar a una determinada.

- Estación 6. Esta estación coloca una placa con capacidad para tres cilindros. La placa puede ser blanca o negra, lo que aumenta las variantes del pedido. Su funcionamiento corre a cargo de diversos cilindros de simple y doble efecto, así como de sensores de posición para los cilindros, y de detección de placas. En esencia funciona del siguiente modo: las placas están colocadas en alimentadores; Se extrae una de las placas de un alimentador mediante un empujador; Una vez detectado que se ha sacado la pieza, se mueve el brazo con ventosas hasta la pieza y se succiona para agarrarla; La pieza se coloca sobre el carrito, y se devuelve el brazo a su posición de reposo.
  Esta estación dispone además de una luz de emergencia.
  También dispone de una botonera, un autómata con módulo CANopen, y diversos módulos de cableado rápido.

- Estación 7. Es el almacén final de pedidos, posee cierta complejidad debido a sus complicados mecanismos de posicionamiento y a la existencia de más de 70 lugares

de almacenamiento. En el momento actual está en desuso, y el autómata que la controla es un modelo antiguo.

En el momento de comenzar el proyecto, todos los elementos descritos hasta el momento estaban parcialmente operativos desde el punto de vista de su funcionamiento mecánico. Adicionalmente, se descartó el uso de la sub-célula de fabricación debido a ciertas carencias respecto a las comunicaciones de algunos de los autómatas que la debían controlar, todo ello se detallará en el apartado correspondiente a los autómatas.
Hay que añadir que dos de las cintas transportadoras se encontraban rotas al comienzo del proyecto, faltaban algunos módulos de comunicaciones, había modelos de autómatas incompatibles entre sí, y existían algunos problemas de incompatibilidad entre el software más moderno de programación de autómatas (Unity Pro), y los autómatas más antiguos. A continuación se detalla el estado inicial de los elementos involucrados más importantes.

2.2. Robot industrial

Una de las partes fundamentales de la sub-célula de gestión de pedidos, incluso quizá la más importante debido a su versatilidad, es el robot industrial presente en la misma.
Se trata de un robot industrial de seis ejes, modelo FANUC ARCMATE 50IL (Más información en [15]). Sus principales características se recogen a continuación:

Nº de ejes	6
Carga máxima	3 kg
Repetibilidad	0.04mm
Peso del robot	41 kg
Distancia alcanzable (Horizontal)	856 mm
Velocidades (Ejes 1 a 3)	1.57 rad/s
Velocidades (Eje 4)	6.98 rad/s
Velocidades (Eje 5)	5.76 rad/s
Velocidades (Eje 6)	8.38 rad/s

Tabla 2.1: Principales características del robot

Este robot está equipado con una garra especialmente diseñada para poder coger los pequeños cilindros que se fabrican en la otra parte de la célula, pero también puede coger otra serie de objetos tales como prismas, esferas, cilindros más grandes, etc. En su momento, el conjunto del robot y la sub-célula de expedición de pedidos, fueron diseñados para que el robot pudiera acceder a la cinta por la que llegan las piezas y a la cinta por la que llegan los carros. En el caso de la cinta de los carros, tiene alcance para coger o colocar piezas en las dos posiciones fijas que pueden adoptar los carros.
Además, también tiene alcance para poder trabajar en una gran superficie que se sitúa al otro lado de las cintas transportadoras, y en la cual se pueden desarrollar múltiples operaciones tal y como se verá en el capítulo dedicado al robot.

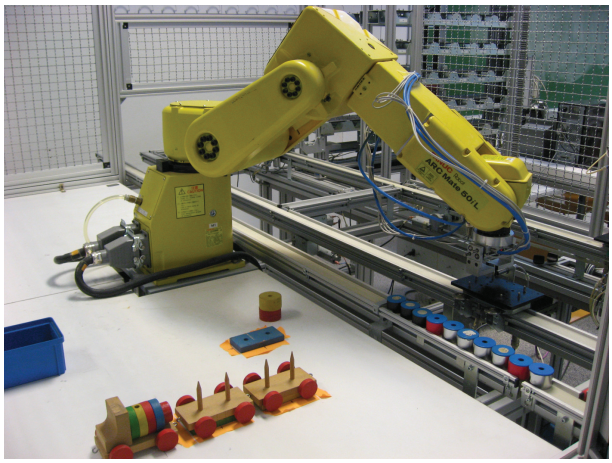

Figura 2.5: Zona de trabajo del robot. Está delimitada por las cintas transportadoras y tres vallas de protección.

Como se ve en la figura 2.5, se aprecian tanto las dos cintas transportadoras a las que puede acceder, como una de las posiciones fijas para los carros. La zona de la izquierda es en la que se ha comentado que se pueden realizar tareas adicionales no necesariamente relacionadas con los cilindros. El estado del robot en el momento de comenzar el proyecto es de plena operatividad, aunque de una forma aislada respecto del resto de la célula de fabricación. Tanto el robot como su controlador estaban en perfectas condiciones de uso. Además, el robot tenía una serie de programas demostrativos que han sido muy útiles para la posterior programación del mismo.

Respecto a la comunicación del robot con otros elementos, se realizaba mediante un módulo de entradas y salidas por red Ethernet, denominado TSX MOMENTUM. El robot tenía cableadas una serie de entradas y salidas digitales a dicho módulo de comunicaciones, el cual las ponía a disposición de la red Ethernet del laboratorio. Pero desde un primer momento se comprobó que esa conexión había dejado de ser operativa, por lo cual (entre otras razones) se terminaría por sustituir ese módulo por otro diferente, como más adelante se abordará.

El robot se puede programar de dos formas diferentes. O bien mediante el terminal de diálogo, o bien mediante lenguaje de alto nivel «KAREL». La forma elegida fue el lenguaje de alto nivel, por su mayor versatilidad y por el hecho de que la programación es mucho más legible al hacerse en un lenguaje de alto nivel que si se hubiera hecho mediante el terminal de diálogo, el cual directamente genera archivos binarios.

Hay que añadir, que el software de comunicación con el robot desde el PC asociado, se encontraba disponible parcialmente. Si bien el compilador de lenguaje KAREL, y el emulador de disquetera virtual funcionaban, otros programas originales del fabricante del robot no lo hacían.

2.3. Autómatas

Cuando se comenzó el proyecto, la célula de fabricación disponía de dos tipos de autómatas. Por un lado existían los más modernos M340 (Más información sobre M340 en [18]), del fabricante Schneider Electric, y por otro lado los algo más antiguos TSX Premium (Información adicional sobre TSX PREMIUM en [19]), del mismo fabricante. Las estaciones números 1, 3, 4 y 6 estaban equipadas con los M340. Mientras que las estaciones 2, 5 y 7 lo estaban con los TSX Premium.

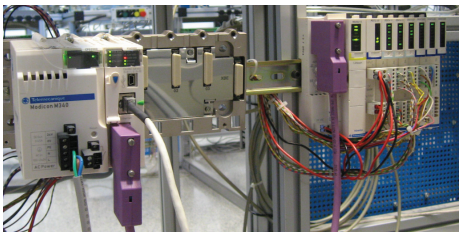

Figura 2.6: Autómata M340 e isla Advantys pertenecientes a la estación 6.

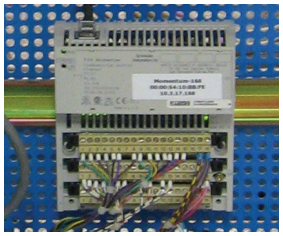

Figura 2.7: Módulo TSX Momentum, perteneciente a una estación de transporte

Debido a problemas de comunicaciones por incompatibilidad de módulos entre los dos tipos de autómata, y a la posibilidad de que trabajar con la célula completa pudiera ser excesivo, se decidió trabajar con la estación 6, las cintas transportadoras de la sub-célula de pedidos, y el robot industrial.

El modo en el que los autómatas controlan las partes de la célula es el siguiente:

- Cada estación tiene todos sus sensores y actuadores cableados físicamente con un módulo de entradas y salidas. Dichos módulos, denominados «islas», son del modelo ADVANTYS STB (Más información de ADVANTYS STB en [20]), y son del fabricante Telemecanique, del grupo Schneider Electric.

- Cada uno de estos módulos de entradas y salidas está conectado mediante un bus CANopen al autómata que controla dicha célula.

- El autómata intercambia información con el módulo de entradas y salidas, y mediante la ejecución del programa que tenga cargado, realiza el control efectivo de la estación. Además, también pone a disposición de otros entes de mayor jerarquía la posibilidad de acceder a las entradas y salidas, así como a otras variables propias del autómata, todo mediante la red Ethernet.

Además, por otra parte están las estaciones que no disponen de autómata propio, que son las de transporte. Todos los sensores y actuadores de cada una de estas dos estaciones están conectados a su respectivo módulo de entradas y salidas modelo TSX MOMENTUM, que las vuelca a la red Ethernet, haciéndolas accesibles a cualquier autómata que disponga de un módulo de control Ethernet.

En los siguientes capítulos se desarrollarán los cambios realizados en la instalación de la célula de fabricación, así como la forma en la que se han utilizado y programado los elementos mencionados.

Capítulo 3

Programación del robot industrial

En este capítulo se presentan las tareas robóticas realizadas y se explica de manera concisa cómo se ha programado el robot y por qué se ha programado así, y se resaltan los aspectos más importantes de la programación que se ha realizado.

3.1. Controlador del robot

El robot industrial FANUC ARCMATE 50IL, en concreto en su versión presente en el laboratorio L 0.6, puede estar controlado mediante dos controladores diferentes. El controlador RJ-2 y el RJ-3, ambos del mismo fabricante que el robot. En el caso de este robot, el controlador presente es como muestra la figura 3.1, el RJ-3.
El tipo de controlador tiene importancia, ya que tratándose del mismo robot, los métodos de programación, y los modos de comunicarse con el exterior pueden ser muy diferentes dependiendo del controlador.
Para el controlador especificado, aparte del conocimiento disponible por el personal con el que se ha colaborado, se han consultado los manuales contenidos en la bibliografía [9] y [11].

Sobre este controlador en cuestión, y la manera en la que se instaló, ya han versado otros proyectos y trabajos previos, por lo que no se pretende repetir de nuevo esa información. No obstante sí que se remarcan los siguientes aspectos importantes de él:

- Incluye un terminal de enseñanza, denominado «Teach Pendant», con el cual se puede realizar la programación completa del robot, y del que se hablará más adelante.

- Dispone de tarjetas para las conexiones físicas de entradas y salidas digitales. Durante la realización de este proyecto, se han llevado a cabo algunas modificaciones respecto a las originales (Ver apéndice A). El resultado final es de 8 salidas digitales (DO) y 9 entradas digitales (DI), además de múltiples entradas y salidas de control y de seguridad hasta completar las 16 entradas y 16 salidas posibles (Limitación debida al módulo de entradas y salidas modelo «ADVANTYS STB» al que están cableadas).

- Como se acaba de comentar, dispone de una serie de entradas y salidas digitales de seguridad, aptas para un control robusto. Dichas entradas y salidas no son dependientes del programa en ejecución, sino que dependen únicamente del hardware, por

Figura 3.1: Armario del controlador RJ-3 del robot FANUC, con el terminal de enseñanza en primer plano.

lo que son consideradas como robustas frente a posibles bloqueos del programa. El cableado completo de estas entradas y salidas en su forma actual, se adjunta en el apéndice A.

- También dispone de entradas y salidas digitales propias para la tarea a realizar por el robot. Sirvan como ejemplo las que controlan la garra del mismo.

- Uno de los aspectos más importantes que posee el controlador es la protección contra sobre-esfuerzos. El controlador es capaz de detectar la sobre-intensidad que se provoca en los motores cuando estos intentan hacer un movimiento, y dicho movimiento no se puede realizar debido a un obstáculo, un atasco... En ese momento se corta la corriente, y el controlador entra en un estado de «Falta». Para salir de dicho estado hay que utilizar el terminal de enseñanza o la botonera del controlador. También existe una entrada digital que puede cumplir esa función, aunque en este proyecto no ha llegado a utilizarse por considerarla potencialmente insegura (La potencial inseguridad se debe al hecho de que se cree una acción anti faltas, que libere al robot automáticamente sin comprobar su estado).

- Además de la ya comentada protección contra sobre-esfuerzos, también dispone el controlador de otros elementos de seguridad tales como fusibles y topes mecánicos. No obstante, es necesario ser extremadamente cuidadoso para no provocar la entrada en funcionamiento de uno de estos sistemas de seguridad, ya que su bloqueo es mucho más costoso de solucionar, incluyendo la posible sustitución de piezas.

3.2. Terminal de enseñanza

El terminal de enseñanza es un elemento muy importante del controlador, y extremadamente útil a la hora de programarlo.
Sin entrar en el modo de programación mediante este elemento, que se detalla extensamente en uno de los manuales disponibles [12], se va a dar una breve descripción de sus funciones y cómo se han usado para programar el robot durante este proyecto.

Figura 3.2: Terminal de enseñanza del robot FANUC. Se aprecia el pulsador de emergencia.

Como se puede observar en la figura 3.2, el terminal de enseñanza (en adelante TPE), dispone de múltiples botones, un selector y un pulsador de emergencia. Además, también dispone de dos pulsadores de «hombre muerto» en la parte de atrás.

Las principales funciones del TPE, y más utilizadas durante el proyecto son las siguientes:

- Guiado del robot entre diferentes posiciones. El robot puede ser guiado manualmente mediante el TPE. Este guiado puede realizarse mediante el uso de un sistema de coordenadas globales (Articulares y cartesianas), coordenadas desde el punto de vista de la herramienta y coordenadas definidas por el usuario.
  Las más utilizadas durante el proyecto han sido las globales, tanto en formato articular como en cartesiano.
  La utilidad que ha tenido en este caso el guiado, ha sido fundamentalmente comprobar la alcanzabilidad de ciertas posiciones, comprobar las coordenadas calculadas previamente, y por supuesto la obtención de coordenadas tanto globales como articulares para su posterior programación en lenguaje de alto nivel.

- Otra de las funciones fundamentales que permite es la variación de la velocidad del robot. Este aspecto, aunque parezca trivial, es muy importante, ya que parte de la precisión que puede alcanzar el robot proviene de la velocidad a la que está trabajando.
  Además, a la hora de ejecutar los programas de prueba, el poder variar la velocidad de una forma rápida es en extremo útil, ya que permite realizar a mayor velocidad los tramos de programa ya seguros, y bajar la velocidad para controlar mejor los más novedosos o inseguros.

- No hay que olvidar la activación de señales digitales, que también puede hacerse de manera manual desde el TPE. Este aspecto ha sido muy útil mientras se realizaba la programación de los autómatas, ya que permitía probar entradas y salidas sin tener que ejecutar un programa entero.

- Aunque pueda resultar cuando menos curioso, otro aspecto para el que es necesario el TPE es para solucionar bloqueos y faltas. Es un caso muy frecuente que se intente hacer pasar al robot cerca de una singularidad (Estado cinemático que el controlador no es capaz de resolver), y en ese momento el robot entre en falta. Entonces, la manera más efectiva de desenclavar el robot es utilizar el guiado manual mediante el TPE, y luego ejecutar un programa o instrucción que lo devuelva a su posición de reposo.
  También es necesario realizar esto cuando la falta ha sido provocada por un sobre-esfuerzo o un bloqueo.

3.3. Hyper-terminal

Se ha comentado ya que la programación se ha realizado en lenguaje Karel, el cual se compila en un PC adjunto al robot. Es momento ahora de hablar sobre cómo se han realizado las cargas de programas al robot. Con el objeto de ser breve, no se contará hasta el último detalle de cómo se carga el programa, si bien sí lo más importante. Todos los detalles y el método completo se encuentran en el manual de lenguaje Karel [10].

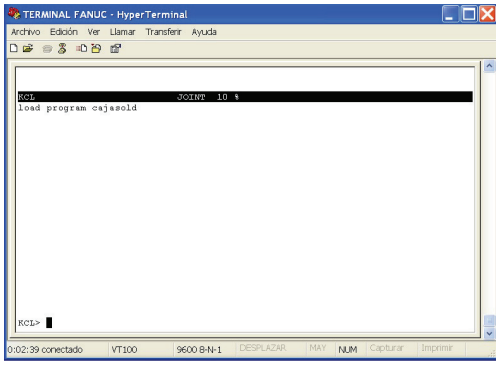

Figura 3.3: Vista del programa de comunicación por puerto serie «Hyper-terminal».

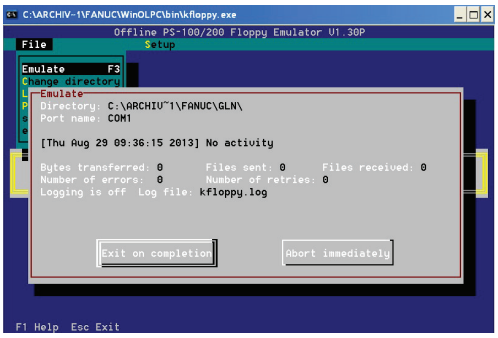

Figura 3.4: Vista del programa «Kfloppy», que emula una disquetera virtual.

En esencia, el robot posee una memoria interna en la que guarda los programas, y a la cual se puede acceder mediante un puerto serie. Para poder utilizar el puerto serie, se dispone del programa «Hyper-terminal», que es el que establece la conexión.
Para poder cargar mediante dicho puerto serie un programa al robot, es necesario emular una disquetera virtual del mismo, lo cual se hace con el programa «kfloppy», incluido dentro de los programas que suministra el fabricante. El programa (que tiene que estar ya compilado y convertido a formato binario), es transmitido al robot mediante una orden dada a través del hyper-terminal. Su ejecución también puede ser solicitada a través del hyper-terminal.

Como resumen de las órdenes que pueden ser dadas a través del hyper-terminal, se pueden considerar las siguientes:

- Cargar un programa.

- Ejecutar un programa.
- Simular una entrada digital.
- Copiar, pegar, cambiar el nombre y eliminar los programas contenidos dentro de la memoria del robot.
- Acciones relativas a la memoria del robot tales como: Restaurarla, repararla, borrarla, o hacer una copia de seguridad.

