Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

DEPARTAMENTO DE INGENIERÍA
DE SISTEMAS Y AUTOMÁTICA

GRADO EN INGENIERÍA

EN TECNOLOGÍA INDUSTRIAL

TRABAJO FIN DE GRADO

DISEÑO MODULAR DE CÉLULA FMS-200 PARA GENERACIÓN AUTOMÁTICA DE PROYECTOS DE INGENIERÍA

Alumno / Alumna: HERRÁN ANGULO, ASIER

Director / Directora (1): ORIVE REVILLAS, DARÍO

Curso: 2018-2019

Fecha: Bilbao, 18 de septiembre del 2018

ÍNDICE

RESUMEN

En el presente proyecto se lleva a cabo la modularización en módulos mecatrónicos de la célula de montaje FMS-200. La modularización es una filosofía de automatización que está tomando cada vez más fuerza en el ámbito industrial, formando parte incluso de la llamada Industria 4.0. La gran flexibilidad que aporta es clave a la hora de responder a las demandas de una sociedad que exige productos cada vez más especializados y personalizados. Es por ello, que en este proyecto se pone a prueba dicha filosofía analizando sus ventajas e inconvenientes, así como sus posibilidades dentro del mundo industrial actual.

En el proyecto se detallan los pasos a seguir para realizar la modularización, desde el diseño y selección de los módulos hasta la generación del proyecto modularizado mediante un programa de generación automática de proyectos de ingeniería.

Por último, y para comprobar la funcionalidad del proyecto y de la propia filosofía, se realizan pruebas tanto en la propia célula de montaje como en la herramienta de simulación NX-MCD.

LABURPENA

Proiektu honen bidez FMS-200 muntatze zelularen modularizazioa egiten da modulu mekatroniekoetan. Modularizazioa industria arloan gero eta indar handiagoa hartzen ari den automatizazio filosofia bat da, eta industria 4.0 edo laugarren iraultza industrialaren barruan sartzen da. Modularizarioak ematen duen malgutasuna garrantzi handia du gero eta produktu pertsonalizatuagoak eta espezializatuagoak eskatzen ari den gizartearen eskaerei erantzuteko. Horregatik, proiektu honetan filosofia hau proban jartzen dugu, bere abantailak eta desabantailak aztertuz, eta baita gaur egundo industria munduan dituen aukerak ikertuz.

Proiektuak, beraz, modularizaria gauzatzeko eman behar diren urratsak zehazten ditu, moduluen autaketa eta disenuaz hasita eta proiektua sorkuntza automatikoko tresna batekin programatzera arte.

Azkenik, proiektuaren era filosofiaren funtsionaltasuna egiaztatzeko probak egiten dira, proiektua muntatze zelulan eta NX-MCD sumulazio tresnan saiatuz.

ABSTRACT

The objective of this project is to modularize into mechatronic modules the FMS-200 assembly cell. Modularization is an automation philosophy that is becoming more and more imoprtant in the industrial world, even becoming part of the so-called Industry 4.0. The The great flexibility it provides is really important to answer to the needs of a society that demands increasingly specialized and personalized products. That is why, in this project, this philosophy is tested by analyzing its advantages and disadvantages, as well as its possibilities within the current industrial world.

The project details the steps to be followed to carry out the modularization, starting with the design and selection of the modules and ending by creating the project using an automatic generation program.

Finally, to check the functionality of the project and the philosophy itself, the project is tested in the assembly cell itself and in NX-MCD simulation program.

LISTA DE TABLAS, ILUSTRACIONES Y ACRÓNIMOS

LISTA DE TABLAS

LISTA DE ILUSTRACIONES

LISTA DE ACRÓNIMOS

TFG: Trabajo de fin de grado

PLC: Controlador Lógico Programable (Programmable Logic Computer)

TIA PORTAL: Totally Integrated Automation Portal

FMS: Sistema de ensamblaje flexible (Flexible Manufacturing System)

MM: Módulo mecatrónico.

OB: Bloque de organización (Organization Block)

FB: Bloque de función (Funcion Block)

FC: Función

DB: Bloque de datos (Data Block)

I/O: Entrada/Salida (Input/Output)

FMS: Sistema de ensamblaje flexible (Flexible Manufacturing System)

IP: Protocolo de Internet (Internet Protocol)

INTRODUCCIÓN

La automatización está plenamente instalada dentro del mundo industrial, tanto es así que uno de los objetivos de la Industria 4.0 es el de lograr empresas completamente automatizadas e inteligentes. Sin embargo, la sociedad avanza y con ello sus necesidades, y es por ello que en el ámbito de la atomatización industrial se están desarrollando nuevas filosofías de trabajo más eficientes.

Una de estas filosofías es la modularización de las células de montaje en módulos mecatrónicos, ya que, de esta forma, aumentaría su flexibilidad y fiabilidad.

Por ello este proyecto tiene como finalidad la modularización de la célula FMS-200, para de este modo y mediante la utilización de un programa de generación automática de proyectos de ingeniería, analizar los pros y contras de esta filosofía.

CONTEXTO

La idea de la modularización en módulos mecatrónicos surge dentro de un proyecto para la creación de un programa que permita la generación proyectos de ingeniería de forma automática.

Este proyecto a su vez, nace dentro de la llamada Industria 4.0 que es una nueva manera de organizar los modelos de producción adaptándolos a las nuevas necesidades de la sociedad actual.

Pero para entenderlo correctamente debemos comprender la situación en la que se encuentra el mundo industrial en la actualidad.

Hoy en día, debido a la globalización, vivimos en una época en la que la competencia empresarial es enorme y de carácter internacional, y en la que la oferta producida por las diferentes empresas es mucho mayor que la demanda generada por los clientes. Esto hace que las empresas tengan que reducir sus costes de fabricación al máximo para poder ofrecer sus productos a precios competitivos.

Además, los clientes exigen cada vez productos más personalizados, lo que, aunque aumenta su valor, hace que su producción sea mucho más compleja y cara.

Por ello, la tendencia es que las máquinas de producción sean cada vez más flexibles, para lograr así satisfacer las demandas de una sociedad cada vez más exigente.

Así pues, uno de los desafíos que se presentan es la programación de las máquinas para lograr esta flexibilidad de producción, tanto a nivel de cantidad como de poder crear productos personalizados. Especialmente a la hora de modificar o incrementar la producción mediante la implementación de otra máquina o línea de montaje, ya que para ello es necesario detener la producción durante el tiempo que dure la puesta en marcha de la nueva línea de producción, lo que supone la pérdida de dinero.

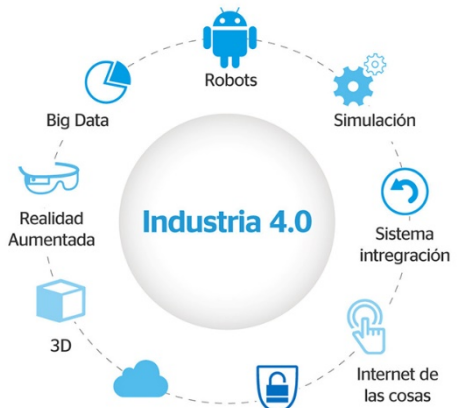

Ilustración 1.- Esquema Industria 4.0

Es por ello que surge la idea de la creación de un programa que permita agilizar este proceso ayudándonos en la generación del código de programación y permitiéndonos así el ahorro de tiempo, y, por consiguiente, de dinero.

Esta idea se desarrollará en el departamento Ingeniería de Sistemas y Automática de la Universidad Técnica Superior de Ingeniería de Bilbao durante el curso académico 2017-2018 por un grupo de estudiantes de dicha escuela.

Dicho proyecto se llevará a cabo por 3 equipos que trabajarán en paralelo. Por un lado, se desarrollará el programa de generación automático, por otro se modelará y simulará la célula de montaje donde se probará, y, por último, el correspondiente a este trabajo, que será la programación de la célula de montaje en módulos mecatrónicos independientes.

OBJETIVO Y ALCANCE

El objetivo del Trabajo es la modularización de la célula de montaje FMS 200 en aras de comprobar en ella el correcto funcionamiento del programa de generación de código automático.

La célula está compuesta por cinco estaciones, no obstante, en este proyecto nos centraremos exclusivamente en la modularización de la tercera y la cuarta.

Para ello, se parte de un código plenamente funcional de dicha célula realizado con la herramienta de automatización industrial TIA Portal. En este código se toma la célula de montaje como un único elemento dividido en 5 estaciones que realizarán su trabajo de forma secuencial.

Lo que se pretende lograr con el nuevo código es la división de cada una de las estaciones en módulos mecatrónicos independientes entre sí. Cada uno de estos módulos tendrá unas variables entrada y unas variables salida que le permitirán relacionarse con los demás, pero estará programado de forma independiente, por lo que podremos actuar sobre él sin necesidad de modificar el resto de la estación.

Esto nos permite convertir la célula de montaje en un elemento mucho más flexible, ya que al poder actuar sobre cada uno de sus módulos independientemente, podremos modificar los productos a fabricar por esta de una forma mucho más rápida y sencilla que si estuviese programada como un único elemento.

Una vez modularizada la estación, se utilizará el programa de generación automática y se generará el código de la estación completa, pudiendo así comprobar la funcionalidad del proyecto tanto en la célula FMS 200 como en simulación mediante NX MCD.

Ilustración 2.- Célula FMS-200

BENEFICIOS QUE APORTA EL PROYECTO

La modularización aporta beneficios en distintos ámbitos. En este apartado se analizarán los beneficios que aporta tanto a nivel técnico como económico.

ASPECTO TÉCNICO

Bajo un punto de vista local, la programación de una célula de montaje dividiéndola en diferentes módulos mecatrónicos independientes, nos aporta una flexibilidad que una célula programa de forma convencional no tiene.

La mayor ventaja es que al tener cada uno de los módulos funcionando de manera independiente, es posible modificarlos, intercambiarlos o incluso apagarlos y añadir módulos nuevos y que el resto de la célula siga funcionando de la forma programada, realizando únicamente pequeñas variaciones (principalmente, cuadrar las variables de entrada y de salida) y, permitiéndonos así variar la secuencia de montaje actuando únicamente en los módulos que se quieren modificar, y no en la totalidad de la célula como en la programación convencional.

Otra ventaja es la posibilidad de reutilizar el código, es decir, al seccionar cada una de las tareas en diversas funciones, el programador podrá reutilizar este código cada vez que quiera realizar esta tarea de nuevo, sin necesidad de volver a escribirlo. Si el código no estuviese seccionado en partes discretas, esto sería imposible.

Además, gracias a la independencia de la que disfruta cada uno de los módulos, la localización de los fallos será más sencilla, al igual que la reparación de estos.

En definitiva, la modularización aporta mayor versatilidad y flexibilidad a la célula de montaje, así como mayor fiabilidad.

ÁMBITO ECONÓMICO

La modularización es una técnica que puede aportar diversos beneficios también a nivel industrial y empresarial.

La flexibilidad que la modularización nos aporta, es clave en la industria actual, ya que, como se ha explicado anteriormente, los clientes buscan cada vez productos más personalizados, por lo que el tener máquinas con la capacidad de realizar un amplio catálogo de productos será crucial para poder satisfacer las necesidades de la sociedad y seguir siendo competitivos.

Aparte de a nivel de flexibilidad en la producción, la modularización nos aporta ventajas a la hora de variar las cantidades que queremos producir, ya que, por ejemplo, a la hora de duplicar la producción mediante otra célula de montaje, la puesta en marcha de la

misma será mucho más sencilla y rápida si esta está modularizada. Y, como es evidente, cuanto menos tiempo tarde la puesta en marcha, menos dinero supondrá.

En conclusión, permite tener un mayor control de la producción, algo absolutamente imprescindible en el mundo industrial actual.

ESTADO DEL ARTE

El ser humano, desde sus orígenes, ha utilizado herramientas y máquinas simples que le han permitido desarrollar trabajos realizando esfuerzos menores. Estas máquinas han ido evolucionando a medida que lo hacían las necesidades de la sociedad, incorporándoseles fuentes de poder externas, como podrían ser el viento o el cauce de un río, y formas de automatización controladas por elementos mecánicos de relojería y similares.

Ya en el siglo XIX la automatización se encontraba a pequeña escala, utilizándose en la realización de tareas sencillas de manufactura, como, por ejemplo, el telar automático de Joseph Marie Jacquard que revolucionó la industrial textil. No obstante, no fue hasta mediados del siglo XX cuando comenzó a tomar un papel clave en la industria.

En 1947, los físicos John Bardeen, Walter Brattain y William Shokkley desarrollaron el primer transistor, y 5 años más tarde Heinrich Grünebaum haría lo propio con el motor Alquist, el primer motor controlado. El 1959 surgió la primera máquina controlada por computador, el controlador SIMATIC que controlaba un torno Capstan. La electrónica de potencia no había aparecido todavía, por lo que se utilizaban relés y rectificadores para controlar los elementos de la máquina. Sin embargo, estos sistemas no eran demasiado eficientes, y su reemplazo era muy costoso, por lo que en 1968 apareció el PLC (Controlador Lógico Programable) que es la herramienta más utilizada actualmente para el control de procesos automatizados.

Ilustración 3.-Modicon 084. El primer PLC

El PLC es una computadora que permite automatizar procesos electromecánicos. Sus principales ventajas respecto a los sistemas basados en relés son su mayor tiempo de vida, su resistencia a los ambientes desfavorables, y sobre todo su facilidad para ser programados.

Desde su creación, los PLCs han evolucionado enormemente. En los primeros años los microprocesadores eran lentos, y únicamente podían compararse con PLCs pequeños, no obstante, a medida que los microprocesadores se desarrollaros y se convirtieron en más veloces, los PLCs más grandes comenzaron a basarse en ellos, aumentando así también su velocidad de respuesta.