3.4. Lenguaje Karel

El robot industrial, como ya se comentó, se puede programar de dos maneras diferentes.

- En primer lugar, está la opción de programarlo íntegramente mediante su terminal de enseñanza (Teach Pendant). Este método de programación produce un código ya binario que el robot es perfectamente capaz de leer. Por contra, la programación es extremadamente compleja debido a la dificultad de uso del TPE, y a la necesidad de utilizar cursores para tareas tan sencillas como poner un nombre.
- Totalmente diferente al método anterior se sitúa el lenguaje de alto nivel KAREL. Éste es un lenguaje propio del fabricante de robots FANUC. Permite instrucciones avanzadas, y su programación en cualquier entorno de trabajo, ya que simplemente hay que generar un archivo con formato «.kl», que luego habrá que compilar para producir el fichero binario. Este apartado está dedicado a este lenguaje, y cómo se ha utilizado en este proyecto. Mucha información útil se ha obtenido del manual de dicho lenguaje [10], disponible en el departamento de Informática e Ingeniería de Sistemas.

La forma en la que se ha programado el robot varía dependiendo de la tarea a realizar. En este proyecto, para los programas más sencillos, lo que se ha hecho es que una vez se da la orden de ejecución del programa (mediante el Hyper-terminal), el programa se ejecuta de forma secuencial hasta su final.

Por contrapartida, hay programas más complejos, como por ejemplo el principal de los realizados, en los que el robot una vez está ejecutando el programa, permanece en estado de reposo hasta que una entrada digital le indica qué parte del programa ejecutar. Dispone así de múltiples rutinas o sub-programas que poder ejecutar en función del control que ejerce un ente de jerarquía superior.

Aún existe una tercera forma de programación, que consistiría en la preparación de diversos programas, cuya ejecución se ordena por medio de una de las entradas digitales reservadas. De esta manera el robot no tiene ningún programa cargado, hasta que le llega la susodicha señal, momento en el cual carga el programa y lo ejecuta. Pudiendo durante esta ejecución recibir o dar más órdenes mediante entradas y salidas digitales.

Los detalles sobre formas de programar se pueden consultar en el manual de programación [10]. En el siguiente apartado, se detallan algunas de las principales características

de los programas creados, si bien el programa principal completo está contenido en el apéndice B.
No obstante, tómense las siguientes breves descripciones de algunos de los programas realizados como referencia:

- Programa «plantilla». Define una serie de constantes, instrucciones y una función totalmente necesaria para continuar la programación en lenguaje Karel. Está pensado como programa base para continuar a partir de él, permitiendo un ahorro de tiempo al programador.

- Programa «cajasold». Simula la soldadura de una caja por parte del robot. En concreto imita la colocación de dos puntos de soldadura, y de dos cordones de soldadura por el borde de una caja. Está inspirado y es similar al programa para un robot «PUMA» que se realiza en una práctica de la asignatura «Control y Programación de Robots».

- Programa «repose». Este programa, es tan útil como elemental. Su función es devolver al robot a una posición de reposo, y terminar el programa de una manera limpia. Con él se permite evitar el tener una botonera externa que ejecute la señal digital de ir a la posición de reposo.

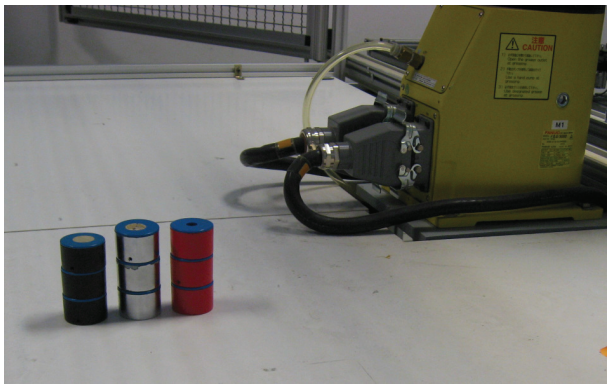

Figura 3.5: Piezas apiladas en tres montones distintos como resultado del programa «Paletizado».

- Programa «paletizado». El robot realiza las acciones de tomar cilindros de la cinta transportadora de suministro, y los coloca en una zona de almacenamiento intermedio. Los cilindros son apilados en tres columnas, mientras el operario no indique lo contrario, mediante el uso de coordenadas relativas.

- Programa «carros». Durante la ejecución de este programa, el robot coge sucesivamente tres piezas de la cinta transportadora de suministro, y los coloca adecuadamente en la placa correspondiente situada encima de un carrito en otra cinta. Además de cargar, también puede ejecutar la opción de descargar en función de la entrada digital que esté activada. Es el primer programa que se realizó en este proyecto que ofrece más de una opción al usuario.

- Programa «integra» (Más detalle en la sección 3.6). Es el programa más complejo de todos los desarrollados, e incluye varios programas propios y extraños [5]. Su funcionamiento es la base de la aplicación que integra toda la sub-célula de expedición de pedidos.
  Su funcionamiento requiere del uso de coordenadas relativas, entradas y salidas digitales, interrupciones y concatenación de trayectorias, así como de múltiples instrucciones simples en lenguaje Karel, pero complejas en su ejecución.

Aparte de estos programas, también se realizaron múltiples programas de prueba, y otros que son suficientemente sencillos para ser comentados, como por ejemplo el que desecha una pieza que llega por la cinta de suministro, o el que la traslada a otra cinta diferente.

3.5. Aspectos relevantes de la programación

Se van a comentar ahora algunos de los aspectos más importantes de la programación realizada. En particular se describen aspectos relativos a manejo de coordenadas de tipo articular, interrupciones, selección entre diferentes tareas a ejecutar, y seguridad relativa a la garra del robot.

Rutina «create_jpos»:

El fragmento de código siguiente, permite la utilización de forma directa de coordenadas articulares al programar en lenguaje Karel. Aunque es un hecho curioso, este lenguaje no dispone de una función propia para esta tarea, que sin embargo es muy utilizada.

```
– Rutina que pone en una variable de tipo JOINTPOS6 «lug_j» ;
– las coordenadas articulares «A,B,C,D,E,F»;
ROUTINE create_jpos(lug_j:JOINTPOS6;A,B,C,D,E,F: REAL)
VAR
vec_pos: ARRAY[6] OF REAL
exito2: INTEGER
BEGIN
vec_pos[1]=A;
vec_pos[2]=B;
vec_pos[3]=C;
vec_pos[4]=D;
vec_pos[5]=E;
vec_pos[6]=F;
CNV_REL_JPOS(vec_pos,lug_j,exito2);
END create_jpos
```

Rutinas que manejan las interrupciones:

El fragmento de código adjunto a continuación, permite la generación de interrupciones, aquí llamado «manejo de condiciones». Con ello lo que se logra es que determinadas variables del programa pasen a tener un nuevo valor dependiendo de ciertas señales digitales de entradas, sin tener que esperar a que el programa llegue a un comando que las actualice. De esta forma se consigue que por muy breves que sean esas señales, queden registradas en una variable y no se pierda esa información.

En esencia, esta selección de código es la que permite al robot realizar una detención del movimiento controlada, y la que le ordena finalizar el programa.

– *Condition handlers*

CONDITION[3]:
WHEN DIN[6] DO
terminar=true
ENDCONDITION;

CONDITION[4]:
WHEN DIN[7] DO
STOP
ENABLE CONDITION[5]
DISABLE CONDITION[4]
ENDCONDITION;

CONDITION[5]:
WHEN NOT DIN[7] DO
RESUME
DISABLE CONDITION[5]
ENABLE CONDITION[4]
ENDCONDITION;

Rutina principal del programa «Integra»:

El bucle siguiente fundamentalmente realiza una de las tareas que tiene definidas como rutina en función de qué entrada digital ha sido activada. En el momento en el que la variable «*terminar*» es cierta, el bucle se acaba.
Aquí se pone de manifiesto, el porqué de utilizar interrupciones para actualizar algunas variables, ya que la variable «*terminar*» debe ser actualizada en cualquier momento, aunque se esté dentro de una sub-rutina.

BEGIN
init_system;
ini_pos;
ENABLE CONDITION[3]; –habilita la posibilidad de apagar el robot
ENABLE CONDITION[4]; –habilita la parada de emergencia si se abre la puerta

– Ejecucion del programa

WHILE NOT terminar **DO**
DOUT[3]=ON;
IF din[1] **THEN**
trasl_cinta;
DELAY 100;
DOUT[2]=OFF;
ENDIF;

IF din[2] **THEN**
paletiza;
DELAY 100;

```
DOUT[2]=OFF;
ENDIF;

IF din[3] THEN
desecha;
DELAY 100;
DOUT[2]=OFF;
ENDIF

IF din[4] THEN
monta_tr;
DELAY 100;
DOUT[2]=OFF;
ENDIF

IF din[5] THEN
desm_tr;
DELAY 100;
DOUT[2]=OFF;
ENDIF

IF din[9] THEN
cargar;
DELAY 100;
DOUT[2]=OFF;
ENDIF

IF din[10] THEN
descargar;
DELAY 100;
DOUT[2]=OFF;
ENDIF

ENDWHILE

$MOTYPE = JOINT;
$TERMTYPE = NODECEL;
$SPEED = max_vel;
MOVE TO P_casa;

END integra
```

Finalmente y a modo de detalle para mostrar que la seguridad se ha tenido siempre en cuenta, se adjunta un pequeño fragmento de código que paraliza el robot en el caso de que se solicite abrir la garra, y esta no se abra, hasta que esté abierta. De esta manera se evita la posibilidad de que el robot tratara de «chafar» un objeto debido al hecho de que la garra debiera estar abierta y no lo estuviera.

```
OPEN HAND 1;
WAIT FOR DIN[8]=OFF;
```

En este caso, la señal 8 está virtualmente cableada (mediante el TPE) con la señal física de apertura de garra.

3.6. Programa principal del robot

A continuación va a describirse el funcionamiento esencial del principal programa realizado en este proyecto para el robot, el programa «integra». El programa «integra» fundamentalmente produce el resultado de que el robot se mantenga en un estado de reposo hasta que se le indica una instrucción desde fuera mediante una señal digital de entrada. Según la señal que le llega, el programa «integra» realiza las siguientes acciones:

- Trasladar pieza. Es un sub-programa muy directo, que coge una pieza de la cinta de suministro, y la traslada a otra cinta diferente. Es un programa muy elemental, compuesto por movimientos articulares y lineales simples.

- Paletizar. Es un programa algo más complejo, que va recogiendo piezas de la cinta de suministro, y las va apilando en tres columnas diferentes. Utiliza movimientos relativos, más complejos que en otros sub-programas, pero habituales en la programación.

- Desechar. Es el sub-programa más elemental de todos. Su misión es descartar a un contenedor aquellas piezas que se consideren que no son adecuadas para su manipulación. Se realiza con movimientos básicos.

- Montar el tren. Como su nombre indica, este sub-programa realiza el montado de las piezas que hay en la mesa sobre el tren de juguete. Utiliza movimientos complejos, así como múltiples almacenamientos de coordenadas de forma relativa.

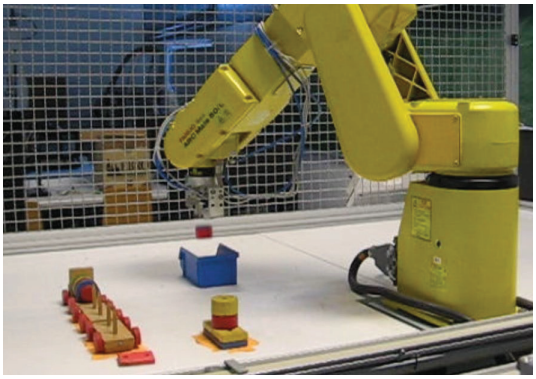

Figura 3.6: Momento en el que el robot descarta una pieza durante el programa «Desechar».

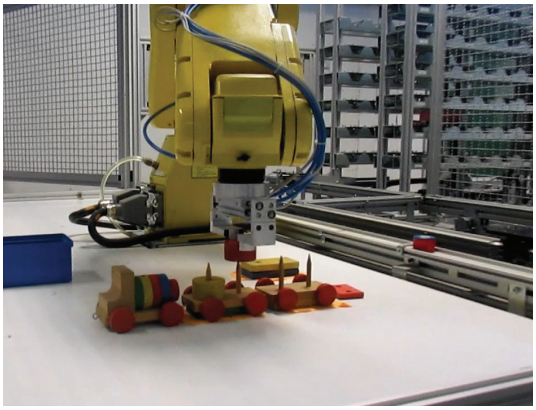

Figura 3.7: Colocación de una de las piezas del tren durante la ejecución del programa «Montar tren».

- Desmontar el tren. Es el programa inverso del anterior. Está programado de la misma manera.

- Detener el robot (temporalmente). Provoca la paralización de los movimientos del robot y de la interpretación del código. No es un sub-programa sino que está programado como una interrupción, por lo que está siempre disponible.

- Finalización de programa. También está programado como una interrupción, solo que su valor se guarda en una variable booleana, de tal manera que en el instante en que acabe el sub-programa en ejecución (si lo hubiera), regresará a la posición de reposo y finalizará el programa principal.
- Carga de un carro. Realiza el cargado con tres cilindros del carrito situado más próximo a la puerta junto a la estación 6. Se utilizan para su programación tanto coordenadas relativas como concatenación de movimientos.
- Descarga de un carro. Es el inverso del anterior, solo que descarga el carro situado al otro lado de las cintas transportadoras, si se toma al robot como centro. Devuelve las piezas a la cinta de suministro.

3.7. Material docente resultante

De lo que es puramente la programación del robot, se ha obtenido material para realizar un guión de prácticas (con su solución), equivalente a la práctica que se realizaba con otro modelo de robot en la asignatura «Control y Programación de Robots», impartida en Ingeniería Industrial.

En esencia, la práctica consiste en tener que manejar mediante el terminal de enseñanza el robot, para obtener las coordenadas de diversas posiciones. Posteriormente hay que programar el robot para que en base a las posiciones conseguidas, simule el proceso de realización de dos puntos de soldadura, y un cordón de soldadura por el borde de una caja de cartón.

El contenido íntegro del guión de la práctica se encuentra en el apéndice G.

Además del material directamente resultante de este capítulo, los conocimientos y parte del material derivado del mismo, junto con el resultado de capítulos posteriores, han servido para crear más material de uso docente, así como útiles manuales sobre el uso de los elementos de la célula de fabricación sobre la que versa gran parte de este proyecto. Es por ello que el trabajo desarrollado y explicado durante este capítulo es una parte básica del que se desarrollará en capítulos posteriores.

Capítulo 4

Control del robot con autómatas

Aunque el objeto principal de este capítulo es explicar cómo se ha realizado la comunicación y el control del robot mediante un autómata programable, se extenderá la explicación al control de todos los elementos de la célula de fabricación mediante los autómatas programables utilizados, dado que se realizó de forma secuencial en el tiempo y mediante idénticos principios básicos.

4.1. Comunicación entre autómata y robot

Un aspecto muy importante, y a su vez un constante reto a lo largo de todo el proyecto ha sido cómo comunicar el robot y el autómata. Si bien desde el primer momento se tuvo claro que lo deseado era que el robot recibiera órdenes simples del autómata, que podían perfectamente ir contenidas en señales digitales, el cómo efectuar ese paso de datos era un problema a resolver.

Como ya se ha comentado en el capítulo 2, el estado original de las comunicaciones del robot era considerablemente diferente al que se ha adoptado finalmente. En primer lugar, el robot se encontraba conectado a un módulo de entradas y salidas modelo «TSX MOMENTUM», y sólo tenía conectadas unas pocas entradas y salidas digitales, una señal de encendido y otra de emergencia. Los motivos para cambiar este módulo fueron múltiples, aunque no se debieron exclusivamente a su inoperatividad, sino también a razones de homogeneizar los módulos conectados a autómatas, y facilidad de uso.

El nuevo módulo que se ha instalado durante la realización de este proyecto, y que es el utilizado actualmente, es como en todas las estaciones que cuentan con un autómata M340, una isla «Advantys STB» con cabecera de bus CANopen.
En un principio se instalaron en la isla 6 módulos de entradas y salidas (tres módulos de cada tipo), alcanzando doce entradas y catorce salidas (ver figura 4.1).
En previsión de posibles ampliaciones y en previsión de que pudiera realizarse una programación que requiriera mayores prestaciones, se decidió cambiar estos módulos por otros más potentes que proporcionan 16 entradas y 16 salidas, con la utilización de sólo dos módulos en lugar de los cinco anteriores (ver figura 4.2).

La idea de utilizar un módulo con cabecera de tipo CANopen se debe a que como en el

Figura 4.1: Isla Advantys STB que se instaló inicialmente.

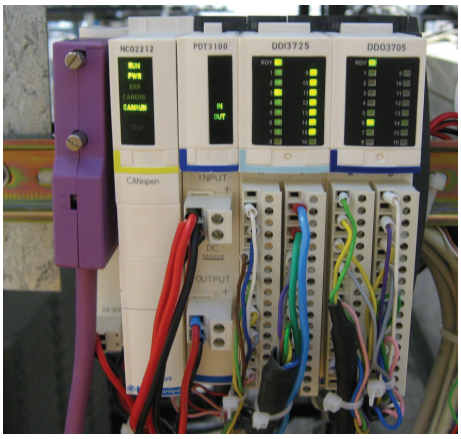

Figura 4.2: Isla Advantys STB que se encuentra actualmente instalada.

resto de estaciones, el deseo era situar un autómata próximo al robot que lo controlara. En un primer momento el autómata que se situó para controlar al robot era un autómata de iguales características que el resto de los que hay en el laboratorio. Más adelante se le añadiría un módulo de comunicaciones «NOE» del que se hablará más adelante en el apartado 4.3.2, y que diferencia a este autómata de los demás en cuanto a prestaciones, ya que las aumenta.

Para poder utilizar la isla Advantys, tanto la primera en ponerse, como la que se colocó después y aún perdura, ha sido necesario realizar la configuración de sus módulos. Dicha configuración se realiza mediante el programa para PC «Advantys», suministrado por el mismo fabricante de las islas Advantys, Schneider Electric. Dado que en el momento de realizar dicha configuración, no se contaba con esos conocimientos, fue necesario consultar diferentes manuales[22], [23], [25] y como resultado de ello y de la experiencia obtenida, se creó el «Manual de configuración de un módulo Advantys para su funcionamiento con un autómata MODICON». Dicho manual se adjunta en el apéndice C.