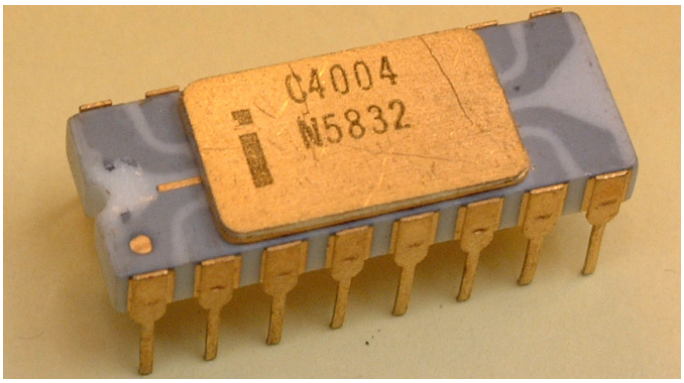

Ilustración 4.- Intel 4004. El primer microprocesador.

En 1973 aparecieron los primeros PLCs con la capacidad de comunicarse entre sí, los Modbus de Modicon. Estos sistemas incluso podían estar alejados de la maquinaria que controlaban.

En los años 80 los componentes electrónicos permitieron un conjunto de operaciones en 16 bits, en lugar de los 4 bits que se utilizaban en los 70, lo que los popularizó en todo el mundo.

Además, se produjo un intento de estandarización de sus comunicaciones por parte de General Motors mediante el protocolo MAP (Manufacturing Automotion Protocol).

El tamaño de los PLCs ha ido disminuyendo desde entonces, a medida que sus capacidades aumentaban, sobre todo con la aparición de los microprocesadores de 32 bits en los 90.

Hoy en día, la comunicación entre PLCs de diferentes marcas es posible, lo que permite la existencia de fábricas completamente automatizadas. Sin embargo, y aunque los PLCs sigue controlando el escenario industrial gracias a su versatilidad y pequeño tamaño, los PCs se han convertido en su gran competidor.

No obstante, no solo los PLCs y las máquinas han evolucionado, sino también la industria, y con ellos los medios de producción, y, por consiguiente, la manera de programar dichas máquinas.

Concretamente, y debido a la necesidad de satisfacer una demanda de productos cada vez más específicos y personalizados, surgió la idea de la Industria 4.0.

Este concepto expresa la idea de que el mundo se encuentra a las puertas de una cuarta revolución industrial. Fue manejado por vez primera en 2011 en la feria de tecnología industrial de Hanover, pero desde entonces ha sido defendido por diversos grupos y empresas en todo el mundo.

La Industria 4.0 busca la completa digitalización de las cadenas de montaje a través la integración de diferentes tipos de tecnologías (procesamiento de datos, software inteligente, sensores…) para así poder predecir, y por consiguiente planear y producir de la manera más óptima posible.

Así pues, nos encontramos en un momento de mejora continua de las tecnologías, donde cualquier mejora en la cadena de valor puede ser un factor diferencial a la hora de tener éxito. Y la tendencia general es que todas las empresas avanzarán hacia procesos cada vez más automatizados y de una manera más avanzada e inteligente.

Ilustración 5.- PLC S7 300

CONCEPTOS PREVIOS

En este apartado, se proporciona información general de la célula FMS 200 y del funcionamiento del programa de generación automática.

CÉLULA FMS 200

La célula FMS 200 es un sistema didáctico modular de ensamblaje flexible. Diseñada por SMC, cuenta con hasta 10 estaciones independientes que, según su disposición, permiten realizar gran variedad de conjuntos. Es lógico, por tanto, pensar en una programación modular, ya que de esta forma una vez programados todos los módulos, simplemente habría que generar el proyecto deseado mediante el programa de generación automática. Sin necesidad de programar toda la célula con cada configuración.

En nuestro caso, disponemos de 5 estaciones:

- FMS 201: Alimentación de la base.

- FMS 202 y 204: Inserción del rodamiento y del eje.

- FMS 205: Inserción de la tapa.

- FMS 208: Almacén automático.

- Estación de transporte del pallet vacío.

Ilustración 6.- Célula FMS 200

PROGRAMA DE GENERACIÓN AUTOMÁTICA

El programa de generación automática, se encarga de, partiendo del código de los módulos mecatrónicos, generar el proyecto completo en TIA Portal. La gran ventaja que tiene es que permite colocar los módulos mecatrónicos como se desee e incluso duplicarlos para utilizarlos repetidas veces en el proyecto o en otros proyectos.

Para ello, el primer paso que realiza es la exportación del software y hardware de cada módulo mediante el software TIA Portal Openess, guardándolo en ficheros .AML y .XML respectivamente. A continuación, estos ficheros se guardan en ExistDB, de forma que puedan ser accesibles desde otras herramientas. Así pues, el programa de generación automática puede acceder a dichos ficheros y comenzar el diseño del proyecto completo.

Cada módulo mecatrónico está relacionado con un esclavo, y tiene una IP y unas direcciones de entradas y salidas. Al reutilizar módulos, estos no pueden ser idénticos, es decir, no pueden tener las mismas variables y direcciones IP puesto que el código daría error al compilar. Para ello, el programa asigna prefijos cada variable, así como permite modificar su IP y direcciones de las entradas y salidas.

Una vez se diseña el proyecto correctamente y no se detectan errores, el programa genera un proyecto de TIA Portal con el código.

Cabe destacar que el programa ha sido diseñado para trabajar exclusivamente con el software de automatización de Siemens Tia Portal y utilizando conexiones tipo Profinet puesto que son las herramientas de las que se dispone en el laboratorio.

DISEÑO

Como se ha dicho anteriormente, el objetivo de este proyecto es la modularización en módulos mecatrónicos de la célula FMS-200 para probar la generación automática de proyectos de ingeniería. En este apartado, se describirán los pasos a seguir para lograr dicha modularización, así como se expondrá el resultado final.

En primer lugar, cabe destacar que, para la realización de este proyecto, se ha partido de un código plenamente funcional, pero sin modularizar, de la célula. Por lo que no se ha tenido que programar el código en sí, si no modificarlo separándolo en módulos mecatrónicos independientes y que fuesen compatibles con el programa de generación automática.

Así pues, el proceso del diseño del código puede dividirse en 2 partes. Por un lado, la elección de los módulos mecatrónicos en los que se va a dividir cada estación, y la realización de dicha división. Y, por otro, la adaptación del código para que sea compatible con el programa de generación automática.

Por otro lado, también se tiene que modificar el hardware de la estación. Cada módulo mecatrónico tiene que tener su propio esclavo, de forma que sea capaz de funcionar independientemente al resto de la célula.

DISEÑO DE LOS MÓDULOS MECATRÓNICOS

La primera decisión que hay que tomar a la hora de modularizar una célula es la elección de los módulos mecatrónicos en los que se va a separar.

En nuestro caso particular, la célula FMS-200 está dividida en cinco estaciones, realizando cada una de ella una parte concreta del proceso de montaje de la pieza.

Este proyecto se centra en las estaciones tercera y cuarta, que pese a realizar tareas diferentes, tienen una estructura similar. Ambas disponen de una cintra transportadora para llevar el pallet hasta la estación y sacarlo hacia la siguiente, una botonera y el bastidor de la estación.

Por lo tanto, estos han sido precisamente los módulos mecatrónicos elegidos, ya que el hecho de que la botonera y la cinta sean iguales en ambas estaciones, nos aporta, como se ha explicado en apartados anteriores, grandes ventajas. La más importante es que, dado que los módulos mecatrónicos funcionan de manera independiente entre sí, únicamente será necesario programarlos una vez y podrán ser utilizados en todas las estaciones que los requieran.

Gracias a esta elección, únicamente será necesario programar 4 módulos mecatrónicos: una botonera, una cinta y los dos bastidores. A diferencia de en el código anterior, donde las botoneras y cintas de transporte deben ser programadas en cada estación.

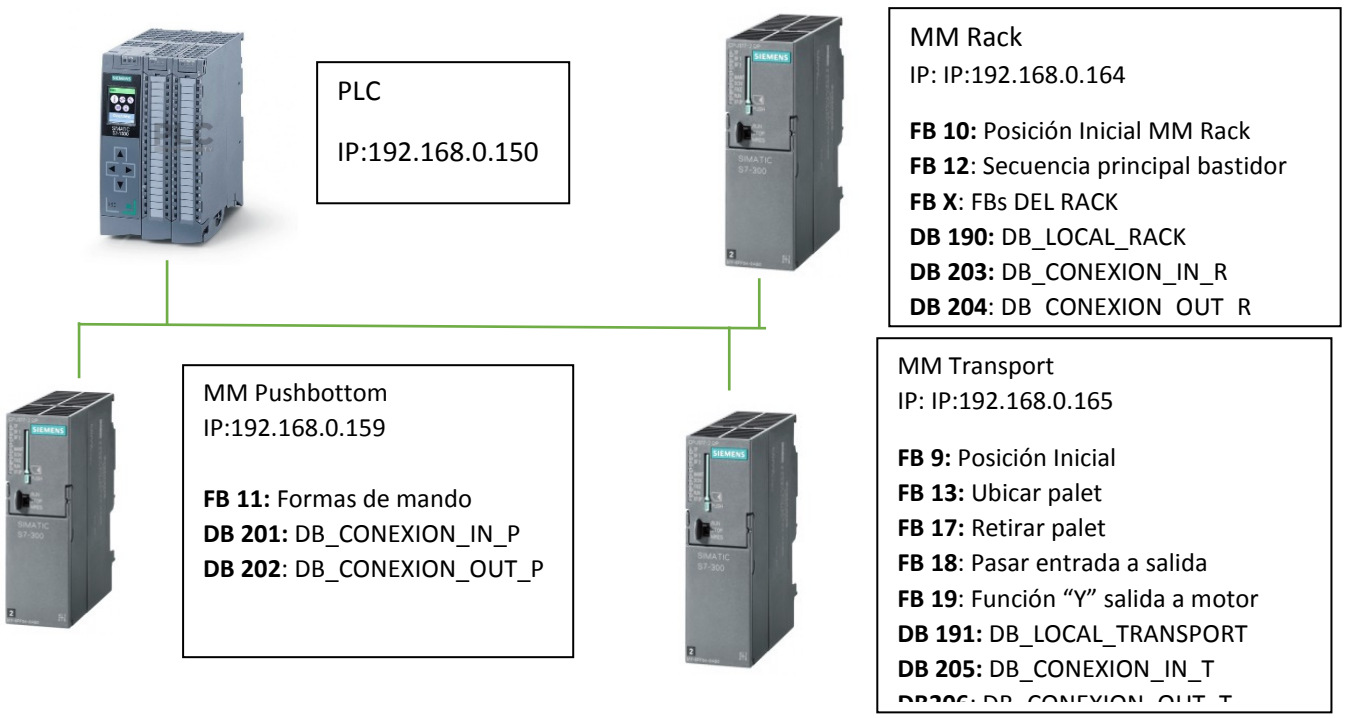

Ilustración 7.- Esquema conexión Módulos

ADAPTACIÓN DEL CÓDIGO

Una vez definidos los módulos mecatrónicos, debemos de programarlos de forma que el programa los acepte y pueda trabajar con ellos. Dado que partimos de un código ya programado, en nuestro caso, bastará con adaptarlo para que sea compatible con el programa.

En primer lugar, es necesario explicar la estructura utilizada para el código de las estaciones. Se ha optado por una estructura en la que desde un OB se llama a un FC sin parámetros, desde el cual se llama a cada uno de los diferentes FBs donde se recogen las secuencias que conformarán el código. Se ha escogido esta configuración por seguridad, ya que de esta forma aseguramos que el programa generador de código no detecte ningún problema.

Ilustración 8.- Esquema llamadas código

Este programa, modifica el nombre original de los FCs y FBs que genera, añadiéndoles prefijos. También hace lo propio con cada una de las variables globales del código, modificándolas directamente en la tabla de variables. Por tanto, en caso de no seguir esta estructura, es posible que el programa devolviese algún error. Por otro lado, y por comodidad, se adoptó la filosofía de trabajar sin marcas, es decir, todas las variables son pasadas por parámetros a través de DBs. Al eliminarse las marcas, el programa no tiene que añadir prefijos a todas ellas, ahorrándose así tiempo y espacio.

Sin embargo, en el código original no solo había llamadas a estos FBs, sino que también había segmentos con sentencias de código. Esto supone un problema puesto que, al no pertenecer a ningún FB, las variables podrían generar errores a la hora de generar el código mediante el programa. Para solucionar esto, se crean nuevos FBs donde se recoge el código que anteriormente estaba en directamente en el FC, pasando las variables por parámetros.

En las siguientes imágenes se aprecia como es el código de la ventosa V en el código antiguo, es decir, programado directamente en el FC, y como es en el nuevo, creando un FB al que se llama desde el FC.

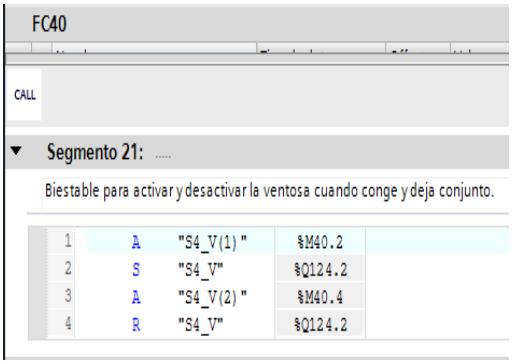

Ilustración 9.-Código ventosa FC

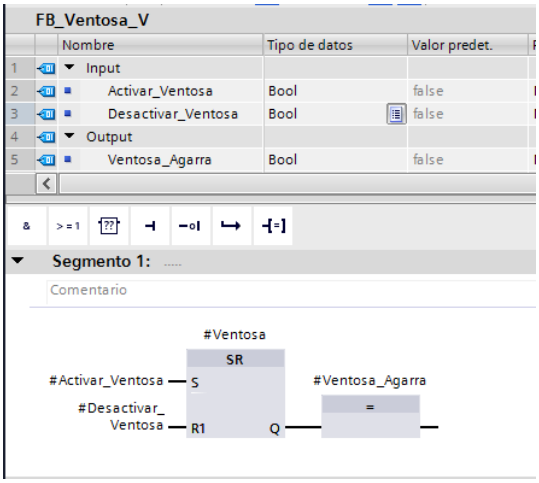

Ilustración 10.- Código ventosa FB

Con respecto a las marcas, como se ha explicado anteriormente, se decidió prescindir de ellas. Sin embargo, al ser el código original de la célula anterior a la idea de la modularización, este está lleno de ellas.

La solución adoptada es la de eliminar todas las marcas, pasando las variables a las que estaban asociadas a través de parámetros de DBs. Esto implica que los FBs, y por consiguiente sus DBs de intancia, tendrán únicamente variables de entrada y de salida.

En las dos imágenes que aparecen a continuación, se aprecia como varían los parámetros en las llamadas a los FBs. Como se observa, en el izquierdo no aparecen marcas, todas las variables son pasadas a traves de DBs. Además, los temporizadores han desaparecido tambien, ya que ahora estan estandarizados.

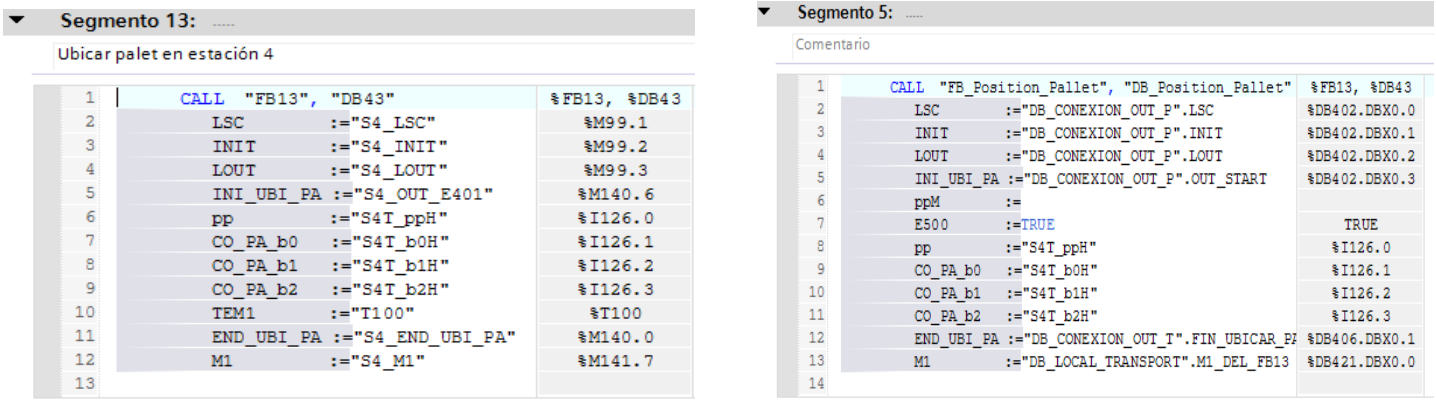

Ilustración 11.- Llamada FB código antiguo (izquierda y nuevo (derecha)

En cuanto a los DBs, se pueden realizar varias distinciones. Por un lado, pueden ser locales o globales. Siendo los primeros bloques de datos en los que los parámetros que se almacenan son utilizados únicamente en el módulo mecatrónico al que pertenece dicho BD. Por otra parte, los DBs globales almacenan datos que es necesario compartir con otros módulos mecatrónicos e incluso con otras estaciones para el correcto funcionamiento de la célula.

Además de esta diferenciación, los DBs pueden estar instanciados a FBs o no estarlos. Los DBs de instancia a FBs, almacenan los datos correspondientes al FB al que están instanciados; siendo posible que estén estanciados a más de un FB. Mientras que los demás DBs, son únicamente bloques de datos necesarios para el código.

Uno de los mayores problemas que se presenta al modularizar una estación, es cómo pasar las variables de unos módulos a otros. Cada uno de ellos es independiente, pero necesita de información de los demás para funcionar correctamente. Para ello se crean unos DBs de conexión. Estos DBs pueden ser de entrada o de salida, y en ellos se guardan las variables que cada módulo tiene que recibir o enviar a otros módulos respectivamente. Cabe destacar, que, al crearse el proyecto completo, únicamente se utilizan los de salida, puesto que de usar los dos, las variables se repetirían.

Por último, otro de los problemas que se presentan es el uso de los temporizadores propios de Siemens. Al ser un programa independiente, no los soporta, por lo que deberán de ser sustituidos por temporizadores multiinstancia estandarizados. Concretamente se utilizan temporizadores con retardo a la conexión. Afortunadamente, en el estándar existe un tipo de temporizador que realiza la misma labor, por lo que con sustituirlos es suficiente.

Ilustración 12.- Temporizador Siemens y Estándar

ADAPTACIÓN DEL HARDWARE

Al disponer de la célula de montaje FMS-200 en el laboratorio, se pudo comprobar la funcionalidad del código modularizado. No obstante, para ello fue necesario realizar varias modificaciones.

EL principal problema fueron los esclavos, mediante ellos el PLC manda las órdenes a la célula, por tanto, y como se ha dicho en la introducción de este apartado, cada módulo mecatrónico necesita su propio esclavo para poder funcionar independientemente y estar completamente modularizado.

Al añadirse un esclavo nuevo a cada estación, se han tenido que modificar las direcciones de algunas variables para que coincidan con las direcciones del nuevo código. Y, evidentemente, se ha modificado el hardware dentro del proyecto del TIA PORTAL.

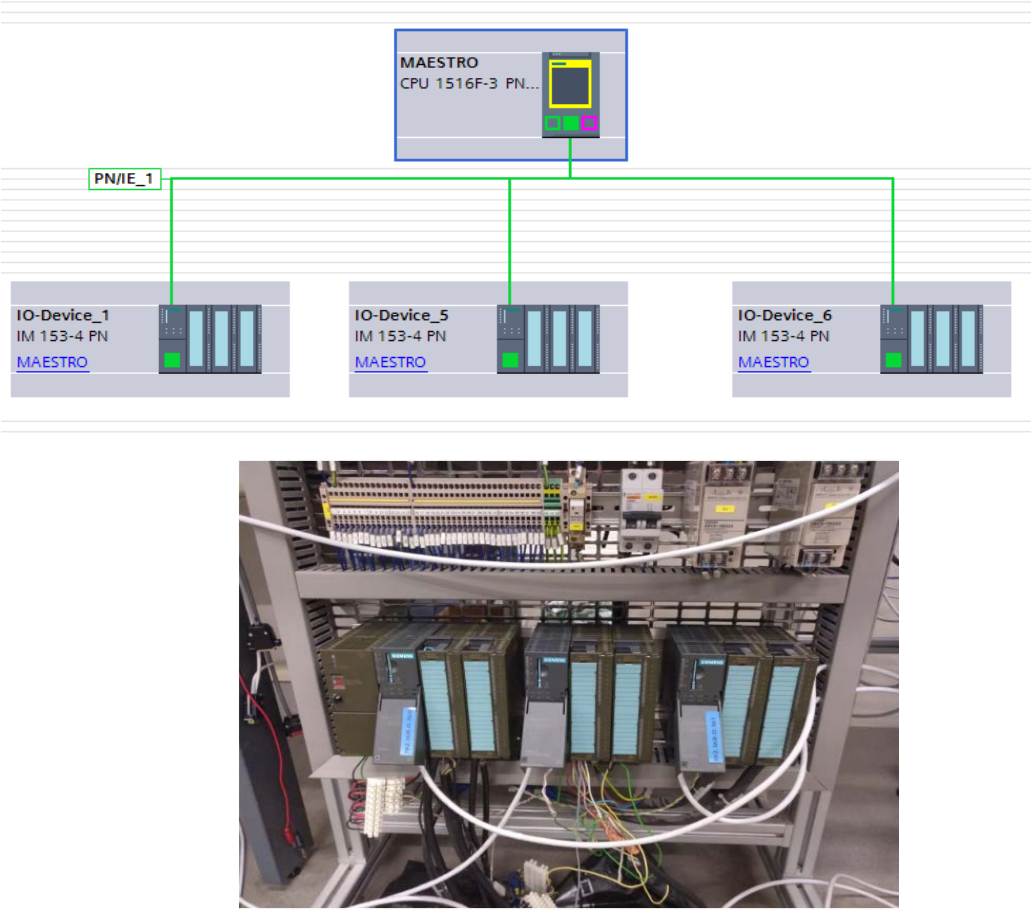

Ilustración 13.- Conexión esclavos

CÓDIGO

En este apartado se explica cómo queda el código tras aplicar los cambios nombrados en los apartados anteriores.

ESTACIÓN 3: ALIMENTACIÓN, SELECCIÓN Y COLOCACIÓN DE LA TAPA

El objetivo de esta estación es la coger el pallet con la base de la estación 2, colocar la tapa correcta en cada uno de los diferentes tipos de pallets que tiene la célula y transportar el pallet con la tapa a la estación 4.

Para realizar esta tarea, y como en el resto de estaciones, se dispone de una botonera para iniciar o detener la estación, de una cinta de transporte para trasladar el pallet con la pieza de una estación a otra y de un bastidor donde se identificará el pallet y se colocará la tapa correspondiente.

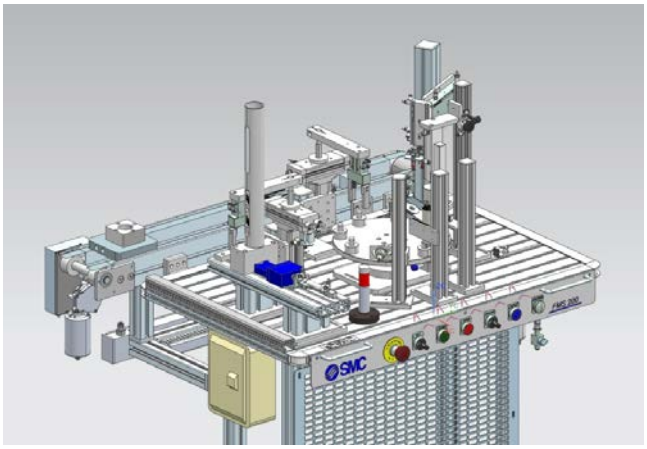

Ilustración 14.- Estación 3

MM PUSHBOTTOM

El módulo mecatrónico botonera contiene los pulsadores y botones para controlar cada una de las estaciones. Concretamente está formado por tres pulsadores que permiten iniciar, parar y resetear la estación; y un cuarto para detenerla en caso de emergencia. Este pulsador, simplemente corta la corriente de alimentación cuando es pulsado, no tiene ninguna entrada asignada y, por lo tanto, no pertenece directamente las formas de mando. Además, posee dos interruptores, uno para encenderla y apagarla y otro para elegir entre el modo de funcionamiento automático y el manual.

En cuanto al software, únicamente tiene un FB en el que están programadas las formas de mando de la estación. Las entradas de este son los interruptores y pulsadores mencionados anteriormente; y sus salidas, los parámetros LSC, INIT, LOUT cuyas funciones son la liberación de la cadena secuencial, la iniciación de las formas de mando y la liberación de las salidas respectivamente, el parámetro OUT_START que se utiliza para iniciar la secuencia de la estación, la señal ALARMA, que se activa en caso de que ocurra algún error durante la ejecución de la secuencia y, por último, las posiciones iniciales del módulo transporte y rack.

Así pues, una vez que se comprueba que la estación está en su posición inicial, y se ha elegido el modo de funcionamiento deseado (automático o manual), permite iniciar la secuencia de la estación.

Esta botonera es común en todas las estaciones, por lo que, gracias a la modularización, únicamente deberá de ser programada una vez, y mediante el programa de generación de código se añadirá al resto de estaciones.

Ilustración 15.- Botonera

[FB 11] FB_OPERATOR_PANEL

- VARIABLES

Nombre	Tipo	Definición
INPUT		
PI_BAS	Bool	Posición inicial del bastidor
PI_TRA	Bool	Posición inicial de la cinta
AUTO_MAN	Bool	Selector automático o manual
START	Bool	Pulsador de marcha
STOP	Bool	Pulsador de paro
EMERGENCY_STOP	Bool	Pulsador de emergencia
RESET	Bool	Pulsador de reset de alarmas
ONDA_CUA	Bool	Onda cuadrada
OUTPUT		
LSC	Bool	Liberación de cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de salidas
OUT_START	Bool	Señal pulsador de marcha
ALARM	Bool	Piloto luminoso

Tabla 1.- FB_Operator_Panel

- FUNCIÓN

En este FB se realizan todas las funciones de la botonera. En él están programadas las formas de mando de la estación, así como la condición de que no inicie la secuencia hasta que la estación esté en posición inicial. Para ello, recibe información de los FBs FB_InitP_Rack y FB_InitP_Transport.

MM_TRANSPORT

El módulo mecatrónico transporte es el encargado de transportar el pallet de una estación a otra. Está formado por una cinta transportadora impulsada mediante un motor de corriente continua, los sensores, que pueden ser de posición, o capacitivos para detectar el número del pallet que se transporta y el cilindro neumático que levanta y baja los sensores para bloquear o dejar pasar al pallet según sea necesario.

Al iniciarse la estación, la cinta se mueve trasladando el pallet hacia la mitad de esta, donde los sensores capacitivos leen el número del pallet. Posteriormente, cuando el bastidor acaba su secuencia colocando la tapa correcta, el cilindro neumático que eleva los pallets baja, y la cinta vuelve a activarse llevando al pallet con la tapa a la siguiente estación.