Haciendo un resumen de las conexiones del robot con los autómatas se concluye que tras el trabajo realizado durante este proyecto:

- El robot está cableado físicamente con un módulo «Advantys STB» de entradas y salidas (Todos los detalles en el apéndice A).
- Ese módulo está conectado, mediante un bus CANopen de dos puntos de acceso, a un autómata M340 situado junto al robot.
- Ese autómata además de poder realizar el control del robot de forma directa, puede servir de pasarela a otros autómatas mediante la red Ethernet.

4.2. Programación de los autómatas

La programación de los autómatas realizada durante el proyecto se va a tratar de abordar de una manera sencilla y concisa. Se intentarán evitar los puntos más técnicos así como

incluir código innecesariamente. No obstante, el código completo se puede encontrar en el apéndice E.

El software utilizado para realizar la programación de los autómatas, ha sido el programa «Unity PRO». Este programa es del propio fabricante de los autómatas, y es un software muy versátil, ya que permite la programación en cuatro lenguajes para autómatas diferentes:

- LD (Ladder Diagram, o Diagrama de Contactos). Se trata de un lenguaje antiguo proveniente de los diagramas de relés. Es poco potente, y no se ha empleado en la programación de este proyecto.
- IL (Lista de Instrucciones). Es un lenguaje de instrucciones extremadamente básicas, muy parecido al ensamblador para el código máquina. Puede ser muy eficiente, pero es asimismo muy complicado de seguir, por lo que tampoco se ha utilizado.
- ST (Texto estructurado). Se trata de un lenguaje muy parecido a los lenguajes de programación de alto nivel más habituales como C++. Es relativamente sencillo de usar y muy versátil. Se ha utilizado asíduamente durante este proyecto.
- SFC (Sequential Function Chart o Diagrama de Funciones Secuenciales). Es un lenguaje que reproduce en bastantes aspectos lo que sería un diagrama Grafcet. Ha sido también muy utilizado en el proyecto por ser un lenguaje muy visual y adecuado para realizar con facilidad su seguimiento durante la ejecución.

Para ilustrar dos de estos lenguajes se adjuntan las figuras 4.3 y 4.4.

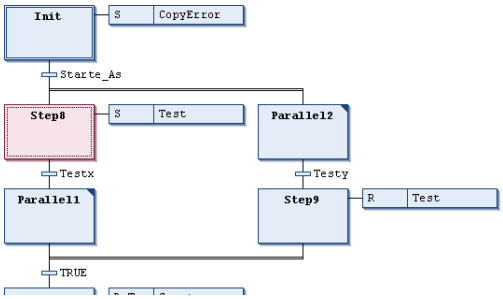

Figura 4.3: Ejemplo de programa SFC. Se aprecian varias acciones secuenciales, y dos en paralelo.

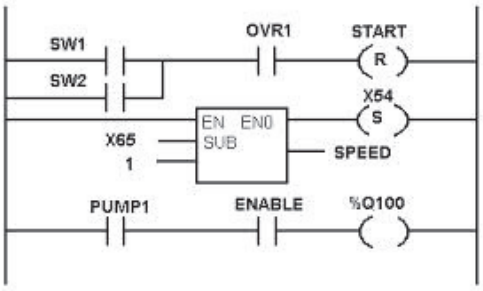

Figura 4.4: Ejemplo de programa LD. Se aprecian diversas condiciones a la izquierda, y algunas acciones a la derecha.

Como regla general sobre la utilización efectuada de los dos lenguajes de programación que se han señalado, se puede indicar lo siguiente:

- Para los programas de control de las estaciones, debido a su gran versatilidad para acciones secuenciales, se ha utilizado el lenguaje SFC. También se ha utilizado este lenguaje a la hora programar la selección de tareas del robot.
- El lenguaje ST se ha utilizado fundamentalmente para realizar las copias de variables, operaciones booleanas de comparación o verificación, y para otras instrucciones y rutinas inmediatas como los desencadenantes de las acciones del robot.

Se pasa a continuación a analizar con algo más de profundidad los puntos básicos de los programas del autómata principal y del autómata de la estación de expedición de palets. La división entre autómata principal y secundarios se analiza exhaustivamente en el apartado 4.3.1.

4.2.1. Autómata principal

El autómata principal (figura 4.5), como se explicará en el apartado 4.3.1, cumple dos funciones: Por una parte es un autómata secundario que controla dos estaciones de la sub-célula de pedidos (el robot y la estación de transporte) de manera independiente. Y por otra parte es el autómata que controla y coordina a todos los demás cuando se quiere que la sub-célula funcione integrada.

Esto sólo es posible mediante una minuciosa programación, en la cual se consideran tres programas, dos de los cuales son independientes entre sí, y un tercero, de jerarquía superior, que es capaz de manejar a los otros dos. De esta manera, el autómata principal contiene los programas correspondientes a lo que serían dos autómatas secundarios, y el programa que ejecutaría el autómata principal.

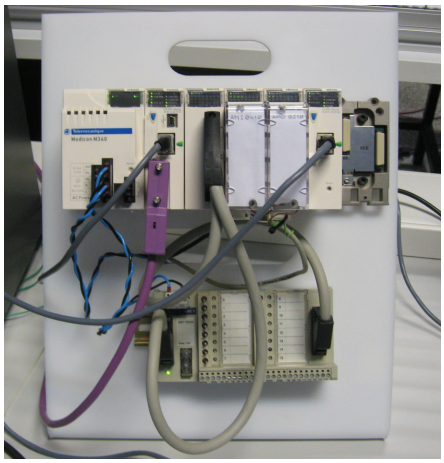

Figura 4.5: Autómata principal, situado junto al robot industrial.

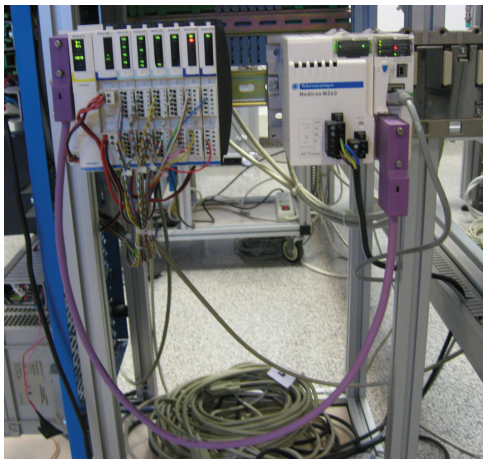

Figura 4.6: Autómata secundario presente en la estación de expedición de palets.

Programa que controla al robot:

El programa que controla al robot, es un programa que se trató de hacer lo más sencillo posible, pero que cumpliera su función. Su funcionamiento se basa en dos premisas:

- El robot tiene tres estados desde el punto de vista de la seguridad: Operativo, apagado o en emergencia. Dentro de operativo además puede estar en reposo (preparado para realizar nuevas tareas), o realizando una tarea.
  El único estado habilitante para poder realizar tareas es el de Operativo. Cualquier petición realizada mientras el robot está en otro estado deberá ser rechazada.

- El robot sólo es capaz de realizar una única tarea al mismo tiempo. Si un usuario solicita dos diferentes, es el autómata el que debe discernir. Por otra parte, si en el

momento en el que el robot se quede libre, la tarea que anteriormente se ha denegado en favor de otra todavía está solicitada, debe poder ejecutarse.

De acuerdo con las premisas anteriores, ambas fruto de la experiencia de las primeras tomas de contacto, se programó el autómata que controla al robot.
El programa en esencia consiste en dos diagramas grafcet disjuntos, que controlan por una parte la selección de tareas del robot, y por otra parte la seguridad.

La selección se realiza de la siguiente forma (Figura 4.7):
Estando el robot en reposo, se abre un abanico de siete posibilidades. De estas siete posibilidades (Coincidentes con las siete tareas comentadas en el apartado 3.6), se pueden elegir todas (Salvo unos casos especiales que se comentarán más adelante). El método de elección depende del operario, ya que están disponibles al menos dos métodos diferentes[1] (Botonera física y terminal de diálogo).
Una vez se ha realizado la selección, se pasa a un estado de pre-ejecución, en el cual se envía la orden al robot. En cuanto el robot confirma el inicio del programa, se pasa a un estado de ejecución de tarea. Este estado durará hasta que el robot indique que ha acabado. Cuando el robot confirma que ha acabado, se vuelve al estado de reposo. Y si no hay ningún problema, se pasa al estado de robot preparado.

Respecto a los casos especiales que se han mencionado anteriormente, son los siguientes:

- El trenecito mencionado en el apartado 3.6, inicialmente se considera desmontado. Por tanto, la tarea «desmontar tren» está deshabilitada hasta que se haya montado una vez. Del mismo modo ocurre con la tarea «montar tren», ya que si el tren ha sido montado queda deshabilitada hasta que se desmonte.

- La tarea «cargar palet» solo está disponible cuatro veces de forma consecutiva, de tal manera que si se han llenado los cuatro carritos, no puede ejecutarse hasta que se vacíe uno de ellos. De una forma parecida ocurre con «descargar palet», que solo puede realizarse si previamente se ha realizado la carga de al menos un palet.

Programa que controla la seguridad del robot:

En realidad se trata de una parte del programa que controla al robot, pero se describe como parte separada a continuación para darle un valor añadido debido a que es muy importante para la seguridad.
Este programa es muy directo, y está basado en el diagrama grafcet que se puede ver en la figura 4.8. Sus funciones, que son muy básicas, son detener el programa del autómata en aquellos casos en los que el robot esté en emergencia o apagado.

Ambos casos son un poco diferentes. En el caso de que el robot esté apagado, lo único que sucede es que se deshabilita la condición de robot preparado, con lo cual el robot no puede alcanzar el estado de robot preparado (ver figura 4.7).
En el otro caso, cuando se entra en una situación de emergencia, la acción es más contundente. Se paraliza el desarrollo del grafcet que controla al robot, y se reinicia ese grafcet

[1] Al menos dos significa que además de la botonera y del terminal de diálogo, también se entra en esa selección cuando se opera de manera automatizada toda la sub-célula, solo que no lo hace un operario.

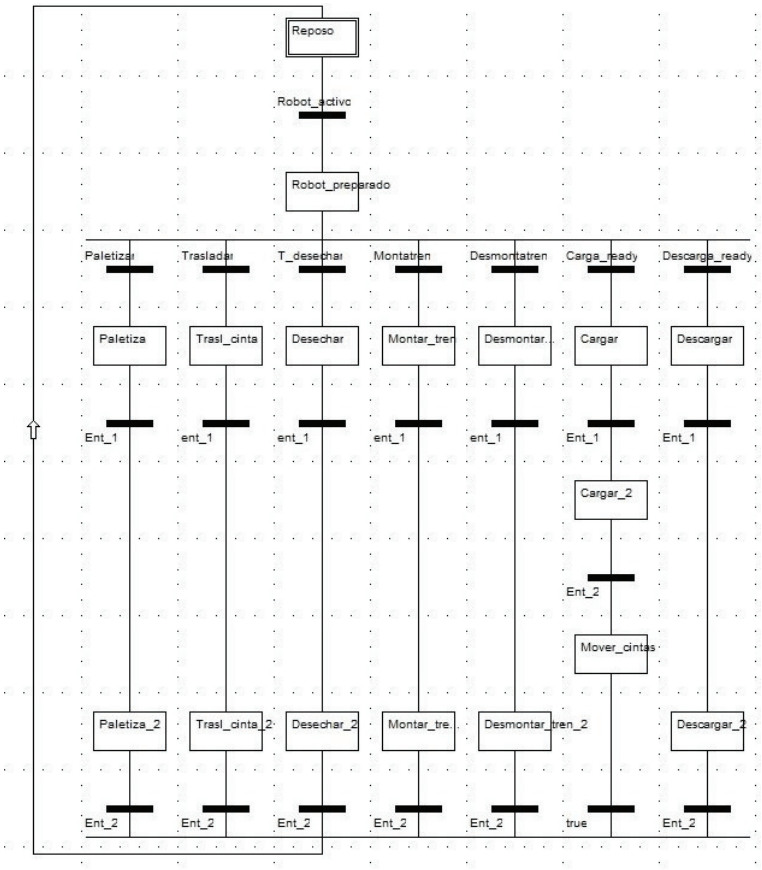

Figura 4.7: Ejemplo de SFC, correspondiente a la selección de programa del robot. Se aprecia una selección y diversas acciones secuenciales.

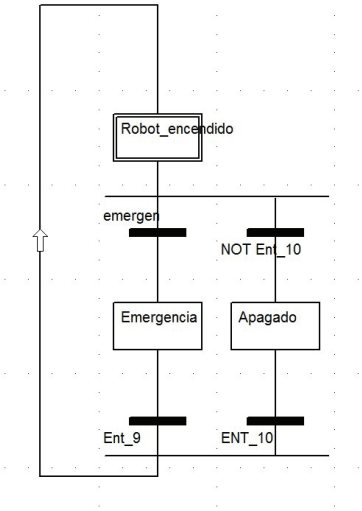

Figura 4.8: SFC correspondiente a la gestión de la seguridad del robot.

una vez se ha solventado la emergencia.
Este segundo caso es más importante ya que si no se reiniciara el grafcet de control del robot, se volvería a ejecutar la misma orden que se había dado antes de la emergencia,

pudiendo volver a provocar el mismo resultado.

Programa que controla la estación de transporte:

Consiste en un programa basado en un grafcet de etapas secuenciales (Figura 4.9), ya que todas las acciones necesarias para controlar la estación de transporte se realizan de forma consecutiva.

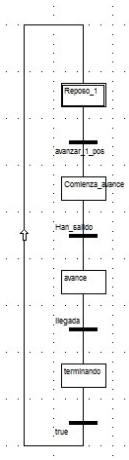

Figura 4.9: SFC correspondiente al control de los transportes (Cintas, topes, desvíos...).

En esencia su funcionamiento es el siguiente: La estación está en reposo hasta que recibe la orden de mover los carros una posición (Habilitar la transición «avanzar_1_pos»); Se produce el desbloqueo de los carros y se ponen en marcha las cintas transportadoras; Transcurrido un breve tiempo (obtenido de forma experimental), se considera que los carros ya están desplazados lo suficiente de sus bases, y se activan los topes; Se sigue produciendo el movimiento de los carros hasta que van llegando a su posición de destino, en la cual se paran mediante el tope correspondiente, y esperan a que todos estén en posición; Una vez en su posición, que se detecta por medio de sensores inductivos y capacitivos, se activan los bloqueos y se paralizan las cintas; Llegados a este punto, el programa termina y se vuelve al estado de reposo, hasta que vuelva a llegar la orden de inicio.

Programa que controla toda la sub-célula de expedición de pedidos:

Éste es realmente el programa que hace que el autómata adosado al robot sea el autómata principal. Hasta el momento se había hablado de programas propios de un autómata secundario, solo que en este caso se implementaban todos a la vez en el mismo autómata debido a que puede soportarlos todos. Pero el programa que se describe a continuación es propio de un autómata de jerarquía superior (superior en nivel de mando, lo que no implica que tenga que ser superior en prestaciones).

El programa que controla la sub-célula completa, se basa nuevamente en un diagrama grafcet con una importante componente secuencial y con carácter cíclico, que se muestra en la figura 4.10.

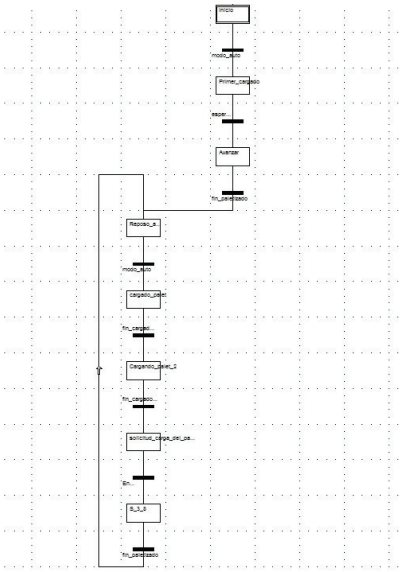

Figura 4.10: SFC correspondiente al control de la sub-célula completa (controlada totalmente por el autómata principal).

Dado que su función es permitir el funcionamiento automático y continuado de la subcélula, su estructura es fundamentalmente de carácter cíclico y se puede dividir en dos partes:
En primer lugar hay una parte no-cíclica, correspondiente a la inicialización de la subcélula. En ella lo que se hace es colocar una placa contenedora en un carro, y avanzarlo una posición, de tal forma que quede listo para ser cargado.
En segundo lugar aparece la parte cíclica del grafcet. Mientras esté activada la condición «modo automático» (resultante de colocar el selector «MAN/AUT» en posición «AUT», el selector «IND/INT» en posición «INT» y haber pulsado «marcha» una vez; Ver figura 4.11), se repite cíclicamente la siguiente secuencia: Se carga un carro con tres cilindros, se carga el carro siguiente con una plaquita del color que se haya definido, y se avanzan los carros una posición. Resultando siempre un carro totalmente vacío en la estación expendedora de palets, y un carro con plaquita y sin cilindros en la estación del robot.

Figura 4.11: Botonera física de la estación de transporte. Se aprecian los diferentes pulsadores y selectores a los que se hace referencia.

La manera que tiene el programa de controlar a los otros programas es muy directa. Los otros programas actúan como si controlaran a sus estaciones de manera autónoma, con la diferencia de que la señal que provoca el comienzo del programa solo puede provenir

desde el autómata de jerarquía superior.

De una manera más gráfica: Cuando las estaciones funcionan de manera autónoma, si en la de transporte se pulsara marcha en la botonera física, se ejecutaría el programa de manera que los carritos avanzarían una posición.
Sin embargo, si la estación está funcionando en el modo integrado, al pulsar el botón «Marcha» de la botonera, no pasa nada. Sólo avanzarán los carritos cuando se reciba la señal adecuada del autómata principal.

4.2.2. Autómata de la estación de expedición de palets

El autómata de la estación 6, aún siendo igual en sus características al de la estación del robot (Excepto por el módulo Ethernet «NOE»), realiza unas funciones inferiores.
Este autómata se dedica exclusivamente a controlar la estación 6 a través de la isla Advantys que tiene conectada por bus CANopen.

Como ya se ha dicho anteriormente, sus funciones son más limitadas, puesto que solo tiene cargado un único programa, que es el que gestiona la estación de carga de palets.
El autómata puede manejar la estación nuevamente de dos formas diferentes: En el contexto de la sub-célula de pedidos funcionando integrada, recibe órdenes exclusivamente del autómata principal.
Cuando la célula funciona de manera autónoma, el operario puede seleccionar el color de la pieza que esta estación expide, el momento de expedir una nueva pieza, y si el control se hace mediante la botonera física o mediante un terminal de diálogo.

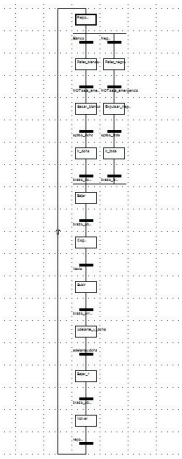

Figura 4.12: Diagrama SFC correspondiente al control de la estación de expedición de palets.

El diagrama grafcet de control de esta estación es muy efectivo y prácticamente secuencial, salvo por una selección de caminos, tal y como se puede observar en la figura 4.12.
Su funcionamiento en esencia es el siguiente:
Cuando se recibe la orden de expedir una nueva placa, dependiendo del color que se haya solicitado, se decanta por uno de los dos caminos posibles. Dependiendo de qué camino

esté siguiendo, el brazo de la estación va hacia unas placas u otras. Coge la placa deseada y la deposita encima del carro que está preparado en su lugar correspondiente.
Terminado esto, vuelve a una posición de reposo.

En este caso se ha optado por programar las instrucciones de seguridad exclusivamente mediante «texto estructurado», ya que son unas instrucciones adecuadas para programar en este código:
Cuando se aprieta el pulsador de emergencia, se paraliza la instrucción mediante una orden propia del autómata. Una vez se ha desenclavado el pulsador de emergencia, se debe pulsar el botón de «Reset» presente en la botonera física, tras lo cual se ejecuta otra orden que reinicia el grafcet de control, y deja la estación preparada para volver a producir.

4.3. Integración del conjunto

Aunque hasta el momento ya se ha comentado un poco sobre este tema y se ha nombrado en varias ocasiones, es momento de detallar cómo se ha realizado la integración del conjunto.

Estación 6		Transportes					
Selector MAN/AUT	Selector IND/INT	Selector MAN/AUT	Selector IND/INT	Estado est.6	Estado robot / transportes	Estado global	Uso terminales de diálogo
MAN	Tipo de placa	MAN	—	Autónoma	Autónoma	No integrado	Solo supervisión
MAN	Tipo de placa	AUT	—	Autónoma	Control externo	No integrado	Supervisión est.6 y control de robot y transportes
AUT	—	MAN	—	Control externo	Autónoma	No integrado	Control est.6 y supervisión de robot y transp.
AUT	—	AUT	IND	Control externo	Control externo	No integrado	Control est.6, robot y transportes
AUT	—	AUT	INT	Integrada	Integrada	Modo contínuo	Solo supervisión

Tabla 4.1: Resumen de los modos de funcionamiento de la sub-célula.

En primer lugar hay que comentar cómo se dejan las estaciones listas para el funcionamiento integrado.
Hay una parte de código programado en lenguaje estructurado, que selecciona desde dónde le llega al programa de control la información sobre cuándo comenzar la ejecución, u otras opciones como en la estación 6 o en la estación del robot.
El código es en concreto un condicional. En el caso de que se seleccione «AUT» en la botonera física, se pasa a la deshabilitación de la botonera. Aún se podrá controlar mediante un terminal de diálogo, o mediante el autómata principal.
En el caso de seleccionar «MAN», se pasa al funcionamiento manual, en el que las órdenes se dan a través de la botonera.

Dentro del modo automático, aún existe una opción más, y es poner el selector «IND/INT» de la botonera del transporte en posición «INT», con lo cual se pasa al funcionamiento continuo en el que se producen continuamente palets llenos de piezas.

Sirva la tabla 4.1 como resumen de todas las posibilidades existentes.

Respecto a los terminales de diálogo, en todo momento se puede visualizar en ellos lo que está sucediendo, y en todo momento está habilitado el pulsador de emergencia de los paneles. No obstante, sólo pueden ordenar la ejecución de un programa cuando se les permite mediante el selector «MAN/AUT».

4.3.1. Jerarquía de los componentes

Otro aspecto a tener en cuenta, y que se ha ido mencionando a lo largo de los capítulos anteriores, es el orden que cada componente posee en la jerarquía del conjunto de la sub-célula.

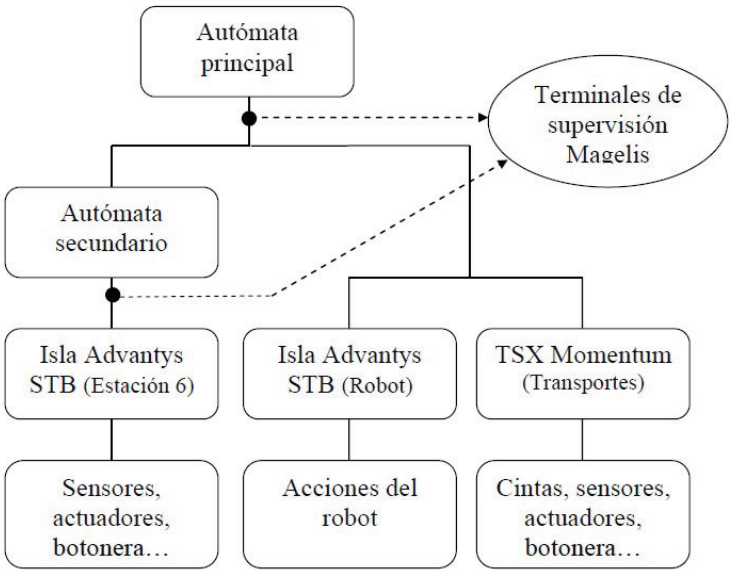

Figura 4.13: Estructura jerárquica del conjunto integrado.

El ente de mayor importancia (Ver figura 4.13) es el autómata principal. Por supuesto es el más importante cuando la sub-célula está en funcionamiento integrado, en otros casos estaría al mismo nivel que el autómata de la estación de expedición de palets.
Un escalón por debajo se encontrarían los terminales de diálogo «Magelis». Este escalón es ficticio porque realmente solo ejercen algún control guiados por un operario. Realmente el escalón lo ocuparía el operario.
Por debajo de estos, se encuentran los autómatas secundarios. En la figura 4.13 no se ha representado más que uno porque los otros están contenidos dentro del principal.
Hasta aquí los entes que son capaces de dar órdenes diferentes (adicionales) a las que

reciben.

El módulo TSX Momentum y las islas Advantys por su parte, son meros transmisores de la información. Extremadamente útiles, pero sin capacidad de decisión. Hay que indicar que otros módulos de comunicaciones sí que son capaces de realizar el procesamiento de tareas de reacción rápida que tienen grabadas en su memoria interna en función de determinadas entradas. Pero éste no es el caso.

En último lugar aparecen los sensores, los actuadores y el robot. El robot es un caso muy especial, ya que dispone de un controlador que perfectamente sería capaz de controlar todas las estaciones de un modo igualmente eficaz al que se hace con los autómatas. No obstante, debido a la limitación en este caso del número de entradas y salidas digitales, y por razones prácticas[2], se ha realizado mediante autómatas.

4.3.2. Módulo de comunicaciones Ethernet

El módulo de comunicaciones «NOE 0100», merece un apartado especial debido a su gran utilidad y simplificación de las comunicaciones con otros autómatas y módulos de entradas y salidas cableados a la red Ethernet.

La gran ventaja que ofrece este módulo es el poder prescindir de unas complicadas funciones de lectura y escritura, que se utilizan para el acceso a las posiciones de memoria de otros autómatas.
Este módulo resuelve ese problema de una manera sencilla respecto a las nombradas variables, y además permite acceder a otros dispositivos que pueden tener diferentes sintaxis de memoria, admitiendo los protocolos «Modbus», «IEC-0» e «IEC-1». Con ello se facilita mucho la comunicación y las interconexiones entre autómatas diferentes y módulos con diferente protocolo de comunicación.

Figura 4.14: Vista de un módulo NOE 0100 como el existente en el laboratorio.

La forma de acceder a las variables externas al autómata por medio de este módulo, y que además es la utilizada en este proyecto, se hace en tres rápidos pasos (Más información

[2]Una importante razón práctica es el hecho de que habría sido necesario volver a cablear las entradas y salidas de la estación 6 y de los transportes para que estuvieran en conexión con el robot.

en [24]):

1. En la interfaz de programación (dentro del programa Unity PRO), se introduce la dirección IP del dispositivo al que se quiere acceder, y la sintaxis que utiliza para las direcciones de memoria.

2. Se introduce la posición de memoria a la que se quiere acceder, y el número de posiciones que quieren ser leídas de manera consecutiva.

3. Se indica la dirección de memoria del autómata con el que se está trabajando en la que se quiere copiar la variable (En el caso de tener varias variables, se indica la primera posición y serán copiadas en esa y sus consecutivas).

De acuerdo con los objetivos de este proyecto, se ha elaborado una pequeña guía de configuración de las entradas y salidas digitales mediante el módulo NOE 0100. Se adjunta en el apéndice D.

4.4. Material docente resultante

Cumpliendo con uno de los objetivos planteados inicialmente, de los capítulos analizados hasta este momento se ha obtenido material para desarrollar una práctica que se basa en el estudio y realización mediante pasos detallados de un programa que permite el uso del robot industrial a través de la botonera de un autómata.

Esta práctica es relativamente compleja, y requiere conocimientos de Robótica Industrial y de programación de autómatas. No obstante se cree que la práctica propuesta puede dar una completa visión sobre algunas de las estrategias posibles a la hora de comunicar entre sí diversos componentes de una célula industrial automatizada, así como mostrar de una manera práctica varios de los puntos más fuertes del temario de varias asignaturas de robótica y automatización industrial.

Se adjunta el guión realizado como primera aproximación a este material en el apéndice H. Aunque está completamente preparado para su empleo, no se recomienda su uso en una sola sesión de prácticas debido a la gran complejidad y necesidad de material y conocimientos previos por parte de los alumnos que esta práctica necesita.

Capítulo 5

Supervisión mediante terminal de diálogo

5.1. Motivación del uso de un terminal de diálogo

Los terminales de diálogo son elementos muy extendidos actualmente en todos los procesos automatizados. Debido entre otras cosas a su gran versatilidad a la hora de ser programados, y al hecho de que aumentan mucho la facilidad de manejo de la instalación ya que si la programación se ha realizado adecuadamente, son bastante intuitivos.
Sirva la figura 5.1 como ilustración de lo que puede ser la pantalla de un terminal de diálogo en su utilización en una industria.

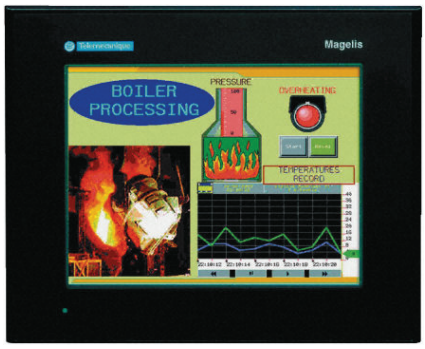

Figura 5.1: Ejemplo de terminal de diálogo en una industria.

Respecto a los motivos de su utilización, se dio además de todas sus virtudes, la circunstancia adicional de que entre el material disponible en el laboratorio L 0.6 del edificio Ada Byron, existen cinco terminales de diálogo, modelo «MAGELIS XBTGT4330», del fabricante Schneider Electric.
Debido a su disponibilidad, a la existencia de material sobre cómo utilizarlos [27], y al gran valor añadido que podían aportar al proceso de integración en curso, se decidió su incorporación al mismo.

5.2. Modificaciones necesarias en la programación de los autómatas

La utilización de estos terminales de diálogo, implicaba la necesidad de replantear cómo se debía acceder al control de los autómatas, ya que como es lógico, no debe de permitirse el acceso desde diferentes elementos al mismo tiempo, ya que podrían incluso dar ordenes contradictorias. Por ello, lo que se decidió fue que el acceso al control de los autómatas viniera determinado por la botonera física presente en cada una de las estaciones. De esta manera se evita la posibilidad de que sean controlados desde dos accesos diferentes.

El selector «MAN/AUT» de la botonera fue el elegido para la disyunción entre el control mediante la botonera y el terminal de diálogo. En el apartado 4.3 se puede consultar la tabla 4.1, en la cual se detalla cómo operar con este selector.

Aunque en el referido apartado 4.3 ya se habla brevemente sobre el código empleado para el uso de los terminales de diálogo, se va a especificar un poco más cómo se realiza y por qué se ha hecho así.

Para la utilización de los terminales Magelis, es necesario el uso de unas variables intermedias pertenecientes a la memoria del autómata. Esto es así porque:

- Los terminales Magelis tienen capacidad para leer y escribir cualquier posición de la memoria interna del autómata. Pero solo de la memoria interna, ya que no es posible acceder a ninguna otra dirección de memoria.
- Originalmente, los programas se diseñaron para ser dirigidos mediante la botonera. Por ello algunos desencadenantes venían dados por las variables en las cuales se escribe el estado de la botonera (A través del bus CANopen).
- El terminal Magelis es capaz de leer las posiciones donde se escribe el estado de la botonera, pero no es capaz de escribir sobre ellas porque son actualizadas cada ciclo de reloj con los valores reales.
- Por tanto, para que el terminal pueda dirigir las operaciones, lo que se hizo es una «copia» de las variables originales. Esa «copia» recibe el valor de las variables de la botonera cuando dicha botonera está habilitada, o es modificada por el terminal Magelis cuando este está habilitado.

Sirva la figura 5.2 para ilustrar cómo funciona la asignación de valor a las variables que manejan los programas de los autómatas.

5.3. Integración en el conjunto

Una vez realizadas las modificaciones necesarias en el código de los autómatas, se pasó a realizar el diseño de los terminales. En concreto se han diseñado dos terminales diferentes: Uno para la supervisión de la célula completa (cuando funciona en modo continuo), que

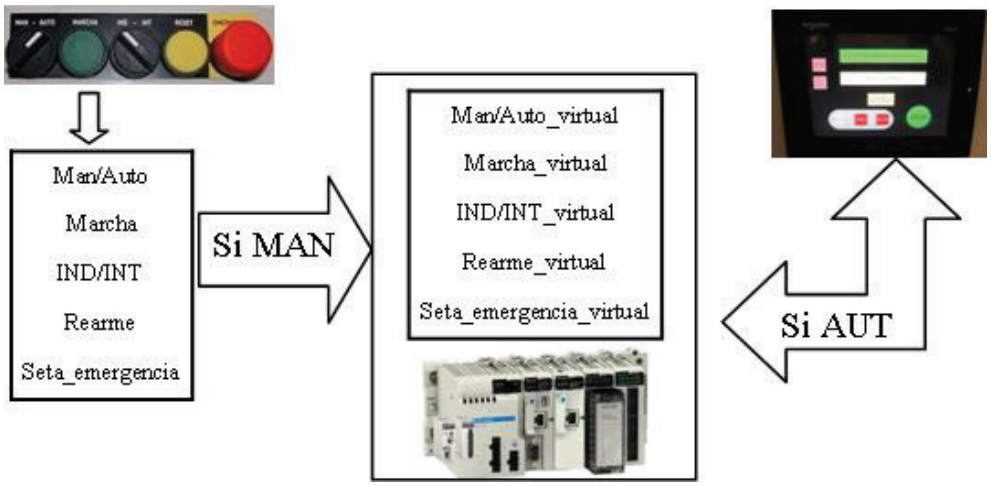

Figura 5.2: Diagrama explicativo del acceso a memoria. La posibilidad de acceder a la memoria está condicionada según el selector MAN/AUT.

también puede controlar al robot y a la estación de transporte (en modo no continuo). Y otro para la supervisión (en modo integrado) de la estación de expedición de palets, y su control cuando se utiliza en modo aislado.

Para la programación de estos terminales se ha utilizado el software «Vijeo-Designer» suministrado por el propio fabricante Schneider Electric.
La documentación de referencia ha sido una práctica de una asignatura de Ingeniería Industrial [27], así como la ayuda contenida dentro del programa.

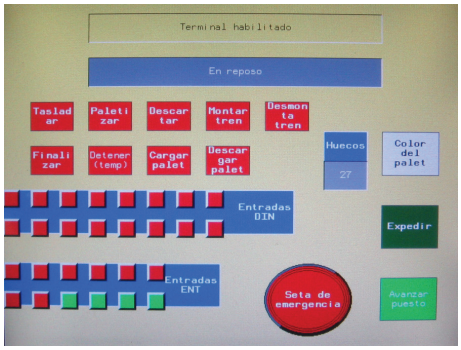

Figura 5.3: Pantalla del terminal diseñado para controlar el conjunto.

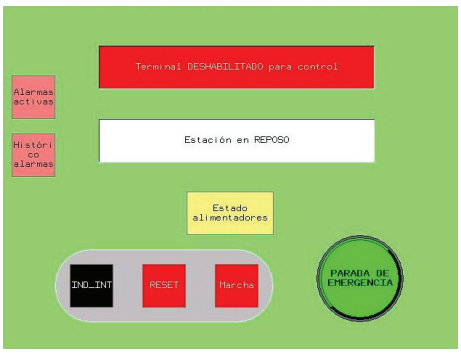

Figura 5.4: Pantalla del terminal diseñado para la estación expendedora de palets.

La programación mediante este software es relativamente intuitiva debido al aspecto de carácter actual del programa, y permite como se puede comprobar en las figuras 5.3 y 5.4 un resultado bastante atractivo y fácil de interpretar.
El objetivo perseguido con los diseños realizados era lograr la mayor facilidad de uso de los terminales, y dar toda la información posible sobre el estado del funcionamiento de la célula.