Respecto al software, su código está formado por cinco FBs. El primero de ellos comprueba que el módulo mecatrónico esté en su posición inicial cuando se inicia la secuencia. Dos de ellos son los encargados de enviar la señal de activación del motor que hace moverse a la cinta transportadora. Otro, a su vez se encarga de enviar estas señales a la salida del motor para que este se active. Por último, el FB restante se utiliza para mandar señales a otros módulos mecatrónicos. En este caso, la señal que comprueba que la estación siguiente esté vacía para poder enviar el pallet y el número del pallet.

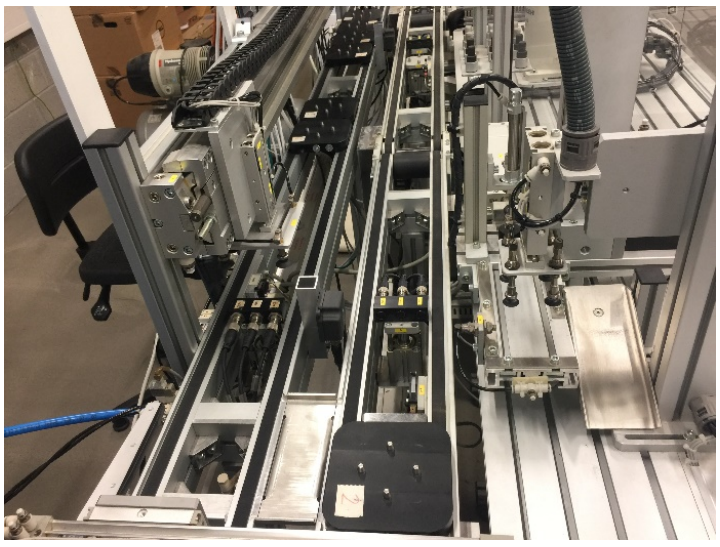

Ilustración 16.- Transporte

[FB9] FB_InitP_Transport

- VARIABLES

Nombre	Tipo	Definición
INPUT		
ppA	Bool	Sensor de presencia de palet
b0A	Bool	Bit 0 del código del palet
b1A	Bool	Bit 1 del código del palet
b2A	Bool	Bit 2 del código del palet
Sensores_Palet_IN	Byte	Byte de los sensores del palet
OUTPUT		
PI_Transporte	Bool	Posición inicial del módulo transporte
Sensores_Palet_OUT	Byte	Señal de los sensores del palet

Tabla 2.-FB_InitP_Transport

- FUNCIÓN

La función de este FB es definir la posición inicial del módulo mecatrónico transporte. Para que pueda iniciarse la secuencia, la cinta de transporte debe estar vacía, es decir, que tanto el sensor PP como los bits de los sensores para detectar el número del pallet tiene que estar a 0. Una vez el módulo esté en su posición inicial, se devuelve un 1 en la variable PI_Transporte. Para finalizar, se envía la información desde Sensores_Palet_In a Sensores_Palet_Out para posteriormente enviarla al módulo rack donde se utilizará para detectar el número del pallet.

[FB13] FB_Position_Pallet

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_UBI_PA	Bool	Señal de comienzo de la secuencia
ppM	Bool	Sensor de presencia de palet de la estación anterior
E500	Bool	Etapa inicial de la estación anterior
PP	Bool	Sensor de presencia del palet
CO_PA_b0	Bool	Bit 0 del código del palet
CO_PA_b1	Bool	Bit 1 del código del palet
CO_PA_b2	Bool	Bit 2 del código del palet
OUTPUT		
END_UBI_PA	Bool	Señal de fin de la secuencia
M1	Bool	Motor de la cinta

Tabla 3.- FB_Position_Pallet

- FUNCIÓN

Este FB es el encargado de trasladar el pallet desde su posición de origen al inicio de cada estación hasta la mitad de esta, donde esperará a que el bastidor haga su trabajo. La secuencia se inicia cuandose detecta un pallet en la posición inicial de la cinta PPM, entonces, se activa la señal INI_UBI_PA y se pone en marcha el motor para trasladar el pallet. Una vez que llega al centro de la estación y los sensores lo detectan, espera dos segundos más y finaliza la secuencia activándose la variable END_UBI_PA. Por último, se esperan tres segundos y se reinicia la secuencia.

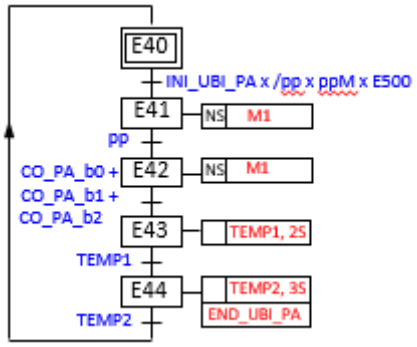

Ilustración 17.- Grafcet FB 13

[FB 17] FB_Take_Pallet

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_RE_PA	Bool	Señal de comienzo de la secuencia
pp	Bool	Sensor de presencia de palet
ES_SI_VA	Bool	Sensor de presencia de palet de la siguiente estación
OUTPUT		
END_RE_PA	Bool	Señal de fin de la secuencia
A1	Bool	Válvula cilindro tope del palet
M1	Bool	Motor de la cinta

Tabla 4.- FB_Take_Pallet

- FUNCIÓN

El objetivo de este FB es trasladar el pallet desde el centro de la estación hasta la estación siguiente. Cuando el bastidor acaba su secuencia y coloca la pieza correspondiente la base que descansa sobre el pallet, se activa la señal INI_RE_PA iniciándose la secuencia.

En primer lugar, se activa el cilindro A1, bajándose y permitiendo al pallet avanzar. Tras un segundo se activa el motor y una vez el sensor PP se desactiva, se vuelve a subir el cilindro. Posteriormente, dos segundos después, cuando el pallet haya llegado a la estación siguiente, se desactiva el motor. Por último, se esperan dos segundos y se finaliza la secuencia activándose la señal END_RE_PA.

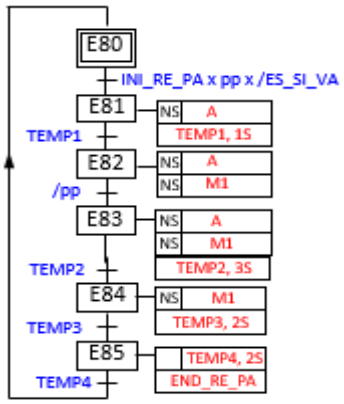

Ilustración 18.- Grafcet FB17

[FB18] FB_pp_Estacion_OUT

- Variables

Nombre	Tipo	Definición
INPUT		
pp_IN	Bool	Sensor de presencia de palet
OUTPUT		
pp_OUT	Bool	Señal de sensor de presencia de palet

- Función

En esta función se pasa la información de pp_IN a pp_OUT. Es un FB simple pero necesario , ya que gracias a él se puede obtener y mandar información a otras estaciones y saber si esta está libre u ocupada por otro pallet.

[FB19] FB_Mover_Cinta

- Variables

Nombre	Tipo	Definición
INPUT		
UBICAR_PALET	Bool	Variable para activar motor de la cinta para ubicar palet
RETIRAR_PALET	Bool	Variable para activar motor de la cinta para retirar palet
S4T_FREE	Bool	Sensor de presencia de palet en la siguiente estación
OUTPUT		
TRANSPORT	Bool	Motor de la cinta

- Función

En este FB se programa la secuencia para arrancar el motor de la cinta. Cuando bien la variable UBICAR_PALET o RETIRAR_PALET se activan y siempre y cuando la estación siguiente esté vacía, es decir, S4T_FREE esté a cero, se pondrá a uno la variable TRANSPORT y la cinta arrancará.

Esta función se añade para evitar que haya segmentos de código directamente en el FC principal.

MM_RACK

El objetivo de este módulo mecatrónico es la colocación de la tapa correcta en cada uno de los diferentes tipos de pallets que tiene la estación. Para ello, dispone de los siguientes elementos:

- Un almacén vertical donde se guardan las tapas con las que se alimenta a la estación.
- Un plato giratorio para trasladar las tapas a través de la estación.
- Tres sensores que detectan el tipo de tapa.
- Tres conjuntos de cilindros neumáticos y ventosas cuyas funciones son:
    - Trasportar las tapas desde el almacén al plato giratorio.
    - Retirar las tapas no deseadas.
    - Colocar la tapa deseada en la base.

Las tapas salen del almacén vertical de forma aleatoria, por ello son necesarios los tres sensores para detectar el tipo de pieza y, hacer que únicamente la correcta sea la que se coloque en la base.

El procedimiento que sigue es el siguiente. El módulo mecatrónico transporte pasa el número del pallet al módulo rack a través de un DB. Una vez se sabe el número, el módulo deduce el tipo de tapa que debe colocarle (negra, plateada o blanca). A continuación, comienza la extracción de tapas del almacén vertical. Estas se van colocando en el plato giratorio, que las va pasando a través de los 3 sensores que las identifican. Una vez salgan de los sensores, si la tapa es la correcta, se coloca en la base y se finalizará la secuencia; si no, se expulsa y el plato sigue girando, haciendo llegar a las demás tapas, hasta que llegue la deseada.

El software está compuesto por 9 FBs que están explicados detenidamente a continuación.

Ilustración 19.- Bastidor estación 3

[FB30] FB_InitP_Rack

- VARIABLES

Nombre	Tipo	Definición
INPUT		
Pt	Bool	Sensor de presencia de tapa fuera del alimentador
k1	Bool	Sensor de posición del cilindro K adelante
a0	Bool	Sensor de posición del cilindro A arriba
a1	Bool	Sensor de posición del cilindro Abajo
b0	Bool	Sensor de posición del cilindro B izquierda
b1	Bool	Sensor de posición del cilindro B derecha
f0	Bool	Sensor de posición del cilindro F atrás
f1	Bool	Sensor de posición del cilindro F adelante
g0	Bool	Sensor de posición del cilindro G arriba
h0	Bool	Sensor de posición del cilindro H arriba
h1	Bool	Sensor de posición del cilindro H abajo
i0	Bool	Sensor de posición del cilindro I izquierda
i1	Bool	Sensor de posición del cilindro I derecha
d0	Bool	Sensor de posición del cilindro D atrás
OUTPUT		
PIS3	Bool	Posición inicial del módulo bastidor

Tabla 5.- FB_InitP_Rack

- FUNCIÓN

En este FB se define la posición inicial del módulo RACK. Cuando los sensores a0, b0, f0, g0, h0, i0 y d0 estén a uno y los sensores pt, k1, a1, b1, f1, h1, y i1 estén a cero, se devuelve activa la variable PIS3, confirmando que el módulo mecatrónico está en su posición inicial.

[FB32] FB_Secuencia_Principal

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	BOOL	Liberación de la cadena secuencia
INIT	BOOL	Inicio de las formas de mando
LOUT	BOOL	Liberación de las salidas
INI_SE_S3	BOOL	Señal de comienzo de la secuencia
T_E304	BOOL	Transición etapa 304
T_E307	BOOL	Transición etapa 307
T_E308_OK	BOOL	Transición etapa 308 tras tapa buena
T_E308_DE	BOOL	Transición etapa 308 tras tapa mala
T_E309	BOOL	Transición etapa 309
T_E312	BOOL	Transición etapa 312
T_E313	BOOL	Transición etapa 313
EVACUAR	BOOL	Señal para evacuar la tapa
OUTPUT		
OUT_E302	BOOL	Salida etapa 302
OUT_E304	BOOL	Salida etapa 304
OUT_E305	BOOL	Salida etapa 305
OUT_E307	BOOL	Salida etapa 307
OUT_E308	BOOL	Salida etapa 308
OUT_E309	BOOL	Salida etapa 309
OUT_E310	BOOL	Salida etapa 3010
OUT_E312	BOOL	Salida etapa 312
OUT_E313	BOOL	Salida etapa 313

Tabla 6.- FB_Secuencia_Principal

- FUNCIÓN

En este FB se programa la secuencia principal del módulo mecatrónico bastidor, es decir, todos los pasos a seguir desde que se inicia con la activación de la variable INI_SE_S3 hasta que finaliza con la finalización de la etapa 313.

Este FB está programado de forma que, cuando se activa una etapa en la que se realiza una llamada a un FB, se active la salida de dicha etapa de forma que dicha llamada se pueda realizar. Una vez el FB llamado termina su secuencia, el FB principal avanza a la siguiente etapa.

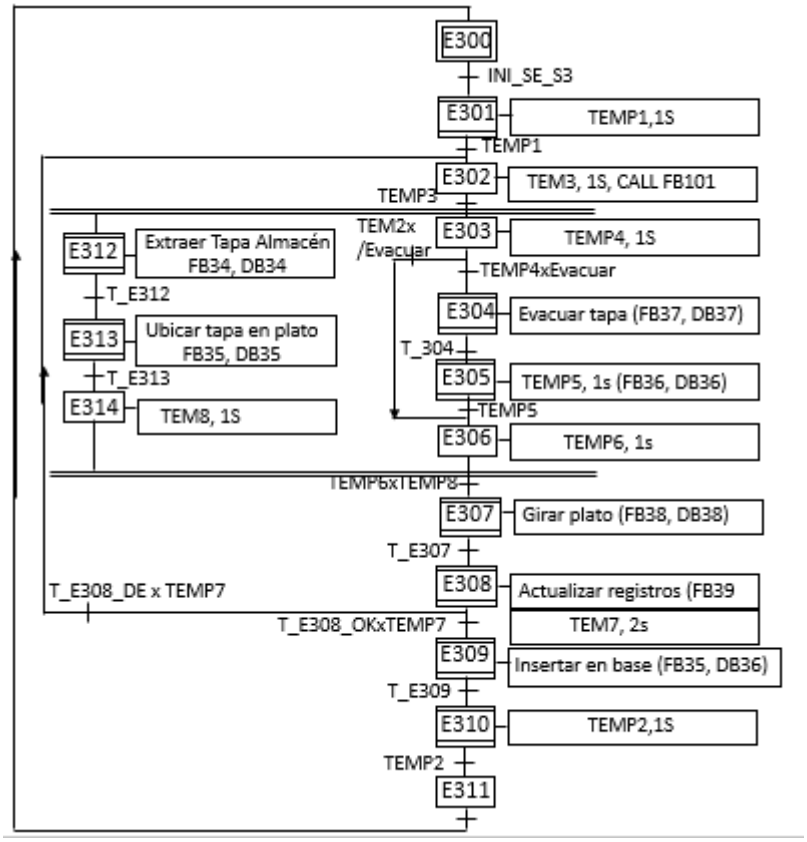

Ilustración 20.- Grafcet FB 32

[FB34] FB_Extraer_Tapa

- VARIBLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_EX_TA	Bool	Señal de comienzo de la secuencia
Pt	Bool	Sensor de presencia de tapa fuera del alimentador
K1	Bool	Sensor de posición del cilindro K adelante
OUTPUT		
END_EX_TA	Bool	Señal de fin de la secuencia
K	Bool	Válvula cilindro K

Tabla 7.- FB_Extraer_Tapa

- FUNCIÓN

La función de este FB es extraer la tapa del almacén vertical. La secuencia comienza cuando se activa la señal INI_EX_TA y los sensores no detectan ninguna tapa fuera del alimentador. Entonces, se activa el cilindro K durante 2 segundos, expulsando a la tapa que se encuentra en la posición inferior del almacén. A continuación, si los sensores Pt y k1 están activados, se espera 2 segundos y se activa la señal END_EX_TA durante dos segundos finalizando así la secuencia.

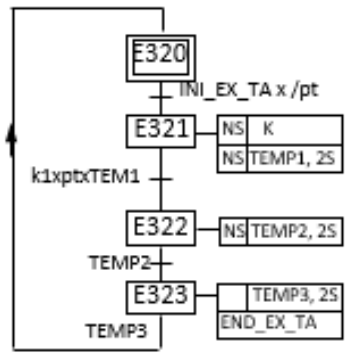

Ilustración 21.- Grafcet FB 34

[FB35] FB_Mover_Tapa

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_MO_TA	Bool	Señal de comienzo de la secuencia
verti0	Bool	Sensor de posición del cilindro vertical arriba
verti1	Bool	Sensor de posición del cilindro vertical abajo
giro0	Bool	Sensor de posición del cilindro de giro izquierda
giro1	Bool	Sensor de posición del cilindro de giro derecha
OUTPUT		
VERTI	Bool	Válvula del cilindro vertical
GIRO	Bool	Válvula del cilindro de giro
PINZA	Bool	Válvula de la pinza
END_MO_TA	Bool	Señal de fin de la secuencia

Tabla 8.- FB_Mover_Tapa

- FUNCIÓN

Este FB se utiliza dos veces durante la secuencia principal. Para trasladar la tapa desde el almacén vertical hasta el plato giratorio y para trasladarla desde la última posición del plato giratorio hasta la base sobre el pallet. La secuencia comienza cuando se activa la señal INI_MO_TA. Una vez esto ocurre, el primer paso es activar el cilindro vertical hasta que llega a la posición abajo. A continuación, se activa la pinza que agarra la tapa, se espera dos segundos y se desactiva el cilindro vertical. Una vez el cilindro vertical vuelve a su posición original, se activa el cilindro de giro haciendo que la pinza con la tapa se coloque sobre el plato giratorio. El cilindro vertical desciende y la pinza se desactiva soltando la tapa sobre el plato. Finalmente, se desactivan el cilindro vertical y el de giro, que vuelven a su posición inicial y se activa la variable END_MO_TA durante dos segundos dando por acabada la secuencia.

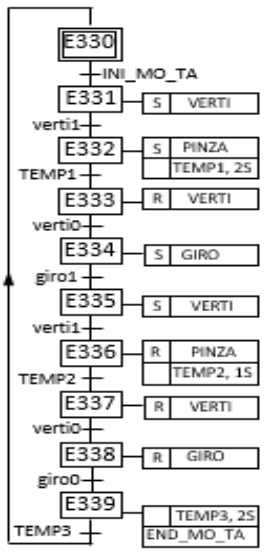

Ilustración 22.- Grafcet FB 35

[FB36] FB_Actualizar_Expulsar

- VARIBLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_ACT_REG	Bool	señal de comienzo de la secuencia
RE_EN_IN	Byte	Registro de entrada del detector inductivo
RE_EN_CA	Byte	Registro de entrada del detector capacitivo
RE_E N_FO	Byte	Registro de entrada del detector fotoeléctrico
OUTPUT		
RE_SA_IN	Byte	Registro de salida del detector inductivo
RE_SA_CA	Byte	Registro de salida del detector capacitivo
RE_SA_FO	Byte	Registro de salida del detector fotoeléctrico

Tabla 9.- FB_Actualizar_Expulsar

- FUNCIÓN

En este FB se lleva se pone a cero el sexto bit del registro tras explulsarse la tapa que ocupaba la posición del puesto 6 del plato giratorio. La secuencia comienza cuando se detecta un flanco positivo en la señal INI_ACT_REG. A continuación, se carga el registro de entrada del detector inductivo y el número BF en hexadecimal (10111111), se realiza un & con ambos. El resultado obtenido se transfiere al registro de salida del detector inductivo. Se realiza el mismo proceso con los registros de los detectores capacitivo y fotoeléctrico.

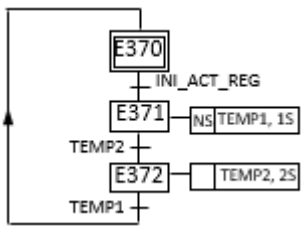

Ilustración 23.- Grafcet FB 36

[FB37] FB_Retirar_Tapa

- VARIBLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_RE_TA	Bool	Señal de comienzo de la secuencia
f0	Bool	Sensor de posición del cilindro F atrás
f1	Bool	Sensor de posición del cilindro F adelante
g0	Bool	Sensor de posición del cilindro G arriba
v1	Bool	Sensor de ventosa V vacío
OUTPUT		
F_mas	Bool	Válvula para avanzar cilindro F
F_menos	Bool	Válvula para retroceder cilindro F
G	Bool	Válvula cilindro G
V	Bool	Válvula ventosa V
END_RE_TA	Bool	Señal de fin de la secuencia

Tabla 10.- FB_Retirar_Tapa

- FUNCIÓN

La función de este FB es retirar las tapas que nos son las correspondientes al pallet que se encuentra en la estación. La secuencia comienza cuando se activa la señal INI_RE_TA. En primer lugar, se activa el cilindro horizontal que se extiende y se coloca sobre la tapa errónea. A continuación, se activa el cilindro vertical y dos segundos después se activa la ventosa que coge la pieza. Posteriormente, los cilindros vuelven a su posición inicial, colocando a la ventosa y la tapa sobre una rampa de salida. El cilindro vertical G desciende y la ventosa suelta la pieza sobre dicha rampa. Por último, los cilindros vuelven a su posición inicial y se finaliza la secuencia activando la señal END_RE_TA durante 2 segundos.

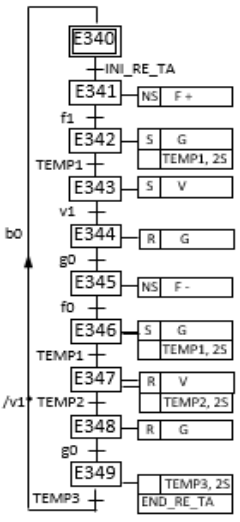

Ilustración 24.- Grafcet FB 37

[FB38] FB_Girar_Plato

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_GI_PL	Bool	Señal de comienzo de la secuencia
d0	Bool	Sensor de posición del cilindro D atrás
OUTPUT		
END_GI_PL	Bool	Señal de fin de la secuencia
D	Bool	Válvula cilindro D
E	Bool	Válvula cilindro E

Tabla 11.- FB_Girar_Plato

- FUNCIÓN

Mediante este FB se hace girar al plato, transportando las tapas a través de los diferentes puestos del bastidor de la estación 3. La secuencia comienza cuando se activa la señal INI_GI_PL. Primero, se activa el cilindro E, que conectando al cilindro D al plato giratorio. Se espera 500ms y se activa el cilindro D que al expandirse hace que el plato gire. Cuando el plato llega a su nueva posición se desactivan ambos cilindros que vuelven a su posición inicial. Se finaliza la secuencia activando la señal END_GI_PL durante 2 segundos.

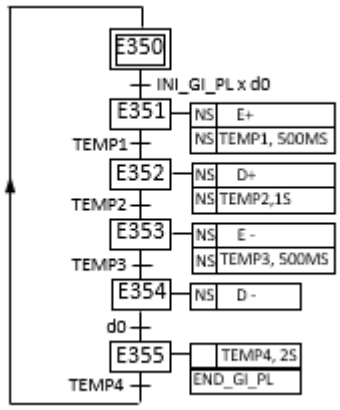

Ilustración 25.- Grafcet FB 38

[FB39] FB_Actualizar_Registros

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	Bool	Liberación de la cadena secuencial
INIT	Bool	Inicio de las formas de mando
LOUT	Bool	Liberación de las salidas
INI_AC_RE	Bool	Señal de comienzo de la secuencia
RE_EN_IN	Byte	Entrada del registro del sensor inductivo
RE_EN_CA	Byte	Entrada del registro del sensor capacitivo
RE_EN_FO	Byte	Entrada del registro del sensor fotoeléctrico
VALOR_SEN	Byte	Sensores para detectar color de tapa
OUTPUT		
END_AC_RE_DE	Bool	Señal de fin de la secuencia para tapa mala
RE_SA_IN	Byte	Salida del registro del sensor inductivo
RE_SA_CA	Byte	Salida del registro del sensor capacitivo
RE_SA_FO	Byte	Salida del registro del sensor fotoeléctrico
END_AC_RE_OK	Bool	Señal de fin de la secuencia para tapa buena

Tabla 12.- FB_Actualizar_Registros

- FUNCIÓN

A medida que el plato gira, las tapas van pasando por los diferentes sensores, y es necesario llevar un registro de ellas. Esto es precisamente lo que se realiza en este FB. La secuencia comienza cuando se activa la señal INI_AC_RE. Para actualizar los registros, primero hay que desplazar los antiguos una posición a la izquierda. A continuación, se leen los 3 sensores y se coloca su información en sus respectivas posiciones de los registros. Por último, se lee la información de la última posición del registro capacitivo actualizado para saber si hay una tapa. Si es un 1, la señal END_AC_RE_OK se activa, si es un 0 se activa la señal END_AC_RE_DE.

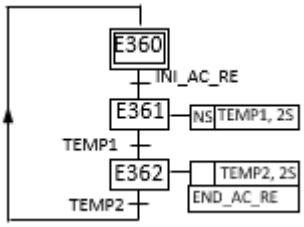

Ilustración 26.- Grafcet FB 39

[FB101] FB_Comparar_Tapa

- VARIABLES

Nombre	Tipo	Definición
INPUT		
INI_COM_TA	Bool	Señal de comienzo de la secuencia
RE_EN_IN	Byte	Entrada del registro del sensor inductivo
RE_EN_CA	Byte	Entrada del registro del sensor capacitivo
RE_EN_FO	Byte	Entrada del registro del sensor fotoeléctrico
CODIGO_PALET	Byte	Código del palet
OUTPUT		
EVACUAR	Bool	Señal para evacuar la tapa

Tabla 13.- FB_Comparar_Tapa

- FUNCIÓN

Mediante este FB se comprueba si la tapa que se encuentra en el puesto 6 del plato giratorio se corresponde con la que el pallet necesita. La secuencia comienza cuando se activa la señal INI_COM_TA. Primero se lee el código del pallet para saber el color de la tapa que hay que poner:

- Tapa Gris: código 1 ó 4.
- Tapa Blanca: código 2 ó 5.
- Tapa Negra: código 3 ó 6.

A continuación, se lee la información de los registros para saber el color de la tapa que se tiene en la estación en el puesto 6:

- Sensor inductivo: Tapa Gris.
- Sensor inductivo desactivado y fotoeléctrico activado: Tapa Blanca.
- Sensor fotoeléctrico desactivado: Tapa Negra.
- Sensor capacitivo desactivado: No hay tapa.

Si el color de la tapa del puesto 6 no coincide con el color de la tapa del código del palet se activa la señal evacuar. Y si lo hace no se hace nada, de forma que la tapa avance hasta la última posición del plato y sea colocada en la base.

ESTACIÓN 4: ALMACÉN DE CONJUNTOS TERMINADOS

La estación 4, al igual las demás está formada por un módulo mecatrónico botonera, uno de transporte y otro de bastidor. Su función es almacenar los conjuntos terminados salientes de la estación 3 en su lugar correspondiente del almacén y retirar el pallet vacío a la estación 5.

Los módulos botonera y transporte son los mismos que en las estaciones anteriores por lo que nos centraremos exclusivamente en el módulo bastidor.