Ambos paneles realizados poseen en común los siguientes elementos:

- Un pulsador de emergencia. El cual se trató de hacer lo más visible posible. Este pulsador está disponible siempre, incluso cuando el terminal está deshabilitado.

- Botones y pilotos. Sirven respectivamente para cambiar el estado de variables de tipo booleano, y para mostrar su estado.

- Cuadros de diálogo. Permiten señalar el estado de la célula mediante la visualización de diferentes textos. El texto a mostrar se selecciona dependiendo de una variable de tipo entero, que es cambiada por el programa del autómata en función de en qué estado se encuentra.

Además de estos puntos en común, también aparecen ventanas emergentes, paneles de información de emergencias y visualizadores numéricos.

Un punto a tener en cuenta en el caso de ambos terminales es que cuando están deshabilitados (Ver figuras 5.5 y 5.6), aquellos botones que pierden su función desaparecen de la pantalla. Facilitando así la detección de que el terminal está deshabilitado. Este estado también se señala mediante los paneles de información.
En cualquier caso, con el panel deshabilitado aún se puede supervisar por separado el estado del conjunto o de la estación 6 (en sus respectivos terminales), mediante la visualización de algunas variables y de los paneles informativos.

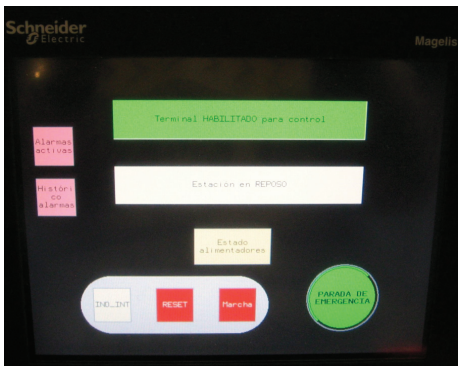

Figura 5.5: Terminal de la estación de expedición de palets en estado habilitado

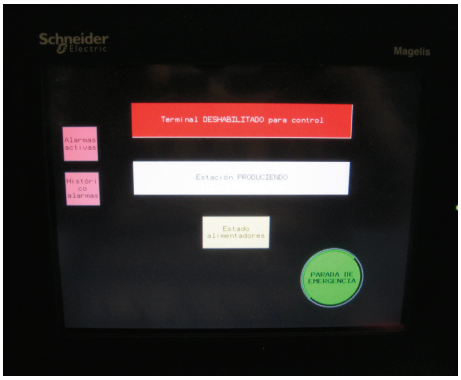

Figura 5.6: Terminal de la misma estación en estado deshabilitado. Desaparece la botonera dado que no se permite su uso.

En las figuras 5.5 y 5.6 se observa claramente cómo algunos botones han desaparecido al deshabilitar el terminal. La deshabilitación además se puede distinguir fácilmente debido a que el cuadro de habilitación ha pasado de verde a rojo. Nótese que el pulsador de emergencias sigue habilitado en ambos casos, así como todos los elementos que sin controlar la estación, dan información sobre ella.

5.4. Material docente resultante

Del conjunto de todos los elementos utilizados hasta ahora, se ha obtenido información para la realización de una propuesta de guión para un práctica muy completa, que aúna tanto la programación de un robot, como de un autómata, y la utilización de terminales de diálogo para la supervisión (y el control si se desea) de la sub-célula de expedición de pedidos.
Dicha práctica, cuya propuesta consta de dos partes, se adjunta en los apéndices H e I.

Además, debido a la posibilidad de que la propuesta anteriormente mencionada fuera excesivamente ambiciosa para una sesión práctica de la duración habitual, y para posibilitar más opciones, se ha desarrollado una segunda propuesta, encaminada únicamente a enseñar cómo usar los terminales de diálogo, para el control de tres estaciones diferentes de la célula completa. En concreto, de las estaciones 1, 3 y 4. El guión propuesto se adjunta en el apéndice J.
Para esta última práctica, también ha sido necesario realizar una programación básica de los autómatas que controlan respectivamente a las estaciones 1, 3 y 4.
Hay que destacar que una versión adaptada de esta propuesta de práctica ya ha sido empleada en una asignatura de grado.

Capítulo 6

Incorporación de una cámara RGB-D

Las cámaras RGB-D son dispositivos de percepción un tanto especiales, ya que aúnan las características de una cámara RGB, con las de un sensor de rango (de profundidad). El resultado que ofrecen es una imagen que combina el color de cada punto con la posición respecto a la cámara de dicho punto. Esta información adicional, convierte a este tipo de dispositivos en elementos extremadamente útiles, en tanto que son capaces de obtener de una misma escena mucha más información que uno solo de los dispositivos que combinan, y de un modo mucho más eficaz que si hubiera que utilizar dos dispositivos independientes.

Figura 6.1: Cámara RGB-D modelo «Asus Xtion Pro Live» utilizada durante este proyecto.

Como se puede observar en la figura 6.1, se distinguen tres dispositivos en el frontal de la cámara. Uno de ellos es la cámara RGB. Los otros dos son el emisor de infrarrojos para la proyección de patrones, y el receptor de infrarrojos para obtener coordenadas a partir de esos patrones (Más información en apartado 6.2).

La Universidad de Zaragoza dispone de varios de estos dispositivos, y dada la posibilidad de utilizarlos, y tras valorar su utilidad (Ver apartado 6.1), se procedió a incorporarlos al proyecto.

6.1. Motivación para el uso de una cámara RGB-D

Aunque en un principio pueda costar el ver la utilidad inmediata que tiene el uso de una cámara RGB-D en un proceso de automatización, tras el desarrollo de algunos programas, y en base a la experiencia adquirida, es posible afirmar que tienen un largo recorrido en

aplicaciones tales como la seguridad, la identificación y la localización de objetos.

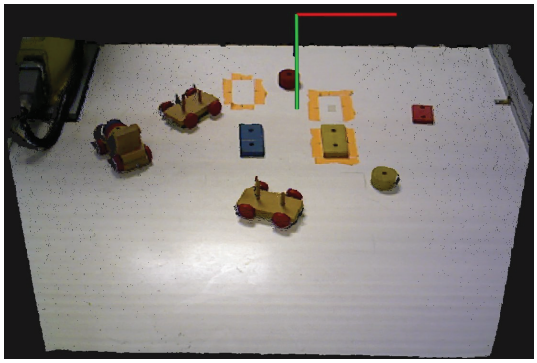

Figura 6.2: Imagen RGB capturada por la cámara.

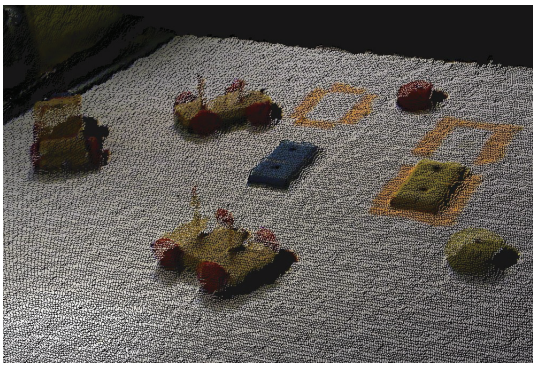

Figura 6.3: Imagen resultado de la combinación del color y la profundidad. Se aprecian puntos con color y posición.

En primer lugar es interesante tener una representación gráfica de qué se obtiene de este tipo de cámaras. Sirvan para ello las figuras 6.2 y 6.3. En estas imágenes, especialmente en la segunda de ellas, se pueden apreciar los puntos que conforman los datos que proporciona esta cámara.

La importancia de tener información sobre las coordenadas de cada punto de la escena es alta, ya que un clásico de la visión por computador es el dedicar un gran esfuerzo a obtener las coordenadas de un punto, mediante la resolución de ecuaciones procedentes de algoritmos complejos de emparejamiento de puntos, a partir de cámaras estéreo.
Por tanto, la ventaja que proporciona este formato de cámaras es la de aligerar mucho los cálculos, con lo que se obtiene un procesamiento de imagen mucho más fluido.

Por parte de la automatización industrial y la robótica, el uso de la visión es ya un hecho consumado, por lo que el interés radicaba en este caso en ver cuán eficientes podían ser los algoritmos de estas cámaras para competir con la visión por computador más clásica, y ya implantada en este sector.
En resumen, el objetivo de introducir la visión por computador mediante cámaras RGB-D en este proyecto era fundamentalmente comprobar la gran potencialidad que se les supone a estas cámaras, en concreto como asistencia a la robótica.

Como avance, se han planteado dos líneas a explorar: Por un lado el aspecto de control de seguridad dentro del espacio de trabajo de un robot. Y por otra parte, la identificación de algunos de los pequeños objetos que están presentes en la célula de transporte sobre la que se ha estado trabajando durante el proyecto. Esta parte del proyecto se ha realizado con la intención de proporcionar información útil sobre la manejabilidad y el potencial de estas cámaras, así como unos ejemplos demostrativos, y que puedan servir de base para futuras líneas a trabajar.

6.2. Obtención de la información

Esta sección no pretende más que esbozar cómo se obtiene la información que más tarde se utilizará. Por ello, en caso de necesitar más información, será necesario consultar las referencias que se citan a lo largo de este capítulo.

El dispositivo utilizado para la obtención de los datos es una cámara de tipo RGB-D, modelo Asus X-tion Pro Live. Se trata de un dispositivo orientado a desarrolladores, de carácter no industrial, cuyo máximo exponente de utilización actualmente podría ser la cámara «KINECT» para videoconsolas XBOX. A día de hoy existen dispositivos RGB-D de carácter industrial, aunque son mucho menos asequibles, y menos adecuados para el ligero desarrollo que se ha hecho durante este proyecto.
Las principales características del modelo empleado en este proyecto son las que se muestran en la tabla 6.1. Más información en [3].

Rango de uso	Entre 0.8 m y 3.5 m
Campo de vista	58º Horizontal, 45º Vertical, 70º Diagonal
Sensores	RGB, profundidad y 2 micrófonos
Tamaño de la imagen de profundidad	VGA (640x480) hasta 30fps (Fotogramas por segundo). QVGA (320x240) hasta 60fps
Interfaz	USB 2.0 ó 3.0
Entorno de trabajo	Interior

Tabla 6.1: Características del dispositivo RGB-D.

La captura de imágenes, ya sean de color o de profundidad, se puede hacer de dos maneras diferentes, dependiendo de las necesidades del programa.
Por una parte, la más sencilla es realizar una captura individual, y trabajar sobre ella extrayendo toda la información. La próxima captura se realizará cuando se haya completado el trabajo con la primera, si es que se solicita una segunda captura.
La otra forma de trabajar consiste en lo que se denomina un función de «devolución de llamada». Esta función realizará fotos continuamente, corriendo en paralelo al programa principal que se esté utilizando. Para aprovechar toda la información que captura, requiere de programas muy ligeros. En caso de que el tiempo de cálculo sea mayor que el tiempo entre dos fotogramas, se perderá el último (o los últimos).

A continuación se realizará una breve explicación del modo en el que se representa y ordena la información capturada mediante este dispositivo. También se explicarán brevemente las librerías de funciones utilizadas.

6.2.1. Representación y almacenamiento de la información

Una manera muy práctica de almacenar la información dada por una cámara RGB-D, y la usada en este proyecto, es lo que se denominan «Nubes de puntos». En concreto según el formato utilizado en las librerías de funciones PCL (*Point Cloud Library*)[16]y[17] , una nube de puntos consiste en una matriz bidimensional, que contiene toda la información

que la cámara es capaz de capturar.
En primer lugar es un vector ordenado, en el que cada posición corresponde a uno de los puntos que la cámara puede captar (307200 en el caso de resolución VGA(640x480)).
Cada una de estas posiciones, contiene un nuevo vector en el cual se almacenan los siguientes datos:

- Color. Con sus tres canales (R, G y B). Dependiendo del formato de datos también puede darse en su formato hexadecimal.
- Posición. Con otros tres parámetros (X, Y, Z). También existe la opción NaN (*Not a Number*) que indica que no se ha podido realizar la medida para ese punto.
- Otra información relevante. Por ejemplo el número de puntos, el vector normal a cada punto, el punto de vista de la cámara...

En la figura 6.4 se explica de un modo más gráfico la manera en la que se almacena toda la información. El tipo de archivo es propio de la librería comentada, y su extensión es «.PCD».

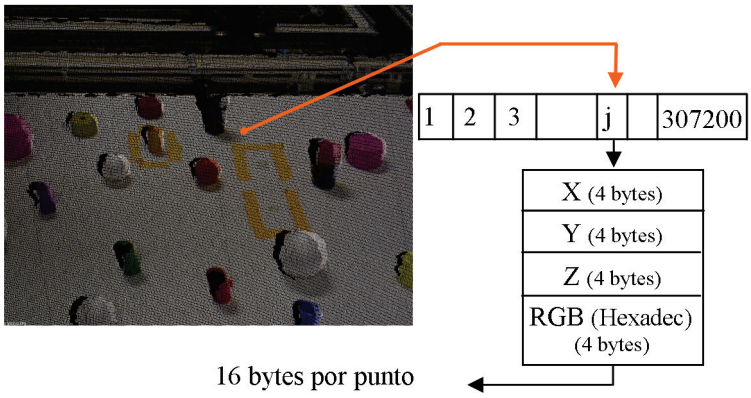

Figura 6.4: Esquema explicativo del almacenamiento de información en formato «Nube de Puntos».

Como se puede entrever, el consumo de memoria es mucho mayor que en el caso de una fotografía normal, ya que la representación del color en las nubes de puntos es solo una cuarta parte del total de información. Lógicamente, esto hace muy costoso en cuanto a cálculos el trabajar con nubes en bruto.
Por ello un tratamiento muy usual es el de pasar un filtro que elimine aquellos puntos que no poseen información relevante de posición. O también un filtro denominado «Voxelgrid», que se encarga dividir la escena en «cubos», y convierte todos los puntos del interior de cada cubo en un único punto, cuyas características (color y posición) son el compendio de todos los puntos.

6.2.2. Segmentación del escenario

En visión por computador, segmentar una imagen consiste en dividirla en múltiples conjuntos de píxeles. El objetivo de esta división es convertir, simplificándola, la repre-

sentación de una imagen en algo que posea mayor significado y sea más fácil de analizar por un computador.

En este proyecto, las características que se buscaban eran diferentes para los dos algoritmos que se han desarrollado.
En el caso del supervisor de seguridad, el objetivo era fundamentalmente comprobar qué puntos cambian de un fotograma al siguiente. Por ello no fue necesario el uso de la segmentación de la escena.
Por su parte, el identificador de objetos sí que necesitaba una partición de la escena, ya que debe realizar complejos cálculos, y es mucho más eficiente realizarlos con aquellas partes de la escena que se identifiquen como posibles objetos, en lugar de hacerlo con la escena completa.

A partir de aquí, los caminos seguidos para la implementación de ambos algoritmos se separan, por lo que serán tratados de diferente manera. Al algoritmo de identificación de objetos le corresponde el apartado 6.3, y al algoritmo de control del espacio de trabajo el apartado 6.4

6.3. Algoritmo para identificar objetos

Este algoritmo, aunque se trata de un ejemplo relativamente elemental de lo que podría ser en realidad un algoritmo completo de identificación de objetos, es capaz de identificar dos tipos de objetos.
Por un lado distingue entre esferas, cilindros y aquello que no es ni esfera ni cilindro. Y además de distinguir esferas y cilindros entre sí, también es capaz de identificar su color.

Posteriormente a su identificación, guarda las nubes de puntos que contienen alguno de los objetos identificados, y también proporciona un archivo con las coordenadas de los diferentes objetos.

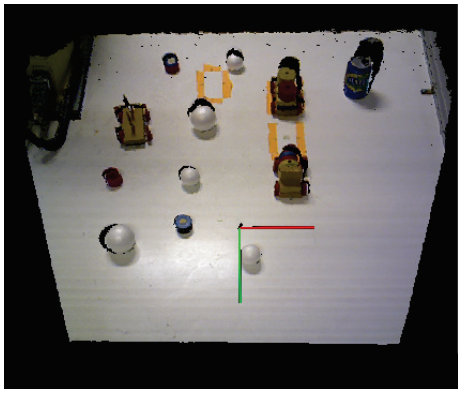

Figura 6.5: Imagen introducida al algoritmo

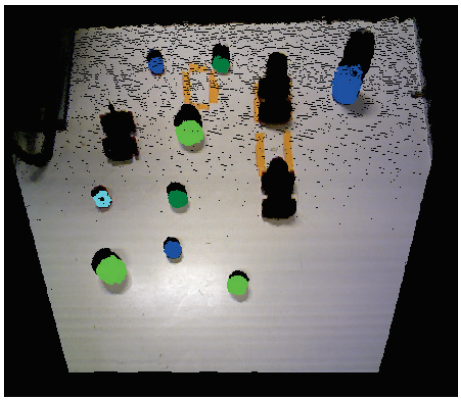

Figura 6.6: Resultado obtenido del algoritmo.

En la figura 6.5 se puede observar cómo es la nube de puntos original con la que va a

trabajar el algoritmo. En ella hay diversos objetos, además del plano correspondiente a la mesa donde está montado el robot, y parte de la valla que lo protege.
Por su parte, en la figura 6.6, se ve cómo el algoritmo ha identificado y dibujado en colores planos los diferentes objetos que ha conseguido identificar.

El proceso seguido se comentará más profundamente en las secciones siguientes, pero fundamentalmente consiste en obtener la orientación de la escena localizando el plano principal; Eliminar los planos más grandes; Dividir la escena que quede en trozos más pequeños; Identificar el contenido de dichos trozos; Obtener alguna clase de información de color para los objetos; Y mostrar la información obtenida de una forma adecuada.

6.3.1. Orientación de la imagen respecto al plano principal

El primer paso que se da para convertir la nube de puntos en bruto en información más manejable, es localizar el plano principal. Una vez localizado, lo que se hará es aplicar una matriz de transformación (obtenida en base a dicho plano) a todos los puntos de la nube, de tal forma que el plano principal pase a ser el plano horizontal.