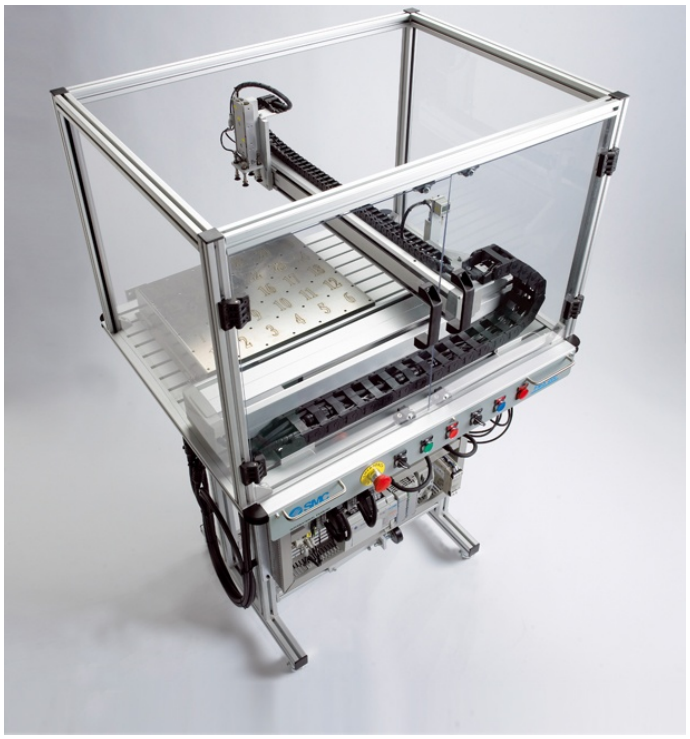

Ilustración 27.- Estación 4

MM RACK 4

La función de este módulo es coger el conjunto montado del pallet y colocarlo en su lugar correspondiente del almacén. Para ello, recibe del módulo transporte el número del conjunto que está colocando, y gracias a un FB en el que se realiza un registro de los conjuntos colocados, deduce la fila y columna del almacén en el que debe depositar la pieza. Nótese, que no tiene que decidir únicamente la fila correspondiente al número del conjunto, si no, que debe llevar la cuenta de cuantos conjuntos como ese ha colocado ya para elegir la columna adecuada.

Este módulo está compuesto por dos servomotores que son los encargados de desplazar la pieza, un conjunto de cilindro neumático y ventosas para coger la pieza y el propio almacén.

La gran diferencia con el resto de módulos es que la conexión con los servos se hace mediante profibus, mientras que la del resto de la estación, mediante profinet. Por ello, es necesario colocar un Gateway que realice la traducción. Así pues, el PLC envía las órdenes al Gateway, y este las traduce y las envía a los servos para que realicen los movimientos deseados.

Respecto a su software, está compuesto por 9 FBs cuyas funciones se detallan a continuación.

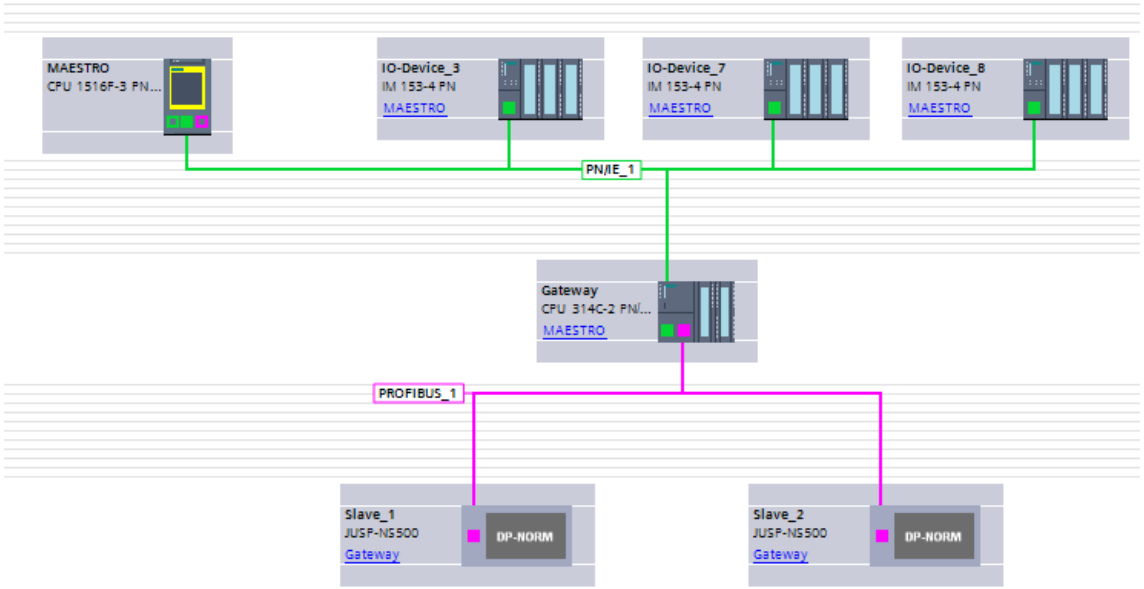

Ilustración 28.- Conexión Gateway

[FB 40] FB_InitP_Rack

Nombre	Tipo	Definición
INPUT		
a0	Bool	Sensor de posición del cilindro A contraído
a1	Bool	Sensor de posición del cilindro A extendido
v1	Bool	Sensor de la ventosa V
OUTPUT		
PIS4	Bool	Señal para indicar que la estación está en su posición inicial

Tabla 14.- FB_InitP_Rack

- FUNCIÓN

En este FB se define la posición inicial del módulo Rack de la estación 4. Es la primera secuencia de la estación y sirve para asegurar que la estación estará en la posición correcta cuando comience la secuencia. Para que pueda iniciar la secuencia, el cilindro A y la ventosa V deben estar en su posición inicial, es el cilindro recogido y la ventosa sin activar. Cuando esto ocurre, la señal PIS4 pasa a 1 habilitando así al siguiente segmento del programa.

[FB42] FB_Secuencia_Principal

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	BOOL	Liberación de la cadena secuencia
INIT	BOOL	Inicio de las formas de mando
LOUT	BOOL	Liberación de las salidas
INI_SE_S4	BOOL	Señal de comienzo de la secuencia
T_E403	BOOL	Transición etapa 403
T_E406	BOOL	Transición etapa 406
T_E407	BOOL	Transición etapa 407
T_E408	BOOL	Transición etapa 408
T_E409	BOOL	Transición etapa 409
T_E410	BOOL	Transición etapa 410
OUTPUT		
OUT_E401	BOOL	Salida etapa 401
OUT_E403	BOOL	Salida etapa 403
OUT_E406	BOOL	Salida etapa 406
OUT_E407	BOOL	Salida etapa 407
OUT_E408	BOOL	Salida etapa 408
OUT_E409	BOOL	Salida etapa 409
OUT_E410	BOOL	Salida etapa 410
OUT_E412	BOOL	Salida etapa 412
COM_SERV_X	DWord	Salida del servo X ante la primera parte de la trama
PAR_SERV_X	DWord	Salida del servo X ante la segunda parte de la trama
COM_SERV_Y	DWord	Salida del servo Y ante la primera parte de la trama
PAR_SERV_Y	DWord	Salida del servo Y ante la segunda parte de la trama

Tabla 15.- FB_Secuencia_Principal

- FUNCIÓN

En este FB se desarrollan cada una de las secuencias de la estación. Se inicia cuando se activa la secuencia INI_SE_S4, es decir, una vez se comprueba que las posiciones son correctas; y acaba cuando los servos vuelven a su posición de referencia.

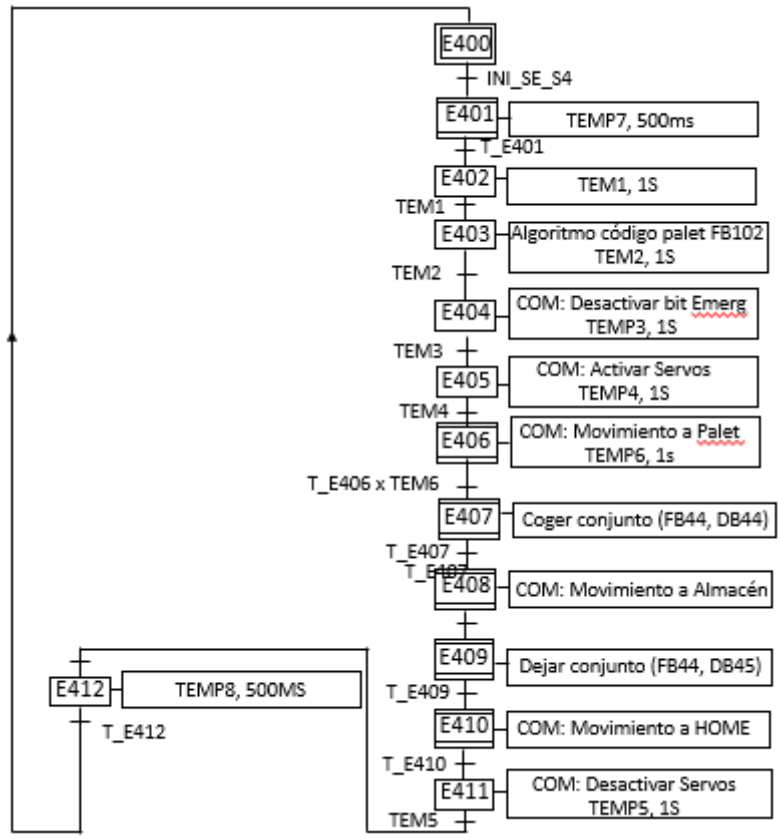

Ilustración 29.- Grafcet FB 42

[FB 61] FB_CODIGO_ALMACEN

- VARIABLES

Nombre	Tipo	Definición
INPUT		
INDEX_COL_1_IN	Byte	Entrada del registro de la columna 1
INDEX_COL_2_IN	Byte	Entrada del registro de la columna 2
INDEX_COL_3_IN	Byte	Entrada del registro de la columna 3
INDEX_COL_4_IN	Byte	Entrada del registro de la columna 4
INDEX_COL_5_IN	Byte	Entrada del registro de la columna 5
INDEX_COL_6_IN	Byte	Entrada del registro de la columna 6
T_E403	BOOL	Transición etapa 403
Sensores_Palet_IN	Byte	Código del pallet
C1	DWord	Registro Columna 1
C2	DWord	Registro Columna 2
C3	DWord	Registro Columna 3
C4	DWord	Registro Columna 4
C5	DWord	Registro Columna 5
C6	DWord	Registro Columna 6
F1	DWord	Registro Fila 1
F2	DWord	Registro Fila 2
F3	DWord	Registro Fila 3
F4	DWord	Registro Fila 4
F5	DWord	Registro Fila 5
OUTPUT		
SALIDA_X	DWord	Salida de la columna a la que hay que llevar la pieza
SALIDA_Y	DWord	Salida de la columna a la que hay que llevar la pieza
INDEX_COL_1_OUT	Byte	Salida del registro de la columna 1
INDEX_COL_2_OUT	Byte	Salida del registro de la columna 2
INDEX_COL_3_OUT	Byte	Salida del registro de la columna 3
INDEX_COL_4_OUT	Byte	Salida del registro de la columna 4
INDEX_COL_5_OUT	Byte	Salida del registro de la columna 5
INDEX_COL_6_OUT	Byte	Salida del registro de la columna 6

Tabla 16.- FB_CODIGO_ALMACEN

- FUNCIÓN

Este FB tiene una doble función, por un lado, lleva el control de las piezas guardadas en el almacén, y, por otro facilita la posición en la que se debe depositar la pieza una vez finalizada. El FB tiene como variable de entrada el código del pallet, gracias se elige la columna a la que debe de ir la pieza. Posteriormente se comprueba cuantas piezas hay en dicha columna y se calcula la fila en la que debe depositarse la pieza. Una vez decidida la posición, se manda a los servos a través de las variables de salida (SALIDA_X, SALIDA_Y).

[FB 46] FB_MOVIMIENTO_A_PALET

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	BOOL	Liberación de la cadena secuencia
INIT	BOOL	Inicio de las formas de mando
LOUT	BOOL	Liberación de las salidas
INI_COM_MOV	BOOL	Señal de comienzo de la secuencia
IMPOS_SERV_X	BOOL	Movimiento del servo X
IMPOS_SERV_Y	BOOL	Movimiento del servo Y
COORD_X	DWord	Coordenada X de la posición a la que mover el servo
COORD_Y	DWord	Coordenada Y de la posición a la que mover el servo
OUTPUT		
COM_SERV_X	DWord	Salida del servo X ante la primera parte de la trama
PAR_SERV_X	DWord	Salida del servo X ante la segunda parte de la trama
COM_SERV_Y	DWord	Salida del servo Y ante la primera parte de la trama
PAR_SERV_Y	DWord	Salida del servo Y ante la segunda parte de la trama
END_COM_MOV	BOOL	Fin de la secuencia

Tabla 17.- FB_MOVIMIENTO_A_PALET

- FUNCIÓN

Este FB se utiliza cada vez que se quiere mover los servos de una posición a otra. Concretamente 3 veces en esta estación:

- DB 46: Movimiento al palet para recoger la pieza.
- DB 56: Movimiento al almacén para dejar la pieza.
- DB 66: Movimiento a la posición de referencia (HOME) una vez dejada la pieza.

El funcionamiento de las 3 es el mismo, una vez que se cumplen las condiciones para iniciar la secuencia, se le pasa desde el DB correspondiente la posición a la que se quiere que vaya. Esta información se pasa a los servos a través del Gateway, y finalmente, los servos se desplazan a la posición deseada.

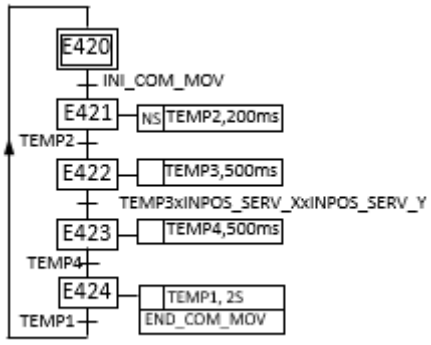

Ilustración 30.- Grafcet FB 46

[FB 44] FB_Coger_Pieza

- VARIABLES

Nombre	Tipo	Definición
INPUT		
LSC	BOOL	Liberación de la cadena secuencia
INIT	BOOL	Inicio de las formas de mando
LOUT	BOOL	Liberación de las salidas
A0	BOOL	Sensor de posición del cilindro A contraído
A1	BOOL	Sensor de posición del cilindro A extendido
V1	BOOL	Sensor de la ventosa Vbn
INI_CO_CO	BOOL	Señal de comienzo de la secuencia
OUTPUT		
A	BOOL	Salida para activar el sensor A
V	BOOL	Salida para activar la válvula V
END_CO_CO	BOOL	Señal de fin de secuencia

Tabla 18.- FB_Coger_Pieza

- FUNCIÓN

Mediante este FB se lleva a cabo tanto la secuencia de coger la pieza (DB 44) como la de dejarla (DB 45). Una vez iniciadas las señales LSC, INIT, LOUT y INI_CO_CO, el cilindro A baja (pasa de a0 a a1) y, en el caso de que se desee coger la pieza, la ventosa V se activará agarrándola mientras que en el caso que se desee dejarla, la ventosa que estaría activada previamente sujetando la pieza, se desactivaría para soltarla.