El comienzo es lógicamente localizar el plano. Primero hay que identificar qué puntos de la escena son parte de dicho plano. Esto es lo que se denomina «Segmentación del plano». Para realizar la segmentación del plano, se utiliza una parte de código ya implementado en las librerías PCL, y que corresponde con el algoritmo de segmentación «RANSAC» *(Random Sample Consensus)* [13]. Dicho algoritmo, lo que hace en resumen es ir probando combinaciones de puntos (pocos) con los que forma un plano (u otra figura que posea una ecuación definida). Con cada plano obtenido aleatoriamente comprueba qué cantidad de puntos pertenecen a dicho plano, y probando diferentes planos maximiza esa cantidad. Por tanto siempre obtendrá el plano más grande de los posibles.
La cantidad de muestras aleatorias que ha de coger y el nivel de precisión al determinar los puntos pertenecientes determinan el tiempo de cálculo. Este algoritmo se vuelve a utilizar, y se detalla más su funcionamiento en el apartado 6.3.4.

El algoritmo «RANSAC», tal y como se ha configurado, devuelve como salida una nube de puntos que contiene el plano encontrado, una nube a la cual ha restado el plano encontrado, y la dirección de la normal del plano. Posteriormente, la manera de obtener una matriz de transformación una vez localizado el plano es bastante directa. Dado que se dispone de la dirección de la normal al plano, y considerando que la cámara solo está rotada respecto de un eje, tan solo hay que resolver un elemental sistema de dos ecuaciones:

Sean dos ejes X e Y, correspondientes al plano horizontal, y un tercero Z perpendicular a los anteriores. Defínase θ como el ángulo de inclinación de la cámara respecto al plano definido por X e Y, es decir: Un giro de dicho plano según el eje X.

$$\tilde{Y} = Y\cos(\theta) - Z\sin(\theta) \tag{6.1}$$

$$\tilde{Z} = Y\sin(\theta) + Z\cos(\theta) \tag{6.2}$$

De las ecuaciones (6.1) y (6.2), se sabe que la coordenada Y del plano debería ser cero, ya que la intención es orientar la cámara en el eje Z. Por lo tanto aún se simplifican más

y se obtiene la relación descrita en (6.3) y (6.4).

$$\tilde{Y} = -1\sin(\theta) - - > \sin(\theta) = -\tilde{Y} \tag{6.3}$$

$$\tilde{Z} = 1\cos(\theta) - - > \cos(\theta) = \tilde{Z} \tag{6.4}$$

Se obtiene por tanto el ángulo θ, que se utilizará para reorientar la nube de puntos aplicándole la matriz de rotación (6.5), válida para el giro respecto a un eje X.

$$Rot(x,\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.5}$$

Una vez se tiene la matriz descrita en (6.5), la forma de aplicarla se muestra en la ecuación (6.6), que como se puede ver es un procedimiento muy directo, que cuenta con una función definida en las librerías PCL para ser aplicado. Una vez hecho esto, se dispone ya de una nube reorientada. El siguiente paso consistirá en segmentar y eliminar los planos.

$$\tilde{U} = Rot(x,\theta)U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{6.6}$$

En la figura 6.7, se muestra el resultado y un paso intermedio del proceso de reorientación de la matriz.

Figura 6.7: Proceso seguido para reorientar una nube de puntos.

La forma de trabajar con estas matrices, que están implementadas en un apartado de las librerías PCL, se ha extraído de la bibliografía [4]. La manera de utilizar las funciones disponibles en la librería PCL se ha obtenido de las guías de dicha librería [17].

6.3.2. Segmentación y eliminación de planos

Como ya se ha comentado antes, la segmentación de los planos se realiza mediante un algoritmo denominado «RANSAC». La manera de realizar la operación de identificar los planos no reviste de mayor complicación una vez se conocen las librerías PCL que a ello están destinadas.
Esta eliminación de planos es importante por dos motivos: Para poder separar mejor los objetos que se quiere identificar; Y para reducir la cantidad de cálculos al identificar los objetos.

El procedimiento es el siguiente:

1. Se busca el plano más grande, y una vez localizado, se elimina de la nube sobre la que se está trabajando. No obstante, se guarda en otra nube auxiliar para su posterior uso.

2. Se repite la operación con los sucesivos planos que se encuentren.

3. Si un plano es paralelo al principal (el primero encontrado), no se elimina. Esto se hace así porque en determinadas ocasiones, el algoritmo de segmentación de planos puede confundir una serie de superficies a la misma altura con un plano.

4. Se deja de buscar planos cuando el plano actual tiene menos puntos que un determinado número. Este número es regulable, y es necesario porque en caso contrario podría encontrar y eliminar planos pertenecientes a algunos de los objetos de interés.

En la figura 6.8 puede verse el procedimiento por el que se han ido eliminando sucesivos planos de una nube de puntos.

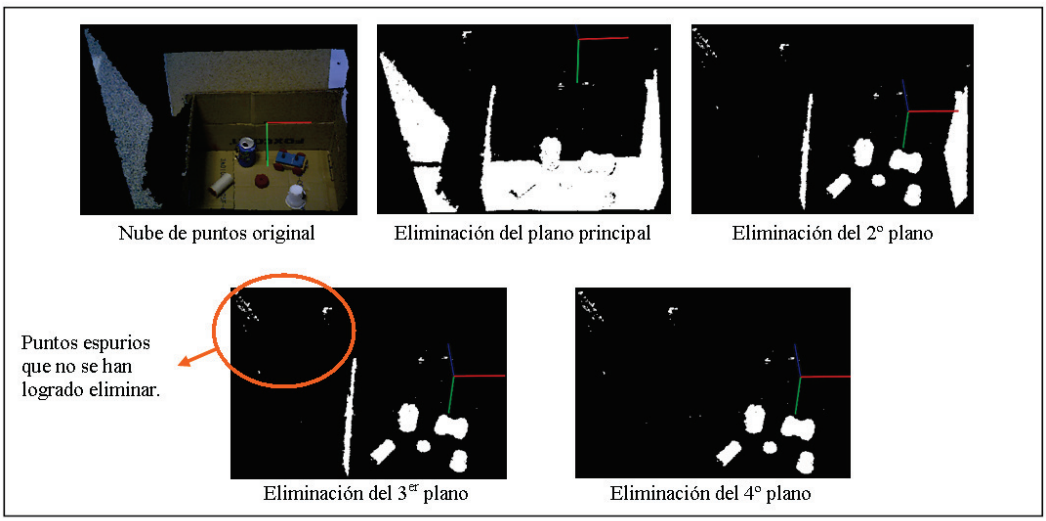

Figura 6.8: Ilustración de cómo se eliminan los planos. Se puede observar cómo se van eliminando secuencialmente de mayor a menor.

6.3.3. División de la imagen resultante en conjuntos de puntos

Una vez se han eliminado los planos, el siguiente paso es dividir la nube resultante en tantas nubes como objetos haya. Para esto lo que se hace es utilizar un algoritmo de división en «clusters». Los «clusters» no son sino agrupaciones de puntos vecinos entre sí. La librería PCL dispone de un algoritmo de obtención de clusters, y su funcionamiento se explica en la bibliografía de dichas librerías [26].

La separación en *clusters* sigue básicamente el siguiente proceso:

1. Crear un «Árbol de vecindad» para todos los puntos de la nube. En él se relacionan los puntos según sus coordenadas físicas en vez de por su posición en la matriz.

2. Crear grupos de puntos vecinos (*Clusters*), con la condición de que los puntos estén lo suficientemente cerca unos de otros (dentro de un radio que es configurable)

3. Cuando el grupo creado tiene menos puntos que un determinado umbral fijado (y modificable), termina el proceso de agrupación.

Los grupos más pequeños de puntos se desprecian, porque tal y como se puede apreciar en la figura 6.8, existen grupúsculos de puntos espurios, resultado de la eliminación imperfecta de los planos, que el algoritmo de agrupación en conjuntos podría interpretar como conjuntos válidos de puntos. Una vez que se tienen los objetos separados dentro de diferentes nubes de puntos, se puede proceder a intentar identificarlos.

6.3.4. Identificación del contenido de los conjuntos

Ahora se aborda la última fase del objetivo de este algoritmo: La identificación de objetos conocidos en un entorno de trabajo controlado.

Dado que ya se dispone de los objetos separados en diferentes nubes de puntos, la tarea consiste en averiguar el contenido de estas nubes. En este proyecto se han abordado solamente cilindros y esferas. Esto ha sido así en primer lugar porque el objetivo de esta parte del proyecto no es realizar un completo algoritmo de identificación, sino mostrar las capacidades del dispositivo RGB-D. Y en segundo lugar, los objetos seleccionados son éstos por su disponibilidad, y porque al ser objetos curvos, poseen una ecuación característica, lo que permite utilizar ciertos algoritmos muy apropiados para su identificación.

En esencia, la identificación se realiza en base a una comprobación de si lo que hay dentro de cada conjunto de puntos es un cilindro, una esfera o ninguno de los dos objetos. El algoritmo utilizado es de nuevo un algoritmo «RANSAC», en este caso con la ecuación del cilindro o la de la esfera, según lo que se desee comprobar.
Se ha esbozado su funcionamiento ligeramente en apartados anteriores, así que debido a su importancia en este apartado, se va a detallar un poco más el funcionamiento de este algoritmo, en concreto en el caso aplicado a una esfera. Los pasos que sigue son los siguientes:

1. Dado un conjunto de puntos, se seleccionan aleatoriamente cuatro de ellos. Con estos puntos se construye la ecuación de la esfera que los contiene. Para ello resuelve un

sistema de cuatro ecuaciones y cuatro incógnitas, obteniendo el centro de la esfera y su radio.

2. Con la ecuación de la esfera obtenida, se comprueba cuántos puntos del conjunto dado pertenecen a la esfera. Para ello se ha dado una distancia que actúa como umbral: si la distancia de un punto a la esfera en prueba es menor que el umbral, ese punto pertenece a la esfera.

3. Cuando se han probado todos los puntos, se almacenan las coordenadas de la esfera probada y la cantidad de puntos que pertenecen a la misma, y se pasa a crear otra ecuación de esfera y probarla. Este proceso se repite hasta que se alcanza un número de esferas a probar determinado por el usuario.

4. Terminada la prueba de las ecuaciones, se selecciona la esfera que más puntos tiene como pertenecientes, y se da como resultado.

La explicación anteriormente dada es igualmente válida para cilindros, solo que mediante la ecuación del cilindro. Además, este método, garantiza obtener (siempre que el número de repeticiones sea suficiente) la mayor esfera o cilindro de los existentes.

Una vez conocida la mayor esfera y el mayor cilindro existente en cada agrupación de puntos, se juzga si el objeto es suficientemente representativo de los puntos que contiene la nube. De una forma experimental, se ha ajustado esta cantidad de puntos necesarios, resultando en que al menos se necesita un 60 % de los puntos de un conjunto para que haya un cilindro dentro de él, y un 85 % o más cantidad de puntos pertenecientes a una esfera para que pueda existir una esfera en el conjunto analizado.

Cuando se detecta que dentro de un conjunto de puntos hay un objeto conocido, pueden darse principalmente dos situaciones:

Se identifica un solo objeto. Si la cantidad de puntos pertenecientes a un único objeto identificado respecto al total del *cluster* es superior al umbral, ese *cluster* se etiqueta como dicho objeto.

Se identifica más de un objeto. Suponiendo que ambas identificaciones superen su umbral mínimo, se considera que el objeto a etiquetar es el que mayor cantidad de puntos pertenecientes tenga. En caso de que la diferencia entre las cantidades de ambos puntos no sea muy grande, se considerará el objeto como confuso, y se etiquetará de forma especial considerando ambas posibilidades. La mínima diferencia necesaria para no considerar un objeto como confuso ha sido fijada experimentalmente y es del 5 % de los puntos.

En la figura 6.9, se pueden observar los objetos que ha separado claramente como cilindros teñidos de color azul oscuro. Los que considera claramente esferas se han teñido de verde brillante.
Por su parte, aquellos objetos confusos pero que el algoritmo cree que son más claramente esferas están teñidos de verde oscuro y los que considera más bien como cilindros están teñidos de azul claro.
Aquellos objetos que no ha considerado ni como esferas ni como cilindros no aparecen. Solo aparece el hueco (y la sombra) que dejan.

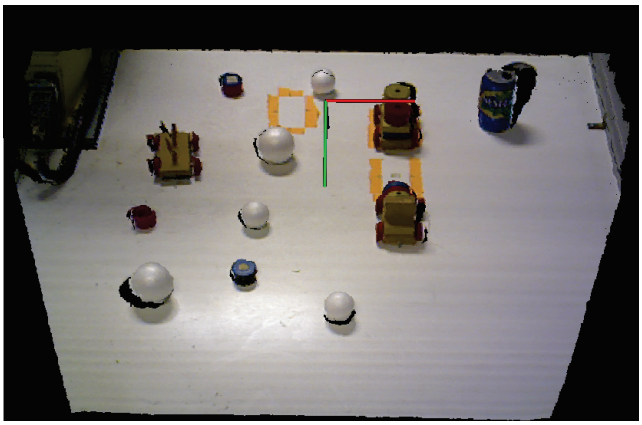
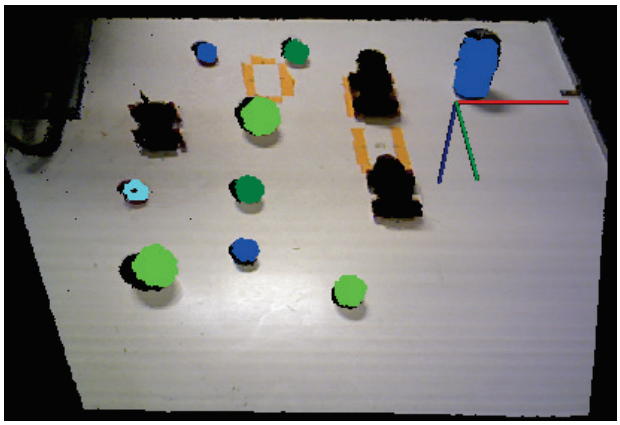

Figura 6.9: Imagen original, e identificación realizada por el algoritmo. Se aprecian objetos identificados, y huecos en aquellos que no se han reconocido.

6.3.5. Información adicional de color para los conjuntos

Como extensión final al algoritmo de identificación de objetos, se ha introducido una diferenciación del color de los objetos.
Para proceder al reconocimiento del color, se ha realizado una discretización en el rango que la cámara es capaz de captar en formato RGB. Originalmente la cámara proporciona 256 valores para cada uno de los tres canales. No obstante, el algoritmo realiza una simplificación de estos valores, permitiendo 3 valores por canal.
Los colores que se han implementado (se les ha proporcionado un nombre y se es capaz de distinguirlos) son los 27 colores diferentes que corresponden al total de los posibles tras la simplificación.

La función del algoritmo que se encarga de identificar los colores predominantes de un conjunto de puntos, y que ha sido íntegramente creada en el proyecto, comienza por hacer la discretización de los colores. Después prosigue haciendo un recuento del número de puntos que tienen cada tipo de color. Por último los ordena y guarda los tres colores mayoritarios, así como la cantidad relativa de cada uno de esos tres colores. En la figura 6.10 se muestra un ejemplo de cómo queda un objeto tras la rectificación del color.

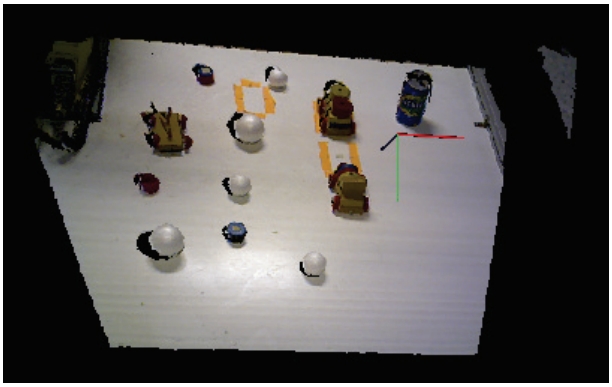
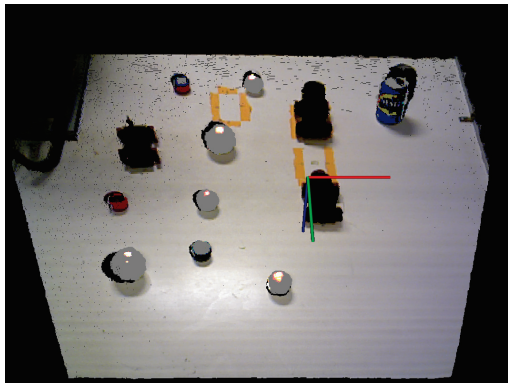

Figura 6.10: Imagen original, y discretización del color realizada por el algoritmo. Se aprecia la adecuada conversión a colores más puros sin perder apenas información.

6.3.6. Resultado en formato cualitativo y coordenadas

En éste último punto se describe cómo proporciona el algoritmo implementado la información que ha obtenido.
En primer lugar, el algoritmo muestra visualmente información de cómo está transcurriendo el proceso. Además también tiene la opción de realizar todas las operaciones paso a paso para poder comprobar posibles fallos.
La otra vía para proporcionar información de la que dispone el algoritmo, es la generación de un archivo de datos en formato de texto (ASCII), en el que plasma la información que ha obtenido, tanto cualitativa como cuantitativa. El archivo creado es de extensión «.KL», que es la extensión de los archivos de programa del robot. Con ello se pretende facilitar su uso por parte del programador del robot. La información cuantitativa viene dada por las coordenadas y tamaños de los objetos encontrados.La cualitativa la da la forma de dichos objetos, y su color.

Se incluye en la figura (6.11)un ejemplo demostrativo del fichero de datos que se obtiene a partir del empleo del algoritmo de reconocimiento de objetos creado en este proyecto.