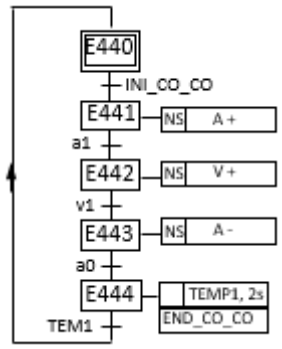

Ilustración 31.- Grafcet FB 44

[FB 48] FB_Cilindro_A

- VARIABLES

Nombre	Tipo	Definición
INPUT		
Coger_Pieza	BOOL	Señal para activar el cilindro A
Dejar_Pieza	BOOL	Señal para desactivar el cilindro A
OUTPUT		
Bajar_Cilindro	BOOL	Salida para activar o desactivar el cilindro A

Tabla 19.- FB_Cilindro_A

- FUNCIÓN

Este FB se utiliza para sustituir a una puerta lógica AND y evitar así el código en lista de instrucciones del FC 40. Tanto cuando se activa la señal Coger_Pieza como la de Dejar_Pieza, el cilindro A baja, pasando de a0 a a1.

[FB 49] FB_Ventosa_V

- VARIABLES

Nombre	Tipo	Definición
INPUT		
Activar_Ventosa	BOOL	Señal para activar la ventosa V
Desactivar_Ventosa	BOOL	Señal para desactivar la ventosa V
OUTPUT		
Ventosa_Agarra	BOOL	Salida para activar o desactivar la ventosa V

Tabla 20.- FB_Ventosa_V

- FUNCIÓN

Este FB sustituye a una puerta lógica Flip/Flop. Se utiliza para activar y desactivar la ventosa sin necesidad de utilizar código en lista de instrucciones en el FC 40. Cuando se activa la señal Activar_Ventosa, se pone a 1 la señal Ventosa_Agarra, activándose así la ventosa; y permanece activa hasta que la señal Desactivar_Ventosa pasa a 1.

[FB 60] FB_CODIGO_SERVO

- VARIABLES

Nombre	Tipo	Definición
INPUT		
Com_Serv_X1	DWord	Señal de la primera parte de la trama del servo X proveniente de la secuencia principal
Com_Serv_X2	DWord	Señal de la primera parte de la trama del servo X proveniente de la secuencia de movimiento a pallet
Com_Serv_X3	DWord	Señal de la primera parte de la trama del servo X proveniente de la secuencia de depositar la pieza en almacén
Com_Serv_X4	DWord	Señal de la primera parte de la trama del servo X proveniente de la secuencia de movimiento a HOME
Com_Serv_Y1	DWord	Señal de la primera parte de la trama del servo Y proveniente de la secuencia principal
Com_Serv_Y2	DWord	Señal de la primera parte de la trama del servo Y proveniente de la secuencia de movimiento a pallet
Com_Serv_Y3	DWord	Señal de la primera parte de la trama del servo Y proveniente de la secuencia de depositar la pieza en almacén
Com_Serv_Y4	DWord	Señal de la primera parte de la trama del servo Y proveniente de la secuencia de movimiento a HOME
Par_Serv_X1	DWord	Señal de la segunda parte de la trama del servo X proveniente de la secuencia principal
Par_Serv_X2	DWord	Señal de la segunda parte de la trama del servo X proveniente de la secuencia de movimiento a pallet
Par_Serv_X3	DWord	Señal de la segunda parte de la trama del servo X proveniente de la secuencia de depositar la pieza en almacén
Par_Serv_X4	DWord	Señal de la segunda parte de la trama del servo X proveniente de la secuencia de movimiento a HOME
Par_Serv_Y1	DWord	Señal de la segunda parte de la trama del servo Y proveniente de la secuencia principal
Par_Serv_Y2	DWord	Señal de la segunda parte de la trama del servo Y proveniente de la secuencia de movimiento a pallet
Par_Serv_Y3	DWord	Señal de la segunda parte de la trama del servo Y proveniente de la secuencia de depositar la pieza en almacén

Par_Serv_Y4	DWord	Señal de la segunda parte de la trama del servo Y proveniente de la secuencia de movimiento a HOME
OUTPUT		
Com_Serv_X	DWord	Salida del servo X ante la primera parte de la trama
Com_Serv_Y	DWord	Salida del servo Y ante la primera parte de la trama
Par_Serv_X	DWord	Salida del servo X ante la segunda parte de la trama
Par_Serv_Y	DWord	Salida del servo Y ante la segunda parte de la trama

Tabla 21.- FB_CODIGO_SERVO

- FUNCIÓN

Este FB se utiliza para mandar las ordenes de movimiento a los servos. Estas órdenes vienen desde distintos FBs, tanto los utilizados para mover los servos al pallet para coger la pieza, los utilizados para llevar la pieza a su posición correspondiente del almacén y dejarla, como para llevar a los servos a su posición de referencia.

El FB en sí es una carga y transferencia de datos. Las señales que llegan desde los FBs anteriormente nombrados, se transfieren a los servos para que realicen los movimientos requeridos.

[FB 80] FB_Referencia_Servos

- VARIABLES

NOMBRE	TIPO	DEFINICIÓN
INPUT		
INPOS_SERV_X	DWord	Movimiento del servo X
INPOS_SERV_Y	DWord	Movimiento del servo Y
COORD_X	DWord	Coordenada X de la posición a la que mover el servo
COORD_Y	DWord	Coordenada Y de la posición a la que mover el servo
OUTPUT		
COM_SERV_X	DWord	Salida del servo X ante la primera parte de la trama
PAR_SERV_X	DWord	Salida del servo X ante la segunda parte de la trama
COM_SERV_Y	DWord	Salida del servo Y ante la primera parte de la trama
PAR_SERV_Y	DWord	Salida del servo Y ante la segunda parte de la trama
IN/OUT		
Referencia	BOOL	Señal utilizada para activar la secuencia para mandar a referencia los servos una única vez al arrancar la estación

Tabla 22.- FB_80

- FUNCIÓN

Mediante este FB se manda a la posición de referencia a los servos al arrancar la célula. Esto es necesario porque cuando la estación se apaga, los servos pierden dicha referencia. Esta secuencia es llamada desde el FC1, y se hace antes de iniciar la FC40, es decir, antes de iniciar las secuencias de la estación 4. Además, gracias a la señal "Referencia", que se resetea una vez finaliza este FB, solo se activa una vez.

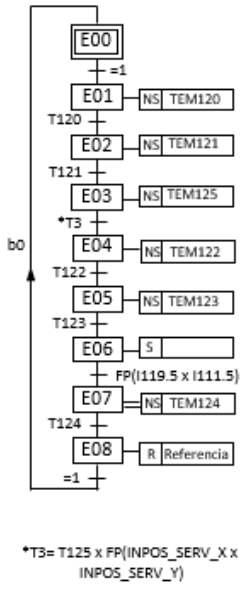

Ilustración 32.- Grafcet FB 80

RIESGOS

En este apartado se desarrollarán y evaluarán los riesgos más destacables durante el desarrollo del proyecto.

A. VERSIONES DIFERENTES DE LAS HERRAMIENTAS

La tecnología avanza con rapidez, y constantemente aparecen actualizaciones y nuevas versiones de los programas y aplicaciones. Esto puede suponer un problema ya que pueden ocurrir incompatibilidades entre las versiones del TIA Portal. Las versiones más modernas pueden abrir y trabajar con proyectos de versiones antiguas, sin embargo, esto no puede realizarse a la inversa. Además, cuando se abre un proyecto con una versión más avanzada, este se actualiza al instante a esa nueva versión, por lo que no podrá ser abierto por versiones antiguas.

-PPROBABILIDAD: Media. Debido a las continuas actualizaciones de las herramientas, es probable que se tengan instaladas diferentes versiones en los equipos de trabajo.

- IMPACTO: Alto. El hecho de que un proyecto sea incompatible con una versión, puede significar el tener que instalar la versión correspondiente o la creación del proyecto en la versión necesaria. Ambas provocando una gran pérdida de tiempo.

-PLAN DE CONTINGENCIA: La mejor solución es asegurarse antes de empezar el proyecto de que todas las versiones son compatibles.

B. FALLOS EN LOS COMPONENTES DE LA CÉLULA

El proyecto incluye la comprobación del código en la célula FMS-200, dicha célula está compuesta por una gran cantidad de sensores, interruptores, cilindros neumáticos, válvulas y demás componentes que pueden sufrir fallos y retrasar el proyecto.

-PROBABILIDAD: Media: Debido al gran número de componentes de la célula, es probable que alguno falle durante el desarrollo del proyecto.

- IMPACTO: Medio. Si bien dependerá del componente que se haya roto, en general se requiere de unas horas de trabajo para sustituirlo por uno nuevo.

-PLAN DE CONTINGENCIA: Es conveniente disponer de recambios adecuados para ellos para solventar este problema lo más rápido posible.

C. ERRORES AL PROGRAMAR LAS VARIABLES

En el código, se trabajan con un gran número de variables, y es posible que, llamarlas desde otra parte del código o al modificarlas, se produzcan errores que provoquen que el proyecto no funcione.

-PROBABILIDAD: Alta. Debido al gran número de variables, es sencillo cometer errores a la hora de programar.

-IMPACTO: Bajo. El impacto dependerá del error cometido, pero, generalmente detectar y solucionar estos errores es sencillo.

-PLAN DE CONTINGENCIA: La mejor forma de evitar esto es documentar las variables que se van utilizando y de las acciones que se van realizando con ellas.

D. ERRORES EN LAS DIRECCIONES Y NOMBRES AL REUTILIZAR EL CÓDIGO

Una de las ventajas de la modularización es que permite reutilizar partes del código. Sin embargo, hay que tener en cuenta, que estás variables duplicadas no pueden llevar el mismo nombre que las originales. Y que, además, su dirección será distinta.

-PROBABILIDAD: Baja. Debido a que el trabajo consiste en la modularización de la célula, se tiene especial cuidado con este tipo de variables, lo que reduce la posibilidad de fallos.

-IMPACTO: Medio. Los errores en estas variables pueden ser complicados de detectar, ya que al compilar el programa no los detecta puesto que hasta que no se junte el proyecto entero a través del programa de generación, estás variables no aparecen como duplicadas.

-PLAN DE CONTINGENCIA: La mejor solución es documentar todas las variables con sus nombres y direcciones, para no cometer errores a la hora de programarlas.

		IMPACTO		
		Bajo 0,2	Medio 0,5	Alto 0,8
PROBABILIDAD	Baja 0,2	Bajo 0,04	D Bajo 0,1	Medio 0,16
	Media 0,5	Bajo 0,1	B Medio 0,25	A Alto 0,4
	Alta 0,8	C Medio 0,16	Alto 0,4	Alto 0,64

Tabla 23: Matriz probabilidad-impacto:

ALTERNATIVAS

Existen diversas metodologías a la hora de programar células de montaje. La más utilizada hasta ahora es la programación secuencial, donde toda la célula se programa en un único proyecto y cada estación no puede funcionar a menos que anteriores lo hayan hecho. Además, un único centro de control supervisa y procesa todas las señales de la instalación. Estas máquinas son rígidas e inflexibles, ideales para procesos de producción en los que se fabrican grandes cantidades de piezas idénticas. Sin embargo, debido a que la sociedad demanda productos cada vez más personalizados y las empresas necesitan piezas cada vez más especializadas, este tipo de programación se está quedando atrás. Ya que modificar y adaptar las células programadas y diseñadas de esta forma requieren grandes inversiones de tiempo y dinero.

Por otro lado, es inviable utilizar este tipo de programación en un programa de generación automática. Al crearse directamente todo el proyecto, no se obtendría ninguna ventaja.

Otra alternativa, es la utilización de otras herramientas para la programación del código en lugar del TIA Portal. Ejemplos de ellas son CX-One o CODESYS. No obstante, podría suponer cambiar los PLCs y demás componentes puesto que podrían surgir incompatibilidades entre ellos. Por lo tanto, y aprovechando las herramientas del departamento, se optó por el TIA Portal.

PRESUPUESTO

En este apartado, se muestra el presupuesto requerido para la realización del proyecto y se detallan cada uno de los costes que lo provocan.

HORAS INTERNAS

Las horas internas son calculadas en base al número de horas que cada individuo ha dedicado a la realización del proyecto. Las horas del director se corresponden con lo que cobra un ingeniero senior, mientras que las del alumno, con lo que cobra un ingeniero junior. Como se observa en la tabla, el director ha dedicado 30 horas a 40 euros la hora, mientras que el alumno, 200 horas a 20 euros cada una. Haciendo un total de 5200 euros.

Concepto	Uso (h)	Precio unitario (€/h)	Subtotal (€)	Total (€)
Horas internas				**5200,00**
Director	30,00	40,00	1200,00	
Alumno	200,00	20,00	4000,00	

Tabla 24.- Horas Internas

AMORTIZACIONES

Para las amortizaciones, se tienen en cuenta las herramientas que podrán ser utilizadas en proyectos futuros, es decir, el ordenador donde se ha programado el código, la licencia del TIA Portal, y el PLC. Haciendo un total de 1399.80 euros.

Concepto	Uso (h)	Precio unitario (€/h)	Subtotal (€)	Total (€)
Amortizaciones				**1399,80**
TIA Portal	150,00	4,00	600,00	
PLC	30,00	20,00	600,00	
Ordenador	180,00	1,11	199,80	

Tabla 25.- Amortizaciones

GASTOS

En este apartado se tienen en cuenta gastos menores de material, mayormente fotocopias y material de escritura. Su coste se ha estimado en 30 euros.

COSTES INDIRECTOS

En este apartado entran los gastos ocasionados por el consumo de electricidad y uso de espacio durante los meses de duración del proyecto. El gasto se estima en un 8% del total.

IMPREVISTOS

Se estima un coste del 10% del total para los posibles imprevistos que pueden aparecer durante el proyecto, mayormente rupturas en componentes de la célula.

COSTES FINANCIEROS

Se calcula que hay que destinar un 5% de los costes totales para costear los costes financieros que se originen durante el proyecto.

COSTES TOTALES

Sumando todos los costes mencionados anteriormente, se obtiene un coste total de 8270.01 euros.

Concepto	Uso (h)	Precio unitario (€/h)	Subtotal (€)	Total (€)
Horas internas				**5200,00**
Director	30,00	40,00	1200,00	
Alumno	200,00	20,00	4000,00	
Amortizaciones				**1399,80**
TIA Portal	150,00	4,00	600,00	
PLC	30,00	20,00	600,00	
Ordenador	180,00	1,11	199,80	
Gastos				**30,00**
Material oficina			30,00	
Subontrataciones				**0,00**
COSTES DIRECTOS				**6629,80**
indirectos	8%			
SUBTOTAL				**7160,18**
Imprevistos	10%			
SUBTOTAL				**7876,20**
Financieros	5%			
TOTAL				**8270,01**

Tabla 26.- Presupuesto

Como puede apreciarse en la gráfica siguiente, la mayor parte de los costes provienen de las horas internas, siendo el 79% del total. El 21% restante pertenece a las amortizaciones, y los costes en castos de material son prácticamente despreciables.

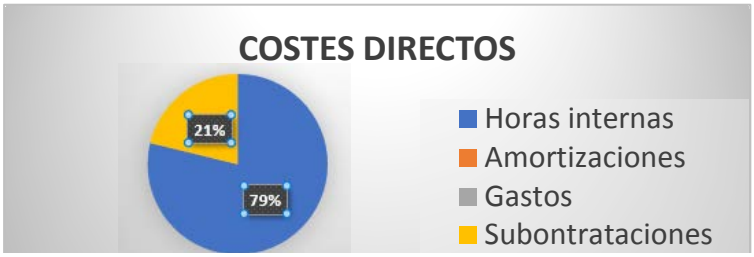

Ilustración 33.- Gráfica Costes Directos

PLANIFICACIÓN Y DIAGRAMA DE GANTT

En este apartado se detallan las tareas a realizar durante el desarrollo del proyecto, así como sus plazos y duraciones.

El proyecto da comienzo el día 30 de enero de 2018 con la primera reunión con el director del TFG, donde se plantea la idea inicial del proyecto, y finaliza el día 23 de julio de ese mismo año con la entrega del proyecto finalizado.

TAREA 1 INGENIERÍA

La primera tarea, y, por tanto, el proyecto, comienza el día 30 de enero con una reunión con el director del TFG donde se plantea la idea del proyecto. Posteriormente, se comienza con la búsqueda de información y análisis de alternativas, en aras de lograr el mejor diseño posible del proyecto, y de realizarlo de la forma más óptima posible. La duración de esta tarea es de 2 meses, hasta el 28 de febrero.

TAREA 2 CONTRATACIONES Y COMPRAS

Una vez recabada la información necesaria y diseñado el proyecto, comienza la búsqueda de las herramientas y programas necesarios para llevarlo a cabo. Esta tarea tiene una duración de 2 semanas, comenzando el 1 de marzo y finalizando el 20.

TAREA 3 TRABAJOS PREVIOS

Tras la obtención de las herramientas necesarias y sus respectivas licencias, se requieren de 3 días para instalarlas y dejarlas listas para las primeras pruebas.

TAREA 4 PRUEBAS Y VALIDACIÓN

Esta tarea abarca toda la programación del código y la validación de los módulos mecatrónicos. Comienza el 23 de marzo con las pruebas de las herramientas instaladas y finaliza el 12 de julio con la comprobación de el correcto funcionamiento del código tanto en la simulación con NX-MCD como en la célula FMS 200.

La célula está compuesta por cuatro estaciones que hay que modularizar, siendo el proceso muy similar en todas ellas. El primer paso es la eliminación de todas las marcas del código, pasando las variables a través de parámetros de DBs. Una vez hecho esto, se han de sustituir los temporizadores de siemens por los del estándar. Y, finalmente, crear

los DBs y FBs necesarios para albergar el código cumpliendo la estructura del programa de generación automática

TAREA 5 REDACCIÓN DEL DOCUMENTO

La última tarea es la redacción del documento. Esta abarca la totalidad del mismo, ya que se va redactando a medida que el proyecto se va desarrollando.

El proyecto se da por finalizado el día 24 de julio del 2018 con la entrega del mismo.

EDT	Nombre de tarea	Duración	Comienzo	Fin
1	**Ingeniería**	**22 días**	**mar 30/01/18**	**mié 28/02/18**
1.1	Propuesta del TFG	1 día	mar 30/01/18	mar 30/01/18
1.2	Búsqueda información	7 días	mié 31/01/18	jue 08/02/18
1.3	Planteamiento TFG	7 días	vie 09/02/18	lun 19/02/18
1.4	Diseño del proyecto	7 días	mar 20/02/18	mié 28/02/18
2	**Contrataciones y compras**	**14 días**	**jue 01/03/18**	**mar 20/03/18**
2.1	**VMware Workstation Player 12**	**7 días**	**jue 01/03/18**	**vie 09/03/18**
2.1.1	Gestión de proveedores y ofertas	5 días	jue 01/03/18	mié 07/03/18
2.1.2	Pedido	1 día	jue 08/03/18	jue 08/03/18
2.1.3	Recepción	1 día	vie 09/03/18	vie 09/03/18
2.2	**DISAW7M Simatic TIA Portal V14**	**7 días**	**lun 12/03/18**	**mar 20/03/18**
2.2.1	Gestión de proveedores y ofertas	5 días	lun 12/03/18	vie 16/03/18
2.2.2	Pedido	1 día	lun 19/03/18	lun 19/03/18
2.2.3	Recepción	1 día	mar 20/03/18	mar 20/03/18
3	**Trabajos previos**	**2 días**	**mié 21/03/18**	**jue 22/03/18**
3.1	Instalación MVware	1 día	mié 21/03/18	mié 21/03/18
3.2	Instaación TIA Portal	1 día	jue 22/03/18	jue 22/03/18
4	**Pruebas y validación del código**	80 días	vie 23/03/18	jue 12/07/18
4.1	**Pruebas programas**	**1 día**	**vie 23/03/18**	**vie 23/03/18**
4.1.1	Prueba TIA Portal	1 día	vie 23/03/18	vie 23/03/18
4.2	**Pruebas de proceso**	**79 días**	**lun 26/03/18**	**jue 12/07/18**
4.2.1	**Generación de código**	79 días	lun 26/03/18	jue 12/07/18
4.2.1.1	**Estación 1**	**47 días**	**lun 26/03/18**	**mar 29/05/18**
4.2.1.1	Eliminacón Marcas	15 días	lun 26/03/18	vie 13/04/18
4.2.1.1	Temporizadores	10 días	lun 16/04/18	vie 27/04/18
4.2.1.1	Generación DBs	15 días	lun 30/04/18	vie 18/05/18
4.2.1.1	Validación	7 días	lun 21/05/18	mar 29/05/18
4.2.1.2	**Estación 4**	**32 días**	**mié 30/05/18**	**jue 12/07/18**
4.2.1.2	Eliminacón Marcas	10 días	mié 30/05/18	mar 12/06/18
4.2.1.2	Temporizadores	5 días	mié 13/06/18	mar 19/06/18
4.2.1.2	Generación DBs	10 días	mié 20/06/18	mar 03/07/18
4.2.1.2	Validación	7 días	mié 04/07/18	jue 12/07/18
5	**Documentación**	**125 días**	**mar 30/01/18**	**lun 23/07/18**
5.1	Redacción	120 días	mar 30/01/18	lun 16/07/18
5.2	Entrega	1 día	lun 23/07/18	lun 23/07/18

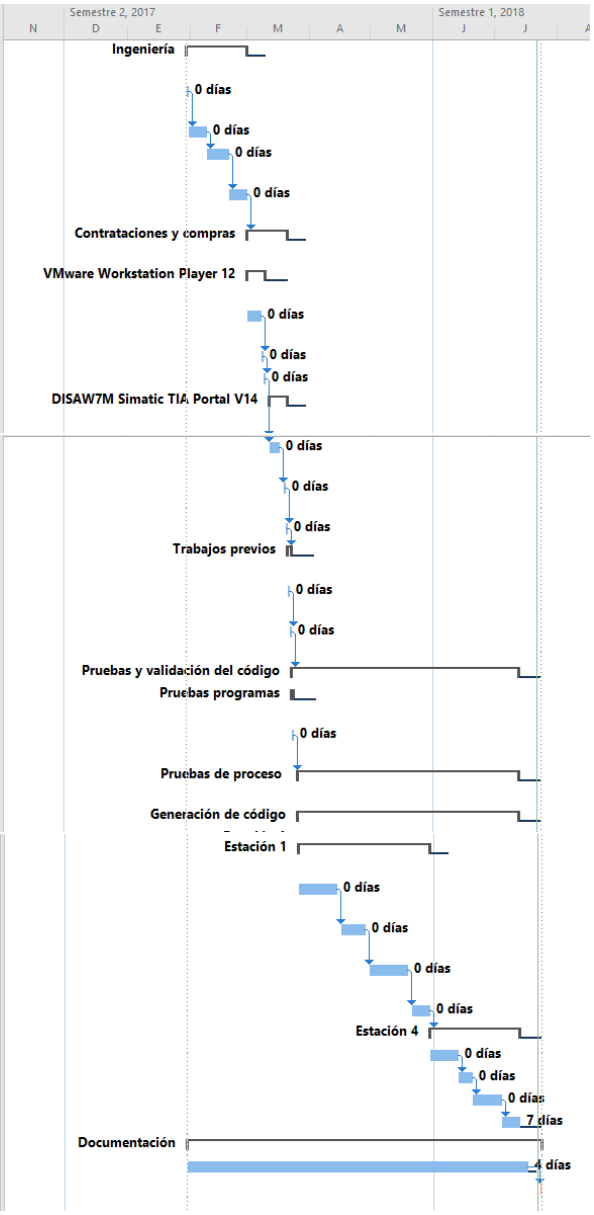

Ilustración 34.- Gantt Planificación

CONCLUSIONES

En este apartado se enumeran las conclusiones a las que se ha llegado tras la realización y análisis del proyecto.

- La modularización aporta grandes ventajas a la hora de automatizar, Simplificando el trabajo y provocando grandes ahorros de tiempo.

- Es la filosofía de automatización que mejor se adapta a las demandas de la sociedad actual.

- El programa de generación de código es una herramienta con gran potencial que simplifica mucho las tareas de automatización.

- La modularización es el presente y futuro de la automatización en las empresas, tomando un papel clave en los objetivos de la Industria 4.0.

LÍNEAS FUTURAS

Tras haberse comprobado el correcto funcionamiento del programa y las ventajas que aporta en la automatización de una célula de montaje, el siguiente paso sería llevarlo a procesos de producción reales. Probarlo en empresas donde realmente se viese su verdadero potencial, así como las ventajas e inconvenientes que tendría en un ámbito de trabajo real.

BIBLIOGRAFÍA

[1] C. Trilnick, << Telar de Jacquard>>. [En línea]. Available: http://proyectoidis.org/telar-de-jacquard/.

[2] «FMS-200 Sistema didáctico Modular de ensamblaje flexible». [En línea]. Available: http://www.smctraining.com/webpage/indexpage/287.

[3] «EVOLUCIÓN DE LA AUTOMATIZACIÓN EN LOS PROCESOS INDUSTRIALES,» [En línea]. Available: https://www.umesl.com/noticias/evolucion-de-la-automatizacion-en-los-procesos-industriales.

[4] J. Gomar, << Cúal fue el primer microprocesador de la historia y quién lo inventó>>, 21 Abril 2018. [En línea]. Available: https://www.profesionalreview.com/2018/04/21/cual-fue-el-primer-microprocesador-de-la-historia-y-quien-lo-invento/

[5] R. Archanco, << ¿Qué es esto de la industria 4.0?>>, 11 Mayo 2016. [En línea]. Available: https://papelesdeinteligencia.com/que-es-industria-4-0/..

[6] D. Méndez, <<Nueva era: Industria 4.0>>, 2 Agosto 2018. [En línea]. Available: https://www.cimaconsulting.cl/single-post/2018/08/02/Nueva-era-Industrias-40?hid=E7781D13979990F96E49C5C6EAFEA3AA&wordfence_logHuman=1.

[7] «EVOLUCIÓN DE LA AUTOMATIZACIÓN EN LOS PROCESOS INDUSTRIALES». [En línea]. Available: https://www.umesl.com/noticias/evolucion-de-la-automatizacion-en-los-procesos-industriales.

[8] X. Pi << Modularidad en la industria de procesos con I4.0>>, 18 Octubre 2017. [En línea] ¡. Available: http://www.infoplc.net/plus-plus/tecnologia/item/104784-modularidad-en-la-industria-de-procesos.

[9] << Historia del PLC, Modicon, Modbus>>. [En línea]. Available: https://unicrom.com/historia-del-plc-modicon-modbus/.

[10] << Controlador lógico programable>>. [En línea]. Available: https://es.wikipedia.org/wiki/Controlador_l%C3%B3gico_programable.

[11] << Industria 4.0>>. [En línea[. Available: https://es.wikipedia.org/wiki/Industria_4.0.