Cluster number	Type	Center point: X	Y	Z	Radious	Axis direction			Main colours in cluster
1	Cylinder	0.295516	-0.891352	0.331982	0.03378	-0.00571337	0.0705223	0.997494Z	Black Navy Black
2	Sphere	-0.302318	-0.388177	0.662241	0.0387919				Gray
3	Sphere	-0.118736	-0.756692	0.662424	0.037082				Gray
4	Sphere	0.00478474	-0.348385	0.673122	0.0291972				Gray
5	Sphere-Cylinder	-0.1453	-0.578019	0.6757	0.0316894				Gray
6	Cylinder	0.283223	-0.514085	0.653958	0.0194874	0.981102	-0.180941	-0.0685449X	Gray Black Red
7	Cylinder-Sphere	-0.34104	-0.49619	0.7345	0.0121867	Y			Dark_red Black Dark_red
8	Sphere-Cylinder	-0.0355183	-0.973221	0.673084	0.0300498				Gray
9	Cylinder	-0.598452	-1.15291	0.674729	0.018039	0.897985	0.43998	0.00634764X	Dark_red Gray Lime
11	Sphere	0	0	0	0				White

Figura 6.11: Ejemplo del archivo de salida con los objetos que el algoritmo ha logrado identificar. Se incluyen el tipo de objeto, su posición, tamaño, y colores principales.

6.4. Algoritmo de control del espacio de trabajo

El otro algoritmo al que se hace mención en el apartado 6.1, es un detector de variaciones en espacios de trabajo controlados.
Su utilidad es manifiesta, ya que su propósito es comparar continuamente mientras esté conectado, la escena que capte a través de la cámara RGB-D, con una escena de referencia, y avisar cuando los cambios superen un umbral previamente establecido de riesgo.
Con ello se puede comprobar la aparición de elementos inesperados o no deseados, o cuándo un elemento de la zona de trabajo está fuera del sitio que le corresponde.

La forma de trabajar de este algoritmo es muy directa: En primer lugar, está continuamente obteniendo capturas de la escena, lo que le permite (dependiendo de la potencia de cálculo del computador asociado) un control de hasta 10 fotogramas por segundo.

Con cada escena, el proceso es efectivo y rápido. En primer lugar crea un árbol de vecindad de los puntos de la nube obtenida. Y posteriormente los compara con los presentes en la escena de referencia.
Los cambios detectados en los puntos (puntos que aparecen o desaparecen), son considerados como puntos cambiantes. Si la cantidad de puntos cambiantes es mayor que un umbral establecido por el usuario, salta una advertencia. y si supera un segundo umbral mayor, salta una alarma. Esta alarma podría servir para detener un proceso, o impedir su inicio, ya que la zona de trabajo no reúne las condiciones en principio previstas.

Un punto a destacar de este algoritmo es que es relativamente inmune a los cambios de luz, ya que no utiliza el color, sino solo la información de profundidad. No obstante, tiene cierta variabilidad con la iluminación, ya que cuanta más luz haya, mayores perturbaciones surgen en el patrón infrarrojo que la cámara emite, y más ruido aparece.

Precisamente debido al ruido existente, se decidió que el programa realizara una pequeña calibración del ruido durante las primeras capturas de escena. En concreto sobre las 100 primeras capturas.
Este ruido de referencia medido mediante el periodo de calibración, no es tenido en cuenta para el cómputo total de las variaciones. Esto es: Con respecto a la cantidad (ya expresada como un dato numérico) de variación detectada, el algoritmo eliminará la cantidad numérica que se ha definido como ruido.

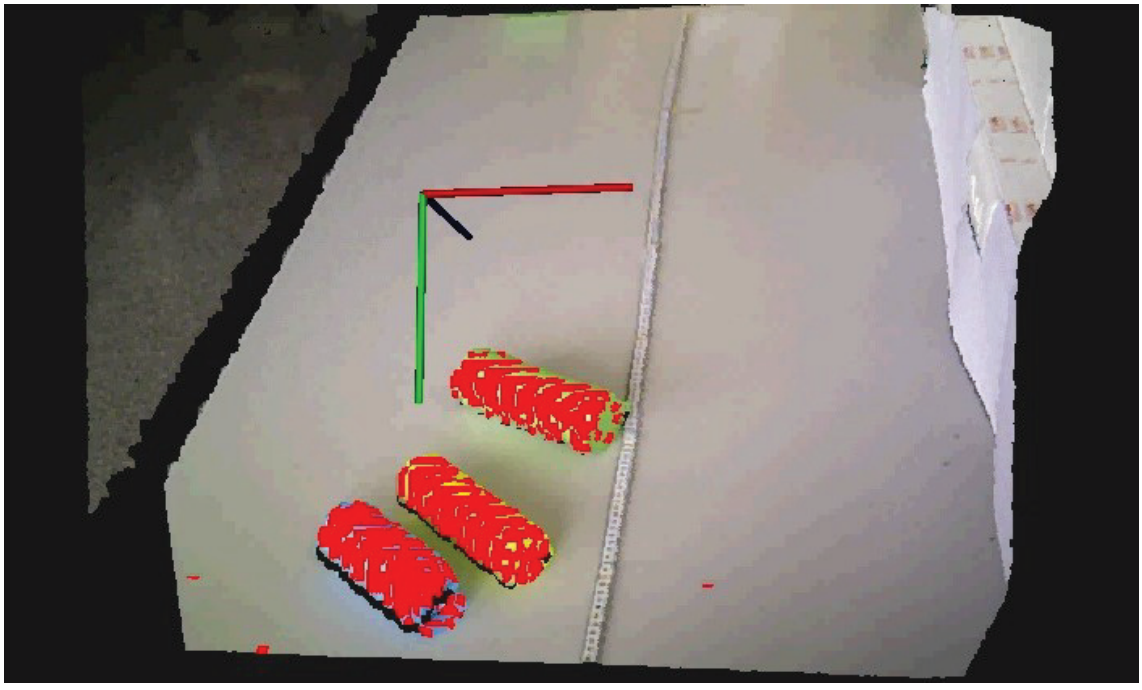

Figura 6.12: Ejemplo de la salida por pantalla del algoritmo de control del espacio de trabajo.

Como punto a ampliar de este algoritmo se propone su interconexión con otros dispositivos de seguridad tales como sensores de presencia, detectores de paso, puertas con

sensores... Ya que el algoritmo requiere ser reiniciado cada vez que se ha producido un cambio intencionado en la disposición de los elementos contenidos en el espacio controlado, o cada vez que en el espacio de trabajo se produzca un movimiento intencionado (como el funcionamiento de un robot).

En la figura 6.12 se puede apreciar un ejemplo del resultado de este algoritmo. En la imagen se muestran en rojo los puntos que son diferentes de los que se tenían como referencia. Se puede apreciar que corresponden claramente con los puntos pertenecientes a objetos que se han puesto ahí posteriormente a ser tomada la imagen de referencia, y por lo tanto se consideran como intrusiones en el espacio.

Capítulo 7

Simulador de robótica industrial

La simulación es una parte esencial en el proceso de diseño de los sistemas automáticos. Hasta ahora, en este proyecto se han abordado los diferentes niveles de implementación de la célula de fabricación, y la incorporación de un elemento adicional que proporciona información de superior nivel como la cámara RGB-D. Falta ahora tratar el nivel superior de diseño de la célula de fabricación mediante herramientas de simulación.

Este capítulo está dedicado a explicar el trabajo que se ha realizado en cuanto a la búsqueda, análisis y elección de un programa sobre simulación de robots, y la elaboración del material correspondiente para docencia sobre dicho simulador.

7.1. Motivación para la búsqueda de un simulador

La utilidad que tienen los simuladores no sólo de robótica, sino de cualquier materia en general, es de sobra conocida ya que ahorran tiempo y dinero a la hora de probar diferentes alternativas antes de proceder a diseños concretos en actividades de casi cualquier ámbito.

En el caso que a este proyecto atañe, el interés por encontrar un simulador actual de robótica viene dado por la intención de reemplazar, de cara a la docencia de las nuevas asignaturas de grado, al simulador «SG-Robot», que ha sido utilizado durante con gran éxito durante un largo periodo de tiempo como herramienta docente en asignaturas de robótica industrial.
Si bien no existía una urgente necesidad de realizar dicha búsqueda, sí es cierto que se conocía la existencia en el mercado de potentes simuladores de robots industriales. Además, el simulador *SG-Robot* empezaba a quedarse algo limitado respecto a los simuladores comerciales, y por lo tanto perdía parte de su valor como herramienta docente.

Por todo ello, se decidió realizar una búsqueda de un nuevo simulador que reuniera las características que habían hecho del *SG-Robot* una herramienta docente tan buena, tales como su fácil manejo, la posibilidad de crear nuevos escenarios, la buena simulación de las singularidades... Y que además, tuviera los parámetros que caracterizan a los más modernos simuladores, tales como un buen entorno gráfico, capacidad para utilizar diferentes robots, manejo de señales...

Posteriormente a esta búsqueda, sería momento de valorar los diferentes simuladores y elegir uno si se juzgaba adecuado, tal y como al final se hizo.

7.2. Estudio y comparación de simuladores

Tras haber definido los requisitos que se querían para el simulador llamado a reemplazar al *SG-Robot*, comenzó la búsqueda. Los requisitos mencionados son fundamentalmente los siguientes:

- Se buscaba un simulador cuyo fundamento fueran los brazos robóticos, pero sin dejar de tener en cuenta la posibilidad de simular otros formatos de robots industriales.
- Una interfaz gráfica actual, acorde con lo que se espera de un programa moderno. Con buen tratamiento gráfico, continuidad en la representación de los movimientos, y posibilidad de grabar vídeo.
- Posibilidad de crear objetos y escenarios de un modo que no fuera en exceso complicado, acorde con el manejo de los modernos programas de diseño 3D.
- Con posibilidad de realizar la programación tanto por guiado como por introducción de coordenadas vía código.
- En la medida de lo posible, que tuviera buena manejabilidad, y un entorno de programación amigable.
- Y por último, que tuviera disponibilidad gratuita, o al menos a un bajo coste para su uso como elemento docente en la Universidad.

Se encontraron múltiples simuladores, de los cuales solo los que más adelante se detallan son los que reunían en mayor medida las características deseadas.
Por supuesto, se encontraron muchos simuladores que eran en exceso simples, y otros de los cuales no se consiguió obtener ni una versión de prueba. Esto sucedió por ejemplo con dos potentes fabricantes de robots como *KUKA*, que dispone del software «KUKA sim PRO», y con *FANUC*, que dispone de su simulador «FANUC RoboGuide». A continuación se incluyen los programas que pasaron las primeras fases de la evaluación. De cada uno se dan algunos datos interesantes.

7.2.1. RoKiSim

El simulador «RoKiSim» (*Robot Kinematics Simulator*), es un simulador bastante sencillo de brazos robóticos, desarrollado por el Laboratorio de Control y Robótica de la *École de Technologie Supérieure* de la Universidad de Montreal [6]. Sus principales características son las siguientes:

- Dispone de varios modelos de robots populares (ABB, KUKA, STÄUBLI, etc).
- Permite introducir geometrías de objetos.

- Simula de una forma bastante buena las posiciones difíciles del robot (Más concretamente las singularidades).
- Es muy fácil de manejar, y está disponible en varios idiomas.

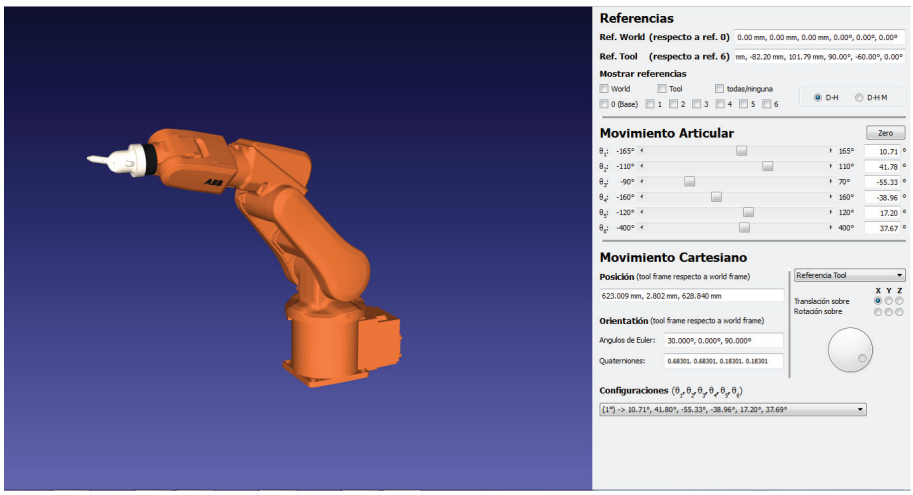

Figura 7.1: Muestra de la pantalla del programa RoKiSim. Se aprecian unos gráficos bien conseguidos y una interfaz lista para el uso.

A pesar de todas sus virtudes, y de ser totalmente gratuito, se terminó eligiendo otro por las aportaciones adicionales respecto al simulador *SG-Robot*, ya que *RoKiSim* en esencia permite las mismas simulaciones que *SG-Robot*, pero con una estética actualizada.

7.2.2. V-rep

El simulador «V-Rep» es un simulador desarrollado por la empresa Coppelia Robotics [7]. Se trata de un software, que en el estado en el que se da por parte del desarrollador, está más destinado al desarrollo e investigación en el sentido de que permite acceder a su código fuente y modificarlo, que a la simulación propiamente dicha.

Es un programa más avanzado que lo que se buscaba con software del estilo del simulador *SG-Robot*. Ya que si bien las posibilidades de simulación que ofrece el programa tal cual se da son menores, su potencial es muy grande a la hora de poder usar cualquier modelo de CAD, además de sus posibilidades en cuanto a poder reprogramarlo y crear nuevas interfaces gráficas, desarrollo dinámico, etc. Estas características lo convierten en un software apropiado a la hora de trabajar en proyectos más complejos y avanzados que unas prácticas de laboratorio.

Sus principales virtudes son las siguientes:

- Tiene un entorno gráfico bueno, amigable y con posibilidad de utilización de ventanas, pestañas, etc. Existentes o creadas.
- Licencia educativa que incluye la posibilidad de modificación del programa.
- Permite la utilización de hasta 6 lenguajes de programación (incluyendo C++ y MatLab).

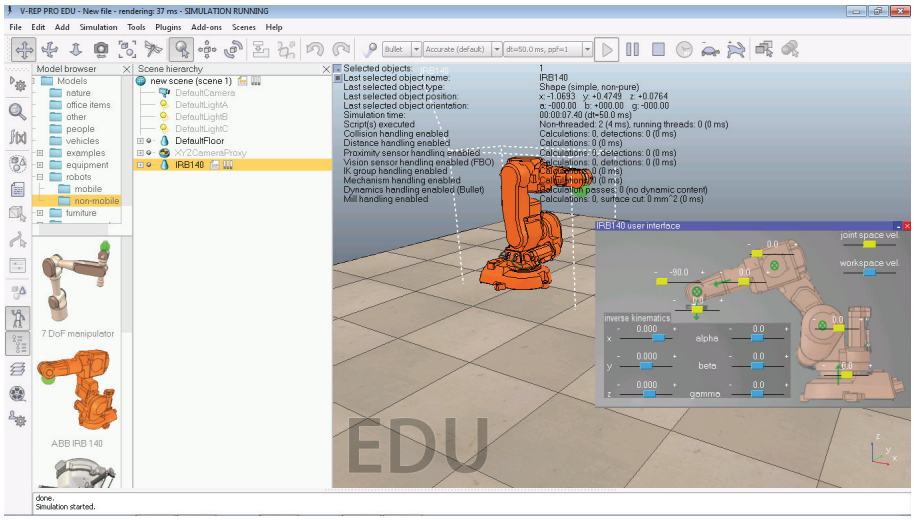

Figura 7.2: Ejemplo de la pantalla del simulador V-Rep. Se puede apreciar mayor complejidad que en otros simuladores.

- Permite simular más allá de robots industriales. Se pueden llegar a incluir robots con sensores de visión, de posición, además de todo objeto cuyo comportamiento se pueda modelar de manera analítica.

No obstante a ser un excelente simulador, también se descartó su uso debido fundamentalmente a su complicado uso, que hubiera requerido de cualquier alumno un tiempo excesivo para aprender su manejo. También hubiera sido necesario reprogramar partes de su código fuente para que hubiera servido adecuadamente a los objetivos iniciales.

7.2.3. Easy-Rob

El simulador de robots «Easy Rob» es otro de los programas que se logró obtener para su evaluación. Se trata de un simulador de robótica industrial de apariencia amigable, basado en un entorno de ventanas y pestañas. Dispone de múltiples botones y menús en los cuales están contenidas todas las funciones existentes en el programa, y que permite utilizarlo de una forma más o menos intuitiva.
La versión demostrativa incluye algunos robots simples y una herramienta de tipo cono. Más información en [8].

Sus puntos fuertes son:

- Entorno gráfico amigable y sencillo de usar.
- Facilidad de utilización y variedad en las vistas 3D y vista libre.
- Posibilidad de simular colisiones.
- Descuentos en la adquisición de licencias con fines educativos.

En este caso, se terminó por descartar este simulador por un lado porque de forma gratuita solo ofrecía una versión de prueba de 15 días, frente a otros programas que

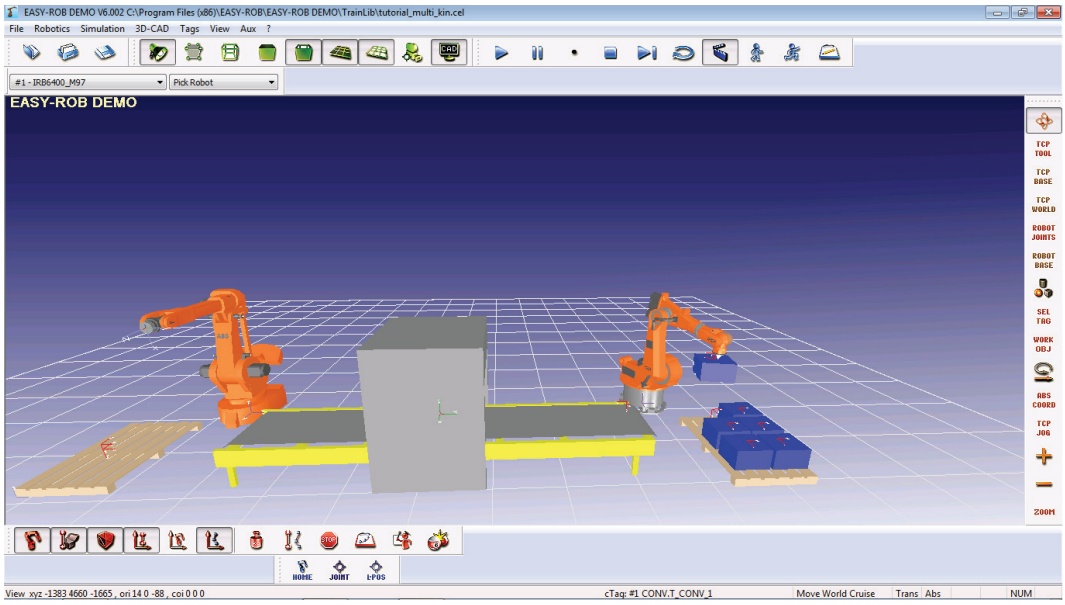

Figura 7.3: Vista general de la pantalla del simulador Easy-Rob. Se observa un buen tratamiento gráfico y múltiples opciones de acceso rápido.

daban versiones educativas gratuitas. Y por otra parte porque de nuevo la única ventaja clara respecto al simulador*SG-Robot* era la actualización de su apariencia.

7.3. ABB Robot-Studio

El programa que finalmente resultó elegido es el simulador «Robot Studio» del fabricante de robots industriales *ABB Robotics*.
Se trata de un simulador potente e intuitivo a la hora de utilizarlo, el cual está basado en imitar el comportamiento real de los robots industriales, para lo cual simula el control de los mismos mediante un «Controlador virtual» que simula todos los detalles de un controlador real de robot.

Entre otras cosas, permite la utilización de varios robots simultáneamente, así como otros elementos típicos de las industrias como cintas transportadoras, o cualquier elemento que se desee modelar mediante la parte del programa a ello dedicada.
Así mismo, permite crear de una manera rápida y simple diversos objetos inanimados para interactuar con el robot, tales como cajas, cilindros, cubos...
Además, también incluye la simulación de señales de entrada y de salida, así como la interacción entre los controladores de varios robots, al estilo de lo que sería una planta industrial real.
Por último, existe gran cantidad de material en internet acerca de este simulador, desde tutoriales hasta vídeos de demostración.

El simulador *Robot-Studio* fue finalmente el elegido por varias razones, pero en términos generales se pueden destacar dos:
A pesar de no ser el más destacado en manejabilidad, ni en posibles prestaciones, es el

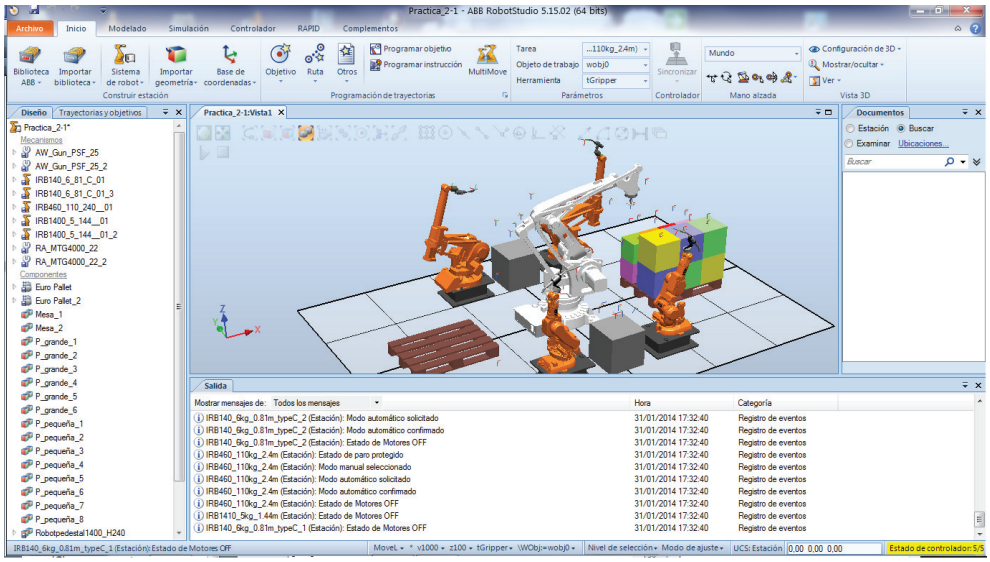

Figura 7.4: Vista de la pantalla principal del simulador Robot-Studio. Se aprecian diferentes pestañas, y se intuye el amplio rango de opciones que tiene.

que mejor aúna todas las características deseadas de los simuladores probados.
Además, es un simulador muy utilizado comercialmente, que se puede incluso usar para programar los robots del fabricante *ABB*.

Como último punto a destacar de este simulador, se han obtenido 100 licencias educativas para uso en la Universidad de Zaragoza.

7.3.1. Aprendizaje y preparación de material

El aprendizaje sobre el programa *Robot-Studio* se ha basado fundamentalmente en tres pilares: El manual del programa [2], diversos tutoriales y vídeos demostrativos existentes en internet, y el aprendizaje autodidacta en base a su utilización.
Con estas tres fuentes de conocimiento es con las que se han desarrollado tanto los manuales y tutoriales (Ver sección 7.4), como todas las simulaciones y pruebas intermedias hasta llegar a los resultados finales.

Como parte del trabajo preparatorio antes de poder obtener el plausible material docente, fue necesario realizar multitud de pruebas incluyendo: Diversos escenarios, utilización de hasta cinco robots al mismo tiempo, creación de objetos, ensayo de herramientas...

Tras realizar esas pruebas, la conclusión a la que se llegó fue la de crear dos tipos de escenarios diferentes. Por un lado escenarios orientados a montar diversos elementos que constituyen una pieza, incluyendo soldaduras, y colocación de pegamento. En general las tareas mencionadas son tareas que puede hacer un único robot de manera independiente, o integrado en una célula de producción con un único robot de manufactura.
El otro tipo de escenario está orientado más hacia el transporte de piezas, incluyendo acciones tales como paletizado, limpieza de objetos y transporte de un lugar a otro. En este caso son necesarios varios robots, y coordinarlos entre sí.

Como elementos comunes a ambos tipos de escenarios están la necesidad de estudiar los modelos, crear posiciones, guiar a los robots y programarlos.

7.4. Material docente resultante

En base a este simulador se ha desarrollado material suficiente para realizar un trabajo de asignatura o varias prácticas de laboratorio.
La documentación y otros elementos desarrollados son los siguientes:

- **Manual de utilización de Robot-Studio.** Es un manual para el aprendizaje guiado del manejo fundamental del programa. Está realizado en base a la experiencia propia y tomando como referencia el manual del programa suministrado por el fabricante[2]. El manual creado está disponible en el apéndice F.
- **Un guión autocontenido para la realización de una práctica con el simulador** *Robot-Studio*. También es válido para la realización de un trabajo de asignatura. Se puede consultar en el apéndice K
- Una serie de **vídeos tutoriales** que ilustran totalmente la utilización básica del simulador.
- **Cuatro escenarios** diferentes y totalmente preparados para su utilización con el guión de prácticas ya mencionado.

Hay que destacar que este material comentado, ya ha sido utilizado durante la impartición de la asignatura «Automática Flexible y Robótica», del Grado en Tecnologías Industriales. Se sabe además que aparte de las dificultades surgidas por ser la primera vez que se utiliza, el simulador ha tenido una buena aceptación entre los alumnos. Se han dado además casos de alumnos que han querido ampliar los requisitos mínimos que se pedían, realizando simulaciones de procesos industriales con robots industriales considerablemente avanzados.
Con ello se quiere resaltar la utilidad del trabajo realizado a lo largo de este proyecto y su inmediata y exitosa aplicación.

Capítulo 8

Conclusiones

El presente proyecto ha servido, además de como es de esperar, para la aprehensión de conocimientos y nuevas habilidades por parte del proyectando, para obtener abundante y útil material de cara a la docencia a corto plazo de asignaturas que tengan contenidos similares a los tratados a lo largo del proyecto, es decir: Asignaturas relacionadas con la robótica industrial y la automatización. No sin olvidar el desarrollo y documentación de dos básicos algoritmos para cámaras *RGB-D*, que puede servir de base para futuros proyectos que se quieran centrar en la visión por computador mediante información de profundidad y color.

A lo largo de este variado proyecto, se han explorado muy diferentes ramas dentro de la Automatización Industrial y la Robótica, desde la programación de autómatas, hasta la simulación de robots, pasando por la obtención de información en base a la visión artificial y las comunicaciones entre los componentes de una célula industrial.

Terminado el desarrollo del proyecto, es un buen momento para tratar de dar una visión de conjunto de todo lo que se ha logrado, y cómo se interrelacionan las diferentes partes del mismo.

- Partiendo de los elementos presentes en el laboratorio L 0.6 del edificio *Ada Byron*, se ha realizado la **programación e integración** de una parte del conjunto **de la célula de fabricación** allí presente para adaptarla a los nuevos planteamientos que se deseaban desde un punto de vista docente. De esta manera se han realizado programas para dos autómatas y un robot industrial (cada uno en su propia plataforma y con su propio lenguaje de programación) de tal manera que puedan funcionar tanto de manera conjunta como por separado. Dichos programas han sido realizados teniendo especial cuidado en la seguridad, y en que fueran suficientemente representativos de las capacidades de la célula robotizada.
  En base a ese trabajo con una parte de la célula de fabricación, se han preparado **diversos manuales sobre su uso y programación**, y se han propuesto **guiones para la realización de prácticas** de laboratorio relativas al trabajo realizado, algunos de los cuales se han llevado ya a la práctica en asignaturas de grado.
- Para hacer más accesible, y en cierta forma completar la programación realizada, haciendo que la maqueta sea más parecida si cabe a una célula industrial, se ha rea-

lizado la **programación de dos terminales táctiles de control y supervisión** para dichos autómatas.
De esta programación de terminales táctiles, también se ha obtenido material docente consistente en una propuesta de **práctica de laboratorio sobre la implementación de terminales de diálogo** para tres de las estaciones de trabajo de la célula.

- Sin salirse del ámbito de la robótica industrial, pero pasando a otra de las disciplinas que la nutren, se ha utilizado una cámara *RGB-D* mediante el desarrollo de dos algoritmos, para realizar la supervisión de seguridad y la localización de objetos predeterminados, en el espacio de trabajo del robot industrial anteriormente nombrado.
  Con estos programas se ha comprobado la gran utilidad que pueden llegar a tener estas sencillas y económicas cámaras como elemento de apoyo a la robótica industrial. Aunque lógicamente, debido a que los algoritmos no se han desarrollado de una forma profesional en cuanto a recursos y tiempo, y a la precisión de la cámara, los resultados no son válidos para emplear a un nivel industrial en este mismo momento. Sí que se ha constatado que evolucionando un poco más los programas desarrollados, podría en un plazo breve obtenerse un resultado suficientemente preciso y robusto como para su uso en ambientes controlados.

- Por último, y como complemento a todo el trabajo realizado desde el punto de vista de la simulación, se ha buscado, evaluado y seleccionado un simulador de robots, del cual se han conseguido licencias gratuitas para uso en la Universidad.
  Con ello se ha pretendido que esté disponible una herramienta potente y versátil para poder demostrar de manera segura cómo programar un robot industrial, y cómo modelar diferentes situaciones reales en las que intervienen.
  Del simulador finalmente seleccionado (ABB Robot-Studio), se han obtenido para uso docente por parte de la Universidad de Zaragoza, 100 licencias completas.
  También se han desarrollado diferentes escenarios para el simulador, así como una serie de vídeos tutoriales, un manual de aprendizaje guiado del mismo, y un guión apto para prácticas de laboratorio o un trabajo de asignatura. Dicho trabajo, en base a un guión adaptado del propuesto, ya ha sido utilizado en una asignatura de grado, resultando en una gran satisfacción de los alumnos.

En consonancia con los objetivos iniciales del proyecto, se han realizado diferentes propuestas de guiones de prácticas, tal y como se ha indicado en los capítulos que les corresponden. Es destacable el hecho de que algunas de estas propuestas de guión, hayan sido ya utilizadas (debidamente adaptadas) en prácticas de asignaturas de grado, tal y como se ha comentado anteriormente. Realzando con ello la componente más inmediata de los resultados de este proyecto.
Es de esperar que con el completo desarrollo de otras asignaturas de grado relacionadas con la robótica industrial y la automatización, pueda aprovecharse una parte muy importante de las diferentes propuestas realizadas a lo largo de este proyecto.

8.1. Líneas abiertas para trabajar

Con la finalización de este proyecto no quedan cerradas en modo alguno las líneas sobre las que se ha venido trabajando a lo largo del mismo. Como muestra de ello se detallan a continuación unas cuantas de las posibilidades que quedan abiertas al finalizar este proyecto, algunas de las cuales pueden considerarse consecuencia directa de la elaboración del mismo.

- **Programación alternativa, y/o ampliación de la programación** realizada sobre el robot y dos de los autómatas de la célula de fabricación del laboratorio L0.6 del edificio *Ada Byron*.
  Todavía quedan cuatro estaciones de manufactura, dos de almacenamiento y una de transporte que se pueden integrar de nuevo en el conjunto realizado durante este proyecto. Hay que señalar que en cierta medida dependerá de los recursos disponibles para la actualización del resto de componentes de la célula.
  Como facilidad para ello, en esta memoria se dan importantes detalles sobre cómo utilizar algunos componentes comunes, y en los apéndices se detalla con mucha mayor profundidad cómo realizar la programación.

- **Desarrollo en profundidad tanto del algoritmo de identificación de objetos como del algoritmo de supervisión de seguridad.**
  Ambos algoritmos se han desarrollado durante el presente proyecto como meros ejemplos del potencial que tiene este tipo de cámaras. Por ello todavía ofrecen ambos algoritmos unas grandes posibilidades de ampliación y mejora. Además, también existe la posibilidad de utilizar esta tecnología para muchos otros usos que no se han descrito aquí, siendo de especial interés la comunicación directa entre un ordenador y el robot.
  Para todo ello, los detalles sobre la programación contenidos en la memoria de este proyecto pueden ser de gran utilidad.

- **Mejora, revisión y ampliación del material creado para el uso del simulador *Robot-Studio*.**
  Las posibilidades que ofrece el simulador seleccionado son extremadamente amplias, por lo que se invita a continuar con el aprendizaje de este software, ya que es un programa de uso real en el mundo industrial, y que tiene complejas e importantes aplicaciones tales como el uso sincronizado de robots para sostener y trabajar sobre la misma pieza. No sin olvidar que es un programa comercial, continuamente actualizado, y con prestigio en el ámbito de la robótica industrial.

Índice de figuras

Índice de tablas

Bibliografía

[1] ABB-Robotics. Abb robot-studio (http://new.abb.com/products/robotics/robotstudio).

[2] ABB-Robotics. *Manual del operador. Robot Studio 5.15.*, 2013.

[3] ASUSTeK-Computer-Inc. Asus xtion pro live. (http://www.asus.com/multimedia/xtion_pro_live/), url = http://www.asus.com/multimedia/Xtion_pro_live/,.

[4] Barrientos, A., Peñin, L. F., Balaguer, C., and Aracil, R. Fundamentos de robótica. Mc-Graw-Hill, Ed.

[5] Beltrán, R., and Romeo, A. Programa tareak, 2003.

[6] École Technologie-Supérieure-Université-Montreal. Información y descarga del programa rokisim (http://www.parallemic.org/rokisim.html).

[7] Coppelia-Robotics. Información y descarga del programa v-rep (http://www.coppeliarobotics.com/).

[8] Easy-Rob-Inc. Información y descarga del programa easy rob (http://www.easy-rob.com/easy-rob/).

[9] FANUC-Robotics. *R–J3iB Mate Controller LR Handling Tool Operator's Manual*.

[10] FANUC-Robotics. *SYSTEM RJ-3 Controller KAREL Reference Manual*.

[11] FANUC-Robotics. *R-J3 Controller CE Maintenance*, 2000.

[12] FANUC-Robotics. *Curso de programación TPE*, 2003.

[13] Fischler, M. A., and Bolles, R. C. Random sample consensus: A paradigm for model fitting with apphcatlons to image analysis and automated cartography. In *Magazine Communications of the ACM* (1981).

[14] OpenNI-organization. Openni libraries (http://www.openni.org/), url=http://www.openni.org/,.

[15] robots.com. http://www.robots.com/fanuc/arcmate-50il.

[16] Rusu, R. B., and Cousins, S. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, May 9-13 2011).

[17] Rusu, R. B., O'Leary, G., and Rublee, E. Point cloud library documentation (www.pointclouds.org).

[18] Schneider-Electric. Información sobre el autómata modicon m340 (http://www.schneider-electric.com/products/co/ls/3900-pac-plc-y-otros-controladores/3950-pacs/1468-modicon-m340/).

[19] Schneider-Electric. Información sobre el autómata tsx premium (http://www.schneider-electric.com/products/ww/en/3900-pac-plc-other-controllers/3950-pacs/537-modicon-premium/).

[20] Schneider-Electric. Información sobre el módulo de entradas y salidas distribuidas «advantys stb» (http://www.schneider-electric.com/products/mx/ls/3900-pac-plc-y-otros-controladores/3930-i-o-distribuida/606-advantys-stb/).

[21] Schneider-Electric. Magelis xbtgt4330 (http://www.schneider-electric.com/site/home/index.cfm/es/).

[22] Schneider-Electric. *Advantys STB, módulo de interfaz de red CANopen básico. Manual de aplicaciones*, 2009.

[23] Schneider-Electric. *Advantys STB, módulos de E/S digitales. Manual de referencia*, 2009.

[24] Schneider-Electric. *Modicon M340 para Ethernet. Procesadores y módulos de comunicaciones. Manual de usuario*, 2009.

[25] Schneider-Electric. *Unity Pro. Lenguajes y estructura del programa. Manual de referencia*, 2009.

[26] Tuerker, S. Euclidean cluster extraction (www.pointclouds.org/).

[27] Universidad-Zaragoza. Sistemas industriales de control. práctica 4. utilización de una red industrial de comunicaciones. programación de una interfaz humano-máquina., 2013.